openstax™

Principles of

Data Science

Principles of Data Science

SENIOR CONTRIBUTING AUTHORS
Dr. Shaun V. Ault, Valdosta State University
Dr. Soohyun Nam Liao, University of California San Diego
Larry Musolino, Pennsylvania State University

openstax™

OpenStax
Rice University
6100 Main Street MS-375
Houston, Texas 77005

To learn more about OpenStax, visit https://openstax.org.
Individual print copies and bulk orders can be purchased through our website.

©2025 Rice University. Textbook content produced by OpenStax is licensed under a Creative Commons Attribution Non-Commercial ShareAlike 4.0 International License (CC BY-NC-SA 4.0). Under this license, any user of this textbook or the textbook contents herein can share, remix, and build upon the content for noncommercial purposes only. Any adaptations must be shared under the same type of license. In any case of sharing the original or adapted material, whether in whole or in part, the user must provide proper attribution as follows:

- If you noncommercially redistribute this textbook in a digital format (including but not limited to PDF and HTML), then you must retain on every page the following attribution:
  "Access for free at openstax.org."
- If you noncommercially redistribute this textbook in a print format, then you must include on every physical page the following attribution:
  "Access for free at openstax.org."
- If you noncommercially redistribute part of this textbook, then you must retain in every digital format page view (including but not limited to PDF and HTML) and on every physical printed page the following attribution:
  "Access for free at openstax.org."
- If you use this textbook as a bibliographic reference, please include https://openstax.org/details/books/principles-data-science in your citation.

For questions regarding this licensing, please contact support@openstax.org.

Trademarks
The OpenStax name, OpenStax logo, OpenStax book covers, OpenStax CNX name, OpenStax CNX logo, OpenStax Tutor name, Openstax Tutor logo, Connexions name, Connexions logo, Rice University name, and Rice University logo are not subject to the license and may not be reproduced without the prior and express written consent of Rice University.

Kendall Hunt and the Kendall Hunt Logo are trademarks of Kendall Hunt. The Kendall Hunt mark is registered in the United States, Canada, and the European Union. These trademarks may not be used without the prior and express written consent of Kendall Hunt.

COLOR PAPERBACK BOOK ISBN-13	**979-8-3851-6185-0**
B&W PAPERBACK BOOK ISBN-13	**979-8-3851-6186-7**
DIGITAL VERSION ISBN-13	**978-1-961584-60-0**
ORIGINAL PUBLICATION YEAR	**2025**

1 2 3 4 5 6 7 8 9 10 CJP 25

OpenStax

OpenStax provides free, peer-reviewed, openly licensed textbooks for introductory college and Advanced Placement® courses and low-cost, personalized courseware that helps students learn. A nonprofit ed tech initiative based at Rice University, we're committed to helping students access the tools they need to complete their courses and meet their educational goals.

Rice University

OpenStax is an initiative of Rice University. As a leading research university with a distinctive commitment to undergraduate education, Rice University aspires to path-breaking research, unsurpassed teaching, and contributions to the betterment of our world. It seeks to fulfill this mission by cultivating a diverse community of learning and discovery that produces leaders across the spectrum of human endeavor.

Philanthropic Support

OpenStax is grateful for the generous philanthropic partners who advance our mission to improve educational access and learning for everyone. To see the impact of our supporter community and our most updated list of partners, please visit openstax.org/foundation.

Arnold Ventures
Chan Zuckerberg Initiative
Chegg, Inc.
Arthur and Carlyse Ciocca Charitable Foundation
Digital Promise
Ann and John Doerr
Bill & Melinda Gates Foundation
Girard Foundation
Google Inc.
The William and Flora Hewlett Foundation
The Hewlett-Packard Company
Intel Inc.
Rusty and John Jaggers
The Calvin K. Kazanjian Economics Foundation
Charles Koch Foundation
Leon Lowenstein Foundation, Inc.
The Maxfield Foundation
Burt and Deedee McMurtry
Michelson 20MM Foundation
National Science Foundation
The Open Society Foundations
Jumee Yhu and David E. Park III
Brian D. Patterson USA-International Foundation
The Bill and Stephanie Sick Fund
Steven L. Smith & Diana T. Go
Stand Together
Robin and Sandy Stuart Foundation
The Stuart Family Foundation
Tammy and Guillermo Treviño
Valhalla Charitable Foundation
White Star Education Foundation
Schmidt Futures
William Marsh Rice University

CONTENTS

Preface 1

UNIT 1 INTRODUCING DATA SCIENCE AND DATA COLLECTION

1 What Are Data and Data Science? 9

Introduction 9
1.1 What Is Data Science? 9
1.2 Data Science in Practice 12
1.3 Data and Datasets 16
1.4 Using Technology for Data Science 29
1.5 Data Science with Python 31
Key Terms 53
Group Project 54
Chapter Review 55
Critical Thinking 55
Quantitative Problems 56
References 56

2 Collecting and Preparing Data 59

Introduction 59
2.1 Overview of Data Collection Methods 60
2.2 Survey Design and Implementation 63
2.3 Web Scraping and Social Media Data Collection 68
2.4 Data Cleaning and Preprocessing 78
2.5 Handling Large Datasets 90
Key Terms 96
Group Project 98
Critical Thinking 99
References 103

UNIT 2 ANALYZING DATA USING STATISTICS

3 Descriptive Statistics: Statistical Measurements and Probability Distributions 105

Introduction 105
3.1 Measures of Center 106
3.2 Measures of Variation 112
3.3 Measures of Position 117
3.4 Probability Theory 121
3.5 Discrete and Continuous Probability Distributions 129
Key Terms 142
Group Project 143
Quantitative Problems 144

4 Inferential Statistics and Regression Analysis 147

Introduction 147
4.1 Statistical Inference and Confidence Intervals 148
4.2 Hypothesis Testing 167
4.3 Correlation and Linear Regression Analysis 189
4.4 Analysis of Variance (ANOVA) 205
Key Terms 210
Group Project 212
Quantitative Problems 212

UNIT 3 PREDICTING AND MODELING USING DATA

5 Time Series and Forecasting 215

Introduction 215
5.1 Introduction to Time Series Analysis 215
5.2 Components of Time Series Analysis 224
5.3 Time Series Forecasting Methods 229
5.4 Forecast Evaluation Methods 256
Key Terms 261
Group Project 262
Critical Thinking 263
Quantitative Problems 264

6 Decision-Making Using Machine Learning Basics 269

Introduction 269
6.1 What Is Machine Learning? 270
6.2 Classification Using Machine Learning 278
6.3 Machine Learning in Regression Analysis 297
6.4 Decision Trees 310
6.5 Other Machine Learning Techniques 320
Key Terms 330
Group Project 331
Chapter Review 332
Critical Thinking 332
Quantitative Problems 332
References 334

7 Deep Learning and AI Basics 335

Introduction 335
7.1 Introduction to Neural Networks 336
7.2 Backpropagation 345
7.3 Introduction to Deep Learning 357
7.4 Convolutional Neural Networks 361
7.5 Natural Language Processing 363
Key Terms 374

Group Project 375
Chapter Review 376
Critical Thinking 377
Quantitative Problems 378
References 379

UNIT 4 MAINTAINING A PROFESSIONAL AND ETHICAL DATA SCIENCE PRACTICE

8 Ethics Throughout the Data Science Cycle 381

Introduction 381
8.1 Ethics in Data Collection 382
8.2 Ethics in Data Analysis and Modeling 392
8.3 Ethics in Visualization and Reporting 399
Key Terms 408
Group Project 409
Chapter Review 411
Critical Thinking 414
References 414

9 Visualizing Data 415

Introduction 415
9.1 Encoding Univariate Data 416
9.2 Encoding Data That Change Over Time 430
9.3 Graphing Probability Distributions 435
9.4 Geospatial and Heatmap Data Visualization Using Python 443
9.5 Multivariate and Network Data Visualization Using Python 449
Key Terms 461
Group Project 462
Critical Thinking 462

10 Reporting Results 465

Introduction 465
10.1 Writing an Informative Report 466
10.2 Validating Your Model 473
10.3 Effective Executive Summaries 488
Key Terms 494
Group Project 495
Chapter Review 495
Critical Thinking 497
References 498

A Appendix A: Review of Excel for Data Science 499

B Appendix B: Review of R Studio for Data Science 517

C Appendix C: Review of Python Algorithms 533

D Appendix D: Review of Python Functions 539

Answer Key 553

Index 557

Preface

About OpenStax

OpenStax is part of Rice University, which is a 501(c)(3) nonprofit charitable corporation. As an educational initiative, it's our mission to improve educational access and learning for everyone. Through our partnerships with philanthropic organizations and our alliance with other educational resource companies, we're breaking down the most common barriers to learning. Because we believe that everyone should and can have access to knowledge.

About OpenStax Resources

Customization

Principles of Data Science is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY NC-SA) license, which means that you can non-commercially distribute, remix, and build upon the content, as long as you provide attribution to OpenStax and its content contributors, under the same license.

Because our books are openly licensed, you are free to use the entire book or select only the sections that are most relevant to the needs of your course. Feel free to remix the content by assigning your students certain chapters and sections in your syllabus, in the order that you prefer. You can even provide a direct link in your syllabus to the sections in the web view of your book.

Instructors also have the option of creating a customized version of their OpenStax book. Visit the Instructor Resources section of your book page on OpenStax.org for more information.

Art Attribution

In *Principles of Data Science*, most art contains attribution to its title, creator or rights holder, host platform, and license within the caption. Because the art is openly licensed, non-commercial users or organizations may reuse the art as long as they provide the same attribution to its original source. (Commercial entities should contact OpenStax to discuss reuse rights and permissions.) To maximize readability and content flow, some art does not include attribution in the text. If you reuse art from this text that does not have attribution provided, use the following attribution: Copyright Rice University, OpenStax, under CC BY-NC-SA 4.0 license.

Errata

All OpenStax textbooks undergo a rigorous review process. However, like any professional-grade textbook, errors sometimes occur. In addition, the wide range of topics, data, technologies, and legal circumstances in data science change frequently, and portions of the textbook may become out of date. Since our books are web-based, we can make updates periodically when deemed pedagogically necessary. If you have a correction to suggest, submit it through the link on your book page on OpenStax.org. Subject matter experts review all errata suggestions. OpenStax is committed to remaining transparent about all updates, so you will also find a list of past and pending errata changes on your book page on OpenStax.org.

Format

You can access this textbook for free in web view or PDF through OpenStax.org, and for a low cost in print. The web view is the recommended format because it is the most accessible – including being WCAG 2.2 AA compliant – and most current. Print versions are available for individual purchase, or they may be ordered through your campus bookstore.

About *Principles of Data Science*

Summary

Principles of Data Science is intended as introductory material for a one- or two-semester course on data science. It is appropriate for undergraduate students interested in the rapidly growing field of data science;

this may include data science majors, data science minors, or students concentrating in business, finance, health care, engineering, the sciences, or a number of other fields where data science has become critically important. The material is designed to prepare students for future coursework and career applications in a data science–related field. It does not assume significant prior coding experience, nor does it assume completion of more than college algebra. The text provides foundational statistics instruction for students who may have a limited statistical background.

Coverage and Scope

Principles of Data Science emphasizes the use of Python code in relevant data science applications. Python provides a versatile programming language with libraries and frameworks for data manipulation, analysis, and machine learning. The book begins with an introduction to Python and presents Python libraries, algorithms, and functions as they are needed throughout. In occasional, focused instances, the authors also use Excel to illustrate the basic manipulation of data using functions, formulas, and tools for calculations, visualization, and financial analysis. R, a programming language used most often for statistical modeling, is briefly described and then summarized and applied to relevant examples in a book appendix. Excel and Python summaries are also provided in appendices at the end of the book.

The table of contents (TOC) is divided into ten chapters, organized in four units, intuitively following the standard data science cycle. The four units are:

Unit 1: ***Introducing Data Science and Data Collection***
Unit 2: ***Analyzing Data Using Statistics***
Unit 3: ***Predicting and Modeling Using Data***
Unit 4: ***Maintaining a Professional and Ethical Data Science Practice***

The learning objectives and curriculum of introductory data science courses vary, so this textbook aims to provide broader and more detailed coverage than an average single-semester course. Instructors can choose which chapters or sections they want to include in their particular course.

To enable this flexibility, chapters in this text can be used in a self-contained manner, although most chapters do cross-reference sections and chapters that precede or follow. More importantly, the authors have taken care to build topics gradually, from chapter to chapter, so instructors should bear this in mind when considering alternate sequence coverage.

Unit 1: Introducing Data Science and Data Collection starts off with **Chapter 1**'s explanation of the data science cycle (data collection and preparation, data analysis, and data reporting) and its practical applications in fields such as medicine, engineering, and business. **Chapter 1** also describes various types of datasets and provides the student with basic data summary tools from the Python `pandas` library. **Chapter 2** describes the processes of data collection and cleaning and the challenges of managing large datasets. It previews some of the qualitative ethical considerations that Chapters 7 and 8 later expand on.

Unit 2: Analyzing Data Using Statistics forms a self-contained unit that instructors may assign on a modular, more optional basis, depending on students' prior coursework. **Chapter 3** focuses on measures of center, variation, and position, leading up to probability theory and illustrating how to use Python with binomial and normal distributions. **Chapter 4** goes deeper into statistical analysis, demonstrating how to use Python to calculate confidence intervals, conduct hypothesis tests, and perform correlation and regression analysis.

The three chapters in ***Unit 3: Predicting and Modeling Using Data*** form the core of the book. **Chapter 5** introduces students to the concept and practical applications of time series. Chapter 5 provides focused examples of both Python and Excel techniques useful in forecasting time series, analyzing seasonality, and identifying measures of error. **Chapter 6** starts with distinguishing supervised vs. unsupervised machine learning and then develops some common methods of data classification, including logistic regression, clustering algorithms, and decision trees. Chapter 6 includes Python techniques for more sophisticated

statistical analyses such as regression with bootstrapping and multivariable regression. Finally, Chapter 6 refers back to the topics of data mining and big data introduced in Chapter 2.

Chapter 7 is a pedagogically rich chapter, with a balance of quantitative and qualitative content, covering the role of neural networks in deep learning and applications in large language models. The first four sections discuss the topics of neural networks (standard, recurrent, and convolutional), backpropagation, and deep learning. The real-life application of classifying handwritten numerals is used as an example. The last section dives into the important and rapidly changing technology of natural language processing (NLP), large language models (LLMs), and artificial intelligence (AI). While in-depth coverage of these evolving subjects is beyond the scope of this textbook, the pros/cons, the examples from technical and artistic applications, and the online resources provided in this section all serve as a good starting point for classroom discussion. This topic also naturally segues into the broader professional responsibility discussed in **Chapter 8**.

The final chapters in ***Unit 4: Maintaining a Professional and Ethical Data Science Practice*** help the student apply and adjust the specific techniques learned in the previous chapters to the real-life data analysis, decision-making, and communication situations they will encounter professionally. **Chapter 8** emphasizes the importance of ethical considerations along each part of the cycle: data collection; data preparation, analysis, and modeling; and reporting and visualization. Coverage of the issues in this chapter makes students aware of the subjective and sensitive aspects of privacy and informed consent at every step in the process. At the professional level, students learn more about the evolving standards for the relatively new field of data science, which may differ among industries or between the United States and other countries.

Chapter 9 circles back to some of the statistical concepts introduced in Chapters 3 and 4, with an emphasis on clear visual analysis of data trends. Chapter 9 provides a range of Python techniques to create boxplots and histograms for univariate data; to create line charts and trend curves for time series; to graph binomial, Poisson, and normal distributions; to generate heatmaps from geospatial data; and to create correlation heatmaps from multidimensional data.

Chapter 10 brings the student back to the practical decision-making setting introduced in Chapter 1. Chapter 10 helps the student address how to tailor the data analysis and presentation to the audience and purpose, how to validate the assumptions of a model, and how to write an effective executive summary.

The four **Appendices (A–D)** provide a practical set of references for Excel commands and commands for R statistical software as well as Python commands and algorithms. **Appendix A** uses a baseball dataset from Chapter 1 to illustrate basic Microsoft® Excel® software commands for manipulating, analyzing, summarizing, and graphing data. **Appendix B** provides a brief overview of data analysis with the open-source statistical computing package R (https://openstax.org/r/project), using a stock price example. **Appendix C** lists the approximately 60 Python algorithms used in the textbook, and **Appendix D** lists the code and syntax for the approximately 75 Python functions demonstrated in the textbook. Both **Appendices C and D** are organized in a tabular format, in consecutive chapter order, hyperlinked to the first significant use of each Python algorithm and function. (Instructors may find Appendices C and D especially useful in developing their teaching plan.)

Pedagogical Foundation

Because this is a practical, introductory-level textbook, math equations and code are presented clearly throughout. Particularly in the core chapters, students are introduced to key mathematical concepts, equations, and formulas, often followed by numbered Example Problems that encourage students to apply the concepts just covered in a variety of situations. Technical illustrations and Python code feature boxes build on and supplement the theory. Students are encouraged to try out the Python code from the feature boxes in the Google Colaboratory (https://openstax.org/r/colabresearch) (Colab) platform.

The authors have included a diverse mix of data types and sources for analysis, illustration, and discussion purposes. Some scenarios are fictional and/or internal to standard Python libraries, while other datasets come from external, real-world sources, both corporate and government (such as Federal Reserve Economic Data

(FRED) (https://openstax.org/r/nasdaq1), Statista (https://openstax.org/r/statista1), and Nasdaq (https://openstax.org/r/nasdaq)). Most scenarios are either summarized in an in-line table, or have datasets provided in a downloadable student spreadsheet for import as a .CSV file (Chapter 1 also discusses the .JSON format) and/or with a hyperlink to the external source. Some examples focus on scientific topics (e.g., the "classic" Iris flower dataset, annual temperature changes), while other datasets reflect phenomena with more nuanced socioeconomic issues (gender-based salary differences, cardiac disease markers in patients).

While the book's foundational chapters illustrate practical "techniques and tools," the more process-oriented chapters iteratively build on and emphasize an underlying framework of professional, responsible, and ethical data science practice. Chapter 1 refers the student to several national and international data science organizations that are developing professional standards. Chapter 2 emphasizes avoiding bias in survey and sample design. Chapter 8 discusses relevant privacy legislation. For further class exploration, Chapters 7 and 8 include online resources on mitigating bias and discrimination in machine learning models, including related Python libraries such as HolisticAI (https://openstax.org/r/holistic) and Fairlens (https://openstax.org/r/projectfairlens). Chapter 10 references several executive dashboards that support transparency in government.

The Group Projects at the end of each chapter encourage students to apply the techniques and considerations covered in the book using either datasets already provided or new data sources that they might receive from their instructors or in their own research. For example, project topics include the following: collecting data on animal extinction due to global warming (Chapter 2), predicting future trends in stock market prices (Chapter 5), diagnosing patients for liver disease (Chapter 7), and analyzing the severity of ransomware attacks (Chapter 8).

Key Features

The key in-chapter features, depending on chapter content and topics, may include the following:

- Learning Outcomes (LOs) to guide the student's progress through the chapter
- Example Problems, demonstrating calculations and solutions in-line
- Python code boxes, providing sample input code for and output from Google Colab
- Note boxes providing instructional tips to help with the practical aspects of the math and coding
- Data tables from a variety of social science and industry settings
- Technical charts and heatmaps to visually demonstrate code output and variable relationships
- Exploring Further boxes, with additional resources and online examples to extend learning
- Mathematical formulas and equations
- Links to downloadable spreadsheet containing key datasets referenced in the chapter for easy manipulation of data

End-of-chapter (EOC) elements, depending on chapter content and topics, may include the following:

- Key Terms
- Group Projects
- Chapter Review Questions
- Critical Thinking Questions
- Quantitative Problems

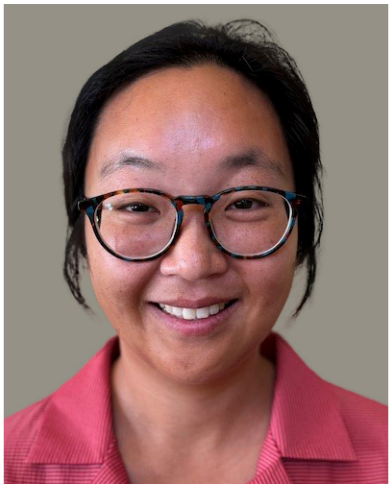
Answers [and Solutions] to Questions in the Book

The student-facing Answer Key at the end of the book provides the correct answer letter and text for Chapter Review questions (multiple-choice). An Instructor Solution Manual (ISM) will be available for verified instructors and downloadable from the restricted OpenStax Instructor Resources web page, with detailed solutions to Quantitative Problems, sample answers for Critical Thinking questions, and a brief explanation of the correct answer for Chapter Review questions. (Sample calculations, tables, code, or figures may be included, as

applicable.) An excerpt of the ISM, consisting of the solutions/sample answers for the odd-numbered questions only, will also be available as a Student Solution Manual (SSM), downloadable from the public OpenStax Student Resources web page. (Answers to the Group Projects are not provided, as they are integrative, exploratory, open-ended assignments.)

About the Authors

Senior Contributing Authors



Senior Contributing Authors (left to right): Shaun V. Ault, Soohyun Nam Liao, Larry Musolino

Dr. Shaun V. Ault, Valdosta State University. Dr. Ault joined the Valdosta State University faculty in 2012, serving as Department Head of Mathematics from 2017 to 2023 and Professor since 2021. He holds a PhD in mathematics from The Ohio State University, a BA in mathematics from Oberlin College, and a Bachelor of Music from the Oberlin Conservatory of Music. He previously taught at Fordham University and The Ohio State University. He is a Certified Institutional Review Board Professional and holds membership in the Mathematical Association of America, American Mathematical Society, and Society for Industrial and Applied Mathematics. He has research interests in algebraic topology and computational mathematics and has published in a number of peer-reviewed journal publications. He has authored two textbooks: *Understanding Topology: A Practical Introduction* (Johns Hopkins University Press, 2018) and, with Charles Kicey, *Counting Lattice Paths Using Fourier Methods. Applied and Numerical Harmonic Analysis*, Springer International (2019).

Dr. Soohyun Nam Liao, University of California San Diego. Dr. Liao joined the UC San Diego faculty in 2015, serving as Assistant Teaching Professor since 2021. She holds PhD and MS degrees in computer science and engineering from UC San Diego and a BS in electronics engineering from Seoul University, South Korea. She previously taught at Princeton University and was an engineer at Qualcomm Inc. She focuses on computer science (CS) education research as a means to support diversity and equity (DEI) in CS programs. Among her recent co-authored papers is, with Yunyi She, Korena S. Klimczak, and Michael E. Levin, "ClearMind Workshop: An ACT-Based Intervention Tailored for Academic Procrastination among Computing Students," *SIGCSE* (1) 2024: 1216-1222. She has received a National Science Foundation grant to develop a toolkit for A14All (data science camps for high school students).

Larry Musolino, Pennsylvania State University. Larry Musolino joined the Penn State, Lehigh Valley, faculty in 2015, serving as Assistant Teaching Professor of Mathematics since 2022. He received an MS in mathematics from Texas A&M University, a MS in statistics from Rochester Institute of Technology (RIT), and MS degrees in computer science and in electrical engineering, both from Lehigh University. He received his BS in electrical engineering from City College of New York (CCNY). He previously was a Distinguished Member of Technical Staff in semiconductor manufacturing at LSI Corporation. He is a member of the Penn State OER (Open Educational Resources) Advisory Group and has authored a calculus open-source textbook. In addition, he co-authored an open-source *Calculus for Engineering* workbook. He has contributed to several OpenStax

textbooks, authoring the statistics chapters in the *Principles of Finance* textbook and editing and revising *Introductory Statistics, 2e,* and *Introductory Business Statistics, 2e*.

The authors wish to express their deep gratitude to Developmental Editor Ann West for her skillful editing and gracious shepherding of this manuscript. The authors also thank Technical Editor Dhawani Shah (PhD Statistics, Gujarat University) for contributing technical reviews of the chapters throughout the content development process.

Contributing Authors
Wisam Bukaita, Lawrence Technological University
Aeron Zentner, Coastline Community College

Reviewers
Wisam Bukaita, Lawrence Technological University
Drew Lazar, Ball State University
J. Hathaway, Brigham Young University-Idaho
Salvatore Morgera, University of South Florida
David H. Olsen, Utah Tech University
Thomas Pfaff, Ithaca College
Jian Yang, University of North Texas
Aeron Zentner, Coastline Community College

Additional Resources

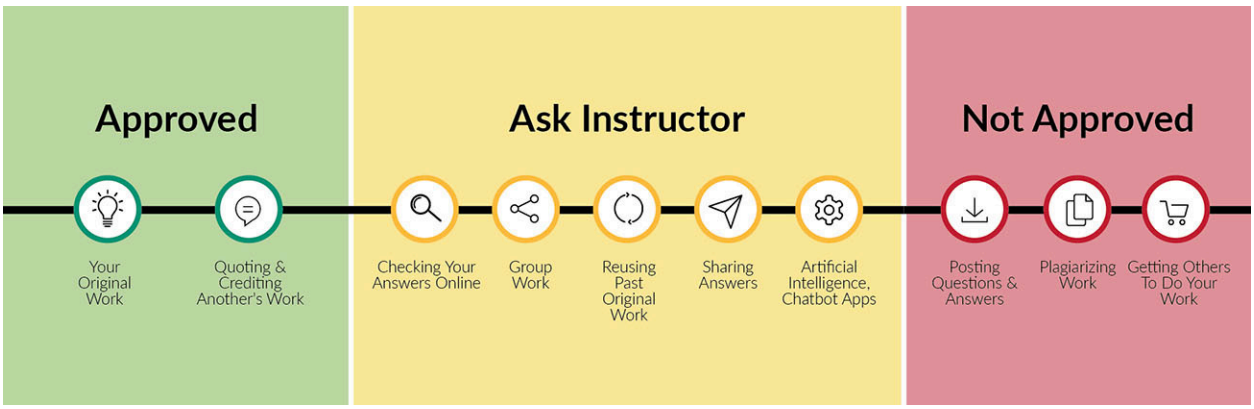
Student and Instructor Resources

We've compiled additional resources for both students and instructors, including Getting Started Guides. Instructor resources require a verified instructor account, which you can apply for when you log in or create your account on OpenStax.org. Take advantage of these resources to supplement your OpenStax book.

Academic Integrity

Academic integrity builds trust, understanding, equity, and genuine learning. While students may encounter significant challenges in their courses and their lives, doing their own work and maintaining a high degree of authenticity will result in meaningful outcomes that will extend far beyond their college career. Faculty, administrators, resource providers, and students should work together to maintain a fair and positive experience.

We realize that students benefit when academic integrity ground rules are established early in the course. To that end, OpenStax has created an interactive to aid with academic integrity discussions in your course.



Visit our academic integrity slider (https://view.genial.ly/61e08a7af6db870d591078c1/interactive-image-defining-academic-integrity-interactive-slider). Click and drag icons along the continuum to align these practices with your institution and course policies. You may then include the graphic on your syllabus, present it in your first course meeting, or create a handout for students. (attribution: Copyright Rice University, OpenStax, under CC BY 4.0 license)

At OpenStax we are also developing resources supporting authentic learning experiences and assessment. Please visit this book's page for updates. For an in-depth review of academic integrity strategies, we highly recommend visiting the International Center of Academic Integrity (ICAI) website at https://academicintegrity.org/ (https://openstax.org/r/academicinte).

Community Hubs

OpenStax partners with the Institute for the Study of Knowledge Management in Education (ISKME) to offer Community Hubs on OER Commons—a platform for instructors to share community-created resources that support OpenStax books, free of charge. Through our Community Hubs, instructors can upload their own materials or download resources to use in their own courses, including additional ancillaries, teaching material, multimedia, and relevant course content. We encourage instructors to join the hubs for the subjects most relevant to your teaching and research as an opportunity both to enrich your courses and to engage with other faculty. To reach the Community Hubs, visit www.oercommons.org/hubs/openstax.

Technology Partners

As allies in making high-quality learning materials accessible, our technology partners offer optional low-cost tools that are integrated with OpenStax books. To access the technology options for your text, visit your book page on OpenStax.org.

1 What Are Data and Data Science?

Figure 1.1 Petroglyphs are one of the earliest types of data generated by humanity, providing vital information about the daily life of the people who created them. (credit: modification of work "Indian petroglyphs (~100 B.C. to ~1540 A.D.) (Newspaper Rock, southeastern Utah, USA) 24" by James St. John/Flickr, CC BY 2.0)

Chapter Outline

1.1 What Is Data Science?
1.2 Data Science in Practice
1.3 Data and Datasets
1.4 Using Technology for Data Science
1.5 Data Science with Python

Introduction

Many of us use the terms "data" and "data science," but not necessarily with a lot of precision. This chapter will define data science terminology and apply the terms in multiple fields. The chapter will also briefly introduce the types of technology (such as statistical software, spreadsheets, and programming languages) that data scientists use to perform their work and will then take a deeper dive into the use of Python for data analysis. The chapter should help you build a technical foundation so that you can practice the more advanced data science concepts covered in future chapters.

1.1 What Is Data Science?

Learning Outcomes

By the end of this section, you should be able to:

- 1.1.1 Describe the goals of data science.
- 1.1.2 Explain the data science cycle and goals of each step in the cycle.
- 1.1.3 Explain the role of data management in the data science process.

Data science is a field of study that investigates how to collect, manage, and analyze data of all types in order to retrieve meaningful information. Although we will describe data in more detail in Data and Datasets, you can consider *data* to be any pieces of evidence or observations that can be analyzed to provide some insights.

In its earliest days, the work of data science was spread across multiple disciplines, including statistics, mathematics, computer science, and social science. It was commonly believed that the job of data collection, management, and analysis would be carried out by different types of experts, with each job independent of one another. To be more specific, *data collection* was considered to be the province of so-called domain experts (e.g., doctors for medical data, psychologists for psychological data, business analysts for sales, logistic, and marketing data, etc.) as they had a full context of the data; *data management* was for computer scientists/engineers as they knew how to store and process data in computing systems (e.g., a single computer, a server, a data warehouse); and *data analysis* was for statisticians and mathematicians as they knew how to derive some meaningful insights from data. Technological advancement brought about the proliferation of data, muddying the boundaries between these jobs, as shown in Figure 1.2. Now, it is expected that a data scientist or data science team will have some expertise in all three domains.

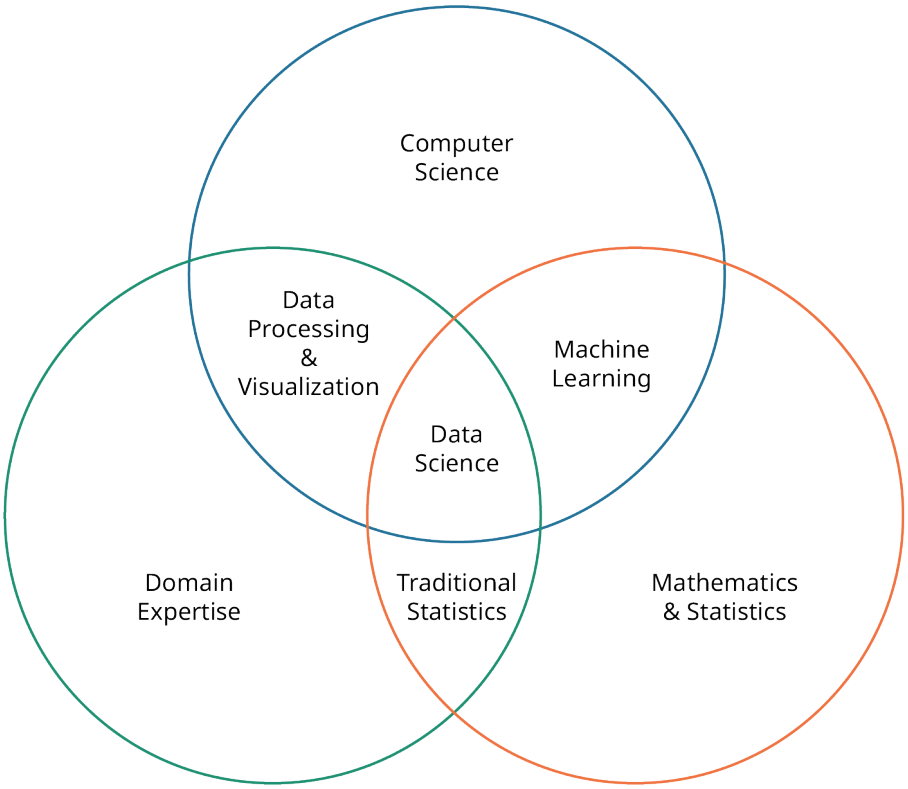


Figure 1.2 The Field of Data Science

One good example of this is the development of personal cell phones. In the past, households typically had only one landline telephone, and the only data that was generated with the telephone was the list of phone numbers called by the household members. Today the majority of consumers own a smartphone, which contains a tremendous amount of data: photos, social media contacts, videos, locations (usually), and perhaps health data (with the consumers' consent), among many other things.

Is the data from a smartphone solely collected by domain experts who are specialized in photos, videos, and such? Probably not. They are automatically logged and collected by the smartphone system itself, which is designed by computer scientists/engineers. For a health care scientist to collect data from many individuals in the "traditional" way, bringing patients into a laboratory and taking vital signs regularly over a period of time takes a lot of time and effort. A smartphone application is a more efficient and productive method, from a data collection perspective.

Data science tasks are often described as a process, and this section provides an overview for each step of that process.

The Data Science Cycle

Data science tasks follow a process, called the **data science cycle**, which includes problem definition, then data collection, preparation, analysis, and reporting, as illustrated in Figure 1.3. See this animation about data science (https://openstax.org/r/youtube), which describes the data science cycle in much the same way.

Figure 1.3 The Data Science Cycle

Although data collection and preparation may sound like simple tasks compared to the more important work of analysis, they are actually the most time- and effort-consuming steps in the data science cycle. According to a survey conducted by Anaconda (2020), data scientists spend about half of the entire process in data collection and cleaning, while data analysis and communication take about a quarter to a third of the time each, depending on the job.

Problem Definition, Data Collection, and Data Preparation

The first step in the data science cycle is a precise definition of the problem statement to establish clear objectives for the goal and scope of the data analysis project. Once the problem has been well defined, the data must be generated and collected. **Data collection** is the systematic process of gathering information on variables of interest. Data is often collected purposefully by domain experts to find answers to predefined problems. One example is data on customer responses to a product satisfaction survey. These survey questions will likely be crafted by the product sales and marketing representatives, who likely have a specific plan for how they wish to use the response data once it is collected.

Not all data is generated this purposefully, though. A lot of data around our daily life is simply a by-product of our activity. These by-products are kept as data because they could be used by others to find some helpful insights later. One example is our web search histories. We use a web search engine like Google to search for information about our interests, and such activity leaves a history of the search text we used on the Google server. Google employees can utilize the records of numerous Google users in order to analyze common search patterns, present accurate search results, and potentially, to display relatable advertisements back to the searchers.

Often the collected data is not in an optimal form for analysis. It needs to be processed somehow so that it can be analyzed, in the phase called **data preparation** or *data processing*. Suppose you work for Google and want to know what kind of food people around the globe search about the most during the night. You have users' search history from around the globe, but you probably cannot use the search history data as is. The search keywords will probably be in different languages, and users live all around the Earth, so nighttime will vary by each user's time zone. Even then, some search keywords would have some typos, simply not make sense, or even remain blank if the Google server somehow failed to store that specific search history record. Note that all these scenarios are possible, and therefore data preparation should address these issues so that the actual analysis can draw more accurate results. There are many different ways to manage these issues, which we will discuss more fully in Collecting and Preparing Data.

Data Analysis

Once the data is collected and prepared, it must be analyzed in order to discover meaningful insights, a process called **data analysis**. There are a variety of data analysis methods to choose from, ranging from simple ones like checking minimum and maximum values, to more advanced ones such as modelling a dependent variable. Most of the time data scientists start with simple methods and then move into more

advanced ones, based on what they want to investigate further. Descriptive Statistics: Statistical Measurements and Probability Distributions and Inferential Statistics and Regression Analysis discuss when and how to use different analysis methods. Time Series and Forecasting and Decision-Making Using Machine Learning Basics discuss forecasting and decision-making.

Data Reporting

Data reporting involves the presentation of data in a way that will best convey the information learned from data analysis. The importance of data reporting cannot be overemphasized. Without it, data scientists cannot effectively communicate to their audience the insights they discovered in the data. Data scientists work with domain experts from different fields, and it is their responsibility to communicate the results of their analysis in a way that those domain experts can understand. **Data visualization** is a graphical way of presenting and reporting data to point out the patterns, trends, and hidden insights; it involves the use of visual elements such as charts, graphs, and maps to present data in a way that is easy to comprehend and analyze. The goal of data visualization is to communicate information effectively and facilitate better decision-making. Data visualization and basic statistical graphing, including how to create graphical presentations of data using Python, are explored in depth in Visualizing Data. Further details on reporting results are discussed in Reporting Results.

Data Management

In the early days of data analysis (when generated data was mostly structured and not quite so “big”), it was possible to keep data in local storage (e.g., on a single computer or a portable hard drive). With this setup, data processing and analysis was all done locally as well.

When so much more data began to be collected—much of it unstructured as well as structured—cloud-based management systems were developed to store all the data on a designated server, outside a local computer. At the same time, data scientists began to see that most of their time was being spent on data processing rather than analysis itself. To address this concern, modern data management systems not only store the data itself but also perform some basic processing on a cloud. These systems, referred to as **data warehousing**, store and manage large volumes of data from various sources in a central location, enabling efficient retrieval and analysis for business intelligence and decision-making. (Data warehousing is covered in more detail in Handling Large Datasets.)

Today, enterprises simply subscribe to a cloud-warehouse service such as Amazon RedShift (which runs on the Amazon Web Services) or Google BigQuery (which runs on the Google Cloud) instead of buying physical storage and configuring data management/processing systems on their own. These services ensure the data is safely stored and processed on the cloud, all without spending money on purchasing/maintaining physical storage.

1.2 Data Science in Practice

Learning Outcomes

By the end of this section, you should be able to:

- 1.2.1 Explain the interdisciplinary nature of data science in various fields.
- 1.2.2 Identify examples of data science applications in various fields.
- 1.2.3 Identify current issues and challenges in the field of data science

While data science has adopted techniques and theories from fields such as math, statistics, and computer science, its applications concern an expanding number of fields. In this section we introduce some examples of how data science is used in business and finance, public policy, health care and medicine, engineering and sciences, and sports and entertainment.

Data Science in Business

Data science plays a key role in many business operations. A variety of data related to customers, products, and sales can be collected and generated within a business. These include customer names and lists of products they have purchased as well as daily revenue. Business analytics investigate these data to launch new products and to maximize the business revenue/profit.

Retail giant Walmart is known for using business analytics to improve the company's bottom line. Walmart collects multiple *petabytes* (1 petabyte = 1,024 terabytes) of unstructured data every hour from millions of customers (commonly referred to as "big data"); as of 2024, Walmart's customer data included roughly 255 million weekly customer visits (Statista, 2024). Walmart uses this big data to investigate consumer patterns and adjust its inventories. Such analysis helps the company avoid overstocking or understocking and resulted in an estimated online sales increase of between 10% and 15%, translating to an extra $1 billion in revenue (ProjectPro, 2015). One often-reported example includes the predictive technology Walmart used to prepare for Hurricane Frances in 2004. A week before the hurricane's arrival, staff were asked to look back at their data on sales during Hurricane Charley, which hit several weeks earlier, and then come up with some forecasts about product demand ahead of Frances (Hays, 2004). Among other insights, the executives discovered that strawberry Pop-Tart sales increased by about sevenfold during that time. As a result, in the days before Hurricane Frances, Walmart shipped extra supplies of strawberry Pop-Tarts to stores in the storm's path (Hays, 2004). The analysis also provided insights on how many checkout associates to assign at different times of the day, where to place popular products, and many other sales and product details. In addition, the company has launched social media analytics efforts to investigate hot keywords on social media and promptly make related products available (ProjectPro, 2015).

Amazon provides another good example. Ever since it launched its Prime membership service, Amazon has focused on how to minimize delivery time and cost. Like Walmart, it started by analyzing consumer patterns and was able to place products close to customers. To do so, Amazon first divided the United States into eight geographic regions and ensured that most items were warehoused and shipped within the same region; this allowed the company to reduce the shipping time and cost. As of 2023, more than 76% of orders were shipped from within the customer's region, and items in same-day shipping facilities could be made ready to put on a delivery truck in just 11 minutes (Herrington, 2023). Amazon also utilizes machine learning algorithms to predict the demand for items in each region and have the highest-demand items available in advance at the fulfillment center of the corresponding region. This predictive strategy has helped Amazon reduce the delivery time for each product and extend the item selections for two-day shipping (Herrington, 2023).

Data science is utilized extensively in finance as well. Detecting and managing fraudulent transactions is now done by automated machine learning algorithms (IABAC, 2023). Based on the customer data and the patterns of past fraudulent activities, an algorithm can determine whether a transaction is fraudulent in real time. Multiple tech companies, such as IBM and Amazon Web Services, offer their own fraud detection solutions to their corporate clients. (For more information, see this online resource on Fraud Detection through Data Analytics (https://openstax.org/r/iabac).)

Data Science in Engineering and Science

Various fields of engineering and science also benefit from data science. Internet of Things (IoT) is a good example of a new technology paradigm that has benefited from data science. **Internet of Things (IoT)** describes the network of multiple objects interacting with each other through the Internet. Data science plays a crucial role in these interactions since behaviors of the objects in a network are often triggered by data collected by another object in the network. For example, a smart doorbell or camera allows us to see a live stream on our phone and alerts us to any unusual activity.

In addition, weather forecasting has always been a data-driven task. Weather analysts collect different measures of the weather such as temperature and humidity and then make their best estimate for the

weather in the future. Data science has made weather forecasting more reliable by adopting more sophisticated prediction methods such as time-series analysis, artificial intelligence (AI), and machine learning (covered in Time Series and Forecasting, Decision-Making Using Machine Learning Basics, and Deep Learning and AI Basics). Such advancement in weather forecasting has also enabled engineers and scientists to predict some natural disasters such as flood or wildfire and has enabled precision farming, with which agricultural engineers can identify an optimal time window to plant, water, and harvest crops. For example, an agronomy consultant, Ag Automation, has partnered with Hitachi to offer a solution that both automates data collection and remotely monitors and controls irrigation for the best efficiency (Hitachi, 2023).

EXPLORING FURTHER

Using AI for Irrigation Control

See this Ag Automation video (https://openstax.org/r/youtube128) demonstrating the use of data collection for controlling irrigation.

Data Science in Public Policy

Smart cities are among the most representative examples of the use of data science in public policy. Multiple cities around the world, including Masdar City in the United Arab Emirates and Songdo in South Korea, have installed thousands of data-collecting sensors used to optimize their energy consumption. The technology is not perfect, and smart cities may not yet live up to their full potential, but many corporations and companies are pursuing the goal of developing smart cities more broadly (Clewlow, 2024). The notion of a smart city has also been applied on a smaller scale, such as to a parking lot, a building, or a street of lights. For example, the city of San Diego installed thousands of sensors on the city streets to control the streetlights using data and smart technology. The sensors measure traffic, parking occupancy, humidity, and temperature and are able to turn on the lights only when necessary (Van Bocxlaer, 2020). New York City has adopted smart garbage bins that monitor the amount of garbage in a bin, allowing garbage collection companies to route their collection efforts more efficiently (Van Bocxlaer, 2020).

EXPLORING FURTHER

Sensor Networks to Monitor Energy Consumption

See how Songdo (https://openstax.org/r/youtube4) monitors energy consumption and safety with sensor networks.

Data Science in Education

Data science also influences education. Traditional instruction, especially in higher education, has been provided in a one-size-fits-all form, such as many students listening to a single instructor's lecture in a classroom. This makes it difficult for an instructor to keep track of each individual student's learning progress. However, many educational platforms these days are online and can produce an enormous amount of student data, allowing instructors to investigate everyone's learning based on these collected data. For example, online learning management systems such as Canvas (https://openstax.org/r/instructure) compile a grade book in one place, and online textbooks such as zyBooks (https://openstax.org/r/zybooks) collect students' mastery level on each topic through their performance on exercises. All these data can be used to capture each student's progress and offer personalized learning experiences such as intelligent tutoring systems or adaptive learning. ALEKS (https://openstax.org/r/aleks), an online adaptive learning application, offers personalized material for each learner based on their past performance.

Data Science in Health Care and Medicine

The fields of health care and medicine also use data science. Often their goal is to offer more accurate diagnosis and treatment using **predictive analytics**—statistical techniques, algorithms, and machine learning that analyze historical data and make predictions about future events. Medical diagnosis and prescription practices have traditionally been based on a patient's verbal description of symptoms and a doctor's or a group of doctors' experience and intuition; this new movement allows health care professionals to make decisions that are more data-driven. Data-driven decisions became more feasible thanks to all the personal data collected through personal gadgets—smartphones, smart watches, and smart bands. Such devices collect daily health/activity records, and this in turn helps health care professionals better capture each patient's situation. All this work will enable patients to receive more accurate diagnoses along with more personalized treatment regimens in the long run.

The Precision Medicine Initiative (https://openstax.org/r/obamawhitehouse) is a long-term research endeavor carried out by the National Institutes of Health (NIH) and other research centers, with the goal of better understanding how a person's genetics, environment, and lifestyle can help determine the best approach to prevent or treat disease. The initiative aims to look at as much data as possible to care for a patient's health more proactively. For example, the initiative includes genome sequencing to look for certain mutations that indicate a higher risk of cancer or other diseases.

Another application of data science in health focuses on lowering the cost of health care services. Using historical records of patients' symptoms and prescription, a chatbot that is powered by artificial intelligence can provide automated health care advice. This will reduce the need for patients to see a pharmacist or doctor, which in turn improves health care accessibility for those who are in greater need.

EXPLORING FURTHER

Big Data In Health Care

The 2015 launch of the National Institutes of Health Precision Medicine Initiative was documented in One Woman's Quest to Cure Her Fatal Brain Disease (https://openstax.org/r/youtubevk). "Promise of Precision Medicine" signaled a new approach to health care in the United States—one heavily reliant on big data.

Data Science in Sports and Entertainment

Data science is prevalent in the sports and the entertainment industry as well. Sports naturally produce much data—about the player, positions, teams, seasons, and so on. Therefore, just as there is the concept of business analytics, the analysis of such data in sports is called **sports analytics**. For example, the Oakland Athletics baseball team famously analyzed player recruitment for the 2002 season. The team's management adapted a statistical approach referred to as **sabermetrics** to recruit and position players. With sabermetrics, the team was able to identify critical yet traditionally overlooked metrics such as on-base percentage and slugging percentage. The team, with its small budget compared to other teams, recruited a number of undervalued players with strong scores on these metrics, and in the process, they became one of the most exciting teams in baseball that year, breaking the American League record for 20 wins in a row. Does this story sound familiar? This story was so dramatic that Michael Lewis wrote a book about it, which was also made into a movie: *Moneyball*.

EXPLORING FURTHER

The Sabermetrics YouTube Channel

Sabermetrics is so popular that there is a YouTube channel devoted to it: Simple Sabermetrics (https://openstax.org/r/simplesabermetrics), with baseball animations and tutorials explaining how data impacts the way today's game is played behind the scenes.

In the entertainment industry, data science is commonly used to make data-driven, personalized suggestions that satisfy consumers known as **recommendation systems**. One example of a recommendation system is on video streaming services such as Netflix. Netflix Research considers subscribers' watch histories, satisfaction with the content, and interaction records (e.g., search history). Their goal is to make perfect personalized recommendations despite some challenges, including the fact that subscribers themselves often not do not know what they want to see.

EXPLORING FURTHER

Careers in Data Science

As you advance in your data science training, consider the many professional options in this evolving field. This helpful graphic from edX (https://openstax.org/r/edx) distinguishes data analyst vs. data science paths. This Coursera article (https://openstax.org/r/coursera) lists typical skill sets by role. Current practitioner discussions are available on forums such as Reddit's r/data science (https://openstax.org/r/reddit).

Trends and Issues in Data Science

Technology has made it possible to collect abundant amounts of data, which has led to challenges in the processing and analyzing of that data. But technology comes to the rescue again! Data scientists now use machine learning to better understand the data, and artificial intelligence can make an automated, data-driven decision on a task. Decision-Making Using Machine Learning Basics and Deep Learning and AI Basics will cover the details of machine learning and artificial intelligence.

With these advances, many people have raised concerns about ethics and privacy. Who is allowed to collect these data, and who has access to them? None of us want someone else to use our personal data (e.g., contact information, health records, location, photos, web search history) without our consent or without knowing the risk of sharing our data. Machine learning algorithms and artificial intelligence are trained to make a decision based on the past data, and when the past data itself inherits some bias, the trained machine learning algorithms and artificial intelligence will make biased decisions as well. Thus, carefully attending to the process of collecting data and evaluating the bias of a trained results is critical. Ethics Throughout the Data Science Cycle will discuss these and other ethical concerns and privacy issues in more depth.

1.3 Data and Datasets

Learning Outcomes

By the end of this section, you should be able to:

- 1.3.1 Define data and dataset.
- 1.3.2 Differentiate among the various data types used in data science.
- 1.3.3 Identify the type of data used in a dataset.
- 1.3.4 Discuss an item and attribute of a dataset.
- 1.3.5 Identify the different data formats and structures used in data science.

What Is Data Science? and Data Science in Practice introduced the many varieties of and uses for data science in today's world. Data science allows us to extract insights and knowledge from data, driving decision-making and innovation in business, health care, entertainment, and so on. As we've seen, the field has roots in math, statistics, and computer science, but it only began to emerge as its own distinct field in the early 2000s with the proliferation of digital data and advances in computing power and technology. It gained significant momentum and recognition around the mid to late 2000s with the rise of big data and the need for sophisticated techniques to analyze and derive insights from large and complex datasets. Its evolution since then has been rapid, and as we can see from the previous discussion, it is quickly becoming a cornerstone of many industries and domains.

Data, however, is not new! Humans have been collecting data and generating datasets from the beginning of time. This started in the Stone Age when people carved some shapes and pictures, called petroglyphs, on rock. The petroglyphs provide insights on how animals looked and how they carried out their daily life, which is valuable "data" for us. Ancient Egyptians invented a first form of paper—papyrus—in order to journal *their* data. Papyrus also made it easier to store data in bulk, such as listing inventories, noting financial transactions, and recording a story for future generations.

Data

"Data" is the plural of the Latin word "datum," which translates as something that is given or used and is often used to mean a single piece of information or a single point of reference in a dataset. When you hear the word "data," you may think of some sort of "numbers." It is true that numbers are usually considered data, but there are many other forms of data all around us. Anything that we can analyze to compile **information**—high-level insights—is considered **data**.

Suppose you are debating whether to take a certain course next semester. What process do you go through in order to make your decision? First, you might check the course evaluations, as shown in Table 1.1.

Semester	Instructor	Class Size	Rating
Fall 2020	A	100	Not recommended at all
Spring 2021	A	50	Highly recommended
Fall 2021	B	120	Not quite recommended
Spring 2022	B	40	Highly recommended
Fall 2022	A	110	Recommended
Spring 2023	B	50	Highly recommended

Table 1.1 Course Evaluation Records

The evaluation record consists of four kinds of data, and they are grouped as columns of the table: Semester, Instructor, Class Size, and Rating. Within each column there are six different pieces of data, located at each row. For example, there are six pieces of text data under the Semester column: "Fall 2020," "Spring 2021," "Fall 2021," "Spring 2022," "Fall 2022," and "Spring 2023."

The course evaluation ratings themselves do not provide an idea on whether to take the course next semester. The ratings are just a *phrase* (e.g., "Highly recommended" or "Not quite recommended") that *encodes* how

recommended the course was in that semester. You need to analyze them to come up with a decision!

Now let's think about how to derive the information from these ratings. You would probably look at all the data, including when in the semester the course was offered, who the instructor is, and class size. These records would allow you to derive information that would help you decide "whether or not to take the course next semester."

EXAMPLE 1.1

Problem

Suppose you want to decide whether or not to put on a jacket today. You research the highest temperatures in the past five days and determine whether you needed a jacket on each day. In this scenario, what data are you using? And what information are you trying to derive?

Solution

Temperature readings and whether you needed a jacket on each of the past five days are two kinds of data you are referring to. Again, they do not indicate anything related to wearing a jacket today. They are simply five pairs of numbers (temperature) and yes/no (whether you needed a jacket) records, with each pair representing a day. Using these data, you are deriving information that you can analyze to help you decide whether to wear a jacket today.

Types of Data

The previous sections talked about how much our daily life is surrounded by data, how much our daily life itself produces new data, and how often we make data-driven decisions without even noticing it. You might have noticed that data comes in various types. Some data are quantitative, which means that they are measured and expressed using numbers. **Quantitative data** deals with quantities and amounts and is usually analyzed using statistical methods. Examples include numerical measurements like height, weight, temperature, heart rate, and sales figures. **Qualitative data** are non-numerical data that generally describe subjective attributes or characteristics and are analyzed using methods such as thematic analysis or content analysis. Examples include descriptions, observations, interviews, and open-ended survey responses (as we'll see in Survey Design and Implementation) that address unquantifiable details (e.g., photos, posts on Reddit). The data types often dictate methods for data analysis, so it is important to be able to identify a type of data. Thus, this section will take a further dive into types of data.

Let's revisit our previous example about deciding whether to take a certain course next semester. In that example, we referred to four pieces of data. They are encoded in different types such as numbers, words, and symbols.

1. The semester the course was offered—Fall 2020, Spring 2021, ..., Fall 2022, Spring 2023
2. The instructor—A and B
3. The class size—100, 50, 120, 40, 110, 50
4. The course rating—"Not recommended at all," ..., "Highly recommended"

There are two primary types of quantitative data—*numeric* and *categorical*—and each of these can be divided into a few subtypes. **Numeric data** is represented in numbers that indicate measurable quantities. It may be followed by some symbols to indicate units. Numeric data is further divided into *continuous data* and *discrete data*. With **continuous data**, the values can be any number. In other words, a value is chosen from an infinite set of numbers. With **discrete data**, the values follow a specific precision, which makes the set of possible values finite.

From the previous example, the class size 100, 150, etc. are numbers with the implied unit "students." Also,

they indicate measurable quantities as they are head counts. Therefore, the class size is *numeric data*. It is also *continuous data* since the size numbers seem to be any natural numbers and these numbers are chosen from an infinite set of numbers, the set of natural numbers. Note that whether data is continuous (or discrete) also depends on the context. For example, the same class size data can be discrete if the campus enforces all classes to be 200 seats or less. Such restriction makes the class size values be chosen from a finite set of 200 numbers: 1, 2, 3, ..., 198, 199, 200.

Categorical data is represented in different forms such as words, symbols, and even numbers. A categorical value is chosen from a finite set of values, and the value does not necessarily indicate a measurable quantity. Categorical data can be divided into *nominal data* and *ordinal data*. For **nominal data**, the set of possible values does not include any ordering notion, whereas with **ordinal data**, the set of possible values includes an ordering notion.

The rest—semester, instructor, and ratings—are *categorical data*. They are represented in symbols (e.g., "Fall 2020," "A") or words (e.g., "Highly recommended"), and these values are chosen from the finite set of those symbols and words (e.g., A vs. B). The former two data are nominal since the semester and instructor do not have orders to follow, while the latter is ordinal since there is a notion of degree (Not recommended at all ~ Highly recommended). You may argue that the semester could have *chronological* ordering: Fall 2020 comes before Spring 2021, Fall 2021 follows Fall 2020. If you want to value that notion for your analysis, you could consider the semester data to be ordinal as well—the chronological ordering is indeed critical when you are looking at a time-series dataset. You will learn more about that in Time Series and Forecasting.

EXAMPLE 1.2

Problem

Consider the jacket scenario in Example 1.1. In that example, we referred to two kinds of data:

1. The temperature during past three days—90°F, 85°F, ...
2. On each of those days, whether you needed a jacket—Yes, No, ...

What is the type of each data?

Solution

The temperatures are numbers followed by the unit degrees Fahrenheit (°F). Also, they indicate measurable quantities as they are specific readings from a thermometer. Therefore, the temperature is numeric data. They are also continuous data since they can be any real number, and the set of real numbers is infinite.

The other type of data—whether or not you needed a jacket—is categorical data. Categorical data are represented in symbols (Yes/No), and the values are chosen from the finite set of those symbols. They are also nominal since Yes/No does not have ordering to follow.

Datasets

A **dataset** is a collection of observations or data entities organized for analysis and interpretation, as shown in Table 1.1. Many datasets can be represented as a table where each row indicates a unique data entity and each column defines the structure of the entities.

Notice that the dataset we used in Table 1.1 has six *entities* (also referred to as **items**, *entries*, or *instances*), distinguished by semester. Each entity is defined by a combination of four **attributes** or characteristics (also known as *features* or *variables*)—Semester, Instructor, Class Size, and Rating. A combination of features characterizes an entry of a dataset.

Although the actual values of the attributes are different across entities, note that all entities have values for

the same four attributes, which makes them a **structured dataset**. As a structured dataset, these items can be listed as a table where each item is listed along the rows of the table.

By contrast, an **unstructured dataset** is one that lacks a predefined or organized data model. While structured datasets are organized in a tabular format with clearly defined fields and relationships, unstructured data lacks a fixed schema. Unstructured data is often in the form of text, images, videos, audio recordings, or other content where the information doesn't fit neatly into rows and columns.

There are plenty of unstructured datasets. Indeed, some people argue there are more unstructured datasets than structured ones. A few examples include Amazon reviews on a set of products, Twitter posts last year, public images on Instagram, popular short videos on TikTok, etc. These unstructured datasets are often processed into a structured one so that data scientists can analyze the data. We'll discuss different data processing techniques in Collecting and Preparing Data.

EXAMPLE 1.3

Problem

Let's revisit the jacket example: deciding whether to wear a jacket to class. Suppose the dataset looks as provided in Table 1.2:

Date	Temperature	Needed a Jacket?
Oct. 10	80°F	No
Oct. 11	60°F	Yes
Oct. 12	65°F	Yes
Oct. 13	75°F	No

Table 1.2 Jacket Dataset

Is this dataset structured or unstructured?

Solution

It is a structured dataset since 1) every individual item is in the same structure with the same three attributes—Date, Temperature, and Needed a Jacket—and 2) each value strictly fits into a cell of a table.

EXAMPLE 1.4

Problem

How many entries and attributes does the dataset in the previous example have?

Solution

The dataset has four entries, each of which is identified with a specific date (Oct. 10, Oct. 11, Oct. 12, Oct. 13). The dataset has three attributes—Date, Temperature, Needed a Jacket.

EXAMPLE 1.5

Problem

A dataset has a list of keywords that were searched on a web search engine in the past week. Is this dataset structured or unstructured?

Solution

The dataset is an unstructured dataset since each entry in the dataset can be a freeform text: a single word, multiple words, or even multiple sentences.

EXAMPLE 1.6

Problem

The dataset from the previous example is processed so that now each search record is summarized as up to three words, along with the timestamp (i.e., when the search occurred). Is this dataset structured or unstructured?

Solution

It is a structured dataset since every entry of this dataset is in the same structure with two attributes: a short keyword along with the timestamp.

Dataset Formats and Structures (CSV, JSON, XML)

Datasets can be stored in different formats, and it's important to be able to recognize the most commonly used formats. This section covers three of the most often used formats for structured datasets—**comma-separated values (CSV)**, **JavaScript Object Notation (JSON)**, and **Extensible Markup Language (XML)**. While CSV is the most intuitive way of encoding a tabular dataset, much of the data we would collect from the web (e.g., websites, mobile applications) is stored in the JSON or XML format. The reason is that JSON is the most suitable for exchanging data between a user and a server, and XML is the most suitable for complex dataset due to its hierarchy-friendly nature. Since they all store data as plain texts, you can open them using any typical text editors such as Notepad, Visual Studio Code, Sublime Text, or VI editor.

Table 1.3 summarizes the advantages and disadvantages of CSV, JSON, and XML dataset formats. Each of these is described in more detail below.

Dataset Format	Pros	Cons	Typical Use
CSV	• Simple	• Difficult to add metadata • Difficult to parse if there are special characters • Flat structure	• Tabular data
JSON	• Simple • Compatible with many languages • Easy to parse	• Difficult to add metadata • Cannot leave comments	• Data that needs to be exchanged between a user and a server
XML	• Structured (so more readable) • Possible to add metadata	• Verbose • Complex structure with tags	• Hierarchical data structures

Table 1.3 Summary of the CSV, JSON, and XML formats

EXPLORING FURTHER

Popular and Reliable Databases to Search for Public Datasets

Multiple online databases offer public datasets for free. When you want to look for a dataset of interest, the following sources can be your initial go-to.

Government data sources include:

Data.gov (https://openstax.org/r/datagov)

Bureau of Labor Statistics (BLS) (https://openstax.org/r/blsgov1)

National Oceanic and Atmospheric Administration (NOAA) (https://openstax.org/r/noaagov)

World Health Organization (WHO) (https://openstax.org/r/who)

Some reputable nongovernment data sources are:

Kaggle (https://openstax.org/r/kaggle1)

Statista (https://openstax.org/r/statista)

Pew Research Center (https://openstax.org/r/pewresearch)

Comma-Separated Values (CSV)

The CSV stores each item in the dataset in a single line. Variable values for each item are listed all in one line, separated by commas (“,”). The previous example about signing up for a course can be stored as a CSV file. Figure 1.4 shows how the dataset looks when opened with a *text editor* (e.g., Notepad, TextEdit, MS Word, Google Doc) or programming software in the form of a *code editor* (e.g., Sublime Text, Visual Studio Code, XCode). Notice that commas are used to separate the attribute values within a single line (see Figure 1.4).

```
Semester,Instructor,Class Size,Rating
Fall 2020,A,100,Not recommended at all
Spring 2021,A,50,Highly recommended
Fall 2021,B,120,Not quite recommended
Spring 2022,B,40,Highly recommended
Fall 2022,A,110,Recommended
Spring 2023,B,50,Highly recommended
```

Figure 1.4 ch1-courseEvaluations.csv Opened with Visual Studio Code

COMMA FOR NEW LINE?

There is some flexibility on how to end a line with CSV files. It is acceptable to end with or without commas, as some software or programming languages automatically add a comma when generating a CSV dataset.

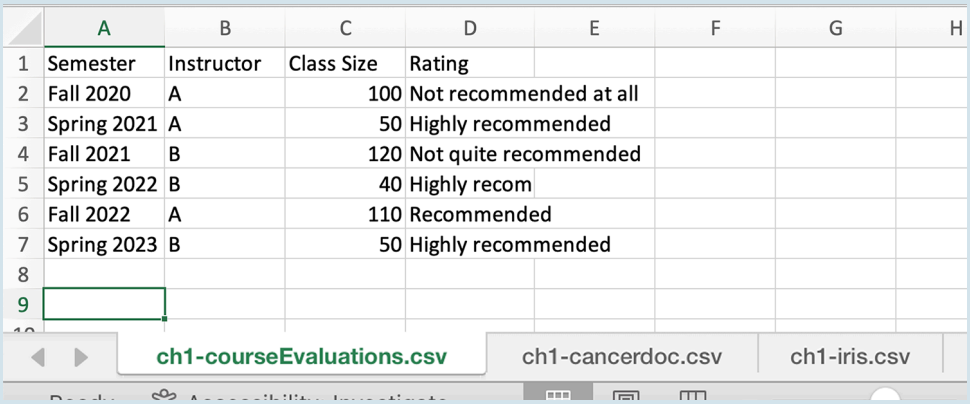
CSV files can be opened with spreadsheet software such as MS Excel and Google Sheets. The spreadsheet software visualizes CSV files more intuitively in the form of a table (see Figure 1.5). We will cover the basic use of Python for analyzing CSV files in Data Science with Python.

	A	B	C	D
1	Semester	Instructor	Class Size	Rating
2	Fall 2020	A	100	Not recommended at all
3	Spring 2021	A	50	Highly recommended
4	Fall 2021	B	120	Not quite recommended
5	Spring 2022	B	40	Highly recommended
6	Fall 2022	A	110	Recommended
7	Spring 2023	B	50	Highly recommended
8				

Figure 1.5 ch1-courseEvaluations.csv Opened with Microsoft Excel (Used with permission from Microsoft)

HOW TO DOWNLOAD AND OPEN A DATASET FROM THE CH1-DATA SPREADSHEET IN THIS TEXT

A spreadsheet file accompanies each chapter of this textbook. The files include multiple tabs corresponding to a single dataset in the chapter. For example, the spreadsheet file for this chapter (https://openstax.org/r/spreadsheet4) is shown in Figure 1.6. Notice that it includes multiple tabs with the names "ch1-courseEvaluations.csv," "ch1-cancerdoc.csv," and "ch1-riris.csv," which are the names of each dataset.



	A	B	C	D	E	F	G	H
1	Semester	Instructor	Class Size	Rating				
2	Fall 2020	A	100	Not recommended at all				
3	Spring 2021	A	50	Highly recommended				
4	Fall 2021	B	120	Not quite recommended				
5	Spring 2022	B	40	Highly recom				
6	Fall 2022	A	110	Recommended				
7	Spring 2023	B	50	Highly recommended				
8								
9								

ch1-courseEvaluations.csv ch1-cancerdoc.csv ch1-iris.csv

Figure 1.6 The dataset spreadsheet file for Chapter 1 (Used with permission from Microsoft)

To save each dataset as a separate CSV file, choose the tab of your interest and select File > Save As ... > CSV File Format. This will only save the current tab as a CSV file. Make sure the file name is set correctly; it may have used the name of the spreadsheet file—"ch1-data.xlsx" in this case. You should name the generated CSV file as the name of the corresponding tab. For example, if you have generated a CSV file for the first tab of "ch1-data.xlsx," make sure the generated file name is "ch1-courseEvaluations.csv." This will prevent future confusion when following instructions in this textbook.

JavaScript Object Notation (JSON)

JSON uses the syntax of a programming language named JavaScript. Specifically, it follows JavaScript's object syntax. Don't worry, though! You do not need to know JavaScript to understand the JSON format.

Figure 1.7 provides an example of the JSON representation of the same dataset depicted in Figure 1.6.

```
1 {"Semester":"Fall 2020","Instructor":"A","Class Size":100,"Rating":"Not recommended at all"}
2 {"Semester":"Spring 2021","Instructor":"A","Class Size":50,"Rating":"Highly recommended"}
3 {"Semester":"Fall 2021","Instructor":"B","Class Size":120,"Rating":"Not quite recommended"}
4 {"Semester":"Spring 2022","Instructor":"B","Class Size":40,"Rating":"Highly recommended"}
5 {"Semester":"Fall 2022","Instructor":"A","Class Size":110,"Rating":"Recommended"}
6 {"Semester":"Spring 2023","Instructor":"B","Class Size":50,"Rating":"Highly recommended"}
```

Figure 1.7 CourseEvaluations.json Opened with Visual Studio Code

Notice that the JSON format starts and ends with a pair of curly braces ({}). Inside, there are multiple pairs of two fields that are separated by a colon (:). These two fields that are placed on the left and right of the colon are called a key and value, respectively,—key `:` `value`. For example, the dataset in Figure 1.7 has five pairs of key-values with the key `"Semester": "Fall 2020"`, `"Semester": "Spring 2021"`, `"Semester": "Fall 2021"`, `"Semester": "Spring 2022"`, `"Semester": "Fall 2022"`, and `"Semester": "Spring 2023"`.

CourseEvaluations.json (https://openstax.org/r/filed1v) has one key-value pair at the highest level: `"Members": [...]`. You can see that each item of the dataset is listed in the form of an *array* or list under the key `"Members"`. Inside the array, each item is also bound by curly braces and has a list of key-value pairs, separated by commas. Keys are used to describe attributes in the dataset, and values are used to define the corresponding values. For example, the first item in the JSON dataset above has four keys, each of which maps to each attribute—Semester, Instructor, Class Size, and Rating. Their values are `"Fall 2020"`, `"A"`, `100`, and `"Not recommended at all"`.

Extensible Markup Language (XML)

The XML format is like JSON, but it lists each item of the dataset using different symbols named *tags*. An **XML tag** is any block of text that consists of a pair of angle brackets (< >) with some text inside. Let's look at the example XML representation of the same dataset in Figure 1.8. Note that the screenshot of

CourseEvaluations.xml below only includes the first three items in the original dataset.

```
<Members>
    <Evaluation>
        <Semester>"Fall 2020"</Semester>
        <Instructor>"A"</Instructor>
        <Classsize>100</Classsize>
        <Rating>"Not recommended at all"</Rating>
    </Evaluation>
    <Evaluation>
        <Semester>"Spring 2021"</Semester>
        <Instructor>"A"</Instructor>
        <Classsize>50</Classsize>
        <Rating>"Highly recommended"</Rating>
    </Evaluation>
    <Evaluation>
        <Semester>"Fall 2021"</Semester>
        <Instructor>"B"</Instructor>
        <Classsize>120</Classsize>
        <Rating>"Not quite recommended"</Rating>
    </Evaluation>
</Members>
```

Figure 1.8 ch1-courseEvaluations.xml with the First Three Entries Only, Opened with Visual Studio Code

CourseEvaluations.xml lists each item of the dataset between a pair of tags, `<members>` and `</members>`. Under `<members>`, each item is defined between `<evaluation>` and `</evaluation>`. Since the dataset in Figure 1.8 has three items, we can see three blocks of `<evaluation>` ... `</evaluation>`. Each item has four attributes, and they are defined as different XML tags as well—`<semester>`, `<instructor>`, `<classsize>`, and `<rating>`. They are also followed by closing tags such as `</semester>`, `</instructor>`, `</classsize>`, and `</rating>`.

PubMed datasets (https://openstax.org/r/pubmed1) provides a list of articles that are published in the National Library of Medicine in XML format. Click Annual Baseline and download/open any .xml file. Note that all the .xml files are so big that they are compressed to .gz files. However, once you download one and attempt to open it by double-clicking, the file will automatically be decompressed and open. You will see a bunch of XML tags along with information about numerous publications, such as published venue, title, published date, etc.

XML and Image Data

The XML format is also commonly used as an attachment to some image data. It is used to note supplementary information about the image. For example, the Small Traffic Light Dataset (https://openstax.org/r/traffic) in Figure 1.9 comes with a set of traffic light images, placed in one of the three directories: `JPEGImages`, `train_images`, and `valid_images`. Each image directory is accompanied with another directory just for annotations such as `Annotations`, `train_annotations`, and `valid_annotations`.

```
├── Annotations
├── class.txt
├── JPEGImages
├── train_annotations
├── train_images
├── valid_annotations
└── valid_images
```

Figure 1.9 The Directory Structure of the Small Traffic Light Dataset

The annotation directories have a list of XML files, each of which corresponds to an image file with the same filename inside the corresponding image directory (Figure 1.10). Figure 1.11 shows that the first XML file in the Annotations directory includes information about the .jpg file with the same filename.

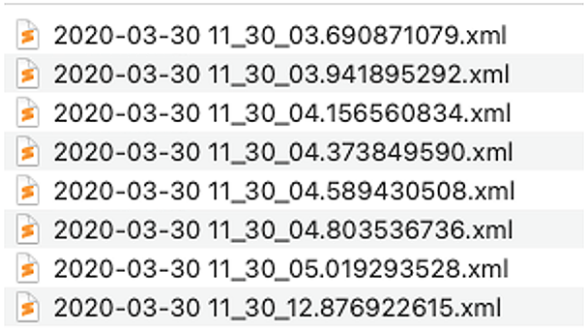


Figure 1.10 List of XML Files under the Annotations Directory in the Small Traffic Light Dataset (source: "Small Traffic Light Dataset," https://www.kaggle.com/datasets/sovitrath/small-traffic-light-dataset-xml-format)

```
<annotation>
  <folder/>
  <filename>2020-03-30 11_30_03.690871079.jpg</filename>
  <database/>
  <annotation/>
  <image/>
  <size>
    <height>1080</height>
    <width>1920</width>
    <depth>3</depth>
  </size>
  <segmented/>
  <object>
    <name>green</name>
    <pose/>
    <truncated/>
    <difficult/>
    <bndbox>
      <xmin>616</xmin>
      <ymin>477</ymin>
      <xmax>633</xmax>
      <ymax>521</ymax>
    </bndbox>
  </object>
```

Figure 1.11 2020-03-30 11_30_03.690871079.xml, an Example XML file within the Small Traffic Light Dataset (Source: "Small Traffic Light Dataset," https://www.kaggle.com/datasets/sovitrath/small-traffic-light-dataset-xml-format)

JSON and XML Dataset Descriptions

Both JSON and XML files often include some description(s) of the dataset itself as well (known as *metadata*), and they are included as a separate entry in the file ({} or <>). In Figure 1.12 and Figure 1.13, the actual data entries are listed inside "itemData" and `<data>`, respectively. The rest are used to provide background information on the dataset. For example:

- "creationDateTime": describes when the dataset was created.
- `<name>` is used to write the name of this dataset.
- `<metadata>` is used to describe each column name of the dataset along with its data type.

```
{
    "creationDateTime": "2023-03-22T11:53:27",
    "datasetJSONVersion": "1.0.0",
    "fileOID": "www.sponsor.org.project123.final",
    "sourceSystemVersion": "1.2.3",
    "clinicalData": {
        "items": [
                    {"OID": "ITEMGROUPDATASEQ", "name": "ITEMGROUPDATASEQ", "label": "Record identifier", "type": "integer"},
                    {"OID": "IT.STUDYID", "name": "STUDYID", "label": "Study identifier", "type": "string"},
                    {"OID": "IT.DOMAIN", "name": "DOMAIN", "label": "Domain identifier", "type": "string", "length": 2},
            ],
            "itemData": [
                [1, "MyStudy1", "D1"],
                [2, "MyStudy2", "D2"],
                ...
            ]
        }
    }
}
```

Figure 1.12 An Example JSON File with Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<dataset>
    <name>CEREALS</name>
    <metadata>
        <version>1.2</version>
        <date>04/02/2024</date>
        <description>a list of some popular cereals</description>
        <col name="NAME" type="string"/>
        <col name="MANUFACTURER" type="string"/>
        <col name="CALORIES_PER_SERVING" type="integer"/>
    </metadata>
    <data>
        <row>
            <value>Cheerios</value>
            <value>General Mills</value>
            <value>140</value>
        </row>
        <row>
            <value>Corn Flakes</value>
            <value>Kellogg's</value>
            <value>150</value>
        </row>
    </data>
</dataset>
```

Figure 1.13 An Example XML Dataset with Metadata

The Face Mask Detection (https://openstax.org/r/andrewmvd) dataset has a set of images of human faces with masks on. It follows a similar structure as well. The dataset consists of two directories—annotations and images. The former is in the XML format. The name of each XML file includes any text description about the

image with the same filename. For example, “maksssksksss0.xml” includes information on “maksssksksss0.png.”

EXAMPLE 1.7

Problem

The Iris Flower dataset (ch1-iris.csv (https://openstax.org/r/filed)) is a classic dataset in the field of data analysis.[1] Download this dataset and open it with a code editor (e.g., Sublime Text, XCode, Visual Studio Code). (We recommend that if you do not have any code editor installed, you install one. All three of these editors are quick and easy to install.) Now, answer these questions:

- How many items are there in the dataset?
- How many attributes are there in the dataset?
- What is the second attribute in the dataset?

Solution

There are 151 rows in the dataset with the header row at the top, totaling 150 items. There are five attributes listed across columns: `sepal_length`, `sepal_width`, `petal_length`, `petal_width`, `species`. The second attribute is `sepal_width`.

EXAMPLE 1.8

Problem

The Jeopardy dataset (ch1-jeopardy.json (https://openstax.org/r/filed15)) is formatted in JSON. Download and open it with a code editor (e.g., Notepad, Sublime Text, Xcode, Visual Studio Code).

- How many items are there in the dataset?
- How many attributes are there in the dataset?
- What is the third item in the dataset?

Solution

There are 409 items in the dataset, and each item has seven attributes: “`category`”, “`air_date`”, “`question`”, “`value`”, “`answer`”, “`round`”, and “`show_number`”. The third item is located at index 2 of the first list as shown in Figure 1.14.

```
"category" : string "EVERYBODY TALKS ABOUT IT..."
"air_date" : string "2004-12-31"
"question" :
string "'The city of Yuma in this state has a record average of 4,055 hours of sunshine each year'"
"value" : string "$200"
"answer" : string "Arizona"
"round" : string "Jeopardy!"
"show_number" : string "4680"
```

Figure 1.14 The Third Item in the Jeopardy Dataset

1 The Iris Flower dataset was introduced by the British statistician and biologist Ronald Fisher in his 1936 paper “The Use of Multiple Measurements in Taxonomic Problems.” This work became a landmark study in the use of multivariate data in classification problems and frequently makes an appearance in data science as a convenient test case for machine learning and neural network algorithms. The Iris Flower dataset is often used as a beginner's dataset to demonstrate various techniques, such as classification of algorithms, formatted in CSV.

1.4 Using Technology for Data Science

Learning Outcomes

By the end of this section, you should be able to:

- 1.4.1 Explain how statistical software can help with data analysis.
- 1.4.2 Explain the uses of different programs and programming languages for data manipulation, analysis, and visualizations.
- 1.4.3 Explain the uses of various data analysis tools used in data science applications.

Technology empowers data analysts, researchers, and organizations to leverage data, extract actionable insights, and make decisions that optimize processes and improve outcomes in many areas of our lives. Specifically, technology provides the tools, platforms, and algorithms that enable users to efficiently process and analyze data—especially complex datasets. The choice of technology used for a data science project will vary depending on the goals of the project, the size of the datasets, and the kind of analysis required.

Spreadsheet Programs

Spreadsheet programs such as Excel and Google Sheets are software applications consisting of electronic worksheets with rows and columns where data can be entered, manipulated, and calculated. Spreadsheet programs offer a variety of functions for data manipulation and can be used to easily create charts and tables. **Excel** is one of the most widely used spreadsheet programs, and as part of Microsoft Office, it integrates well with other Office products. Excel was first released by Microsoft in 1987, and it has become one of the most popular choices for loading and analyzing tabular data in a spreadsheet format. You are likely to have used Excel in some form or other —perhaps to organize the possible roommate combinations in your dorm room or to plan a party, or in some instructional context. We refer to the use of Excel to manipulate data in some of the examples of this text because sometimes a spreadsheet is simply the easiest way to work with certain datasets. (See Appendix A: Review of Excel for Data Science for a review of Excel functionality.)

Google Sheets is a cloud-based spreadsheet program provided by Google as part of the Google Workspace. Because it is cloud-based, it is possible to access spreadsheets from any device with an internet connection. This accessibility allows for collaboration and real-time updates among multiple users, enhancing communication within a team and making it ideal for team projects or data sharing among colleagues. Users can leave comments, track changes, and communicate within the spreadsheet itself.

The user interfaces for both Excel and Google Sheets make these programs very user-friendly for many applications. But these programs have some limitations when it comes to large databases or complex analyses. In these instances, data scientists will often turn to a programming language such as Python, R, or SPSS.

Programming Languages

A **programming language** is a formal language that consists of a set of instructions or commands used to communicate with a computer and to instruct it to perform specific tasks that may include data manipulation, computation, and input/output operations. Programming languages allow developers to write algorithms, create software applications, and automate tasks and are better suited than spreadsheet programs to handle complex analyses.

Python and R are two of the most commonly used programming languages today. Both are open-source programming languages. While **Python** started as a general-purpose language that covers various types of tasks (e.g., numerical computation, data analysis, image processing, and web development), **R** is more specifically designed for statistical computing and graphics. Both use simple syntax compared to conventional programming languages such as Java or C/C++. They offer a broad collection of packages for data manipulation, statistical analysis, visualization, machine learning, and complex data modeling tasks.

We have chosen to focus on Python in this text because of its straightforward and intuitive syntax, which makes it especially easy for beginners to learn and apply. Also, Python skills can apply to a wide range of computing work. Python also contains a vast network of libraries and frameworks specifically designed for data analysis, machine learning, and scientific computing. As we'll see, Python libraries such as NumPy (https://openstax.org/r/nump), Pandas (https://openstax.org/r/panda), Matplotlib (https://openstax.org/r/matplot), and Seaborn (https://openstax.org/r/seabo) provide powerful tools for data manipulation, visualization, and machine learning tasks, making Python a versatile choice for handling datasets. You'll find the basics of R and various R code examples in Appendix B: Review of R Studio for Data Science. If you want to learn how to use R, you may want to practice with RStudio (https://openstax.org/r/posit1), a commonly used software application to edit/run R programs.

PYTHON IN EXCEL

Microsoft recently launched a new feature for Excel named "Python in Excel." This feature allows a user to run Python code to analyze data directly in Excel. This textbook does not cover this feature, but instead presents Python separately, as it is also crucial for you to know how to use each tool in its more commonly used environment. If interested, refer to Microsoft's announcement (https://openstax.org/r/youtu).

EXPLORING FURTHER

Specialized Programming Languages

Other programming languages are more specialized for a particular task with data. These include SQL, Scala, and Julia, among others, as briefly described in this article on "The Nine Top Programming Languages for Data Science (https://openstax.org/r/9top)."

Other Data Analysis/Visualization Tools

There are a few other data analysis tools that are strong in data visualization. Tableau (https://openstax.org/r/tableau1) and PowerBI (https://openstax.org/r/micro) are user-friendly applications for data visualization. They are known for offering more sophisticated, interactive visualizations for high-dimensional data. They also offer a relatively easy user interface for compiling an analysis dashboard. Both allow users to run a simple data analysis as well before visualizing the results, similar to Excel.

In general, data visualization aims to make complex data more understandable and usable. The tools and technology described in this section offer a variety of ways to go about creating visualizations that are most accessible. Refer to Visualizing Data for a deeper discussion of the types of visualizations—charts, graphs, boxplots, histograms, etc.—that can be generated to help find the meaning in data.

EXPLORING FURTHER

Evolving Professional Standards

Data science is a field that is changing daily; the introduction of artificial intelligence (AI) has increased this pace. Technological, social, and ethical challenges with AI are discussed in Natural Language Processing, and ethical issues associated with the whole data science process, including the use of machine learning and artificial intelligence, are covered in Ethics Throughout the Data Science Cycle. A variety of data science professional organizations are working to define and update process and ethical standards on an ongoing basis. Useful references may include the following:

- Initiative for Analytics and Data Science Standards (IADSS) (https://openstax.org/r/iadss1)
- Data Science Association (DSA) (https://openstax.org/r/datascienceassn1)
- Association of Data Scientists (ADaSci) (https://openstax.org/r/adasci1)

1.5 Data Science with Python

Learning Outcomes

By the end of this section, you should be able to

- 1.5.1 Load data to Python.
- 1.5.2 Perform basic data analysis using Python.
- 1.5.3 Use visualization principles to graphically plot data using Python.

Multiple tools are available for writing and executing Python programs. Jupyter Notebook is one convenient and user-friendly tool. The next section explains how to set up the Jupyter Notebook environment using Google Colaboratory (Colab) and then provides the basics of two open-source Python libraries named `Pandas` and `Matplotlib`. These libraries are specialized for data analysis and data visualization, respectively.

EXPLORING FURTHER

Python Programming

In the discussion below, we assume you are familiar with basic Python syntax and know how to write a simple program using Python. If you need a refresher on the basics, please refer to Das, U., Lawson, A., Mayfield, C., & Norouzi, N. (2024). *Introduction to Python Programming*. OpenStax. https://openstax.org/books/introduction-python-programming/pages/1-introduction (https://openstax.org/r/page1).

Jupyter Notebook on Google Colaboratory

Jupyter Notebook is a web-based environment that allows you to run a Python program more interactively, using programming code, math equations, visualizations, and plain texts. There are multiple web applications or software you could use to edit a Jupyter Notebook, but in this textbook we will use Google's free application named **Google Colaboratory (Colab)** (https://openstax.org/r/colab1), often abbreviated as Colab. It is a cloud-based platform, which means that you can open, edit, run, and save a Jupyter Notebook on your Google Drive.

Setting up Colab is simple. On your Google Drive, click New > More. If your Google Drive has already installed Colab before, you will see Colaboratory under More. If not, click "Connect more apps" and install Colab by searching "Colaboratory" on the app store (Figure 1.15). For further information, see the Google Colaboratory Ecosystem (https://openstax.org/r/1pp) animation.

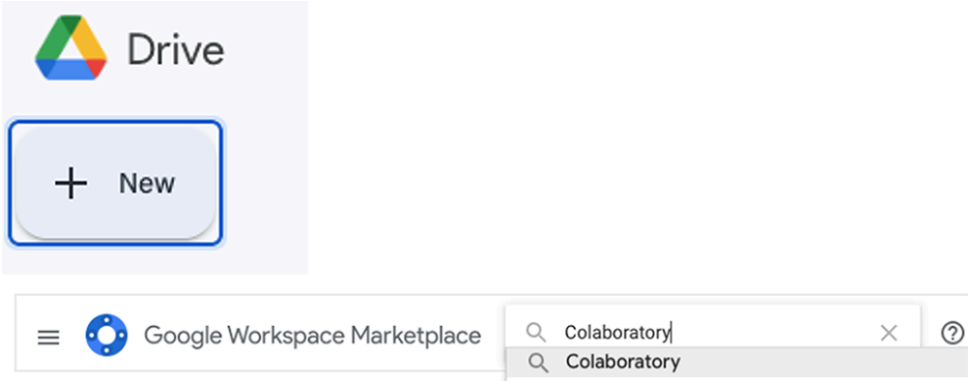


Figure 1.15 Install Google Colaboratory (Colab)

Now click New > More > Google Laboratory. A new, empty Jupyter Notebook will show up as in Figure 1.16.

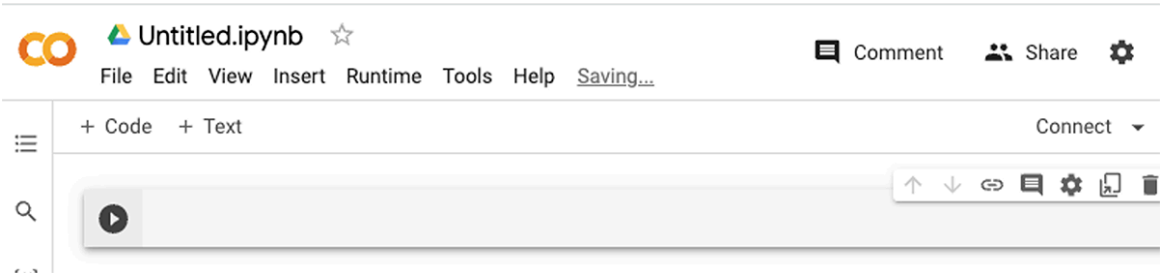


Figure 1.16 Google Colaboratory Notebook

The gray area with the play button is called a cell. A **cell** is a block where you can type either code or plain text. Notice that there are two buttons on top of the first cell—"+ Code" and "+ Text." These two buttons add a code or text cell, respectively. A code cell is for the code you want to run; a text cell is to add any text description or note.

Let's run a Python program on Colab. Type the following code in a code cell.

PYTHON CODE

```
print ("hello world!")
```

The resulting output will look like this:

```
hello world!
```

You can write a Python program across multiple cells and put text cells in between. Colab would treat all the code cells as part of a single program, running from the top to bottom of the current Jupyter Notebook. For example, the two code cells below run as if it is a single program.

When running one cell at a time from the top, we see the following outputs under each cell.

PYTHON CODE

```
a = 1
print ("The a value in the first cell:", a)
```

The resulting output will look like this:

```
The a value in the first cell: 1
```

PYTHON CODE

```
b = 3
print ("a in the second cell:", a)
print ("b in the second cell:", b)
a + b
```

The resulting output will look like this:

```
a in the second cell: 1
b in the second cell: 3
4
```

CONVENTIONAL PYTHON VERSUS JUPYTER NOTEBOOK SYNTAX

While conventional Python syntax requires `print()` syntax to print something to the program console, Jupyter Notebook does not require `print()`. On Jupyter Notebook, the line `a+b` instead of `print(a+b)` also prints the value of `a+b` as an output. But keep in mind that if there are multiple lines of code that trigger printing some values, *only* the output from the last line will show.

You can also run multiple cells in bulk. Click Runtime on the menu, and you will see there are multiple ways of running multiple cells at once (Figure 1.17). The two commonly used ones are "Run all" and "Run before." "Run all" runs all the cells in order from the top; "Run before" runs all the cells before the currently selected one.

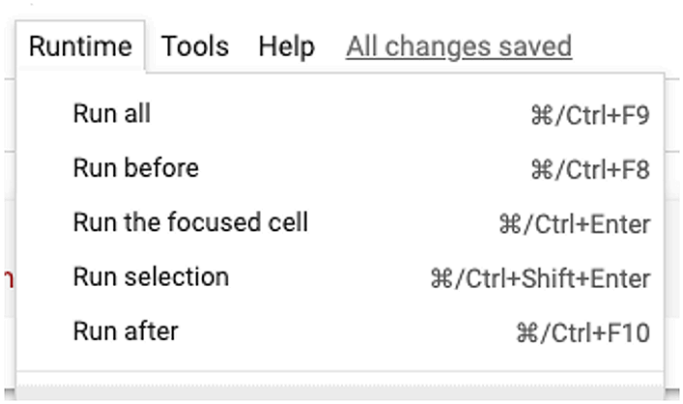


Figure 1.17 Multiple Ways of Running Cells on Colab

One thing to keep in mind is that being able to split a long program into multiple blocks and run one block at a time raises chances of user error. Let's look at a modified code from the previous example.

PYTHON CODE

```
a = 1
print ("the value in the first cell:", a)
```

The resulting output will look like this:

```
the value in the first cell: 1
```

PYTHON CODE

```
b = 3
print ("a in the second cell:", a)
print ("b in the second cell:", b)
a + b
```

The resulting output will look like this:

```
a in the second cell: 1
b in the second cell: 3
4
```

PYTHON CODE

```
a = 2
a + b
```

The resulting output will look like this:

```
5
```

The modified code has an additional cell at the end, updating `a` from 1 to 2. Notice that now `a+b` returns 5 as `a` has been changed to 2. Now suppose you need to run the second cell for some reason, so you run the second cell again.

PYTHON CODE

```
a = 1
print ("the a value in the first cell:", a)
```

The resulting output will look like this:

```
the a value in the first cell: 1
```

PYTHON CODE

```
b = 3
print ("a in the second cell:", a)
print ("b in the second cell:", b)
a + b
```

The resulting output will look like this:

```
a in the second cell: 2
b in the second cell: 3
5
```

PYTHON CODE

```
a = 2
a + b
```

The resulting output will look like this:

```
5
```

The value of `a` has changed to 2. This implies that the execution order of each cell matters! If you have run the third cell before the second cell, the value of `a` will have the value from the third one even though the third cell is located below the second cell. Therefore, it is recommended to use “Run all” or “Run before” after you make changes across multiple cells of code. This way your code is guaranteed to run sequentially from the top.

Python Pandas

One of the strengths of Python is that it includes a variety of free, open-source libraries. Libraries are a set of already-implemented methods that a programmer can refer to, allowing a programmer to avoid building common functions from scratch.

`Pandas` is a Python library specialized for data manipulation and analysis, and it is very commonly used among data scientists. It offers a variety of methods, which allows data scientists to quickly use them for data analysis. You will learn how to analyze data using `Pandas` throughout this textbook.

Colab already has `Pandas` installed, so you just need to import `Pandas` and you are set to use all the methods in `Pandas`. Note that it is convention to abbreviate pandas to pd so that when you call a method from `Pandas`, you can do so by using `pd` instead of having to type out `Pandas` every time. It offers a bit of convenience for a programmer!

PYTHON CODE

```
# import Pandas and assign an abbreviated identifier "pd"
import pandas as pd
```

EXPLORING FURTHER

Installing Pandas on Your Computer

If you wish to install Pandas on your own computer, refer to the installation page of the Pandas website (https://openstax.org/r/pyd).

Load Data Using Python Pandas

The first step for data analysis is to load the data of your interest to your Notebook. Let's create a folder on Google Drive where you can keep a CSV file for the dataset and a Notebook for data analysis. Download a public dataset, ch1-movieprofit.csv (https://openstax.org/r/filed), and store it in a Google Drive folder. Then open a new Notebook in that folder by entering that folder and clicking New > More > Google Colaboratory.

Open the Notebook and allow it to access files in your Google Drive by following these steps:

First, click the Files icon on the side tab (Figure 1.18).

Figure 1.18 Side Tab of Colab

Then click the Mount Drive icon (Figure 1.19) and select "Connect to Google Drive" on the pop-up window.

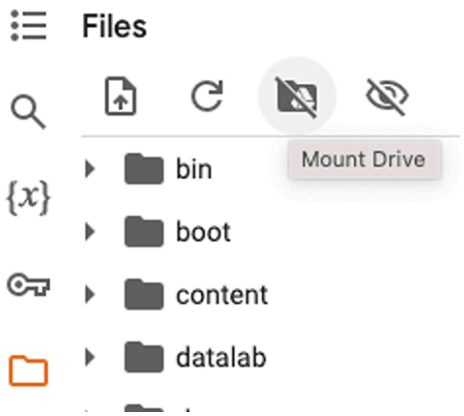


Figure 1.19 Features under Files on Colab

Notice that a new cell has been inserted on the Notebook as a result (Figure 1.20).

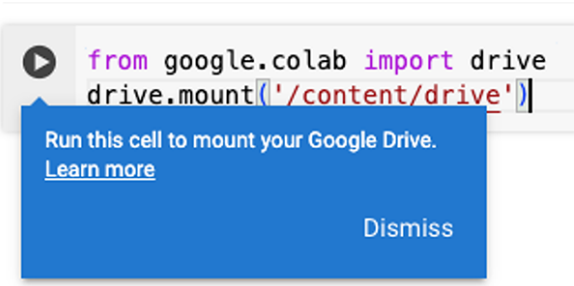


Figure 1.20 An Inserted Cell to Mount Your Google Drive

Connect your Google Drive by running the cell, and now your Notebook file can access all the files under content/drive. Navigate folders under drive to find your Notebook and ch1-movieprofit.csv (https://openstax.org/r/filed) files. Then click "..." > Copy Path (Figure 1.21).

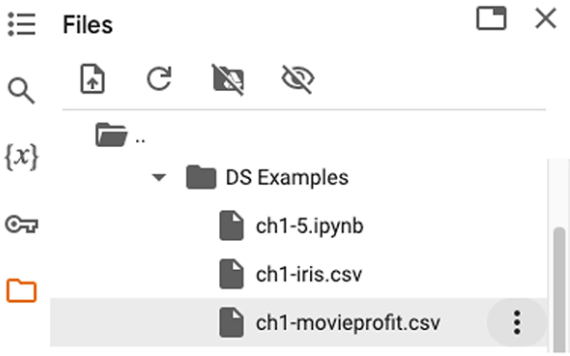


Figure 1.21 Copying the Path of a CSV File Located in a Google Drive Folder

Now replace [Path] with the copied path in the below code. Run the code and you will see the dataset has been loaded as a table and stored as a Python variable data.

PYTHON CODE

```
# import Pandas and assign an abbreviated identifier "pd"
import pandas as pd
```

```
data = pd.read_csv("[Path]")
data
```

The resulting output will look like this:

	Unnamed: 0	Title	Year	Genre	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million	Votes
0	1	Avatar	2009	Action	7.8	162	760.51	2847.40	1,236,962
1	2	Avengers: Endgame	2019	Action	8.4	181	858.37	2797.50	1,108,641
2	3	Titanic	1997	Drama	7.9	194	659.33	2201.65	1,162,142
3	4	Star Wars: Episode VII - The Force Awakens	2015	Action	7.8	138	936.66	2069.52	925,551
4	5	Avengers: Infinity War	2018	Action	8.4	149	678.82	2048.36	1,062,517
...	...	...	...	...	...	...	...	...	...
961	962	The A-Team	2010	Action	6.7	117	77.22	177.24	259,316
962	963	Tootsie	1982	Comedy	7.4	116	177.20	177.20	107,311
963	964	In the Line of Fire	1993	Action	7.2	128	102.31	177.00	104,598
964	965	Analyze This	1999	Comedy	6.7	103	106.89	176.89	154,726
965	966	The Hitman's Bodyguard	2017	Action	6.9	118	75.47	176.60	230,821

The `read_csv()` method in `Pandas` loads a CSV file and stores it as a DataFrame. A **DataFrame** is a data type that `Pandas` uses to store multi-column tabular data. Therefore, the variable data holds the table in ch1-movieprofit.csv (https://openstax.org/r/filed) in the form of a `Pandas` DataFrame.

DATAFRAME VERSUS SERIES

`Pandas` defines two data types for tabular data—DataFrame and Series. While DataFrame is used for multi-column tabular data, Series is used for single-column data. Many methods in `Pandas` support both DataFrame and Series, but some are only for one or the other. It is always good to check if the method you are using works as you expect. For more information, refer to the Pandas documentation (https://openstax.org/r/docs) or Das, U., Lawson, A., Mayfield, C., & Norouzi, N. (2024). *Introduction to Python Programming*. OpenStax. https://openstax.org/books/introduction-python-programming/pages/1-introduction (https://openstax.org/r/page1).

EXAMPLE 1.9

Problem

Remember the Iris dataset we used in Data and Datasets? Load the dataset ch1-iris.csv (https://openstax.org/r/filed) to a Python program using `Pandas`.

Solution

The following code loads the ch1-iris.csv (https://openstax.org/r/filed) that is stored in a Google Drive. Make sure to replace the path with the actual path to ch1-iris.csv (https://openstax.org/r/filed) on your Google Drive.

PYTHON CODE

```
import pandas as pd

data = pd.read_csv("[Path to ch1-iris.csv]") # Replace the path
data
```

The resulting output will look like this:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

EXPLORING FURTHER

Can I load a file that is uploaded to someone else's Google Drive and shared with me?

Yes! This is useful especially when your Google Drive runs out of space. Simply add the shortcut of the shared file to your own drive. Right-click > Organize > Add Shortcut will let you select where to store the shortcut. Once done, you can call `pd.read_csv()` using the path of the shortcut.

Summarize Data Using Python Pandas

You can compute basic statistics for data quite quickly by using the `DataFrame.describe()` method. Add and run the following code in a new cell. It calls the `describe()` method upon `data`, the DataFrame we defined earlier with ch1-movieprofit.csv (https://openstax.org/r/filed).

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-movieprofit.csv]")
data.describe()
```

like this:

	Unnamed: 0	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million
count	966.00000	966.000000	966.000000	966.000000	966.000000
mean	483.50000	6.814286	117.506211	156.158975	410.140600
std	279.00448	0.894383	21.615612	110.629617	294.758791
min	1.00000	3.300000	69.000000	0.010000	176.600000
25%	242.25000	6.200000	101.250000	90.832500	223.277500
50%	483.50000	6.800000	116.000000	129.245000	309.345000
75%	724.75000	7.400000	130.000000	187.090000	472.645000
max	966.00000	9.200000	238.000000	936.660000	2847.400000

`describe()` returns a table whose columns are a subset of the columns in the entire dataset and whose rows are different statistics. The statistics include the number of unique values in a column (`count`), mean (`mean`), standard deviation (`std`), minimum and maximum values (`min/max`), and different quartiles (`25%/50%/75%`), which you will learn about in Measures of Variation. Using this representation, you can compute such statistics of different columns easily.

EXAMPLE 1.10

Problem

Summarize the IRIS dataset using `describe()` of ch1-iris.csv (https://openstax.org/r/filed) you loaded in the previous example.

Solution

The following code in a new cell returns the summary of the dataset.

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-iriscsv]")
data.describe()
```

The resulting output will look like this:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Select Data Using Python Pandas

The `Pandas` DataFrame allows a programmer to use the column name itself when selecting a column. For example, the following code prints all the values in the "US_Gross_Million" column in the form of a Series (remember the data from a single column is stored in the Series type in `Pandas`).

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-movieprofit.csv]")

data["US_Gross_Million"]
```

like this:

```
0   760.51
1   858.37
2   659.33
3   936.66
4   678.82
    ...
961   77.22
962  177.20
963  102.31
964  106.89
965   75.47
Name: US_Gross_Million, Length: 966, dtype: float64
```

`DataFrame.iloc[]` enables a more powerful selection—it lets a programmer select by both column and row, using column and row indices. Let's look at some code examples below.

PYTHON CODE

```
data.iloc[:, 2] # select all values in the second column
```

The resulting output will look like this:

```
0   2009
1   2019
2   1997
3   2015
4   2018
    ...
961  2010
962  1982
963  1993
964  1999
965  2017
Name: Year, Length: 966, dtype: object
```

PYTHON CODE

```
data.iloc[2,:] # select all values in the third row
```

The resulting output will look like this:

```
Unnamed: 0             3
Title            Titanic
Year              1997
Genre             Drama
Rating              7.9
Duration             194
US_Gross_Million       659.33
Worldwide_Gross_Million   2201.65
Votes           1,162,142
Name: 2, dtype: object
```

To pinpoint a specific value within the “US_Gross_Million” column, you can use an index number.

PYTHON CODE

```
print (data["US_Gross_Million"][0]) # index 0 refers to the top row
print (data["US_Gross_Million"][2]) # index 2 refers to the third row
```

The resulting output will look like this:

```
760.51
659.33
```

You can also use `DataFrame.iloc[]` to select a specific group of cells on the table. The example code below shows different ways of using `iloc[]`. There are multiple ways of using `iloc[]`, but this chapter introduces a couple of common ones. You will learn more techniques for working with data throughout this textbook.

PYTHON CODE

```
data.iloc[:, 1] # select all values in the second column (index 1)
```

The resulting output will look like this:

```
0                      Avatar
1                Avengers: Endgame
2                      Titanic
3   Star Wars: Episode VII - The Force Awakens
4             Avengers: Infinity War
             ...
961                  The A-Team
962                    Tootsie
963              In the Line of Fire
964                 Analyze This
965            The Hitman's Bodyguard
Name: Title, Length: 966, dtype: object
```

PYTHON CODE

```
data.iloc[[1, 3], [2, 3]]
# select the rows at index 1 and 3, the columns at index 2 and 3
```

The resulting output will look like this:

	Year	Genre
1	2019	Action
3	2015	Action

EXAMPLE 1.11

Problem

Select a “sepal_width” column of the IRIS dataset using the column name.

Solution

The following code in a new cell returns the “sepal_width” column.

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-iris.csv]")

data["sepal_width"]
```

The resulting output will look like this:

```
0   3.5
1   3.0
2   3.2
3   3.1
4   3.6
   ...
145  3.0
146  2.5
147  3.0
148  3.4
149  3.0
Name: sepal_width, Length: 150, dtype: float64
```

EXAMPLE 1.12

Problem

Select a “petal_length” column of the IRIS dataset using `iloc[]`.

Solution

The following code in a new cell returns the “petal_length” column.

PYTHON CODE

```
data.iloc[:, 2]
```

The resulting output will look like this:

```
0   1.4
1   1.4
2   1.3
3   1.5
4   1.4
   ...
145  5.2
146  5.0
147  5.2
148  5.4
149  5.1
Name: petal_length, Length: 150, dtype: float64
```

Search Data Using Python Pandas

To search for some data entries that fulfill specific criteria (i.e., filter), you can use `DataFrame.loc[]` of `Pandas`. When you indicate the filtering criteria inside the brackets, [], the output returns the filtered rows within the DataFrame. For example, the code below filters out the rows whose genre is comedy. Notice that the output only has 307 out of the full 3,400 rows. You can check the output on your own, and you will see their Genre values are all “Comedy.”

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-movieprofit.csv]")

data.loc[data['Genre'] == 'Comedy']
```

The resulting output will look like this:

	Unnamed: 0	Title	Year	Genre	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million	Votes
161	162	Mamma Mia!	2008	Comedy	6.5	108	144.13	611.26	245,380
170	171	The Hangover Part II	2011	Comedy	6.4	102	254.46	586.76	499,126
183	184	Mei ren yu	2016	Comedy	6.2	94	3.23	553.81	9,374
186	187	Ted	2012	Comedy	6.9	106	218.82	549.37	612,897
208	209	Meet the Fockers	2004	Comedy	6.3	115	279.26	522.66	271,402
...	...	...	...	...	...	...	...	...	...
954	955	How to Lose a Guy in 10 Days	2003	Comedy	6.4	116	105.81	177.50	238,604
957	958	The 40 Year Old Virgin	2005	Comedy	7.1	116	109.45	177.38	436,221
960	961	The Descendants	2011	Comedy	7.3	115	82.58	177.24	242,388
962	963	Tootsie	1982	Comedy	7.4	116	177.20	177.20	107,311
964	965	Analyze This	1999	Comedy	6.7	103	106.89	176.89	154,726

EXAMPLE 1.13

Problem

Using `DataFrame.loc[]`, search for all the items of *Iris-virginica* species in the IRIS dataset.

Solution

The following code returns a filtered DataFrame whose species are *Iris-virginica*. All such rows show up as an output.

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-iris.csv]")

data.loc[data['species'] == 'Iris-virginica']
```

The resulting figure will look like this:

	sepal_length	sepal_width	petal_length	petal_width	species
100	6.3	3.3	6.0	2.5	Iris-virginica
101	5.8	2.7	5.1	1.9	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
104	6.5	3.0	5.8	2.2	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
106	4.9	2.5	4.5	1.7	Iris-virginica
107	7.3	2.9	6.3	1.8	Iris-virginica
108	6.7	2.5	5.8	1.8	Iris-virginica

```
(Rows 109 through 149 not shown.)
```

EXAMPLE 1.14

Problem

This time, search for all the items whose species is *Iris-virginica* and whose sepal width is wider than 3.2.

Solution

You can use a Boolean expression—in other words, an expression that evaluates as either True or False—inside `data.loc[]`.

PYTHON CODE

```
data.loc[(data['species'] == 'Iris-virginica') & (data['sepal_width'] > 3.2)]
```

The resulting output will look like this:

	sepal_length	sepal_width	petal_length	petal_width	species
100	6.3	3.3	6.0	2.5	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica

Visualize Data Using Python Matplotlib

There are multiple ways to draw plots of data in Python. The most common and straightforward way is to import another library, `Matplotlib`, which is specialized for data visualization. `Matplotlib` is a huge library, and to draw the plots you only need to import a submodule named `pyplot`.

Type the following import statement in a new cell. Note it is convention to denote `matplotlib.pyplot` with `plt`, similarly to denoting `Pandas` with `pd`.

PYTHON CODE

```
import matplotlib.pyplot as plt
```

`Matplotlib` offers a method for each type of plot, and you will learn the `Matplotlib` methods for all of the commonly used types throughout this textbook. In this chapter, however, let's briefly look at how to draw a plot using `Matplotlib` in general.

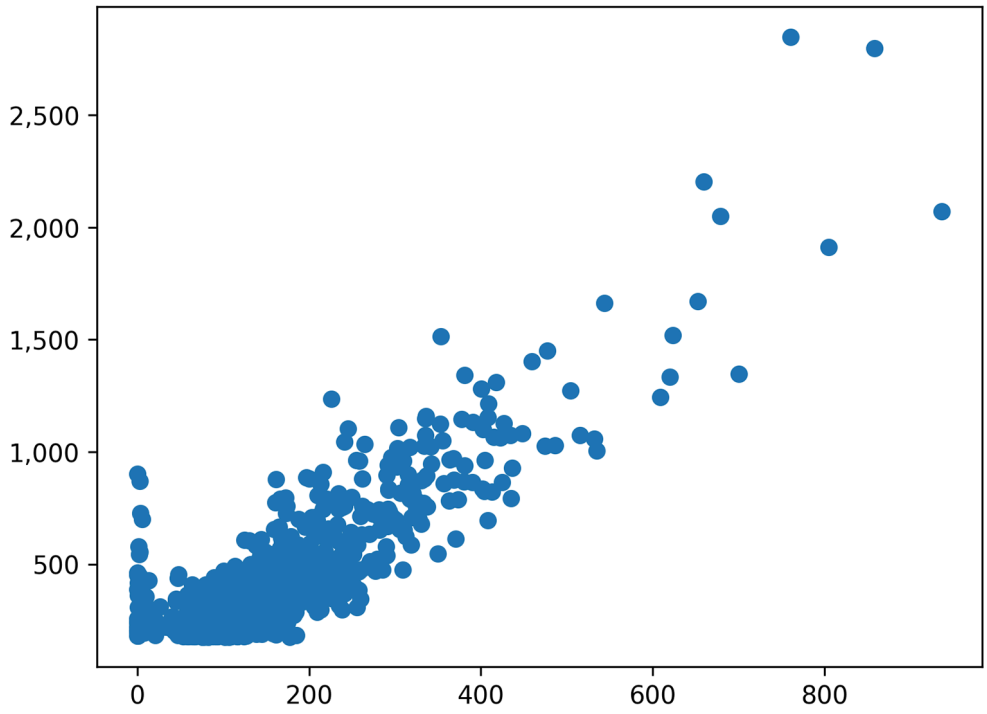
Suppose you want to draw a scatterplot between "US_Gross_Million" and "Worldwide_Gross_Million" of the movie profit dataset (ch1-movieprofit.csv (https://openstax.org/r/filed)). You will investigate scatterplots in more detail in Correlation and Linear Regression Analysis. The example code below draws such a scatterplot using the method `scatter()`. `scatter()` takes the two columns of your interest—data["US_Gross_Million"] and data["Worldwide_Gross_Million"]—as the inputs and assigns them for the *x*- and *y*-axes, respectively.

PYTHON CODE

```
data = pd.read_csv("[Path to ch1-movieprofit.csv]")
```

```
# draw a scatterplot using matplotlib's scatter()
plt.scatter(data["US_Gross_Million"], data["Worldwide_Gross_Million"])
```

The resulting output will look like this:

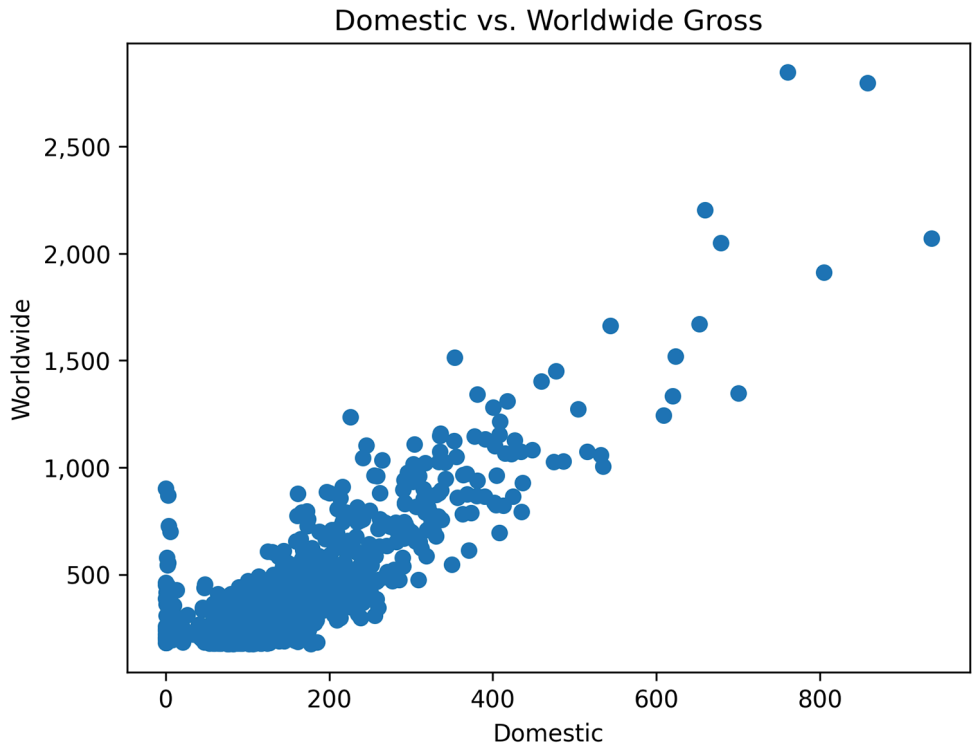


Notice that it simply has a set of dots on a white plane. The plot itself does not show what each axis represents, what this plot is about, etc. Without them, it is difficult to capture what the plot shows. You can set these with the following code. The resulting plot below indicates that there is a positive correlation between domestic gross and worldwide gross.

PYTHON CODE

```
# draw a scatterplot
plt.scatter(data["US_Gross_Million"], data["Worldwide_Gross_Million"])

# set the title
plt.title("Domestic vs. Worldwide Gross")

# set the x-axis label
plt.xlabel("Domestic")

# set the y-axis label
plt.ylabel("Worldwide")
```

The resulting output will look like this:

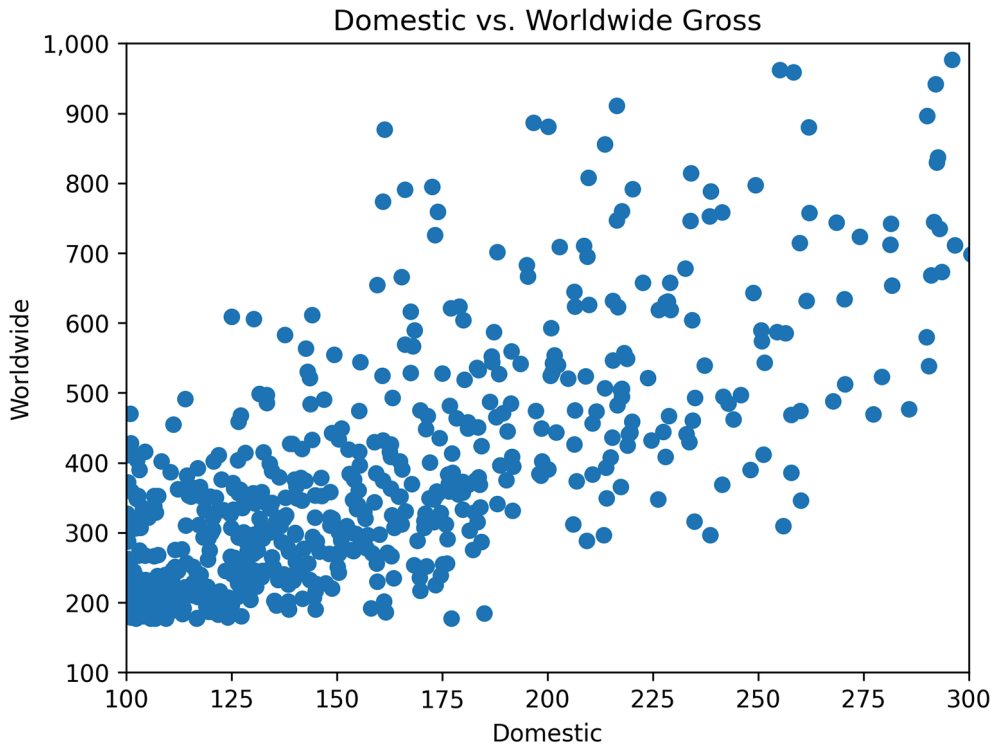


You can also change the range of numbers along the *x*- and *y*-axes with `plt.xlim()` and `plt.ylim()`. Add the following two lines of code to the cell in the previous Python code example, which plots the scatterplot.

PYTHON CODE

```
# draw a scatterplot
plt.scatter(data["US_Gross_Million"], data["Worldwide_Gross_Million"])

# set the title
plt.title("Domestic vs. Worldwide Gross")

# set the x-axis label
plt.xlabel("Domestic")

# set the y-axis label
plt.ylabel("Worldwide")

# set the range of values of the x- and y-axes
plt.xlim(1*10**2, 3*10**2) # x axis: 100 to 300
plt.ylim(1*10**2, 1*10**3) # y axis: 100 to 1,000
```

The resulting output will look like this:



The resulting plot with the additional lines of code has a narrower range of values along the *x*- and *y*-axes.

EXAMPLE 1.15

Problem

Using the iris dataset, draw a scatterplot between petal length and height of *Setosa Iris*. Set the title, *x*-axis label, and *y*-axis label properly as well.

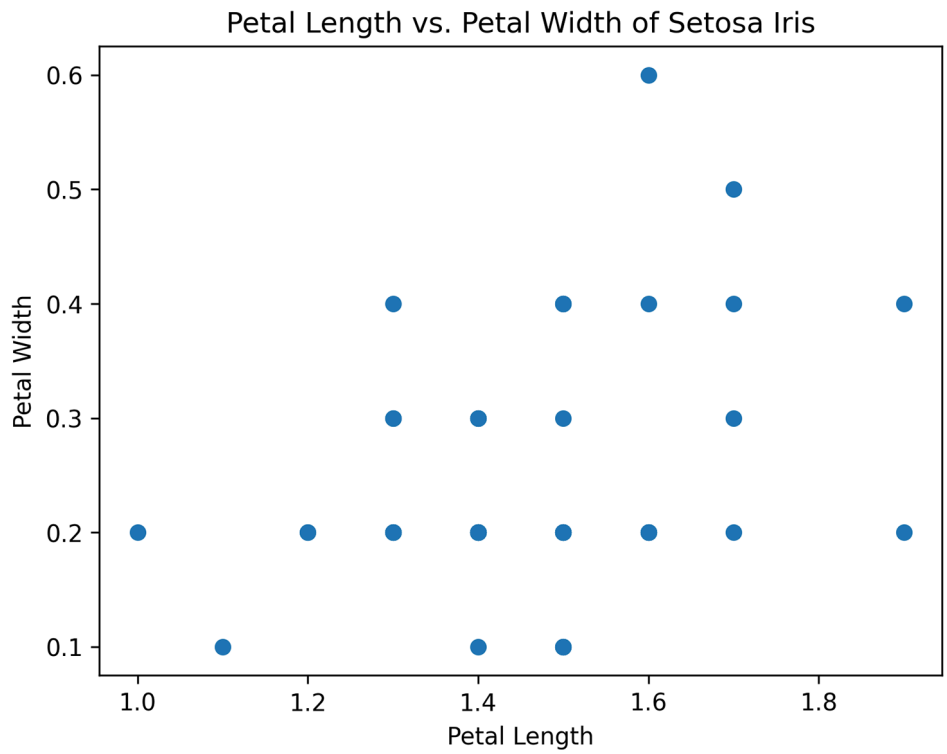
Solution

PYTHON CODE

```
import matplotlib.pyplot as plt

data = pd.read_csv("[Path to ch1-iris.csv]")

# select the rows whose species are Setosa Iris
setosa = data.loc[(data['species'] == 'Iris-setosa')]

# draw a scatterplot
plt.scatter(setosa["petal_length"], setosa["petal_width"])
```

```
# set the title
plt.title("Petal Length vs. Petal Width of Setosa Iris")

# set the x-axis label
plt.xlabel("Petal Length")

# set the y-axis label
plt.ylabel("Petal Width")
```

The resulting output will look like this:



Datasets

Note: The primary datasets referenced in the chapter code may also be [downloaded here (https://openstax.org/r/spreadsheetsd1)](https://openstax.org/r/spreadsheetsd1).

Key Terms

attribute characteristic or feature that defines an item in a dataset
categorical data data that is represented in different forms and do not indicate measurable quantities
cell a block or rectangle on a table that is specified with a combination of a row number and a column number
comma-separated values (CSV) format of a dataset in which each item takes up a single line and its values are separated by commas (“,”)
continuous data data whose value is chosen from an infinite set of numbers
data anything that we can analyze to compile some high-level insights
data analysis the process of examining and interpreting raw data to uncover patterns, discover meaningful insights, and make informed decisions
data collection the systematic process of gathering information on variables of interest
data preparation (data processing) the second step within the data science cycle; converts the collected data into an optimal form for analysis
data reporting the presentation of data in a way that will best convey the information learned from data analysis
data science a field of study that investigates how to collect, manage, and analyze data in order to retrieve meaningful information from some seemingly arbitrary data
data science cycle a process used when investigating data
data visualization the graphical representation of data to point out the patterns and trends involving the use of visual elements such as charts, graphs, and maps
data warehousing the process of storing and managing large volumes of data from various sources in a central location for easier access and analysis by businesses
DataFrame a data type that `Pandas` uses to store a multi-column tabular data
dataset a collection of related and organized information or data points grouped together for reference or analysis
discrete data data that follows a specific precision
Excel a spreadsheet program with a graphical user interface developed by Microsoft to help with the manipulation and analysis of data
Extensible Markup Language (XML) format of a dataset with which uses tags
Google Colaboratory (Colab) software for editing and running Jupyter Notebook files
Google Sheets a spreadsheet program with a graphical user interface developed by Google to help with the manipulation and analysis of data
information some high-level insights that are compiled from data
Internet of Things (IoT) the network of multiple objects interacting with each other through the Internet
item an element that makes up a dataset; also referred to as an *entry* and an *instance*
JavaScript Object Notation (JSON) format of a dataset that follows the syntax of the JavaScript programming language
Jupyter Notebook a web-based document that helps users run Python programs more interactively
nominal data data whose values do not include any ordering notion
numeric data data that are represented in numbers and indicate measurable quantities
ordinal data data whose values include an ordering notion
Pandas a Python library specialized for data manipulation and analysis
predictive analytics statistical techniques, algorithms, and machine learning that analyze historical data and make predictions about future events, an approach often used in medicine to offer more accurate diagnosis and treatment
programming language a formal language that consists of a set of instructions or commands used to communicate with a computer and instruct it to perform specific tasks
Python a programming language that has extensive libraries and is commonly used for data analysis

qualitative data non-numerical data that generally describe subjective attributes or characteristics and are analyzed using methods such as thematic analysis or content analysis
quantitative data data that can be measured by specific quantities and amounts and are often analyzed using statistical methods
R an open-source programming language that is specifically designed for statistical computing and graphics
recommendation system a system that makes data-driven, personalized suggestions for users
sabermetrics a statistical approach to sports team management
sports analytics use of data and business analytics in sports
spreadsheet program a software application consisting of electronic worksheets with rows and columns where data can be entered, manipulated, and calculated
structured data dataset whose individual items have the same structure
unstructured data dataset whose individual items have different structures
XML tag any block of text that consists of a pair of angle brackets (< >) with some text inside

Group Project

Project A: Data Source Quality

As a student of, or a new professional working in, data science, you will not always be collecting new primary data. It's just as important to be able to locate, critically evaluate, and properly clean existing sources of secondary data. (Collecting and Preparing Data will cover the topic of data collection and cleaning in more detail.)

Some reputable government data sources are:
Data.gov (https://openstax.org/r/datagov)

Bureau of Labor Statistics (BLS) (https://openstax.org/r/blsgov1)
National Oceanic and Atmospheric Administration (NOAA) (https://openstax.org/r/noaagov)

Some reputable nongovernment data sources are:
Kaggle (https://openstax.org/r/kaggle1)
Statista (https://openstax.org/r/statista)
Pew Research Center (https://openstax.org/r/pewresearch)

Using the suggested sources or similar-quality sources that you research on the Internet, find two to three datasets about the field or industry in which you intend to work. (You might also try to determine whether similar data sets are available at the national, state/province, and local/city levels.) In a group, formulate a specific, typical policy issue or business decision that managers in these organizations might make. For the datasets you found, compare and contrast their size, collection methods, types of data, update frequency and recency, and relevance to the decision question you have identified.

Project B: Data Visualization

Using one of the data sources mentioned in the previous project, find a dataset that interests you. Download it as a CSV file. Use Python to read in the CSV file as a `Pandas` DataFrame. As a group, think of a specific question that might be addressed using this dataset, discuss which features of the data seem most important to answer your question, and then use the Python libraries `Pandas` and `Matplotlib` to select the features and make graphs that might help to answer your question about the data. Note, you will learn many sophisticated techniques for doing data analysis in later chapters, but for this project, you should stick to simply isolating some data and visualizing it using the tools present in this chapter. Write a brief report on your findings.

Project C: Privacy, Ethics, and Bias

Identify at least one example from recent current events or news articles that is related to each of the

following themes (starting references given in parentheses):

a. Privacy concerns related to data collection (See the Protecting Personal Privacy (https://openstax.org/r/gaog) website of the U.S. Government Accountability Office.)
b. Ethics concerns related to data collection, including fair use of copyrighted materials (See the U.S. Copyright Office guidelines (https://openstax.org/r/fairuse).)
c. Bias concerns related to data collection (See the National Cancer Institute (NCI) article (https://openstax.org/r/bias) on data bias.)

Suppose that you are part of a data science team working for an organization on data collection for a major project or product. Discuss as a team how the issues of privacy, ethics, and equity (avoiding bias) could be addressed, depending on your position in the organization and the type of project or product.

Chapter Review

1. Select the incorrect step and goal pair of the data science cycle.
   a. Data collection: collect the data so that you have something for analysis.
   b. Data preparation: have the collected data stored in a server as is so that you can start the analysis.
   c. Data analysis: analyze the prepared data to retrieve some meaningful insights.
   d. Data reporting: present the data in an effective way so that you can highlight the insights found from the analysis.

2. Which of the following best describes the evolution of data management in the data science process?
   a. Initially, data was stored locally on individual computers, but with the advent of cloud-based systems, data is now stored on designated servers outside of local storage.
   b. Data management has remained static over time, with most data scientists continuing to store and process data locally on individual computers.
   c. The need for data management arose as a result of structured data becoming unmanageable, leading to the development of cloud-based systems for data storage.
   d. Data management systems have primarily focused on analysis rather than processing, resulting in the development of modern data warehousing solutions.

3. Which of the following best exemplifies the interdisciplinary nature of data science in various fields?
   a. A historian traveling to Italy to study ancient manuscripts to uncover historical insights about the Roman Empire
   b. A mathematician solving complex equations to model physical phenomena
   c. A biologist analyzing a large dataset of genetic sequences to gain insights about the genetic basis of diseases
   d. A chemist synthesizing new compounds in a laboratory

Critical Thinking

1. For each dataset (https://openstax.org/r/spreadsheetsd1), list the attributes.
   a. Spotify dataset
   b. CancerDoc dataset

2. For each dataset (https://openstax.org/r/spreadsheetsd1), define the type of the data based on following criteria and explain why:
   - Numeric vs. categorical
   - If it is numeric, continuous vs. discrete; if it is categorical, nominal vs. ordinal

   a. “artist_count” attribute of Spotify dataset
   b. “mode” attribute of Spotify dataset

   c. "key" attribute of Spotify dataset
   d. the second column in CancerDoc dataset

3. For each dataset (https://openstax.org/r/spreadsheetsd1), identify the type of the dataset—structured vs. unstructured. Explain why.
   a. Spotify dataset
   b. CancerDoc dataset

4. For each dataset (https://openstax.org/r/spreadsheetsd1), list the first data entry.
   a. Spotify dataset
   b. CancerDoc dataset

5. Open the WikiHow dataset (ch1-wikiHow.json (https://openstax.org/r/filed)) and list the attributes of the dataset.

6. Draw scatterplot between bpm (*x*-axis) and danceability (*y*-axis) of the Spotify dataset (https://openstax.org/r/filed) using:
   a. Python `Matplotlib`
   b. A spreadsheet program such as MS Excel or Google Sheets (Hint: Search "Scatterplot" on Help.)

7. Regenerate the scatterplot of the Spotify dataset (https://openstax.org/r/filed), but with a custom title and *x*-/*y*-axis label. The title should be "BPM vs. Danceability." The *x*-axis label should be titled "bpm" and range from the minimum to the maximum bpm value. The *y*-axis label should be titled "danceability" and range from the minimum to the maximum Danceability value.
   a. Python `Matplotlib` (Hint: `DataFrame.min()` and `DataFrame.max()` methods return min and max values of the DataFrame. You can call these methods upon a specific column of a DataFrame as well. For example, if a DataFrame is named `df` and has a column named "col1", `df["col1"].min()` will return the minimum value of the "col1" column of `df`.)
   b. A spreadsheet program such as MS Excel or Google Sheets (Hint: Calculate the minimum and maximum value of each column somewhere else first, then simply use the value when editing the scatterplot.)

8. Based on the Spotify dataset (https://openstax.org/r/spreadsheet4), filter the following using Python `Pandas:`
   a. Tracks whose artist is Taylor Swift
   b. Tracks that were sung by Taylor Swift and released earlier than 2020

Quantitative Problems

1. Based on the Spotify dataset (https://openstax.org/r/spreadsheet4), calculate the average bpm of the songs released in 2023 using:
   a. Python `Pandas`
   b. A spreadsheet program such as MS Excel or Google Sheets (Hint: The formula AVERAGE() computes the average across the cells specified in the parentheses. For example, within Excel, typing in the command "=AVERAGE(A1:A10)" in any empty cell will calculate the numeric average for the contents of cells A1 through A10. Search "AVERAGE function" on Help as well.)

References

Anaconda. (2020). *2020 state of data science*. https://www.anaconda.com/resources/whitepapers/state-of-data-science-2020

Clewlow, A. (2024, January 26). *Three smart cities that failed within five years of launch*. Intelligent Build.tech. https://www.intelligentbuild.tech/2024/01/26/three-smart-cities-that-failed-within-five-years-of-launch/

Hays, C. L. (2004, November 14). What Wal-Mart knows about customers' habits. *New York Times*. https://www.nytimes.com/2004/11/14/business/yourmoney/what-walmart-knows-about-customers-habits.html

Herrington, D. (2023, July 31). *Amazon is delivering its largest selection of products to U.S. Prime members at the fastest speeds ever*. Amazon News. https://www.aboutamazon.com/news/operations/doug-herrington-amazon-prime-delivery-speed

Hitachi. (2023, February 22). *Ag Automation and Hitachi drive farming efficiency with sustainable digital solutions*. Hitachi Social Innovation. https://social-innovation.hitachi/en-us/case_studies/digital-solutions-in-agriculture-drive-sustainability-in-farming/

IABAC. (2023, September 20). *Fraud detection through data analytics: Identifying anomalies and patterns*. International Association of Business Analytics Certification. https://iabac.org/blog/fraud-detection-through-data-analytics-identifying-anomalies-and-patterns

Statista. (2024, May 10). *Walmart: weekly customer visits to stores worldwide FY2017-FY2024*. https://www.statista.com/statistics/818929/number-of-weekly-customer-visits-to-walmart-stores-worldwide/

Van Bocxlaer, A. (2020, 20 August). *Sensors for a smart city*. RFID & Wireless IoT. https://www.rfid-wiot-search.com/rfid-wiot-global-sensors-for-a-smart-city

2 Collecting and Preparing Data

Figure 2.1 Periodic population surveys such as censuses help governments plan resources to supply public services. (credit: modification of work "Census 2010 @ La Fuente" by Jenn Turner/Flickr, CC BY 2.0)

Chapter Outline

2.1 Overview of Data Collection Methods
2.2 Survey Design and Implementation
2.3 Web Scraping and Social Media Data Collection
2.4 Data Cleaning and Preprocessing
2.5 Handling Large Datasets

Introduction

Data collection and preparation are the first steps in the data science cycle. They involve systematically gathering the necessary data to meet a project's objectives and ensuring its readiness for further analysis. Well-executed data collection and preparation serve as a solid foundation for effective, data-driven decision-making and aid in detecting patterns, trends, and insights that can drive business growth and efficiency.

With today's ever-increasing volume of data, a *robust* approach to data collection is crucial for ensuring accurate and meaningful results. This process requires following a comprehensive and systematic methodology designed to ensure the quality, reliability, and validity of data gathered for analysis. It involves identifying and sourcing relevant data from diverse sources, including internal databases, external repositories, websites, and user-generated information. And it requires meticulous planning and execution to guarantee the accuracy, comprehensiveness, and reliability of the collected data.

Preparing, or "wrangling," the collected data adequately prior to analysis is equally important. Preparation involves scrubbing, organizing, and transforming the data into a format suitable for analysis. Data preparation plays a pivotal role in detecting and resolving any inconsistencies or errors present in the data, thereby enabling accurate analysis. The rapidly advancing technology and widespread use of the internet have added complexity to the data collection and preparation processes. As a result, data analysts and organizations face many challenges, such as identifying relevant data sources, managing large data volumes, identifying outliers or erroneous data, and handling unstructured data. By mastering the art and science of collecting and preparing data, organizations can leverage valuable insights to drive informed decision-making and achieve

business success.

2.1 Overview of Data Collection Methods

Learning Outcomes

By the end of this section, you should be able to:

- 2.1.1 Define data collection and its role in data science.
- 2.1.2 Describe different data collection methods commonly used in data science, such as surveys and experiments.
- 2.1.3 Recognize scenarios where specific data collection methods are most appropriate.

Data collection refers to the systematic and well-organized process of gathering and accurately conveying important information and aspects related to a specific phenomenon or event. This involves using statistical tools and techniques to collect data, identify its attributes, and capture relevant contextual information. The gathered data is crucial for making sound interpretations and gaining meaningful insights. Additionally, it is important to take note of the environment and geographic location from where the data was obtained, as it can significantly influence the decision-making process and overall conclusions drawn from the data.

Data collection can be carried out through various methods, depending on the nature of the research or project and the type of data being collected. Some common methods for data collection include experiments, surveys, observation, focus groups, interviews, and document analysis.

This chapter will focus on the use of surveys and experiments to collect data. Social scientists, marketing specialists, and political analysts regularly use surveys to gather data on topics such as public opinion, customer satisfaction, and demographic information. Pharmaceutical companies heavily rely on experimental data from clinical trials to test the safety and efficacy of new drugs. This data is then used by their legal teams to gain regulatory approval and bring drugs to market.

Before collecting data, it is essential for a data scientist to have a clear understanding of the project's objectives, which involves identifying the research question or problem and defining the target population or sample. If a survey or experiment is used, the design of the survey/experiment is also a critical step, requiring careful consideration of the type of questions, response options, and overall structure. A survey may be conducted online, via phone, or in person, while experimental research requires a controlled environment to ensure data validity and reliability.

Types of Data

Observational and transactional data play important roles in data analysis and related decision-making, each offering unique insights into different aspects of real-world phenomena and business operations. **Observational data**, often used in qualitative research, is collected by systematically observing and recording behavior without the active participation of the researcher. **Transactional data** refers to any type of information related to transactions or interactions between individuals, businesses, or systems, and it is more often used in quantitative research.

Many fields of study use observational data for their research. Table 2.1 summarizes some examples of fields that rely on observational data, the type of data they collect, and the purpose of their data collection.

Field	Data Collected By	Purpose
Education	Teachers	To monitor and assess student behavior and learning progress in the classroom
Psychology	Therapists and psychologists	To gather information about their clients' behavior, thoughts, and emotions

Table 2.1 Fields Where Observation Methods Are Used

Field	Data Collected By	Purpose
Health care	Medical professionals	To diagnose and monitor patients' conditions and progress
Market research	Businesses	To gather information about consumer behavior and preferences to improve their products, services, and marketing strategies
Environmental science	Scientists	To gather data about the natural environment and track changes over time
Criminal investigations	Law enforcement officers	To gather evidence and information about criminal activity
Animal behavior	Zoologists	To study and understand the behavior of various animal species
Transportation planning	Urban planners and engineers	To collect data on traffic patterns and transportation usage to make informed decisions about infrastructure and transit systems

Table 2.1 Fields Where Observation Methods Are Used

Transactional data is collected by directly recording transactions that occur in a particular setting, such as a retail store or an online platform that allows for accurate and detailed information on actual consumer behavior. It can include financial data, but it also includes data related to customer purchases, website clicks, user interactions, or any other type of activity that is recorded and tracked.

Transactional data can be used to understand patterns and trends, make predictions and recommendations, and identify potential opportunities or areas for improvement. For example, the health care industry may focus on transactional data related to patient interactions with health care providers and facilities, such as appointments, treatments, and medications prescribed. The retail industry may use transactional data on customer purchases and product returns, while the transportation industry may analyze data related to ticket sales and passenger traffic.

While observational data provides detailed descriptions of behavior, transactional data provides numerical data for statistical analysis. There are strengths and limitations with each of these, and the examples in this chapter will make use of both types.

EXAMPLE 2.1

Problem

Ashley loves setting up a bird feeder in her backyard and watching the different types of birds that come to feed. She has always been curious about the typical number of birds that visit her feeder each day and has estimated the number based on the amount of food consumed. However, she has to visit her grandmother's house for three days and is worried about leaving the birds without enough food. In order to prepare the right amount of bird food for her absence, Ashley has decided to measure the total amount of feed eaten each day to determine the total amount of food needed for her three-day absence. Which method of data collection is best suited for Ashley's research on determining the total amount of food required for her three-day absence—observational or transactional? Provide a step-by-step explanation of the chosen method.

Solution

Ashley wants to ensure that there is enough food for her local birds while she is away for three days. To do this, she will carefully observe the feeder daily for two consecutive weeks. She will record the total amount of feed eaten each day and make sure to refill the feeder each morning before the observation. This will provide a consistent amount of food available for the birds. After two weeks, Ashley will use the total amount of food consumed and divide it by the number of days observed to estimate the required daily food. Then, she will multiply the daily food by three to determine the total amount of bird food needed for her three-day absence. By directly observing and recording the bird food, as well as collecting data for two weeks, Ashley will gather accurate and reliable information. This will help her confidently prepare the necessary amount of bird food for her feathered friends while she is away, thus ensuring that the birds are well-fed and taken care of during her absence.

EXAMPLE 2.2

Problem

A group of data scientists working for a large hospital have been tasked with analyzing their transactional data to identify areas for improvement. In the past year, the hospital has seen an increase in patient complaints about long wait times for appointments and difficulties scheduling follow-up visits. Samantha is one of the data scientists tasked to collect data in order to analyze these issues.

a. What methodology should be employed by Samantha to collect pertinent data for analyzing the recent surge in patient complaints regarding extended appointment wait times and difficulties in scheduling follow-up visits at the hospital?
b. What strategies could be used to analyze the data?

Solution

a. Explore the stored information as transactional data
b. Collecting transactional data for analysis can be achieved by utilizing various sources within the hospital setting. These sources include:

1. Electronic Health Records (EHRs): Samantha can gather data from the hospital's electronic health records system. This data may include patients' appointment schedules, visit durations, and wait times. This information can help identify patterns and trends in appointment scheduling and wait times.
2. Appointment Booking System: Samantha can gather data from the hospital's appointment booking system. This data can include appointment wait times, appointment types (e.g., primary care, specialist), and scheduling difficulties (e.g., appointment availability, cancellations). This information can help identify areas where the booking system may be causing delays or challenges for patients.
3. Hospital Call Center: Samantha can gather data from the hospital's call center, which is responsible for booking appointments over the phone. This data can include call wait times, call duration, and reasons for call escalations. This information can help identify areas for improvement in the call center's processes and procedures.
4. Historical Data: Samantha can analyze historical data, such as appointment wait times and scheduling patterns, to identify any changes that may have contributed to the recent increase in complaints. This data can also be compared to current data to track progress and improvements in wait times and scheduling.

Collecting Data Through Experiments

Collecting data through scientific experiments requires a well-designed experimental scheme, describing the research objectives, variables, and procedures. The establishment of a control specimen is crucial, and data is obtained through systematic properties, measurements, or characteristics. It is crucial to follow ethical guidelines for the proper documentation and ethical utilization of the collected data (see Ethics in Data Collection).

Consider this example: Scientist Sally aimed to investigate the impact of sunlight on plant growth. The research inquiry was to determine whether increased exposure to sunlight enhances the growth of plants. Sally experimented with two groups of plants wherein one group received eight hours of sunlight per day, while the other only received four hours. The height of each plant was measured and documented every week for four consecutive weeks. The main research objective was to determine the growth rate of plants exposed to eight hours of sunlight compared to those with only four hours. A total of 20 identical potted plants were used, with one group allocated to the "sunlight" condition and the other to the "limited sunlight" condition. Both groups were maintained under identical environmental conditions, including temperature, humidity, and soil moisture. Adequate watering was provided to ensure equal hydration of all plants. The measurements of plant height were obtained and accurately recorded every week. This approach allowed for the collection of precise and reliable data on the impact of sunlight on plant growth, which can serve as a valuable resource for further research and understanding of this relationship.

2.2 Survey Design and Implementation

Learning Outcomes

By the end of this section, you should be able to:

- 2.2.1 Describe the elements of survey design and identify the steps data scientists take to ensure the reliability of survey results.
- 2.2.2 Describe methods for avoiding bias in survey questions.
- 2.2.3 Describe various sampling techniques and the advantages of each.

Surveys are a common strategy for gathering data in a wide range of domains, including market research, social sciences, and education. Surveys collect information from a *sample* of individuals and often use questionnaires to collect data. **Sampling** is the process of selecting a subset of a larger population to represent and analyze information about that population.

Designing the Survey

The process of data collection through surveys is a crucial aspect of research—and one that requires careful planning and execution to gather accurate and reliable data. The first step, as stated earlier, is to clearly define the research objectives and determine the appropriate target population. This will help you structure the survey and identify the specific questions that need to be included.

Constructing good surveys is hard. A survey should begin with simple and easy-to-answer questions and progress to more complex or sensitive ones. This can help build a rapport with the respondents and increase their willingness to answer more difficult questions. Additionally, the researcher may consider mixing up the response options for multiple-choice questions to avoid response bias. To ensure the quality of the data collected, the survey questionnaire should undergo a pilot test with a small group of individuals from the target population. This allows the researcher to identify any potential issues or confusion with the questions and make necessary adjustments before administering the survey to the larger population.

Open-Ended Versus Closed-Ended Questions

Surveys should generally contain a mix of closed-ended and open-ended questions to gather both quantitative and qualitative data.

Open-ended questions allow for more in-depth responses and provide the opportunity for unexpected insights. They also allow respondents to elaborate on their thoughts and provide detailed and personal responses. **Closed-ended questions** have predetermined answer choices and are effective in gathering quantitative data. They are quick and easy to answer, and their clear and structured format allows for quantifiable results.

Avoiding Bias in Survey Questions

Unbiased sampling and unbiased survey methodology are essential for ensuring accurate and reliable results. One well-known real-life instance of sampling bias leading to inaccurate findings is the 1936 *Literary Digest* poll. This survey aimed to forecast the results of the US presidential election and utilized a mailing list of telephone and automobile owners. This approach was considered biased toward affluent individuals and therefore favored Republican voters. As a consequence, the poll predicted a victory for Republican nominee Alf Landon. However, the actual outcome was a landslide win for Franklin D. Roosevelt (Lusinchi, 2012). This discrepancy can be attributed to the biased sampling method as well as the use of primarily closed-ended questions, which may not have accurately captured the opinions of all voters.

An example of a biased survey question in a survey conducted by a shampoo company might be "Do you prefer our brand of shampoo over cheaper alternatives?" This question is biased because it assumes that the respondent prefers the company's brand over others. A more unbiased and accurate question would be "What factors do you consider when choosing a shampoo brand?" This allows for a more detailed and accurate response. The biased question could have led to inflated results in favor of the company's brand.

Sampling

The next step in the data collection process is to choose a participant *sample* to ideally represent the restaurant's customer base. Sampling could be achieved by randomly selecting customers, using customer databases, or targeting specific demographics, such as age or location.

Sampling is necessary in a wide range of data science projects to make data collection more manageable and cost-effective while still drawing meaningful conclusions. A variety of techniques can be employed to determine a subset of data from a larger population to perform research or construct hypotheses about the entire population. The choice of a sampling technique depends upon the nature and features of the population being studied as well as the objectives of the research. When using a survey, researchers must also consider the tool(s) that will be used for distributing the survey, such as through email, social media, or physically distributing questionnaires at the restaurant. It's crucial to make the survey easily accessible to the chosen sample to achieve a higher response rate.

A number of sampling techniques and their advantages are described below. The most frequently used among these are *simple random selection*, *stratified sampling*, *cluster sampling*, and *convenience sampling*.

1. **Simple random selection.** Simple random selection is a statistical technique used to pick a representative sample from a larger population. This process involves randomly choosing individuals or items from the population, ensuring that each selected member of the population has an identical chance of being contained in the sample. The main step in simple random selection is to define the population of interest and assign a unique identification number to each member. This could be done using a random number generator, a computer program designed to generate a sequence of random numbers, or a random number table, which lists numbers in a random sequence. The primary benefit of this technique is its ability to minimize bias and deliver a fair representation of the population.

   In the health care field, simple random sampling is utilized to select patients for medical trials or surveys, allowing for a diverse and unbiased sample (Elfil & Negida, 2017). Similarly, in finance, simple random sampling can be applied to gather data on consumer behavior and guide decision-making in financial institutions. In engineering, this technique is used to select random samples of materials or components

for quality control testing. In the political arena, simple random sampling is commonly used to select randomly registered voters for polls or surveys, ensuring equal representation and minimizing bias in the data collected.

2. **Stratified sampling.** Stratified sampling involves splitting the population into subgroups based on specified factors, such as age, area, income, or education level, and taking a random sample from each stratum in proportion to its size in the population. Stratified sampling allows for a more accurate representation of the population as it ensures that all subgroups are adequately represented in the sample. This can be especially useful when the variables being studied vary significantly between the stratified groups.
3. **Cluster sampling.** With cluster sampling, the population is divided into natural groups or clusters, such as schools, communities, or cities, with a random sample of these clusters picked and all members within the chosen clusters included in the sample. Cluster sampling is helpful to represent the entire population even if it is difficult or time-consuming due to challenges such as identifying clusters, sourcing a list of clusters, traveling to different clusters, and communicating with them. Additionally, data analysis and sample size calculation may be more complex, and there is a risk of bias in the sample. However, cluster sampling can be more cost-effective.

   An example of cluster sampling would be a study on the effectiveness of a new educational program in a state. The state is divided into clusters based on school districts. The researcher uses a random selection process to choose a sample of school districts and then collects data from all the schools within those districts. This method allows the researcher to obtain a representative sample of the state's student population without having to visit each individual school, saving time and resources.
4. **Convenience sampling.** Convenience sampling applies to selecting people or items for the sample based on their availability and convenience to the data science research. For example, a researcher may choose to survey students in their classroom or manipulate data from social media users. Convenience sampling is effortless to achieve, and it is useful for exploratory studies. However, it may not provide a representative sample as it is prone to selection bias in that individuals who are more readily available or willing to participate may be overrepresented.

   An example of convenience sampling would be conducting a survey about a new grocery store in a busy shopping mall. A researcher stands in front of the store and approaches people who are coming out of the store to ask them about their shopping experience. The researcher only includes responses from those who agreed to participate, resulting in a sample that is convenient but may not be representative of the entire population of shoppers in the mall.
5. **Systematic sampling.** Systematic sampling is based on starting at a random location in the dataset and then selecting every nth member from a population to be contained in the sample. This process is straightforward to implement, and it provides a representative sample when the population is randomly distributed. However, if there is a pattern in the **sampling frame** (the organizing structure that represents the population from which a sample is drawn), it may lead to a biased sample.

   Suppose a researcher wants to study the dietary habits of students in a high school. The researcher has a list of all the students enrolled in the school, which is approximately 1,000 students. Instead of randomly selecting a sample of students, the researcher decides to use systematic sampling. The researcher first assigns a number to each student, going from 1 to 1,000. Then, the researcher randomly selects a number from 1 to 10—let's say they select 4. This number will be the starting point for selecting the sample of students. The researcher will then select every 10th student from the list, which means every student with a number ending in 4 (14, 24, 34, etc.) will be included in the sample. This way, the researcher will have a representative sample of 100 students from the high school, which is 10% of the population. The sample will consist of students from different grades, genders, and backgrounds, making it a diverse and representative sample.

6. **Purposive sampling.** With purposive sampling, one or more specific criteria are used to select participants who are likely to provide the most relevant and useful information for the research study. This can involve selecting participants based on their expertise, characteristics, experiences, or behaviors that are relevant to the research question.

   For example, if a researcher is conducting a study on the effects of exercise on mental health, they may use purposive sampling to select participants who have a strong interest or experience in physical fitness and have a history of mental health issues. This sampling technique allows the researcher to target a specific population that is most relevant to the research question, making the findings more applicable and generalizable to that particular group. The main advantage of purposive sampling is that it can save time and resources by focusing on individuals who are most likely to provide valuable insights and information. However, researchers need to be transparent about their sampling strategy and potential biases that may arise from purposely selecting certain individuals.
7. **Snowball sampling.** Snowball sampling is typically used in situations where it is difficult to access a particular population; it relies on the assumption that people with similar characteristics or experiences tend to associate with each other and can provide valuable referrals. This type of sampling can be useful in studying hard-to-reach or sensitive populations, but it may also be biased and limit the generalizability of findings.
8. **Quota sampling.** Quota sampling is a non-probability sampling technique in which experimenters select participants based on predetermined quotas to guarantee that a certain number or percentage of the population of interest is represented in the sample. These quotas are based on specific demographic characteristics, such as age, gender, ethnicity, and occupation, which are believed to have a direct or indirect relationship with the research topic. Quota sampling is generally used in market research and opinion polls, as it allows for a fast and cost-effective way to gather data from a diverse range of individuals. However, it is important to note that the results of quota sampling may not accurately represent the entire population, as the sample is not randomly selected and may be biased toward certain characteristics. Therefore, the findings from studies using quota sampling should be interpreted with caution.
9. **Volunteer sampling.** Volunteer sampling refers to the fact that the participants are not picked at random by the researcher, but instead volunteer themselves to be a part of the study. This type of sampling is commonly used in studies that involve recruiting participants from a specific population, such as a specific community or organization. It is also often used in studies where convenience and accessibility are important factors, as participants may be more likely to volunteer if the study is easily accessible to them. Volunteer sampling is not considered a random or representative sampling technique, as the participants may not accurately represent the larger population. Therefore, the results obtained from volunteer sampling may not be generalizable to the entire population.

Sampling Error

Sampling error is the difference between the results obtained from a sample and the true value of the population parameter it is intended to represent. It is caused by chance and is inherent in any sampling method. The goal of researchers is to minimize sampling errors and increase the accuracy of the results. To avoid sampling error, researchers can increase sample size, use probability sampling methods, control for extraneous variables, use multiple modes of data collection, and pay careful attention to question formulation.

Sampling Bias

Sampling bias occurs when the sample used in a study isn't representative of the population it intends to generalize to, leading to skewed or inaccurate conclusions. This bias can take many forms, such as *selection bias*, where certain groups are systematically over- or underrepresented, or *volunteer bias*, where only a specific subset of the population participates. Researchers use the sampling techniques summarized earlier to avoid sampling bias and ensure that each member of the population has an equal chance of being included in

the sample. Additionally, careful consideration of the sampling frame should ideally encompass all members of the target population and provide a clear and accessible way to identify and select individuals or units for inclusion in the sample. Sampling bias can occur at various stages of the sampling process, and it can greatly impact the accuracy and validity of research findings.

Measurement Error

Measurement errors are inaccuracies or discrepancies that surface during the process of collecting, recording, or analyzing data. They may occur due to human error, environmental factors, or inherent inconsistencies in the phenomena being studied. *Random error*, which arises unpredictably, can affect the precision of measurements, and *systematic error* may consistently bias measurements in a particular direction. In data analysis, addressing measurement error is crucial for ensuring the reliability and validity of results. Techniques for mitigating measurement error include improving data collection methods, calibrating instruments, conducting validation studies, and employing statistical methods like error modeling or sensitivity analysis to account for and minimize the impact of measurement inaccuracies on the analysis outcomes.

A Sampling Case Study

Consider a research study that wants to randomly select a group of college students from a larger population to examine the effects of exercise on their mental health outcomes. Using student ID numbers generated by a computer program, 100 participants from the larger population were randomly selected to participate in the study to achieve the desired accuracy. This process ensured that every student in the university had an equal chance of being selected to participate. The participants were then randomly assigned to either the exercise group or the control group. This method of random sampling ensures that the sample is representative of the larger population, providing a more accurate representation of the relationship between exercise and mental health outcomes for college students.

Types of sampling error that could occur in this study include the following:

1. **Sampling bias.** One potential source of bias in this study is *self-selection bias*. As the participants are all college students, they may not be representative of the larger population, as college students tend to have more access and motivation to exercise compared to the general population. This could limit the generalizability of the study's findings. In addition, if the researchers only recruit participants from one university, there may be *under-coverage bias*. This means that certain groups of individuals, such as nonstudents or students from other universities, may be excluded from the study, potentially leading to biased results.
2. **Measurement error.** Measurement errors could occur, particularly if the researchers are measuring the participants' exercise and mental health outcomes through self-report measures. Participants may not accurately report their exercise habits or mental health symptoms, leading to inaccurate data.
3. **Non-response bias.** Some participants in the study may choose not to participate or may drop out before the study is completed. This could introduce non-response bias, as those who choose not to participate or drop out may differ from those who remain in the study in terms of their exercise habits or mental health outcomes.
4. **Sampling variability.** The sample of 100 participants is a relatively small subset of the larger population. As a result, there may be sampling variability, meaning that the characteristics and outcomes of the participants may differ from those of the larger population simply due to chance.
5. **Sampling error in random assignment.** In this study, the researchers randomly assign participants to either the exercise group or the control group. However, there is always a possibility of sampling error in the random assignment process, meaning that the groups may not be perfectly balanced in terms of their exercise habits or other characteristics.

   These types of sampling errors can affect the accuracy and generalizability of the study's findings.

Researchers need to be aware of these potential errors and take steps to minimize them when designing and conducting their studies.

EXAMPLE 2.3

Problem

Mark is a data scientist who works for a marketing research company. He has been tasked to lead a study to understand consumer behavior toward a new product that is about to be launched in the market. As data scientists, they know the importance of using the right sampling technique to collect accurate and reliable data. Mark divided the population into different groups based on factors such as age, education, and income. This ensures that he gets a representative sample from each group, providing a more accurate understanding of consumer behavior. What is the name of the sampling technique used by Mark to ensure a representative sample from different groups of consumers for his study on consumer behavior toward a new product?

Solution

The sampling technique used by Mark is called stratified sampling. This involves dividing the population into subgroups or strata based on certain characteristics and then randomly selecting participants from each subgroup. This ensures that each subgroup is represented in the sample, providing a more accurate representation of the entire population. This type of sampling is often used in market research studies to get a more comprehensive understanding of consumer behavior and preferences. By using stratified sampling, Mark can make more reliable conclusions and recommendations for the new product launch based on the data he collects.

2.3 Web Scraping and Social Media Data Collection

Learning Outcomes:

By the end of this section, you should be able to:

- 2.3.1 Discuss the uses of web scraping for collecting and preparing data for analysis.
- 2.3.2 Apply regular expressions for data manipulation and pattern matching.
- 2.3.3 Write Python code to scrape data from the web.
- 2.3.4 Apply various methods for parsing, extracting, processing, and storing data.

Web scraping and social media data collection are two approaches used to gather data from the internet. **Web scraping** involves pulling information and data from websites using a web data extraction tool, often known as a web scraper. One example would be a travel company looking to gather information about hotel prices and availability from different booking websites. Web scraping can be used to automatically gather this data from the various websites and create a comprehensive list for the company to use in its business strategy without the need for manual work.

Social media data collection involves gathering information from various platforms like Twitter and Instagram using application programming interface or monitoring tools. An **application programming interface (API)** is a set of protocols, tools, and definitions for building software applications allowing different software systems to communicate and interact with each other and enabling developers to access data and services from other applications, operating systems, or platforms. Both web scraping and social media data collection require determining the data to be collected and analyzing it for accuracy and relevance.

Web Scraping

There are several techniques and approaches for scraping data from websites. See Table 2.2 for some of the

common techniques used. (Note: The techniques used for web scraping will vary depending on the website and the type of data being collected. It may require a combination of different techniques to effectively scrape data from a website.)

Web Scraping Technique	Details
Web Crawling	• Follows links on a web page to navigate to other pages and collect data from them • Useful for scraping data from multiple pages of a website
XPath	• Powerful query language • Navigates through the elements in an HTML document • Often used in combination with HTML parsing to select specific elements to scrape
Regular Expressions	• Search for and extract specific patterns of text from a web page • Useful for scraping data that follows a particular format, such as dates, phone numbers, or email addresses
HTML Parsing	• Analyzes the HTML (HyperText Markup Language) structure of a web page and identifies the specific tags and elements that contain the desired data • Often used for simple scraping tasks
Application Programming Interfaces (APIs)	• Authorize developers to access and retrieve data instantly without the need for web scraping • Often a more efficient and reliable method for data collection
(XML) API Subset	• XML (Extensible Markup Language) is another markup language used exchanging data • This method works similarly to using the HTML API subset by making HTTP requests to the website's API endpoints and then parsing the data received in XML format
(JSON) API Subset	• JSON (JavaScript Object Notation) is a lightweight data interchange format that is commonly used for sending and receiving data between servers and web applications • Many websites provide APIs in the form of JSON, making it another efficient method for scraping data

Table 2.2 Techniques and Approaches for Scraping Data from Websites

Social Media Data Collection

Social media data collection can be carried out through various methods such as API integration, social listening, social media surveys, network analysis, and image and video analysis. APIs provided by social media platforms allow data scientists to collect structured data on user interactions and content. **Social listening** involves monitoring online conversations for insights on customer behavior and trends. Surveys conducted on social media can provide information on customer preferences and opinions. **Network analysis**, or the examination of relationships and connections between users, data, or entities within a network, can reveal influential users and communities. It involves identifying and analyzing influential individuals or groups as well as understanding patterns and trends within the network. Image and video analysis can provide insights into visual trends and user behavior.

An example of social media data collection is conducting a Twitter survey on customer satisfaction for a food delivery company. Data scientists can use Twitter's API to collect tweets containing specific hashtags related to the company and analyze them to understand customers' opinions and preferences. They can also use social listening to monitor conversations and identify trends in customer behavior. Additionally, creating a social media survey on Twitter can provide more targeted insights into customer satisfaction and preferences. This data can then be analyzed using data science techniques to identify key areas for improvement and drive informed business decisions.

Using Python to Scrape Data from the Web

As noted previously, web scraping is a strategy of gathering data from the internet using automated mechanisms or programs. Python is one of the popular programming languages used for web scraping due to its various libraries and frameworks that make it easy to pull and process data from websites.

To scrape data such as a table from a website using Python, we follow these steps:

1. **Import the `pandas` library.** The first step is to import the pandas library (https://openstax.org/r/pandas), which is a popular Python library for data analysis and manipulation.

```
import pandas as pd
```

2. **Use the `read_html()` function.** This function is used to read HTML tables from a web page and convert them into a list of DataFrame objects. Recall from What Are Data and Data Science? that a **DataFrame** is a data type that `pandas` uses to store multi-column tabular data.

```
df = pd.read_html("https://.....
```

3. **Access the desired data.** If the data on the web page is divided into different tables, we need to specify which table we want to extract. We have used indexing to access the desired table (for example: index 4) from the list of DataFrame objects returned by the `read_html()` function. The index here represents the table order in the web page.
4. **Store the data in a DataFrame.** The result of the `read_html()` function is a list of DataFrame objects, and each DataFrame represents a table from the web page. We can store the desired data in a DataFrame variable for further analysis and manipulation.
5. **Display the DataFrame.** By accessing the DataFrame variable, we can see the extracted data in a tabular format.
6. **Convert strings to numbers.** As noted in Chapter 1, a **string** is a data type used to represent a sequence of characters, such as letters, numbers, and symbols that are enclosed by matching single (') or double (") quotes. If the data in the table is in string format and we want to perform any numerical operations on it, we need to convert the data to numerical format. We can use the `to_numeric()` function from `pandas` to convert strings to numbers and then store the result in a new column in the DataFrame.

```
df['column_name'] = pd.to_numeric(df['column_name'])
```

This will create a new column in the DataFrame with the converted numerical values, which can then be used for further analysis or visualization.

In computer programming, indexing usually starts from 0. This is because most programming languages use 0 as the initial index for arrays, matrices, or other data structures. This convention has been adopted to simplify the implementation of some algorithms and to make it easier for programmers to access and manipulate data. Additionally, it aligns with the way computers store and access data in memory. In the context of parsing tables from HTML pages, using 0 as the initial index allows programmers to easily access and manipulate data from different tables on the same web page. This enables efficient data processing and analysis, making the

task more manageable and streamlined.

EXAMPLE 2.4

Problem

Extract data table "Current Population Survey: Household Data: (Table A-13). Employed and unemployed persons by occupation, Not seasonally adjusted" from the FRED (Federal Reserve Economic Data) (https://openstax.org/r/fred) website in the link (https://fred.stlouisfed.org/release/tables?rid=50&eid=3149#snid=4498 (https://openstax.org/r/stlouisfed)) using Python code. The data in this table provides a representation of the overall employment and unemployment situation in the United States. The table is organized into two main sections: employed persons and unemployed persons.

Solution

PYTHON CODE

```python
# Import pandas
import pandas as pd

# Read data from the URL
df_list = pd.read_html(
"https://fred.stlouisfed.org/release/tables?rid=50&eid=3149#snid=4498")

# Since pd.read_html() returns a list of DataFrames, select the first DataFrame
df = df_list[0]

# Print the first 5 rows of the DataFrame
print(df.head(5))
```

The resulting output will look like this:

```
   Unnamed: 0                                                Name  \
0         NaN                                   Monthly, Employed
1         NaN                              Total, 16 years and over
2         NaN  Management, professional, and related occupations
3         NaN  Management, business, and financial operations...
4         NaN                 Professional and related occupations

                        May 2024                        Apr 2024  \
0                            NaN                             NaN
1  161,341  Thousands of Persons  161,590  Thousands of Persons
2   70,897  Thousands of Persons   70,548  Thousands of Persons
3   30,910  Thousands of Persons   30,172  Thousands of Persons
4   39,987  Thousands of Persons   40,376  Thousands of Persons

                        May 2023              Units
0                            NaN                NaN
1  161,002  Thousands of Persons  Thous. of Persons
2   70,388  Thousands of Persons  Thous. of Persons
3   30,830  Thousands of Persons  Thous. of Persons
4   39,557  Thousands of Persons  Thous. of Persons
```

In Python, there are several libraries and methods that can be used for parsing and extracting data from text. These include the following:

1. Regular expressions (regex or RE) (https://openstax.org/r/docpython). This is a built-in library in Python that allows for pattern matching and extraction of data from strings. It uses a specific syntax to define patterns and rules for data extraction.
2. Beautiful Soup (https://openstax.org/r/pypi). This is an external library that is mostly used for scraping and parsing HTML and XML code. It can be utilized to extract specific data from web pages or documents.
3. Natural Language Toolkit (NLTK) (https://openstax.org/r/nltk). This is a powerful library for natural language processing in Python. It provides various tools for tokenizing, parsing, and extracting data from text data. (**Tokenizing** is the process of breaking down a piece of text or string of characters into smaller units called tokens, which can be words, phrases, symbols, or individual characters.)
4. TextBlob (https://openstax.org/r/textblob). This library provides a simple interface for most natural language processing assignments, such as argument and part-of-speech tagging. It can also be utilized for parsing and extracting data from text.
5. SpaCy (https://openstax.org/r/spacy). This is a popular open-source library for natural language processing. It provides efficient methods for tokenizing, parsing, and extracting data from text data.

Overall, the library or method used for parsing and extracting data will depend on the specific task and type of data being analyzed. It is important to research and determine the best approach for a given project.

Regular Expressions in Python

Regular expressions, also known as **regex**, are a set of symbols used to define a search pattern in text data. In Python, these expressions are supported by the `re` module (function), and their syntax is similar to other programming languages. The use of regular expressions offers researchers a robust method for identifying and manipulating patterns in text. With this powerful tool, specific words, characters, or patterns of characters

can be searched and matched. Typical applications include data parsing, input validation, and extracting targeted information from larger text sources. Common use cases in Python involve recognizing various types of data, such as dates, email addresses, phone numbers, and URLs, within extensive text files. Moreover, regular expressions are valuable for tasks like data cleaning and text processing. Despite their versatility, regular expressions can be elaborate, allowing for advanced search patterns utilizing meta-characters like `*`, `?`, and `+`. However, working with these expressions can present challenges, as they require a thorough understanding and careful debugging to ensure successful implementation.

USING META-CHARACTERS IN REGULAR EXPRESSIONS

- The `*` character is known as the “star” or “asterisk” and is used to match zero or more occurrences of the preceding character or group in a regular expression. For example, the regular expression `"a*"` would match an `"a"` followed by any number (including zero) of additional `"a"`s, such as `"a"`, `"aa"`, `"aaa"`, etc.
- The `?` character is known as the "question mark" and is used to indicate that the preceding character or group is optional. It matches either zero or one occurrences of the preceding character or group. For example, the regular expression `"a?b"` would match either `"ab"` or `"b"`.
- The `+` character is known as the "plus sign" and is used to match one or more occurrences of the preceding character or group. For example, the regular expression `"a+b"` would match one or more `"a"`s followed by a `"b"`, such as `"ab"`, `"aab"`, `"aaab"`, etc. If there are no `"a"`s, the match will fail. This is different from the `*` character, which would match zero or more `"a"`s followed by a `"b"`, allowing for a possible match without any `"a"`s.

EXAMPLE 2.5

Problem

Write Python code using regular expressions to search for a selected word “Python” in a given string and print the number of times it appears.

Solution

PYTHON CODE

```
## import the regular expression module
import re
# create a string with story problem
story = "Samantha is a sixth-grade student who uses the popular coding language
Python to collect and analyze weather data for science project. She creates a
program that collects data from an online weather API and stores it in a CSV file
for 10 days. With the help of her teacher, she uses Python to visualize the data
and discovers patterns in temperature, humidity, and precipitation."
# create a regex pattern to match a selected word
pattern = "Python"
```

```
# use the "findall" to search for matching patterns in the data string
words = re.findall(pattern, story)

# print the number of repeated python
print ("The word 'python' is repeated", len(words), "times in the story.")

# print the matched patterns
print(words) # output: ['Python']
```

The resulting output will look like this:

```
The word 'python' is repeated 2 times in the story.
['Python', 'Python']
```

In Python, "`len`" is short for "length" and is used to determine the number of items in a collection. It returns the total count of items in the collection, including spaces and punctuation marks in a string.

Parsing and Extracting Data

Splitting and slicing are two methods used to manipulate text strings in programming. **Splitting a string** means dividing a text string into smaller parts or substrings based on a specified separator. The separator can be a character, string, or regular expression. This can be useful for separating words, phrases, or data values within a larger string. For example, the string `"Data Science"` can be split into two substrings `"Data"` and `"Science"` by using a space as the separator.

Slicing a string refers to extracting a portion or section of a string based on a specified range of indices. An index refers to the position of a character in a string, starting from 0 for the first character. The range specifies the start and end indices for the slice, and the resulting substring includes all characters within that range. For example, the string `"Data Science"` can be sliced to extract `"Data"` by specifying the range from index 0 to 4, which includes the first four characters. Slicing can also be used to manipulate strings by replacing, deleting, or inserting new content into specific positions within the string.

Parsing and extracting data involves the analysis of a given dataset or string to extract specific pieces of information. This is accomplished using various techniques and functions, such as splitting and slicing strings, which allow for the structured retrieval of data. This process is particularly valuable when working with large and complex datasets, as it provides a more efficient means of locating desired data compared to traditional search methods. Note that parsing and extracting data differs from the use of regular expressions, as regular expressions serve as a specialized tool for pattern matching and text manipulation. In contrast, parsing and data extraction offers a comprehensive approach to identifying and extracting specific data within a dataset.

Parsing and extracting data using Python involves using the programming language to locate and extract specific information from a given text. This is achieved by utilizing the `re` library, which enables the use of regular expressions to identify and retrieve data based on defined patterns. This process can be demonstrated through an example of extracting data related to a person purchasing an iPhone at an Apple store.

The code in the following Python feature box uses regular expressions (regex) to match and extract specific data from a string. The string is a paragraph containing information about a person purchasing a new phone

from the Apple store. The objective is to extract the product name, model, and price of the phone. First, the code starts by importing the necessary library for using regular expressions. Then, the string data is defined as a variable. Next, regex is used to search for specific patterns in the string. The first pattern searches for the words `"product: "` and captures anything that comes after it until it reaches a comma. The result is then stored in a variable named `"product"`. Similarly, the second pattern looks for the words `"model: "` and captures anything that comes after it until it reaches a comma. The result is saved in a variable named `"model"`. Finally, the third pattern searches for the words `"price: "` and captures any sequence of numbers or symbols that follows it until the end of the string. The result is saved in a variable named `"price"`. After all the data is extracted, it is printed out to the screen, using concatenation to add appropriate labels before each variable.

This application of Python code demonstrates the effective use of regex and the `re` library to parse and extract specific data from a given data. By using this method, the desired information can be easily located and retrieved for further analysis or use.

PYTHON CODE

```
## import necessary library
import re
# Define data to be parsed
data = "Samantha went to the Apple store to purchase a new phone. She was
specifically looking for the latest and most expensive model available. As she
looked at the different options, she came across the product: iPhone 12, the
product name caught her attention, as it was the newest version on the market. She
then noticed the model: A2172, which confirmed that this was indeed the latest and
most expensive model she was looking for. The price made her hesitate for a moment,
but she decided that it was worth it price: $799. She purchased the iPhone 12 and
was excited to show off her new phone to her friends."

# Use regex to match and extract data based on specific pattern
product = re.search(r"product: (.+?),", data).group(1)
model = re.search(r"model: (.+?),", data).group(1)
price = re.search(r"price: (.+?.+?.+?.+?)", data).group(1)

# Print the extracted data
print("product: " + product)
print("model: " + model)
print("price: " + price)
```

The resulting output will look like this:

```
product: iPhone 12 model: A2172 price: $799
```

Processing and Storing Data

Once the data is collected, it should be processed and stored in a suitable format for further analysis. This is where data processing and storage come into play. **Data processing** manages the raw data first by cleaning it through the removal of irrelevant information and then by transforming it into a structured format. The

cleaning process includes identifying and correcting any errors, inconsistencies, and missing values in a dataset and is essential for ensuring that the data is accurate, reliable, and usable for analysis or other purposes. Python is often utilized for data processing due to its flexibility and ease of use, and it offers a wide range of tools and libraries specifically designed for data processing. Once the data is processed, it needs to be stored for future use. (We will cover data storage in Data Cleaning and Preprocessing.) Python has several libraries that allow for efficient storage and manipulation of data in the form of DataFrames.

One method of storing data using Python is using the `pandas` library to create a DataFrame and then using the `to_csv()` function to save the DataFrame as a CSV (comma-separated values) file. This file can then be easily opened and accessed for future analysis or visualization. For example, the code in the following Python sidebar is a Python script that creates a *dictionary* with data about the presidents of the United States (https://openstax.org/r/wikipedia), including their ordered number and state of birth. A **data dictionary** is a data structure that stores data in key-value pairs, allowing for efficient retrieval of data using its key. It then uses the built-in CSV library (https://openstax.org/r/librarycsv) to create a CSV file and write the data to it. This code is used to store the US presidents' data in a structured format for future use, analysis, or display.

PYTHON CODE

```python
import csv

# Create a dictionary to store the data
presidents = {
    "1": ["George Washington", "Virginia"],
    "2": ["John Adams", "Massachusetts"],
    "3": ["Thomas Jefferson", "Virginia"],
    "4": ["James Madison", "Virginia"],
    "5": ["James Monroe", "Virginia"],
    "6": ["John Quincy Adams", "Massachusetts"],
    "7": ["Andrew Jackson", "South Carolina"],
    "8": ["Martin Van Buren", "New York"],
    "9": ["William Henry Harrison", "Virginia"],
    "10": ["John Tyler", "Virginia"],
    "11": ["James K. Polk", "North Carolina"],
    "12": ["Zachary Taylor", "Virginia"],
    "13": ["Millard Fillmore", "New York"],
    "14": ["Franklin Pierce", "New Hampshire"],
    "15": ["James Buchanan", "Pennsylvania"],
    "16": ["Abraham Lincoln", "Kentucky"],
    "17": ["Andrew Johnson", "North Carolina"],
    "18": ["Ulysses S. Grant", "Ohio"],
    "19": ["Rutherford B. Hayes", "Ohio"],
    "20": ["James A. Garfield", "Ohio"],
    "21": ["Chester A. Arthur", "Vermont"],
    "22": ["Grover Cleveland", "New Jersey"],
    "23": ["Benjamin Harrison", "Ohio"],
    "24": ["Grover Cleveland", "New Jersey"],
    "25": ["William McKinley", "Ohio"],
```

```
    "26": ["Theodore Roosevelt", "New York"],
    "27": ["William Howard Taft", "Ohio"],
    "28": ["Woodrow Wilson", "Virginia"],
    "29": ["Warren G. Harding", "Ohio"],
    "30": ["Calvin Coolidge", "Vermont"],
    "31": ["Herbert Hoover", "Iowa"],
    "32": ["Franklin D. Roosevelt", "New York"],
    "33": ["Harry S. Truman", "Missouri"],
    "34": ["Dwight D. Eisenhower", "Texas"],
    "35": ["John F. Kennedy", "Massachusetts"],
    "36": ["Lyndon B. Johnson", "Texas"],
    "37": ["Richard Nixon", "California"],
    "38": ["Gerald Ford", "Nebraska"],
    "39": ["Jimmy Carter", "Georgia"],
    "40": ["Ronald Reagan", "Illinois"],
    "41": ["George H. W. Bush", "Massachusetts"],
    "42": ["Bill Clinton", "Arkansas"],
    "43": ["George W. Bush", "Connecticut"],
    "44": ["Barack Obama", "Hawaii"],
    "45": ["Donald Trump", "New York"],
    "46": ["Joe Biden", "Pennsylvania"]
}

# Open a new CSV file in write mode
with open("presidents.csv", "w", newline='') as csv_file:
    # Specify the fieldnames for the columns
    fieldnames = ["Number", "Name", "State of Birth"]
    # Create a writer object
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    # Write the header row
    writer.writeheader()
    # Loop through the presidents dictionary
    for key, value in presidents.items():
        # Write the data for each president to the CSV file
        writer.writerow({
            "Number": key,
            "Name": value[0],
            "State of Birth": value[1]
            })
# Print a success message
print("Data successfully stored in CSV file.")
```

The resulting output will look like this:

```
Data successfully stored in CSV file.
```

2.4 Data Cleaning and Preprocessing

Learning Outcomes

By the end of this section, you should be able to:

- 2.4.1 Apply methods to deal with missing data and outliers.
- 2.4.2 Explain data standardization techniques, such as normalization, transformation, and aggregation.
- 2.4.3 Identify sources of noise in data and apply various data preprocessing methods to reduce noise.

Data cleaning and preprocessing is an important stage in any data science task. It refers to the technique of organizing and converting raw data into usable structures for further analysis. It involves extracting irrelevant or duplicate data, handling missing values, and correcting errors or inconsistencies. This ensures that the data is accurate, comprehensive, and ready for analysis. Data cleaning and preprocessing typically involve the following steps:

1. **Data integration. Data integration** refers to merging data from multiple sources into a single dataset.
2. **Data cleaning.** In this step, data is assessed for any errors or inconsistencies, and appropriate actions are taken to correct them. This may include removing duplicate values, handling missing data, and correcting formatting misconceptions.
3. **Data transformation.** This step prepares the data for the next step by transforming the data into a format that is suitable for further analysis. This may involve converting data types, **scaling** or normalizing numerical data, or encoding categorical variables.
4. **Data reduction.** If the dataset contains a large number of columns or features, data reduction techniques may be used to select only the most appropriate ones for analysis.
5. **Data discretization. Data discretization** involves grouping continuous data into categories or ranges, which can help facilitate analysis.
6. **Data sampling.** In some cases, the data may be too large to analyze in its entirety. In such cases, a sample of the data can be taken for analysis while still maintaining the overall characteristics of the original dataset.

The goal of data cleaning and preprocessing is to guarantee that the data used for analysis is accurate, consistent, and relevant. It helps to improve the quality of the results and increase the efficiency of the analysis process. A well-prepared dataset can lead to more accurate insights and better decision-making.

Handling Missing Data and Outliers

Missing data refers to any data points or values that are not present in a dataset. This could be due to data collection errors, data corruption, or nonresponse from participants in a study. Missing data can impact the accuracy and validity of an analysis, as it reduces the sample size and potentially introduces bias.

Some specific examples of missing data include the following:

1. A survey participant forgetting to answer a question
2. A malfunction in data collection equipment resulting in missing values
3. A participant choosing not to answer a question due to sensitivity or discomfort

An **outlier** is a data point that differs significantly from other data points in a given dataset. This can be due to human error, measurement error, or a true outlier value in the data. Outliers can skew statistical analysis and bias results, which is why it is important to identify and handle them properly before analysis.

Missing data and outliers are common problems that can affect the accuracy and reliability of results. It is important to identify and handle these issues properly to ensure the integrity of the data and the validity of the analysis. You will find more details about outliers in Measures of Center, but here we summarize the

measures typically used to handle missing data and outliers in a data science project:

1. **Identify the missing data and outliers.** The first stage is to identify which data points are missing or appear to be outliers. This can be done through visualization techniques, such as scatterplots, box plots, IQR (interquartile range), or histograms, or through statistical methods, such as calculating the mean, median, and standard deviation (see Measures of Center and Measures of Variation as well as Encoding Univariate Data).

   It is important to distinguish between different types of missing data. **MCAR (missing completely at random) data** is missing data not related to any other variables, with no underlying cause for its absence. Consider data collection with a survey asking about driving habits. One of the demographic questions asks for income level. Some respondents accidentally skip this question, and so there is missing data for income, but this is not related to the variables being collected related to driving habits.

   MAR (missing at random) data is missing data related to other variables but not to any unknown or unmeasured variables. As an example, during data collection, a survey is sent to respondents and the survey asks about loneliness. One of the questions asks about memory retention. Some older respondents might skip this question since they may be unwilling to share this type of information. The likelihood of missing data for loneliness factors is related to age (older respondents). Thus, the missing data is related to an observed variable of age but not directly related to loneliness measurements.

   MNAR (missing not at random) data refers to a situation in which the absence of data depends on observed data but not on unobserved data. For example, during data collection, a survey is sent to respondents and the survey asks about debt levels. One of the questions asks about outstanding debt that the customers have such as credit card debt. Some respondents with high credit card debt are less likely to respond to certain questions. Here the missing credit card information is related to the unobserved debt levels.
2. **Determine the reasons behind missing data and outliers.** It is helpful to understand the reasons behind the missing data and outliers. Some expected reasons for missing data include measurement errors, human error, or data not being collected for a particular variable. Similarly, outliers can be caused by incorrect data entry, measurement errors, or genuine extreme values in the data.
3. **Determine how to solve missing data issues.** Several approaches can be utilized to handle missing data. One option is to withdraw the missing data altogether, but this can lead to a loss of important information. Other methods include **imputation**, where the absent values are replaced with estimated values based on the remaining data or using predictive models to fill in the missing values.
4. **Consider the influence of outliers.** Outliers can greatly affect the results of the analysis, so it is important to carefully consider their impact. One approach is to delete the outliers from the dataset, but this can also lead to a loss of valuable information. Another option is to deal with the outliers as an individual group and analyze them separately from the rest of the data.
5. **Use robust statistical methods.** When dealing with outliers, it is important to use statistical methods that are not affected by extreme values. This includes using median instead of mean and using nonparametric tests instead of parametric tests, as explained in Statistical Inference and Confidence Intervals.
6. **Validate the results.** After handling the missing data and outliers, it is important to validate the results to ensure that they are robust and accurate. This can be done through various methods, such as cross-validation or comparing the results to external data sources.

Handling missing data and outliers in a data science task requires careful consideration and appropriate methods. It is important to understand the reasons behind these issues and to carefully document the process to ensure the validity of the results.

EXAMPLE 2.6

Problem

Starting in 1939, the United States Bureau of Labor Statistics tracked employment on a monthly basis. The number of employers in the construction field between 1939 and 2019 is presented in Figure 2.2.

a. Determine if there is any outlier in the dataset that deviates significantly from the overall trend.
b. In the event that the outlier is not a reflection of real employment numbers, how would you handle the outliers?

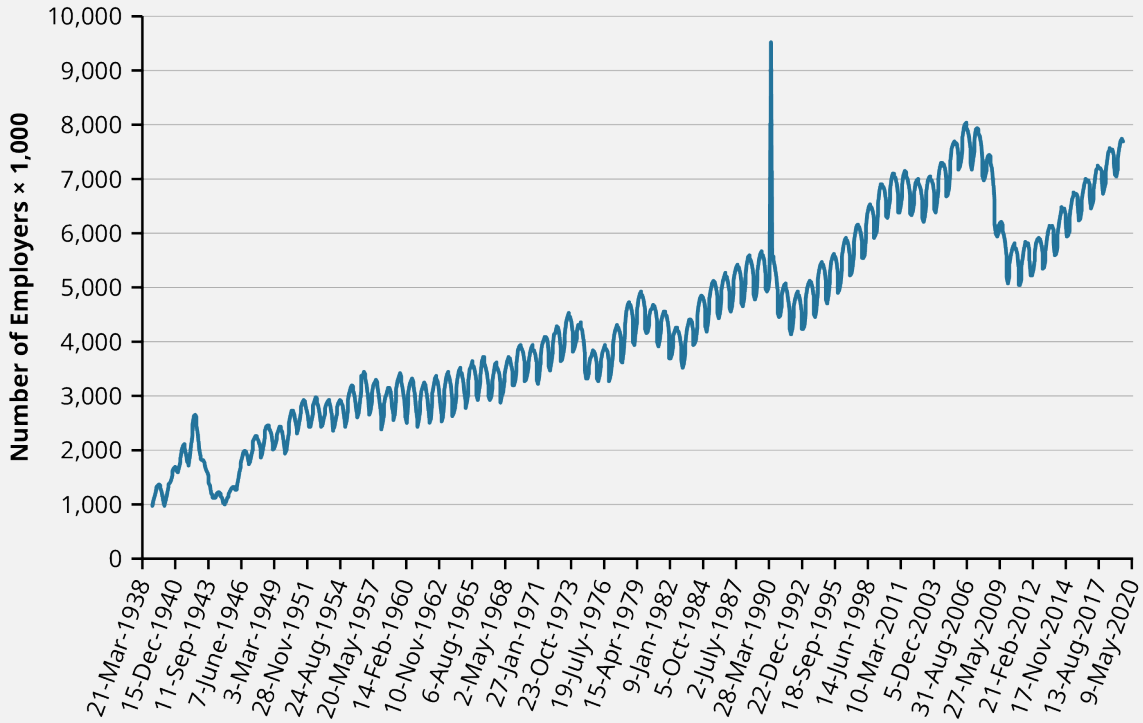


Figure 2.2 US Retail Employment with Outliers
(Data source: Bureau of Labor Statistics; credit: modification of work by Hyndman, R.J., & Athanasopoulos, G. (2021) Forecasting: principles and practice, 3rd edition, OTexts: Melbourne, Australia. OTexts.com/fpp3. Accessed on April 23, 2024)

Solution

a. Based on the visual evidence, it appears that an outlier is present in the employment data for March 28, 1990. The employment level during this month shows a significant jump from approximately 5,400 to 9,500, exceeding the overall maximum employment level recorded between 1939 and 2019.
b. A possible approach to addressing this outlier is to replace it with median, calculated by taking the mean of the points before and after the outlier. This method can help to improve the smoothness and realism of the employment curve as well as mitigate any potential impacts the outlier may have on statistical analysis or modeling processes. (See Table 2.3.)

Date	Number of Employers × 1000
1/1/1990	4974
2/1/1990	4933
3/1/1990	4989

Date	Number of Employers × 1000
4/1/1990	5174
5/1/1990	5370
6/1/1990	9523
7/1/1990	5580
8/1/1990	5581
9/1/1990	5487
10/1/1990	5379
11/1/1990	5208
12/1/1990	4963

Table 2.3 Monthly Employment over 1990

The median employment level from May 1, 1990, to July 1, 1990, is 5,289, as a replacement value for the outlier on May 28, 1990.

The adjusted data is presented in Figure 2.3.

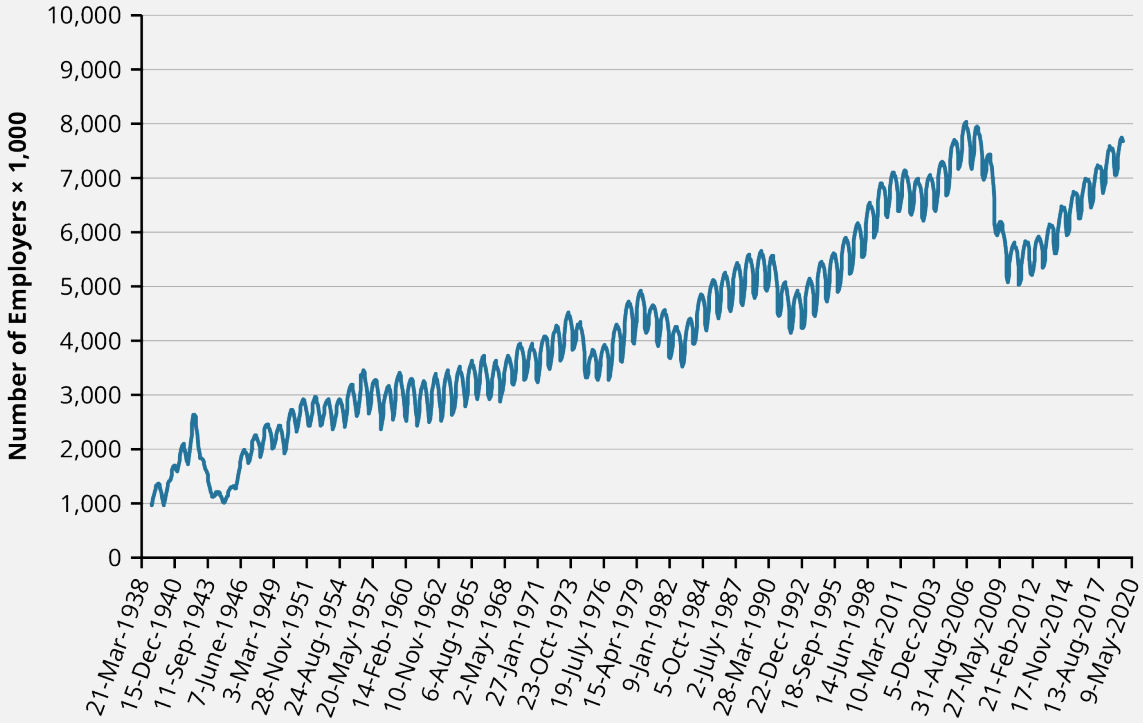


Figure 2.3 US Retail Employment Without Outliers
(Data source: Bureau of Labor Statistics; credit: modification of work by Hyndman, R.J., & Athanasopoulos, G. (2021) Forecasting: principles and practice, 3rd edition, OTexts: Melbourne, Australia. OTexts.com/fpp3. Accessed on April 23, 2024)

Data Standardization, Transformation, and Validation

Data standardization, transformation, and validation are critical steps in the data analysis preprocessing pipeline. **Data standardization** is the process of systematically transforming collected information into a consistent and manageable format. This procedure involves the elimination of inconsistencies, errors, and

duplicates as well as converting data from various sources into a unified format, often termed a *normal form* (defined in the next section). **Data transformation** involves modifying the data to make it more suitable for the analysis that is planned. **Data validation** ensures that the data is accurate and consistent and meets certain criteria or standards. Table 2.4 summarizes these processes, which are explored in more depth later in the chapter.

	Purpose	Techniques	Example
Normalization	To eliminate redundant data and ensure data dependencies make sense	Involve breaking down a larger database into smaller, more manageable tables and establishing relationships between them.	A customer database table with redundant data can be normalized by splitting it into two tables for customer information and orders and establishing the relationship between them.
Transformation	To make the data more consistent, coherent, and meaningful for analysis.	Include data cleaning, data merging, data splitting, data conversion, and data aggregation.	A dataset with different date formats, such as "MM/DD/YYYY" and "YYYY/MM/DD." Data transformation can be used to convert all dates to a single format
Validation	To improve the quality of data used in analysis. It ensures that the data is accurate, relevant, and consistent.	Include data profiling, data audits, and data cleansing.	In a survey, the respondent's age is recorded as 150 years. Through data validation, this value can be identified as erroneous.

Table 2.4 Comparison of Data Standardization, Transformation, and Validation Processes

Data Normalization

The first step in **standardizing data** is to establish guidelines and rules for formatting and structuring the data. This may include setting conventions for naming, data types, and formatting A **normal form (NF)** is a guideline or set of rules used in database design to ensure that a database is well-structured, organized, and free from certain types of data irregularities, such as redundancy and inconsistency. The most commonly used normal forms are *1NF, 2NF, 3NF* (First, Second, and Third Normal Form), and ***BCNF* (Boyce-Codd Normal Form)**.

Normalization is the process of applying these rules to a database. The data must be organized and cleaned, which involves removing duplicates and erroneous data, filling in missing values, and logically arranging the data. To uphold data standardization, regular quality control measures should be implemented, including periodic data audits to ascertain the accuracy and consistency of the data. It is also important to document the standardization process, including the guidelines and procedures followed. Periodic review and updates of data standards are necessary to ensure the ongoing reliability and relevance of the data.

Data normalization ensures that data is maintainable regardless of its source. Consider a marketing team collecting data on their customers' purchasing behavior so they can make some decisions about product placement. The data is collected from numerous sources, such as online sales transactions, in-store purchases, and customer feedback surveys. In its raw form, this data could be disorganized and unreliable, making it

difficult to analyze. It is hard to draw meaningful insights from badly organized data.

To normalize this data, the marketing team would go through multiple steps. First, they would identify the key data elements, such as customer name, product purchased, and transaction date. Then, they would ensure that these elements are consistently formatted in all data sources. For instance, they might employ the same date format across all data sources or standardize customer names to the first name and the last name fields. Subsequently, they would eliminate any redundant or irrelevant data elements. In this case, if the data is collected from both online and in-store purchases, they might choose one or the other to avoid duplication. The marketing team would ensure that the data is properly structured and organized. This could involve creating a data table with domains for each data element, such as a customer ID, product code, and purchase amount. By normalizing the data, the marketing team can efficiently follow and investigate the customers' purchasing behavior, identify patterns and trends, and make data-driven judgments to enhance their marketing systems.

The normalization formula is a statistical formula utilized to scale down a dataset, typically to between one and zero. The largest data would have a normalized value of one, and the smallest data point would be zero. Note that the presence of outliers can significantly impact the calculated values of minimum/maximum. As such, it is important to first remove any outliers from the dataset before performing normalization. This ensures more accurate and representative results.

The normalization formula:

$$x_{\text{norm}} = \frac{x - \min}{\max - \min}$$

EXAMPLE 2.7

Problem

A retail company with eight branches across the country wants to analyze its product sales to identify its top-selling items. The company collects data from each branch and stores it in Table 2.5, listing the sales and profits for each product. From previous reports, it discovered that its top-selling products are jewelry, TV accessories, beauty products, DVDs, kids' toys, video games, women's boutique apparel, and designer and fashion sunglasses. However, the company wants to arrange these products in order from highest to lowest based on best sales and profits. Determine which product is the top-selling product by normalizing the data in Table 2.5.

Branch	Product	Sales ($)	Profits ($)
Branch 1	Jewelry	50000	20000
Branch 2	TV Accessories	25000	12500
Branch 3	Beauty Products	30000	15000
Branch 4	DVDs	15000	7500
Branch 5	Kids' Toys	45000	22500
Branch 6	Video Games	35000	17500
Branch 7	Women's Boutique Apparel	40000	20000
Branch 8	Designer & Fashion Sunglasses	55000	27500

Table 2.5 Retail Company Sales and Profits

Solution

Using the normalization formula, the maximum sale is $55,000, and the minimum sale is $15,000, as shown in Table 2.6.

Branch	Product	Sales ($)	Profits ($)	Normalization Scale
Branch 1	Jewelry	50000	20000	0.88
Branch 2	TV Accessories	25000	12500	0.25
Branch 3	Beauty Products	30000	15000	0.38
Branch 4	DVDs	15000	7500	0.00
Branch 5	Kids' Toys	45000	22500	0.75
Branch 6	Video Games	35000	17500	0.50
Branch 7	Women's Boutique Apparel	40000	20000	0.63
Branch 8	Designer & Fashion Sunglasses	55000	27500	1.00

Table 2.6 Data of Retail Company Sales on Normalization Scale

Overall, the retail company's top-selling products generate the most profits for the company, with "Designer & Fashion Sunglasses" being the highest in the normalization scale. The company can use this information to focus on promoting and restocking these items at each branch to continue driving sales and profits.

Data Transformation

Data transformation is a statistical technique used to modify the original structure of the data in order to make it more suitable for analysis. Data transformation can involve various mathematical operations such as logarithmic, square root, or exponential transformations. One of the main reasons for transforming data is to address issues related to statistical assumptions. For example, some statistical models assume that the data is normally distributed. If the data is not distributed normally, this can lead to incorrect results and interpretations. In such cases, transforming the data can help to make it closer to a normal distribution and improve the accuracy of the analysis.

One commonly used data transformation technique is the **log transformation**, which requires taking the logarithm of the data values. Log transformation is often used when the data is highly skewed, meaning most of the data points fall toward one end of the distribution. This can cause problems in data analysis as the data may not follow a normal distribution. By taking the logarithm of the values, the distribution can be shifted toward a more symmetrical shape, making it easier to analyze. Another common transformation technique is the **square root transformation**, which involves taking the square root of the data values. Similar to log transformation, square root transformation is often used to address issues with skewness and make the data more normally distributed. Square root transformation is also useful when the data contains values close to zero, as taking the square root of these values can bring them closer to the rest of the data and reduce the impact of extreme values. **Exponential transformations** involve taking the exponent of the data values. Whichever operation is used, data transformation can be a useful tool for data analysts to address issues with data distribution and improve the accuracy of their analyses.

Dealing with Noisy Data

Noisy data refers to data that retains errors, outliers, or irrelevant information that can conceal true patterns and relationships within the dataset. The presence of noisy data in the dataset causes difficulty in drawing

accurate conclusions and making predictions from the data. Most noisy data is caused by human errors in data entry, technical errors in data collection or transmission, or natural variability in the data itself. Noisy data is removed and cleaned by identifying and correcting errors, removing outliers, and filtering out irrelevant information. Noisy data can negatively impact data analysis and modeling, and it may indicate that there are issues with the model's structure or assumptions. Noisy data is unwanted information that can be removed.

Strategies to reduce the noisy data are summarized in Table 2.7.

Strategy	Example
Data cleaning	Removing duplicate or irrelevant data from a dataset, such as deleting repeated rows in a spreadsheet or filtering out incomplete or error-prone data entries.
Data smoothing	A technique used in data analysis to remove outliers or noise from a dataset in order to reveal underlying patterns or trends. One example is smoothing a dataset of daily stock market index values over the course of a month. The values may fluctuate greatly on a day-to-day basis, making it difficult to see any overall trend. By calculating a seven-day average, we can smooth out these fluctuations and see the overall trend of the index over the course of the month.
Imputation	An example of imputation is in a hospital setting where a patient's medical history is incomplete due to missing information. The hospital staff can use imputation to estimate the missing data based on the patient's known medical conditions and past treatments.
Binning	A researcher is studying the age demographics of a population in a city. Instead of looking at individual ages, the researcher decides to bin the data into age groups of 10 years (e.g., 0–10, 10–20, 20–30, etc.). This allows for a more comprehensive and easier analysis of the data.
Data transformation	Consider the following dataset that shows the number of COVID-19 cases recorded in a country at different points in time—01/01/2020, 02/02/2020, 03/01/2020—are 1000, 10000, 100000, respectively. To transform this data using log, we can take the log base 10 of the number of cases column. This would result in a transformed data as follows: 01/01/2020, 02/02/2020, 03/01/2020 are 3, 4, 5, respectively.
Dimensionality reduction	The original dataset would have high dimensionality due to the large number of variables (100 companies) and time points (5 years). By applying principal component analysis, we can reduce the dimensionality of the dataset to a few principal components that represent the overall trends and patterns in the stock market.
Ensemble methods	An example of an ensemble method is the random forest algorithm. It combines multiple decision trees, each trained on a random subset of the data, to make a more accurate prediction. This helps reduce overfitting and increase the overall performance of the model. The final prediction is made by aggregating the predictions of each individual tree.

Table 2.7 Strategies to Reduce Noisy Data

Data Validation

Data validation is the process of ensuring the accuracy and quality of data examined against defined rules and standards. This approach involves identifying and correcting any errors or inconsistencies in the collected data as well as ensuring that the data is relevant and reliable for analysis. Data validation can be performed

through a variety of techniques, including manual checks, automated scripts, and statistical analysis. Some common inspections in data validation include checking for duplicates, checking for mislaid values, and verifying data against external sources or references. Before collecting data, it is important to determine the conditions or criteria that the data needs to meet to be considered valid. This can include factors such as precision, completeness, consistency, and timeliness. For example, a company may set up a data validation process to ensure that all customer information entered into its database follows a specific format. This would involve checking for correct spellings and proper formatting of phone numbers and addresses and validating the correctness of customer names and account numbers. The data would also be checked against external sources, such as official government records, to verify the accuracy of the information. Any discrepancies or errors would be flagged for correction before the data is used for analysis or decision-making purposes. Through this data validation process, the company can ensure that its customer data is accurate, reliable, and compliant with industry standards.

Another method to assess the data is to cross-reference it with reliable sources to identify any discrepancies or errors in the collected data. A number of tools and techniques can be used to validate data. These can include statistical analysis, data sampling, data profiling, and data auditing. It is important to identify and remove outliers before validating the data. Reasonability checks involve using common sense to check if the data is logical and makes sense—for example, checking if a person's age is within a reasonable range or if a company's revenue is within a reasonable range for its industry. If possible, data should be verified with the source to ensure its accuracy. This can involve contacting the person or organization who provided the data or checking against official records. It is always a good idea to involve multiple team members or experts in the validation process to catch any errors or inconsistencies that may have been overlooked by a single person. Documentation of the validation process, including the steps taken and any issues identified, is important in future data audits or for reference purposes. Data validation is a continuous process, and data should be monitored and updated to ensure its accuracy and validity.

Consider a marketing company conducting a survey on customer satisfaction for a new product launch. The company collected data from 1,000 respondents, but when the company started analyzing the data, it noticed several inconsistencies and missing values. The company's data analyst realized that the data standardization and validation processes were not adequately performed before the survey results were recorded. To correct this issue, the data analyst first identified and removed all duplicate entries, reducing the total number of responses to 900. Then, they used automated scripts to identify and fill in missing values, which accounted for 95 responses. The remaining 805 responses were then checked for data accuracy using statistical analysis. After the data standardization and validation process, the company had a clean and reliable dataset of 805 responses. The results showed that the product had a satisfaction rate of 85%, which was significantly higher than the initial analysis of 78%. As a result of this correction, the marketing team was able to confidently report the actual satisfaction rate and make better-informed decisions for future product development.

Data Aggregation

Data aggregation is the process by which information from multiple origins is gathered and merged into a single set that provides insights and meaningful conclusions. It involves gathering, managing, and delivering data from different sources in a structured manner to facilitate analysis and decision-making. Data aggregation can be performed manually or by using automated tools and techniques. The data aggregation process is utilized to identify patterns and trends between different data points, which extracts valuable insights. Some standard types of data aggregation are spatial aggregation, statistical aggregation, attribute aggregation, and temporal aggregation. This methodology is commonly operated in marketing, finance, health care, and research to analyze large sets of data. Data aggregation is used in various industries to combine and analyze large sets of data. Examples include calculating total sales for a company from different departments, determining average temperature for a region including multiple cities, and analyzing website traffic by country. It is also used in fields such as stock market indices, population growth, customer satisfaction scores, credit scores, and airline flight delays. Governments and utility companies also utilize data

aggregation to study energy consumption patterns.

Text Preprocessing

Text preprocessing is a technique of preparing data in text format for further analysis and natural language processing tasks. It involves transforming unstructured text data into a more structured format to be interpreted by algorithms and models. Some common techniques used in text preprocessing are reviewed in Table 2.8.

Preprocessing Technique	Explanation	Example
Tokenization	Breaking text data into individual words or phrases (tokens)	Original Text: "Tokenization is the process of breaking down a sentence, paragraph, or entire text into smaller parts called tokens." Tokenized Text: "Tokenization", "is", "the", "process", "of", "breaking", "down", "a", "sentence", ",", "paragraph", ",", "or", "entire", "text", "into", "smaller", "parts", "called", "tokens", "."
Lowercasing	Converting all text to lowercase to avoid multiple representations of the identical word	Consider the following sentence: "John likes to eat pizza." After lowercasing it, the sentence becomes "john likes to eat pizza."
Stopwords removal	Filtering out commonly occurring words that do not add meaning or context	Consider: "The sun was shining bright in the sky and the birds were chirping. It was a lovely day in the park and people were enjoying the beautiful weather." After removing stopwords, the paragraph would be transformed into: "Sun shining bright sky birds chirping. Lovely day park people enjoying beautiful weather."
Lemmatization and stemming	Reducing words to their root forms to reduce complexity and improve model performance	For example, the words "running," "runs," and "ran" would all be lemmatized to the base form "run."

Table 2.8 Summary of Text Preprocessing Techniques

Preprocessing Technique	Explanation	Example
Part-of-speech tagging	Identifying the grammatical components of each word where each word in a sentence is assigned a specific part-of-speech tag (e.g., noun, verb, or adjective)	In the sentence "I went to the market to buy some fruits," the words "went" and "buy" would be tagged as verbs, "market" and "fruits" as a noun, and "some" as adjectives.
Named entity recognition	Recognizing and categorizing named entities such as people, places, and organizations	Consider the following text: "John went to Paris last summer with his colleagues at Microsoft." By using named entity recognition, we can tag the named entities in this sentence as follows: "John (person) went to Paris (location) last summer with his colleagues at Microsoft (organization)."
Sentiment analysis	Identifying and categorizing the emotions expressed in text	Let's say a customer has left a review for a new laptop they purchased. The review reads: "I am extremely satisfied with my purchase. The laptop has exceeded all of my expectations and has greatly improved my work efficiency. Thanks for an amazing product!" To perform sentiment analysis, the text will first undergo preprocessing, which involves cleaning and preparing the text data for analysis. This may include removing punctuation, converting all letters to lowercase, and removing stopwords (commonly used words that do not add much meaning to a sentence, such as "the" or "and"). After preprocessing, sentiment analysis techniques will be applied to analyze the emotions and opinions expressed in the review. The analysis may identify key positive words such as "satisfied" and "amazing" and measure their overall sentiment score. It may also take into account the context and tone of the review to accurately determine the sentiment.

Table 2.8 Summary of Text Preprocessing Techniques

Preprocessing Technique	Explanation	Example
Spell-checking and correction	Correcting spelling errors to improve accuracy	Suppose we have a text like: "The writting was very inpprecise and had many trgical mistakes." With spell-checking and correction, this text can be processed and corrected to: "The writing was very imprecise and had many tragic mistakes." This involves identifying and correcting misspelled words. In this case, "writting" was corrected to "writing," "inpprecise" to "imprecise," and "trgical" to "tragic." This not only improves the readability and accuracy of the text but also helps in better understanding and analysis of the text.
Encoding	Converting text into a numerical representation for machine learning algorithms to process	Let's say we have a dataset of customer reviews for a restaurant. Each review is a string of text, such as: "I had a great experience at this restaurant, the food was delicious and the service was top-notch." To encode this text data, we can use techniques such as one-hot encoding or word embedding. One-hot encoding involves creating a binary vector for each word in the review, where the size of the vector is equal to the size of the vocabulary. For example, if the review contains the words "great," "experience," "restaurant," "food," "delicious," "service," and "top-notch," the one-hot encoded vectors for these words would be: great: [1, 0, 0, 0, 0, 0, 0]; experience: [0, 1, 0, 0, 0, 0, 0]; restaurant: [0, 0, 1, 0, 0, 0, 0]; food: [0, 0, 0, 1, 0, 0, 0]; delicious: [0, 0, 0, 0, 1, 0, 0]; service: [0, 0, 0, 0, 0, 1, 0]; top-notch: [0, 0, 0, 0, 0, 0, 1]. These one-hot encoded vectors can now be used as input features for machine learning models.
Removing special characters and punctuation	Simplifying the text for analysis	Consider the input: "Hello, this is a text with @special# characters!*" and the output: "Hello this is a text with special characters"

Table 2.8 Summary of Text Preprocessing Techniques

Text preprocessing is crucial to effectively use text data for tasks such as text classification and information extraction. By transforming the raw text data, the accuracy and performance of machine learning models can greatly improve and provide meaningful insights and predictions.

Text preprocessing is especially important in artificial intelligence, as it can lay the foundation for effective text analysis, classification, information retrieval, sentiment analysis, and many other *natural language processing* tasks (see Natural Language Processing). A well-designed preprocessing pipeline can lead to better model performance, improved efficiency, and more accurate insights from text data.

An example of text preprocessing in artificial intelligence involves the conversion of text data into a more standardized and consistent format. This can include tasks such as removing accents and diacritics, expanding contractions, and converting numbers into their written representations (e.g., "10" to "ten"). Normalizing text data helps to reduce the number of variations and therefore improves the efficiency and accuracy of natural

language processing tasks. It also makes the data more easily interpretable for machine learning algorithms. For example, when analyzing customer reviews, it may be beneficial to normalize text data so that variations of the same word (e.g., “colour” and “color”) are treated as the same, providing a more accurate understanding of sentiment toward a product or service.

2.5 Handling Large Datasets

Learning Outcomes

By the end of this section, you should be able to:

- 2.5.1 Recognize the challenges associated with large data, including storage, processing, and analysis limitations.
- 2.5.2 Implement techniques for efficient storage and retrieval of large datasets, including compression, indexing, and chunking.
- 2.5.3 Discuss database management systems and cloud computing and their key characteristics with regard to large datasets.

Large datasets, also known as **big data**, are extremely large and complex sets of data that traditional data processing methods and tools are unable to handle. These datasets typically include sizeable volumes, variety, and velocity of data, making it challenging to process, manage, and analyze them using traditional methods. Large datasets can be generated by a variety of sources, including social media, sensors, and financial transactions. They generally possess a high degree of complexity and may have structured, unstructured, or semi-structured data. Large datasets are covered in more depth in Other Machine Learning Techniques.

We have also already discussed a number of the techniques and strategies used to gain meaningful insights from big data. Survey Design and Implementation discussed sampling techniques that allow those working with data to analyze and examine large datasets by select a representative subset, or *representative random sample*. Preprocessing techniques covered in Data Cleaning and Preprocessing are also used to clean, normalize, and transform data to make sure it is consistent before it can be analyzed. This includes handling missing values, removing outliers, and standardizing data formats.

In this section we will consider several other aspects of data management that are especially useful with big data, including data compression, data storage, data indexing, and chunking. In addition, we’ll discuss database management systems—software that allows for the organization, manipulation, and retrieval of data that is stored in a structured format—and cloud computing.

Data Compression

Data compression is a method of reducing file size while retaining essential information; it can be applied to many types of databases. Data compression is classified into two categories, lossy and lossless:

- **Lossy compression** reduces the size of data by permanently extracting particular data that is considered irrelevant or redundant. This method can significantly decrease file sizes, but it also results in a loss of some data. Lossy compression is often utilized for multimedia files and images, where slight modifications in quality may not be noticeable to the human eye. Examples of lossy compression include MP3 and JPEG.
- **Lossless compression** aims to reduce file size without removing any data. This method achieves compression by finding patterns and redundancies in the data and representing them more efficiently. This allows for the reconstruction of the original data without any loss in quality. Lossless compression is commonly used for text and numerical data, where every piece of information is crucial. Examples of lossless compression include ZIP, RAR, and PNG.

  There are several methods of data lossless compression, including Huffman coding. **Huffman coding** works by assigning shorter binary codes to the most frequently used characters or symbols in a given dataset and longer codes to less frequently used characters, as shown in Figure 2.4. This results in a more

efficient use of binary digits and reduces the overall size of the data without losing any information during compression. Huffman coding is applied to data that require fast and efficient compression, such as video compression, image compression, and data transmission and storage.

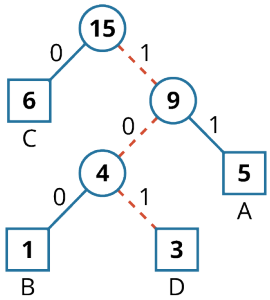


Figure 2.4 Huffman Tree Diagrams

Data Storage

Big data requires storage solutions that can handle large volumes of diverse data types, offer high performance for data access and processing, guarantee scalability for growing datasets, and are thoroughly reliable. The choice of storage solution will depend on data volume, variety, velocity, performance requirements, scalability needs, budget constraints, and existing infrastructure. Organizations often deploy a combination of storage technologies to address different uses and requirements within their big data environments. Some common types of storage solutions used for big data include:

1. **Relational databases**: **Relational databases**. organize data in tables and uses structured query language (SQL) for data retrieval and management. They are commonly used for traditional, structured data such as financial data.
2. **NoSQL databases**. These databases are designed to handle unstructured data, such as social media content or data from sensors, and use non-relational data models.
3. **Data warehouses.** A **data warehouse** is a centralized repository of data that combines data from multiple sources and allows for complex queries and analysis. It is commonly used for business intelligence and reporting intentions.
4. **Cloud storage**. **Cloud storage** involves storing data in remote servers accessed over the internet. It offers scalability, cost-effectiveness, and remote accessibility.
5. **Object storage**. With **object storage**, data are stored as objects that consist of both data and metadata. This method is often used for storing large volumes of unstructured data, such as images and videos.

Data Indexing

Data indexing is an important aspect of data management and makes it easier to retrieve specific data quickly and efficiently. It is a crucial strategy for optimizing the performance of databases and other data storage systems. **Data indexing** refers to the process of managing and saving the collected and generated data in a database or other data storage system in a way that allows for efficient and fast return of specific data.

Indexing techniques vary in how they organize and store data, but they all aim to improve data retrieval performance. *B-tree indexing* is illustrated in Figure 2.5. It involves organizing data in a tree-like structure with a root node and branches that contain pointers to other nodes. Each node contains a range of data values and pointers to child nodes, allowing for efficient searching within a specific range of values.

Hashes indexing involves using a hash function to map data to a specific index in a table. This allows for direct access to the data based on its hashed value, making retrieval faster than traditional sequential searching. *Bitmap indexing* is a technique that involves creating a bitmap for each different value in a dataset. The bitmaps are then combined to quickly identify records that match a specific set of values, allowing efficient data retrieval.

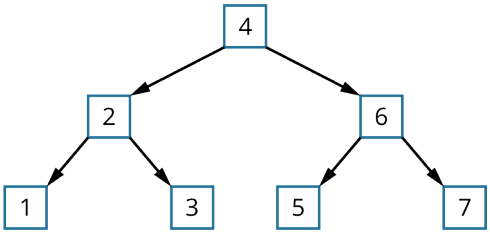


Figure 2.5 B-Tree Indexing

Data Chunking

Data chunking, also known as *data segmentation* or *data partitioning*, is a technique used to break down large datasets into smaller, more manageable chunks and make them easier to manage, process, analyze, and store. Chunking is particularly useful when datasets are too large to be processed or analyzed as a single unit. By dividing the data into smaller chunks, various processing tasks can be distributed across multiple computing nodes or processing units.

Data chunking is used in data storage systems and data transmission over networks, and it is especially useful when working with large datasets that exceed the capacity of a single machine or when transferring data over a network with limited bandwidth. The divided data in data chunking is known as a *chunk* or *block*. The size of each chunk can vary depending on the requirements and range from a few kilobytes to several gigabytes. These chunks are typically organized sequentially, with each chunk containing a set of data from the larger dataset. The process of data chunking also involves adding **metadata** (data that provides information about other data), such as the chunk number and the total number of chunks, to each chunk. This metadata allows the chunks to be reassembled into the original dataset after being transmitted or stored separately. Data chunking has several advantages, including the following:

1. **Increased speed.** By dividing a large dataset into smaller chunks, data processing and transmission can be performed more quickly, reducing the overall processing time.
2. **Better utilization of resources.** Data chunking enables data to be distributed and processed across multiple machines, making better use of available computing resources.
3. **Increased fault tolerance.** In case of data corruption or loss, data chunking allows for the retrieval of only the affected chunk rather than the entire dataset.
4. **Flexibility.** Data chunking allows for the transfer and processing of only the required chunks rather than the entire dataset, providing flexibility in managing large datasets.

Database Management Systems

Database management is a crucial aspect of data science projects as it involves organizing, storing, retrieving, and managing large volumes of data. In a data science project, a database management system (DBMS) is used to ensure the efficient storage and retrieval of data. Database management systems are software tools used for managing data in a structured format. Their functions are summarized in Table 2.9.

DBMS Function	Description	Benefit
Data storage	Provides a centralized warehouse for storing different types of data in a structured format	Makes data easy to retrieve and analyze
Data retrieval	Allows for efficient and fast retrieval of data from the database using queries and filters	Makes data more accessible to data scientists

Table 2.9 Functions of Database Management Systems (DBMS)

DBMS Function	Description	Benefit
Data organization	Helps to manage data in a structured format	Makes data more manageable for performing analysis and identifying patterns or relationships between different data points
Data security	Provides strong security measures to protect sensitive data from unauthorized access	Protects sensitive data such as personal information or financial data from unauthorized access
Data integration	Permits the integration of data from multiple sources	Makes it possible to combine and analyze data from different datasets

Table 2.9 Functions of Database Management Systems (DBMS)

Implementation of database management techniques has become important for hospitals in achieving better patient outcomes and reducing costs. This strategy involves the collection and analysis of patient data from different sources, including electronic health records, medical imaging, and lab results. For example, hospitals are utilizing this approach to improve treatment for patients with chronic conditions such as diabetes and heart disease. By leveraging data-driven insights and identifying patterns, health care experts can develop personalized remedy plans for each patient, guiding them to enhanced health and wellness as well as cost savings for the hospital. This showcases the significance of incorporating advanced data management techniques in health care systems. Through accurate and efficient management and analysis of patient data, hospitals and health care providers are able to make informed decisions, eventually resulting in a more efficient and effective health care system overall.

Cloud Computing

Cloud computing delivers a cost-effective solution for storing vast amounts of data, enabling seamless collaboration and data transfer among remote groups. This technology comprises remote-access tools for storage, processing, and analytics, facilitating multiple users' access regardless of their physical location. Moreover, cloud computing boasts a diverse range of data collection tools, including machine learning, and data warehouses, streamlining the data assembly operation and improving overall efficiency. Cloud computing equips data scientists with the necessary resources and flexibility to effectively collect, manage, and analyze data for their projects. Some examples of cloud storage are Amazon AWS, Microsoft Azure, and Google Cloud.

EXAMPLE 2.8

Problem

The CEO of a large insurance company faced the challenge of increasing digital processes and documents, leading to a need for more storage capacity and rising costs for maintaining servers and hardware. What are all the available options for the insurance company facing the need for additional storage capacity at a higher cost, and which solution would be the most effective for decreasing costs and increasing capacity while ensuring data security?

Solution

1. Migrating to a cloud storage system: This would allow the company to offload the burden of maintaining physical servers and hardware while also providing virtually unlimited storage capacity. The cost of cloud storage is also generally more flexible and scalable, allowing the company to only pay for the storage it needs. Additionally, with the use of cloud-based document management systems, the company can streamline and automate its processes, reducing the need for physical documentation

and increasing efficiency. However, this decision would require careful consideration of security measures and potentially training for employees to adapt to the new system.

2. Implementing a data archiving strategy: Instead of immediately migrating to the cloud or investing in new technology, the company could consider implementing a data archiving strategy. This involves identifying and storing infrequently used data in a separate, low-cost storage system, freeing up space on servers and reducing costs.
3. Outsourcing data storage and management: The company could consider outsourcing its data storage and management to a third-party provider. This would eliminate the need for maintaining physical servers and hardware, and the provider may also offer advanced security measures and data backup options.
4. Consolidating data and processes: The company could assess its current processes and systems to identify areas where data and processes can be consolidated to reduce the need for multiple storage systems and streamline workflows.
5. Implementing a virtual desktop infrastructure: A virtual desktop infrastructure allows employees to access their desktop and applications from a central server, reducing the need for individual storage space on devices. This can also improve security and data backup capabilities.
6. Upgrading or redesigning its current storage system: The company could invest in upgrading or redesigning its current storage system to improve efficiency and increase storage capacity.
7. Utilizing hybrid cloud storage: Instead of fully migrating to the cloud, the company could consider a hybrid approach where certain sensitive data is stored on-premises while less critical data is stored in the cloud. This approach can offer the benefits of both on-premises and cloud storage.
8. Conducting regular audits of data usage and storage: The company could conduct regular audits of its data usage and storage to identify areas of redundancy or inefficiency and adjust accordingly. This can help optimize storage and reduce costs over time.

EXAMPLE 2.9

Problem

What is the descending order of storage capacity in Example 2.8, starting from highest to lowest, while maintaining the same cost and the same level of security?

Solution

The available storage options in Example 2.8, ranked from highest to lowest capacity at the same cost and security level, are:

1. Cloud storage system: This option would provide virtually unlimited storage capacity, as the company can scale up as needed at a relatively low cost. However, the overall capacity and cost would depend on the specific cloud storage plan chosen.
2. Data archiving strategy: By identifying and storing infrequently used data, the company can free up space on its current servers and reduce costs. However, the storage capacity will be limited to the existing servers.
3. Outsourcing data storage and management: By outsourcing to a third-party provider, the company can potentially access higher storage capacity than its current setup, as the provider may have more advanced storage options. However, this would also depend on the specific plan and cost negotiated.
4. Implementing a virtual desktop infrastructure: This option may provide slightly lower storage capacity compared to outsourcing, as it still relies on a central server. However, it can still be an improvement compared to the company's current setup.

5. Upgrading or redesigning its current storage system: This option may also result in lower storage capacity compared to outsourcing, as it only involves improving the existing server setup rather than using an external provider. However, it can still provide a significant increase in capacity depending on the specific upgrades implemented.
6. Hybrid cloud storage: This option can provide a mix of higher and lower storage capacity depending on the specific data stored in the cloud and on-premises. Sensitive data may have lower capacity on the cloud, while less critical data can be stored at higher capacity.
7. Consolidating data and processes: By streamlining processes and eliminating redundancies, the company can potentially reduce the need for excessive storage capacity. However, this would also depend on the company's current setup and level of optimization.
8. Regular audits of data usage and storage: This option may not directly impact storage capacity, but by identifying and eliminating redundancies, the company can optimize its existing storage capacity and potentially reduce costs in the long run.

Datasets

Note: The primary datasets referenced in the chapter code may also be [downloaded here (https://openstax.org/r/drive)](https://openstax.org/r/drive).)

Key Terms

application programming interface (API) set of protocols, tools, and definitions for building software applications that allow different software systems to communicate and interact with each other and enable developers to access data and services from other applications, operating systems, or platforms

***BCNF* (Boyce-Codd Normal Form)** a normal form that is similar to 3NF but stricter in its requirements for data independence; it ensures that all attributes in a table are functionally dependent on the primary key and nothing else

big data complex, high-volume and varied data that are challenging to process with traditional methods

closed-ended questions clear, structured questions with predetermined answer choices that are effective in gathering quantitative data

cloud computing the delivery of computing services, such as storage, processing power, and software applications, over the internet

cloud storage type of online storage that allows users and organizations to store, access, and manage data over the internet with data. This data is stored on remote servers managed by a cloud storage service provider.

data aggregation the process of collecting and combining information from multiple sources into a single database or dataset

data chunking a technique used to break down large datasets into smaller, more manageable chunks and make them easier to manage, process, analyze, and store; also known as data segmentation

data compression the process of reducing the size of a data file or transmission to make it easier and faster to store, transmit, or manipulate

data dictionary a data structure that stores data in key-value pairs, allowing for efficient retrieval of data using its key

data discretization the process of converting continuous data into discrete categories or intervals to allow for easier analysis and processing of the data

data indexing the process of organizing and storing data in a way that makes it easier to search and retrieve information

data integration the process of combining data from multiple sources, such as databases, applications, and files, into a unified view

data processing the process of collecting, organizing, and manipulating data to produce meaningful information involving the transformation of raw data into a more useful and understandable format

data standardization the process of transforming data into a common scale or range to help remove any unit differences or variations in magnitude

data transformation the process of modifying data to make it more suitable for the planned analysis

data validation procedure to ensure that the data is reliable and trustworthy for use in the data science project, done by performing checks and tests to identify and correct any errors in the data

data warehouse a large, organized repository of integrated data from various sources used to support decision-making processes within a company or organization

DataFrame a two-dimensional data structure in Python that is used for data manipulation and analysis

exponential transformation a data transformation operation that involves taking the exponent of the data values

Huffman coding reduces file size by assigning shorter binary codes to the most frequently used characters or symbols in a given dataset and longer codes to less frequently used characters

imputation the process of estimating missing values in a dataset by substituting them with a statistically reasonable value

log transformation data transformation technique that requires taking the logarithm of the data values and is often used when the data is highly skewed

lossless compression aims to reduce file size without removing any data, achieving compression by finding patterns and redundancies in the data and representing them more efficiently

lossy compression reduces the size of data by permanently extracting particular data that is considered irrelevant or redundant

measurement error inaccuracies or discrepancies that surface during the process of collecting, recording, or analyzing data caused by human error, environmental factors, or inconsistencies in the data

metadata data that provides information about other data

missing at random (MAR) data missing data related to other variables but not to any unknown or unmeasured variables; the missing data can be accounted for and included in the analysis through statistical techniques

missing completely at random (MCAR) data type of missing data that is not related to any other variables, with no underlying cause for its absence

missing not at random (MNAR) data type of data missing in a way that depends on unobserved data

network analysis the examination of relationships and connections between users, data, or entities within a network to identify and analyze influential individuals or groups and understand patterns and trends within the network

noisy data data that contains errors, inconsistencies, or random fluctuations that can negatively impact the accuracy and reliability of data analysis and interpretation

normal form a guideline or set of rules used in database design to ensure that a database is well-structured, organized, and free from certain types of data irregularities, such as redundancy and inconsistency

normalization the process of transforming numerical data into a standard or common range, usually between 0 and 1, to eliminate differences in scale and magnitude that may exist between different variables in a dataset

NoSQL databases databases designed to handle unstructured data, such as social media content or data from sensors, using non-relational data models

object storage method of storage where data are stored as objects that consist of both data and metadata and often used for storing large volumes of unstructured data, such as images and videos

observational data data that is collected by observing and recording natural events or behaviors in their normal setting without any manipulation or experimentation; typically collected through direct observation or through the use of instruments such as cameras or sensors

open-ended questions survey questions that allow for more in-depth responses and provide the opportunity for unexpected insights

outlier a data point that differs significantly from other data points in a given dataset

regular expressions (regex) a sequence of characters that define a search pattern, used for matching and manipulating strings of text; commonly used for cleaning, processing, and manipulating data in a structured or unstructured format

relational databases databases that organize data in tables and use structured query language (SQL) for data retrieval and management; often used with financial data

sampling the process of selecting a subset of a larger population to represent and gather information about that population

sampling bias occurs when the sample used in a study isn't representative of the population it intends to generalize; can lead to skewed or inaccurate conclusions

sampling error the difference between the results obtained from a sample and the true value of the population parameter it is intended to represent

sampling frame serves as a guide for selecting a representative sample of the population and determining the coverage of the survey and ultimately affects the validity and accuracy of the results

scaling one of the steps involved in data normalization; refers to the process of transforming the numerical values of a feature to a particular range

slicing a string extracting a portion or section of a string based on a specified range of indices

social listening the process of monitoring and analyzing conversations and discussions happening online, specifically on social media platforms, to track mentions, keywords, and hashtags related to a specific topic or business to understand the sentiment, trends, and overall public perception

splitting a string dividing a text string into smaller parts or substrings based on a specified separator
square root transformation data transformation technique that requires taking the square root of the data values; is useful when the data contains values close to zero
standardizing data the process of transforming data into a common scale or range to help remove any unit differences or variations in magnitude
string a data type used to represent a sequence of characters, such as letters, numbers, and symbols, in a computer program
text preprocessing the process of cleaning, formatting, and transforming raw text data into a form that is more suitable and easier for natural language processing to understand and analyze
tokenizing the process of breaking down a piece of text or string of characters into smaller units called tokens, which can be words, phrases, symbols, or individual characters
transactional data a type of information that records financial transactions, including purchases, sales, and payments. It includes details such as the date, time, and parties involved in the transaction.
web scraping a method of collecting data from websites using software or scripts. It involves extracting data from the HTML code of a web page and converting it into a usable format.

Group Project

Project A: Collecting Data on Extinct Animals Due to the Impact of Global Warming

The main objective of this project is to gather data before and after 1900 on the various species that have gone extinct in order to analyze the impact of climate change on these animals and determine the most affected species. To collect data on extinct animals, you need to utilize various sources such as biodiversity databases, academic papers, government reports, and conservation organizations' data.

As a group, complete the following steps:

1. **Selection of data collection method.** The students will determine the most appropriate data collection method based on the research topic and available resources. This could include observational techniques, transactional data, surveys, experimental data, or web scraping. The method chosen should allow for the collection of reliable and relevant data on extinct animals.
2. **Independent research on reliable sources of data.** Each student will conduct independent research to locate credible sources of data on animal extinctions.
3. **Data organization into a table format.** Using the collected information, students will create a table to present data on the distribution and causes of animal extinctions per continent. The table will include columns for the animal's common name, scientific name, date of extinction, continent, and primary reason for extinction. Table 2.10 provides an example.

Common Name	Scientific Name	Date of Extinction	Continent	Primary Reason for Extinction
Passenger pigeon	*Ectopistes migratorius*	1914	North America	Overhunting
Quagga	*Equus quagga*	1883	Africa	Habitat loss

Table 2.10 Project A Required Data and Example

4. **Collaborative data cleaning.** Students will work together to clean the collected data, identifying and addressing any missing or duplicate information. They will also ensure that the data is standardized, allowing for accurate analysis and comparison.
5. **Data storage determination.** In this step, the students will determine the storage size of the collected

data, taking into consideration the amount of information gathered from various sources. They will also decide on the most efficient and accessible way to store the data, ensuring all group members have access to it for further analysis and collaboration.

6. **Source citation.** As part of the data collection process, students will document and list all the sources used to gather information on extinct animals. This will ensure proper credit is given to the original sources and allow for further verification of the data.

Project B: Data Collection Using Experimental Technique

Your group project will involve recording temperature changes at a specific location over the course of 24 hours. This project will be completed in subgroups, with each group member responsible for recording data during a specific time segment of the day. Before beginning your research, your group will select a location to observe. (This could be a park, a city street, a backyard, or any other outdoor location. It is important that the location is accessible and allows for consistent monitoring throughout the day.) Each subgroup will be responsible for recording temperature data for a six-hour time period. The first subgroup will record data from midnight to 6 a.m., the second subgroup from 6 a.m. to 12 p.m., the third subgroup from 12 p.m. to 6 p.m., and the fourth subgroup from 6 p.m. to midnight.

It is crucial that each subgroup records data at the exact same time each day in order to maintain consistency. During the data collection process, it is important for each subgroup to maintain the same surrounding conditions. This includes using the same tools, such as a thermometer, and staying in the same location throughout the time period. Any uncontrolled changes or errors should be noted and recorded to ensure accuracy in the data.

Repeat this process for seven days to gather a sufficient amount of data, and once all the data has been collected, calculate the average temperature for each time segment of the day by adding up all the recorded temperatures and dividing by the number of data points.

Once all data has been collected, the group will come together to analyze and graph the temperature changes over the course of 24 hours. This will allow for a visual representation of the temperature patterns throughout the day.

Discuss potential sources of error and ways to improve the data collection process in the future. Then summarize the collected data process, present the project to the rest of the group, and discuss any insights or lessons learned from the data collection process.

Critical Thinking

1. Megan is interested in opening a gaming center that utilizes virtual reality (VR) technology. She wants to gauge the level of interest in VR gaming among the residents of her city. She plans to conduct a survey using a random sampling method and then use this data to determine the potential demand for a VR gaming center in her city. What is the nature of the collected data, experimental or observational?

2. A high school student is interested in determining the relationship between the weight of an object and the distance it can travel when launched from a catapult. To collect data for analysis, the student plans to launch objects of varying weight and then measure the corresponding distance traveled. Using this data, the student plans to plot a graph with weight on the *x*-axis and distance on the *y*-axis.
   a. Will this data collection result in observational data or experimental data?
   b. How can the student gather the necessary data to establish this relationship?

3. A high school is conducting a survey to understand the opinions of the school's students toward the food options available in the cafeteria. They have divided the student population into four groups based on grade level (9th, 10th, 11th, 12th). A random sample of 60 students is selected from each group and asked to rate the cafeteria food on a scale of 1–10. The results are shown in the following table:

Grade Level	Number of Students	Average Rating
9th	60	8.1
10th	60	7.4
11th	60	6.3
12th	60	6.1

Table 2.11 High School Survey

What sampling technique was used to collect this data?

4. A research study is being conducted to understand the opinions of people on the new city budget. The population consists of residents from different neighborhoods in the city with varying income levels. The following table provides the data of 130 participants randomly selected for the study, categorized by their neighborhood and income level.

Neighborhood	Income Level	Number of Participants
Northside	Low Income	35
Southside	Middle Income	50
Eastside	High Income	45

Table 2.12 Neighborhood and Income level Category

What sampling technique was used to collect this data?

5. John is looking to gather data on the popularity of 3D printing among local college students to determine the potential demand for a 3D printing lab on campus. He has decided on using surveys as his method of data collection. What sampling methods would be most effective for obtaining accurate results?

6. A group of researchers were tasked with gathering data on the price of electronics from various online retailers for a project. Suggest an optimal approach for collecting data on electronics prices from different online stores for a research project.

7. As the pandemic continues to surge, the US Centers for Disease Control and Prevention (CDC) releases daily data on COVID-19 cases and deaths in the United States, but the CDC does not report on Saturdays and Sundays, due to delays in receiving reports from states and counties. Then the Saturday and Sunday cases are added to the Monday cases. This results in a sudden increase in reported numbers of Monday cases, causing confusion and concern among the public. (See the table provided.) The CDC explains that the spike is due to the inclusion of delayed or improperly recorded data from the previous week. This causes speculation and panic among the public, but the CDC reassures them that there is no hidden outbreak happening.

Date	Day	New Cases
10/18/2021	Monday	3115
10/19/2021	Tuesday	4849
10/20/2021	Wednesday	3940
10/21/2021	Thursday	4821
10/22/2021	Friday	4357
10/23/2021	Saturday	0
10/24/2021	Sunday	0
10/25/2021	Monday	8572
10/26/2021	Tuesday	4463
10/27/2021	Wednesday	5323
10/28/2021	Thursday	5012
10/29/2021	Friday	4710
10/30/2021	Saturday	0
10/31/2021	Sunday	0
11/1/2021	Monday	10415
11/2/2021	Tuesday	5096
11/3/2021	Wednesday	6882
11/4/2021	Thursday	5400
11/5/2021	Friday	6759
11/6/2021	Saturday	0
11/7/2021	Sunday	0

Table 2.13 Sample of COVID-19 Data Cases Within 23 Days (source: https://data.cdc.gov/Case-Surveillance)

11/8/2021	Monday	10069
11/9/2021	Tuesday	5297

Table 2.13 Sample of COVID-19 Data Cases Within 23 Days (source: https://data.cdc.gov/Case-Surveillance)

a. What simple strategies can the data scientists utilize to address the issue of missing data in their analysis of COVID-19 data from the CDC?

b. How can data scientists incorporate advanced methods to tackle the issue of missing data in their examination of COVID-19 cases and deaths reported by the CDC?

8. What distinguishes data standardization from validation when gathering data for a data science project?

9. Paul was trying to analyze the results of an experiment on the effects of music on plant growth, but he kept facing unexpected data. He was using automated instrumentation to collect growth data on the plants at periodic time intervals. He found that some of the data points were much higher or lower than anticipated, and he suspected that there was noise (error) present in the data. He wanted to investigate the main source of the error in the data so that he could take corrective actions and thus minimize the impact of the error(s) on the experiment results. What is the most likely source of error that Paul might encounter in this experiment: human error, instrumentation error, or sampling error? Consider the various sources of noise (error).

10. The government is choosing the best hospital in the United States based on data such as how many patients they have, their mortality rates, and overall quality of care. They want to collect this data accurately and fairly to improve the country's health care.

a. What is the best way to collect this data? Can we use web scraping to gather data from hospital websites? Can we also use social media data collection to gather patient reviews and feedback about the hospitals?

b. How can we ensure the data is unbiased and accurate? Can we use a mix of these methods to create a comprehensive analysis?

11. Sarah, a data scientist, was in charge of hiring a new sales representative for a big company. She had received a large number of applications. To make the hiring process more efficient, she decided to streamline the process by using text processing to analyze the candidates' applications to identify their sentiment, tone, and relevance. This would help Sarah gain a better understanding of their communication skills, problem-solving abilities, and overall fit for the sales representative role. She compiled two lists of keywords, one for successful candidates and one for rejected candidates. After generating summaries of the keywords from each application, Sarah was ready to make informed and efficient hiring decisions. Identify the best candidate based on the list of keywords associated with each candidate, as reviewed in the figure shown. Your answer should be supported with strong reasoning and evidence.

Candidates	Customer-Centric	Team Player	Collaborative	Sales Experience	Leadership	Multitasking	Goal-Oriented	Team-Oriented	Effective	Inflexible	Uncooperative	Indecisive	Negative	Lacking	Limited
John	✓	✓	✓	✓	✓										
Sarah		✓	✓	✓			✓		✓						
Lin			✓	✓										✓	✓
Aysha	✓				✓					✓	✓	✓	✓	✓	
Alex		✓		✓			✓		✓	✓					
Peter	✓	✓	✓	✓						✓	✓		✓	✓	
Jamal	✓	✓		✓		✓	✓	✓							
Brian	✓	✓	✓	✓	✓	✓	✓					✓		✓	
Samantha	✓	✓	✓	✓	✓		✓	✓	✓						
Miguel					✓	✓	✓	✓	✓	✓	✓	✓			

Figure 2.6 Data Collection Based on Text Processing

12. How can a data analyst for a large airline company efficiently collect and organize a significant amount of data pertaining to flight routes, passenger demographics, and operational metrics in a manner that ensures reliable and up-to-date records, complies with industry standards, and adheres to ethical guidelines and privacy regulations? The data will be utilized to identify potential areas for improvement, such as flight delays and high passenger traffic, so that the company can implement effective strategies to address these issues.

13. A chain of restaurants that serve fast food needs to improve their menu by providing healthier options to their customers. They want to track the nutrition information of their dishes, including burgers, fries, and milkshakes, to make it easier for the managers to compare their calorie, fat, and sugar content. What data management tools can restaurants utilize to track nutritional data to make healthier options readily available and enhance their menu?

14. The NFL team's talent acquisition specialist is on a mission to find the most talented high school football player to join their team. The specialist is responsible for gathering extensive information on potential players, such as their stats, performance history, and overall potential. The objective is to pinpoint the standout athlete who will bring a significant advantage to the NFL team. To achieve this, the specialist utilizes cloud computing to sift through vast amounts of data and identify the ideal high school player who can make a valuable contribution to the team's success in the upcoming season. Why is cloud computing the perfect tool for collecting and analyzing data on high school football players?

References

1. Elfil, M., & Negida, A. (2017). Sampling methods in clinical research; an educational review. *Emergency (Tehran, Iran)*, *5*(1), e52, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5325924/

2. Lusinchi, D. (2012). President Landon and the 1936 Literary Digest poll: Were automobile and telephone owners to blame? *Social Science History*, *36*(1), 23–54. https://doi.org/10.1017/S014555320001035X

3 Descriptive Statistics: Statistical Measurements and Probability Distributions

Figure 3.1 Statistics and data science play significant roles in stock market analysis, offering insights into market trends and risk assessment for better financial decision-making. (credit: modification of work "That was supposed to be going up, wasn't it?" by Rafael Matsunaga/Flickr, CC BY 2.0)

Chapter Outline

3.1 Measures of Center
3.2 Measures of Variation
3.3 Measures of Position
3.4 Probability Theory
3.5 Discrete and Continuous Probability Distributions

Introduction

Statistical analysis is the science of collecting, organizing, and interpreting data to make decisions. Statistical analysis lies at the core of data science, with applications ranging from consumer analysis (e.g., credit scores, retirement planning, and insurance) to government and business concerns (e.g., predicting inflation rates) to medical and engineering analysis.

Statistical analysis is an essential aspect of data science, involving the systematic collection, organization, and interpretation of data for decision-making. It serves as the foundation for various applications in consumer analysis such as credit scoring, retirement planning, and insurance as well as in government and business decision-making processes such as inflation rate prediction and marketing strategies. As a consumer, statistical analysis plays a significant role in various decision-making processes. For instance, when considering a large financial decision such as purchasing a house, the probability of interest rate fluctuations and their impact on mortgage financing must be taken into account.

Part of statistical analysis involves **descriptive statistics**, which refers to the collection, organization, summarization, and presentation of data using various graphs and displays. Once data is collected and summarized using descriptive statistics methods, the next step is to analyze the data using various probability tools and probability distributions in order to come to conclusions about the dataset and formulate predictions that will be useful for planning and estimation purposes.

Descriptive statistics includes measures of the center and dispersion of data, measures of position, and generation of various graphical displays. For example, a human resources administrator might be interested in generating some statistical measurements for the distribution of salaries at a certain company. This might involve calculating the mean salary, the median salary, and standard deviation, among others. The administrator might want to present the data to other employees, so generating various graphical displays such as histograms, box plots, and scatter plots might also be appropriate.

The human resources administrator might also want to derive estimates and predictions about the average salary in the company five years into the future, taking into account inflation effects, or they might want to create a model to predict an individual employee's salary based on the employee's years of experience in the field.

Such analyses are based on the statistical methods and techniques we will discuss in this chapter, building on the introduction to statistical analysis presented in What Are Data and Data Science? and Collecting and Preparing Data. Statistical analysis utilizes a variety of technological tools to automate statistical calculations and generate graphical displays. This chapter also will demonstrate the use of Python to generate results and graphs. The supplementary material at the end of this book will show the same using Excel (Appendix A) and R (Appendix B).

In this chapter, you will also study probability concepts and probability distributions. Probability theory helps us deal with quantifying uncertainty, which is always inherent in real-world data. We will see that in real-world datasets we often have to deal with noise and randomness that will be analyzed using statistical analyses.

Probability analysis provides the tools to model, understand, and quantify uncertainties, allowing data scientists to make informed decisions from data. Probability theory also forms the basis for different types of analyses, such as confidence intervals and hypothesis testing, which will be discussed further in Inferential Statistics and Regression Analysis. Such methods rely on probability models to make predictions and detect patterns. In addition, machine learning (discussed in Time Series and Forecasting) uses probabilistic models to help represent uncertainty and make predictions based on the collected data. Probability analysis helps data scientists to select data models, interpret the results, and assess the reliability of these conclusions. For a more detailed review of statistical concepts, please refer to Introductory Statistics 2e (https://openstax.org/books/introductory-statistics-2e/pages/1-introduction).

3.1 Measures of Center

Learning Outcomes

By the end of this section, you should be able to:

- 3.1.1 Define and calculate mean, trimmed mean, median, and mode for a dataset.
- 3.1.2 Determine the effect of outliers on the mean and the median.
- 3.1.3 Use Python to calculate measures of center for a dataset.

Measures of center are statistical measurements that provide a central, or typical, representation of a dataset. These measures can help indicate where the bulk of the data is concentrated and are often called the data's **central tendency**. The most widely used measures of the center of a dataset are the *mean* (average), the *median*, and the *mode*.

Mean and Trimmed Mean

The **mean**, or average, (sometimes referred to as the *arithmetic mean*) is the most commonly used measure of the center of a dataset. Sometimes the mean can be skewed by the presence of **outliers**, or data values that are significantly different as compared to the remainder of the dataset. In these instances, the *trimmed mean* is often used to provide a more representative measure of the center of the dataset, as we will discuss in the following section.

Mean

To calculate the mean, add the values of all the items in a dataset and divide by the number of items. For example, if the scores on your last three exams were 87, 92, and 73, then the mean score would be $\frac{87+92+73}{3} = 84$. If you had a large number of data values, you would proceed in the same way. For example, to calculate the mean value of 50 exam scores, add the 50 scores together and divide by 50. If the 50 scores add up to 4,050, for example, the mean score is $\frac{4050}{50}$, or 81.

In data science applications, you will encounter two types of datasets: sample data and population data. **Population data** represents all the outcomes or measurements that are of interest. **Sample data** represents outcomes or measurements collected from a subset, or part, of the population of interest. Of course in many applications, collecting data from an entire population is not practical or feasible, and so we often rely on sample data.

The notation $\bar{x}$ is used to indicate the **sample mean**, where the mean is calculated based on data taken from a sample. The notation $\sum x$ is used to denote the sum of the data values, and n is used to indicate the number of data values in the sample, also known as the **sample size**.

The sample mean can be calculated using the following formula:

$$\bar{x} = \frac{\sum x}{n}$$

The notation μ is used to indicate the **population mean**, where the mean is calculated based on data taken from the entire population, and N is used to indicate the number of data values in the population, also known as the **population size**. The population mean can be calculated using the following formula:

$$\mu = \frac{\sum x}{N}$$

The mean can also be determined by its frequency distribution. For every unique data value in the dataset, the **frequency distribution** gives the number of times, or **frequency**, that this unique value appears in the dataset. In this type of situation, the mean can be calculated by multiplying each distinct value by its frequency, summing these values, and then dividing this sum by the total number of data values. Here is the corresponding formula for the sample mean using the frequency distribution:

$$\bar{x} = \frac{\sum x \cdot f}{n}$$

When all the values in the dataset are unique, this reduces to the previous formula given for the sample mean.

EXAMPLE 3.1

Problem

During a clinical trial, a sample is taken of 10 patients and pulse rates are measured in beats per minute:

68, 92, 76, 51, 65, 83, 94, 72, 88, 59

Calculate the mean pulse rate for this sample.

Solution

Add the 10 data values, and the sum is 748. Divide this sum by the number of data values, which is 10. The result is:

$$\bar{x} = \frac{\sum x}{n} = \frac{748}{10} = 74.8$$

EXAMPLE 3.2

Problem

A college professor records the ages of 25 students in a data science class as shown:

Student Age	Number of Students (Frequency)
19	3
20	4
21	8
22	6
23	2
27	1
31	1
Total	**25**

Calculate the mean age for this sample of students.

Solution

Substitute the values from the table in the following formula:

$$\bar{x} = \frac{\sum x \cdot f}{n} = \frac{19 \cdot 3 + 20 \cdot 4 + 21 \cdot 8 + 22 \cdot 6 + 23 \cdot 2 + 27 \cdot 1 + 31 \cdot 1}{25} = \frac{541}{25} = 21.64$$

Trimmed Mean

A **trimmed mean** helps mitigate the effects of outliers, which are data values that are significantly different from most of the other data values in the dataset. In the dataset given in Example 3.1, a pulse rate of 35 or 120 would be considered outlier data values since these pulse rates are significantly different as compared to the rest of the values in Example 3.1. We will see that there is a formal method for determining outliers, and in fact there are several methods to identify outliers in a dataset.

The presence of outlier data values tends to disproportionally skew the mean and produce a potentially misleading result for the mean.

To calculate the trimmed mean for a dataset, first sort the data in ascending order (from smallest to largest). Then decide on a certain percentage of data values to be deleted from the lower and upper ends of the dataset. This might represent the extent of outliers in the dataset; trimmed mean percentages of 10% and 20% are common. Then delete the specified percentage of data values from both the lower end and upper end of the dataset. Then find the mean for the remaining undeleted data values.

As an example, to calculate a 10% trimmed mean, first sort the data values from smallest to largest. Then delete the lower 10% of the data values and delete the upper 10% of the data values. Then calculate the mean for the resulting dataset. Any outliers would tend to be deleted as part of the trimmed mean calculation, and thus the trimmed mean would then be a more representative measure of the center of the data for datasets containing outliers.

EXAMPLE 3.3

Problem

A real estate agent collects data on a sample of recently sold homes in a certain neighborhood, and the data are shown in the following dataset:

397900, 452600, 507400, 488300, 623400, 573200, 1689300, 403890, 612300, 599000, 2345800, 499000, 525000, 675000, 385000

1. Calculate the mean of the dataset.
2. Calculate a 20% trimmed mean rate for the dataset.

Solution

1. For the mean, add the 15 data values, and the sum is 10,777,090. Divide this sum by the number of data values, which is 15. The result is:

$$\bar{x} = \frac{\sum x}{n} = \frac{10{,}777{,}090}{15} = 718{,}472.70$$

2. For the trimmed mean, first order the data from smallest to largest. The sorted dataset is:

   385000, 397900, 403890, 452600, 488300, 499000, 507400, 525000, 573200, 599000, 612300, 623400, 675000, 1689300, 2345800

   Twenty percent of 15 data values is 3, and this indicates that 3 data values are to be deleted from each of the lower end and upper end of the dataset. The resulting 9 undeleted data values are:

   452600, 488300, 499000, 507400, 525000, 573200, 599000, 612300, 623400

   Then find the mean for the remaining data values. The sum of these 9 data values is 4,880,200. Divide this sum by the number of data values (9). The result is:

$$\bar{x} = \frac{\sum x}{n} = \frac{4{,}880{,}200}{9} = 542{,}244.40$$

Notice how the mean calculated in Part (a) is significantly larger as compared to the trimmed mean calculated in Part (b). The reason is the presence of several large outlier home prices. Once these outlier data values are removed by the trimmed mean calculation, the resulting trimmed mean is more representative of the typical home price in this neighborhood as compared to the mean.

Median

The median provides another measure of central tendency for a dataset. The **median** is generally a better measure of the central tendency when there are outliers (extreme values) in the dataset. Since the median focuses on the middle value of the ordered dataset, the median is preferred when outliers are present because the median is not affected by the numerical values of the outliers.

To determine the median of a dataset, first order the data from smallest to largest, and then find the middle value in the ordered dataset. For example, to find the median value of 50 exam scores, find the score that splits the data into two equal parts. The exam scores for 25 students will be below the median, and 25 students will have exam scores above the median.

If there is an odd number of data values in the dataset, then there will be one data value that represents the middle value, and this is the median. If there is an even number of data values in the dataset, then to find the median, add the two middle values together and divide by 2 (this is essentially finding the mean of the two middle values in the dataset).

EXAMPLE 3.4

Problem

The same dataset of pulse rates from Example 3.1 is:

68, 92, 76, 51, 65, 83, 94, 72, 88, 59

Calculate the median pulse rate for this sample.

Solution

First, order the 10 data values from smallest to largest. Divide this sum by the number of data values, which is 10. The result is:

51, 59, 65, 68, 72, 76, 83, 88, 92, 94

Since there is an even number of data values, add the two middle values together and divide by 2.
The two middle values are 72 and 76.

$$\text{Median } = \frac{72+76}{2} = \frac{148}{2} = 74$$

You can also quickly find the sample median of a dataset as follows.

Let n represent the number of data values in the sample.

- If n is odd, then the median is the data value in position $\frac{n+1}{2}$.
- If n is even, the median is the mean of the observations in position $\frac{n}{2}$ and position $\frac{n}{2}+1$.

For example, let's say a dataset has 25 data values. Since n is odd, to identify the position of the median, calculate $\frac{n+1}{2}$, which is $\frac{25+1}{2}$, or 13. This indicates that the median is located in the 13th data position.

As another example, let's say a dataset has 100 data values. Since n is even, to identify the position of the median, calculate $\frac{n}{2}$, which is $\frac{100}{2}$, which is 50, and also calculate $\frac{n}{2}+1$, which is $50+1$, which is 51. This indicates that the median is calculated as the mean of the 50th and 51st data values.

Mode

Another measure of center is the **mode**. The mode is the data value that occurs with the greatest frequency. If there are no repeating data values in a dataset, then there is no mode. If two data values occur with same greatest frequency, then there are two modes, and we say the data is bimodal. For example, assume that the weekly closing stock price for a technology stock, in dollars, is recorded for 20 consecutive weeks as follows:

50, 53, 59, 59, 63, 63, 72, 72, 72, 72, 72, 76, 78, 81, 83, 84, 84, 84, 90, 93

To find the mode, determine the most frequent score, which is 72, which occurs five times. Thus, the mode of this dataset is 72.

The mode can also be applied to non-numeric (qualitative) data, whereas the mean and the median can only be applied for numeric (quantitative) data. For example, a restaurant manager might want to determine the mode for responses to customer surveys on the quality of the service of a restaurant, as shown in Table 3.1.

Customer Service Rating	Number of Respondents
Excellent	267
Very Good	410
Good	392
Fair	107
Poor	18

Table 3.1 Customer Survey Results for Customer Survey Rating

Based on the survey responses, the mode is the Customer Service Rating of "Very Good," since this is the data value with the greatest frequency.

Influence of Outliers on Measures of Center

As mentioned earlier, when outliers are present in a dataset, the mean may not represent the center of the dataset, and the median will provide a better measure of center. The reason is that the median focuses on the middle value of the ordered dataset. Thus, any outliers at the lower end of the dataset or any outliers at the upper end of the dataset will not affect the median. Note: A formal method for identifying outliers is presented in Measures of Position when measures of position are discussed. The following example illustrates the point that the median is a better measure of central tendency when potential outliers are present.

EXAMPLE 3.5

Problem

Suppose that in a small company of 40 employees, one person earns a salary of $3 million per year, and the other 39 individuals each earn $40,000. Which is the better measure of center: the mean or the median?

Solution

The mean, in dollars, would be arrived at mathematically as follows:

$$\bar{x} = \frac{3{,}000{,}000 + 39(40{,}000)}{40} = 114{,}000$$

However, the median would be $40,000 since this is the middle data value in the ordered dataset. There are 39 people who earn $40,000 and one person who earns $3,000,000.

Notice that the mean is not representative of the typical value in the dataset since $114,000 is not reflective of the average salary for most employees (who are earning $40,000). The median is a much better measure of the "average" than the mean in this case because 39 of the values are $40,000 and one is $3,000,000. The data value of $3,000,000 is an outlier. The median result of $40,000 gives us a better sense of the center of the dataset.

Using Python for Measures of Center

We learned in What Are Data and Data Science? how the `DataFrame.describe()` method is used to

summarize data. Recall that the method `describe()` is defined for a DataFrame type object so should be called upon a DataFrame type variable (e.g. given a DataFrame `d`, use `d.describe()`).

Figure 3.2 shows the output of `DataFrame.describe()` on the "Movie Profit" dataset we used in What Are Data and Data Science?, movie_profit.csv. The mean and 50% quartile show the average and median of each column. For example, the average worldwide gross earnings are $410.14 million, and the median earnings are $309.35 million. Note that average and/or median of some columns are not as meaningful as the others. The first column—Unnamed: 0—was simply used as an identifier of each item in the dataset, so the average and mean of this column is not quite useful. `DataFrame.describe()` still computes the values because it can (and it does *not* care which column is meaningful to do so or not).

	Unnamed: 0	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million
count	966.00000	966.000000	966.000000	966.000000	966.000000
mean	483.50000	6.814286	117.506211	156.158975	410.140600
std	279.00448	0.894383	21.615612	110.629617	294.758791
min	1.00000	3.300000	69.000000	0.010000	176.600000
25%	242.25000	6.200000	101.250000	90.832500	223.277500
50%	483.50000	6.800000	116.000000	129.245000	309.345000
75%	724.75000	7.400000	130.000000	187.090000	472.645000
max	966.00000	9.200000	238.000000	936.660000	2847.400000

Figure 3.2 The Output of `DataFrame.describe()` with the Movie Profit Dataset

EXPLORING FURTHER

Working with Python

See the Python website (https://openstax.org/r/python) for more details on using, installing, and working with Python. See this additional documentation, for more specific information on the statistics module (https://openstax.org/r/docspython).

3.2 Measures of Variation

Learning Outcomes

By the end of this section, you should be able to:

- 3.2.1 Define and calculate the range, the variance, and the standard deviation for a dataset.
- 3.2.2 Use Python to calculate measures of variation for a dataset.

Providing some measure of the spread, or *variation*, in a dataset is crucial to a comprehensive summary of the dataset. Two datasets may have the same mean but can exhibit very different spread, and so a measure of dispersion for a dataset is very important. While measures of central tendency (like mean, median, and mode) describe the center or average value of a distribution, measures of dispersion give insights into how much individual data points deviate from this central value.

The following two datasets are the exam scores for a group of three students in a biology course and in a statistics course.

Dataset A: Exam scores for students in a biology course: 40, 70, 100
Dataset B: Exam scores for students in a statistics course: 69, 70, 71

Notice that the mean score for both Dataset A and Dataset B is 70.

However, the datasets are significantly different from one another:

Dataset A has larger variability where one student scored 30 points below the mean and another student scored 30 points above the mean.
Dataset B has smaller variability where the exam scores are much more tightly clustered around the mean of 70.

This example illustrates that publishing the mean of a dataset is often inadequate to fully communicate the characteristics of the dataset. Instead, data scientists will typically include a measure of variation as well.

The three primary measures of variability are range, variance, and standard deviation, and these are described next.

Range

Range is a measure of dispersion for a dataset that is calculated by subtracting the minimum from the maximum of the dataset:

$$\text{Range} = \text{Max} - \text{Min}$$

Range is a straightforward calculation but makes use of only two of the data values in a dataset. The range can also be affected by outliers.

EXAMPLE 3.6

Problem

Calculate the range for Dataset A and Dataset B:

Dataset A: Exam scores for students in a biology course: 40, 70, 100
Dataset B: Exam scores for students in a statistics course: 69, 70, 71

Solution

For Dataset A, the maximum data value is 100 and the minimum data value is 40.
The range is then calculated as:

$$\begin{aligned} \text{Range} &= \text{Max} - \text{Min} \\ \text{Range} &= 100 - 40 \\ \text{Range} &= 60 \end{aligned}$$

For Dataset B, the maximum data value is 71 and the minimum data value is 69.
The range is then calculated as:

$$\begin{aligned} \text{Range} &= \text{Max} - \text{Min} \\ \text{Range} &= 71 - 69 \\ \text{Range} &= 2 \end{aligned}$$

The range clearly indicates that there is much less spread in Dataset B as compared to Dataset A.

One drawback to the use of the range is that it doesn't take into account every data value. The range only uses two data values from the dataset: the minimum (min) and the maximum (max). Also the range is influenced by outliers since an outlier might appear as a minimum or maximum data value and thus skew the results. For these reasons, we typically use other measures of variation, such as variance or standard deviation.

Variance

The **variance** provides a measure of the spread of data values by using the squared deviations from the mean. The more the individual data values differ from the mean, the larger the variance.

A financial advisor might use variance to determine the volatility of an investment and therefore help guide financial decisions. For example, a more cautious investor might opt for investments with low volatility.

The formula used to calculate variance also depends on whether the data is collected from a sample or a population. The notation s^2 is used to represent the *sample variance*, and the notation σ^2 is used to represent the *population variance*.

Formula for the sample variance:

$$s^2 = \frac{\sum(x - \bar{x})^2}{n - 1}$$

Formula for the population variance:

$$\sigma^2 = \frac{\sum(x - \mu)^2}{N}$$

In these formulas:
x represents the individual data values
$\bar{x}$ represents the sample mean
n represents the sample size
μ represents the population mean
N represents the population size

ALTERNATE FORMULA FOR VARIANCE

An alternate formula for the variance is available. It is sometimes used for more efficient computations:

$$\sigma^2 = \frac{\sum x^2}{N} - \mu^2$$

In the formulas for sample variance and population variance, notice the denominator for the sample variance is $n - 1$, whereas the denominator for the population variance is N. The use of $n - 1$ in the denominator of the sample variance is used to provide the best estimate for the population variance, in the sense that if repeated samples of size n are taken and the sample mean computed each time, then the average of those sample means will tend to the population mean as the number of repeated samples increase.

It is important to note that in many data science applications, population data is unavailable, and so we typically calculate the sample variance. For example, if a researcher wanted to estimate the percentage of smokers for all adults in the United States, it would be impractical to collect data from every adult in the United States.

Notice that the sample variance is a sum of squares. Its units of measurement are squares of the units of measurement of the original data. Since these square units are different than the units in the original data, this can be confusing. By contrast, standard deviation is measured in the same units as the original dataset, and thus the standard deviation is more commonly used to measure the spread of a dataset.

Standard Deviation

The **standard deviation** of a dataset provides a numerical measure of the overall amount of variation *in a dataset in the same units as the data*; it can be used to determine whether a particular data value is close to or

far from the mean, relative to the typical distance from the mean.

The standard deviation is always positive or zero. It is small when the data values are all concentrated close to the mean, exhibiting little variation, or spread. It is larger when the data values are spread out more from the mean, exhibiting more variation. A smaller standard deviation implies less variability in a dataset, and a larger standard deviation implies more variability in a dataset.

Suppose that we are studying the variability of two companies (A and B) with respect to employee salaries. The average salary for both companies is $60,000. For Company A, the standard deviation of salaries is $8,000, whereas the standard deviation for salaries for Company B is $19,000. Because Company B has a higher standard deviation, we know that there is more variation in the employee salaries for Company B as compared to Company A.

There are two different formulas for calculating standard deviation. Which formula to use depends on whether the data represents a sample or a population. The notation s is used to represent the sample standard deviation, and the notation σ is used to represent the population standard deviation. In the formulas shown, $\bar{x}$ is the sample mean, μ is the population mean, n is the sample size, and N is the population size.

Formula for the sample standard deviation:

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

Formula for the population standard deviation:

$$\sigma = \sqrt{\frac{\sum(x - \mu)^2}{N}}$$

Notice that the sample standard deviation is calculated as the square root of the variance. This means that once the sample variance has been calculated, the sample standard deviation can then be easily calculated as the square root of the sample variance, as in Example 3.7.

EXAMPLE 3.7

Problem

A biologist calculates that the sample variance for the amount of plant growth for a sample of plants is 8.7 cm^2. Calculate the sample standard deviation.

Solution

The sample standard deviation (s) is calculated as the square root of the variance.

$$s = \sqrt{s^2} = \sqrt{8.7} = 2.9 \text{ cm}$$

EXAMPLE 3.8

Problem

Assume the sample variance (s^2) for a dataset is calculated as 42.2. Based on this, calculate the sample standard deviation.

Solution

The sample standard deviation (s) is calculated as the square root of the variance.

$$s = \sqrt{s^2} = \sqrt{42.2} = 6.5 \text{ years}$$

This result indicates that the standard deviation is about 6.5 years.

Notice that the sample variance is the square of the sample standard deviation, so if the sample standard deviation is known, the sample variance can easily be calculated.

USE OF TECHNOLOGY FOR CALCULATING MEASURES OF VARIABILITY

Due to the complexity of calculating variance and standard deviation, technology is typically utilized to calculate these measures of variability. For example, refer to the examples shown in Coefficient of Variation on using Python for measures of variation.

Coefficient of Variation

A data scientist might be interested in comparing variation with different units of measurement of different means, and in these scenarios the **coefficient of variation (CV)** can be used. The coefficient of variation measures the variation of a dataset by calculating the standard deviation as a percentage of the mean. Note: coefficient of variation is typically expressed in a percentage format.

$$\begin{aligned} \text{CV} &= \frac{\sigma}{\mu} \times 100\% \\ \text{Sample CV} &= \frac{s}{\bar{x}} \times 100\% \end{aligned}$$

EXAMPLE 3.9

Problem

Compare the relative variability for Company A versus Company B using the coefficient of variation, based on the following sample data:

Company A: $\text{Sample Mean } = \$68{,}000, \text{ Sample Standard Deviation } = \$9{,}200$

Company B: $\text{Sample Mean } = \$71{,}000, \text{ Sample Standard Deviation } = \$6{,}400$

Solution

Calculate the coefficient of variation for each company:

$$\text{CV for Company A} = \frac{s}{\bar{x}} \times 100\% = \frac{9{,}200}{68{,}000} \times 100\% = 13.5\%$$

$$\text{CV for Company B} = \frac{s}{\bar{x}} \times 100\% = \frac{6{,}400}{71{,}000} \times 100\% = 9.0\%$$

Company A exhibits more variability relative to the mean as compared to Company B.

Using Python for Measures of Variation

`DataFrame.describe()` computes standard deviation as well on each column of a dataset. The **std** lists the standard deviation of each column (See Figure 3.3).

	Unnamed: 0	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million
count	966.00000	966.000000	966.000000	966.000000	966.000000
mean	483.50000	6.814286	117.506211	156.158975	410.140600
std	279.00448	0.894383	21.615612	110.629617	294.758791
min	1.00000	3.300000	69.000000	0.010000	176.600000
25%	242.25000	6.200000	101.250000	90.832500	223.277500
50%	483.50000	6.800000	116.000000	129.245000	309.345000
75%	724.75000	7.400000	130.000000	187.090000	472.645000
max	966.00000	9.200000	238.000000	936.660000	2847.400000

Figure 3.3 The Output of `DataFrame.describe()` with the Movie Profit Dataset

3.3 Measures of Position

Learning Outcomes

By the end of this section, you should be able to:

- 3.3.1 Define and calculate percentiles, quartiles, and z-scores for a dataset.
- 3.3.2 Use Python to calculate measures of position for a dataset.

Common measures of position include percentiles and quartiles as well as z-scores, all of which are used to indicate the relative location of a particular datapoint.

Percentiles

If a student scores 47 on a biology exam, it is difficult to know if the student did well or poorly compared to the population of all other students taking the exam. **Percentiles** provide a way to assess and compare the distribution of values and the position of a specific data point in relation to the entire dataset by indicating the percentage of data points that fall below it. Specifically, a percentile is a value on a scale of one hundred that indicates the percentage of a distribution that is equal to or below it. Let's say the student learns they scored in the 90th percentile on the biology exam. This percentile indicates that the student has an exam score higher than 90% of all other students taking the test. This is the same as saying that the student's score places the student in the top 10% of all students taking the biology test. Thus, this student scoring in the 90th percentile did very well on the exam, even if the actual score was 47.

To calculate percentiles, the data must be ordered from smallest to largest and then the ordered data divided into hundredths. If you score in the 80th percentile on an aptitude test, that does not necessarily mean that you scored 80% on the test. It means that 80% of the test scores are the same as or less than your score and the remaining 20% of the scores are the same as or greater than your score.

Percentiles are useful for comparing many types of values. For example, a stock market mutual fund might report that the performance for the fund over the past year was in the 80th percentile of all mutual funds in the peer group. This indicates that the fund performed better than 80% of all other funds in the peer group. This also indicates that 20% of the funds performed better than this particular fund.

To calculate percentiles for a *specific data value* in a dataset, first order the dataset from smallest to largest and count the number of data values in the dataset. Locate the measurement of interest and count how many data values fall below the measurement. Then the percentile for the measurement is calculated as follows:

$$\text{Percentile} = \frac{\text{number of data values below the measurement}}{\text{total number of data values}} \times 100\% = \frac{n}{N} \times 100\%$$

EXAMPLE 3.10

Problem

The following ordered dataset represents the scores of 15 employees on an aptitude test:

51, 63, 65, 68, 71, 75, 75, 77, 79, 82, 88, 89, 89, 92, 95

Determine the percentile for the employee who scored 88 on the aptitude test.

Solution

There are 15 data values in total, and there are 10 data values below 88.

$$\text{Percentile } = \frac{\text{number of data values below the measurement}}{\text{total number of data values}} \times 100\% = \frac{10}{15} \times 100\% = 67\text{th percentile}$$

Quartiles

While percentiles separate data into 100 equal parts, **quartiles** separate data into quarters, or four equal parts. To find the quartiles, first find the median, or second quartile. The first quartile, Q_1, is the middle value, or median, of the lower half of the data, and the third quartile, Q_3, is the middle value of the upper half of the data.

Note the following correspondence between quartiles and percentiles:

- The first quartile corresponds to the 25th percentile.
- The second quartile (which is the median) corresponds to the 50th percentile.
- The third quartile corresponds to the 75th percentile.

EXAMPLE 3.11

Problem

Consider the following ordered dataset, which represents the time in seconds for an athlete to complete a 40-yard run:

5.4, 6.0, 6.3, 6.8, 7.1, 7.2, 7.4, 7.5, 7.9, 8.2, 8.7

Solution

The median, or second quartile, is the middle value in this dataset, which is 7.2. Notice that 50% of the data values are below the median, and 50% of the data values are above the median. The lower half of the data values are 5.4, 6.0, 6.3, 6.8, 7.1. Note that these are the data values below the median. The upper half of the data values are 7.4, 7.5, 7.9, 8.2, 8.7, which are the data values above the median.

To find the first quartile, Q_1, locate the middle value of the lower half of the data (5.4, 6.0, 6.3, 6.8, 7.1). The middle value of the lower half of the dataset is 6.3. Notice that one-fourth, or 25%, of the data values are below this first quartile, and 75% of the data values are above this first quartile.

To find the third quartile, Q_3, locate the middle value of the upper half of the data (7.4, 7.5, 7.9, 8.2, 8.7). The middle value of the upper half of the dataset is 7.9. Notice that one-fourth, or 25%, of the data values are above this third quartile, and 75% of the data values are below this third quartile.

Thus, the quartiles Q_1, Q_2, Q_3 for this dataset are 6.3, 7.2, 7.9, respectively.

The **interquartile range (IQR)** is a number that indicates the spread of the middle half, or the middle 50%, of the data. It is the difference between the third quartile, Q_3, and the first quartile, Q_1.

$$\text{IQR} = Q_3 - Q_1$$

Note that the IQR provides a measure of variability that excludes outliers.

In Example 3.11, the IQR can be calculated as:

$$\text{IQR} = Q_3 - Q_1 = 7.9 - 6.3 = 1.6$$

Quartiles and the IQR can be used to flag possible outliers in a dataset. For example, if most employees at a company earn about $50,000 and the CEO of the company earns $2.5 million, then we consider the CEO's salary to be an outlier data value because this salary is significantly different from all the other salaries in the dataset. An outlier data value can also be a value much lower than the other data values in a dataset, so if one employee only makes $15,000, then this employee's low salary might also be considered an outlier.

To detect outliers, you can use the quartiles and the IQR to calculate a lower and an upper bound for outliers. Then any data values below the lower bound or above the upper bound will be flagged as outliers. These data values should be further investigated to determine the nature of the outlier condition and whether the data values are valid or not.

To calculate the lower and upper bounds for outliers, use the following formulas:

$$\begin{aligned} \text{Lower Bound for Outliers} &= Q_1 - (1.5 \cdot \text{IQR}) \\ \text{Upper Bound for Outliers} &= Q_3 + (1.5 \cdot \text{IQR}) \end{aligned}$$

These formulas typically use 1.5 as a cutoff value to identify outliers in a dataset.

EXAMPLE 3.12

Problem

Calculate the IQR for the following 13 home prices and determine if any of the home prices values are potential outliers. Data values are in US dollars.

389950, 230500, 158000, 479000, 639000, 114950, 5500000, 387000, 659000, 529000, 575000, 488800, 1095000

Solution

Order the data from smallest to largest.

114950, 158000, 230500, 387000, 389950, 479000, 488800, 529000, 575000, 639000, 659000, 1095000, 5500000

First, determine the median of the dataset. There are 13 data values, so the median is the middle data value, which is 488,800.

Next, calculate the Q_1 and Q_3.

For the first quartile, look at the data values below the median. The two middle data values in this lower half of the data are 230,500 and 387,000. To determine the first quartile, find the mean of these two data values.

For the third quartile, look at the data values above the median. The two middle data values in this upper half of the data are 639,000 and 659,000. To determine the third quartile, find the mean of these two data values.

$$\begin{aligned} Q_1 &= \frac{230{,}500+387{,}000}{2} = 308{,}750 \\ Q_3 &= \frac{639{,}000+659{,}000}{2} = 649{,}000 \end{aligned}$$

Now, calculate the interquartile range (IQR):

$$\text{IQR } = 649{,}000 - 308{,}750 = 340{,}250$$

Calculate the value of 1.5 interquartile range (IQR):

$$(1.5)(\text{IQR }) = (1.5)(340{,}250) = 510{,}375$$

Calculate the lower and upper bound for outliers:

$$\begin{aligned} \text{Lower Bound } &= Q_1 - (1.5)(\text{IQR }) = 308{,}750 - 510{,}375 = -201{,}625 \\ \text{Upper Bound } &= Q_3 + (1.5)(\text{IQR }) = 649{,}000 + 510{,}375 = 1{,}159{,}375 \end{aligned}$$

The lower bound for outliers is −201,625. Of course, no home price is less than −201,625, so no outliers are present for the lower end of the dataset.

The upper bound for outliers is 1,159,375. The data value of 5,500,000 is greater than the upper bound of 1,159,375. Therefore, the home price of $5,500,000 is a potential outlier. This is important because the presence of outliers could potentially indicate data errors or some other anomalies in the dataset that should be investigated. For example, there may have been a data entry error and a home price of $550,000 was erroneously entered as $5,500,000.

z-scores

The z**-score** is a measure of the position of an entry in a dataset that makes use of the mean and standard deviation of the data. It represents the number of standard deviations by which a data value differs from the mean. For example, suppose that in a certain neighborhood, the mean selling price of a home is $350,000 and the standard deviation is $40,000. A particular home sells for $270,000. Based on the selling price of this home, we can calculate the relative standing of this home compared to other home sales in the same neighborhood.

The corresponding z-score of a measurement considers the given measurement in relation to the mean and standard deviation for the entire population. The formula for a z-score calculation is as follows:

$$z = \frac{x - \mu}{\sigma}$$

Where:
x is the measurement
μ is the mean
σ is the standard deviation

Notice that when a measurement is below the mean, the corresponding z-score will be a negative value. If the measurement is exactly equal to the mean, the corresponding z-score will be zero. If the measurement is above the mean, the corresponding z-score will be a positive value.

z-scores can also be used to identify outliers. Since z-scores measure the number of standard deviations from the mean for a data value, a z-score of 3 would indicate a data value that is 3 standard deviations above the mean. This would represent a data value that is significantly displaced from the mean, and typically, a z-score less than −3 or a z-score greater than +3 can be used to flag outliers.

EXAMPLE 3.13

Problem

For the home example in Example 3.12, the x value is the home price of $270,000, the mean μ is $350,000, and the standard deviation σ is $40,000. Calculate the z-score.

Solution

The z-score can be calculated as follows:

$$z = \frac{x - \mu}{\sigma} = \frac{270{,}000 - 350{,}000}{40{,}000} = \frac{-80{,}000}{40{,}000} = -2$$

This z-score of −2 indicates that the selling price for this home is 2 standard deviations below the mean, which represents a data value that is significantly below the mean.

Using Python to Calculate Measures of Position for a Dataset

`DataFrame.describe()` computes different measures of position as well on each column of a dataset. See min, 25%, 50%, 75%, and max in Figure 3.4.

	Unnamed: 0	Rating	Duration	US_Gross_Million	Worldwide_Gross_Million
count	966.00000	966.000000	966.000000	966.000000	966.000000
mean	483.50000	6.814286	117.506211	156.158975	410.140600
std	279.00448	0.894383	21.615612	110.629617	294.758791
min	1.00000	3.300000	69.000000	0.010000	176.600000
25%	242.25000	6.200000	101.250000	90.832500	223.277500
50%	483.50000	6.800000	116.000000	129.245000	309.345000
75%	724.75000	7.400000	130.000000	187.090000	472.645000
max	966.00000	9.200000	238.000000	936.660000	2847.400000

Figure 3.4 The Output of `DataFrame.describe()` with the Movie Profit Dataset

3.4 Probability Theory

Learning Outcomes

By the end of this section, you should be able to:

- 3.4.1 Describe the basic concepts of probability and apply these concepts to real-world applications in data science.
- 3.4.2 Apply conditional probability and Bayes' Theorem.

Probability is a numerical measure that assesses the likelihood of occurrence of an event. Probability applications are ubiquitous in data science since many decisions in business, science, and engineering are based on probability considerations. We all use probability calculations every day as we decide, for instance, whether to take an umbrella to work, the optimal route for a morning commute, or the choice of a college major.

Basic Concepts of Probability

We have all used probability in one way or another on a day-to-day basis. Before leaving the house, you might

want to know the probability of rain. The probability of obtaining heads on one flip of a coin is one-half, or 0.5.

A data scientist is in interested in expressing probability as a number between 0 and 1 (inclusive), where 0 indicates impossibility (the event will not occur) and 1 indicates certainty (the event will occur). The probability of an event falling between 0 and 1 reflects the degree of uncertainty associated with the event.

Here is some terminology we will be using in probability-related analysis:

- An **outcome** is the result of a single trial in a probability experiment.
- The **sample space** is the set of all possible outcomes in a probability experiment.
- An **event** is some subset of the sample space. For example, an event could be rolling an even number on a six-sided die. This event corresponds to three outcomes, namely rolling a 2, 4, or 6 on the die.

To calculate probabilities, we can use several approaches, including relative frequency probability, which is based on actual data, and theoretical probability, which is based on theoretical conditions.

Relative Frequency Probability

Relative frequency probability is a method of determining the likelihood of an event occurring based on the observed frequency of its occurrence in a given sample or population. A data scientist conducts or observes a procedure and determines the number of times a certain Event A occurs. The probability of Event A, denoted as $P(A)$, is then calculated based on data that has been collected from the experiment, as follows:

$$P(A) = \frac{\text{number of times Event } A \text{ has occured}}{\text{number of times the procedure was repeated}}$$

EXAMPLE 3.14

Problem

A polling organization asks a sample of 400 people if they are in favor of increased funding for local schools; 312 of the respondents indicate they are in favor of increased funding. Calculate the probability that a randomly selected person will be in favor of increased funding for local schools.

Solution

Using the data collected from this polling, a total of 400 people were asked the question, and 312 people were in favor of increased school funding. The probability for a randomly selected person being in favor of increased funding can then be calculated as follows (notice that Event A in this example corresponds to the event that a person is in favor of the increased funding):

$$\begin{aligned} P(A) &= \text{Probability of In Favor} \\ &= \frac{\text{number of times Event } A \text{ has occurred}}{\text{number of times the procedure was repeated}} = \frac{312}{400} = 0.78 = 78\% \end{aligned}$$

EXAMPLE 3.15

Problem

A medical patient is told they need knee surgery, and they ask the doctor for an estimate of the probability of success for the surgical procedure. The doctor reviews data from the past two years and determines there were 200 such knee surgeries performed and 188 of them were successful. Based on this past data, the doctor calculates the probability of success for the knee surgery (notice that Event A in this example corresponds to the event that a patient has a successful knee surgery result).

Solution

Using the data collected from the past two years, there were 200 surgeries performed, with 188 successes. The probability can then be calculated as:

$$P(A) = \text{Probability of Success} = \frac{\text{number of times Event } A \text{ has occured}}{\text{number of times the procedure was repeated}} = \frac{188}{200} = 0.94$$

The doctor informs the patient that there is a 94% chance of success for the pending knee surgery.

Theoretical Probability

Theoretical probability is the method used when the outcomes in a probability experiment are equally likely—that is, under theoretical conditions.

The formula used for theoretical probability is similar to the formula used for **empirical probability**. Theoretical probability considers all the possible outcomes for an experiment that are known ahead of time so that past data is not needed in the calculation for theoretical probability.

$$\text{Theoretical Probability} = \frac{\text{number of outcomes for the event of interest}}{\text{total number of outcomes in the sample space}}$$

For example, the theoretical probability of rolling an even number when rolling a six-sided die is $\frac{3}{6}$ (which is $\frac{1}{2}$, or 0.5). There are 3 outcomes corresponding to rolling an even number, and there are 6 outcomes total in the sample space. Notice this calculation can be done without conducting any experiments since the outcomes are equally likely.

EXAMPLE 3.16

Problem

A student is working on a multiple-choice question that has 5 possible answers. The student does not have any idea about the correct answer, so the student randomly guesses. What is the probability that the student selects the correct answer?

Solution

Since the student is guessing, each answer choice is equally likely to be selected. There is 1 correct answer out of 5 possible choices. The probability of selecting the correct answer can be calculated as:

$$\begin{aligned}\text{Probability of Correct Answer} &= \frac{\text{number of outcomes for the event of interest}}{\text{total number of outcomes in the sample space}} \\ &= \frac{1}{5} = 0.20 = 20\%\end{aligned}$$

Notice in Example 3.16 that probabilities can be written as fractions, decimals, or percentages.

Also note that any probability must be between 0 and 1 inclusive. An event with a probability of zero will never occur, and an event with a probability of 1 is certain to occur. A probability greater than 1 is not possible, and a negative probability is not possible.

Complement of an Event

The **complement of an event** is the set of all outcomes in the sample space that are *not* included in the event. The complement of Event A is usually denoted by A' (*A* prime). To find the probability of the complement of Event A, subtract the probability of Event A from 1.

$$P(A') = 1 - P(A)$$

EXAMPLE 3.17

Problem

A company estimates that the probability that an employee will provide confidential information to a hacker is 0.1%. Determine the probability that an employee will not provide any confidential information during a hacking attempt.

Solution

Let Event A be the event that the employee will provide confidential information to a hacker. Then the complement of this Event A' is the event that an employee will not provide any confidential information during a hacking attempt.

$$\begin{aligned} P(A') &= 1 - P(A) \\ P(A') &= 1 - 0.001 = 0.999 \end{aligned}$$

There is a 99.9% probability that an employee will not provide any confidential information during a hacking attempt.

Conditional Probability and Bayes' Theorem

Data scientists are often interested in determining conditional probabilities, or the occurrence of one event that is conditional or dependent on another event. For example, a medical researcher might be interested to know if an asthma diagnosis for a patient is dependent on the patient's exposure to air pollutants. In addition, when calculating conditional probabilities, we can sometimes revise a probability estimate based on additional information that is obtained. As we'll see in the following section, Bayes' Theorem allows new information to be used to refine a probability estimate.

Conditional Probability

A **conditional probability** is the probability of an event given that another event has already occurred. The notation for conditional probability is $P(A|B)$, which denotes the probability of Event A, given that Event B has occurred. The vertical line between A and B denotes the "given" condition. (In this notation, the vertical line does not denote division).

For example, we might want to know the probability of a person getting a parking ticket given that a person did not put any money in a parking meter. Or a medical researcher might be interested in the probability of a patient developing heart disease given that the patient is a smoker.

If the occurrence of one event affects the probability of occurrence for another event, we say that the events are *dependent*; otherwise, the events are *independent*. **Dependent events** are events where the occurrence of one event affects the probability of occurrence of another event. **Independent events** are events where the probability of occurrence of one event is *not* affected by the occurrent of another event. The dependence of events has important implications in many fields such as marketing, engineering, psychology, and medicine.

EXAMPLE 3.18

Problem

Determine if the two events are dependent or independent:

1. Rolling a 3 on one roll of a die, rolling a 4 on a second roll of a die

2. Obtaining heads on one flip of a coin and obtaining tails on a second flip of a coin
3. Selecting five basketball players from a professional basketball team and a player's height is greater than 6 feet
4. Selecting an Ace from a deck of 52 cards, returning the card back to the original stack, and then selecting a King
5. Selecting an Ace from a deck of 52 cards, not returning the card back to the original stack, and then selecting a King

Solution

1. The result of one roll does not affect the result for the next roll, so these events are independent.
2. The results of one flip of the coin do not affect the results for any other flip of the coin, so these events are independent.
3. Typically, basketball players are tall individuals, and so they are more likely to have heights greater than 6 feet as opposed to the general public, so these events are dependent.
4. By selecting an Ace from a deck of 52 cards and then replacing the card, this restores the deck of cards to its original state, so the probability of selecting a King is not affected by the selection of the Ace. So these events are independent.
5. By selecting an Ace from a deck of 52 cards and then not replacing the card, this will result in only 51 cards remaining in the deck. Thus, the probability of selecting a King is affected by the selection of the Ace, so these events are dependent.

There are several ways to use conditional probabilities in data science applications.

Conditional probability can be defined as follows:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}, \text{ where } P(B) \neq 0$$

When assessing the conditional probability of $P(A|B)$, if the two events are independent, this indicates that Event A is not affected by the occurrence of Event B, so we can write that $P(A|B) = P(A)$ for independent events.

If we determine that the $P(A|B)$ is not equal to $P(A)$, this indicates that the events are dependent.

$P(A|B) = P(A)$ implies independent events, where $P(B) \neq 0$.

$P(A|B) \neq P(A)$ implies dependent events.

EXAMPLE 3.19

Problem

Table 3.2 shows the number of nursing degrees and non-nursing degrees at a university for a specific year, and the data is broken out by age groups. Calculate the probability that a randomly chosen graduate obtained a nursing degree, given that the graduate is in the age group of 23 and older.

Age Group	Nursing Degrees	Non-Nursing Degrees	Total
22 and under	1036	1287	2323
23 and older	986	932	1918
Total	2022	2219	4241

Table 3.2 Number of Nursing and Non-Nursing Degrees at a University by Age Group

Solution

Since we are given that the group of interest are those graduates in the age group of 23 and older, focus only on the second row in the table.

Looking only at the second row in the table, we are interested in the probability that a randomly chosen graduate obtained a nursing degree. The reduced sample space consists of 1,918 graduates, and 986 of them received nursing degrees. So the probability can be calculated as:

$$P(\text{nursing degree} \mid \text{age group of 23 and older}) = \frac{986}{1{,}918} = 0.514$$

Another method to analyze this example is to rewrite the conditional probability using the equation for $P(A \text{ and } B)$, as follows:

$$\begin{aligned} P(A \text{ and } B) &= P(A) \cdot P(B|A) \\ \text{rewrite as: } P(B|A) &= \frac{P(A \text{ and } B)}{P(A)} \end{aligned}$$

We can now use this equation to calculate the probability that a randomly chosen graduate obtained a nursing degree, given that the graduate is in the age group of 23 and older. The probability of A and B is the probability that a graduate received a nursing degree and is also in the age group of 23 and older. From the table, there are 986 graduates who earned a nursing degree and are also in the age group of 23 and older. Since this number of graduates is out of the total sample size of 4,241, we can write the probability of Events A and B as:

$$P(A \text{ and } B) = \frac{986}{4{,}241}$$

We can also calculate the probability that a graduate is in the age group of 23 and older. From the table, there are 1,918 graduates in this age group out of the total sample size of 4,241, so we can write the probability for Event A as:

$$P(A) = \frac{1{,}918}{4{,}241}$$

Next, we can substitute these probabilities into the formula for $P(B|A)$, as follows:

$$P(B|A) = \frac{P(A \text{ and } B)}{P(A)} = \frac{\frac{986}{4{,}241}}{\frac{1{,}918}{4{,}241}} = \frac{986}{4{,}241} \cdot \frac{4{,}241}{1{,}918} = \frac{986}{1{,}918} = 0.514$$

Probability of *At Least One*

The probability of *at least one occurrence* of an event is often of interest in many data science applications. For example, a doctor might be interested to know the probability that at least one surgery to be performed this week will involve an infection of some type.

The phrase "at least one" implies the condition of one or more successes. From a sample space perspective, one or more successes is the complement of "no successes." Using the complement rule discussed earlier, we can write the following probability formula:

$$P(\text{at least one success}) = 1 - P(\text{no successes})$$

As an example, we can find the probability of rolling a die 3 times and obtaining at least one four on any of the rolls. This can be calculated by first finding the probability of not observing a four on any of the rolls and then subtracting this probability from 1. The probability of not observing a four on a roll of the die is 5/6. Thus, the probability of rolling a die 3 times and obtaining at least one four on any of the rolls is $1 - \left(\frac{5}{6}\right)^3 = 0.421$.

EXAMPLE 3.20

Problem

From past data, hospital administrators determine the probability that a knee surgery will be successful is 0.89.

1. During a certain day, the hospital schedules four knee surgeries to be performed. Calculate the probability that all four of these surgeries will be successful.
2. Calculate the probability that none of these knee surgeries will be successful.
3. Calculate the probability that at least one of the knee surgeries will be successful.

Solution

1. For all four surgeries to be successful, we can interpret that as the first surgery will be successful, and the second surgery will be successful, and the third surgery will be successful, and the fourth surgery will be successful. Since the probability of success for one knee surgery does not affect the probability of success for another knee surgery, we can assume these events are independent. Based on this, the probability that all four surgeries will be successful can be calculated using the probability formula for $P(A \text{ and } B)$ by multiplying the probabilities together:

$$\begin{aligned} P(A \text{ and } B \text{ and } C \text{ and } D) &= P(A) \cdot P(B) \cdot P(C) \cdot P(D) \\ &= 0.89 \cdot 0.89 \cdot 0.89 \cdot 0.89 = 0.627 \end{aligned}$$

   There is about a 63% chance that all four knee surgeries will be successful.

2. The probability that a knee surgery will be unsuccessful can be calculated using the complement rule. If the probability of a successful surgery is 0.89, then the probability that the surgery will be unsuccessful is 0.11:

$$P(A') = 1 - P(A) = 1 - 0.89 = 0.11$$

   Based on this, the probability that all four surgeries will be unsuccessful can be calculated using the probability formula for $P(A \text{ and } B)$ by multiplying the probabilities together:

$$\begin{aligned}(A' \text{ and } B' \text{ and } C' \text{ and } D') &= P(A') \cdot P(B') \cdot P(C') \cdot P(D') \\ &= 0.11 \cdot 0.11 \cdot 0.11 \cdot 0.11 = 0.00015\end{aligned}$$

Since this is a very small probability, it is very unlikely that none of the surgeries will be successful.

3. To calculate the probability that at least one of the knee surgeries will be successful, use the probability formula for "at least one," which is calculated as the complement of the event "none are successful."

$$\begin{aligned}P(\text{at least one success}) &= 1 - P(\text{no successes}) \\ P(\text{at least one success}) &= 1 - 0.00015 = 0.99985\end{aligned}$$

This indicates there is a very high probability that at least one of the knee surgeries will be successful.

Bayes' Theorem

Bayes' Theorem is a statistical technique that allows for the revision of probability estimates based on new information or evidence that allows for more accurate and efficient decision-making in uncertain situations. Bayes' Theorem is often used to help assess probabilities associated with medical diagnoses such as the probability a patient will develop cancer based on test screening results. This can be important in medical analysis to help assess the impact of a *false positive*, which is the scenario where the patient does not have the ailment but the screening test gives a false indication that the patient does have the ailment.

Bayes' Theorem allows the calculation of the conditional probability $P(A|B)$. There are several forms of Bayes' Theorem, as shown:

$$\begin{aligned}P(A|B) &= \frac{P(A) \cdot P(B|A)}{P(B)} \\ P(A|B) &= \frac{P(A) \cdot P(B|A)}{P(A) \cdot P(B|A) + P(A') \cdot P(B|A')}\end{aligned}$$

EXAMPLE 3.21

Problem

Assume that a certain type of cancer affects 3% of the population. Call the event that a person has cancer "Event A," so:

$$P(A) = 0.03$$

A patient can undergo a screening test for this type of cancer. Assume the probability of a true positive from the screening test is 75%, which indicates that probability that a person has a positive test result given that they actually have cancer is 0.75. Also assume the probability of a false positive from the screening test is 15%, which indicates that probability that a person has a positive test result given that they do not have cancer is 0.15.

A medical researcher is interested in calculating the probability that a patient actually has cancer given that the screening test shows a positive result.

The researcher is interested in calculating $P(A|B)$, where Event A is the person actually has cancer and Event B is the event that the person shows a positive result in the screening test. Use Bayes' Theorem to calculate this conditional probability.

Solution

From the example, the following probabilities are known:

$$\begin{aligned} P(A) &= 0.03 \\ P(A') &= 0.97 \end{aligned}$$

The conditional probabilities can be interpreted as follows:

$$\begin{aligned} P(B|A) &= P(\text{positive test result} \mid \text{patient has cancer}) = 0.75 \\ P(B|A') &= P(\text{positive test result} \mid \text{patient does not have cancer}) = 0.15 \end{aligned}$$

Substituting these probabilities into the formula for Bayes' Theorem results in the following:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(A) \cdot P(B|A) + P(A') \cdot P(B|A')}$$

$$P(\text{ patient has cancer} \mid \text{positive test result}) = \frac{0.03 \cdot 0.75}{0.03 \cdot 0.75 + 0.97 \cdot 0.15} = 0.134$$

This result from Bayes' Theorem indicates that even if a patient receives a positive test result from the screening test, this does not imply a high likelihood that the patient has cancer. There is only a 13% chance that the patient has cancer given a positive test result from the screening test.

3.5 Discrete and Continuous Probability Distributions

Learning Outcomes

By the end of this section, you should be able to:

- 3.5.1 Describe fundamental aspects of probability distributions.
- 3.5.2 Apply discrete probability distributions including binomial and Poisson distributions.
- 3.5.3 Apply continuous probability distributions including exponential and normal distributions.
- 3.5.4 Use Python to apply various probability distributions for probability applications.

Probability distributions are used to model various scenarios to help with probability analysis and predictions, and they are used extensively to help formulate probability-based decisions. For example, if a doctor knows that the weights of newborn infants follow a normal (bell-shaped) distribution, the doctor can use this information to help identify potentially underweight newborn infants, which might indicate a medical condition warranting further investigation. Using a normal distribution, the doctor can calculate that only a small percentage of babies have weights below a certain threshold, which might prompt the doctor to further investigate the cause of the low weight. Or a medical researcher might be interested in the probability that a person will have high blood pressure or the probability that a person will have type O blood.

Overview of Probability Distributions

To begin our discussion of probability distributions, some terminology will be helpful:

- **Random variable**—a variable where a single numerical value is assigned to a specific outcome from an experiment. Typically the letter x is used to denote a random variable. For example, assign the numerical values 1, 2, 3, ... 13 to the cards selected from a standard 52-card deck of Ace, 2, 3, ... 10, Jack, Queen, King. Notice we cannot use "Jack" as the value of the random variable since by definition a random variable must be a numerical value.
- **Discrete random variable**—a random variable is considered discrete if there is a finite or countable number of values that the random variable can take on. (If there are infinitely many values, the number of values is countable if it possible to count them individually.) Typically, a discrete random variable is the result of a count of some kind. For example, if the random variable x represents the number of cars in a

parking lot, then the values that x can take on can only be whole numbers since it would not make sense to have $x = 15.37$ cars in the parking lot.

- **Continuous random variable**—a random variable is considered continuous if the value of the random variable can take on any value within an interval. Typically, a continuous random variable is the result of a measurement of some kind. For example, if the random variable x represents the weight of a bag of apples, then x can take on any value such as $x = 2.45734$ pounds of apples.

To summarize, the difference between discrete and continuous probability distributions has to do with the nature of the random variables they represent. **Discrete probability distributions** are associated with variables that take on a finite or countably infinite number of distinct values. **Continuous probability distributions** deal with random variables that can take on any value within a given range or interval. It is important to identify and distinguish between discrete and continuous random variables since different statistical methods are used to analyze each type.

EXAMPLE 3.22

Problem

A coin is flipped three times. Determine a possible random variable that can be assigned to represent the number of heads observed in this experiment.

Solution

One possible random variable assignment could be to let x count the number of heads observed in each possible outcome in the sample space. When flipping a coin three times, there are eight possible outcomes, and x will be the numerical count corresponding to the number of heads observed for each outcome. Notice that the possible values for the random variable x are 0, 1, 2 and 3, as shown in Table 3.3.

Result for Flip #1	Result for Flip #2	Result for Flip #3	Value of Random Variable x
Heads	Heads	Heads	3
Heads	Heads	Tails	2
Heads	Tails	Heads	2
Heads	Tails	Tails	1
Tails	Heads	Heads	2
Tails	Heads	Tails	1
Tails	Tails	Heads	1
Tails	Tails	Tails	0

Table 3.3 Result of Three Random Coin Flips

EXAMPLE 3.23

Problem

Identify the following random variables as either discrete or continuous random variables:

1. The amount of gas, in gallons, used to fill a gas tank
2. Number of children per household in a certain neighborhood

3. Number of text messages sent by a certain student during a particular day
4. Number of hurricanes affecting Florida in a given year
5. The amount of rain, in inches, in Detroit, Michigan, in a certain month

Solution

1. The number of gallons of gas used to fill a gas tank can take on any value, such as 12.3489, so this represents a continuous random variable.
2. The number of children per household in a certain neighborhood can only take on certain discrete values such as 0, 1, 2, 3, etc., so this represents a discrete random variable.
3. The number of text messages sent by a certain student during a particular day can only take on certain discrete values such as 26, 10, 17, etc., so this represents a discrete random variable.
4. The number of hurricanes affecting Florida in a given year can only take on certain values such as 0, 1, 2, 3, etc., so this represents a discrete random variable.
5. The number of inches of rain in Detroit, Michigan, in a certain month can take on any value, such as 2.0563, so this represents a continuous random variable.

Discrete Probability Distributions: Binomial and Poisson

Discrete random variables are of interest in many data science applications, and there are several probability distributions that apply to discrete random variables. In this chapter, we present the binomial distribution and the Poisson distribution, which are two commonly used probability distributions used to model discrete random variables for different types of events.

Binomial Distribution

The **binomial distribution** is used in applications where there are two possible outcomes for each trial in an experiment and the two possible outcomes can be considered as success or failure. For example, when a baseball player is at-bat, the player either gets a hit or does not get a hit. There are many applications of binomial experiments that occur in medicine, psychology, engineering, science, marketing, and other fields.

There are many statistical experiments where the results of each trial can be considered as either a success or a failure. For example, when flipping a coin, the two outcomes are heads or tails. When rolling a die, the two outcomes can be considered to be an even number appears on the face of the die or an odd number appears on the face of the die. When conducting a marketing study, a customer can be asked if they like or dislike a certain product. Note that the word “success” here does not necessarily imply a good outcome. For example, if a survey was conducted of adults and each adult was asked if they smoke, we can consider the answer “yes” to be a success and the answer “no” to be a failure. This means that the researcher can define success and failure in any way; however, the binomial distribution is applicable when there are only two outcomes in each trial of an experiment.

The requirements to identify a binomial experiment and apply the binomial distribution include:

- The experiment of interest is repeated for a fixed number of trials, and each trial is independent of other trials. For example, a market researcher might select a sample of 20 people to be surveyed where each respondent will reply with a “yes” or “no” answer. This experiment consists of 20 trials, and each person’s response to the survey question can be considered as independent of another person’s response.
- There are only two possible outcomes for each trial, which can be labeled as “success” or “failure.”
- The probability of success remains the same for each trial of the experiment. For example, from past data we know that 35% of people prefer vanilla as their favorite ice cream flavor. If a group of 15 individuals are surveyed to ask if vanilla is their favorite ice cream flavor, the probability of success for each trial will be 0.35.
- The random variable x will count the number of successes in the experiment. Notice that since x will count

the number of successes, this implies that x will be a discrete random variable. For example, if the researcher is counting the number of people in the group of 15 that respond to say vanilla is their favorite ice cream flavor, then x can take on values such as 3 or 7 or 12, but x could not equal 5.28 since x is counting the number of people.

When working with a binomial experiment, it is useful to identify two specific parameters in a binomial experiment:

1. The number of trials in the experiment. Label this as n.
2. The probability of success for each trial (which is a constant value). Label this as p.

We then count the number of successes of interest as the value of the discrete random variable. Label this as x.

EXAMPLE 3.24

Problem

A medical researcher is conducting a study related to a certain type of shoulder surgery. A sample of 20 patients who have recently undergone the surgery is selected, and the researcher wants to determine the probability that 18 of the 20 patients had a successful result from the surgery. From past data, the researcher knows that the probability of success for this type of surgery is 92%.

1. Does this experiment meet the requirements for a binomial experiment?
2. If so, identify the values of n, p, and x in the experiment.

Solution

1. This experiment does meet the requirements for a binomial experiment since the experiment will be repeated for 20 trials, and each response from a patient will be independent of other responses. Each reply from a patient will be one of two responses—the surgery was successful or the surgery was not successful. The probability of success remains the same for each trial at 92%. The random variable x can be used to count the number of patients who respond that the surgery was successful.
2. The number of trials is 20 since 20 patients are being surveyed, so $n = 20$.
   The probability of success for each surgery is 92%, so $p = 0.92$.
   The number of successes of interest is 18 since the researcher wants to determine the probability that 18 of the 20 patients had a successful result from the surgery, so $x = 18$.

When calculating the probability for x successes in a binomial experiment, a binomial probability formula can be used, but in many cases technology is used instead to streamline the calculations.

The **probability mass function (PMF)** for the binomial distribution describes the probability of getting exactly x successes in n independent Bernoulli trials, each with a probability p of success. The PMF is given by the formula:

$$P(X) = x = \binom{n}{k} p^x (1-p)^{n-x}$$

Where:
$P(X = x)$ is the probability that the random variable X takes on the value of exactly x successes
n is the number of trials in the experiment
p is the probability of success
x is the number of successes in the experiment

$\binom{n}{k}$ refers to the number of ways to choose x successes from $\binom{n}{k} = \frac{n!}{(n-x)!x!}$

Note: The notation $n!$ is read as n factorial and is a mathematical notation used to express the multiplication of $n(n-1)(n-2)\ldots(3)(2)(1)$. For example, $5! = (5)(4)(3)(2)(1) = 120$.

EXAMPLE 3.25

Problem

For the binomial experiment discussed in Example 3.24, calculate the probability that 18 out of the 20 patients will respond to indicate that the surgery was successful. Also, show a graph of the binomial distribution to show the probability distribution for all values of the random variable x.

Solution

In Example 3.24, the parameters of the binomial experiment are:
$n = 20$
$p = 0.92$
$x = 18$

Substituting these values into the binomial probability formula, the probability for 18 successes can be calculated as follows:

$$\begin{aligned} P(x) &= \frac{n!}{(n-x)!x!}p^x(1-p)^{n-x} \\ P(18) &= \frac{20!}{(20-18)!18!}0.92^{18}(1-0.92)^{20-18} \\ P(18) &= \frac{20!}{2!18!}0.92^{18}(0.18)^2 \\ P(18) &= 190(0.223)(0.032) \\ P(18) &= 0.271 \end{aligned}$$

Based on this result, the probability that 18 out of the 20 patients will respond to indicate that the surgery was successful is 0.271, or approximately 27%.

Figure 3.5 illustrates this binomial distribution, where the horizontal axis shows the values of the random variable x, and the vertical axis shows the binomial probability for each value of x. Note that values of x less than 14 are not shown on the graph since these corresponding probabilities are very close to zero.

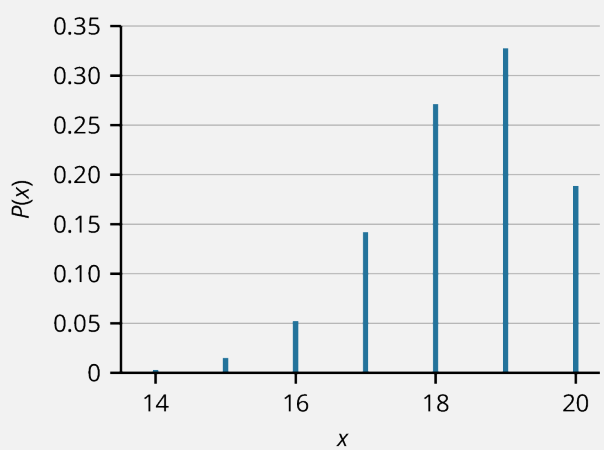


Figure 3.5 Graph of the Binomial Distribution for $n = 20$ and $p = 0.92$

Since these computations tend to be complicated and time-consuming, most data scientists will use technology (such as Python, R, Excel, or others) to calculate binomial probabilities.

Poisson Distribution

The goal of a binomial experiment is to calculate the probability of a certain number of successes in a specific number of trials. However, there are certain scenarios where a data scientist might be interested to know the probability of a certain number of occurrences for a random variable in a specific interval, such as an interval of time.

For example, a website developer might be interested in knowing the probability that a certain number of users visit a website per minute. Or a traffic engineer might be interested in calculating the probability of a certain number of accidents per month at a busy intersection.

The **Poisson distribution** is applied when counting the number of occurrences in a certain interval. The random variable then counts the number of occurrences in the interval.

A common application for the Poisson distribution is to model arrivals of customers for a queue, such as when there might be 6 customers per minute arriving at a checkout lane in the grocery store and the store manager wants to ensure that customers are serviced within a certain amount of time.

The Poisson distribution is a discrete probability distribution used in these types of situations where the interest is in a specific certain number of occurrences for a random variable in a certain interval such as time or area.

The Poisson distribution is used where the following conditions are met:

- The experiment is based on counting the number of occurrences in a specific interval where the interval could represent time, area, volume, etc.
- The number of occurrences in one specific interval is independent of the number of occurrences in a different interval.

Notice that when we count the number of occurrences that a random variable x occurs in a specific interval, this will represent a discrete random variable. For example, the count of the number of customers that arrive per hour to a queue for a bank teller might be 21 or 15, but the count could not be 13.32 since we are counting the number of customers and hence the random variable will be discrete.

To calculate the probability of x successes, the Poisson probability formula can be used, as follows:

$$P(x) = \frac{\mu^x e^{-\mu}}{x!}, \text{ where } x = 0, 1, 2, \ldots$$

Where:
μ is the average or mean number of occurrences per interval
e is the constant 2.71828...

EXAMPLE 3.26

Problem

From past data, a traffic engineer determines the mean number of vehicles entering a parking garage is 7 per 10-minute period. Calculate the probability that the number of vehicles entering the garage is 9 in a certain 10-minute period. Also, show a graph of the Poisson distribution to show the probability distribution for various values of the random variable x.

Solution

This example represents a Poisson distribution in that the random variable x is based on the number of vehicles entering a parking garage per time interval (in this example, the time interval of interest is 10 minutes). Since the average is 7 vehicles per 10-minute interval, we label the mean μ as 7. Since the

engineer want to know the probability that 9 vehicles enter the garage in the same time period, the value of the random variable x is 9.

Thus, in this example, the parameters of the Poisson distribution are:

$$\mu = 7$$
$$x = 9$$

Substituting these values into the Poisson probability formula, the probability for 9 vehicles entering the garage in a 10-minute interval can be calculated as follows:

$$\begin{aligned} P(x) &= \frac{\mu^x e^{-\mu}}{x!} \\ P(9) &= \frac{7^9 e^{-7}}{9!} = 0.101 \end{aligned}$$

Thus, there is about a 10% probability of 9 vehicles entering the garage in a 10-minute interval.

Figure 3.6 illustrates this Poisson distribution, where the horizontal axis shows the values of the random variable x and the vertical axis shows the Poisson probability for each value of x.

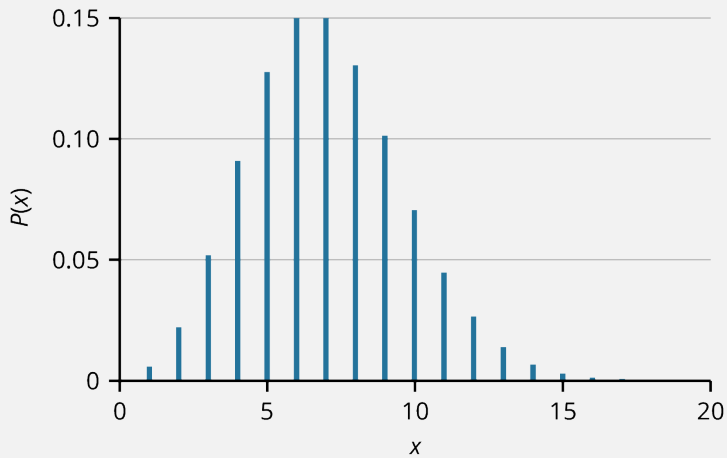


Figure 3.6 Poisson Distribution for $\mu = 7$

As with calculations involving the binomial distribution, data scientists will typically use technology to solve problems involving the Poisson distribution.

Normal Continuous Probability Distributions

Recall that a random variable is considered continuous if the value of the random variable can take on any of infinitely many values. We used the example about that if the random variable x represents the weight of a bag of apples, then x can take on any value such as $x = 2.45734$ pounds of apples.

Many probability distributions apply to continuous random variables. These distributions rely on determining the probability that the random variable falls within a distinct range of values, which can be calculated using a probability density function (PDF). The **probability density function (PDF)** calculates the corresponding area under the probability density curve to determine the probability that the random variable will fall within this specific range of values. For example, to determine the probability that a salary falls between $50,000 and $70,000, we can calculate the area under the probability density function between these two salaries.

Note that the total area under the probability density function will always equal 1. The probability that a continuous random variable takes on a specific value x is 0, so we will always calculate the probability for a random variable falling within some interval of values.

In this section, we will examine an important continuous probability distribution that relies on the probability density function, namely the *normal distribution*. Many variables, such as heights, weights, salaries, and blood pressure measurements, follow a normal distribution, making it especially important in statistical analysis. In addition, the normal distribution forms the basis for more advanced statistical analysis such as confidence intervals and hypothesis testing, which are discussed in Inferential Statistics and Regression Analysis.

The **normal distribution** is a continuous probability distribution that is symmetrical and bell-shaped. It is used when the frequency of data values decreases with data values above and below the mean. The normal distribution has applications in many fields including engineering, science, finance, medicine, marketing, and psychology.

The normal distribution has two parameters: the mean, μ, and the standard deviation, σ. The mean represents the center of the distribution, and the standard deviation measures the spread, or dispersion, of the distribution. The variable x represents the realization, or observed value, of the random variable X that follows a normal distribution.

The typical notation used to indicate that a random variable follows a normal distribution is as follows: $X \sim N(\mu, \sigma)$ (see Figure 3.7). For example, the notation $X \sim N(5.2, 3.7)$ indicates that the random variable follows a normal distribution with mean of 5.2 and standard deviation of 3.7.

A normal distribution with mean of 0 and standard deviation of 1 is called the **standard normal distribution** and can be notated as $X \sim N(0, 1)$. Any normal distribution can be standardized by converting its values to z-scores. Recall that a z-score tells you how many standard deviations from the mean there are for a given measurement.

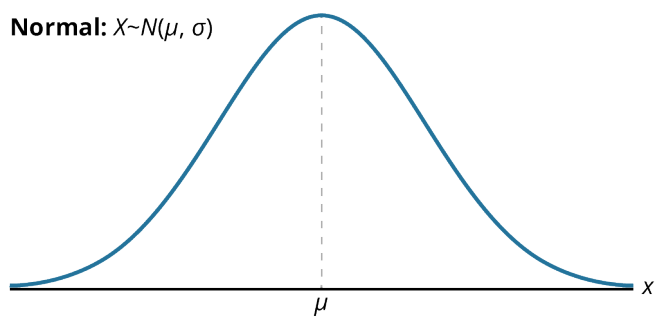


Figure 3.7 Graph of the Normal (Bell-Shaped) Distribution

The curve in Figure 3.7 is symmetric on either side of a vertical line drawn through the mean, μ. The mean is the same as the median, which is the same as the mode, because the graph is symmetric about μ. As the notation indicates, the normal distribution depends only on the mean and the standard deviation. Because the area under the curve must equal 1, a change in the standard deviation, σ, causes a change in the shape of the normal curve; the curve becomes fatter and wider or skinnier and taller depending on σ. A change in μ causes the graph to shift to the left or right. This means there are an infinite number of normal probability distributions.

To determine probabilities associated with the normal distribution, we find specific areas under the normal curve. There are several methods for finding this area under the normal curve, and we typically use some form of technology. Python, Excel, and R all provide built-in functions for calculating areas under the normal curve.

EXAMPLE 3.27

Problem

Suppose that at a software company, the mean employee salary is $60,000 with a standard deviation of $7,500. Assume salaries at this company follow a normal distribution. Use Python to calculate the

probability that a random employee earns more than $68,000.

Solution

A normal curve can be drawn to represent this scenario, in which the mean of $60,000 would be plotted on the horizontal axis, corresponding to the peak of the curve. Then, to find the probability that an employee earns more than $68,000, calculate the area under the normal curve to the right of the data value $68,000.

Figure 3.8 illustrates the area under the normal curve to the right of a salary of $68,000 as the shaded-in region.

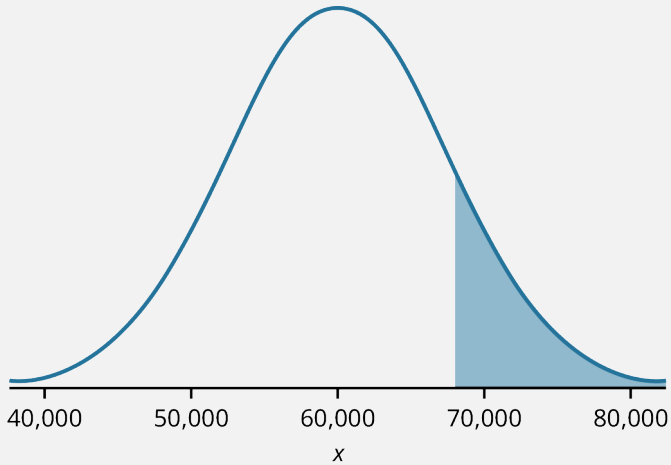


Figure 3.8 Bell-Shaped Distribution for Example 3.27. The shaded region under the normal curve corresponds to the probability that an employee earns more than $68,000.

To find the actual area under the curve, a Python command can be used to find the area under the normal probability density curve to the right of the data value of $68,000. See Using Python with Probability Distributions for the specific Python program and results. The resulting probability is calculated as 0.143.

Thus, there is a probability of about 14% that a random employee has a salary greater than $75,000.

The **empirical rule** is a method for determining approximate areas under the normal curve for measurements that fall within one, two, and three standard deviations from the mean for the normal (bell-shaped) distribution. (See Figure 3.9).

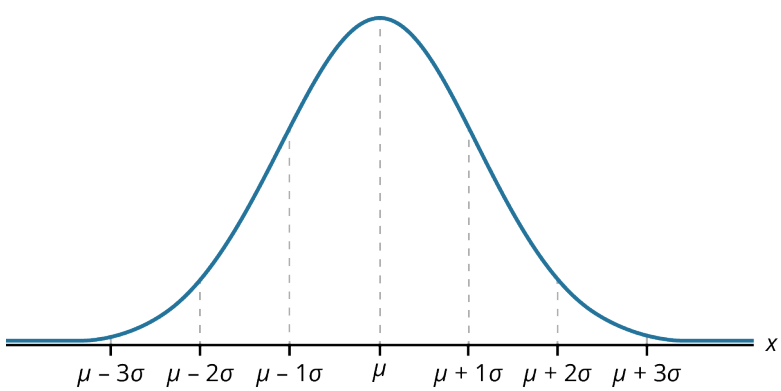


Figure 3.9 Normal Distribution Showing Mean and Increments of Standard Deviation

If x is a continuous random variable and has a normal distribution with mean μ and standard deviation σ, then the empirical rule states that:

- About 68% of the x-values lie between -1σ and $+1\sigma$ units from the mean μ (within one standard deviation

of the mean).

- About 95% of the x-values lie between -2σ and $+2\sigma$ units from the mean μ (within two standard deviations of the mean).
- About 99.7% of the x-values lie between -3σ and $+3\sigma$ units from the mean μ (within three standard deviations of the mean). *Notice that almost all the x-values lie within three standard deviations of the mean.*
- The z-scores for $+1\sigma$ and -1σ are $+1$ and -1, respectively.
- The z-scores for $+2\sigma$ and -2σ are $+2$ and -2, respectively.
- The z-scores for $+3\sigma$ and -3σ are $+3$ and -3, respectively.

EXAMPLE 3.28

Problem

An automotive designer is interested in designing automotive seats to accommodate the heights for about 95% of customers. Assume the heights of adults follow a normal distribution with mean of 68 inches and standard deviation of 3 inches. For what range of heights should the designer model the car seats to accommodate 95% of drivers?

Solution

According to the empirical rule, the area under the normal curve within two standard deviations of the mean is 95%. Thus, the designer should design the seats to accommodate heights that are two standard deviations away from the mean. The lower bound of heights would be $68 - 2(3)$ inches, and the upper bound of heights would be $68 + 2(3)$ inches. Thus, the car seats should be designed to accommodate driver heights between 62 and 74 inches.

EXPLORING FURTHER

Statistical Applets to Explore Statistical Concepts

Applets are very useful tools to help visualize statistical concepts in action. Many applets can simulate statistical concepts such as probabilities for the normal distribution, use of the empirical rule, creating box plots, etc.

Visit the Utah State University applet website (https://openstax.org/r/usu) and experiment with various statistical tools.

Using Python with Probability Distributions

Python provides a number of built-in functions for calculating probabilities associated with both discrete and continuous probability distributions such as binomial distribution and the normal distribution. These functions are part of a library called scipy.stats (https://openstax.org/r/scipy).

Here are a few of these probability density functions available within Python:

- `binom()`—calculate probabilities associated with the binomial distribution
- `poisson()`—calculate probabilities associated with the Poisson distribution
- `expon()`—calculate probabilities associated with the exponential distribution
- `norm()`—calculate probabilities associated with the normal distribution

To import these probability density functions within Python, use the import command. For example, to import

the `binom()` function use the following command:

```
from scipy.stats import binom
```

Using Python with the Binomial Distribution

The `binom()` function in Python allows calculations of binomial probabilities. The probability mass function for the binomial distribution within Python is referred to as `binom.pmf()`.

The syntax for using this function is `binom.pmf(x, n, p)`

Where:
n is the number of trials in the experiment
p is the probability of success
x is the number of successes in the experiment

Consider the previous Example 3.24 worked out using the Python `binom.pmf()` function. A medical researcher is conducting a study related to a certain type of shoulder surgery. A sample of 20 patients who have recently undergone the surgery is selected, and the researcher wants to determine the probability that 18 of the 20 patients had a successful result from the surgery. From past data, the researcher knows that the probability of success for this type of surgery is 92%. Round your answer to 3 decimal places.

In this example:
n is the number of trials in the experiment $= 20$
p is the probability of success $= 0.92$
x is the number of successes in the experiment $= 18$

The corresponding function in Python is written as:

```
binom.pmf (18, 20, 0.92)
```

The `round()` function is then used to round the probability result to 3 decimal places.

Here is the input and output of this Python program:

PYTHON CODE

```python
# import the binom function from the scipy.stats library
from scipy.stats import binom

# define parameters x, n, and p:
x = 18
n = 20
p = 0.92

# use binom.pmf() function to calculate binomial probability
# use round() function to round answer to 3 decimal places
round (binom.pmf(x, n, p), 3)
```

The resulting output will look like this:

```
0.271
```

Using Python with the Normal Distribution

The `norm()` function in Python allows calculations of normal probabilities. The probability density function is sometimes called the cumulative density function, and so this is referred to as `norm.cdf()` within Python. The `norm.cdf()` function returns the area under the normal probability density function to the left of a specified measurement.

The syntax for using this function is
`norm.cdf(x, mean, standard_deviation)`

Where:
`x` is the measurement of interest
`mean` is the mean of the normal distribution
`standard_deviation` is the standard deviation of the normal distribution

Let's work out the previous Example 3.27 using the Python `norm.cdf()` function.

Suppose that at a software company, the mean employee salary is $60,000 with a standard deviation of $7,500. Use Python to calculate the probability that a random employee earns more than $68,000.

In this example:
x is the measurement of interest $= 68{,}000$
mean is the mean of the normal distribution $= 60{,}000$
standard deviation is the standard deviation of the normal distribution $= 7{,}500$

The corresponding function in Python is written as:
`norm.cdf(68000, 60000, 7500)`

The `round()` function is then used to round the probability result to 3 decimal places.

Notice that since this example asks to find the area to the right of a salary of $68,000, we can first find the area to the left using the `norm.cdf()` function and subtract this area from 1 to then calculate the desired area to the right.

Here is the input and output of the Python program:

PYTHON CODE

```
# import the norm function from the scipy.stats library
from scipy.stats import norm
# define parameters x, mean and standard_deviation:
x = 68000
mean = 60000
standard_deviation = 7500
# use norm.cdf() function to calculate normal probability - note this is
# the area to the left
# subtract this result from 1 to obtain area to the right of the x-value
# use round() function to round answer to 3 decimal places
round (1 - norm.cdf(x, mean, standard_deviation), 3)
```

The resulting output will look like this:

```
0.143
```

Key Terms

Bayes' Theorem a method used to calculate a conditional probability when additional information is obtained to refine a probability estimate

binomial distribution a probability distribution for a discrete random variable that is the number of successes in n independent trials. Each of these trials has two possible outcomes, success or failure, with the probability of success in each trial the same.

central tendency the statistical measure that represents the center of a dataset

coefficient of variation (CV) measures the variation of a dataset by calculating the standard deviation as a percentage of the mean

complement of an event the set of all outcomes in the sample space that are not included in the event

conditional probability the probability of an event given that another event has already occurred

continuous probability distribution probability distribution that deals with random variables that can take on any value within a given range or interval

continuous random variable a random variable where there is an infinite number of values that the variable can take on

dependent events events where the probability of occurrence of one event is affected by the occurrence of another event

descriptive statistics the organization, summarization, and graphical display of data

discrete probability distribution probability distribution associated with variables that take on a finite or countably infinite number of distinct values

discrete random variable a random variable where there is only a finite or countable infinite number of values that the variable can take on

empirical probability a probability that is calculated based on data that has been collected from an experiment

empirical rule a rule that provides the percentages of data values falling within one, two, and three standard deviations from the mean for a bell-shaped (normal) distribution

event a subset of the sample space

frequency a count of the number of times that an event or observation occurs in an experiment or study

frequency distribution a method of organizing and summarizing a dataset that provides the frequency with which each value in the dataset occurs

independent events events where the probability of occurrence of one event is not affected by the occurrence of another event

interquartile range (IQR) a number that indicates the spread of the middle half, or middle 50%, of the data; the difference between the third quartile (Q_3) and the first quartile (Q_1)

mean (also called arithmetic mean) a measure of center of a dataset, calculated by adding up the data values and dividing the sum by the number of data values; average

median the middle value in an ordered dataset

mode the most frequently occurring data value in a dataset

mutually exclusive events events that cannot occur at the same time

normal distribution a bell-shaped distribution curve that is used to model many measurements, including IQ scores, salaries, heights, weights, blood pressures, etc.

outcome the result of a single trial in a probability experiment

outliers data values that are significantly different from the other data values in a dataset

percentiles numbers that divide an ordered dataset into hundredths; used to describe the relative standing of a particular value within a dataset by indicating the percentage of data points that fall below it

Poisson distribution a probability distribution for discrete random variables used to calculate probabilities for a certain number of occurrences in a specific interval

population data data representing all the outcomes or measurements that are of interest

population mean the average for all measurements of interest corresponding to the entire group under

study

population size the number of measurements for the entire group under study

probability a numerical measure that assesses the likelihood of occurrence of an event

probability analysis provides the tools to model, understand, and quantify uncertainties, allowing data scientists to make informed decisions from data

probability density function (PDF) a function that is used to describe the probability distribution of a continuous random variable

probability distribution a mathematical function that assigns probabilities to various outcomes

probability mass function (PMF) a function that is used to define the probability distribution of a discrete random variable

quartiles numbers that divide an ordered dataset into quarters; the second quartile is the same as the median

random variable a variable where a single numerical value is assigned to a specific outcome from an experiment

range a measure of dispersion for a dataset calculated by subtracting the minimum from the maximum of the dataset

relative frequency probability a method of determining the likelihood of an event occurring based on the observed frequency of its occurrence in a given sample or population

sample data data representing outcomes or measurements collected from a subset or part of a population

sample mean the average for a subset of the measurements of interest

sample size the number of measurements for the subset taken from the overall population

sample space the set of all possible outcomes in a probability experiment

standard deviation a measure of the spread of a dataset, given in the same units as the data, that indicates how far a typical data value is from the mean

standard normal distribution a normal distribution with mean of 0 and standard deviation of 1

statistical analysis the science of collecting, organizing, and interpreting data to make decisions

theoretical probability a probability that is calculated based on an assessment of equally likely outcomes

trimmed mean a calculation for the average or mean of a dataset where some percentage of data values are removed from the lower and upper end of the dataset; typically used to mitigate the effects of outliers on the mean

variance the measure of the spread of data values in a dataset based on the squared deviations from the mean, which is the average of the squared deviations of the observations from the mean

z-score a measure of the position of a data value in the dataset, calculated by subtracting the mean from the data value and then dividing the difference by the standard deviation

Group Project

Analysis of Salary Data by Gender

There are many sources of salary data where median salaries for a certain year are published by gender. One source for data on earnings by demographics is the Bureau of Labor Statistics (https://openstax.org/r/bls).

As a group:

- Research salary data by gender for a certain year.
- Compile descriptive statistics where available for statistical measurements including the mean, median, quartiles, and standard deviation by gender.
- Create graphical presentations for this data using histograms, side-by-side box plots, and time series graphs.
- Discuss the following:
  a. From your analysis, does there appear to be a wage gap for men's vs. women's salaries?

b. If so, is the wage gap improving over time or worsening over time?

Quantitative Problems

1.

Dataset A Description	Data
Number of cars entering a parking garage in a 1-hour time period	Sample of 10 counts: 15, 18, 4, 15, 8, 11, 13, 7, 16, 24

a. For Dataset A, calculate the mean and a 10% trimmed mean (round answers to 1 decimal place). Use technology as appropriate.
b. Is there any benefit in using the trimmed mean versus the mean for this dataset?

2.

Dataset B Description	Data
Weights of packages on a delivery truck (weights in pounds)	Sample of 12 packages: 3.2, 9.1, 4.3, 1.9, 2.6, 4.7, 0.7, 1.3, 3.9, 6.4, 5.0, 25.4

a. For Dataset B, calculate the mean and median (round answers to 1 decimal place). Use technology as appropriate.
b. Is there any preference in using the mean versus the median for this dataset?
c. If so, would the mean or the median be preferred as a measure of central tendency for this dataset?

3.

Dataset C Description	Data
Amount of time for a pain reliever to provide relief from migraine headache (time in minutes)	Sample of 10 patients: 12.1, 13.8, 9.4, 15.9, 11.5, 14.6, 18.1, 12.7, 11.0, 14.2

a. For Dataset C, calculate the range, standard deviation, and variance (round answers to 1 decimal place). Use technology as appropriate and include appropriate units as part of your answers.
b. What does the standard deviation of this dataset communicate regarding the amount of time to provide relief from a migraine headache?
c. Would a medical researcher prefer to use the standard deviation or the variation for this dataset, and why?

4.

Dataset D Description	Data
Credit card balances for consumers (in dollars)	Sample of 9 consumers: $878.32, $671.54, $990.35, $801.85, $4303.76, $1285.03, $289.44, $78.61, $952.83

a. For Dataset D, calculate the quartiles of the dataset.
b. For this same dataset, calculate the interquartile range (IQR).
c. Determine if there are any potential outliers in this dataset.

5.

Dataset E Description	Data
Salaries for employees at a technology company (in dollars)	Sample of 10 employees: $72,500; $84,300; $58,900; $55,100; $82,000; $94,600; $82,200; $104,600; $63,800; $77,500

For Dataset E, determine the percentile for the employee with a salary of $82,000. Round your answer to

the nearest whole number percentile.

6. At a website development company, the average number of sick days taken by employees per year is 9 days with a standard deviation of 2.3 days.
   a. A certain employee has taken 2 sick days for the year. Calculate the z-score for this employee (round your answer to 2 decimal places).
   b. What does the z-score communicate about this employee's number of sick days as compared to other employees in the company?
   c. A different employee has taken 17 sick days for the year. Calculate the z-score for this employee (round your answer to 2 decimal places).
   d. What does the z-score communicate about this employee's number of sick days as compared to other employees in the company?

7. Two cards are selected at random from a standard 52-card deck. Find the probability that the second card is an Ace, given that the first card selected was an Ace (assume the first card is not replaced before selecting the second card). Round your answer to 4 decimal places.

8. From past data, it's known that the probability that an individual owns a dog is 0.42. Three random people are asked if they own a dog. (Round answers to 3 decimal places).
   a. Find the probability that all three people own a dog.
   b. Find the probability that none of the three people own a dog.
   c. Find the probability that at least one of the people owns a dog.

9. In a group of 100 people, 35 people wear glasses, 40 people wear contacts, 12 wear both glasses and contacts, and 25 wear neither glasses nor contacts.
   a. Are the events "person wears glasses" and "person wears contacts" mutually exclusive?
   b. Calculate the probability that a randomly chosen person wears glasses or contacts. Round your answer to 3 decimal places.

10. A medical researcher collects statistical data as follows:
   - 23% of senior citizens (65 or older) get the flu each year.
   - 32% of people under 65 get the flu each year.
   - In the population, there are 15% senior citizens.

   a. Are the events "being a senior" and "getting the flu" dependent or independent?
   b. Find the probability that a person selected at random is a senior and will get the flu.
   c. Find the probability that a person selected at random is under 65 years old and will get the flu.

11. A certain type of COVID-related virus affects 5% of the population. A screening test is available to detect the virus. Assume the probability of a true positive from the screening test is 85%, meaning that the probability that a person has a positive test result, given that they actually have the virus, is 0.85. Also assume the probability of a false positive from the screening test is 12%, which indicates the probability that a person has a positive test result, given that they do not have the virus, is 0.12. Calculate the probability that a person actually has the virus given that the screening test shows a positive result. Round your answer to 3 decimal places.

12. You are taking a multiple-choice test with 10 questions where each question has 5 possible answers: (a), (b), (c), (d), and (e). You did not study, so you will guess at each question. What is the probability of getting exactly 5 questions correct?

13. The average arrival rate of trucks at a truck loading station is 15 per day. Calculate the probability that 17 trucks arrive at the loading station on a particular day.

14. In a certain city, cell phone bills are known to follow a normal distribution with a mean of $92 and standard deviation of $14.

a. Find the probability that a randomly selected cell phone customer has a monthly bill less than $75.
b. Find the probability that a randomly selected cell phone customer has a monthly bill more than $85.

4 Inferential Statistics and Regression Analysis

Figure 4.1 Inferential statistics is used extensively in data science to draw conclusions about a larger population—and drive decision-making. (credit: modification of work "Compiling Health Statistics" by Ernie Branson/Wikimedia Commons, Public Domain)

Chapter Outline

4.1 Statistical Inference and Confidence Intervals
4.2 Hypothesis Testing
4.3 Correlation and Linear Regression Analysis
4.4 Analysis of Variance (ANOVA)

Introduction

Inferential statistics plays a key role in data science applications, as its techniques allow researchers to infer or generalize observations from samples to the larger population from which they were selected. If the researcher had access to a full set of population data, then these methods would not be needed. But in most real-world scenarios, population data cannot be obtained or is impractical to obtain, making inferential analysis essential. This chapter will explore the techniques of inferential statistics and their applications in data science.

Confidence intervals and hypothesis testing allow a data scientist to formulate conclusions regarding population parameters based on sample data.

One technique, **correlation analysis**, allows the determination of a statistical relationship between two numeric quantities, often referred to as variables. A **variable** is a characteristic or attribute that can be measured or observed. A **correlation** between two variables is said to exist where there is an association between them. Finance professionals often use correlation analysis to predict future trends and mitigate risk in a stock portfolio. For example, if two investments are strongly correlated, an investor might not want to have both investments in a certain portfolio since the two investments would tend to move in the same direction as market prices rose or fell. To diversify a portfolio, an investor might seek investments that are not strongly correlated with one another.

Regression analysis takes correlation analysis one step further by *modeling* the relationship between the two numeric quantities or variables when a correlation exists. In statistics, modeling refers specifically to the

process of creating a mathematical representation that describes the relationship between different variables in a dataset. The model is then used to understand, explain, and predict the behavior of the data.

This chapter focuses on **linear regression**, which is analysis of the relationship between one *dependent variable* and one *independent variable*, where the relationship can be modeled using a linear equation. The foundations of regression analysis have many applications in data science, including in machine learning models where a mathematical model is created to determine a relationship between input and output variables of a dataset. Several such applications of regression analysis in machine learning are further explored in Decision-Making Using Machine Learning Basics. In Time Series and Forecasting, we will use time series models to analyze and predict data points for data collected at different points in time.

4.1 Statistical Inference and Confidence Intervals

Learning Outcomes

By the end of this section, you should be able to:

- 4.1.1 Estimate parameters, create confidence intervals, and calculate sample size requirements.
- 4.1.2 Apply bootstrapping methods for parameter estimation.
- 4.1.3 Use Python to calculate confidence intervals and conduct hypothesis tests.

Data scientists interested in inferring the value of a population truth or parameter such as a population mean or a population proportion turn to inferential statistics. A data scientist is often interested in making generalizations about a population based on the characteristics derived from a sample; inferential statistics allows a data scientist to draw conclusions about a population based on sample data. In addition, inferential statistics is used by data scientists to assess model performance and compare different algorithms in machine learning application. Inferential statistics provides methods for generating predictive forecasting models, and this allows data scientists to generate predictions and trends to assist in effective and accurate decision-making. In this section, we explore the use of confidence intervals, which is used extensively in inferential statistical analysis.

We begin by introducing confidence intervals, which are used to estimate the range within which a population parameter is likely to fall. We discuss estimation of parameters for the mean both when the standard deviation is known and when it is not known. We discuss sample size determination, building on the sampling techniques presented in Collecting and Preparing Data. And we discuss bootstrapping, a method used to construct a confidence interval based on repeated sampling. Hypothesis Testing will move on to hypothesis testing, which is used to make inferences about unknown parameters.

Estimating Parameters with Confidence Intervals

A **point estimate** is a single value that is used to estimate a population parameter. For example, a **sample mean** is a *point estimate* that is representative of the true population mean in that the sample mean is used as an estimate for the unknown population mean. When researchers collect data from a sample to make inferences about a population, they calculate a point estimate based on the observed sample data. (See Survey Design and Implementation for coverage of sampling techniques.) The point estimate serves as the best guess or approximation for the parameter's actual value.

A **confidence interval** estimates the *range* within which a population parameter, such as a mean or a proportion, is likely to fall. The confidence interval provides a level of uncertainty associated with the estimate and is expressed as a range of values. A confidence interval will provide both a lower and an upper bound for the population parameter, where the point estimate is centered within the interval.

Table 4.1 describes the point estimates and corresponding population parameters for the mean and proportion.

	Population Parameter	Point Estimate
Mean	Population mean is denoted as μ.	Point estimate is the sample mean $\bar{x}$.
Proportion	Population proportion is denoted as p.	Point estimate is the sample proportion $\hat{p}$.

Table 4.1 Population Parameters and Point Estimates for Mean and Proportion

Let's say a researcher is interested in estimating the mean income for all residents of California. Since it is not feasible to collect data from every resident of California, the researcher selects a random sample of 1,000 residents and calculates the sample mean income for these 1,000 people. This average income estimate from the sample then provides an estimate for the population mean income of all California residents. The sample mean is chosen as the point estimate for the population mean in that the sample mean provides the most *unbiased estimate* for the population mean. In the same way, the **sample proportion** provides the most unbiased estimate for the population proportion. An **unbiased estimator** is a statistic that provides an accurate estimate for the corresponding population parameter without overestimating or underestimating the parameter.

In order to calculate a confidence interval, two quantities are needed:

- The point estimate
- The margin of error

As noted, the point estimate is a single number that is used to estimate the population parameter. The **margin of error** (usually denoted by *E*) provides an indication of the maximum error of the estimate. The margin of error can be viewed as the maximum distance around the point estimate where the population parameter exists based on a specified confidence.

Once the point estimate and margin of error are determined, the confidence interval is calculated as follows:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\end{aligned}$$

The margin of error reflects the desired *confidence level* of the researcher. The **confidence level** is the probability that the interval estimate will contain the population parameter, given that the estimation process on the parameter is repeated over and over. Confidence levels typically range from 80% to 95% confidence.

In addition to the confidence level, both the variability of the sample and the sample size will affect the margin of error.

Once the confidence interval has been calculated, a concluding statement is made that reflects both the confidence level and the unknown population parameter. The concluding statement indicates that there is a certain level of confidence that the population parameter is contained within the bounds of the confidence interval.

As an example, consider a data scientist who is interested in forecasting the median income for all residents of California. Income data is collected from a sample of 1,000 residents, and the median income level is $68,500. Also assume the margin of error for a 95% confidence interval is $4,500 (more details on the calculation of margin of error are provided next). Now the data scientist can construct a 95% confidence interval to forecast income levels as follows:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ &= \$68500 - \$4500\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\\ &= \$68500 + \$4500 = \$73000\end{aligned}$$

The data scientist can then state the following conclusion: There is 95% confidence that the forecast for median income for all residents of California is between $64,000 and $73,000.

EXAMPLE 4.1

Problem

A medical researcher is interested in estimating the population mean age for patients suffering from arthritis. A sample of 100 patients is taken, and the sample mean is determined to be 64 years old. Assume that the corresponding margin of error for a 95% confidence interval is calculated to be 4 years.

Calculate the confidence interval and provide a conclusion regarding the confidence interval.

Solution

The point estimate is the sample mean, which is 64, and the margin of error is given as 4 years.

The 95% confidence interval can be calculated as follows:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ &= 64 - 4 = 60\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\\ &= 64 + 4 = 68\end{aligned}$$

Concluding statement:
The researcher is 95% confident that the mean population age for patients suffering from arthritis is contained in the interval from 60 to 68 years of age.

Sampling Distribution for the Mean

A researcher takes repeated samples of size 1,000 from the residents of New York to collect data on mean income of residents of New York.

For each sample of size 1,000, we can calculate a sample mean, $\bar{x}$. If the researcher were to take 50 such samples (each of sample size 1,000), a series of sample means can be calculated:

$$\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_{50}$$

A probability distribution based on all possible random samples of a certain size from a population—or **sampling (or sample) distribution**—can then be analyzed. For example, we can calculate the mean of these sample means, and we can calculate the standard deviation of these sample means.

There are two important properties of the sample distribution of these sample means:

1. The mean of the sample means (notated as $\mu_{\bar{x}}$) is equal to the population mean μ.
   a. Written mathematically: $\mu_{\bar{x}} = \mu$
2. The standard deviation of the sample means (notated as $\sigma_{\bar{x}}$) is equal to the population standard deviation σ divided by the square root of the sample size n.
   a. Written mathematically: $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$

The **central limit theorem** describes the relationship between the sample distribution of sample means and

the underlying population. This theorem is an important tool that allows data scientists and researchers to use sample data to generate inferences for population parameters.

Conditions of the central limit theorem:

- If random samples are taken from any population with mean μ and standard deviation, where the sample size is at least 30, then the distribution of the sample means approximates a normal (bell-shaped) distribution.
- If random samples are taken from a population that is normally distributed with mean μ and standard deviation σ, then the distribution of the sample means approximates a normal (bell-shaped) distribution for any sample size

EXAMPLE 4.2

Problem

An economist takes random samples of size 50 to estimate the mean salary of chemical engineers. Assume that the population mean salary is $85,000 and the population standard deviation is $9,000. It is unknown if the distribution of salaries follows a normal distribution.

Calculate the mean and standard deviation of the sampling distribution of the sample means and comment on the shape of the distribution for the sample means.

Solution

The mean of the sample means is equal to the population mean μ.

$$\mu_{\overline{x}} = \mu = \$85{,}000$$

The standard deviation of the sample means is equal to the population standard deviation σ divided by the square root of the sample size n.

$$\sigma_{\overline{x}} = \frac{\sigma}{\sqrt{n}} = \frac{\$9{,}000}{\sqrt{50}} = \$1272.8$$

Since the sample size of 50 is greater than 30, the distribution of the sample means approximates a normal (bell-shaped) distribution.

Confidence Interval for the Mean When the Population Standard Deviation Is Known

Although in many situations the population standard deviation is unknown, in some cases a reasonable estimate for the population standard deviation can be obtained from past studies or historical data.

Here is what is needed to calculate the confidence interval for the mean when the population standard deviation is known:

a. A random sample is selected from the population.
b. The sample size is at least 30, or the underlying population is known to follow a normal distribution.
c. The population standard deviation (σ) is known.

Once these conditions are met, the margin of error E is calculated according to the following formula:

$$E = z_c \frac{\sigma}{\sqrt{n}}$$

Where:

z_c is called the critical value of the normal distribution and is calculated as the z-score, which includes the area

corresponding to the confidence level between $-z_c$ and $+z_c$. For example, for a 95% confidence interval, the corresponding critical value is $z_c = 1.96$ since an area of 0.95 under the normal curve is contained between the z-scores of -1.96 and $+1.96$.

σ is the population standard deviation.

n is the sample size.

Note: $\frac{\sigma}{\sqrt{n}}$ is called the **standard error of the mean**.

Typical values of z_c for various confidence levels are shown in Table 4.2.

Confidence Level	Value of z_c
80% confidence	1.280
90% confidence	1.645
95% confidence	1.960
99% confidence	2.575

Table 4.2 Typical Values of z_c for Various Confidence Levels

Graphically, the critical values can be marked on the normal distribution curve, as shown in Figure 4.2. (See Discrete and Continuous Probability Distributions for a review of the normal distribution curve.) Figure 4.2 is an example for a 95% confidence interval where the area of 0.95 is centered as the area under the standard normal curve showing the values of $-z_c$ and $+z_c$. Since the standard normal curve is symmetric, and since the total area under the curve is known to be 1, the area in each of the two tails can be calculated to be 0.025. Thus, the **critical value** is that z-score which cuts off an area of 0.025 in the upper tail of the standard normal curve.

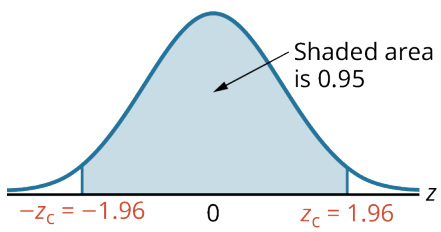


Figure 4.2 Critical Values of z_c Marked on Standard Normal Curve for 95% Confidence Interval

Once the margin of error has been calculated, the confidence interval can be calculated as follows:

$$\text{Sample Mean } \pm \text{Margin of Error}$$

Note that this is the general format for constructing any confidence interval, namely the margin of error is added and subtracted to the sample statistic to generate the upper and lower bounds of the confidence interval, respectively. A **sample statistic** describes some aspect of the sample, such as a sample mean or sample proportion.

General Form of a Confidence Interval: $\text{Sample Statistic } \pm \text{Margin of Error}$

EXAMPLE 4.3

Problem

A college professor collects data on the amount of time spent on homework assignments per week for a

sample of 50 students in a statistics course. From the sample of 50 students, the mean amount of time spent on homework assignments per week is 12.5 hours. The population standard deviation is known to be 6.3 hours.

The professor would like to forecast the amount of time spent on homework in future semesters. Create a forecasted confidence interval using both a 90% and 95% confidence interval and provide a conclusion regarding the confidence interval. Also compare the widths of the two confidence intervals. Which confidence interval is wider?

Solution

Calculation for 90% Confidence Interval

For a 90% confidence interval, the corresponding critical value is $z_c = 1.645$.

The margin of error is calculated as follows:

$$E = z_c \frac{\sigma}{\sqrt{n}} = 1.645 \cdot \frac{6.3}{\sqrt{50}} = 1.47$$

The 90% confidence interval is calculated as follows:

$$\begin{aligned} \text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error} \\ &= 12.5 - 1.47 = 11.03 \\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error} \\ &= 12.5 + 1.47 = 13.97 \end{aligned}$$

Concluding statement:

The college professor can forecast with 90% confident that the mean amount of time spent on homework assignments in a future semester is the interval from 11.03 to 13.97 hours per week.

Calculation for 95% Confidence Interval

For a 95% confidence interval, the corresponding critical value is $z_c = 1.960$.

The margin of error is calculated as follows:

$$E = z_c \frac{\sigma}{\sqrt{n}} = 1.960 \cdot \frac{6.3}{\sqrt{50}} = 1.75$$

The 95% confidence interval is calculated as follows:

$$\begin{aligned} \text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error} \\ &= 12.5 - 1.75 = 10.75 \\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error} \\ &= 12.5 + 1.75 = 14.25 \end{aligned}$$

Concluding statement:

The college professor can forecast with 95% confident that the mean amount of time spent on homework assignments in a future semester is the interval from 10.75 to 14.25 hours per week.

Comparison of 90% and 95% confidence intervals:

The 90% confidence interval extends from 11.03 to 13.97.
The 95% confidence interval extends from 10.75 to 14.25.

Notice that the 95% confidence interval is wider. If the confidence level is increased, with all other

parameters held constant, we should expect that the confidence interval will become wider. Another way to consider this: the wider the confidence interval, the more likely the interval is to contain the true population mean. This makes intuitive sense in that if you want to be more certain that the true value of the parameter is within an interval, then the interval needs to be wider to account for a larger range of potential values. As an analogy, consider a person trying to catch a fish. The wider the net used, the higher the probability of catching a fish.

You might notice that the sample size n is located in the denominator of the formula for margin of error. This indicates that as the sample size increases, the margin of error will decrease, which will then result in a narrower confidence interval. On the other hand, as the sample size decreases, the margin of error increases and the confidence interval becomes wider.

Confidence Interval for the Mean When the Population Standard Deviation Is Unknown

A confidence interval can still be determined when the population standard deviation is unknown by calculating a sample standard deviation based on the sample data. This is actually the more common application of confidence intervals.

Here are the conditions required to use this procedure:

a. A random sample is selected from the population.
b. The sample size is at least 30, or the underlying population is known to follow a normal distribution.
c. The population standard deviation (σ) is unknown; the sample standard deviation (s) can be calculated.

Once these requirements are met, the margin of error (E) is calculated according to the following formula:

$$E = t_c \frac{s}{\sqrt{n}}$$

t_c is called the *critical value of the t-distribution*. The **t-distribution** is a bell-shaped, symmetric distribution similar to the normal distribution, though the t-distribution has "thicker tails" as compared to the normal distribution. The comparison of the normal and t-distribution curves is shown in Figure 4.3.

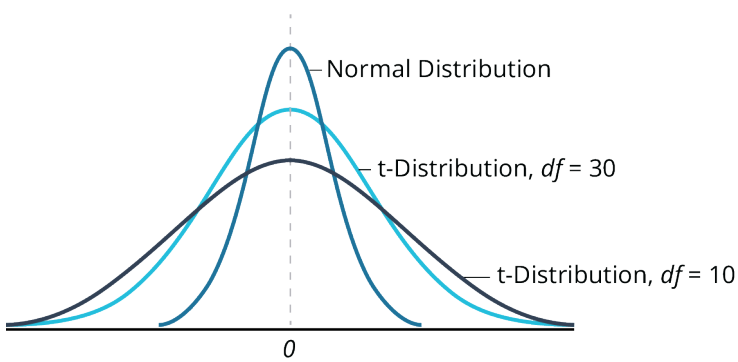


Figure 4.3 Comparison of t-Distribution and Normal Distribution Curves

The t-distribution is actually a family of curves, determined by a parameter called *degrees of freedom (df)*, where *df* is equal to $n - 1$. The critical value t_c is similar to a *z*-score and specifies the area under the t-distribution curve corresponding to the confidence level between $-t_c$ and $+t_c$. Values of t_c can be obtained using either a look-up table or technology.

For example, for a 95% confidence interval and sample size of 30, the corresponding critical value is $t_c = 2.045$ since an area of 0.95 under the t-distribution curve is contained between the t-scores of -2.045 and $+2.045$.

s is the sample standard deviation.

n is the sample size.

df is degrees of freedom, where $df = n - 1$.

Typical values of t_c for various confidence levels and degrees of freedom are shown in Table 4.3.

	Confidence Level		
Degrees of Freedom (*df*)	90% Confidence	95% Confidence	99% Confidence
1	6.314	12.706	63.657
2	2.920	4.203	9.925
3	2.353	3.182	5.841
4	2.132	2.776	4.604
5	2.015	2.571	4.032
10	1.812	2.228	3.169
15	1.753	2.131	2.947
20	1.725	2.086	2.845
25	1.708	2.060	2.787
30	1.697	2.042	2.750

Table 4.3 Typical Values of t_c for Various Confidence Levels and Degrees of Freedom

Note that Python can be used to calculate these critical values. Python provides a function called `t.ppf()` that generates the value of the t-distribution corresponding to a specified area under the t-distribution curve and specified degrees of freedom. This function is part of the scipy.stats library (https://openstax.org/r/stats).

The syntax for the function is:
`t.ppf(area to left, degrees of freedom)`

For example, for a 95% confidence interval, there is an area of 0.95 centered under the t-distribution curve, which leaves a remaining area of 0.05 for the two tails of the distribution, which implies an area of 0.025 in each tail. To find a critical value corresponding to the upper 0.025 area, note that the area to the left of this critical value will then be 0.975.

Here is an example of Python code to generate the critical value t_c for a 95% confidence interval and 15 degrees of freedom.

PYTHON CODE

```
# import function from scipy.stats library
from scipy.stats import t

# define parameters called area to left and degrees of freedom df
Area_to_left = 0.975
df = 15

# use t.ppf function to calculate critical values associated with t-
```

```
# use round function to round answer to 3 decimal places
round (t.ppf(Area_to_left, df), 3)
```

The resulting output will look like this:

```
2.131
```

Once the margin of error is calculated, the confidence interval is formed in the same way as the previous section, namely:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\end{aligned}$$

EXAMPLE 4.4

Problem

A company's human resource administrator wants to estimate the average commuting distance for all 5,000 employees at the company. Since it is impractical to collect commuting distances from all 5,000 employees, the administrator decides to sample 16 employees and collects data on commuting distance from each employee in the sample. The sample data indicates a sample mean of 15.8 miles with a standard deviation of 3.2 miles. Calculate a 99% confidence interval and provide a conclusion regarding the confidence interval.

Solution

Since the sample size is 16 employees, the degrees of freedom is one less than the sample size, which is $df = 15$. For a 99% confidence interval, the corresponding critical value is $t_c = 2.947$.

The margin of error is calculated as follows:

$$E = t_c \frac{s}{\sqrt{n}} = 2.947 \cdot \frac{3.2}{\sqrt{16}} = 2.36$$

The 99% confidence interval is calculated as follows:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ &= 15.8 - 2.36 = 13.44\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\\ &= 15.8 + 2.36 = 18.16\end{aligned}$$

Concluding statement:

The administrator can be 99% confident that the true population mean commuting distance for all employees is between 13.44 to 18.16 miles.

Confidence Interval for Proportions

We can also calculate a confidence interval for a population proportion based on the use of sample data. The basis for the confidence interval will be the application of a normal approximation to the binomial distribution. Recall from Discrete and Continuous Probability Distributions that a *binomial distribution* is a probability distribution for a discrete random variable where there are only two possible outcomes of an experiment. A **proportion** measures the number of *successes* in a sample. For example, if 10 survey respondents out of 50

indicate they are planning to travel internationally within the next 12 months, then the proportion is 10 out of 50, which is 0.2, or 20%. Note that the term *success* does not necessarily imply a positive outcome. For example, a researcher might be interested in the proportion of smokers among U.S. adults, and the number of smokers would be considered the number of successes.

Some terminology will be helpful:
p represents the population proportion, which is typically unknown.
$\hat{p}$ represents the sample proportion.
x represents the number of successes in the sample.
n represents the sample size.

Here are the requirements to use this procedure:

a. A random sample is selected from the population.
b. Verify that the normal approximation to the binomial distribution is appropriate by ensuring that both $n\hat{p}$ and $n(1-\hat{p})$ are both at least 5, where $\hat{p}$ represents the sample proportion.

The sample proportion $\hat{p}$ is calculated as the number of successes divided by the sample size:

$$\hat{p} = \frac{x}{n}$$

When these requirements are met, the margin of error (*E*) for a confidence interval for proportions is calculated according to the following formula:

$$E = z_c\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Where:

z_c is called the critical value of the normal distribution and is calculated as the *z*-score, which includes the area corresponding to the confidence level between $-z_c$ and $+z_c$.

$\hat{p}$ represents the sample proportion.

n is the sample size.

Once the margin of error is calculated, the confidence interval is formed in the same way as the previous section. In this case, the point estimate is the sample proportion $\hat{p}$:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\end{aligned}$$

EXAMPLE 4.5

Problem

A medical researcher wants to know if there has been a statistically significant change in the proportion of smokers from five years ago, when the proportion of adult smokers in the United States was approximately 28%. The researcher selects a random sample of 1,500 adults, and of those, 360 respond that they are smokers. Calculate a 95% confidence interval for the true population proportion of adults who smoke. Also, determine if there has been a statistically significant change in the proportion of smokers as compared to five years ago.

Solution

For a 95% confidence interval, the corresponding critical value is $z_c = 1.960$.

Start off by calculating the sample proportion $\hat{p}$:

$$\hat{p} = \frac{x}{n} = \frac{360}{1{,}500} = 0.24$$

Verify that the normal approximation to the binomial distribution is appropriate by ensuring that both $n\hat{p}$ and $n(1-\hat{p})$ are both at least 5, where $\hat{p}$ represents the sample proportion.

For this example, $n\hat{p} = (1{,}500)(0.24) = 360$, and $n(1-\hat{p}) = (1{,}500)(1-0.24) = 1{,}140$. Both of these results are at least 5, which verifies that the normal approximation to the binomial distribution is appropriate.

The margin of error is then calculated as follows:

$$E = z_c\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} = 1.960\sqrt{\frac{0.24(1-0.24)}{1{,}500}} = 0.022$$

The 95% confidence interval is calculated as follows:

$$\begin{aligned}\text{Lower Bound of the Confidence Interval} &= \text{Point Estimate} - \text{Margin of Error}\\ &= 0.24 - 0.022 = 0.218\\ \text{Upper Bound of the Confidence Interval} &= \text{Point Estimate} + \text{Margin of Error}\\ &= 0.24 + 0.022 = 0.262\end{aligned}$$

Concluding statement:

The researcher can be 95% confident that the true population of adult smokers in the United States is between 0.218 and 0.262, which can also be written as 21.8% to 26.2%. Since this 95% confidence interval excludes the previous value of 28% from five years ago, the researcher can state that there has been a statistically significant decrease in the proportion of smokers as compared to five years ago.

Sample Size Determination

When collecting sample data in order to construct a confidence interval for a mean or proportion, how does the researcher determine the optimal sample size? Too small of a sample size may lead to a wide confidence interval that is not very useful. Too large of a sample size can result in wasted resources if a smaller sample size would be sufficient. Sampling is covered in some depth in Handling Large Datasets. Here, we review methods to determine minimum sample size requirements when constructing confidence intervals for means or proportions. Note that the desired margin of error plays a key role in this sample size determination.

Sample Size for Confidence Interval for the Mean

When determining a confidence interval for a mean, a researcher can use the margin of error formula for the mean; solving this formula algebraically for the sample size (n), the following minimum sample size formula can be used to determine the minimum sample size needed to achieve a certain margin of error:

Sample size formula for confidence interval for mean:

$$E = z_c\frac{\sigma}{\sqrt{n}} \;\rightarrow\; n = \left(\frac{z_c\sigma}{E}\right)^2$$

Where:
z_c is the critical value of the normal distribution.
σ is the population standard deviation.
E is the desired margin of error.

Note that for sample size calculations, sample size results are rounded up to the next higher whole number.

For example, if the formula previously shown results in a sample size of 59.2, then the sample size will be rounded to 60.

EXAMPLE 4.6

Problem

A benefits analyst is interested in a 90% confidence interval for the mean salary for chemical engineers. What sample size should the analyst use if a margin of error of $1,000 is desired? Assume a population standard deviation of $8,000.

Solution

For a 90% confidence interval, the corresponding critical value is $z_c = 1.645$. The population standard deviation is given as $8,000, and the margin of error is given as $1,000.

Using the sample size formula:

$$n = \left(\frac{z_c \sigma}{E}\right)^2 = \left(\frac{1.645 \cdot 8000}{1000}\right)^2 = 173.2$$

Round to the next higher whole number, so the desired sample size is 174.

The analyst should target a sample size of 174 chemical engineers for a salary-related survey in order to them calculate a 90% confidence interval for the mean salary of chemical engineers.

Sample Size for Confidence interval for a Proportion

In Example 4.5, how would the researcher know the optimal sample size to use to collect sample data on the proportion of adults who are smokers? The margin of error formula can be used to derive the sample size for a proportion as follows:

$$n = \hat{p}\,(1 - \hat{p})\left(\frac{z_c}{E}\right)^2$$

Where:
z_c is the critical value of the normal distribution.
$\hat{p}$ is the sample proportion.
E is the desired margin of error.

Notice in this formula, it's assumed that some prior estimate for the sample proportion $\hat{p}$ is available, perhaps from historical data or previous studies. If a prior estimate for the sample proportion is not available, then use the value of 0.5 for $\hat{p}$.

DETERMINING SAMPLE SIZE

When determining sample size needed for a confidence interval for a proportion:

If a prior estimate for the sample proportion is available, then that prior estimate should be utilized.

- If a prior estimate for the sample proportion is not available, then use $\hat{p} = 0.5$.

EXAMPLE 4.7

Problem

Political candidate Smith is planning a survey to determine a 95% confidence interval for the proportion of voters who plan to vote for Smith. How many people should be surveyed? Assume a margin of error is 3%.

a. Assume there is no prior estimate for the proportion of voters who plan to vote for candidate Smith.
b. Assume that from prior election results, approximately 42% of people previously voted for candidate Smith.

Solution

a. For a 95% confidence interval, the corresponding critical value is $z_c = 1.960$. The margin of error is specified as 0.03. Since a prior estimate for the sample proportion is unknown, use a value of 0.5 for $\hat{p}$. Using the sample size formula:

$$n = \hat{p}\,(1 - \hat{p})\left(\frac{z_c}{E}\right)^2 = 0.5(1 - 0.5)\left(\frac{1.960}{0.03}\right)^2 = 1{,}067.1$$

Round to the next higher whole number, so the desired sample size is 1,068 people to be surveyed.

b. Since a prior estimate for the sample proportion is available, use the value of 0.42 for $\hat{p}$. Using the sample size formula:

$$n = \hat{p}\,(1 - \hat{p})\left(\frac{z_c}{E}\right)^2 = 0.42\,(1 - 0.42)\left(\frac{1.960}{0.03}\right)^2 = 1039.8$$

Round to the next higher whole number, so the desired sample size is 1,040 people to be surveyed. Note that having a prior estimate of the sample proportion will result in a smaller sample size requirement. This is a general conclusion, so if a researcher has a prior estimate for the population proportion, this will result in a smaller sample size requirement as compared to the situation where no prior estimate is available.

Bootstrapping Methods

In the earlier discussion of confidence intervals, certain requirements needed to be met in order to use the estimation methods. For example, when calculating the confidence interval for a mean, this requirement was indicated:

- The sample size is at least 30, or the underlying population is known to follow a normal distribution.

When calculating the confidence interval for a proportion, this requirement was noted:

- Verify that the normal approximation to the binomial distribution is appropriate by ensuring that both $n\hat{p}$ and $n(1 - \hat{p})$ are at least 5.

What should a researcher do when these requirements are not met? Fortunately, there is another option called "bootstrapping" that can be used to find confidence intervals when the underlying distribution is unknown or if one of the conditions is not met. This bootstrapping method involves repeatedly *taking samples with replacement*.

Sampling with replacement means that when a sample is selected, the sample is replaced back in the dataset before selecting the next sample. For example, a casino dealer plans to take a sample of two cards from a standard 52-card deck. If the sampling is to be done with replacement, then after selecting the sample of the first card, this card is then returned to the deck before the second card is selected. This implies that the same data value can appear multiple times in a given sample since once a data value is selected for the sample, it is

replaced back in the dataset and is eligible to be selected again for the sample.

For example, a researcher who has collected 100 samples to use for inference may want to estimate a confidence interval for the population mean. The researcher can then resample, with replacement, from this limited set of 100 observations. They would repeatedly sample 100 and calculate the sample mean. These means would then represent the distribution of means (remember the discussion of sampling distribution). Typically, for *bootstrapping*, repeated samples are taken hundreds or thousands of times and then the sample mean is calculated for each sample.

The term "bootstrapping" comes from the old saying "pull yourself up by your bootstraps" to imply a task that was accomplished without any outside help. In the statistical sense, **bootstrapping** refers to the ability to estimate parameters based solely on one sample of data from the population without any other assumptions.

The bootstrapping method is considered a **nonparametric method** since the technique makes no assumptions about the probability distribution from which the data are sampled. (**Parametric methods**, by contrast, assume a specific form for the underlying distribution and require estimating parameters.)

Since bootstrapping requires a large number of repeated samples, software (such as Excel, Python, or R) is often used to automate the repetitive sampling procedures and construct the confidence intervals.

The bootstrapping procedure for confidence interval estimation for a mean or proportion is as follows:

1. Start out with a random sample of size n. Collect many random bootstrap samples of size n—for example, hundreds or thousands of such samples. Keep in mind that the sampling is done with replacement.
2. For each sample, calculate the sample statistic, which is the sample mean $\bar{x}$ or the sample proportion $\hat{p}$.
3. Rank order the sample statistics from smallest to largest.
4. For a 95% confidence interval, find the percentiles $P_{2.5}$ and $P_{97.5}$ in the ranked data. These values establish the 95% confidence interval.
5. For a 90% confidence interval, find the percentiles P_5 and P_{95}. These values establish the 90% confidence interval.

EXAMPLE 4.8

Problem

A college administrator is developing new marketing materials to increase enrollment at the college, and the administrator is interested in a 90% confidence interval for the mean age of students attending the college.

The administrator believes the underlying distribution of ages is skewed (does not follow a normal distribution), so a bootstrapping method will be used to construct the confidence interval. The administrator selects a random sample of 20 students and records their ages as shown in Table 4.4. Use the bootstrapping method to construct the 90% confidence interval by taking repeated samples of size 10.

Student ID	Student Age
001	22
002	24
003	21
004	34

Table 4.4 Ages of 20 Randomly

Student ID	Student Age
005	29
006	23
007	21
008	20
009	19
010	21
011	25
012	28
013	22
014	37
015	24
016	31
017	23
018	19
019	26
020	20

Table 4.4 Ages of 20 Randomly Selected Students from One College

Solution

For the bootstrapping process, we will form samples of size 10 for convenience. Note that the sampling is with replacement, so once an age is selected, the age is returned back to the dataset and then that particular age might be selected again as part of the sample.

Imagine forming hundreds or thousands of such samples, each of sample size 10. For each one of these samples, calculate the sample mean (see column labeled “Sample Means”).

In the example shown in Figure 4.4, only 20 samples are taken for space considerations since showing the results of thousands of samples is not practical for this text. However, typically a bootstrapping process involves many such samples—on the order of thousands of samples—made possible by using software (such as Excel, Python, or R).

Sample Number	Random Samples of Size 10										Sample Means	Sorted Sample Means
1	37	19	34	31	34	26	26	21	20	21	26.9	21.9
2	37	19	24	37	19	31	29	34	29	28	28.7	22.2
3	31	19	20	37	20	26	25	24	37	23	26.2	22.7
4	22	24	20	22	20	24	37	19	24	26	23.8	22.9
5	23	24	19	34	20	31	22	23	28	22	24.6	23.8
6	34	19	31	37	21	21	25	28	19	21	25.6	24.4
7	25	24	21	29	21	23	21	34	24	37	25.9	24.6
8	22	26	37	20	22	21	22	29	37	25	26.1	24.6
9	23	22	31	22	24	37	29	21	22	20	25.1	25.1
10	23	31	19	34	37	22	25	20	37	37	28.5	25.5
11	29	31	31	22	37	23	23	28	20	21	26.5	25.6
12	31	22	19	22	28	22	21	23	37	37	26.2	25.9
13	21	20	24	28	21	20	22	22	21	23	22.2	25.9
14	22	37	21	31	19	26	19	23	23	34	25.5	26.1
15	29	31	22	25	26	19	22	26	25	21	24.6	26.2
16	26	20	23	19	23	26	19	29	19	23	22.7	26.2
17	24	31	19	21	19	22	19	19	21	24	21.9	26.5
18	21	20	20	34	34	21	25	23	22	24	24.4	26.9
19	20	26	22	24	21	24	21	26	26	19	22.9	28.5
20	21	31	19	21	34	24	25	28	28	28	25.9	28.7

Figure 4.4 Bootstrap Samples and Corresponding Sample Means

Next, sort the sample means from smallest to largest (see column labeled "Sorted Sample Means").

The last step is to calculate the 90% confidence interval based on these sorted sample means. For a 90% confidence interval, find the percentiles P_5 and P_{95}. These values establish the 90% confidence interval.

To find the 5th percentile (P_5), multiply the percentile (5%) times the number of data values, which is 20. The result is 1, so to find the 5th percentile, add the first and second data values together and divide by 2. For this example, add 21.9 to 22.2 and divide by 2. The result is 22.05.

To find the 95th percentile (P_{95}), multiply the percentile (95%) times the number of data values, which is 20. The result is 19, so to find the 95th percentile, add the 19th and 20th data values together and divide by 2. For this example, add 28.5 to 28.6 and divide by 2. The result is 28.6.

Based on the bootstrapping method, the 90% confidence interval is (22.05, 28.6).

Python provides a function called `bootstrap()` as part of the `scipy.stats` library to automate this bootstrapping process.

Within the `bootstrap()` function, the user can specify the number of resamples to be performed as part of the bootstrapping process. In this example, we will use 50,000 samples by way of the *n_resample* parameter.

PYTHON CODE

```
from scipy.stats import bootstrap
import numpy as np
```

```
#define random sample of ages
ages = [22, 23, 25, 31, 24, 21, 28, 23, 21, 20, 22, 19, 34, 19, 37, 26, 29, 21,
24, 20]

#convert ages to sequence
ages = (ages,)

#use bootstrap function for confidence interval for the mean
conf_interval = bootstrap(ages, np.mean, confidence_level=0.95,
                          random_state=1, n_resamples = 50000,
method='percentile')

#print the confidence interval
print(conf_interval.confidence_interval)
```

The resulting output will look like this:

```
ConfidenceInterval(low=22.45, high=26.7)
```

Using 50,000 bootstrapped samples, a 95% confidence interval is generated as (22.45, 26.7).

EXPLORING FURTHER

Simulating Confidence Intervals

A number of websites and resources, such as the online textbook Online Statistics Education: A Multimedia Course of Study (https://openstax.org/r/estimation) (http://onlinestatbook.com/) (project leader: David M. Lane, Rice University), allow the user to simulate confidence intervals for means or proportions using simulated sample data. It is useful to observe how many intervals contain an assumed value for the population mean or population proportion. The Stapplet (https://openstax.org/r/stapplet) website provides a similar tool.

Using Python to Calculate Confidence Intervals

When calculating confidence intervals, data scientists typically make use of technology to help streamline and automate the analysis. Python provides built-in functions for confidence interval calculations, and several examples are shown in Table 4.5. Students are encouraged to try examples themselves and experiment to use Python to assist with these calculations.

Table 4.5 provides a summary of various functions available within the `SciPy library` (https://openstax.org/r/spy) for confidence interval calculations:

Usage	Python Function Name	Syntax
Calculate confidence interval for the mean when population standard deviation is known, given sample mean, population standard deviation, and sample size (uses normal distribution).	`norm.interval()`	`norm.interval(conf_level, sample_mean,standard_dev/ sqrt (n))`
Calculate confidence interval for the mean when population standard deviation is unknown, given sample mean, sample standard deviation, and sample size (uses t-distribution).	`t.interval()`	`t.interval(conf_level, degrees_freedom, sample_mean,standard_dev/ sqrt (n))`
Calculate confidence interval for a proportion (uses normal distribution).	`proportion_confint()`	`proportion_confint(success, sample_size, 1 - confidence_level)`

Table 4.5 Python Functions for Confidence Intervals

EXAMPLE 4.9

Problem

Repeat Example 4.3 to calculate a 90% confidence interval, but use Python functions to calculate the confidence interval.

Recall from Example 4.3, the sample mean is 12.5 hours with a population standard deviation of 6.3 hours. The sample size is 50 students, and we are interested in calculating a 90% confidence interval.

Solution

Use the Python function `norm.interval()` as shown:

PYTHON CODE

```
import scipy.stats as stats
import numpy as np
import math

# Enter sample mean, population standard deviation and sample size
sample_mean = 12.5
population_standard_deviation = 6.3
sample_size = 50
```

```
# Confidence level
confidence_level = 0.90

# standard error
standard_error = population_standard_deviation / math.sqrt(sample_size)

# Calculate confidence interval using norm.interval function
stats.norm.interval(confidence_level, sample_mean, standard_error)
```

The resulting output will look like this:

```
(11.03451018636739, 13.965489813632608)
```

EXAMPLE 4.10

Problem

Repeat Example 4.4 to calculate a 99% confidence interval, but use Python functions to calculate the confidence interval.

Recall from Example 4.4, the sample mean is 15.8 miles with a standard deviation of 3.2 miles. The sample size is 26 employees, and we are interested in calculating a 99% confidence interval.

Solution

Use the Python function `t.interval()` as shown:

PYTHON CODE

```
# Enter sample mean, sample standard deviation, and sample size
sample_mean = 15.8
sample_standard_deviation = 3.2
sample_size = 26

# Degrees of freedom (sample size - 1)
degrees_of_freedom = sample_size - 1

# Confidence level
confidence_level = 0.99

# standard error
standard_error = sample_standard_deviation / math.sqrt(sample_size)
```

```
# Calculate confidence interval using t.interval function
t.interval(confidence_level, degrees_of_freedom, sample_mean, standard_error)
```

The resulting output will look like this:

```
(14.050684356083625, 17.549315643916376)
```

4.2 Hypothesis Testing

Learning Outcomes

By the end of this section, you should be able to:

- 4.2.1 Apply hypothesis testing methods to test statistical claims involving one sample.
- 4.2.2 Use Python to assist with hypothesis testing calculations.
- 4.2.3 Conduct hypothesis tests to compare two means, two proportions, and matched pairs data.

Constructing confidence intervals is one method used to estimate population parameters. Another important statistical method to test claims regarding population parameters is called *hypothesis testing*.

Hypothesis testing is an important tool for data scientists in that it is used to draw conclusions using sample data, and it is also used to quantify uncertainty associated with these conclusions.

As an example to illustrate the use of hypothesis testing in a data science applications, consider an online retailer seeking to increase revenue though a text messaging ad campaign. The retailer is interested in examining the hypothesis that a text messaging ad campaign leads to a corresponding increase in revenue. To test out the hypothesis, the retailer sends out targeted text messages to certain customers but does not send the text messages to another group of customers. The retailer then collects data for purchases made by customers who received the text message ads versus customers who did not receive the ads. Using hypothesis testing methods, the online retailer can come to a conclusion regarding the marketing campaign to transmit ads via text messaging. **Hypothesis testing** involves setting up a null hypothesis and alternative hypotheses based on a claim to be tested and then collecting sample data. A **null hypothesis** represents a statement of no effect or no change in the population; the null hypothesis provides a statement that the value of a population parameter is equal to a value. An **alternative hypothesis** is a complementary statement to the null hypothesis and makes a statement that the population parameter has a value that differs from the null hypothesis.

Hypothesis testing allows a data scientist to assess the likelihood of observing results under the assumption of the null hypothesis and make judgements about the strength of the evidence against the null hypothesis. It provides the foundation for various data science investigations and plays a role at different stages of the analyses such as data collection and validation, modeling related tasks, and determination of statistical significance.

For example, a local restaurant might claim that the average delivery time for food orders is 30 minutes or less. To test the claim, food orders can be placed and corresponding delivery times recorded. If the sample data appears to be inconsistent with the null hypothesis, then the decision is to reject the null hypothesis.

Testing Claims Based on One Sample

For our discussion, we will examine hypothesis testing methods for claims involving means or proportions. (Hypothesis testing can also be performed for standard deviation or variance, though that is beyond the scope of this text.)

The steps for hypothesis testing are as follows:

1. Set up a null and alternative hypothesis based on the claim. Identify whether the null or the alternative hypothesis represents the claim.
2. Collect relevant sample data to investigate the claim.
3. Determine the correct distribution to be used to analyze the hypothesis test (e.g., normal distribution or t-distribution).
4. Analyze the sample data to determine if the sample data is consistent with the null hypothesis. This will involve statistical calculations involving what is called a *test statistic* and a *p*-value (discussed in the next section).
5. If the sample data is inconsistent with the null hypothesis, then the decision is to "reject the null hypothesis." If the sample data is consistent with the null hypothesis, then the decision is to "fail to reject the null hypothesis."

The first step in the hypothesis testing process is to write what is a called a "null" hypothesis and an "alternative" hypothesis. These hypotheses will be statements regarding an unknown population parameter, such as a population mean or a population proportion. These two hypotheses are based on the claim under study and will be written as complementary statements to each other. When one of the hypotheses is true, the other hypothesis must be false (since they are complements of one another). Note that either the null or alternative hypothesis can represent the claim.

The null hypothesis is labeled as H_0 and is a statement of equality—for example, the null hypothesis might state that the mean time to complete an exam is 42 minutes. For this example, the null hypothesis would be written as:

$$H_0 : \mu = 42 \text{ minutes}$$

Where the notation μ refers to the population mean time to complete an exam.
The null hypothesis will always contain one of the following three symbols: $=, \leq, \text{or } \geq$.

The alternative hypothesis is labeled as H_a and will be the complement of the null hypothesis. For example, if the null hypothesis contains an equals symbol, then the alternative hypothesis will contain a not equals symbol.

If the null hypothesis contains a $\geq$ symbol, then the alternative hypothesis will contain a $<$ symbol.
If the null hypothesis contains a $\leq$ symbol, then the alternative hypothesis will contain a $>$ symbol.

Note that the alternative hypothesis will always contain one of the following three symbols: $\neq, <, \text{or } >$.

To write the null and alternative hypotheses, start off by translating the claim into a mathematical statement involving the unknown population parameter. Depending on the wording, the claim can be placed in either the null or the alternative hypothesis. Then write the complement of this statement as the other hypothesis (see Table 4.6).

When translating claims about a population mean, use the symbol μ as part of the null and alternative hypotheses. When translating claims about a population proportion, use the symbol *p* as part of the null and alternative hypotheses.

Table 4.6 provides the three possible setups for the null and alternative hypotheses when conducting a one-sample hypothesis test for a population mean. Notice that for each setup, the null and alternative hypotheses are complements of one another. Note that the value of "*k*" will be replaced by some numerical constant taken from the stated claim.

Setup A	Setup B	Setup C
$Ho: \mu = k$	$Ho: \mu \leq k$	$Ho: \mu \geq k$
$Ha: \mu \neq k$	$Ha: \mu > k$	$Ha: \mu < k$

Table 4.6 Possible Setups for the Null and Alternative Hypotheses for a Population Mean (One Sample)

Table 4.7 provides the three possible setups for the null and alternative hypotheses when conducting a one-sample hypothesis test for a population proportion. Note that the value of "k" will be replaced by some numerical constant taken from the stated claim.

Setup D	Setup E	Setup F
$Ho: p = k$	$Ho: p \leq k$	$Ho: p \geq k$
$Ha: p \neq k$	$Ha: p > k$	$Ha: p < k$

Table 4.7 Possible Setups for the Null and Alternative Hypotheses for a Population Proportion (One Sample)

Follow these steps to determine which of these setups to use for a given hypothesis test:

1. Determine if the hypothesis test involves a claim for a population mean or population proportion. If the hypothesis test involves a population mean, use either Setup A, B, or C.
2. If the hypothesis test involves a population proportion, use either Setup D, E, or F.
3. Translate a phrase from the claim into a mathematical symbol and then match this mathematical symbol with one of the setups in Table 4.6 and Table 4.7.

For example, if a claim for a hypothesis test for a mean references the phrase "at least," note that the phrase "at least" corresponds to a *greater than or equals symbol*. A greater than or equals symbol appears in Setup C in Table 4.6, so that would be the correct setup to use.

If a claim for a hypothesis test for a proportion references the phrase "different than," note that the phrase "different than" corresponds to a *not equals* symbol. A not equals symbol appears in Setup D in Table 4.6, so that would be the correct setup to use.

EXAMPLE 4.11

Problem

Write the null and alternative hypotheses for the following claim:
An auto repair facility claims the average time for an oil change is less than 20 minutes.

Solution

Note that the claim involves the phrase "less than," which translates to a less than symbol. A less than symbol must be used in the alternative hypothesis. The complement of less than is a greater than or equals symbol, which will be used in the null hypothesis. The claim refers to the population mean time for an oil change, so the symbol μ will be used. Notice that this setup will correspond to Setup C from Table 4.6. Thus the setup of the null and alternative hypothesis will be as follows:

$$H_0: \mu \geq 20$$
$$H_a: \mu < 20$$

EXAMPLE 4.12

Problem

Write the null and alternative hypotheses for the following claim:

A medical researcher claims that the proportion of adults in the United States who are smokers is at most 25%.

Solution

Note that the claim involves the phrase "at most," which translates to a less than or equals symbol. A less than or equals symbol must be used in the null hypothesis. The complement of less than or equals is a greater than symbol, which will be used in the alternative hypothesis. The claim refers to the population proportion of smokers, so the symbol p will be used. Notice that this setup will correspond to Setup E from Table 4.7. Thus, the setup of the null and alternative hypotheses will be as follows:

$$H_0 : p \leq 0.25$$
$$H_a : p > 0.25$$

The basis of the hypothesis test procedure is to assume that the null hypothesis is true to begin with and then examine the sample data to determine if the sample data is consistent with the null hypothesis. Based on this analysis, you will decide to either reject the null hypothesis or fail to reject the null hypothesis.

It is important to note that these are the only two possible decisions for any hypothesis test, namely:

- Reject the null hypothesis, or
- Fail to reject the null hypothesis.

Keep in mind that this decision will be based on statistical analysis of sample data, so there is a small chance of coming to a wrong decision. The only way to be absolutely certain of your decision is to test the entire population, which is typically not feasible. Since the hypothesis test decision is based on a sample, there are two possible errors that can be made by the researcher:

- The researcher rejects the null hypothesis when in fact the null hypothesis is actually true, or a **Type I error**.
- A researcher fails to reject the null hypothesis when the null hypothesis is actually false, or a **Type II error**.

In hypothesis testing, the maximum allowed probability of making a Type I error is called the **level of significance**, denoted by the Greek letter alpha, α. From a practical standpoint, the level of significance is the probability value used to determine when the sample data indicates significant evidence against the null hypothesis. This level of significance is typically set to a small value, which indicates that the researcher wants the probability of rejecting a true null hypothesis to be small. Typical values of the level of significance used in hypothesis testing are as follows: $\alpha = 0.01$, $\alpha = 0.05$, or $\alpha = 0.10$.

Once the null and alternative hypotheses are written and a level of significance is selected, the next step is for the researcher to collect relevant sample data from the population under study. It's important that a representative random sample is selected, as discussed in Handling Large Datasets. After the sample data is collected, two calculations are performed: *test statistic* and *p*-value.

The **test statistic** is a numerical value used to assess the strength of evidence against a null hypothesis and is calculated from sample data that is used in hypothesis testing. As noted in Estimating Parameters with Confidence Intervals, a *sample statistic* is a numerical value calculated from a sample of observations drawn from a larger population such as a sample mean or sample proportion. The sample statistic is converted to a **standardized test statistic** such as a *z*-score or t-score based on the assumption that the null hypothesis is

true.

Once the test statistic has been determined, a probability, or *p*-value, is created. A ***p*-value** is the probability of obtaining a sample statistic with a value as extreme as (or more extreme than) the value determined by the sample data under the assumption that the null hypothesis is true. To calculate a *p*-value, we will find the area under the relevant probability distribution, such as finding the area under the normal distribution curve or the area under the t-distribution curve. *Since the p-value is a probability, any p-value must always be a numerical value between 0 and 1 inclusive*.

When calculating a *p*-value as the area under the probability distribution curve, the corresponding area will be determined using the location under the curve, which favors the rejection of the null hypothesis as follows:

1. If the alternative hypothesis contains a "less than" symbol, the hypothesis test is called a "left tailed" test and the *p*-value will be calculated as the area to the *left* of the test statistic.
2. If the alternative hypothesis contains a "greater than" symbol, the hypothesis test is called a "right tailed" test and the *p*-value will be calculated as the area to the *right* of the test statistic.
3. If the alternative hypothesis contained a "not equals" symbol, the hypothesis test is called a "two tailed" test and the *p*-value will be calculated as the sum of the area to the *left* of the negative test statistic and the area to the *right* of the positive test statistic.

The smaller the p-value, the more the sample data deviates from the null hypothesis, and this is more evidence to indicate that the null hypothesis should be rejected.

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance and applying the following rule:

- If the *p*-value $\leq$ level of significance, then the decision is to "reject the null hypothesis."
- If the *p*-value $>$ level of significance, then the decision is to "fail to reject the null hypothesis."

Recall that these are the only two possible decisions that a researcher can deduce when conducting a hypothesis test.

Often, we would like to translate these decisions into a conclusion that is easier to interpret for someone without a statistical background. For example, a decision to "fail to reject the null hypothesis" might be difficult to interpret for those not familiar with the terminology of hypothesis testing. These decisions can be translated into concluding statements such as those shown in Table 4.8.

	Select the decision from the hypothesis test:	Is the claim in the null or the alternative hypothesis?	Then, use this concluding statement:
Row 1	Reject the null hypothesis	Claim is the null hypothesis	*There is enough evidence to reject the claim.*
Row 2	Reject the null hypothesis	Claim is the alternative hypothesis	*There is enough evidence to support the claim.*
Row 3	Fail to reject the null hypothesis	Claim is the null hypothesis	*There is not enough evidence to reject the claim.*
Row 4	Fail to reject the null hypothesis	Claim is the alternative hypothesis	*There is not enough evidence to support the claim.*

Table 4.8 How to Establish a Conclusion from a Hypothesis Test

Testing Claims for the Mean When the Population Standard Deviation Is Known

Recall that in the discussion for confidence intervals we examined the confidence interval for the population mean when the population standard deviation is known (normal distribution is used) or when the population standard deviation is unknown (t-distribution is used). We follow the same procedure when conducting hypothesis tests.

In this section, we discuss hypothesis testing for the mean when the population standard deviation is known. Although the population standard deviation is typically unknown, in some cases a reasonable estimate for the population standard deviation can be obtained from past studies or historical data.

Here are the requirements to use this procedure:

a. A random sample is selected from the population.
b. The sample size is at least 30, or the underlying population is known to follow a normal distribution.
c. The population standard deviation (σ) is known.

When these requirements are met, the normal distribution is used as the basis for conducting the hypothesis test.

For this procedure, the test statistic will be a *z*-score and the standardized test statistic is calculated as follows:

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

Where:
$\bar{x}$ is the sample mean.
μ is the hypothesized value of the population mean, which comes from the null hypothesis.
σ is the population standard deviation.
n is the sample size.

Example 4.13 provides a step-by-step example to illustrate the hypothesis testing process.

EXAMPLE 4.13

Problem

A random sample of 100 college students finds that students spend an average of 14.4 hours per week on social media. Use this sample data to test the claim that college students spend at least 15 hours per week on social media. Use a level of significance of 0.05 and assume the population standard deviation is 3.25 hours.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the claim involves the phrase “at least,” which translates to a greater than or equals symbol. A greater than or equals symbol must be used in the null hypothesis. The complement of greater than or equals is a less than symbol, which will be used in the alternative hypothesis. The claim refers to the population mean time that college students spend on social media, so the symbol μ will be used. The claim corresponds to the null hypothesis.

Notice that this setup will correspond to Setup C from Table 4.6. Thus the setup of the null and alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : \mu &\geq 15 \text{ [claim]} \\ H_a : \mu &< 15 \end{aligned}$$

Since the population standard deviation is known in this example, the normal distribution will be used to conduct the hypothesis test.

Next, calculate the standardized test statistic:

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} = \frac{14.4 - 15}{3.25/\sqrt{100}} = -1.846$$

Notice that since the alternative hypothesis contains a "less than" symbol, the hypothesis test is called a "left tailed" test and the *p*-value will be calculated as the area to the *left* of the test statistic. The area under the standard normal curve to the left of a *z*-score of -1.846 is approximately 0.0324. Thus, the *p*-value for this hypothesis test is 0.0324.

The area under the standard normal curve can be found using software such as Excel, Python, or R.

For example, in Python, the area under the normal curve to the left of a *z*-score of -1.846 can be obtained using the function `norm.cdf()`.

The syntax for using this function is
`norm.cdf(x, mean, standard_deviation)`

Where:
`x` is the measurement of interest—in this case, this will be the test statistic of -1.846.
`mean` is the mean of the standard normal distribution, which is 0.
`standard_deviation` is the standard deviation of the standard normal distribution, which is 1.

Here is the Python code to calculate the *p*-value for this example:

PYTHON CODE

```
# from scipy.stats import norm
from scipy.stats import norm

# define parameters x, mean and standard_deviation:
x = -1.846
mean = 0
standard_deviation = 1
# use norm.cdf function to calculate normal probability - note this is the area
to the left of the x-value
# subtract this result from 1 to obtain area to the right of the x-value
# use round function to round answer to 4 decimal places
round (norm.cdf(x, mean, standard_deviation), 4)
```

The resulting output will look like this:

```
0.0324
```

The final step in the hypothesis testing procedure is to come to a final decision regarding the null

hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.032 and the level of significance is 0.05: since the *p*-value $\leq$ level of significance, then the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis; the claim corresponds to the null hypothesis. This scenario corresponds to Row 1 in Table 4.8, which means "there is enough evidence to reject the claim" that college students spend at least 15 hours per week on social media.

Testing Claims for the Mean When the Population Standard Deviation Is Unknown

In this section, we discuss one sample hypothesis testing for the mean when the population standard deviation is unknown. This is a more common application of hypothesis testing for the mean in that the population standard deviation is typically unknown; however, the sample standard deviation can be calculated based on the sample data.

Here are the requirements to use this procedure:

a. A random sample is selected from the population.
b. The sample size is at least 30, or the underlying population is known to follow a normal distribution.
c. The population standard deviation (σ) is unknown; the sample standard deviation (s) is known.

When these requirements are met, the t-distribution is used as the basis for conducting the hypothesis test.

For this procedure, the test statistic will be a t-score and the standardized test statistic is calculated as follows:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

Where:
$\bar{x}$ is the sample mean.
μ is the hypothesized value of the population mean, which comes from the null hypothesis.
s is the sample standard deviation.
n is the sample size.

Example 4.14 provides a step-by-step example to illustrate this process.

EXAMPLE 4.14

Problem

A smartphone manufacturer claims that the mean battery life of its latest smartphone model is 25 hours. A consumer group decides to test this hypothesis and collects a sample of 50 smartphones and determines that the mean battery life of the sample is 24.1 hours with a sample standard deviation of 4.1 hours. Use this sample data to test the claim made by the smartphone manufacturer. Use a level of significance of 0.10 for this analysis.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the claim involves the phrase "is 25 hours," where "is" will translate to an equals symbol. An equals symbol must be used in the null hypothesis. The complement of an "equals" symbol is a "not equals" symbol, which will be used in the alternative hypothesis. The claim refers to the population mean battery life of smartphones, so the symbol μ will be used. The claim corresponds to the null hypothesis. Notice that this setup will correspond to Setup A from Table 4.6. Thus the setup of the null and

alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : \mu &= 25 \text{ [claim]} \\ H_a : \mu &\neq 25 \end{aligned}$$

Since the population standard deviation is unknown in this example, the t-distribution will be used to conduct the hypothesis test.

Next, calculate the standardized test statistic as follows:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{24.1 - 25}{4.1/\sqrt{50}} = -1.552$$

Notice that since the alternative hypothesis contains a "not equals" symbol, the hypothesis test is called a "two tailed" test and the *p*-value will be calculated as the area to the *left* of the negative test statistic plus the area to the right of the positive test statistic. The area under the t-distribution curve to the left of a t-score of -1.552 is approximately 0.0635 (using degrees of freedom $df = 49$). The area under the t-distribution curve to the right of a t-score of 1.552 is approximately 0.0635. Thus, the *p*-value for this hypothesis test is the sum of the areas, which is 0.127.

The area under the t-distribution curve can be found using software.

For example, in Python, the area under the t-distribution curve to the left of a t-score of -1.552 can be obtained using the function `t.cdf()`

The syntax for using this function is
`t.cdf(x, df)`

Where:
`x` is the measurement of interest—in this case, this will be the test statistic of -1.846.
`df` is the degrees of freedom.

Here is the Python code to calculate the *p*-value for this example:

PYTHON CODE

```python
# from scipy.stats import t
from scipy.stats import t

# define parameters test_statistic and degrees of freedom df
x = -1.552
df = 49

# use t.cdf function to calculate area under t-distribution curve - note this is
the area to the left of the x-value
# subtract this result from 1 to obtain area to the right of the x-value
# use round function to round answer to 4 decimal places
round (t.cdf(x, df=49), 4)
```

The resulting output will look like this:

```
0.0635
```

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.127 and the level of significance is 0.10: since the *p*-value > level of significance, then the decision is to "fail to reject the null hypothesis."

Conclusion: The decision is to fail to reject the null hypothesis; the claim corresponds to the null hypothesis. This scenario corresponds to Row 3 in Table 4.8, which means "there is not enough evidence to reject the claim" that the mean battery life of its latest smartphone model is 25 hours.

Testing Claims for the Proportion

In this section, we discuss one sample hypothesis testing for proportions. Recall that a sample proportion is measuring "how many out of the total." For example, a medical researcher might be interested in testing a claim that the proportion of patients experiencing side effects from a new blood pressure–lowering medication is less than 4%.

Here are the requirements to use this procedure:

a. A random sample is selected from the population.
b. Verify that the normal approximation to the binomial distribution is appropriate by ensuring that both $n\hat{p}$ and $n(1-\hat{p})$ are both at least 5.

The sample proportion $\hat{p}$ is calculated as the number of successes divided by the sample size:

$$\hat{p} = \frac{x}{n}$$

When these requirements are met, the normal distribution is used as the basis for conducting the hypothesis test.

For this procedure, the test statistic will be a *z*-score and the standardized test statistic is calculated as follows:

$$z = \frac{\hat{p} - p}{\sqrt{p(1-p)/n}}$$

Where:
$\hat{p}$ represents the sample proportion.
p is the hypothesized value of the population proportion, which is taken from the null hypothesis.
n is the sample size.

Here is a step-by-step example to illustrate this hypothesis testing process.

EXAMPLE 4.15

Problem

A college professor claims that the proportion of students using artificial intelligence (AI) tools as part of their coursework is less than 45%. To test the claim, the professor selects a sample of 200 students and surveys the students to determine if they use AI tools as part of their coursework. The results from the survey indicate that 74 out of 200 students use AI tools. Use this sample data to test the claim made by the

college professor. Use a level of significance of 0.05 for this analysis.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypothesis based on the claim. Notice that the claim involves the phrase "less than," which will translate to a less than symbol. A less than symbol must be used in the alternative hypothesis. The complement of a "less than" symbol is a "greater than or equal" symbol, which will be used in the null hypothesis. The claim refers to the population proportion of students using AI tools, so the symbol p will be used. Notice that this setup will correspond to Setup F from Table 4.7. Thus, the setup of the null and alternative hypotheses will be as follows:

$$H_0 : p \geq 0.45 \quad H_a : p < 0.45 \text{ [claim]}$$

The sample proportion can be calculated as follows:

$$\hat{p} = \frac{x}{n} = \frac{74}{200} = 0.37$$

Verify that the normal approximation to the binomial distribution is appropriate by ensuring that both $n\hat{p}$ and $n(1 - \hat{p})$ are both at least 5, where $\hat{p}$ represents the sample proportion.

For this example, $n\hat{p} = (200)0.37 = 74$, and $n(1 - \hat{p}) = (200)(1 - 0.37) = 126$. Both of these results are at least 5, which verifies that the normal approximation to the binomial distribution is appropriate.

Since the requirements for a hypothesis test for a proportion have been met, the normal distribution will be used to conduct the hypothesis test.

Next, calculate the standardized test statistic as follows:

$$z = \frac{\hat{p} - p}{\sqrt{p(1-p)/n}} = \frac{0.37 - 0.45}{\sqrt{0.45(1-0.45)/200}} = -2.274$$

Notice that since the alternative hypothesis contains a "not equals" symbol, the hypothesis test is called a "left tailed" test and the *p*-value will be calculated as the area to the left of the test statistic. The area under the normal curve to the *left* of a *z*-score of -2.274 is approximately 0.012. Thus, the *p*-value for this hypothesis test is 0.012.

Here is the Python code to calculate the *p*-value for this example:

PYTHON CODE

```
# from scipy.stats import norm
from scipy.stats import norm

# define parameters x, mean, and standard_deviation:
x = -2.274
mean = 0
standard_deviation = 1
# use norm.cdf function to calculate normal probability - note this is the area
to the left of the x-value
```

```
# subtract this result from 1 to obtain area to the right of the x-value
# use round function to round answer to 3 decimal places
round (norm.cdf(x, mean, standard_deviation), 4)
```

The resulting output will look like this:

```
0.0115
```

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.012 and the level of significance is 0.05: since the *p*-value < level of significance, then the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis; the claim corresponds to the alternative hypothesis. This scenario corresponds to Row 2 in Table 4.8, which means "there is enough evidence to support the claim" that the proportion of students using artificial intelligence (AI) tools as part of their coursework is less than 45%.

Using Python to Conduct Hypothesis Tests

Python provides several functions to assist with hypothesis testing, as the `ztest()` can be used for hypothesis testing for the mean when population standard deviation is known and another Python function called `ttest_1samp()` can be used for hypothesis testing for the mean when population standard deviation is unknown. See the example:

EXAMPLE 4.16

Problem

A medical researcher wants to test a claim that the average oxygen level for females is greater than 75 mm Hg. In order to test the claim, a sample of 15 female patients is selected and the following oxygen levels are recorded:

76, 75, 74, 81, 76, 77, 71, 74, 73, 79

Use the `ttest_1samp()` function in Python to conduct a hypothesis test to test the claim. Assume a level of significance of 0.05.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the claim involves the phrase "greater than 75 mm Hg," where "greater than" will translate to a greater than symbol. A greater than symbol must be used in the alternative hypothesis. The complement of a "greater than" symbol is a "less than or equals" symbol, which will be used in the null hypothesis. The claim refers to the population mean oxygen level, so the symbol μ will be used. The claim corresponds to the alternative hypothesis. Notice that this setup will correspond to Setup B from Table 4.6. Thus the setup of the null and alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : \mu &\leq 75 \\ H_a : \mu &> 75 \text{ [claim]} \end{aligned}$$

Since the population standard deviation is unknown in this example, the t-distribution will be used to conduct the hypothesis test.

To use the `ttest_1samp()` function in Python, the syntax is as follows:

`Ttest_1samp(data_array, null_hypothesis_mean)`

The function will then return the value of the test statistic and the two-tailed *p*-value.

In this example we are interested in the right tailed *p*-value, so the Python program will calculate one-half of the *p*-value returned by the function.

Here is the Python code for this example:

PYTHON CODE

```
import scipy.stats as stats
import numpy as np

# Enter data array
oxygen_level = np.array([76, 75, 74, 81, 76, 77, 71, 74, 73, 79])

# Hypothesized population mean
null_hypothesis_mean = 75

# Perform one-sample t-test
test_stat, two_tailed_p_value = stats.ttest_1samp(oxygen_level,
null_hypothesis_mean)

# Obtain the one tailed p-value
one_tailed_p_value = two_tailed_p_value / 2

# print out test statistic and p-value and round to 3 decimal places
print("Test Statistic = ", test_stat)
print("p-value = ", one_tailed_p_value)
```

The resulting output will look like this:

```
Test Statistic = 0.6512171447631733
p-value = 0.2655900985964262
```

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.266 and the level of significance is 0.05: since the *p*-value > level of significance, the decision is to "fail to reject the null hypothesis."

Conclusion: The decision is to fail to reject the null hypothesis; the claim corresponds to the alternative hypothesis. This scenario corresponds to Row 4 in Table 4.8, which means "there is not enough evidence to support the claim" that the average oxygen level for female patients is greater than 75 mm Hg.

Hypothesis Testing for Two Samples

In the previous section, we examined methods to conduct hypothesis testing for one sample. Researchers and data scientists are also often interested in conducting hypothesis tests for two samples. A researcher might want to compare two means or two proportions to determine if two groups are significantly different from one another. For example, a medical researcher might be interested in comparing the mean blood pressure between two groups of individuals: one group that is taking a blood pressure–lowering medication and another group that is taking a placebo. A government researcher might be interested in comparing the proportion of smokers for male versus female adults.

The same general hypothesis testing procedure will be used for two samples, but the setup of the null and alternative hypotheses will be structured to reflect the comparison of two means or two proportions.

As a reminder, the general hypothesis testing method introduced in the previous section includes the following steps:

1. Set up a null and an alternative hypothesis based on the claim.
2. Collect relevant sample data to investigate the claim.
3. Determine the correct distribution to be used to analyze the hypothesis test (e.g., normal distribution or t-distribution).
4. Analyze the sample data to determine if the sample data is consistent with the null hypothesis. This will involve statistical calculations involving a test statistic and a *p*-value, which are discussed next.
5. If the sample data is inconsistent with the null hypothesis, then the decision is to "reject the null hypothesis." If the sample data is consistent with the null hypothesis, then the decision is to "fail to reject the null hypothesis."

As discussed for one-sample hypothesis testing, when writing the null and alternative hypotheses, start off by translating the claim into a mathematical statement involving the unknown population parameters. The claim could correspond to either the null or the alternative hypothesis. Then write the complement of this statement in order to write the other hypothesis. The difference now as compared to the previous section is that the hypotheses will be comparing two means or two proportions instead of just one mean or one proportion.

When translating claims about a population mean, use the symbols μ_1 and μ_2 as part of the null and alternative hypotheses to represent the two unknown population means. When translating claims about a population proportions, use the symbols $p1$ and $p2$ as part of the null and alternative hypotheses to represent the two unknown population proportions. Refer to Table 4.9 and Table 4.10 for the various setups to construct

the null and alternative hypotheses for two-sample hypothesis testing.

Setup A	Setup B	Setup C
$H_0 : \mu_1 = \mu_2$	$H_0 : \mu_1 \leq \mu_2$	$H_0 : \mu_1 \geq \mu_2$
$H_a : \mu_1 \neq \mu_2$	$H_a : \mu_1 > \mu_2$	$H_a : \mu_1 < \mu_2$

Table 4.9 Three Possible Setups for the Null and Alternative Hypotheses for a Population Mean (Two Samples)

Setup D	Setup E	Setup F
$H_0 : p_1 = p_2$	$H_0 : p_1 \leq p_2$	$H_0 : p_1 \geq p_2$
$H_a : p_1 \neq p_2$	$H_a : p_1 > p_2$	$H_a : p_1 < p_2$

Table 4.10 Three Possible Setups for the Null and Alternative Hypotheses for a Population Proportion (Two Samples)

Comparing Two Means

In this section, we discuss two-sample hypothesis testing for the mean when the population standard deviations are unknown. Typically, in real-world applications, population standard deviations are unknown, so this method has considerable practical appeal.

One other consideration when conducting hypothesis testing for two samples is to determine if the two samples are dependent or independent. Two samples are considered to be **independent samples** when the sample values from one population are not related to the sample values taken from the second population. In other words, there is not an inherent paired or matched relationship between the two samples. Two samples are considered to be **dependent samples** if the samples from one population can be paired or matched to the samples taken from the second population. Dependent samples are sometimes also called *paired samples*.

For example, if a researcher were to collect data for the ages for a random sample of 15 men and a random sample of 15 women, there would be no inherent relationship between these two samples, and so these samples are considered independent samples.

On the other hand, let's say a researcher were to collect data on the ages of a random sample of 15 adults and also collect data on the ages of those adults' corresponding (15) spouses/partners; these would be considered dependent samples in that there is an inherent pairing of domestic partners.

EXAMPLE 4.17

Problem

Determine if the following samples are dependent or independent samples:

Sample #1: Scores on a statistics exam for 40 women college students

Sample #2: Scores on a statistics exam for 35 men college students

Solution

These two samples are independent because there is no inherent match-up between the women students from Sample #1 and the men students from Sample #2; it is not possible to analyze either of the samples as dependent, or paired, samples.

EXAMPLE 4.18

Problem

Determine if the following samples are dependent or independent samples:

Sample #1: Blood pressure readings on a sample of 50 patients in a clinical trial before taking any medication. Sample #2: Blood pressure readings on the same group of 50 patients after taking a blood pressure–lowering medication.

Solution

These two samples are dependent. The samples can be paired as before/after blood pressure readings for the same patient.

Here are the requirements to use this procedure:

a. The samples are random and independent.
b. For each sample, the sample sizes are both at least 30, or the underlying populations are known to follow a normal distribution.
c. The population standard deviations (σ) are unknown; the sample standard deviations (s) are known.

When these requirements are met, the t-distribution is used as the basis for conducting the hypothesis test.

In this discussion, we assume that the population variances are not equal, which leads to the following formula for the test statistic.

For this procedure, the test statistic will be a t-score and the standardized test statistic is calculated as follows:

$$t = \frac{\left(\bar{x}_1 - \bar{x}_2\right) - \left(\mu_1 - \mu_2\right)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Where:

$\bar{x}_1, \bar{x}_2$ are the sample means.

μ_1, μ_2 are the hypothesized values of the population means.

s_1, s_s are the sample standard deviations.

n_1, n_2 are the sample sizes.

Note: For the t-distribution, the degrees of freedom will be the smaller of $n_1 - 1$ and $n_2 - 1$.

Example 4.19 illustrates this hypothesis testing process.

EXAMPLE 4.19

Problem

A software engineer at a communications company claims that the average salary of women engineers is less than that of men engineers. To test the claim, a human resource administrator selects a random sample of 45 women engineers and 50 men engineers and collects statistical data as shown in Table 4.11. (Dollar amounts are in thousands of dollars.)

	Women Engineers	Men Engineers
Sample Mean	$\bar{x}_1 = 72700$	$\bar{x}_2 = 76900$
Sample Standard Deviation	$s_1 = 9800$	$s_2 = 10700$
Sample Size	$n_1 = 45$	$n_2 = 50$

Table 4.11 Sample Data for Women and Men Engineers

Use this sample data to test the claim made by the software engineer. Use a level of significance of 0.05 for this analysis.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the claim mentions "average salary of women engineers is less than that of men engineers, where "less than" will translate to a less than symbol. A less than symbol must be used in the alternative hypothesis. The complement of a "less than" symbol is a "greater than or equals" symbol, which will be used in the null hypothesis. Let μ_1 represent the population mean salary for women engineers and let μ_2 represent the population mean salary for men engineers. The null hypothesis can be considered to represent that we expect there to be "no difference" between the salaries of women and men engineers. Note that this setup will correspond to Setup C from Table 4.9. The claim corresponds to the alternative hypothesis. Thus, the setup of the null and alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : \mu_1 &\geq \mu_2 \\ H_a : \mu_1 &< \mu_2 \text{ [claim]} \end{aligned}$$

Since the population standard deviations are unknown in this example, the t-distribution will be used to conduct the hypothesis test.

Next, calculate the standardized test statistic as follows:

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{(72700 - 76900) - (0)}{\sqrt{\frac{9800^2}{45} + \frac{10700^2}{50}}} = -1.997$$

Notice that since the alternative hypothesis contains a "less than" symbol, the hypothesis test is called a "left tailed" test and the *p*-value will be calculated as the area under the t-distribution curve to the *left* of the test statistic. The area under the t-distribution curve to the *left* of a t-score of -1.997 (using degrees of freedom $df = 44$) is approximately 0.025. Thus, the *p*-value for this hypothesis test is 0.025.

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.025 and the level of significance is 0.04: since the *p*-value $<$ level of significance, then the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis; the claim corresponds to the alternative hypothesis. This scenario corresponds to Row 2 in Table 4.8, and thus "there is enough evidence to support the claim" that the average salary of women engineers is less than that of men engineers.

This problem can also be solved using the Python function called `ttest_ind_from_stats()`.

There is also a Python function to perform a two-sample t-test given raw data for the two groups, and this Python function is `ttest_ind()`.

For the function `ttest_ind_from_stats()`, the syntax is as follows:

PYTHON CODE

```
stats.ttest_ind_from_stats(sample_mean1, sample_standard_deviation1,
sample_size1, sample_mean2, sample_standard_deviation2, sample_size2)
```

Here is the Python code that can be used to solve Example 4.19. Note that this function returns the two-tailed *p*-value. Since Example 4.19 involves a left-tailed test, the desired *p*-value is one-half of the two-tailed *p*-value.

PYTHON CODE

```
import scipy.stats as stats
import numpy as np

#Example 4.19 - Two Sample t-test given sample means, sample standard deviations

sample_mean1 = 72700
sample_standard_deviation1 = 9800
sample_size1 = 45

sample_mean2 = 76900
sample_standard_deviation2 = 10700
sample_size2 = 50

test_statistic, two_tailed_p_value = stats.ttest_ind_from_stats(sample_mean1,
sample_standard_deviation1, sample_size1, sample_mean2,
sample_standard_deviation2, sample_size2)

left_tailed_p_value = two_tailed_p_value / 2

print("test statistic = ", "%.3f" % test_statistic)
print("p-value = ", "%.3f" % left_tailed_p_value)
```

The resulting output will look like this:

```
test statistic = -1.988
p-value = 0.025
```

Comparing Matched Pairs Data

Hypothesis Testing for Two Samples discussed hypothesis testing for two *independent samples*. In this section, we turn our attention to hypothesis testing for *dependent samples*, often called **matched pairs**, paired samples, or paired data.

Recall that two samples are considered to be dependent if the samples from one population can be paired, or matched, to the samples taken from the second population.

This analysis method is often used to analyze before/after data to determine if a significant change has occurred. In a clinical trial, a medical researcher may be interested in comparing before and after blood pressure readings on a patient to assess the efficacy of a blood pressure–lowering medication. Or an employer might compare scores on a safety exam for a sample of employees before and after employees have taken a course on safety procedures and practices. A parent might want to compare before/after SAT scores for students who have taken an SAT preparation course. This type of analysis can help to assess the effectiveness and value of such training.

When conducting a hypothesis test for paired data such as before/after data, a key step will be to calculate the "difference" for each pair of data values, as follows:

$$\text{Difference (d)} = \text{(before data)} - \text{(after data)}$$

Note that when calculating these differences, the result can be either a positive or a negative value.

Here are the requirements to use this procedure:

1. The samples are random and dependent (i.e., paired data).
2. The populations are normally distributed or the number of data pairs is at least 30.

When these requirements are met, the t-distribution is used as the basis for conducting the hypothesis test.

The test statistic will be the average of the differences across the data pairs (labeled as $\bar{d}$), where:

$$\bar{d} = \frac{\Sigma d}{n}$$

For this procedure, the standardized test statistic is calculated as follows:

$$t = \frac{\bar{d} - \mu_d}{s_d / \sqrt{n}}$$

Where:

$\bar{d}$ is the mean of the differences between paired data.

μ_d is the hypothesized population mean of the differences.

s_d is the standard deviation of the differences between paired data.

n is the number of data pairs.

Note: For the t-distribution, the degrees of freedom will be $n - 1$, where n is the number of data pairs.

The setup of the null and alternative hypotheses will follow one of the setups shown in Table 4.12:

Setup A	Setup B	Setup C
$H_0 : \mu_d = k$ $H_a : \mu_d \neq k$	$H_0 : \mu_d \leq k$ $H_a : \mu_d > k$	$H_0 : \mu_d \geq k$ $H_a : \mu_d < k$

Table 4.12 Possible Setups for Null and Alternative Hypothesis for Paired Data Hypothesis Testing

Example 4.20 illustrates this hypothesis testing process.

EXAMPLE 4.20

Problem

A pharmaceutical company is testing a cholesterol-lowering medication in a clinical trial. The company claims that the medication helps to lower cholesterol levels. To test the claim, a group of 10 patients is selected for a clinical trial. Before and after cholesterol measurements are taken on the same patients, where the "after" measurement is recorded after the patient has taken the medication for a six-month time period as shown in Table 4.13.

Patient	Cholesterol Level Before Taking the Medication	Cholesterol Level After Taking the Medication	Difference (Before Reading - After Reading)
1	218	210	8
2	232	241	-9
3	259	223	36
4	265	244	21
5	248	227	21
6	298	273	25
7	263	252	11
8	281	276	5
9	290	281	9
10	271	259	12
			$\bar{d} = 13.9$ $s_d = 12.414$

Table 4.13 Sample Data for Paired Data Example

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the company claims that the medication helps to lower cholesterol levels. If the medication did reduce cholesterol levels, then we would expect the after data to be less than the before data. In this case, the difference for "before - after" would be a positive value, which is greater than zero. A greater than symbol must be used in the alternative hypothesis. The complement of a "greater than" symbol is a "less than or equals" symbol, which will be used in the null hypothesis. This setup will correspond to Setup C from Table 4.12. The claim corresponds to the alternative hypothesis. Thus the setup

of the null and alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : \mu_d &\leq 0 \\ H_a : \mu_d &> 0 \text{ [claim]} \end{aligned}$$

The mean of the differences is average of the differences column shown in Table 4.13. The mean of the differences column is 13.9, and the standard deviation of the differences column is 12.414.

Next, calculate the standardized test statistic as follows:

$$t = \frac{\bar{d} - \mu_d}{s_d/\sqrt{n}} = \frac{13.9 - 0}{12.414/\sqrt{10}} = 3.541$$

Notice that since the alternative hypothesis contains a "greater than" symbol, the hypothesis test is called a "right tailed" test and the *p*-value will be calculated as the area under the t-distribution curve to the *right* of the test statistic. The area under the t-distribution curve to the right of a t-score of 3.541 (using degrees of freedom $df = 9$) is approximately 0.003. Thus the *p*-value for this hypothesis test is 0.003.

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.003 and the level of significance is 0.05: since the *p*-value < level of significance, then the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis; the claim corresponds to the alternative hypothesis. This scenario corresponds to Row 2 in Table 4.8, and "there is enough evidence to support the claim that the medication helps to lower cholesterol levels."

Testing Claims for Two Proportions

Data scientists and researchers are often interested in comparing two proportions. For example, a medical researcher might be interested in comparing the percentage of people contracting a respiratory virus for those taking a vaccine versus not taking a vaccine.

In this section, we discuss two-sample hypothesis testing for proportions, where the notation p_1 refers to the population proportion for the first group and p_2 refers to the population proportion for the second group.

A researcher will take samples from the two populations and then calculate two sample proportions ($\hat{p}$) as follows:

$$\begin{aligned} \hat{p}_1 &= \frac{x_1}{n_1} \\ \hat{p}_2 &= \frac{x_2}{n_2} \end{aligned}$$

Where:
x_1 is the number of successes in the first sample.
n_1 is the sample size for the first sample.

x_2 is the number of successes in the second sample.
n_2 is the sample size for the second sample.

$\hat{p}_1$ is the sample proportion for the first sample.
$\hat{p}_2$ is the sample proportion for the second sample.

It is also useful to calculate a weighted estimate ($\bar{p}$) for $\hat{p}_1$ and $\hat{p}_2$, as follows:

$$\bar{p} = \frac{x_1 + x_2}{n_1 + n_2}$$

Here are the requirements to use this procedure:

a. The samples are random and independent.
b. Sample sizes are large enough to use a normal distribution as an approximation for a binomial distribution. To confirm this, ensure that the following four quantities are all at least 5: $n_1\overline{p}$, $n_1\left(1-\overline{p}\right)$, $n_2\overline{p}$, $n_2\left(1-\overline{p}\right)$

When these requirements are met, the normal distribution is used as the basis for conducting the hypothesis test.

For this procedure, the test statistic will be a *z*-score, and the standardized test statistic is calculated as follows:

$$z = \frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\overline{p}\left(1-\overline{p}\right)\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}}$$

Here is a step-by-step example to illustrate this hypothesis testing process.

EXAMPLE 4.21

Problem

A drug is under investigation to address lung cancer. As part of a clinical trial, 250 people took the drug and 230 people took a placebo (a substance with no therapeutic effect). From the results of the study, 211 people taking the drug were cancer free, and 172 taking the placebo were cancer free. At a level of significance of 0.01, can you support the claim by the pharmaceutical company that the proportion of subjects who are cancer free after taking the actual cancer drug is greater than the proportion for those who took the placebo?

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypotheses based on the claim. Notice that the claim mentions “proportion of subjects who are cancer free after taking the actual cancer drug is greater than the proportion for those who took the placebo,” where “greater than” will translate to a greater than symbol. A greater than symbol must be used in the alternative hypothesis. The complement of a “greater than” symbol is a “less than or equals” symbol, which will be used in the null hypothesis. Let p_1 represent the proportion of people who take the medication and are cancer free, and let p_2 represent the proportion of people who take the placebo and are cancer free. The null hypothesis assumes there is no difference in the two proportions such that p_1 is equal to p_2 (which can also be considered as $p_1 - p_2 = 0$). This setup will correspond to Setup E from Table 4.10. The claim corresponds to the alternative hypothesis. Thus the setup of the null and alternative hypotheses will be as follows:

$$\begin{aligned} H_0 : p_1 &\leq p_2 \\ H_a : p_1 &> p_2 \quad \text{[claim]} \end{aligned}$$

First, check that the requirements for this hypothesis test are met.

Calculate the weighted estimate ($\overline{p}$) for p_1 and p_2, as follows:

$$\overline{p} = \frac{x_1 + x_2}{n_1 + n_2} = \frac{211 + 172}{250 + 230} = \frac{383}{480} = 0.798$$

Check that the following four quantities are all at least 5: $n_1\overline{p}$, $n_1\left(1-\overline{p}\right)$, $n_2\overline{p}$,$n_2\left(1-\overline{p}\right)$.

$$\begin{aligned} n_1\bar{p} &= (250)(0.798) = 199.5 \\ n_1\left(1-\bar{p}\right) &= (250)(1-0.798) = 50.5 \\ n_2\bar{p} &= (230)(0.798) = 183.54 \\ n_2\left(1-\bar{p}\right) &= (230)(1-0.798) = 46.46 \end{aligned}$$

Since these results are at least 5, the requirements are met.

Calculate the sample proportions as follows:

$$\begin{aligned} \hat{p}_1 &= \frac{x_1}{n_1} = \frac{211}{250} = 0.844 \\ \hat{p}_2 &= \frac{x_2}{n_2} = \frac{172}{230} = 0.748 \end{aligned}$$

Next, calculate the standardized test statistic as follows:

$$z = \frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\bar{p}\left(1-\bar{p}\right)\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}} = \frac{(0.844-0.748)-0}{\sqrt{\left(0.798\right)\left(1-0.798\right)\left(\frac{1}{250}+\frac{1}{230}\right)}} = 2.621$$

Notice that since the alternative hypothesis contains a "greater than" symbol, the hypothesis test is called a "right tailed" test and the *p*-value will be calculated as the area under the normal distribution curve to the right of the test statistic. The area under the normal distribution curve to the right of a *z*-score of 2.621 is approximately 0.004. Thus the *p*-value for this hypothesis test is 0.004.

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.004 and the level of significance is 0.01: since the *p*-value $<$ level of significance, then the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis; the claim corresponds to the alternative hypothesis. This scenario corresponds to row #2 in Table 4.8, and "there is enough evidence to support the claim" that the proportion of subjects who are cancer free after taking the actual cancer drug is greater than the proportion for those who took the placebo. This indicates that the medication is effective in addressing lung cancer.

4.3 Correlation and Linear Regression Analysis

Learning Outcomes

By the end of this section, you should be able to:

- 4.3.1 Create scatterplots and calculate and interpret correlation coefficients.
- 4.3.2 Perform linear regression and determine the best-fit linear equation.
- 4.3.3 Use Python to calculate correlation coefficients and determine equations of linear regression models.

We briefly introduced correlation analysis at the beginning of this chapter, but now we want to dig in a little deeper. Data scientists are often interested in knowing if there are relationships, or a *correlation*, between two numeric quantities. For example, a medical researcher might be interested in investigating if there is a relationship between age and blood pressure. A college professor might be interested to know if there is a relationship between time spent on social media by students and corresponding academic grades.

Correlation analysis allows for the determination of a statistical relationship between two numeric quantities, or variables—an independent variable and a dependent variable. The **independent variable** is the variable that you can change or control, while the **dependent variable** is what you measure or observe to see how it

responds to those changes. Regression analysis takes this a step further by quantifying the relationship between the two variables, and it can be used to predict one quantity based on a second quantity, assuming there is a significant correlation between the two quantities. For example, the value of a car can be predicted based on the age of the car.

Earlier we noted many business-related applications where correlation and regression analysis are used. For instance, regression analysis can be used to establish a mathematical equation that relates a dependent variable (such as sales) to an independent variable (such as advertising expenditure). But it has many applications apart from business as well. An automotive engineer is interested in the correlation between outside temperature and battery life for an electric vehicle, for instance.

Our discussion here will focus on linear regression—analyzing the relationship between one dependent variable and one independent variable, where the relationship can be modeled using a linear equation.

Scatterplots and Correlation

In correlation analysis, we study the relationship between **bivariate data**, which is data collected on two variables where the data values are paired with one another. Correlation measures the association between two numeric variables. We may be interested in knowing if there is a correlation between bond prices and interest rates or between the age of a car and the value of the car. To investigate the correlation between two numeric quantities, the first step is to collect (x, y) data for the two numeric quantities of interest and then create a scatterplot that will graph the (x, y) ordered pairs. The independent, or explanatory, quantity is labeled the x-variable, and the dependent, or response, quantity is labeled the y-variable.

For example, let's say that a financial analyst wants to know if the price of Nike stock is correlated with the value of the S&P 500 (Standard & Poor's 500 stock market index). To investigate this, monthly data can be collected for Nike stock prices and value of the S&P 500 for a period of time, and a scatterplot can be created and examined. A **scatterplot, or scatter diagram**, is a graphical display intended to show the relationship between two variables. The setup of the scatterplot is that one variable is plotted on the horizontal axis and the other variable is plotted on the vertical axis. Each pair of data values is considered as an (x, y) point, and the various points are plotted on the diagram. A visual inspection of the plot is then made to detect any patterns or trends on the scatter diagram. Table 4.14 shows the relationship between the Nike stock price and its S&P value on a monthly basis over a one-year time period.

Date	S&P 500 (x)	Nike Stock Price (y)
Month 1	2912.43	87.18
Month 2	3044.31	98.58
Month 3	3100.29	98.05
Month 4	3271.12	97.61
Month 5	3500.31	111.89
Month 6	3363.00	125.54
Month 7	3269.96	120.08
Month 8	3621.63	134.70
Month 9	3756.07	141.47
Month 10	3714.24	133.59

Table 4.14 Nike Stock Price ($) and Value of S&P 500 over a One-Year Time Period
(source: Yahoo! Finance)

Date	S&P 500 (x)	Nike Stock Price (y)
Month 11	3811.15	134.78
Month 12	3943.34	140.45

Table 4.14 Nike Stock Price ($) and Value of S&P 500 over a One-Year Time Period
(source: Yahoo! Finance)

To assess **linear correlation**, examine the graphical trend of the data points on the scatterplot to determine if a straight-line pattern exists (see Figure 4.5). If a linear pattern exists, the correlation may indicate either a positive or a negative correlation. A positive correlation indicates that as the independent variable increases, the dependent variable tends to increase as well, or as the independent variable decreases, the dependent variable tends to decrease (the two quantities move in the same direction). A negative correlation indicates that as the independent variable increases, the dependent variable decreases, or as the independent variable decreases, the dependent variable increases (the two quantities move in opposite directions). If there is no relationship or association between the two quantities, where one quantity changing does not affect the other quantity, we conclude that there is no correlation between the two variables.

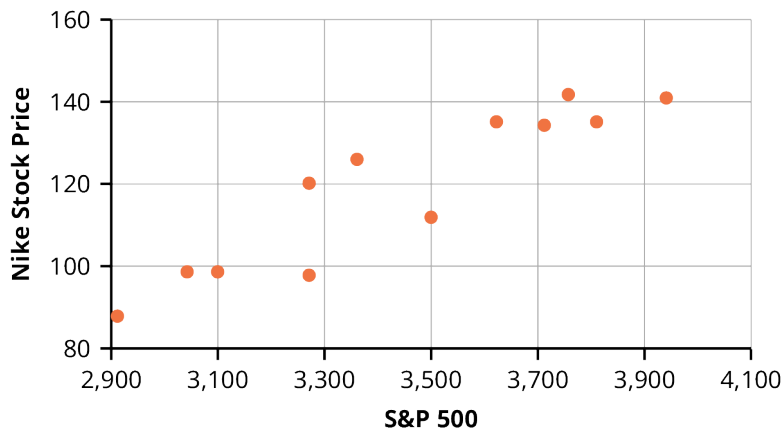


Figure 4.5 Scatterplot of Nike Stock Price ($) and Value of S&P 500
(source: Yahoo! Finance)

From the scatterplot in the Nike stock versus S&P 500 example, we note that the trend reflects a positive correlation in that as the value of the S&P 500 increases, the price of Nike stock tends to increase as well.

When inspecting a scatterplot, it may be difficult to assess a correlation based on a visual inspection of the graph alone. A more precise assessment of the correlation between the two quantities can be obtained by calculating the numeric correlation coefficient (referred to using the symbol r).

The **correlation coefficient** is a measure of the strength and direction of the correlation between the independent variable x and the dependent variable y.

The formula for r is shown; however, software is typically used to calculate the correlation coefficient.

$$r = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{n\Sigma x^2 - (\Sigma x)^2}\sqrt{n\Sigma y^2 - (\Sigma y)^2}}$$

where n refers to the number of data pairs and the symbol Σx indicates to sum the x-values.

Note that this value of "r" is sometimes referred to as the "Pearson correlation coefficient."

Table 4.15 provides a step-by-step procedure on how to calculate the correlation coefficient r.

Step	Representation in Symbols
1. Calculate the sum of the *x*-values.	Σx
2. Calculate the sum of the *y*-values.	Σy
3. Multiply each *x*-value by the corresponding *y*-value and calculate the sum of these xy products.	Σxy
4. Square each *x*-value and then calculate the sum of these squared values.	Σx^2
5. Square each *y*-value and then calculate the sum of these squared values.	Σy^2
6. Determine the value of n, which is the number of data pairs.	n
7. Use these results to then substitute into the formula for the correlation coefficient.	$r = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{n\Sigma x^2 - (\Sigma x)^2}\sqrt{n\Sigma y^2 - (\Sigma y)^2}}$

Table 4.15 Steps for Calculating the Correlation Coefficient

Note that since *r* is calculated using sample data, *r* is considered a *sample statistic* and is used to measure the strength of the correlation for the two population variables. Recall that sample data indicates data based on a subset of the entire population.

As mentioned, given the complexity of this calculation, software is typically used to calculate the correlation coefficient. There are several options for calculating the correlation coefficient in Python. The example shown uses the `scipy.stats` library in Python which includes the function `pearsonr()` for calculating the Pearson correlation coefficient (*r*).

We will create two numerical arrays in Python using the `np.array()` function and then use the `pearsonr()` function to calculate the Pearson correlation coefficient for the (x, y) data shown in Table 4.15.

The following Python code provides the correlation coefficient for the Nike Stock Price dataset as 0.923.

PYTHON CODE

```
# import libraries
import numpy as np
from scipy.stats  import pearsonr

# establish x and y data, x-data is S&P500 value, y-data is Nike Stock Price

SP500 = np.array([2912.43, 3044.31, 3100.29, 3271.12, 3500.31, 3363.00, 3269.96,
3621.63, 3756.07, 3714.24, 3811.15, 3943.34])

Nike = np.array([87.18, 98.58, 98.05, 97.61, 111.89, 125.54, 120.08, 134.70,
141.47, 133.59, 134.78, 140.45])

#pearson r returns both value of r and corresponding p-value
```

```
r, p = pearsonr(SP500, Nike)

#print value of r , rounded to 3 decimal places

print("Correlation coefficient: ", round(r, 3))
```

The resulting output will look like this:

```
Correlation coefficient: 0.923
```

Interpret a Correlation Coefficient

Once the value of *r* is calculated, this measurement provides two indicators for the correlation:

a. the strength of the correlation based on the *value* of *r*
b. the direction of the correlation based on the *sign* of *r*

The *value* of *r* gives us this information:

- The value of *r* is always between -1 and $+1$: $-1 \leq r \leq 1$.
- The size of the correlation *r* indicates the strength of the linear relationship between the two variables. Values of *r* close to -1 or to $+1$ indicate a stronger linear relationship.
- If $r = 0$, there is no linear relationship between the two variables (*no linear correlation*).
- If $r = 1$, there is perfect positive correlation. If $r = -1$, there is perfect negative correlation. In both of these cases, all the original data points lie on a straight line.

The sign of *r* gives us this information:

- A positive value of *r* means that when x increases, y tends to increase, and when x decreases, y tends to decrease (*positive correlation*).
- A negative value of *r* means that when x increases, y tends to decrease, and when x decreases, y tends to increase (*negative correlation*).

From the statistical results shown in the Python output, the correlation coefficient *r* is 0.923, which indicates that the relationship between Nike stock and the value of the S&P 500 over this time period represents a strong, positive correlation.

Test a Correlation Coefficient for Significance

The correlation coefficient, *r*, tells us about the strength and direction of the linear relationship between x and y. The sample data are used to compute *r*, the correlation coefficient for the sample. If we had data for the entire population (that is, all measurements of interest), we could find the population correlation coefficient, which is labeled as the Greek letter ρ (pronounced "rho"). But because we have only sample data, we cannot calculate the population correlation coefficient. The sample correlation coefficient, *r*, is our estimate of the unknown population correlation coefficient.

- $\rho = \text{population correlation coefficient (unknown)}$
- $r = \text{sample correlation coefficient (known; calculated from sample data)}$

An important step in the correlation analysis is to determine if the correlation is significant. By this, we are asking if the correlation is strong enough to allow meaningful **predictions** for y based on values of x. One method to test the significance of the correlation is to employ a hypothesis test. The hypothesis test lets us decide whether the value of the population correlation coefficient ρ is close to zero or significantly different from zero. We decide this based on the sample correlation coefficient *r* and the sample size n.

If the test concludes that the correlation coefficient is significantly different from zero, we say that the correlation coefficient is significant.

- Conclusion: There is sufficient evidence to conclude that there is a significant linear relationship between the two variables because the correlation coefficient is significantly different from zero.
- What the conclusion means: There is a significant linear relationship between the two variables. If the test concludes that the correlation coefficient is not significantly different from zero (it is close to zero), we say that correlation coefficient is not significant.

A hypothesis test can be performed to test if the correlation is significant. A hypothesis test is a statistical method that uses sample data to test a claim regarding the value of a population parameter. In this case, the hypothesis test will be used to test the claim that the population correlation coefficient ρ is equal to zero.

Use these hypotheses when performing the hypothesis test:

- Null hypothesis: $H_0 : \rho = 0$
- Alternate hypothesis: $H_a : \rho \neq 0$

The hypotheses can be stated in words as follows:

- Null hypothesis H_0: The population correlation coefficient *is not* significantly different from zero. There *is not* a significant linear relationship (correlation) between x and y in the population.
- Alternate hypothesis H_a: The population correlation coefficient is significantly different from zero. There *is* a significant linear relationship (correlation) between x and y in the population.

A quick shorthand way to test correlations is the relationship between the sample size and the correlation. If $|r| \geq \frac{2}{\sqrt{n}}$, then this implies that the correlation between the two variables demonstrates that a linear relationship exists and is statistically significant at approximately the 0.05 level of significance. As the formula indicates, there is an inverse relationship between the sample size and the required correlation for significance of a linear relationship. With only 10 observations, the required correlation for significance is 0.6325; for 30 observations, the required correlation for significance decreases to 0.3651; and at 100 observations, the required level is only 0.2000.

INTERPRETING THE SIGNIFICANCE OF THE SAMPLE CORRELATION COEFFICIENT

- If *r* is significant and the scatterplot shows a linear trend, the line can be used to predict the value of y for values of x that are within the domain of observed *x*-values.
- If *r* is not significant OR if the scatterplot does not show a linear trend, the line should *not* be used for prediction.
- If *r* is significant and the scatterplot shows a linear trend, the line may *not* be appropriate or reliable for prediction *outside* the domain of observed *x*-values in the data.

EXAMPLE 4.22

Problem

Suppose that the chief financial officer (CFO) of a corporation is investigating the correlation between stock prices and unemployment rate over a period of 10 years and finds the correlation coefficient to be -0.68. There are 10 (x, y) data points in the dataset. Should the CFO conclude that the correlation is significant for the relationship between stock prices and unemployment rate based on a level of significance of 0.05?

Solution

When using a level of significance of 0.05, if $|r| \geq \frac{2}{\sqrt{n}}$, then this implies that the correlation between the two variables demonstrates a linear relationship that is statistically significant at approximately the 0.05 level of significance. In this example, we check if $|-0.68|$ is greater than or equal to $\frac{2}{\sqrt{n}}$ where $n = 10$. Since $\frac{2}{\sqrt{10}}$ is approximately 0.632, this indicates that the *r*-value of -0.68 is greater than $\frac{2}{\sqrt{n}}$, and thus the correlation is deemed significant.

Linear Regression

A regression model is typically developed once the correlation coefficient has been calculated and a determination has been made that the correlation is significant. The regression model can then be used to predict one quantity (the dependent variable) based on the other (the independent variable). For example, a business may want to establish a correlation between the amount the company spent on advertising (the independent variable) versus its recorded sales (the dependent variable). If a strong enough correlation is established, then the business manager can predict sales based on the amount spent on advertising for a given period. In this discussion we will focus on linear regression, where a straight line is used to model the relationship between the two variables. Once a straight-line model is developed, this model can then be used to predict the value of the dependent variable for a specific value of the independent variable.

Method of Least Squares and Residuals

To create a linear model that best fits the (x, y) points on a scatterplot, researchers generally use the **least squares method** based on a mathematical algorithm designed to minimize the squared distances from the (x, y) data points to the fitted line.

Recall from algebra that the equation of a straight line is given by:

$$y = mx + b$$

where m is the slope of the line and b is the *y*-intercept of the line.

The slope measures the steepness of the line, and the *y*-intercept is that point on the y-axis where the graph crosses, or intercepts, the y-axis.

In linear regression analysis, the equation of the straight line is written in a slightly different way using the model

$$\hat{y} = a + bx$$

In this format, b is the slope of the line, and a is the *y*-intercept of the line. The notation $\hat{y}$ is called y-hat and is used to indicate a predicted value of the dependent variable y for a certain value of the independent variable x.

If a line extends uphill from left to right, the slope is a positive value (see Figure 4.6; if the line extends downhill from left to right, the slope is a negative value).

(a) (b) (c)

Figure 4.6 Three examples of graphs of $\hat{y} = a + bx$. (a) If $b > 0$, the line slopes upward to the right. (b) If $b = 0$, the line is horizontal. (c) If $b < 0$, the line slopes downward to the right.

When generating the equation of a line in algebra using $y = mx + b$, two (x, y) points were required to generate the equation. However, in regression analysis, all the (x, y) points in the dataset will be utilized to develop the linear regression model.

The first step in any regression analysis is to create the scatterplot. Then proceed to calculate the correlation coefficient *r* and check this value for significance. If we think that the points show a linear relationship, we would like to draw a line on the scatterplot. This line can be calculated through a process called linear regression. However, we only calculate a regression line if one of the variables helps to explain or predict the other variable. If x is the independent variable and y the dependent variable, then we can use a regression line to predict y for a given value of x.

As an example of a regression equation, assume that a correlation exists between the monthly amount spent on advertising and the monthly revenue for a Fortune 500 company. After collecting (x, y) data for a certain time period, the company determines the regression equation is of the form

$$\hat{y} = 9376.7 + 61.8x$$

where x represents the monthly amount spent on advertising (in thousands of dollars), $\hat{y}$ represents the monthly revenues for the company (in thousands of dollars), the slope is 61.8, and the *y*-intercept is 9376.7.

A scatterplot of the (x, y) data is shown in Figure 4.7.

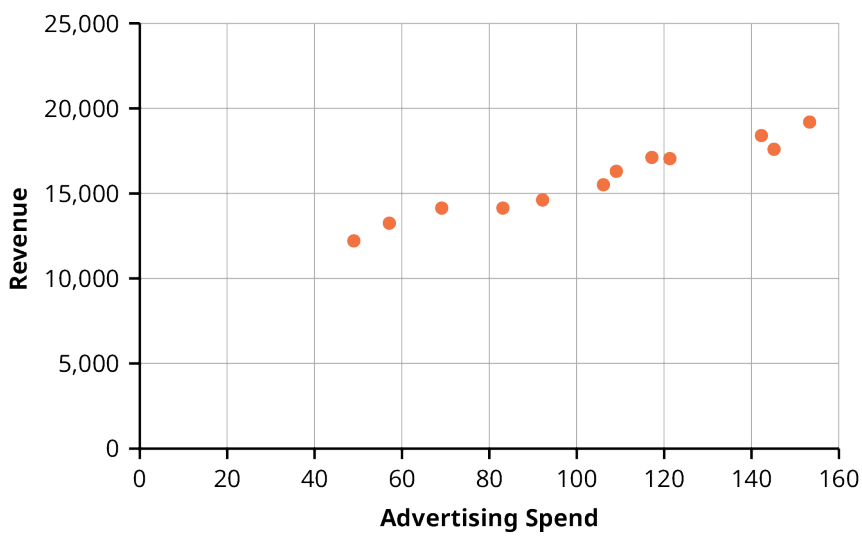


Figure 4.7 Scatterplot of Revenue versus Advertising for a Fortune 500 Company ($000s)

The company would like to predict the monthly revenue if its executives decide to spend $150,000 on advertising next month. To determine the estimate of monthly revenue, let $x = 150$ in the regression equation and calculate a corresponding value for $\hat{y}$:

$$\begin{aligned}\hat{y} &= 9376.7 + 61.8x \\ \hat{y} &= 9376.7 + 61.8(150) \\ \hat{y} &= 18646.7\end{aligned}$$

This predicted value of y indicates that the anticipated revenue would be $18,646,700, given the advertising spend of $150,000.

Notice that from past data, there may have been a month where the company actually did spend $150,000 on advertising, and thus the company may have an actual result for the monthly revenue. This actual, or observed, amount can be compared to the prediction from the linear regression model to calculate what is called a residual.

A **residual** is the difference between an observed *y*-value and the predicted *y*-value obtained from the linear regression equation. As an example, assume that in a previous month, the actual monthly revenue for an advertising spend of $150,000 was $19,200,000, and thus $y = 19{,}200$. The residual for this data point can be calculated as follows:

$$\begin{aligned}\text{Residual} &= (\text{observed } y\text{-value}) - (\text{predicted } y\text{-value}) \\ \text{Residual} &= y - \hat{y} \\ \text{Residual} &= 19200 - 18646.7 = 553.3\end{aligned}$$

Notice that residuals can be positive, negative, or zero. If the observed *y*-value exactly matches the predicted *y*-value, then the residual will be zero. If the observed *y*-value is greater than the predicted *y*-value, then the residual will be a positive value. If the observed *y*-value is less than the predicted *y*-value, then the residual will be a negative value.

When formulating the best-fit linear regression line to the points on the scatterplot, the mathematical analysis generates a linear equation where the sum of the squared residuals is minimized. This analysis is referred to as the **method of least squares**. The result is that the analysis generates a linear equation that is the "best fit" to the points on the scatterplot, in the sense that the line minimizes the differences between the predicted values and observed values for y; this is generally referred to as the **best-fit linear equation**.

EXAMPLE 4.23

Problem

Suppose that the chief financial officer of a corporation has created a linear model for the relationship between the company stock and interest rates. When interest rates are at 5%, the company stock has a value of $94. Using the linear model, when interest rates are at 5%, the model predicts the value of the company stock to be $99. Calculate the residual for this data point.

Solution

A residual is the difference between an observed *y*-value and the predicted *y*-value obtained from the linear regression equation:

$$\begin{aligned}\text{Residual} &= (\text{observed } y\text{-value}) - (\text{predicted } y\text{-value}) \\ \text{Residual} &= y - \hat{y} \\ \text{Residual} &= 94 - 99 = -5\end{aligned}$$

The goal in regression analysis is to determine the coefficients a and b in the following regression equation:

$$\hat{y} = a + bx$$

Once the (x, y) has been collected, the slope (b) and *y*-intercept (a) can be calculated using the following

formulas:

$$b = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$
$$a = \frac{\Sigma y}{n} - b\frac{\Sigma x}{n}$$

where n refers to the number of data pairs and Σx indicates sum of the *x*-values.

Notice that the formula for the *y*-intercept requires the use of the slope result (b), and thus the slope should be calculated first, and the *y*-intercept should be calculated second.

When making predictions for y, it is always important to plot a scatter diagram first. If the scatterplot indicates that there is a linear relationship between the variables, then it is reasonable to use a best-fit line to make predictions for y, given x is within the domain of *x*-values in the sample data, *but not necessarily for x-values outside that domain*.

USING TECHNOLOGY FOR LINEAR REGRESSION

Typically, technology is used to calculate the best-fit linear model as well as calculate correlation coefficients and scatterplot. Details of using Python for these calculations are provided in Using Python for Correlation and Linear Regression.

Assumptions for Linear Regression

Testing the significance of the correlation coefficient requires that certain assumptions about the data are satisfied. The premise of this test is that the data are a sample of observed points taken from a larger population. We have not examined the entire population because it is not possible or feasible to do so. We are examining the sample to draw a conclusion about whether the linear relationship that we see between x and y data in the sample data provides strong enough evidence that we can conclude that there actually is a linear relationship between x and y data in the population.

The regression line equation that we calculate from the sample data gives the best-fit line for our particular sample. We want to use this best-fit line for the sample as an estimate of the best-fit line for the population (Figure 4.8). Examining the scatterplot and testing the significance of the correlation coefficient helps us determine if it is appropriate to do this.

These are the assumptions underlying the test of significance:

1. There is a linear relationship in the population that models the average value of y for varying values of x. In other words, the expected value of y for each particular value lies on a straight line in the population. (We do not know the equation for the line for the population. Our regression line from the sample is our best estimate of this line in the population.)
2. The *y*-values for any particular *x*-value are normally distributed about the line. This implies that there are more *y*-values scattered closer to the line than are scattered farther away. Assumption (1) implies that these normal distributions are centered on the line: the means of these normal distributions of *y*-values lie on the line.
3. The standard deviations of the population *y*-values about the line are equal for each value of x. In other words, each of these normal distributions of *y*-values has the same shape and spread about the line.
4. The residual errors are mutually independent (no pattern).
5. The data are produced from a well-designed random sample or randomized experiment.

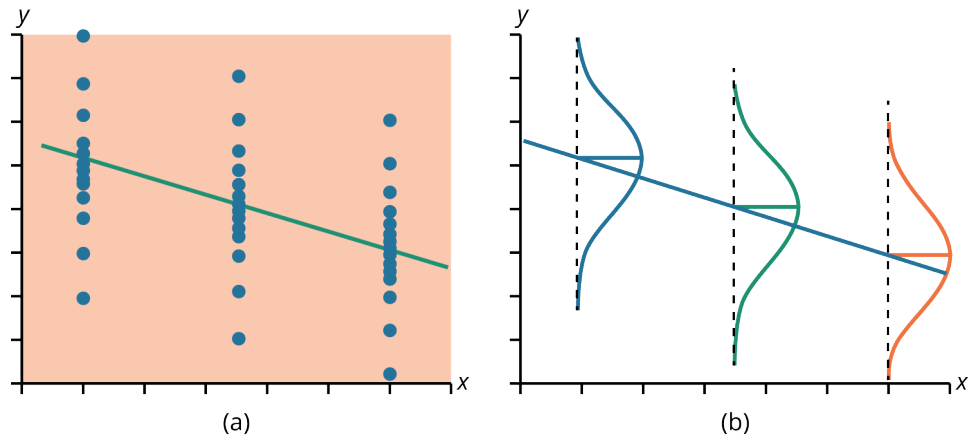


Figure 4.8 Best-Fit Line. Note that the *y*-values for each *x*-value are normally distributed about the line with the same standard deviation. For each *x*-value, the mean of the *y*-values lies on the regression line. More *y*-values lie near the line than are scattered farther away from the line

Using Python for Correlation and Linear Regression

Once a correlation has been deemed significant, a linear regression model is developed. The goal in the regression analysis is to determine the coefficients a and b in the following regression equation:

$$\hat{y} = a + bx$$

These formulas can be quite cumbersome, especially for a significant number of data pairs, and thus software is often used (such as Excel, Python, or R).

Develop a Linear Regression Model

In the following example, Python will be used for the following analysis based on a given dataset of (x, y) data.

- Create a scatterplot.
- Calculate the correlation coefficient.
- Construct the best-fit linear equation.
- Predict values of the dependent variable for given values of the independent variable.

There are various ways to accomplish these tasks in Python, but we will use several functions available within the `scipy` library, such as:

`Pearsonr()` for correlation coefficient

`linregress()` for linear regression model

`plt.scatter()` for scatterplot generation

EXAMPLE 4.24

Problem

A marketing manager is interested in studying the correlation between the amount spent on advertising and revenue at a certain company. Twelve months of data were collected as shown in Table 4.16. (Dollar amounts are in thousands of dollars.)

Use Python to perform the following analysis:

a. Generate a scatterplot.
b. Calculate the correlation coefficient r.
c. Determine if the correlation is significant (use level of significance of 0.05).

d. Construct a linear regression model.
e. Use the model to predict the revenue for a certain month where the amount spent on advertising is $100.

Month	Advertising Spend	Revenue
Jan	49	12210
Feb	145	17590
Mar	57	13215
Apr	153	19200
May	92	14600
Jun	83	14100
Jul	117	17100
Aug	142	18400
Sep	69	14100
Oct	106	15500
Nov	109	16300
Dec	121	17020

Table 4.16 Revenue versus Advertising for a Company (in $000s)

Solution

a. The first step in the analysis is to generate a scatterplot. (Recall that scatterplots were also discussed in Scatterplots and Correlation.)
The Python function `plt.scatter()` can be used to generate the scatterplot for the (x, y) data. Note that we consider advertising to be the independent variable (x-data) and revenue to be the dependent variable (y-data) since revenue depends on the amount of advertising.

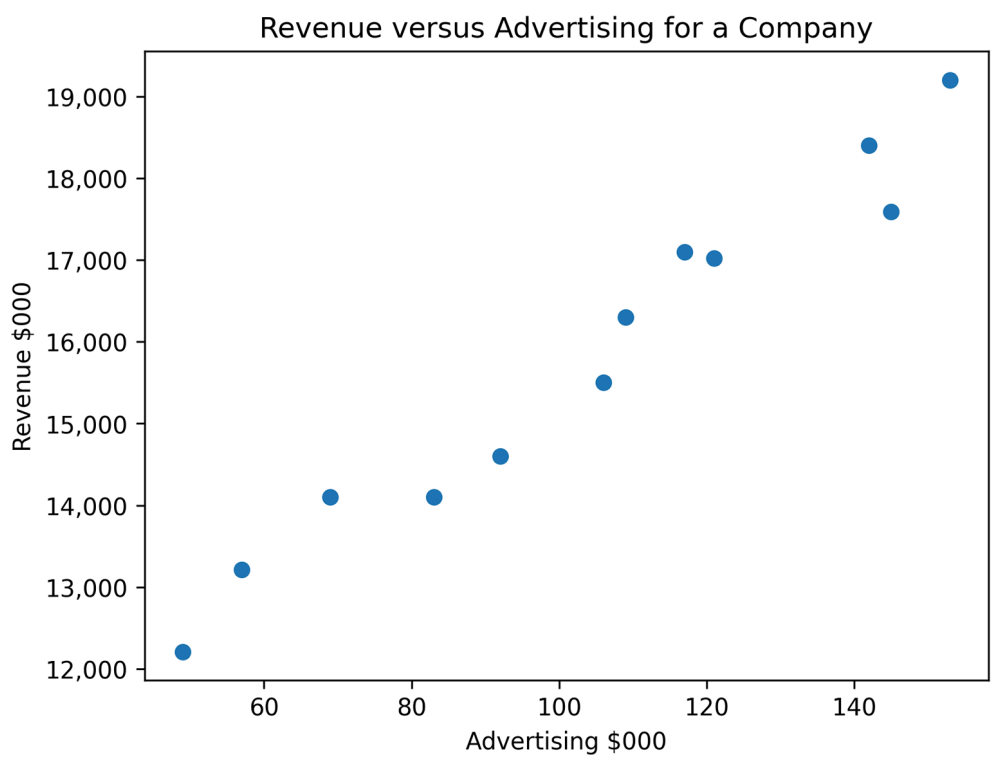
Here is the Python code to generate the scatterplot:

PYTHON CODE

```
# import the matplotlib library
import matplotlib.pyplot as plt

# define the x-data, which is amount spent on advertising
x = [49, 145, 57, 153, 92, 83, 117, 142, 69, 106, 109, 121]

# define the y-data, which is revenue
y = [12210, 17590, 13215, 19200, 14600, 14100, 17100, 18400, 14100, 15500, 16300,
17020]
```

```
# use the scatter function to generate a time series graph
plt.scatter(x, y)

# Add a title using the title function
plt.title("Revenue versus Advertising for a Company")

# Add labels to the x and y axes by using xlabel and ylabel functions
plt.xlabel("Advertising $000")
plt.ylabel ("Revenue $000")
```

The resulting output will look like this:



b. The Python function `pearsonr()` can be used to generate the correlation coefficient, r.

Here is the Python code to calculate the correlation coefficient:

PYTHON CODE

```
# import libraries
import numpy as np
from scipy.stats  import pearsonr
```

```
# define the x-data, which is amount spent on advertising
x = [49, 145, 57, 153, 92, 83, 117, 142, 69, 106, 109, 121]

# define the y-data, which is revenue
y = [12210, 17590, 13215, 19200, 14600, 14100, 17100, 18400, 14100, 15500, 16300,
17020]

#pearson r returns both value of r and corresponding p-value
r, p = pearsonr(x, y)

#print value of r, rounded to 3 decimal places

print("Correlation coefficient: ", round(r, 3))
```

The resulting output will look like this:

```
Correlation coefficient: 0.981
This value of r = 0.981 indicates a strong, positive correlation between
advertising spend and revenue.
```

c. When using a level of significance of 0.05, if $|r| \geq \frac{2}{\sqrt{n}}$, then this implies that the correlation between the two variables demonstrates a linear relationship that is statistically significant at approximately the 0.05 level of significance. In this example, we check if $|0.981|$ is greater than or equal to $\frac{2}{\sqrt{n}}$ where $n = 12$. Since $\frac{2}{\sqrt{12}}$ is approximately 0.577, this indicates that the *r*-value of 0.981 is greater than $\frac{2}{\sqrt{n}}$, and thus the correlation is deemed significant.

d. The linear regression model will be of the form:

$$\hat{y} = a + bx$$

To determine the slope and *y*-intercept of the regression model, the Python function `linregress()` can be used.
This function takes the x and y data as inputs and provides the slope and intercept as well as other regression-related output.

Here is the Python code to generate the slope and *y*-intercept of the best-fit line:

PYTHON CODE

```
# import libraries
import numpy as np
from scipy.stats  import linregress
```

```
# define the x-data, which is amount spent on advertising
x = [49, 145, 57, 153, 92, 83, 117, 142, 69, 106, 109, 121]

# define the y-data, which is revenue
y = [12210, 17590, 13215, 19200, 14600, 14100, 17100, 18400, 14100, 15500, 16300,
17020]

#pearson r returns both value of r and corresponding p-value
slope, intercept, r_value, p_value, std_err  = linregress(x, y)

print ("slope  =", slope)
print("intercept = ", intercept)
```

The resulting output will look like this:

```
slope = 61.79775818816664
intercept = 9376.698881009072
```

Based on this Python equation, the regression equation can be written as

$$\begin{aligned} \hat{y} &= a + bx \\ \hat{y} &= 9376.7 + 61.8x \end{aligned}$$

where x represents the amount spent on advertising (in thousands of dollars) and y represents the amount of revenue (in thousands of dollars).

It is also useful to plot this best-fit line superimposed on the scatterplot to show the strong correlation between advertising spend and revenue, as follows (see Figure 4.9):

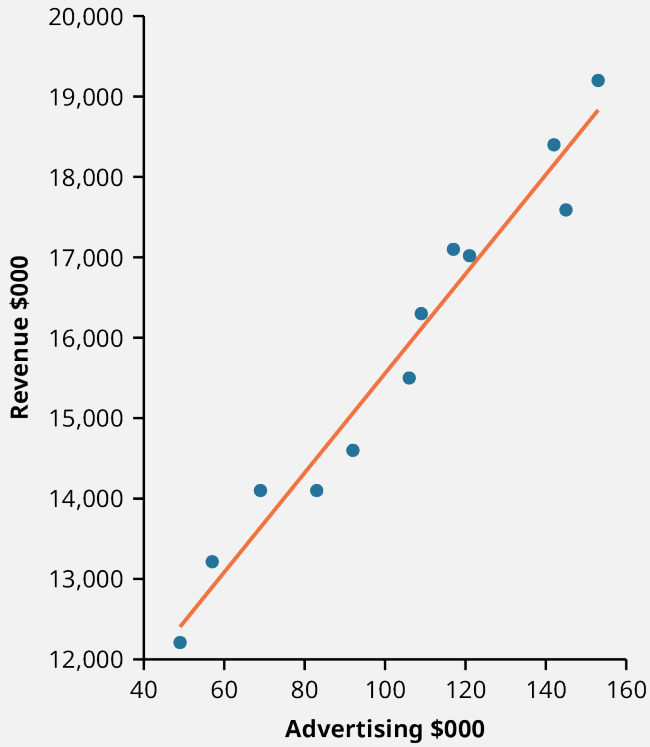


Figure 4.9 Best-Fit Line Superimposed on the Scatterplot

e. The regression equation is: $\hat{y} = 9{,}376.7 + 61.8x$.
The given amount of advertising spend of $100 corresponds to a given *x*-value, so replace "x" with 100 in this regression equation to generate a predicted revenue:

$$\begin{aligned} \hat{y} &= 9376.7 + 61.8x \\ \hat{y} &= 9376.7 + 61.8(100) \\ \hat{y} &= 15556.7 \end{aligned}$$

Since both advertising and revenue are expressed in thousands of dollars, the conclusion is that if advertising spend is $100,000, the predicted revenue for the company will be $15,556,700.

Interpret and Apply the Slope and *y*-Intercept

The slope of the line, b, describes how changes in the variables are related. It is important to interpret the slope of the line in the context of the situation represented by the data. You should be able to write a sentence interpreting the slope in plain English.

Interpretation of the Slope

The slope of the best-fit line tells us how the dependent variable (y) changes for every one-unit increase in the independent (x) variable, on average.

In the previous example, the linear regression model for the monthly amount spent on advertising and the monthly revenue for a company for 12 months was generated as follows:

$$\begin{aligned} \hat{y} &= a + bx \\ \hat{y} &= 9376.7 + 61.8x \end{aligned}$$

Since the slope was determined to be 61.8, the company can interpret this to mean that for every $1,000 spent on advertising, on average, this will result in an increase in revenues of $61,800.

The intercept of the regression equation is the corresponding *y*-value when $x = 0$.

Interpretation of the Intercept

The intercept of the best-fit line tells us the expected mean value of y in the case where the x-variable is equal to zero.

However, in many scenarios it may not make sense to have the x-variable equal zero, and in these cases, the intercept does not have any meaning in the context of the problem. In other examples, the *x*-value of zero is outside the range of the x-data that was collected. In this case, we should not assign any interpretation to the *y*-intercept.

In the previous example, the range of data collected for the x-variable was from $49,000 to $153,000 spent per month on advertising. Since this interval does not include an *x*-value of zero, we would not provide any interpretation for the intercept.

4.4 Analysis of Variance (ANOVA)

Learning Outcomes

By the end of this section, you should be able to:

- 4.4.1 Set up and apply one-way analysis of variance hypothesis test.
- 4.4.2 Use Python to conduct one-way analysis of variance.

In Testing Claims Based on One Sample, we reviewed methods to conduct hypothesis testing for two means; however, there are scenarios where a researcher might want to compare three or more means to determine if any of the means are different from one another. For example, a medical researcher might want to compare three different pain relief medications to compare the average time to provide relief from a migraine headache. **Analysis of variance (ANOVA)** is a statistical method that allows a researcher to compare three or more means and determine if the means are all statistically the same or if at least one mean is different from the others. The **one-way ANOVA** focuses on one independent variable. For two independent variables, a method called "two-way ANOVA" is applicable, but this method is beyond the scope of this text. We explore the one-way ANOVA next.

One-Way ANOVA

To compare three or more means, the sample sizes for each group can be different sample sizes (samples sizes do not need to be identical). For one-way ANOVA hypothesis testing, we will follow the same general outline of steps for hypothesis testing that was discussed in Hypothesis Testing.

The first step will be to write the null and alternative hypotheses. For one-way ANOVA, the null and alternative hypotheses are always stated as the following:

H_0: All population means are equal: $\mu_1 = \mu_2 = \mu_3 = \ldots$

H_a: At least one population mean is different from the others.

Here are the requirements to use this procedure:

a. The samples are random and selected from approximately normal distributions.
b. The samples are independent of one another.
c. The population variances are approximately equal.

In this discussion, we assume that the population variances are not equal.

When these requirements are met, the *F*-distribution is used as the basis for conducting the hypothesis test.

The ***F*-distribution** is a skewed distribution (skewed to the right), and the shape of the distribution depends on two different degrees of freedom, referred to as degrees of freedom for the numerator and degrees of freedom for the denominator.

Figure 4.10 shows the typical shape of the F-distribution:

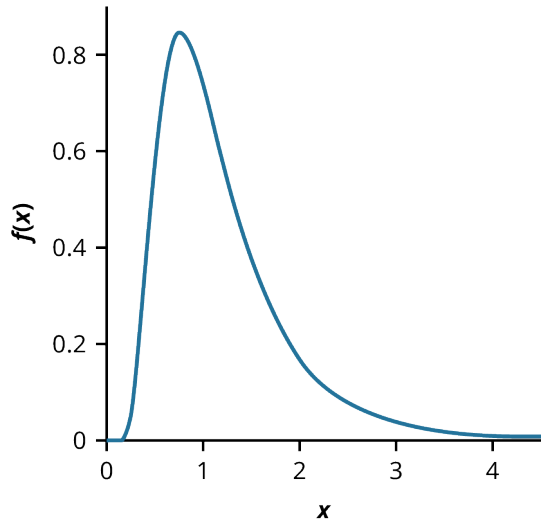


Figure 4.10 Shape of the *F*-Distribution

The test statistic for this hypothesis test will be the ratio of two variances, namely the ratio of the variance between samples to the ratio of the variances within the samples.

$$\text{Test Statistic } = \frac{\text{Variance between samples}}{\text{Variance within samples}}$$

The numerator of the test statistic (variance between samples) is sometimes referred to as variation due to treatment or explained variation.

The denominator of the test statistic (variance within samples) is sometimes referred to as variation due to error, or unexplained variation.

The details for a manual calculation of the test statistic are provided next; however, it is very common to use software for ANOVA analysis.

For the purposes of this discussion, we will assume we are comparing three means, but keep in mind the ANOVA method can be applied to three or more means.

1. Find the mean for each of the three samples. Label these means as $\bar{x}_1, \bar{x}_2, \bar{x}_3$. Find the variance for each of the three samples. Label these variances as s_1^2, s_2^2, s_3^2.
2. Find the grand mean (label this as $\bar{\bar{x}}$). This is the sum of all the data values from the three samples, divided by the overall sample size.
3. Calculate the quantity called "sum of squares between (SSB)" according to the following formula:
$$SSB = n_1\left(\bar{x}_1 - \bar{\bar{x}}\right)^2 + n_2\left(\bar{x}_2 - \bar{\bar{x}}\right)^2 + n_3\left(\bar{x}_3 - \bar{\bar{x}}\right)^2$$
4. Calculate the quantity called "sum of squares within (SSW)" according to the following formula:
$$SSW = \left(n_1 - 1\right)s_1^2 + \left(n_2 - 1\right)s_2^2 + \left(n_3 - 1\right)s_3^2$$
5. Calculate the degrees of freedom for the numerator (df_n) and degrees of freedom for the denominator (df_d).
   For degrees of freedom for the numerator, this is calculated as the number of groups minus 1. For example, if a medical researcher is comparing three different pain relief medications to compare the average time to provide relief from a migraine headache, there are three groups, and thus the degrees of freedom for the numerator would be $3 - 1 = 2$.
   For degrees of freedom for the denominator, this is calculated as the total of the sample sizes minus the number of groups.

6. Calculate the quantity called "mean square between (MSB)" according to the following formula:
$$MSB = \frac{SSB}{df_n}$$
7. Calculate the quantity called "mean square within (MSW)" according to the following formula:
$$MSW = \frac{SSW}{df_d}$$
8. Calculate the test statistic (F), according to the following formula:
$$F = \frac{MSB}{MSW}$$

Once the test statistic is obtained, the *p*-value is calculated as the area to the right of the test statistic under the F-distribution curve (note that the ANOVA hypothesis test is always considered a "right-tail" test).

Usually, the results of these computations are organized in an ANOVA summary table like Table 4.17.

Variation	Sum of Squares	Degrees of Freedom	Mean Squares	F Test Statistic	*p*-value
Between	SSB	df_n	MSB	$\frac{MSB}{MSW}$	Area to the right of the F test statistic
Within	SSW	df_d	MSW		N/A

Table 4.17 ANOVA Summary Table

Using Python for One-Way ANOVA

As mentioned earlier, due to the complexity of these calculations, technology is typically used to calculate the test statistic and *p*-value.

Using Python, the `f_oneway()` function is provided as part of the `scipy` library, and this function provides both the test statistic and *p*-value for a one-way ANOVA hypothesis test.

The syntax is to call the function with the data arrays as arguments, as in:

```
f_oneway(Array1, Array2, Array3, …)
```

The function returns both the F test statistic and the *p*-value for the one-way ANOVA hypothesis test.

The details of this Python function are shown in Example 4.25.

Here is a step-by-step example to illustrate this ANOVA hypothesis testing process.

EXAMPLE 4.25

Problem

An airline manager claims that the average arrival delays for flights from Boston to Dallas are the same for three airlines: Airlines A, B, and C. The following arrival delay data (in minutes) is collected on samples of flights for the three airlines (Table 4.18):

Airline A	Airline B	Airline C
8	13	11
12	0	15
7	2	9
9	7	11
11	6	12
4	5	N/A

Table 4.18 Arrival Delay Data (in Minutes) for Three Airlines

Use this sample data to test the claim that the average arrival delays are the same for three airlines: Airlines A, B, and C. Use a level of significance of 0.05 for this analysis. Assume the samples are independent and taken from approximately normal distributions and each population has approximately the same variance.

Solution

The first step in the hypothesis testing procedure is to write the null and alternative hypothesis based on the claim.

For an ANOVA hypothesis test, the setup of the null and alternative hypotheses is shown here. Notice that the claim mentions "the average arrival delays are the same for three airlines," which corresponds to the null hypothesis.

H_0: All three population means are equal: $\mu_A = \mu_B = \mu_C$ [Claim]

H_a: At least one population mean is different from the others.

The following Python program generates the corresponding F test statistic and *p*-value for this ANOVA analysis:

PYTHON CODE

```python
# import libraries
import numpy as np
from scipy.stats  import f_oneway

# enter airline arrival data

AirlineA = np.array([8, 12, 7, 9, 11, 4])
AirlineB = np.array([13, 0, 2, 7, 6, 5])
AirlineC = np.array([11, 15, 9, 11, 12])

#f_oneway function returns both the test statistic and p-value
f_oneway(AirlineA, AirlineB, AirlineC)
```

The resulting output will look like this:

```
F_onewayResult(statistic=4.388264306955828, pvalue=0.03314999857295516)
```

From this Python output, the test statistic is $F = 4.388$ and the *p*-value is 0.033.

The final step in the hypothesis testing procedure is to come to a final decision regarding the null hypothesis. This is accomplished by comparing the *p*-value with the level of significance. In this example, the *p*-value is 0.033 and the level of significance is 0.05: since the *p*-value $<$ level of significance, the decision is to "reject the null hypothesis."

Conclusion: The decision is to reject the null hypothesis. The claim corresponds to the null hypothesis. This scenario corresponds to the first row in Table 4.18 and thus the conclusion is: There is enough evidence to reject the claim that the average arrival delays are the same for three airlines. This implies that at least one of the means is different from the others.

Note: The ANOVA test method does not specify which mean is different from the others. Further statistical analysis is needed to make this determination.

Key Terms

alternative hypothesis a complementary statement regarding an unknown population parameter used in hypothesis testing

analysis of variance (ANOVA) statistical method to compare three or more means and determine if the means are all statistically the same or if at least one mean is different from the others

best-fit linear equation an equation of the form $\hat{y} = a + bx$ that provides the best-fit straight line to the (x, y) data points

bivariate data data collected on two variables where the data values are paired with one another

bootstrapping a method to construct a confidence interval that is based on repeated sampling and does not rely on any assumptions regarding the underlying distribution

central limit theorem describes the relationship between the sample distribution of sample means and the underlying population

confidence interval an interval where sample data is used to provide an estimate for a population parameter

confidence level the probability that the interval estimate will contain the population parameter, given that the estimation process on the parameter is repeated over and over

correlation a measure of association between two numeric variables

correlation analysis a statistical method used to evaluate and quantify the strength and direction of the linear relationship between two quantitative variables

correlation coefficient a measure of the strength and direction of the linear relationship between two variables

critical value *z*-score that cuts off an area under the normal curve corresponding to a specified confidence level

dependent samples samples from one population that can be paired or matched to the samples taken from the second population

dependent variable in correlation analysis, the variable being studied or measured; the dependent variable is the outcome that is measured or observed to determine the impact of changes in the independent variable

F distribution a skewed probability distribution that arises in statistical hypothesis testing, such as ANOVA analysis

hypothesis testing a statistical method to test claims regarding population parameters using sample data

independent samples the sample from one population that is not related to the sample taken from the second population

independent variable in correlation analysis, the variable that is manipulated or changed in an experiment or study; the value of the independent variable is controlled or chosen by the experimenter to observe its effect on the dependent variable

inferential statistics statistical methods that allow researchers to infer or generalize observations from samples to the larger population from which they were selected

least squares method a method used in linear regression that generates a straight line fit to the data values such that the sum of the squares of the residual is the least sum possible

level of significance (a) the maximum allowed probability of making a Type I error; the level of significance is the probability value used to determine when the sample data indicates significant evidence against the null hypothesis

linear correlation a measure of the association between two variables that exhibit an approximate straight-line fit when plotted on a scatterplot

margin of error an indication of the maximum error of the estimate

matched pairs samples from one population that can be paired or matched to the samples taken from the second population

method of least squares a mathematical method to generate a linear equation that is the “best fit” to the

points on the scatterplot in the sense that the line minimizes the differences between the predicted values and observed values for y

modeling the process of creating a mathematical representation that describes the relationship between different variables in a dataset; the model is then used to understand, explain, and predict the behavior of the data

nonparametric methods statistical methods that do not rely on any assumptions regarding the underlying distribution

null hypothesis statement of no effect or no change in the population

***p*-value** the probability of obtaining a sample statistic with a value as extreme as (or more extreme than) the value determined by the sample data under the assumption that the null hypothesis is true

parametric methods statistical methods that assume a specific form for the underlying distribution

point estimate a sample statistic used to estimate a population parameter

prediction a forecast for the dependent variable based on a specific value of the independent variable generated using the linear model

proportion a measure that expresses the relationship between a part and the whole; a proportion represents the fraction or percentage of a dataset that exhibits a particular characteristic or falls into a specific category

regression analysis a statistical technique used to model the relationship between a dependent variable and one or more independent variables

residual the difference between an observed *y*-value and the predicted *y*-value obtained from the linear regression equation

sample mean a point estimate for the unknown population mean chosen as the most unbiased estimate of the population

sample proportion chosen as the most unbiased estimate of the population, calculated as the number of successes divided by the sample size: $p = \frac{x}{n}$

sample statistic a numerical summary or measure that describes a characteristic of a sample, such as a sample mean or sample proportion

sampling distribution a probability distribution of a sample statistic based on all possible random samples of a certain size from a population or the distribution of a statistic (such as the mean) that would result from taking random samples from the same population repeatedly and calculating the statistic for each sample

scatterplot (or scatter diagram) graphical display that shows values of the independent variable plotted on the x-axis and values of the dependent variable plotted on the y-axis

standard error of the mean the standard deviation of the sample mean, calculated as the population standard deviation divided by the square root of the sample size

standardized test statistic a numerical measure that describes how many standard deviations a particular value is from the mean of a distribution; a standardized test statistic is typically used to assess whether an observed sample statistic is significantly different from what would be expected under a null hypothesis

t-distribution a bell-shaped, symmetric distribution similar to the normal distribution, though the t-distribution has “thicker tails” as compared to the normal distribution

test statistic a numerical value used to assess the strength of evidence against a null hypothesis, calculated from sample data that is used in hypothesis testing

Type I error an error made in hypothesis testing where a researcher rejects the null hypothesis when in fact the null hypothesis is actually true

Type II error an error made in hypothesis testing where a researcher fails to reject the null hypothesis when the null hypothesis is actually false

unbiased estimator a statistic that provides a valid estimate for the corresponding population parameter without overestimating or underestimating the parameter

variable a characteristic or attribute that can be measured or observed.

Group Project

Project A: Confidence Intervals

Conduct a survey of students in your class or at your school to collect data on the number of hours worked per week by one or two predefined sociodemographic characteristics, such as gender, age, geographic neighborhood of residence, first-generation college student, etc. As a group, create 90% confidence intervals for the average number of hours worked for each category of students. Compare the confidence intervals and come to any relevant conclusions for the average number of hours worked by each category of students.

Project B: Hypothesis Testing

As a group, develop a hypothesis for the proportion of white cars in your city. To collect sample data, go to your school parking lot or a nearby parking lot of some kind and determine the sample proportion of white cars. Then, use this data to conduct a hypothesis test for your original hypothesis.

Project C: Correlation and Regression Analysis

For each student, collect data on shoe length and person's height. Create a scatterplot of this (x, y) data and calculate the correlation coefficient. Test the correlation coefficient for significance. If the correlation is significant, develop a linear regression model to predict a person's height based on shoe length.

As a second assignment on the same subject, students should research a certain model car, such as Honda Accord or Toyota Camry. Search websites for used car prices for the selected model and record (x, y) data where x is the age of the car and y is the price of the car. Create a scatterplot of this (x, y) data and calculate the correlation coefficient. Test the correlation coefficient for significance. If the correlation is significant, develop a linear regression model to predict a car's price based on age of the vehicle.

Quantitative Problems

1. An automotive designer wants to create a 90% confidence interval for the mean length of a body panel on a new model electric vehicle. From a sample of 35 cars, the sample mean length is 37 cm and the sample standard deviation is 2.8 cm.
   a. Create a 90% confidence interval for the population mean length of the body panel using Python.
   b. Create a 90% confidence interval for the population mean length of the body panel using Excel.
   c. Create a 95% confidence interval for the population mean length of the body panel using Python.
   d. Create a 95% confidence interval for the population mean length of the body panel using Excel.
   e. Which confidence interval is narrower, the 90% or 95% confidence interval?

2. A political candidate wants to conduct a survey to generate a 95% confidence interval for the proportion of voters planning to vote for this candidate. Use a margin of error of 3% and assume a prior estimate for the proportion of voters planning to vote for this candidate is 58%. What sample size should be used for this survey?

3. A medical researcher wants to test the claim that the mean time for ibuprofen to relieve body pain is 12 minutes. A sample of 25 participants are put in a medical study, and data is collected on the time it takes for ibuprofen to relieve body pain. The sample mean time is 13.2 minutes with a sample standard deviation of 3.7 minutes.
   a. Calculate the *p*-value for this hypothesis test using Python.
   b. Calculate the *p*-value for this hypothesis test using Excel.
   c. Test the researcher's claim at a 5% level of significance.

4. A consumer group is studying the percentage of drivers who wear seat belts. A local police agency states that the percentage of drivers wearing seat belts is at least 70%. To test the claim, the consumer group selects a random sample of 150 drivers and finds that 117 are wearing seat belts.

a. Calculate the *p*-value for this hypothesis test using Python.
b. Calculate the *p*-value for this hypothesis test using Excel.
c. Use a level of significance of 0.10 to test the claim made by the local police agency.

5. A local newspaper is comparing the costs of a meal at two local fast-food restaurants. The newspaper claims that there is no difference in the average price of a meal at the two restaurants. A sample of diners at each of the two restaurants is taken with the following results:

Restaurant A: Sample mean price of a meal $= \$17.55$, sample standard deviation $= \$7.45$, sample size $= 15$
Restaurant B: Sample mean price of a meal $= \$22.75$, sample standard deviation $= \$8.10$, sample size $= 18$

a. Calculate the *p*-value for this hypothesis test using Python.
b. Calculate the *p*-value for this hypothesis test using Excel.
c. Use a level of significance of 0.10 to test the claim made by the newspaper. Assume population variances are not equal.

6. A health official claims the proportion of men smokers is greater than the proportion of women smokers. To test the claim, the following data was collected:

Men: Sample size $= 150$, Number of smokers $= 41$
Women: Sample size $= 140$, Number of smokers $= 31$

a. Calculate the *p*-value for this hypothesis test using Python
b. Calculate the *p*-value for this hypothesis test using Excel.
c. Use a level of significance of 0.05 to test the claim made by the health official.

7. A software company has been tracking revenues versus cash flow in recent years, and the data is shown in the table below. Consider cash flow to be the dependent variable.

Revenues ($000s)	Cash Flow ($000s)
212	81
239	84
218	81
297	99
287	91

Table 4.19 Software Co. Revenue vs. Cash Flow

a. Calculate the correlation coefficient for this data (all dollar amounts are in thousands) using Python.
b. Calculate the correlation coefficient using Excel.
c. Determine if the correlation is significant using a significance level of 0.05.
d. Construct the best-fit linear equation for this dataset.
e. Predict the cash flow when revenue is 250.
f. Calculate the residual for the revenue value of 239.

8. A human resources administrator calculates the correlation coefficient for salary versus years of experience as 0.68. The dataset contains 10 (x, y) data points. Determine if the correlation is significant or not significant at the 0.05 level of significance.

9. A biologist is comparing three treatments of light intensity on plant growth and is interested to know if the average plant growth is different for any of the three treatments.
Sample data is collected on plant growth (in centimeters) for the three treatments, and data is summarized in the table below.
Use an ANOVA hypothesis test method with 0.05 level of significance.

Treatment A	Treatment B	Treatment C
4.3	3.9	2.8
3.7	4.1	2.9
3.1	3.6	3.5
3.4	3.5	3.9
4.1	4.3	4.4
4.0	4.4	4.3
N/A	n/a	4.1

Table 4.20

a. Calculate the *p*-value for this hypothesis test using Python.
b. Determine if the average plant growth is different for any of the three treatments.

5 Time Series and Forecasting

Figure 5.1 Time series analysis involves the examination of data points collected at specific time intervals; it is crucial for identifying trends and patterns that inform predictions and strategic decisions. (credit: modification of work "Bitcoin steigend" by Tim Reckmann/Flickr, CC BY 2.0)

Chapter Outline

5.1 Introduction to Time Series Analysis
5.2 Components of Time Series Analysis
5.3 Time Series Forecasting Methods
5.4 Forecast Evaluation Methods

Introduction

The future can be very unpredictable. Wouldn't it be wonderful to know whether a given stock price will rise or fall in value over the next year, which baseball team will win the next World Series, or if it might rain on the particular day that you happen to be planning to go to the beach? A manufacturer may want to know how the demand for their products changes over time. Politicians would like to know how high their favorability with voters will be on election day. While we cannot see the future, there may be patterns that point to certain outcomes as being more likely than others. In this chapter we explore the concept of time series and develop the necessary tools to discover trends and repeating patterns hidden within time series data to forecast future values of the series and quantify the uncertainty of those predictions.

5.1 Introduction to Time Series Analysis

Learning Outcomes

By the end of this section, you should be able to:

- 5.1.1 Define time series data.
- 5.1.2 Identify examples of time series data in real-world applications.

Data can change over time, and it is important to be able to identify the ways that the data may change in order to make reasonable forecasts of future data. Does the data seem to be steadily rising or falling overall? Are there ups and downs that occur at regular intervals? Maybe there are predictable fluctuations in the data

that are tied closely to the seasons of the year or multiyear business cycles.

Time series analysis allows for the examination of data points collected or recorded at specific time intervals, enabling the identification of trends, patterns, and seasonal variations crucial for making informed predictions and decisions across a variety of industries. It is widely used in fields such as business, finance, economics, environmental and weather science, information science, and many other domains in which data is collected over time.

Some examples of the areas in which time series analysis is used to solve real-world problems include the following:

- **Forecasting and prediction.** Making predictions about future (unknown) data based on current and historical data may be the most important use of time series analysis. Businesses use such methods to drive strategy decisions and increase future revenues. Policymakers rely on forecasting to predict demographic changes that might influence elections and the need for public services.
- **Risk management.** Time series analysis can help to quantify risk. For example, insurance companies may forecast trends that affect the number and total dollar amounts of claims homeowners may make in the future, which directly affects insurance premiums.
- **Anomaly detection.** If data seems to deviate wildly from predictions, then this could point to a major change in some feature of the data or uncover a hidden factor that will need to be analyzed more carefully. For example, if a family's spending patterns increase significantly beyond predicted levels, this could indicate either a sudden increase in household earnings or potential fraud.
- **Health care and epidemiology.** Time series analysis is often used to monitor patient's vitals, study how cancer progresses in individuals from various populations, track disease outbreaks, and predict future need for health care resources such as vaccines or additional beds in hospitals, among other things.

Time series analysis involves a number of statistical and computational techniques to identify patterns, make predictions, or gain insights into the data. The basic steps in time series analysis closely follow the data science cycle as described in What Are Data and Data Science? and are as follows:

1. **Problem Identification/Definition.** What is it that you need to know? Do you want to predict sales of a product over the next few years? Are you interested in studying the patterns of sunspots over time? Is your investment in a stock likely to lead to profit or loss?
2. **Data Gathering.** Once you have a question in mind, now you need data to analyze. This step can be very time-consuming if a suitable dataset does not already exist. In the best-case scenario, others have already collected data that you might be able to use. In the worst-case scenario, you would need to arrange for data collection by surveys, experiments, observations, or other means. Note that data collection is covered in some detail in Overview of Data Collection Methods.
3. **Data Cleaning.** Real-world data rarely comes in a form that is immediately able to be analyzed. Most often, the data must be cleaned, which includes dealing with missing data in some way and formatting the data appropriately so that a computer can understand it. This topic is covered in Web Scraping and Social Media Data Collection.
4. **Model Selection.** The next step is to fit the data to a model. Indeed, you may find that you must create numerous models and variations on those models before you are satisfied with the result.
5. **Model Evaluation.** This is perhaps the most important step. Any model that you create will be limited in its usefulness if it is not accurate enough. The model should include some indication of how confident you can be in its predictions. Moreover, when new observations become available, the model's forecasts should be checked against them and adjusted as necessary to better fit the data.

What Is a Time Series?

Any set of data that consists of numerical measurements of the same variable collected and organized according to regular time intervals may be regarded as **time series** data. For example, see Table 5.1 on the

S&P 500, an aggregate index of the top 500 publicly traded companies on the stock market, for the past several years. Here, the variable is the S&P index value at the last trading day of the year (*point-in-time* data).

Year	S&P Index at Year-End
2013	1848.36
2014	2058.90
2015	2043.94
2016	2238.83
2017	2673.61
2018	2506.85
2019	3230.78
2020	3756.07
2021	4766.18
2022	3839.50
2023	4769.83

Table 5.1 S&P Index Values from 2013 through 2023 (source: https://www.nasdaq.com/market-activity/index/spx/historical)

While the table is informative, it is not particularly easy to use to find trends or make predictions. A visualization would be better. Figure 5.2 displays the time series data using a time series chart (essentially a line chart). Notice that while the general trend is increasing, there are also dips in the years 2018 and 2022.

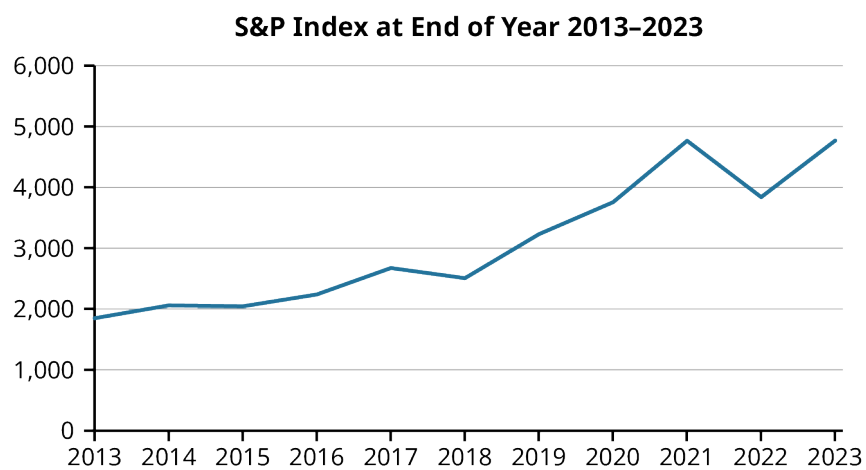


Figure 5.2 Line Chart (or Time Series Chart) of the S&P 500 Index. The chart makes it easier to spot the general upward trend over the past decade. (data source: adapted from S&P 500 [SPX] Historical Data)

The example shown in Figure 5.3 is a simple time series with only a single measure (S&P 500 index value) tracked over time. As long as we keep in mind that this data represents yearly values of the S&P 500 index starting in the year 2013, it is more efficient to consider the values alone as an ordered list.

1848.36, 2058.9, 2043.94, 2238.83, 2673.61, 2506.85, 3230.78, 3756.07, 4766.18, 3839.5, 4769.83

In mathematics, an ordered list of numbers is called a **sequence**. The individual values of the sequence are called **terms**. An abstract sequence may be denoted by (x_n) or $(x_n)_{1 \leq n \leq N}$. In both notations, n represents the index of each value of the sequence, while the latter notation also specifies the range of index values for the

sequence (n takes on all index values from 1 to N). That is,

$$(x_n)_{1 \le n \le N} = (x_1, x_2, x_3, x_4, \ldots, x_N)$$

We will use the standard term *time series* to refer to a sequence of time-labeled data and use the term *sequence* when discussing the terms of the time series as an ordered list of values.

Not every sequence of data points is a time series, though. The set of current populations of the countries of the world is not a time series because the data are not measured at different times. However, if we focused on a single country and tracked its population year by year, then that would be a time series. What about a list of most popular baby names by year? While there is a time component, the data is not numerical, and so this would not fall under the category of time series. On the other hand, information about how many babies are named "Jennifer" each year would constitute time series data.

Typically, we assume that time series data are taken at equal time intervals and that there are no missing values. The terms of a time series tend to depend on previous terms in some way; otherwise, it may be impossible to make any predictions about future terms.

Time Series Models

A **time series model** is a function, algorithm, or method for finding, approximating, or predicting the values of a given time series. The basic idea behind time series models is that previous values should provide some indication as to how future values behave. In other words, there is some kind of function that takes the previous values of the time series as input and produces the next value as output:

$$x_{n+1} = f(x_n, x_{n-1}, x_{n-2}, \ldots, x_1)$$

However, in all but the most ideal situations, a function that predicts the next value of the time series with perfect accuracy does not exist. Random variation and other factors not accounted for in the model will produce **error**, which is defined as the extent to which predictions differ from actual observations. Thus, we should always incorporate an error term (often denoted by the Greek letter ε, called "epsilon") into the model. Moreover, instead of expecting the model to produce the next term exactly, the model generates predicted values. Often, the predicted values of a times series are denoted by $(\hat{x}_n)$ to distinguish them from the actual values (x_n). Thus,

$$\hat{x}_{n+1} = f(x_n, x_{n-1}, x_{n-2}, \ldots, x_1, \varepsilon)$$

In Time Series Forecasting Methods and Forecast Evaluation Methods, we will delve into the details of building time series models and accounting for error.

Time Series Forecasting

Typically, the goal of time series analysis is to make predictions, or *extrapolations*, about future values of the time series, a process known as **forecasting**. As a general rule, the accuracy of a forecast *decreases* as predictions are made further into the future. When future predictions become no more accurate than tossing a coin or rolling a die, then forecasting at the point or beyond becomes ineffective. In practice, time series models are updated regularly to accommodate new data.

Depending on the situation and the nature of the data, there are many different ways to forecast future data. The simplest of all methods, which is known as the **naïve** or **flat forecasting method**, is to use the most recent value as the best guess for the next value. For example, since the S&P 500 had a value of 3,839.5 at the end of 2022, it is reasonable to assume that the value will be relatively close to 3,839.5 at the end of 2023. Note, this would correspond to the time series model, $\hat{x}_{n+1} = x_n$. The naïve method has only limited use in practice.

Instead of using only the last observed value to predict the next one, a better approach might be to take into consideration a number of values, $x_n, x_{n-1}, x_{n-2}, \ldots$, to find the estimate $\hat{x}_{n+1}$. One might average the last T

values together (for some predefined value of T). This is known as a *simple moving average*, which will be defined explicitly in Components of Time Series Analysis. For now, let's illustrate the idea intuitively. Suppose we used the average of the most recent $T = 3$ terms to estimate the next term. The time series model would be:

$$\hat{x}_{n+1} = \frac{x_n + x_{n-1} + x_{n-2}}{3}$$

Based on the data in Table 5.1, the prediction for the S&P index value at the end of 2023 would work out as follows.

$$\frac{4{,}769.83 + 3{,}839.5 + 4{,}766.18}{3} = 4{,}458.5$$

Another simple method of forecasting is to fit the data to a linear regression model and then use the regression model to predict future values. (See Linear Regression for details on linear regression.) Linear regression does a better job at capturing the overall direction of the data compared to using only the last data point. On the other hand, linear regression will not be able to model more subtle structures in the data such as cyclic patterns. Moreover, there is a hidden assumption that the data rises or falls more or less uniformly throughout the period we would like to predict, which is often not a valid assumption to make.

EXAMPLE 5.1

Problem

Find a linear regression model for the data from Table 5.1 and use it to forecast the value of the S&P at the end of the years 2024 and 2025.

Solution

Using standard statistical software, the linear regression is found to be $\hat{y} = 304.437t - 611{,}288$, where t is the year. Alternatively, in Excel, the regression line can be added to the line graph using the "Trendline" feature, with option "Linear." A graph of the regression line is shown in Figure 5.3.

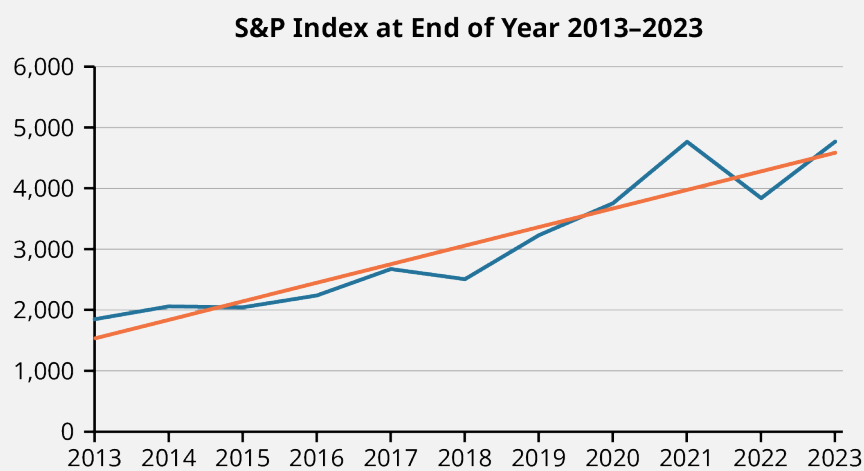


Figure 5.3 Line Chart of the S&P 500 Index with the Linear Regression (data source: adapted from S&P 500 [SPX] Historical Data)

The forecast for the end of year 2024 is $\hat{y} = 304.437(2024) - 611{,}288 = 4{,}892.5$.

Similarly, the prediction for 2025 is $\hat{y} = 304.437(2025) - 611{,}288 = 5{,}196.9$.

In this chapter, we will develop more sophisticated tools that can detect patterns that averages or linear models simply cannot find, including autoregressive models, moving averages, and the autoregressive

integrated moving average (ARIMA) model. However, creating a fancy model for time series data is only part of the process. An essential part of the process is testing and evaluating your model. Thus, we will also explore measures of error and uncertainty that can determine how good the model is and how accurate its forecasts might be.

Examples of Time Series Data

Time series data is typically measured or observed at regular time intervals, such as daily, weekly, monthly, or annually. While we do hope to find some structure, pattern, or overall trend in the data, there is no requirement that time series data follows a simple, predictable pattern. Thus, we should always be careful when using any statistical or predictive models to analyze time series data. Regular validation and testing of the chosen models against new data are essential to ensure their reliability and effectiveness, which are concepts that we will explore in later sections.

Time series data could represent short-term, frequent observations, such as minute-by-minute monitoring of stock prices. It makes sense to measure stock prices at very short time intervals because they tend to be so volatile. Data is considered to be *volatile* or displaying high **volatility** if the individual data points show significant fluctuations from the mean, typically due to factors that are difficult to analyze or predict. Volatility can be measured using statistics such as variance and standard deviation; however, it is important to realize that even the level of volatility can change over time. Measuring data at more frequent intervals does help to reduce or control volatility, but it is still difficult to forecast volatile data to make future predictions.

Other sets of data represent more long-term measurements. For example, meteorologists may collect weather data, including temperature, atmospheric pressure, rainfall, wind speed and direction, in a particular location on a daily basis. Over multiple years, seasonal patterns emerge. Time series analysis may be used to pick up on the seasonal variations and then, after taking those variations into account, may find that the overall trend for temperatures is rising or falling in the long term.

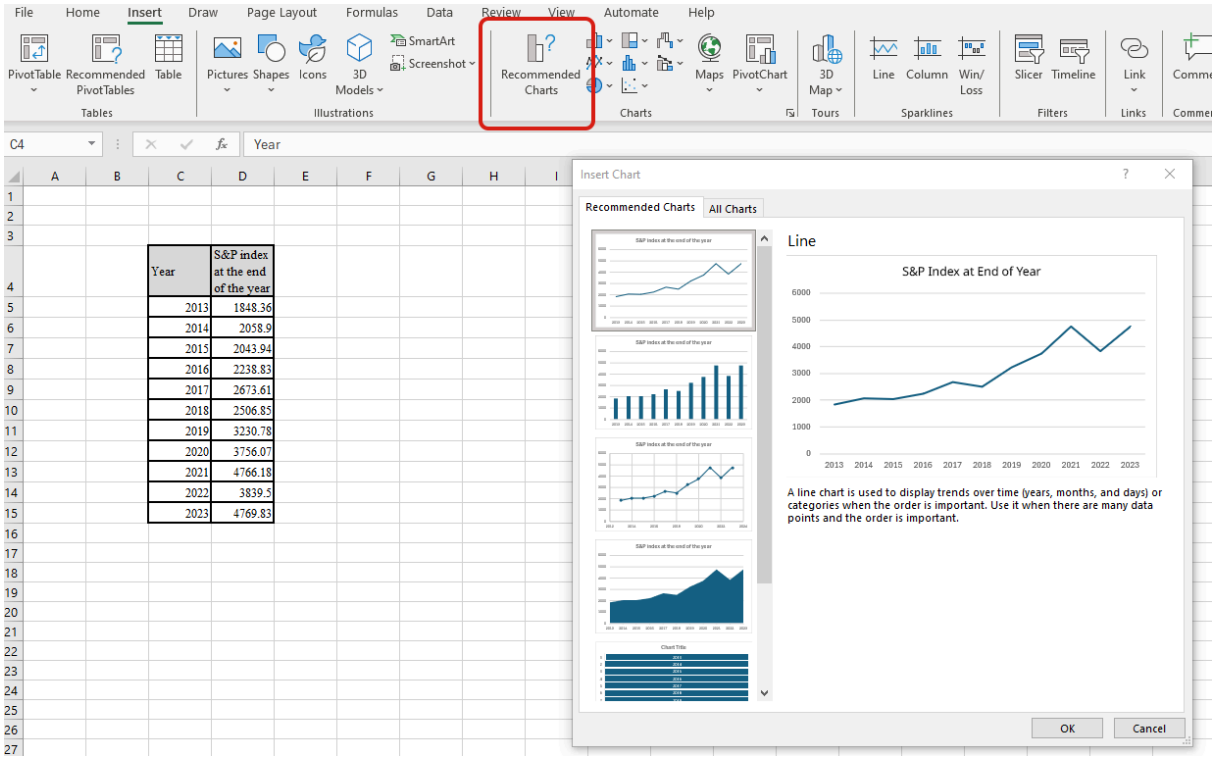
Sales data may be measured weekly or monthly. This kind of data is useful in identifying peak sales periods and slumps when customers are not buying as much. Important business decisions are often driven by analysis of time series data.

Finally, very long-term time series may be useful in the sciences. Monitoring levels of atmospheric carbon over time and comparing these measurements with the levels found by extracting ice cores from Antarctica can demonstrate a statistically significant change to long-term cycles and patterns.

Working with and Visualizing Time Series Data in Excel

You may recall from What Are Data and Data Science? that spreadsheet applications are very well suited for basic analysis and visualization of time series data. Simply create two columns, one for the time periods and the other for the data. Highlight both columns (together with their headings) and insert a line chart using the "Recommended Charts" feature in Excel[1], resulting in a chart such as the one shown in Figure 5.4.

1 Microsoft Excel for Microsoft 365



Year	S&P index at the end of the year
2013	1848.36
2014	2058.9
2015	2043.94
2016	2238.83
2017	2673.61
2018	2506.85
2019	3230.78
2020	3756.07
2021	4766.18
2022	3839.5
2023	4769.83

Figure 5.4 Using Excel to Create a Line Chart. Instead of using "Recommended Charts," you could also choose line chart from the panel of choices located on the same tab. (Used with permission from Microsoft)

Working with and Visualizing Time Series Data in Python

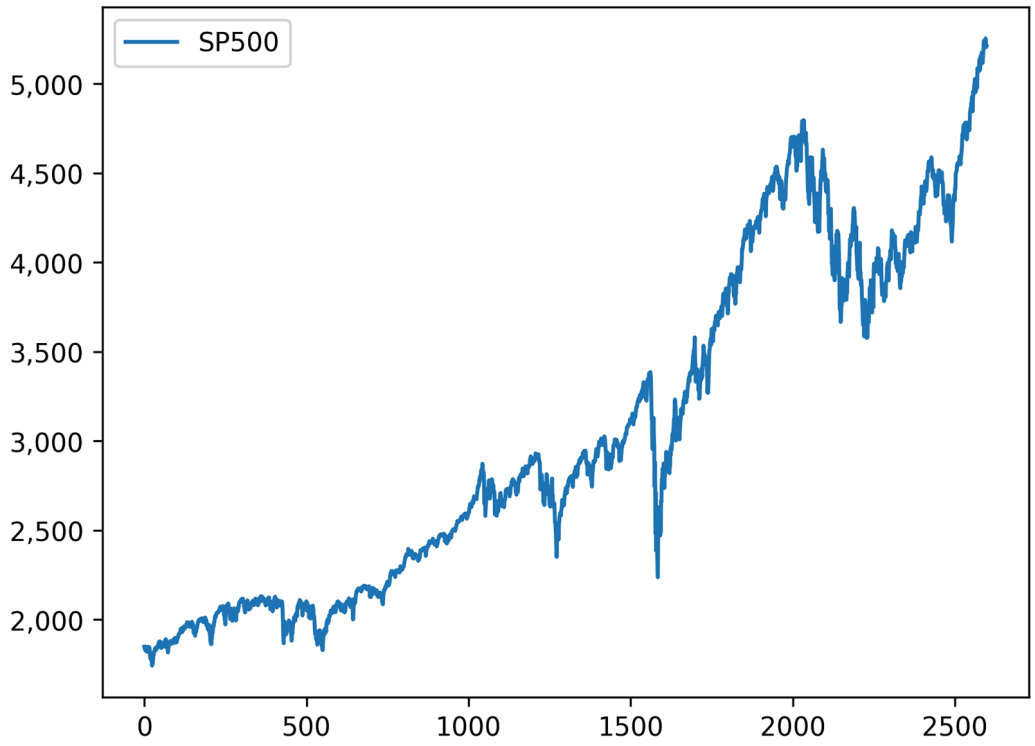
In Python, the pandas library (https://openstax.org/r/pydata) is incredibly useful for reading in CSV files and managing them as data structures called *DataFrames*. It also includes a simple command for visualizing the time series, `plot()`. Here is how to create a line chart of the daily values of the S & P 500 Index from 2014 through 2024 from the dataset SP500.csv (https://openstax.org/r/SP500).

PYTHON CODE

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv('SP500.csv')

# Create a plot
fig, ax = plt.subplots()
df.plot(ax=ax)

# Use FuncFormatter to add commas to the y-axis
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos:
'{:,}'.format(int(x))))
```

```
# Display the plot
plt.show()
```

The resulting output will look like this:



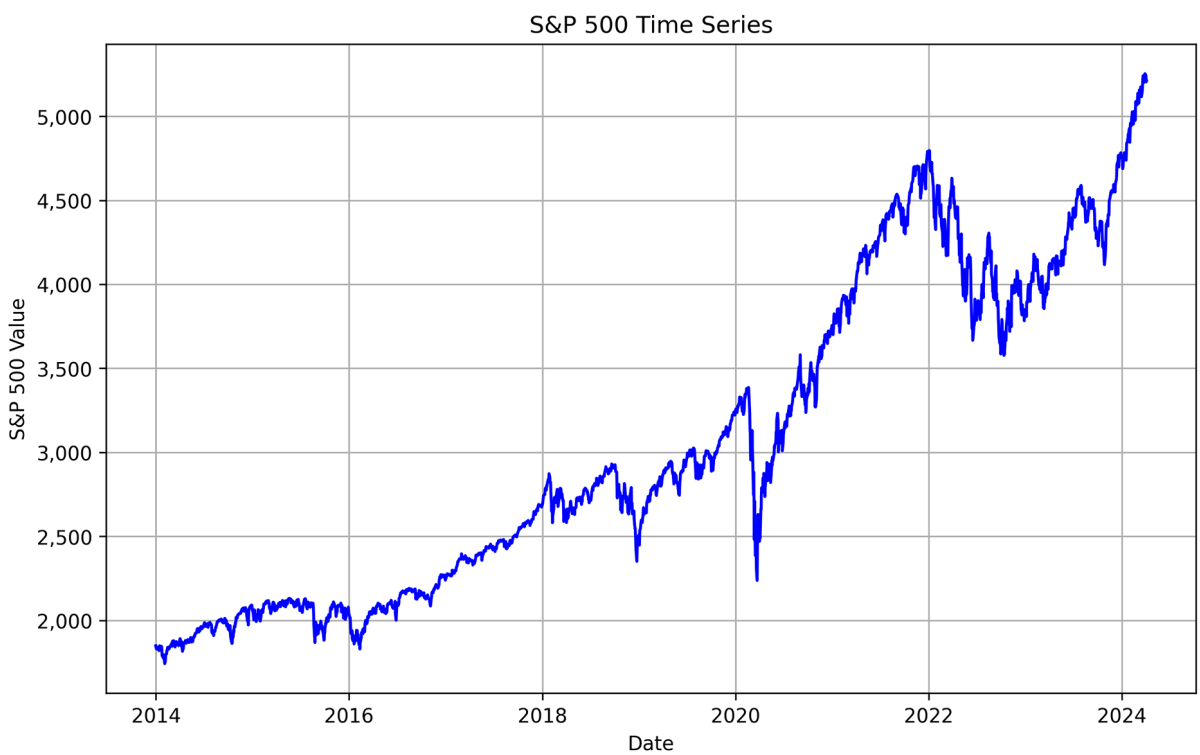
Notice that the simple plot feature did not label the x or y axes. Moreover, the x-axis does not show actual dates, but instead the number of time steps. For simple visualizations that you do not intend to share with a wider audience, this may be ok, but if you want to create better-looking and more informative graphs, use a library such as matplotlib.pyplot (https://openstax.org/r/matplotlib). In the next set of Python code, you will see the same dataset visualized using `matplotlib`. (Note that there is a line of code that uses the `to_datatime()` command. Dates and times are notoriously inconsistent from dataset to dataset, and software packages may not interpret them correctly without a little help.)

PYTHON CODE

```
# Import libraries
import pandas as pd ## for dataset management
import matplotlib.pyplot as plt ## for visualizations
from matplotlib.ticker import FuncFormatter ## formatting Y-axis values

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv('SP500.csv')

# Convert the 'DATE' column to datetime format
```

```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Function to format the Y-axis values
def y_format(value, tick_number):
  return f'{value:,.0f}'

# Plotting the time series data using Matplotlib
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['SP500'], marker='', linestyle='-', color='b')
plt.title('S&P 500 Time Series')
plt.xlabel('Date')
plt.ylabel('S&P 500 Value')
plt.grid(True)

# Apply the formatter to the Y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(y_format))

plt.show()
```

The resulting output will look like this:



5.2 Components of Time Series Analysis

Learning Outcomes

By the end of this section, you should be able to:

- 5.2.1 Define the trend curve of a time series.
- 5.2.2 Define and identify seasonal and cyclic variations in time series data.
- 5.2.3 Discuss how noise and error affect a time series.

Time series data may be *decomposed*, or broken up into several key components. There may be an overall direction (up or down) that the data seems to be following, also known as the *trend*. Cycles and seasonal patterns may be present. Finally, noise or random variation can often be separated from the rest of the data. Time series analysis begins with identifying and extracting the components that contribute to the structure of the data.

Summarizing, the four main components of a time series are defined as follows:

1. **Trend**. The long-term direction of time series data in the absence of any other variation.
2. **Cyclic component**. Large variations in the data that recur over longer time periods than seasonal fluctuations but not with a fixed frequency.
3. **Seasonal component** (often referred to as **seasonality**). Variation in the data that occurs at fixed time periods, such as every year in the same months or every week on the same days.
4. **Noise/Random Error**. Random, unpredictable variations that occur in the time series that cannot be attributed to any of the other components previously listed. (Note, the term "random" is not the same as *statistically independent*.)

In practice, the cyclic component is often included with the trend, which is known as the **trend-cycle component**, since the cyclic component cannot be predicted in the same way that a seasonal component might be. Thus, time series data may be considered to be made up of three components: the trend (or trend-cycle) component, the seasonal component (or components), and noise or random error. Indeed, you might consider that trend, seasonal, and noise components are three separate time series that all contribute to the time series data. Although we still want to think of each component as a sequence of values, it is customary to drop the parentheses in the notation. Thus, the time series will be denoted simply by x_n rather than (x_n), and the three components will be denoted in a similar way.

Now the ways in which the components are combined in a time series analysis varies. There are additive decompositions, multiplicative decompositions, and hybrid decompositions.

In a simple **additive decomposition**, the time series is decomposed into the sum of its three components, t_n for the trend, s_n for the seasonal variation, and ε_n for the error term, as shown:

$$x_n = t_n + s_n + \varepsilon_n$$

In a strictly **multiplicative decomposition**, the time series is decomposed into the product of its components (which will look quite different from their additive counterparts, and so capital letters are used to distinguish them).

$$x_n = T_n \cdot S_n \cdot E_n$$

Finally, a hybrid decomposition in which the trend-cycle and seasonal components are multiplied while the error term remains as an additive component is often useful. In this case, we might write:

$$x_n = T_n \cdot S_n + \varepsilon_n$$

Multiplicative and hybrid decompositions can be more useful if the variation above and below the trend curve seems to be proportional to the level of the trend itself. In other words, when the trend is higher, the variation around the trend is also larger. We will focus only on additive decomposition in this text. In fact, there is a

simple transformation using properties of logarithms that changes a strictly multiplicative decomposition into an additive one as long as all of the components consist of positive values.

$$\log(x_n) = \log(T_n \cdot S_n \cdot E_n) = \log(T_n) + \log(S_n) + \log(E_n)$$

Note that regression methods may be used in conjunction with decomposition. This section will cover the fundamentals of decomposition and regression for time series analysis.

The Trend Curve and Trend-Cycle Component

As previously defined, the *trend* is the direction that the data follows when variations such as seasonal ups and downs and random noise and error are ignored. The **trend curve** (or **trendline**) models the trend by a line or curve, representing something like a *mean value* for the time series in any given time frame. By this definition, the linear regression shown in Example 5.1 would be a trend curve for the S&P 500 Index values. If there is no significant rise or fall in the long-term (that is, only short-term fluctuations or patterns exist), then the trend may be estimated by a constant called the **level**, which is the mean of all the time series data.

Visualizing a Trend Curve

There is no single agreed-upon approach to finding the trend of a time series. The mechanics of isolating a trend curve or trend-cycle component are covered in detail in Time Series Forecasting Methods. For now, we shall see by example what a trend curve might look like in a dataset.

Figure 5.5 shows the monthly consumption of coal for generating electricity in the United States from 2016 through 2022, as provided in the dataset MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j) (source: https://www.statista.com/statistics/1170053/monthly-coal-energy-consumption-in-the-us/ (https://openstax.org/r/statistics). Note: The data was cleaned in the following ways: *Month* converted to a standard date format from string format; *Value* converted to numerical data from string format.)

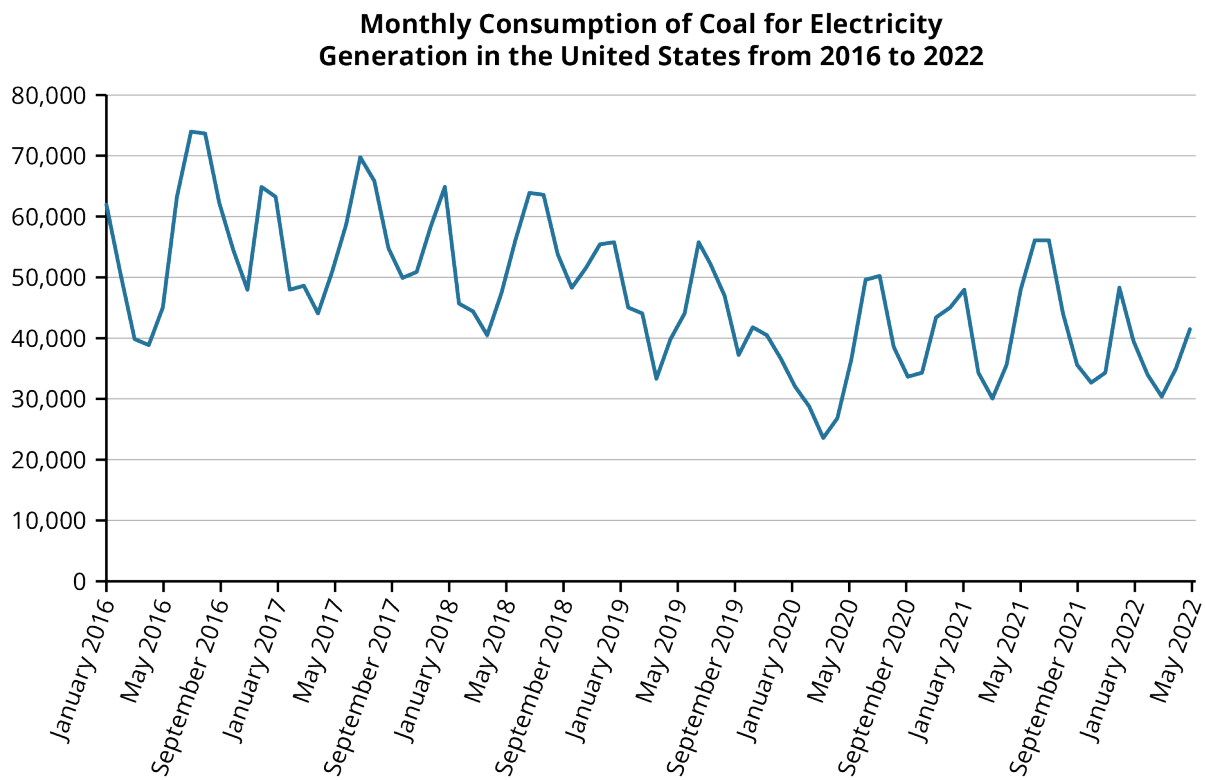


Figure 5.5 Monthly Consumption of Coal for Electricity Generation in the United States from 2016 to 2022, in 1000s of Tons (data source: https://www.statista.com/statistics/1170053/monthly-coal-energy-consumption-in-the-us/)

Although there are numerous ups and downs, there is clearly an overall trajectory of the data. If you were given the task to draw a curve that best captures the underlying trend, you might draw something like the

curve shown in Figure 5.6. (This is what we refer to as a *trendline* or *trend curve*.)

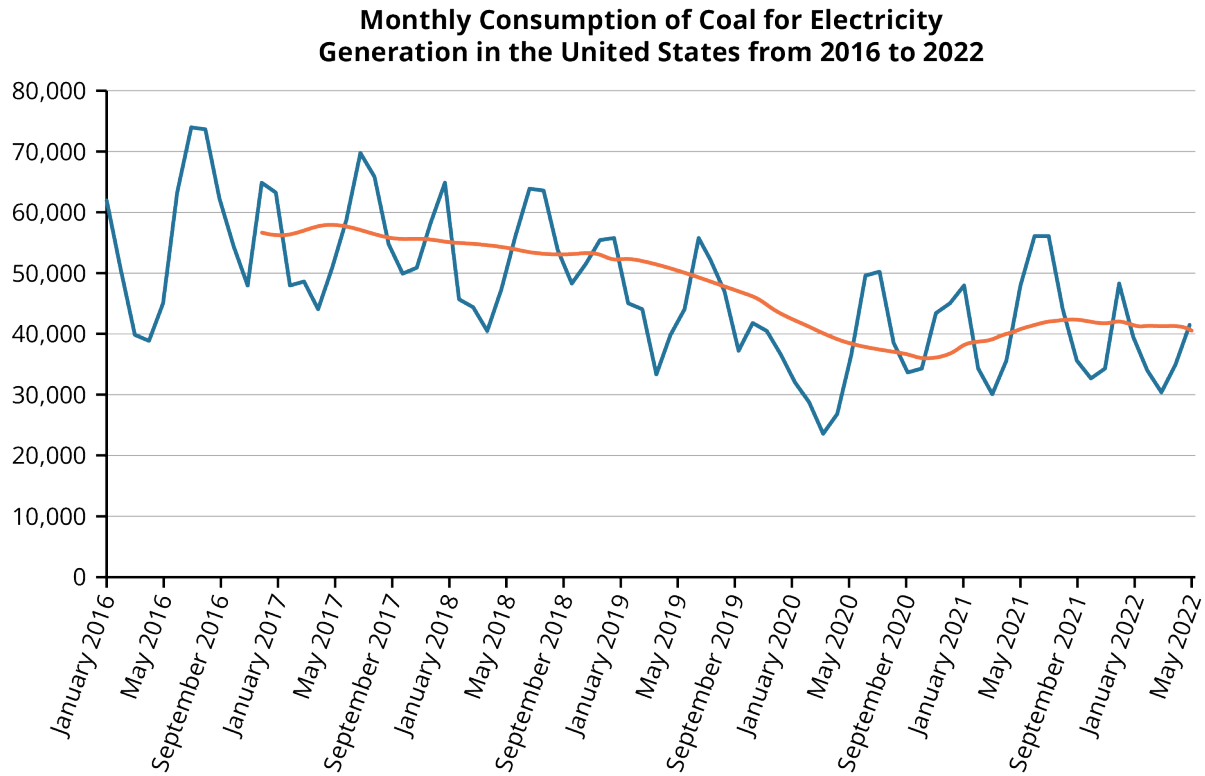


Figure 5.6 Monthly Consumption of Coal for Electricity Generation in the United States from 2016 to 2022, in 1000s of Tons with Trend Curve Shown (data source: https://www.statista.com/statistics/1170053/monthly-coal-energy-consumption-in-the-us/)

Trendline in Excel

Excel has some built-in features that make it easy to include trend curves directly onto the graph of a time series. First, make a line graph of the data in the same way as shown in Introduction to Time Series Analysis. (Note: If "Recommended Charts" does not display the type of chart that you want, you can click on the "All Charts" tab and choose another. For Figure 5.7, we chose the line chart rather than the default column chart.) Next, you can click on "Add Chart Element" in the "Chart Design" area of Excel. One of the options is "Trendline" (see Figure 5.7). To get a trendline like the one in Figure 5.6, select "More Options" and adjust the period of the trendline to 12. After adding the trendline, you can change its color and design if you wish.

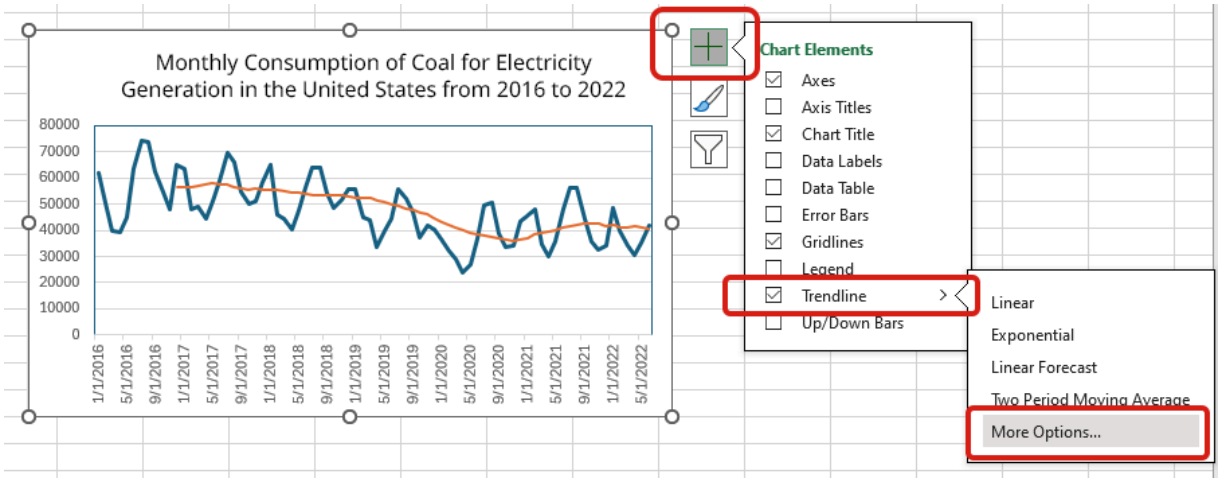


Figure 5.7 Adding a Trendline to a Line Chart in Excel (Used with permission from Microsoft)

Seasonal Variations

We've seen that *seasonality*, or seasonal variations, are predictable changes in the data occurring at regular intervals. The smallest interval of time at which the pattern seems to repeat (at least approximately so) is called its **period**.

For example, suppose you live in a temperate climate in the northern hemisphere and you record the temperature every day for a period of multiple years. You will find that temperatures are warmer in the summer months and colder in the winter months. There is a natural and predicatable pattern of rise and fall on a time scale that repeats every year. This is a classic example of seasonal variation with period 1 year.

However, the period of seasonal variation may be much longer or much shorter than a year. Suppose that you record temperatures near your home every hour for a week or so. Then you will notice a clear repeating cycle of rising and falling temperatures each day—warmer in the daytime and colder each night. This daily cycle of rising and falling temperatures is also a seasonal variation in the time series of period 24 hours. Indeed, any given time series may have multiple seasonal variation patterns at different periods.

EXAMPLE 5.2

Problem

Walt's Water Adventures is a local company that rents kayaks and canoes for river excursions. Due to mild winters in the area, WWA is able to rent these items year-round. Quarterly revenues from rentals were recorded for three consecutive years and also shown in Table 5.2. Is there a clear seasonal variation in the dataset, and if so, what is the period of the seasonal variation? When do the high points and low points generally occur within each seasonal pattern?

Quarter	Revenues ($)
2021 Spring	1799
2021 Summer	3304
2021 Fall	1279
2021 Winter	999
2022 Spring	2284
2022 Summer	3459
2022 Fall	1964
2022 Winter	1594
2023 Spring	2704
2023 Summer	3849
2023 Fall	2131
2023 Winter	1939

Table 5.2 Quarterly Revenues for Kayak and Canoe Rentals

Solution

The period is 1 year, or 4 quarters, with high values occurring in summer of each year and low values occurring in winter of each year as shown in Figure 5.8.

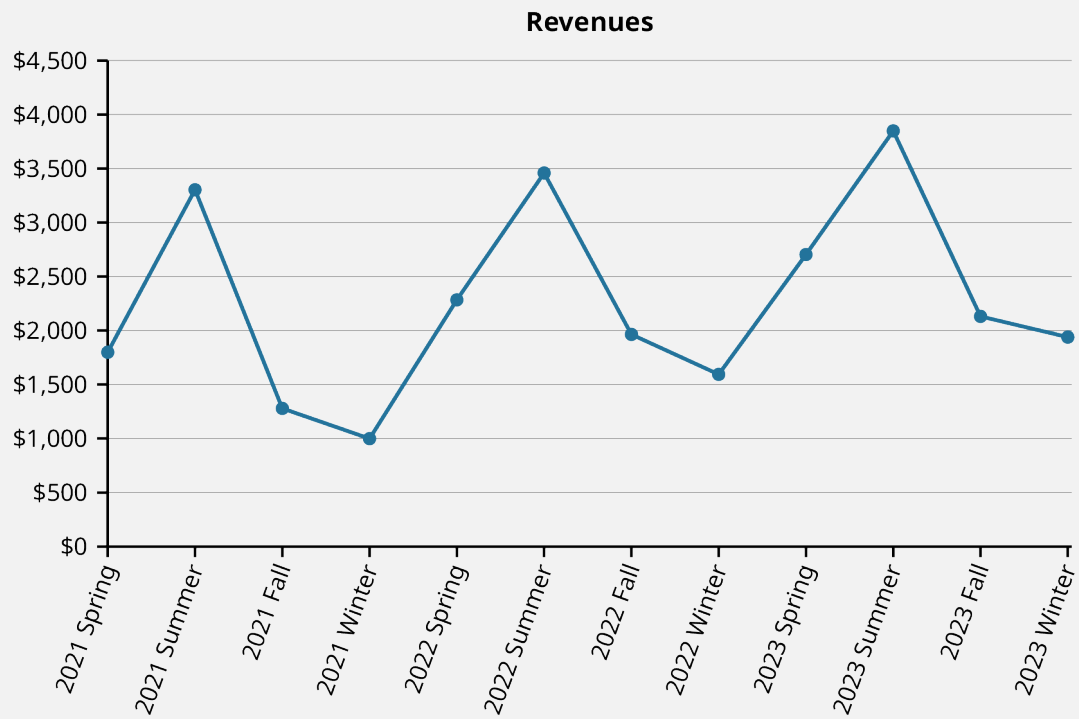


Figure 5.8 Quarterly Revenues for Kayak and Canoe Rentals

Noise and Random Variation

All data is noisy. There are often complex factors that affect data in ways that are difficult or impossible to predict. Even if all factors could be accounted for, there will inevitably be some error in the data due to mistakes, imprecise measurements, or even how the values are rounded.

Noise shows up in time series data as small random ups and downs without an identifiable pattern. When visualizing the time series, noise is the component of the data that makes the line graph look jagged. In the absence of noise or random variation, the plot of a time series would look fairly smooth.

In a time series, once the trend, cyclic variation, and seasonal variation are all quantified, whatever is "left over" in the data will generally be noise or random fluctuations. In other words, once the trend (or trend-cycle component) t_n and seasonal variation s_n are accounted for, then the noise component is just the remaining component:

$$\varepsilon_n = x_n - t_n - s_n$$

In the real world, we rarely have access to the actual t_n or s_n series directly. Instead, almost all time series forecast methods produce a model $(\hat{x}_n)$ that tries to capture the trend-cycle curve and seasonal variation. The **residuals**, which are the time series of differences between the given time series (x_n) and the model, provide a measure of the error of the model with respect to the original series. If the model is good, in the sense that it captures as much of the predictable aspects of the time series as possible, then the residuals should give an estimate of the noise component of the time series. Therefore, it is customary to use the same notation for residuals as for the noise term. Thus,

$$\varepsilon_n = x_n - \hat{x}_n$$

The residuals should have the characteristics of a truly random time series, known as **white noise**, which has the following key characteristics:

1. **Constant zero mean.** The mean should be close to zero anywhere in the series.

2. **Constant variance.** The variance should be approximately the same anywhere in the series. (Note: *Variance* is defined in Measures of Variation.)
3. **Uncorrelated.** The sequence of values at one point in the series should have a correlation close to zero when compared to any sequence of values of the same size anywhere else in the series. (Note: *Correlation* is covered in Correlation and Linear Regression Analysis.)

In the definitions given, the phrase *anywhere in the series* refers to a sample of consecutive terms taken from the series $(x_k, x_{k+1}, x_{k+2}, x_{k+3}, \ldots, x_{k+r})$, which is also called a **window**. How many terms are included in a window? That depends on the time series, but just keep in mind that the window should be large enough to accurately reflect the mean and variance of nearby points while not being so large that it fails to detect actual ups and downs present in the data. Statistical software can be used to check whether a time series has the characteristics of white noise, and we will discuss the topic further in the next section. For now, check out Figure 5.9, which displays a time series of white noise.

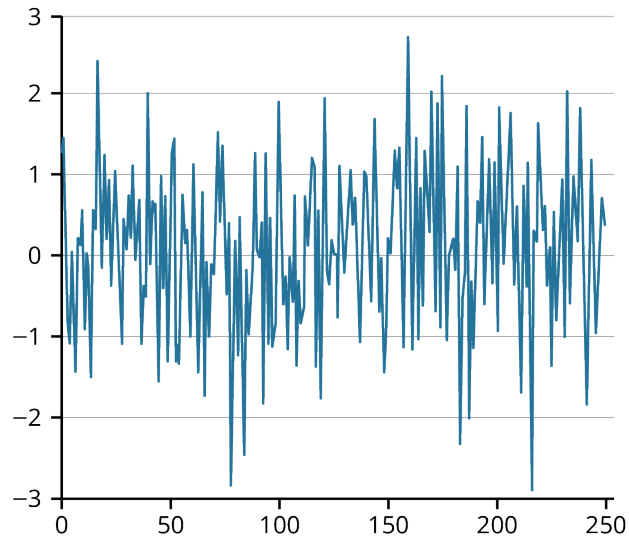


Figure 5.9 White Noise. White noise is a type of random noise. The mean is constantly near zero. Variance is pretty constant everywhere in the series, close to a value of 1. There are no observable patterns that repeat.

5.3 Time Series Forecasting Methods

Learning Outcomes

By the end of this section, you should be able to:

- 5.3.1 Analyze a time series by decomposing it into its components, including trend, seasonal and cyclic variation, and residual noise.
- 5.3.2 Compute various kinds of moving averages, weighted moving averages, and related smoothing techniques.
- 5.3.3 Describe the ARIMA procedure and how autocorrelation functions can be used to find appropriate parameters.
- 5.3.4 Use ARIMA to make forecasts, with the aid of Python.

This section focuses on analysis of a time series into its components for the purpose of building a robust model for forecasting. Although much of the analysis can be done by hand, numerous computer software applications have been developed to handle this task. Ultimately, we will see how Python can be used to decompose a time series and create a general-purpose model called the autoregressive integrated moving average model (commonly known as ARIMA) to make forecasts.

Detrending the Data and Smoothing Techniques

Detrending a time series refers to the process of separating the underlying trend component t_n from the time

series as a whole through one of two complementary operations: *isolating the trend* and then removing it or *smoothing out the trend* leaving only the seasonal and error components. Methods that reduce the error term and flatten out seasonal variations result in a trend-cycle component t_n that still closely follows the long-term ups and downs in the time series. In this case, the other components of the time series can be found by removing the trend component:

$$x_n - t_n = s_n + \varepsilon_n$$

On the other hand, there are detrending methods that transform the data into a new time series with a trendline closer to a constant line at the level of 0. In other words, after applying such a transformation, the new time series, z_n, would consist of only the seasonal and noise components:

$$z_n = s_n + \varepsilon_n$$

Common methods for isolating or removing the trend or trend-cycle component include the following:

1. **Simple moving average (SMA).** The **simple moving average (SMA)** is found by taking the average (arithmetic mean) of a fixed number of consecutive data points. This has the effect of smoothing out short-term variations in the data and isolating the trend. If the data has seasonal variation, then an SMA of the same length can effectively remove the seasonal component completely—there is more about this in Time Series Forecasting Methods.
2. **Weighted moving average (WMA).** There are several variations on the moving average idea, in which previous terms of the series are given different weights, thus termed **weighted moving average (WMA)**. One of the most common weighted moving averages is the **exponential moving average (EMA)**, in which the weights follow exponential decay, giving more recent data points much more weight than those further away in time. These techniques are more often used in forecasting as discussed later in this section, but they may also be employed as a smoothing method to isolate a trend in the data.
3. **Differencing.** The (first-order) difference of a time series is found by **differencing**, or taking differences of consecutive terms—that is, the nth term of the difference would be $x_{n+1} - x_n$. If the original time series had a linear trend, then the difference would display a relatively constant trend with only noise and seasonal variation present. Similarly, the difference of the difference, or second-order difference, can transform a quadratic trend into a constant trend. Higher-order polynomial trends can be identified using higher-order differences. Thus, we can use differencing to check for the overall trend of a certain type (linear, quadratic, etc.) while at the same time removing the original trend from the time series.
4. **Regression.** As mentioned in Example 5.1, linear regression may be applied to a time series that seems to display a uniform upward or downward trend. Using differencing to identify whether there is a trend of some particular order, then linear, quadratic, or higher-order polynomial regression could be used to find that trend.
5. **Seasonal-Trend decomposition using LOESS (STL). Seasonal-Trend decomposition using LOESS (STL)** is a powerful tool that decomposes a time series into the trend-cycle, seasonal, and noise components. Later in this section we will find out how to perform STL using Python. (Note: LOESS is a regression technique the fits a low-order polynomial—typically linear or quadratic—to small portions of the data at a time. A full understanding of LOESS is not required for using STL.)

There are, of course, more complex methods for analyzing and isolating the trends of time series, but in this text we will stick to the basics and make use of technology when appropriate.

Simple Moving Averages (SMA)

Recall that we use the term *window* to refer to consecutive terms of a time series. In what follows, assume that the size of the window is a constant, T. Given a time series with values (x_n), that is, a sequence $(x_1, x_2, x_3, x_4, \ldots, x_N)$, the simple moving average with window size T is defined as a new series (y_n) with values given by the formula

$$y_n = \frac{x_n + x_{n-1} + x_{n-2} + \ldots + x_{n-T+1}}{T} = \frac{1}{T}\sum_{k=1}^{T} x_{n-T+k}$$

This formula is not as difficult as it looks. Just think of it as the average of the last T terms. For example, given the sequence (5, 4, 2, 6, 2, 3, 1, 7), the SMA of length 4 would be found by averaging 4 consecutive terms at a time. Note that the first term of the SMA is y_4 because at least 4 terms are needed in the average.

Term $k = 4$: $\left(\mathbf{5}, \mathbf{4}, \mathbf{2}, \mathbf{6}, 2, 3, 1, 7\right) \rightarrow y_4 = \frac{5+4+2+6}{4} = \frac{17}{4} = 4.25$

Term $k = 5$: $\left(5, \mathbf{4}, \mathbf{2}, \mathbf{6}, \mathbf{2}, 3, 1, 7\right) \rightarrow y_5 = \frac{4+2+6+2}{4} = \frac{14}{4} = 3.5$

Term $k = 6$: $\left(5, 4, \mathbf{2}, \mathbf{6}, \mathbf{2}, \mathbf{3}, 1, 7\right) \rightarrow y_6 = \frac{2+6+2+3}{4} = \frac{13}{4} = 3.25$

Term $k = 7$: $\left(5, 4, 2, \mathbf{6}, \mathbf{2}, \mathbf{3}, \mathbf{1}, 7\right) \rightarrow y_7 = \frac{6+2+3+1}{4} = \frac{12}{4} = 3$

Term $k = 8$: $\left(5, 4, 2, 6, \mathbf{2}, \mathbf{3}, \mathbf{1}, \mathbf{7}\right) \rightarrow y_8 = \frac{2+3+1+7}{4} = \frac{13}{4} = 3.25$

Thus, the SMA of length 4 is the sequence (4.25, 3.5, 3.25, 3, 3.25). Note how the values of the SMA do not jump around as much as the values of the original sequence.

The number of terms to use in a simple moving average depends on the dataset and desired outcome. If the intent is to reduce noise, then a small number of terms (say, 3 to 5) may be sufficient. Sometimes inconsistencies in data collection can be mitigated by a judicious use of SMA. For example, suppose data is collected on the number of walk-in customers at the department of motor vehicles, which only operates Monday through Friday. No data is collected on the weekends, while the number of walk-ins on Mondays may be disproportionately high because people have had to wait until Monday to use their services. Here, an SMA of 7 days would smooth out the data, spreading the weekly walk-in data more evenly over all the days of the week.

EXAMPLE 5.3

Problem

Find the simple moving average of window size 3 for the data from Table 5.1 and compare with the original line graph.

Solution

The SMA will begin at index $k = 3$.

$$y_3 = \frac{1{,}848.36 + 2{,}058.9 + 2{,}043.94}{3} = 1{,}983.73$$

$$y_4 = \frac{2{,}058.9 + 2{,}043.94 + 2{,}238.83}{3} = 2{,}113.89$$

$$y_5 = \frac{2{,}043.94 + 2{,}238.83 + 2{,}673.61}{3} = 2{,}318.79$$

$$y_6 = \frac{2{,}238.83 + 2{,}673.61 + 2{,}506.85}{3} = 2{,}473.1$$

$$y_7 = \frac{2{,}673.61 + 2{,}506.85 + 3{,}230.78}{3} = 2{,}803.75$$

$$y_8 = \frac{2{,}506.85 + 3{,}230.78 + 3{,}756.07}{3} = 3{,}164.57$$

$$y_9 = \frac{3{,}230.78 + 3{,}756.07 + 4{,}766.18}{3} = 3{,}917.68$$

$$y_{10} = \frac{3{,}756.07 + 4{,}766.18 + 3{,}839.5}{3} = 4{,}120.58$$

$$y_{11} = \frac{4{,}766.18 + 3{,}839.5 + 4{,}769.83}{3} = 4{,}458.5$$

The SMA is graphed together with the original time series in Figure 5.10.

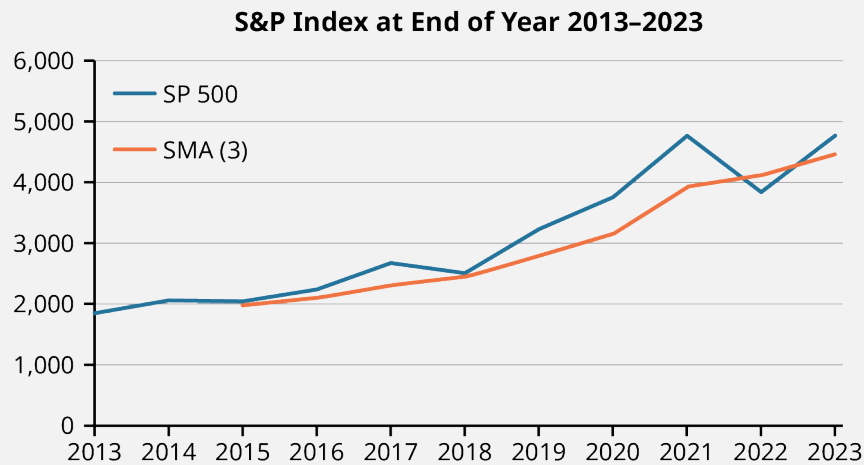


Figure 5.10 Line Chart of the S&P 500 Index Together with the Simple Moving Average of Length 3 (data source: adapted from S&P 500 [SPX] Historical Data)

Simple moving averages provide an easy way to identify a trend-cycle curve in the time series. In effect, the SMA can serve as an estimate of the t_n component of the time series. In practice, when using SMA to isolate the trend-cycle component, the terms of the SMA are **centered**, which means that when computing the terms of the SMA, y_k, the x_k term shows up right in the middle of the window, with the same number of terms to the left and to the right.

For example, the SMA found in Example 5.3 may be shifted so that $y_k = \frac{x_{k-1}+x_k+x_{k+1}}{3}$. This has the effect of shifting the index of the SMA terms back 1 so that the SMA begins at index $k = 2$.

$$y_2 = \frac{x_1 + x_2 + x_3}{3} = \frac{1{,}848.36 + 2{,}058.9 + 2{,}043.94}{3} = 1{,}983.73$$

This procedure works well when the window size is odd, but when it is even, a common practice is to compute two SMAs in which x_k is very nearly in the middle of the sequence and then average their results. For example, using the same data as in Example 5.3, if we wanted to compute a centered SMA of length 4, then the first term would be y_3, and we would find two averages.

$$\frac{x_1 + x_2 + x_3 + x_4}{4} = \frac{1{,}848.36 + 2{,}058.9 + 2{,}043.94 + 2{,}238.83}{4} = 2{,}047.51$$

$$\frac{x_2 + x_3 + x_4 + x_5}{4} = \frac{2{,}058.9 + 2{,}043.94 + 2{,}238.83 + 2{,}673.61}{4} = 2{,}253.82$$

$$y_3 = \frac{2{,}047.51 + 2{,}253.82}{2} = 2{,}150.67$$

The average of the two averages is equivalent to the following more efficient formula. First consider the previous example.

$$y_3 = \frac{1}{2}\left(\frac{x_1 + x_2 + x_3 + x_4}{4} + \frac{x_2 + x_3 + x_4 + x_5}{4}\right) = \frac{x_1}{8} + \frac{x_2}{4} + \frac{x_3}{4} + \frac{x_4}{4} + \frac{x_5}{8}$$

This suggests a general pattern for even window sizes. In summary, we have two closely related formulas for

centered SMA, depending on whether T is even or odd.

$$y_k = \frac{1}{T}\left(x_{k-(T-1)/2} + \ldots + x_{k+(T-1)/2}\right), \text{ for } T \text{ odd}$$

$$y_k = \frac{x_{k-T/2}}{2T} + \frac{1}{T-2}\left(x_{k-T/2+1} + \ldots + x_{k+T/2-1}\right) + \frac{x_{k+T/2}}{2T}, \text{ for } T \text{ even}$$

Of course, there are functions in statistical software that can work out the centered SMA automatically for you.

Differencing

In mathematics, the first-order difference of a sequence $(x_n) = (x_1, x_2, x_3, x_4, \ldots, x_N)$ is another sequence, denoted $\Delta(x_k)$, consisting of the differences of consecutive terms. That is,

$$\begin{aligned} \Delta(x_n) &= \left(x_{n+1} - x_n\right) \\ &= \left(x_2 - x_1, x_3 - x_2, x_4 - x_3, \ldots, x_N - x_{N-1}\right) \end{aligned}$$

Note that there is one fewer term in the difference $\Delta(x_n)$ compared to the original time series (x_n). Differencing has the effect of leveling out a uniform upward or downward trend without affecting other factors such as seasonal variation and noise.

Moreover, the average (arithmetic mean) of the terms of difference series measures of the rate of change of the original time series, much as the slope of the linear regression does. (Indeed, if you have seen some calculus, you might notice that there is a strong connection between differencing and computing the *derivative* of a continuous function.)

EXAMPLE 5.4

Problem

Find the first-order difference time series for the data from Table 5.1 and compare with the original line graph. Find the average of the difference series and compare the value to the slope of the linear regression found in Example 5.1.

Solution

Since the original time series from Table 5.1 has $N = 10$ terms, the difference will have only 9 terms, as shown in Table 5.3.

Term n	S&P Index Term x_n	n^{th} Term of $\Delta(x_n)$
1	1848.36	N/A
2	2058.9	$2058.9 - 1848.36 = 210.54$
3	2043.94	$2043.94 - 2058.9 = -14.96$
4	2238.83	$2238.83 - 2043.94 = 194.89$
5	2673.61	$2673.61 - 2238.84 = 434.78$
6	2506.85	$2506.85 - 2673.61 = -166.76$
7	3230.78	$3230.78 - 2506.85 = 723.93$

Table 5.3 First-Order Difference Time Series for the Data in Table 5.1 (source: adapted from https://www.nasdaq.com/market-activity/index/spx/historical)

Term n	S&P Index Term x_n	n^{th} Term of $\Delta(x_n)$
8	3756.07	$3756.07 - 3230.78 = 525.29$
9	4766.18	$4766.18 - 3756.07 = 1010.11$
10	3839.5	$3839.5 - 4766.18 = -926.68$
11	4769.83	$4769.83 - 3839.5 = 930.33$

Table 5.3 First-Order Difference Time Series for the Data in Table 5.1 (source: adapted from https://www.nasdaq.com/market-activity/index/spx/historical)

Figure 5.11 shows the graph of the original data together with its difference series.

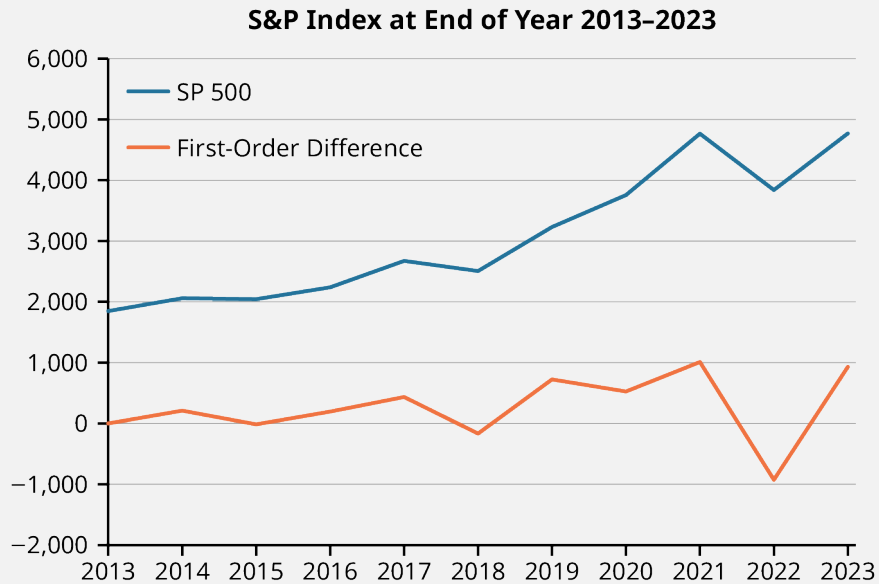


Figure 5.11 Comparison of S&P Index Values and Their Difference Series (data source: adapted from S&P 500 [SPX] Historical Data)

The average value of $\Delta(x_n)$ is:

$$\frac{1}{10}(210.54 + (-14.96) + 194.89 + 434.78 + (-166.76) + 723.93 + 525.29 + 1{,}010.11 + (-926.68) + 930.33) = 292.15$$

Comparing with the slope of the linear regression, which is about 304 (recall Example 5.1), we find the two values are relatively close to each other.

Differencing can be used to check whether a time series trend curve is essentially linear, quadratic, or polynomial of higher order. First, we need to recall a definition from algebra that is typically applied to polynomial functions but can also be applied to sequences. If a sequence (x_n) is given by a polynomial formula of the form

$$x_n = f(n) = a_r n^r + a_{r-1} n^{r-1} + \ldots + a_1 n + a_0$$

where r is a whole number, each of the coefficients a_k (for $k = 1, 2, 3, \ldots, n$) is a constant, and the leading term is nonzero ($a_r \neq 0$), then the sequence has *order* or *degree* r. (**Order** or **degree** refers to the number of terms or lags used in a model to describe the time series.)

For example, any sequence of the form $x_n = an + b$ has order 1 (and may be called *linear*). A sequence of the

form $x_n = an^2 + bn + c$ has order 2 (and is called *quadratic*). Quadratic data fits well to a parabola, which would indicate a curving trend in the data.

If a sequence (x_n) has order $r > 0$, then the difference sequence $\Delta(x_n)$ has order $r - 1$. If the original sequence already is of order 0, which means the sequence is a constant sequence (all terms being the same constant), then the difference sequence simply produces another constant sequence (order 0) in which every term is equal to 0. This important result can be proved mathematically, but we will not present the proof in this text.

As an example, consider the sequence $(x_n) = (2, 5, 8, 11, 14, 17, 20)$, which has order 1, since you can find the values using the formula $x_n = 3n - 1$. The difference sequence is $\Delta(x_n) = (3, 3, 3, 3, 3, 3)$, which is a constant sequence, so it has order 0. Notice that taking the difference one more time yields the constant 0 sequence, $\Delta(\Delta(x_n)) = (0, 0, 0, 0, 0)$, and taking further differences would only reduce the number of terms, but each term would remain at the value 0. (For simplicity, we will assume that there are always enough terms in a sequence so that the difference is also a non-empty sequence.)

The *second-order* difference, $\Delta^2(x_n) = \Delta\big(\Delta(x_n)\big)$, or difference of the difference, would reduce the order of an r-order sequence twice to $r - 2$ if $r \geq 2$ or to the constant zero sequence if $r \leq 1$. In general, a d-order difference, defined by $\Delta^d(x_n) = \Delta\big(\Delta^{d-1}(x_n)\big)$, reduces the order of an r-order sequence to $r - d$ if $r \geq d$ or to order 0 otherwise. With a little thought, you may realize that we now have a simple procedure to determine the order of a sequence that does not require us to have the formula for x_n in hand.

If a sequence (x_n) has a well-defined order (in other words, there is a polynomial formula that is a good model for the terms of the sequence, even if we do not know what that formula is yet), then the order r is the smallest whole number such that the r-order difference, $\Delta^{r+1}\left(x_n\right)$, is equal to either a constant 0 sequence or white noise.

EXAMPLE 5.5

Problem

Determine the order of the sequence (10, 9, 6, 7, 18, 45, 94, 171) using differencing.

Solution

Let (x_n) stand for the given sequence.

First-order difference: $\Delta(x_n) = (-1, -3, 1, 11, 27, 49, 77)$

Second-order difference: $\Delta^2(x_n) = (-2, 4, 10, 16, 22, 28)$

Third-order difference: $\Delta^3(x_n) = (6, 6, 6, 6, 6)$

Fourth-order difference: $\Delta^4(x_n) = (0, 0, 0, 0)$

The given sequence has order 3 since the $(3 + 1)$-order difference resulted in all zeros.

We cannot hope to encounter real-world time series data this is exactly linear, quadratic, or any given degree. Every time series will have some noise in addition to possible seasonal patterns. However, the trend-cycle curve may still be well approximated by a polynomial model of some order.

SMA and Differencing in Excel

Excel can easily do simple moving averages and differencing. The ability to write one equation in a cell and then drag it down so that it works on all the rows of data is a very powerful and convenient feature. By way of example, let's explore the dataset MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j).

First, you will need to save the file as an Excel worksheet, as CSV files do not support equations.

Let's include a centered 13-term window SMA (see Figure 5.12). In cell C8, type "=AVERAGE(B2:B14)" as shown in Figure 5.12. Note: There should be exactly 6 rows above and 6 rows below the center row at which the formula is entered.

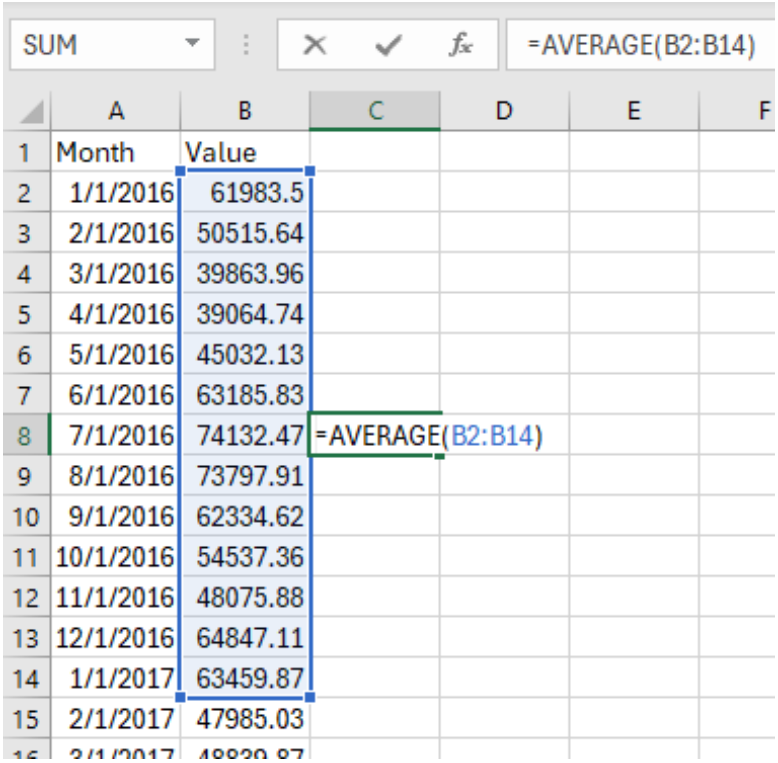


Figure 5.12 Screenshot of a 13-term Excel Window SMA (Used with permission from Microsoft)

Next, click on the square in the lower right corner of cell C8 and drag it all the way down to cell C173, which is 6 rows before the final data value. Column C now contains the SMA time series for your data, as shown in Figure 5.13.

	A	B	C
1	Month	Value	SMA
2	1/1/2016	61983.5	
3	2/1/2016	50515.64	
4	3/1/2016	39863.96	
5	4/1/2016	39064.74	
6	5/1/2016	45032.13	
7	6/1/2016	63185.83	
8	7/1/2016	74132.47	56987
9	8/1/2016	73797.91	55910.2
10	9/1/2016	62334.62	55781.29
11	10/1/2016	54537.36	56120.9
12	11/1/2016	48075.88	57031.12
13	12/1/2016	64847.11	58094.19
14	1/1/2017	63459.87	58600.56
15	2/1/2017	47985.03	57956.64
16	3/1/2017	48839.87	56488.58
17	4/1/2017	44278.92	55540.93
18	5/1/2017	50897.5	55259.73
19	6/1/2017	58852.12	56058.3

Figure 5.13 Screenshot Showing the SMA Time Series in Column C (Used with permission from Microsoft)

You can generate the graph of the time series and the SMA together, which is shown in Figure 5.14. Compare your computed SMA with the trendline created by Excel in Figure 5.7.

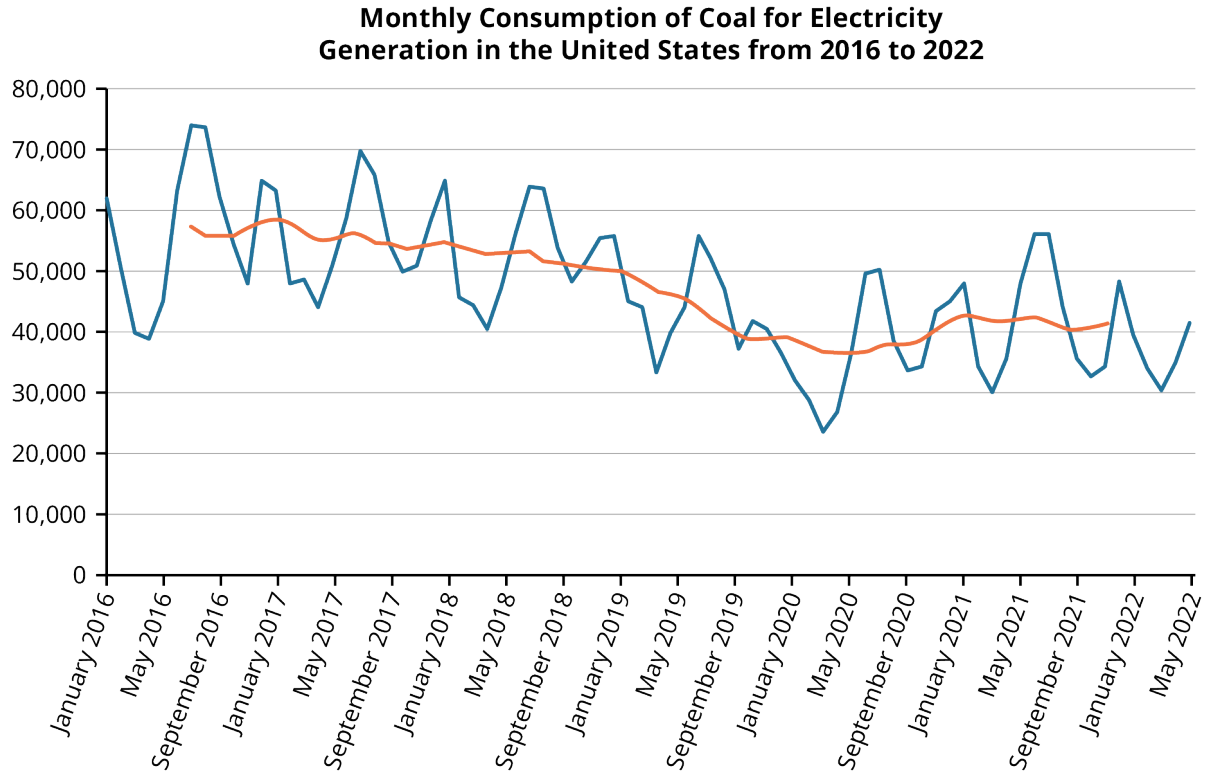


Figure 5.14 Monthly Consumption of Coal for Electricity Generation in the United States from 2016 to 2022. The SMA of window length 13 has been added to the graph in orange. (data source: https://www.statista.com/statistics/1170053/monthly-coal-energy-consumption-in-the-us/)

Let's now work out the first-order difference. The idea is very similar. Put the formula for the difference, which is nothing more than "=B3-B2," into cell D3, as in Figure 5.15.

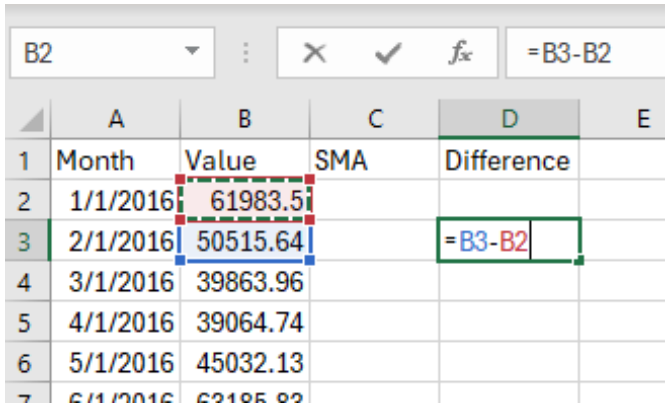


Figure 5.15 Screenshot of Excel Showing Formula for Difference in Column D (Used with permission from Microsoft)

Then drag the formula down to the end of the column. The result is shown in Figure 5.16.

	A	B	C	D
1	Month	Value	SMA	Difference
2	1/1/2016	61983.5		
3	2/1/2016	50515.64		-11467.86
4	3/1/2016	39863.96		-10651.68
5	4/1/2016	39064.74		-799.22
6	5/1/2016	45032.13		5967.39
7	6/1/2016	63185.83		18153.7
8	7/1/2016	74132.47	56987	10946.64
9	8/1/2016	73797.91	55910.2	-334.56
10	9/1/2016	62334.62	55781.29	-11463.29
11	10/1/2016	54537.36	56120.9	-7797.26
12	11/1/2016	48075.88	57031.12	-6461.48
13	12/1/2016	64847.11	58094.19	16771.23
14	1/1/2017	63459.87	58600.56	-1387.24
15	2/1/2017	47985.03	57956.64	-15474.84
16	3/1/2017	48839.87	56488.58	854.84
17	4/1/2017	44278.92	55540.93	-4560.95
18	5/1/2017	50897.5	55259.73	6618.58
19	6/1/2017	58852.12	56058.3	7954.62

Figure 5.16 Screenshot of Excel Showing Difference in Column D (Used with permission from Microsoft)

If you need to compute the second-order differences, just do the same procedure to find the difference of column D. Higher-order differences can easily be computed in the analogous way.

SMA and Differencing in Python

Computing an SMA in Python takes only a single line of code. Here is how it works on the Coal Consumption dataset (found in MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j)), using a window size of 12. (Before, we used a window of 13 because odd-length windows are easier to work with than even-length ones. However, Python easily handles all cases automatically.) The key here is the function `rolling()` that creates a DataFrame of rolling windows of specified length. Then the `mean()` command computes the

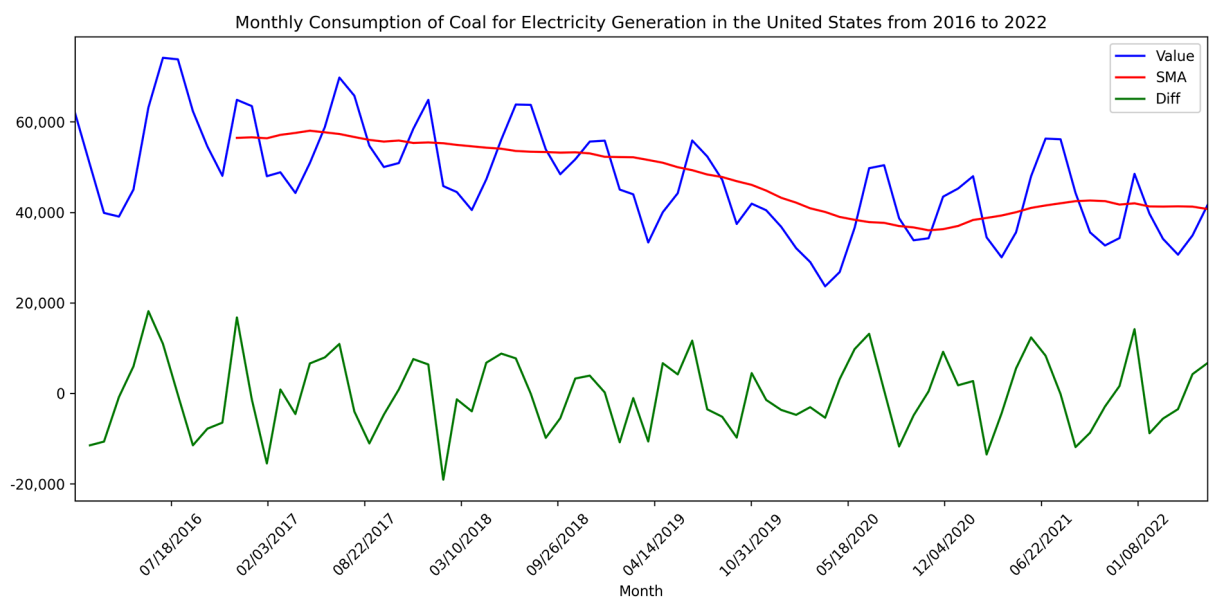
average of each window of data. Computing the first-order difference is just as easy, using the `diff()` command.

PYTHON CODE

```python
# Import libraries
import pandas as pd # for dataset management
import matplotlib.pyplot as plt # for visualizations
from matplotlib.ticker import MaxNLocator, FuncFormatter # for graph formatting and
y-axis formatting
import matplotlib.dates as mdates # for date formatting

# Read the Excel file into a Pandas DataFrame
df = pd.read_excel('MonthlyCoalConsumption.xlsx')

# Convert 'Month' column to datetime format
df['Month'] = pd.to_datetime(df['Month'], format='%m/%d/%Y')

# Calculate a simple moving average with a window size of 12
df['SMA'] = df['Value'].rolling(window=12).mean()

# Calculate the first-order difference
df['Diff'] = df['Value'].diff()

# Function to format the Y-axis values
def y_format(value, tick_number):
  return f'{int(round(value)):,}'

# Create the plot
plt.figure(figsize=(12, 6))

# Plot the time series, SMA, and difference
plt.plot(df['Month'], df['Value'], marker='', linestyle='-', color='b',
label='Value')
plt.plot(df['Month'], df['SMA'], marker='', linestyle='-', color='r', label='SMA')
plt.plot(df['Month'], df['Diff'], marker='', linestyle='-', color='g',
label='Diff')

# Set the x-axis limits to start from the first date and end at the maximum date in
the data
plt.xlim(pd.to_datetime('01/01/2016', format='%m/%d/%Y'), df['Month'].max())

# Format the x-axis to show dates as MM/DD/YYYY and set the locator for a fixed
number of ticks
ax = plt.gca()
```

```
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m/%d/%Y'))
ax.xaxis.set_major_locator(MaxNLocator(nbins=12)) # Set the number of ticks to a
reasonable value

# Manually set the rotation of date labels to 45 degrees
plt.xticks(rotation=45)

# Title and labels
plt.title('Monthly Consumption of Coal for Electricity Generation in the United
States from 2016 to 2022')
plt.xlabel('Month')
plt.legend()

# Apply the formatter to the Y-axis
ax.yaxis.set_major_formatter(FuncFormatter(y_format))

plt.tight_layout() # Adjust layout to ensure labels fit well
plt.show()
```

The resulting output will look like this:



Analyzing Seasonality

Once the time series has been detrended using one of the methods already discussed (or other, more sophisticated methods), what is left over should represent only the **seasonality**, or *seasonal variation*, s_n, and error term, ε_n.

$$z_n = s_n + \varepsilon_n$$

How could we analyze and model s_n? Suppose there were no error term at all ($\varepsilon_n = 0$) and so we are looking at a time series with pure seasonal variation and no other factors. In this case, the component s_n would look like a repeating pattern, such as the pattern shown in Figure 5.17. As mentioned in Components of Time Series Analysis, the time interval that it takes before the pattern completes once and begins to repeat is called the period. In a purely seasonal component, we would expect $s_{n+P} = s_n$, where P is the period, for all values of n.

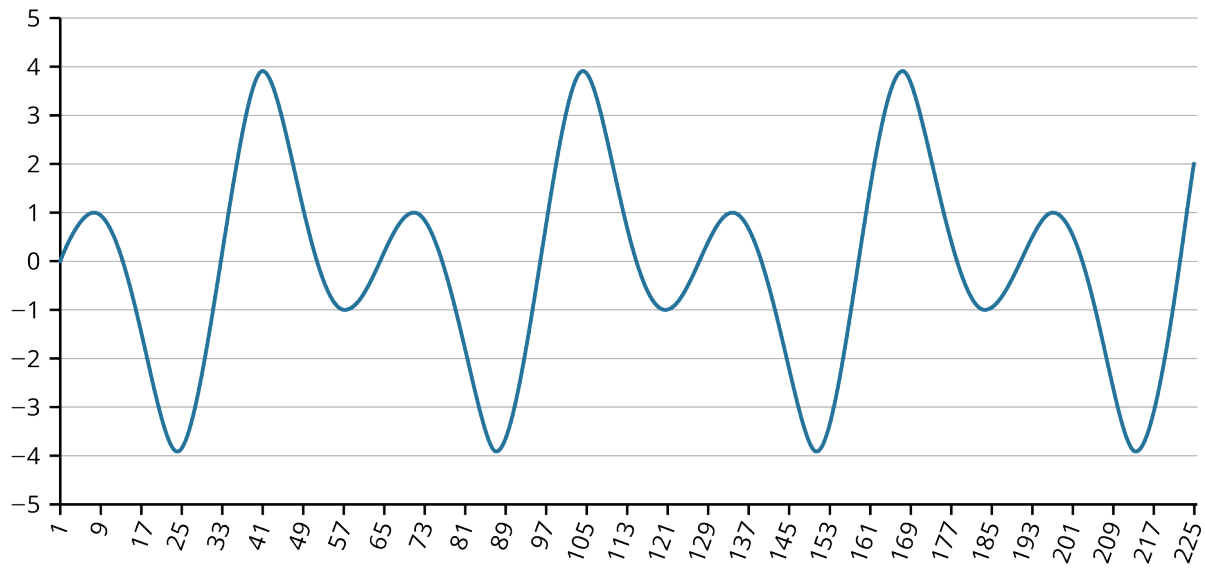


Figure 5.17 Pure Seasonal Variation. The same pattern repeats after 64 time steps, and so the period is 64.

While it is easy to spot the repeating pattern in a time series that does not have noise, the same cannot be said for the general case. There are several tools available for isolating the seasonal component of a detrended time series.

Identifying the Period

First we need to identify the period of the seasonal data. The simplest method is to use prior knowledge about the time series data, along with a visual inspection. Often, a seasonal pattern is exactly that—a pattern that repeats over the course of one year. Thus, the period would be $P = 1$ year (or 12 months, 52 weeks, 365 days, etc., depending on the time steps of your time series). For example, even without detrending the data, it was easy to identify a period of 4 quarters (1 year) in the data in Example 5.2.

If repetition of a seasonal pattern is not apparent from the data, then it may be discovered using standard statistical measures such as autocorrelation. **Autocorrelation** is the measure of correlation between a time series and the same time series shifted by some number of time steps. (Correlation and Linear Regression Analysis discussed correlation between two sets of data. Autocorrelation is the same idea applied to the same dataset with a shifted copy of itself.) The size of the shift is called the **lag**.

The **autocorrelation function (ACF)** measures the correlation at each lag starting with 0 or 1. Note, lag 0 always produces an autocorrelation of 1 since the time series is being compared to an exact copy of itself. Thus, we are only interested in the correlation coefficient at positive lags. However, small lags generally show high correlation with the original time series as well, simply because data measured within shorter time periods are generally similar to each other. So, even if the ACF shows relatively large correlation at lags of 1, 2, 3, etc., there may not be a true seasonal component present. The positive lag corresponding to the first significant **peak**, or autocorrelation value that is much larger than the previous values, may indicate a seasonal component with that period.

Later, we will see how to compute ACF of a time series using Python, but first let's explore further topics regarding seasonal components.

Seasonally Adjusted Data

Once you have determined the period of the seasonal component, you can then extract s_n from the time series. As usual, there are a number of different ways to proceed.

Let's assume that you have detrended the time series so that what remains consists of a seasonal component and noise term:

$$z_n = x_n - t_n = s_n + \varepsilon_n$$

Moreover, suppose that the period of s_n is P. Of course, with the noise term still present in the time series, we cannot expect that $s_{n+P} = s_n$ holds, but it should be the case that $s_{n+P} \approx s_n$ for every n. In other words, the data should look approximately the same when shifted by P time steps. Our job now is to isolate one complete period of data that models the seasonal variation throughout the time series. In other words, we want to produce an estimate $\hat{s}_n$ that is exactly periodic ($\hat{s}_{n+P} = \hat{s}_n$, for all n) and models the underlying seasonal component as closely as possible.

For any given index $n < P$, we expect the data points $\left(z_n, z_{n+P}, z_{n+2P}, z_{n+3P}, \ldots\right)$ to hover around the same value or, in statistical language, to have rather low variance (much lower than the time series as a whole, that is). So it seems reasonable to use the mean of this slice of data to define the value of $\hat{s}_n$. For simplicity, assume that the time series has exactly $M \cdot P$ data points (if not, then the data can be padded in an appropriate way). Then the estimate for the seasonal component would be defined for $1 \leq n < P$ by the formula

$$\hat{s}_n = \frac{1}{M}\left(z_n + z_{n+P} + z_{n+2P} + z_{n+3P} + \ldots + z_{n+(M-1)P}\right)$$

As usual, the formula looks more intimidating than the concept that it's based upon. As previously mentioned, this represents nothing more than the average of all the values of the time series that occur every P time steps.

Once $\hat{s}_n$ is computed in the range $1 \leq n < P$, extend it by repeating the same values in each period until you reach the last index of the original series ($n = MP$). (Note: The noise term may now be estimated as the series of residuals, $\varepsilon_n \approx s_n - \hat{s}_n$, which will occupy our interest in Forecast Evaluation Methods.)

Finally, the estimate $\hat{s}_n$ can then be used in conjunction with the original time series (x_n) to smooth out the seasonal variation. Note that $\hat{s}_n$ essentially measures the fluctuations of observed data x_n above or below the trend. We expect the mean of $\hat{s}_n$ to be close to zero in any window of time. Define the **seasonally adjusted** data as the time series

$$x_n - \hat{s}_n$$

A seasonally adjusted time series is closely related to the underlying trend-cycle curve, t_n, though generally it contains the noise and other random factors that t_n does not generally include.

EXAMPLE 5.6

Problem

The revenue data from Walt's Water Adventures (see Table 5.2) seems to display seasonality of period 4. Using the level as the trend, detrend the data, compute the seasonal component $\hat{s}_n$, and find the seasonally adjusted time series.

Solution

First, we need to detrend the data. The level is the average of all 12 observations. Using a calculator or software, the level is found to be $L = 2{,}275.42$. Subtract $x_n - L$ to detrend the data (see Table 5.4). For example, the first term of the detrended data would be $z_1 = x_1 - L = 1{,}799 - 2{,}275.42 = -476.42$.

There are a total of 12 observations, so there will be $M = 12/4 = 3$ periods of size $P = 4$. Using the values of z_n in the formula for $\hat{s}_n$, we find:

$$\begin{aligned}
\hat{s}_1 &= \tfrac{1}{3}(z_1 + z_5 + z_9) = \tfrac{1}{3}((-476.42) + 8.58 + 428.58) = -13.08 \\
\hat{s}_2 &= \tfrac{1}{3}(z_2 + z_6 + z_{10}) = \tfrac{1}{3}(1{,}028.58 + 1{,}183.58 + 1{,}573.58) = 1{,}261.92 \\
\hat{s}_3 &= \tfrac{1}{3}(z_3 + z_7 + z_{11}) = \tfrac{1}{3}((-996.42) + (-311.42) + (-144.42)) = -484.08 \\
\hat{s}_4 &= \tfrac{1}{3}\left(z_4 + z_8 + z_{12}\right) = \tfrac{1}{3}((-1{,}276.42) + (-681.42) + (-336.42)) = -764.75
\end{aligned}$$

Then repeat this block of 4 values as needed to obtain the entire seasonal component. Finally, the seasonally adjusted time series is the difference between the observed data and seasonal component (see Table 5.4).

k	Quarter	Revenues	Detrended z_k	Seas. Comp. $\hat{s}_k$	Seasonally Adj. $x_n - \hat{s}_n$
1	2021 Spring	1799	−476.42	−13.08	1812.08
2	2021 Summer	3304	1028.58	1261.92	2042.08
3	2021 Fall	1279	−996.42	−484.08	1763.08
4	2021 Winter	999	−1276.42	−764.75	1763.75
5	2022 Spring	2284	8.58	−13.08	2297.08
6	2022 Summer	3459	1183.58	1261.92	2197.08
7	2022 Fall	1964	−311.42	−484.08	2448.08
8	2022 Winter	1594	−681.42	−764.75	2358.75
9	2023 Spring	2704	428.58	−13.08	2717.08
10	2023 Summer	3849	1573.58	1261.92	2587.08
11	2023 Fall	2131	−144.42	−484.08	2615.08
12	2023 Winter	1939	−336.42	−764.75	2703.75

Table 5.4 Quarterly Revenues for Kayak and Canoe Rentals (all columns in $). Columns for detrended data, seasonal component estimate, and seasonally adjusted data are included.

Figure 5.18 shows the data along with the seasonal component and seasonally adjusted data.

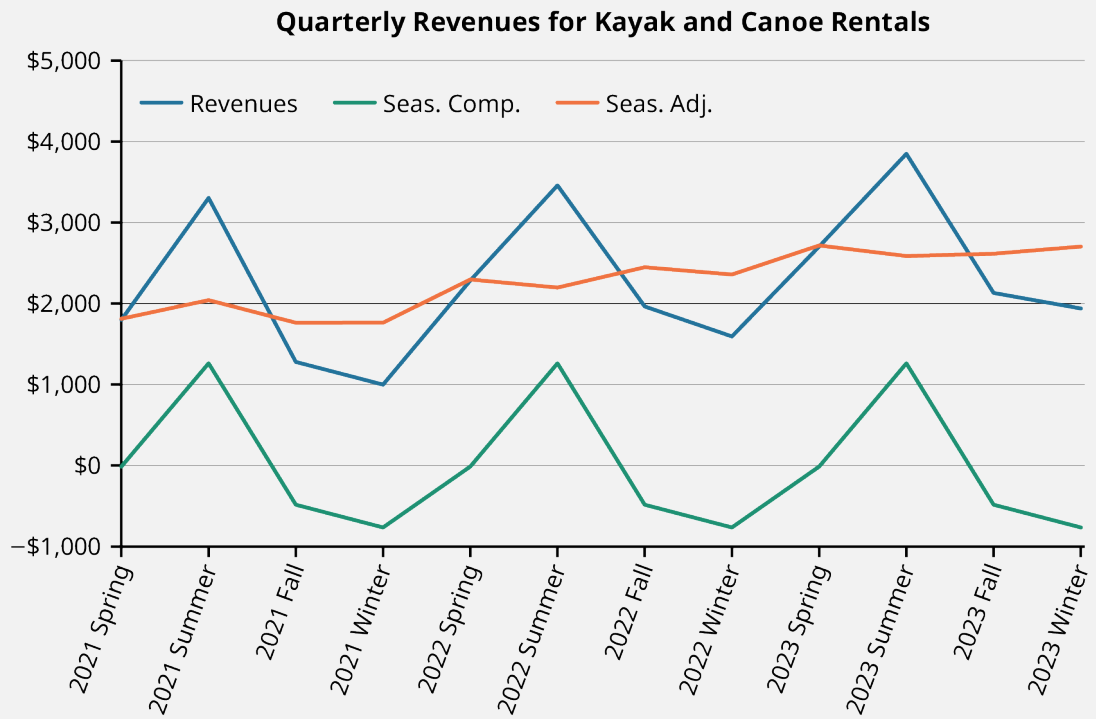


Figure 5.18 Quarterly Revenues for Kayak and Canoe Rentals. Seasonal component and seasonally adjusted time series are included on the graph.

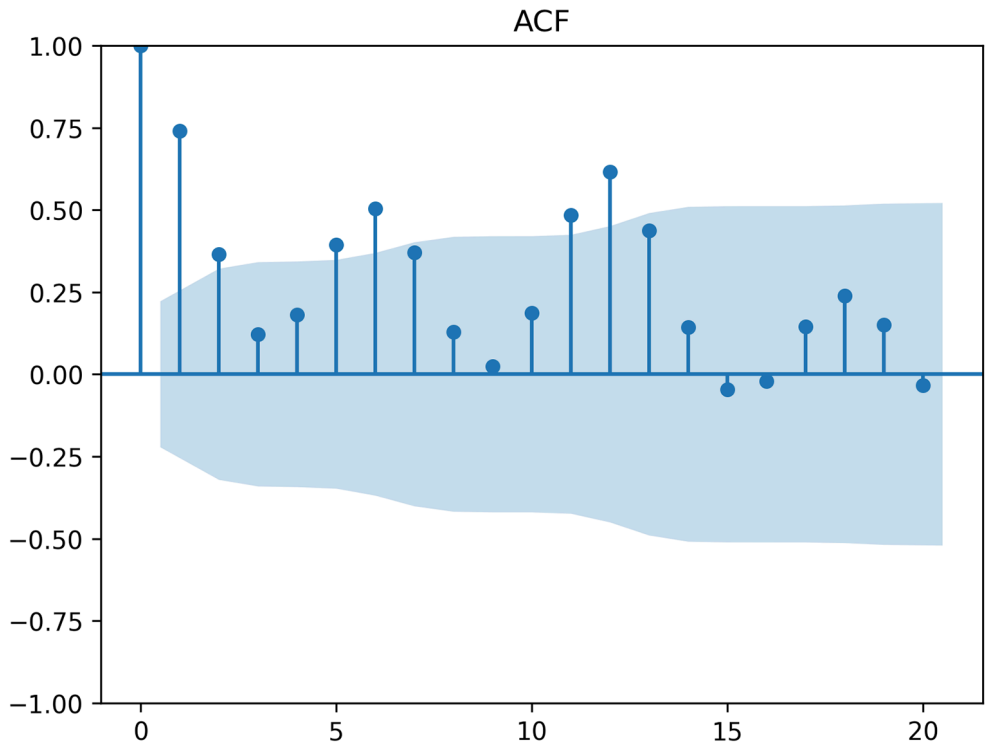
Decomposing a Time Series into Components Using Python

There is a powerful Python library known as statsmodels (https://openstax.org/r/statsmodels) that collects many useful statistical functions together. We will use the `STL` module to decompose the time series MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j) into its components. First, we plot an ACF (autocorrelation function) to determine the best value for periodicity.

PYTHON CODE

```
# Import libraries
from statsmodels.graphics.tsaplots import plot_acf

# Read the Excel file into a Pandas Dataframe
df = pd.read_excel('MonthlyCoalConsumption.xlsx')

# Plot the ACF
plot_acf(df['Value'], lags=20, title='ACF');
```

The resulting output will look like this:

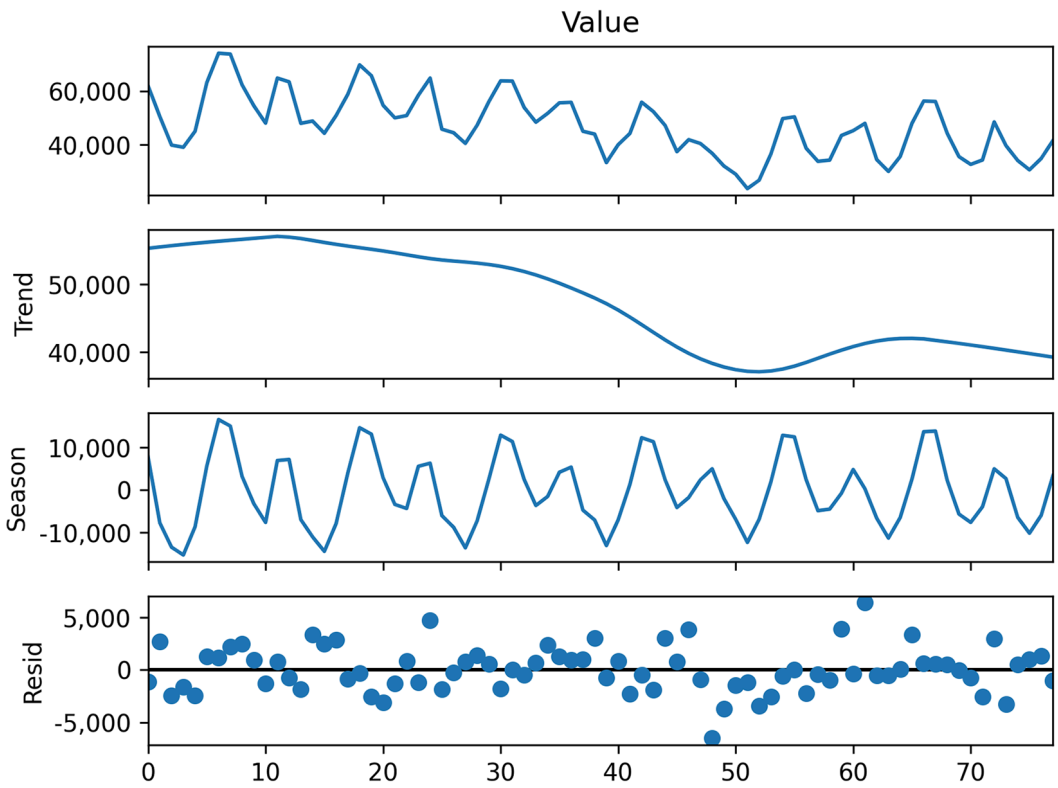


Points that are within the blue shaded area correspond to lags at which the autocorrelation is statistically insignificant. The lags that show the most significance are at 0, 1, 6, and 12. $\text{Lag} = 0$ always gives 100% autocorrelation because we are comparing the unshifted series to itself. At $\text{lag} = 1$, a high autocorrelation corresponds to the fact that consecutive data points are relatively close to each other (which is also to be expected in most datasets). Of special note are lags at 6 and 12, corresponding to a possible seasonal period of 6 or 12 in the data. We will use period 12 in our STL decomposition.

PYTHON CODE

```
# Import libraries
from statsmodels.tsa.seasonal import STL
import matplotlib.pyplot as plt ## for data visualization

# Perform seasonal decomposition using STL
stl = STL(df['Value'], period=12)
result = stl.fit()

# Plot components and format the y-axis labels
fig = result.plot()
for ax in fig.axes:
  ax.yaxis.set_major_formatter(FuncFormatter(y_format))

plt.show()
```

The resulting output will look like this:

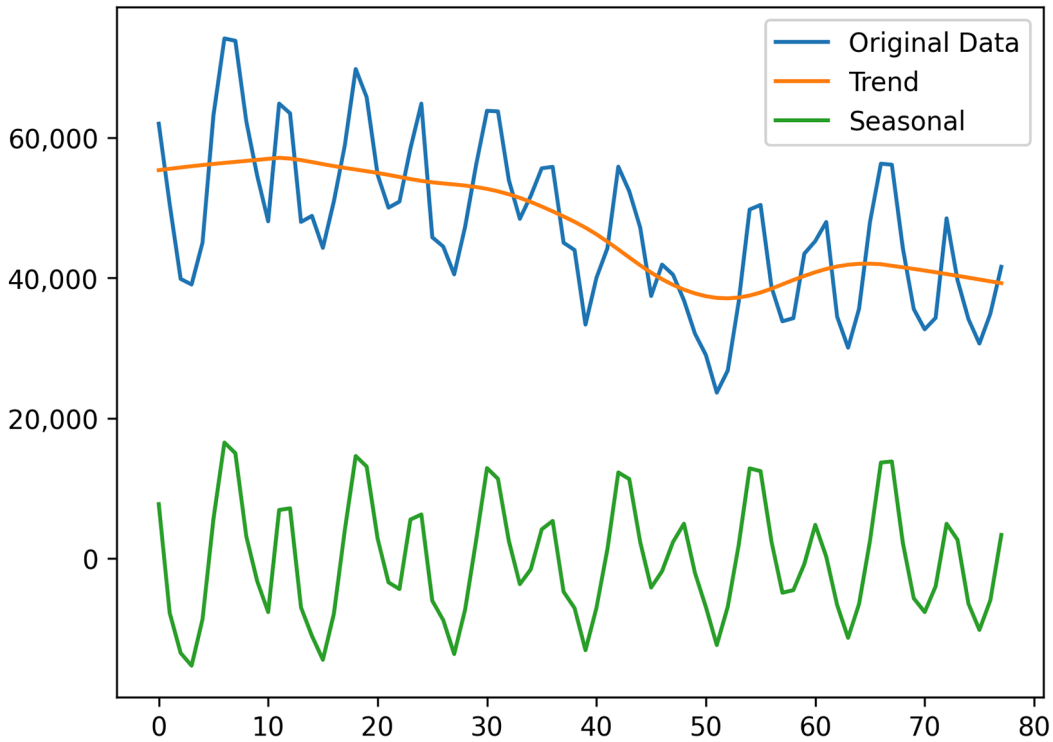


The trend, seasonal variation, and residual (noise) components are all graphed in separate plots. To see them all on the same set of axes, we can use matplotlib (noise omitted).

PYTHON CODE

```
import matplotlib.pyplot as plt ## for data visualization

# Extract the trend and seasonal components
trend = result.trend
seasonal = result.seasonal

# Plot the original data, trend, and seasonally adjusted data
plt.plot(df['Value'], label='Original Data')
plt.plot(trend, label='Trend')
plt.plot(seasonal, label='Seasonal')
plt.legend()

# Apply the formatter to the Y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(y_format))
plt.show()
```

The resulting output will look like this:



Weighted Moving Averages and Exponential Moving Averages

Simple moving averages were discussed in detail at the beginning of this section. Now we expand the discussion to include more sophisticated weighted moving averages and their uses in smoothing data and forecasting. Recall from our discussion in Time Series Forecasting Methods that **weighted moving averages** are moving averages in which terms are given *weights* according to some formula or rule.

Suppose that a time series (x_n) has no significant seasonal component (or the seasonal component has been removed so that remaining series is seasonally adjusted). Thus, the series has only trend-cycle and noise components:

$$x_n = t_n + \varepsilon_n$$

As already discussed, a simple moving average (SMA) has the effect of smoothing out the data and providing an easy method for predicting the next term of the sequence. For reference, here is the SMA prediction model, with a window of T terms:

$$\hat{x}_{n+1} = \frac{x_n + x_{n-1} + x_{n-2} + \ldots + x_{n-T+1}}{T}$$

This model may be regarded as a weighted moving average model (WMA) in which the weights of the most recent T terms are all equal to $\frac{1}{T}$, while all other weights are set to 0. For example, in an SMA model with $T = 3$, the weights would be $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0, \ldots$

$$\hat{x}_{n+1} = \frac{x_n + x_{n-1} + x_{n-2}}{3} = \frac{1}{3}x_n + \frac{1}{3}x_{n-1} + \frac{1}{3}x_{n-2} + 0x_{n-3} + 0x_{n-4} + \ldots$$

Even the naïve model, $\hat{x}_{n+1} = x_n$, counts as a WMA, with weights 1, 0, 0, ...

In principle, a weighted moving average model may be built using any coefficients, giving a general form for any WMA:

$$\hat{x}_{n+1} = a_n x_n + a_{n-1} x_{n-1} + a_{n-2} x_{n-2} + a_{n-3} x_{n-3} + \ldots$$

Of course, if we want the model to do a good job in forecasting new values of the time series, then the coefficients must be chosen wisely. Typically, the weights must be **normalized**, which means that they sum to 1.

$$a_n + a_{n-1} + a_{n-2} + a_{n-3} + \ldots = 1$$

However, there is no hard-and-fast requirement that the sum must be equal to 1.

Rather than giving the same weight to some number of previous terms as the SMA model does, you might choose to give greater weight to more recent terms. For example, one might use powers of $\frac{1}{2}$ as weights.

$$\hat{x}_{n+1} = \frac{1}{2} x_n + \frac{1}{4} x_{n-1} + \frac{1}{8} x_{n-2} + \frac{1}{16} x_{n-3} + \ldots$$

This method allows the most recent term to dominate the sum, and each successive term contributes half as much in turn. (Note: The sequence $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots$ *would* be a normalized sequence if infinitely many terms were included, but even a small finite number of terms, like 5 or so, adds up to a sum that is pretty close to 1.)

This is an example of an **exponential moving average**, or **EMA**. The most general formula for the EMA model is shown. (Note that the previous example corresponds to an EMA model with $\alpha = \frac{1}{2}$).

$$\hat{x}_{n+1} = \alpha x_n + \alpha\left(1 - \alpha\right) x_{n-1} + \alpha(1-\alpha)^2 x_{n-2} + \alpha(1-\alpha)^3 x_{n-3} + \ldots$$

EXAMPLE 5.7

Problem

Use EMA models with $\alpha = \frac{1}{2}$ (equivalently, 0.5) and $\alpha = 0.7$ for the S&P Index data from Table 5.1 to estimate the value of the S&P Index at the end of year 2023.

Solution

First, for $\alpha = \frac{1}{2}$, the model is:

$$\hat{x}_{n+1} = \frac{1}{2} x_n + \frac{1}{4} x_{n-1} + \frac{1}{8} x_{n-2} + \frac{1}{16} x_{n-3} + \ldots$$

Now plug in the data (remember to use the most recent data point first and work your way backward). Eight terms were used, as the terms beyond that point are rather close to 0. Table 5.2 records the results.

For $\alpha = 0.7$, use the formula for weights,

$\alpha = 0.7$, $\alpha(1-\alpha) = 0.7(1-0.7) = 0.7(0.3) = 0.21$, $\alpha(1-\alpha)^2 = 0.7(0.3)^2 = 0.063$, etc. This time, the coefficients decrease quicker than in the case $\alpha = \frac{1}{2}$, so only 5 terms are significant. Table 5.5 shows the results.

Year	S&P Index at Year-End	Exponentially Weighted $\alpha = \frac{1}{2}$	Exponentially Weighted $\alpha = 0.7$
2013	1848.36	≈ 0	≈ 0
2014	2058.9	≈ 0	≈ 0
2015	2043.94	$\frac{1}{512} \cdot 2043.94 = 3.99$	≈ 0
2016	2238.83	$\frac{1}{256} \cdot 2238.83 = 8.75$	≈ 0

Table 5.5 EMA Models (source: adapted from https://www.nasdaq.com/market-activity/index/spx/historical)

Year	S&P Index at Year-End	Exponentially Weighted $\alpha = \frac{1}{2}$	Exponentially Weighted $\alpha = 0.7$
2017	2673.61	$\frac{1}{128} \cdot 2673.61 = 20.89$	≈ 0
2018	2506.85	$\frac{1}{64} \cdot 2506.85 = 39.17$	$0.00567 \cdot 2506.9 = 4.26$
2019	3230.78	$\frac{1}{32} \cdot 3230.78 = 100.96$	$0.00567 \cdot 3230.78 = 18.32$
2020	3756.07	$\frac{1}{16} \cdot 3756.07 = 234.75$	$0.0189 \cdot 3756.07 = 70.99$
2021	4766.18	$\frac{1}{8} \cdot 4766.18 = 595.77$	$0.063 \cdot 4766.18 = 300.27$
2022	3839.5	$\frac{1}{4} \cdot 3839.5 = 959.88$	$0.21 \cdot 3839.5 = 806.30$
2023	4769.83	$\frac{1}{2} \cdot 4769.83 = 2384.92$	$0.7 \cdot 4769.83 = 3338.88$
2024		SUM $= 4349$	SUM $= 4541$

Table 5.5 EMA Models (source: adapted from https://www.nasdaq.com/market-activity/index/spx/historical)

Therefore, we find estimates of 4,349 and 4,541 for the S&P value at the end of 2024, by EMA models with $\alpha = \frac{1}{2}$ and $\alpha = 0.7$, respectively. Note that larger values of α emphasize the most recent data, while smaller values allow more of the past data points to influence the average. If the time series trend experiences more variability (due to external factors that we have no control over), then an EMA with a relatively large α -value would be more appropriate than one with a small α-value.

EMA models may also serve as smoothing techniques, isolating the trend-cycle component. In this context, it's better to use an equivalent form of the EMA model:

$$\hat{x}_{n+1} = \alpha x_n + (1 - \alpha)\,\hat{x}_n$$

In this form, we build a series of estimates $(\hat{x}_n)$ recursively by incorporating the terms of the original time series one at a time. The idea is that the next estimate, $\hat{x}_{n+1}$, is found by weighted average of the current data point, x_n, and the current estimate, $\hat{x}_n$. Note, we may set $\hat{x}_1 = x_1$ since we need to begin the process somewhere. It can be shown mathematically that this procedure is essentially the same as working out the sum of exponentially weight terms directly and is much more efficient.

EXAMPLE 5.8

Problem

Use the recursive EMA formula with $\alpha = 0.75$ to smooth the S&P Index time series (Table 5.1), and then use the EMA model to estimate the value of the S&P Index at the end of the 2024. Plot the EMA together with the original time series.

Solution

With $\alpha = 0.75$, the recursive EMA formula becomes: $\hat{x}_{n+1} = 0.75x_n + 0.25\hat{x}_n$. Table 5.6 shows the results of the computation.

Year	S&P Index at Year-End	EMA Estimate
2013	$x_1 = 1848.36$	$\hat{x}_1 = x_1 = 1848.36$
2014	$x_2 = 2058.9$	$\hat{x}_2 = 0.75x_1 + 0.25\hat{x}_1 = 0.75(1848.36) + 0.25(1848.36) = 1848.36$
2015	$x_3 = 2043.94$	$\hat{x}_3 = 0.75x_2 + 0.25\hat{x}_2 = 0.75(2058.9) + 0.25(1848.36) = 2006.27$
2016	$x_4 = 2238.83$	$\hat{x}_4 = 0.75x_3 + 0.25\hat{x}_3 = 0.75(2043.94) + 0.25(2006.27) = 2034.52$
2017	$x_5 = 2673.61$	$\hat{x}_5 = 0.75x_4 + 0.25\hat{x}_4 = 0.75(2238.83) + 0.25(2034.52) = 2187.75$
2018	$x_6 = 2506.85$	$\hat{x}_6 = 0.75x_5 + 0.25\hat{x}_5 = 0.75(2673.61) + 0.25(2187.75) = 2552.15$
2019	$x_7 = 3230.78$	$\hat{x}_7 = 0.75x_6 + 0.25\hat{x}_6 = 0.75(2506.85) + 0.25(2552.15) = 2518.17$
2020	$x_8 = 3756.07$	$\hat{x}_8 = 0.75x_7 + 0.25\hat{x}_7 = 0.75(3230.78) + 0.25(2518.17) = 3052.63$
2021	$x_9 = 4766.18$	$\hat{x}_9 = 0.75x_8 + 0.25\hat{x}_8 = 0.75(3756.07) + 0.25(3052.63) = 3580.21$
2022	$x_{10} = 3839.5$	$\hat{x}_{10} = 0.75x_9 + 0.25\hat{x}_9 = 0.75(4766.18) + 0.25(3580.21) = 4469.69$
2023	$x_{11} = 4769.83$	$\hat{x}_{11} = 0.75x_{10} + 0.25\hat{x}_{10} = 0.75(3839.5) + 0.25(4469.69) = 3997.05$
2024	N/A	$\hat{x}_{12} = 0.75x_{11} + 0.25\hat{x}_{11} = 0.75(4769.83) + 0.25)3997.05) = 4576.63$

Table 5.6 EMA Smoothing for the S&P Index Time Series (source: adapted from https://www.nasdaq.com/market-activity/index/spx/historical)

According to this model, the S&P Index will be about 4577 at the end of 2024. Figure 5.19 shows the time series together with the exponential smoothing model.

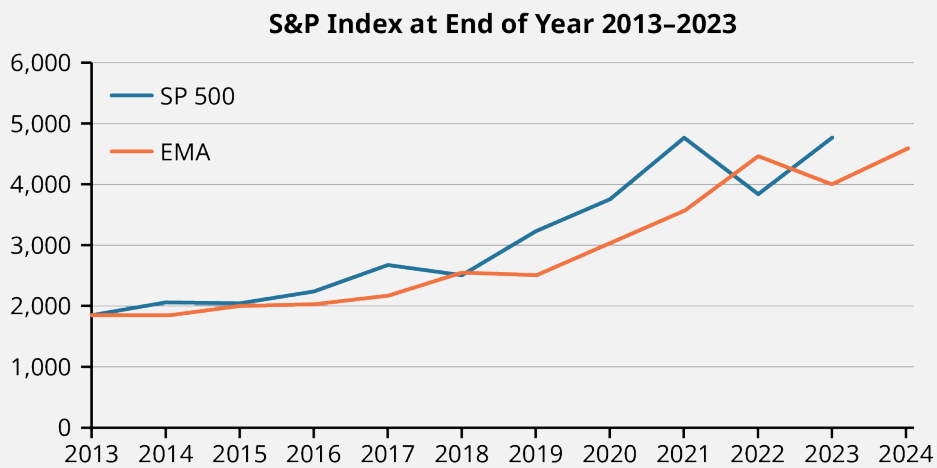


Figure 5.19 Exponential Smoothing Model for the S&P Index (data source: adapted from S&P 500 [SPX] Historical Data)

Note, EMA smoothing can be done very easily in Excel. Simply type in the formulas =B2 into cell C2 and =0.75*B2+0.25*C2 into cell C3. Then drag cell C3 down to the end of the data to compute the EMA. You can also do EMA smoothing in Python using the `ewm()` function applied to a dataset. Note, the "span" value should be set to $\frac{2}{\alpha} - 1$, as shown in this code.

PYTHON CODE

```python
# Import libraries
import pandas as pd ## for dataset management

# Creating a dictionary with the given data
data = {
  'Year': [2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023],
  'SP500': [1848.36, 2058.9, 2043.94, 2238.83, 2673.61, 2506.85, 3230.78,
3756.07, 4766.18, 3839.5, 4769.83]
}
# Creating DataFrame
df = pd.DataFrame(data)

alpha = 0.75
df['ema']=df['SP500'].ewm(span=2/alpha-1, adjust=False).mean()

print(df['ema'])
```

The resulting output will look like this:

```
0   1848.360000
1   2006.265000
2   2034.521250
3   2187.752812
4   2552.145703
5   2518.173926
6   3052.628481
7   3580.209620
8   4469.687405
9   3997.046851
10  4576.634213
Name: ema, dtype: float64
```

Autoregressive Integrated Moving Average (ARIMA)

In this section, we develop a very powerful tool that combines multiple models together and is able to capture trend and seasonality. The method is called **autoregressive integrated moving average**, or **ARIMA** for short, and it consists of three main components:

1. **Autoregressive (AR) model**. This component captures patterns in the time series that are repeated, including seasonal components as well as correlation with recent values.
2. **Integrative (I) component**. This component represents the differencing operation discussed in the section on detrending a time series.
3. **Moving average (MA) model.** Despite the name, this component is not the same as the moving average discussed previously, although there is a mathematical connection. Instead of focusing on averages of existing data points, MA models incorporate the influence of past forecast errors to predict the next value of the time series.

Before getting into the specifics of each component of ARIMA, we need to talk about the term **stationary**,

which refers to a time series in which the variance is relatively constant over time, an overall upward or downward trend cannot be found, and no seasonal patterns exist. White noise is an example of a stationary time series; however, the term stationary is a bit more general, as it includes the possibility of a cyclic component in the data. Recall, a cyclic component (despite its name) is a longer-term rise or fall in the values of a time series that may occur from time to time, but the occurrences are unpredictable.

The AR and MA models work best on stationary time series. The integrative component is the tool that can detrend the data, transforming a nonstationary time series into a stationary one so that AR and/or MA may be used.

Autoregressive Models

The autoregressive models take into account a fixed number of previous values of the time series in order to predict the next value. In this way, AR models are much like WMA models previously discussed, in the sense that the next term of the time series depends on a sum of weighted terms of previous values; however, in the AR model, the terms used are the terms of the model itself.

Suppose we want to model a time series, (x_n), using an AR model. First, we should pick a value p, called the order of the model, which specifies how many previous terms to include in the model. Then we may call the model an $\mathrm{AR}\,(p)$ model. If there is a strong seasonal component of period$p$, then it makes sense to choose that value as the order.

An $\mathrm{AR}\,(p)$ model is a recursive formula of the form

$$\mathrm{AR}\,(p)_n = w_1 y_{n-1} + w_2 y_{n-2} + w_3 y_{n-3} + \ldots + w_p y_{n-p} + \varepsilon_n$$

Here, the parameters or weights, $w_1, w_2, w_3, \ldots w_p$, are constants that must be chosen so that the values of the time series (y_n) fit the known data as closely as possible (i.e., $y_n \approx x_n$ for all known observations). There are also certain restrictions on the possible values of each weight c, but we do not need to delve into those specifics in this text as we will rely on software to do the work of finding the parameters for us.

Integrative Component

The "I" in ARIMA stands for *integrative*, which is closely related to the word *integral* in calculus. Now if you have seen some calculus, you might recall that the integral is just the opposite of a *derivative*. The derivative, in turn, is closely related to the idea of differencing in a time series. Of course, you don't have to know calculus to understand the idea of an integrative model.

Rather than producing another component of the time series, the integrative component of ARIMA represents the procedure of taking differences, $\Delta^d\,(x_n)$ (up to some specific order d), until the time series becomes stationary. Often, only the first-order difference is needed, and you can check for stationarity visually. A more sophisticated method, called the **Augmented Dickey-Fuller (ADF) test**, sets up a test statistic that measures the presence of non-stationarity in a time series. Using ADF, a *p*-value is computed. If the *p*-value is below a set threshold (such as $p < 0.01$), then the null hypothesis of non-stationarity can be rejected. Next, we will show how to run the ADF test on a time series and its differences $\Delta^d\,(x_n)$ to find out the best value for d in ARIMA.

Then the AR and MA models can be produced on the resulting stationary time series. Integrating (which is the opposite of differencing) builds the model back up so that it can ultimately be used to predict values of the original time series (x_n).

Moving Average Models

An MA model of order q, or $\mathrm{MA}\,(q)$, takes the form

$$\mathrm{MA}\,(q)_n = L + \varepsilon_n + b_1 \varepsilon_{n-1} + b_2 \varepsilon_{n-2} + b_3 \varepsilon_{n-3} + \ldots + b_q \varepsilon_{n-q}$$

Here, the ε_n terms are residuals (forecast errors), and L is the level of the time series (mean value of all the terms). Again, there are a multitude of parameters that need to be chosen in order for the model to make

sense. We will rely on software to handle this in general.

ARIMA Model in Python

The full ARIMA model, $\text{ARIMA}\ (p, d, q)$, incorporates all three components—autoregressive of order p, integrative with d-order differencing, and moving average of order q. For the time series MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j), it turns out that the second-order difference is stationary, as shown by a series of ADF tests.

PYTHON CODE

```python
# Import libraries
import pandas as pd ## for dataset management
# Import adfuller from the statsmodels library
from statsmodels.tsa.stattools import adfuller

# Read data
df = pd.read_excel('MonthlyCoalConsumption.xlsx')

print('d = 0')
result = adfuller(df['Value'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])

## Get the first difference
print('\n d = 1')
df1 = df['Value'].diff().dropna()
result = adfuller(df1)
print('ADF Statistic:', result[0])
print('p-value:', result[1])

## Get the second difference
print('\n d = 2')
df2 = df1.diff().dropna()
result = adfuller(df2)
print('ADF Statistic:', result[0])
print('p-value:', result[1])
```

The resulting output will look like this:

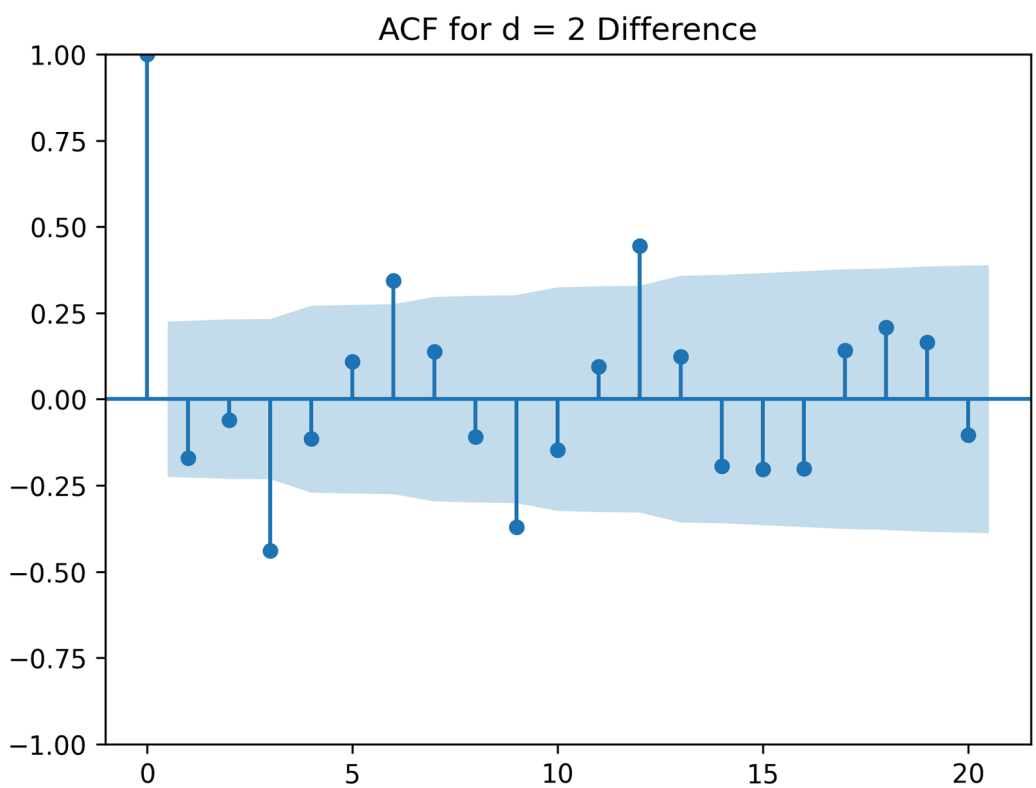
```
d = 0
ADF Statistic: -1.174289795062662
p-value: 0.6845566772896323

d = 1
ADF Statistic: -1.7838215415905252
p-value: 0.38852291349101137
```

```
d = 2
ADF Statistic: -8.578323839498081
p-value: 7.852009937900099e-14
```

Here, a *p*-value of 7.852×10^{-14} indicates that the second-order difference is stationary, so we will set $d = 2$ in the ARIMA model. An ACF plot of the second difference can be used to determine p.

PYTHON CODE

```python
# Import libraries
from statsmodels.graphics.tsaplots import plot_acf

# Plot the ACF of the second difference
plot_acf(df2, lags=20, title='ACF for d = 2 Difference');
```

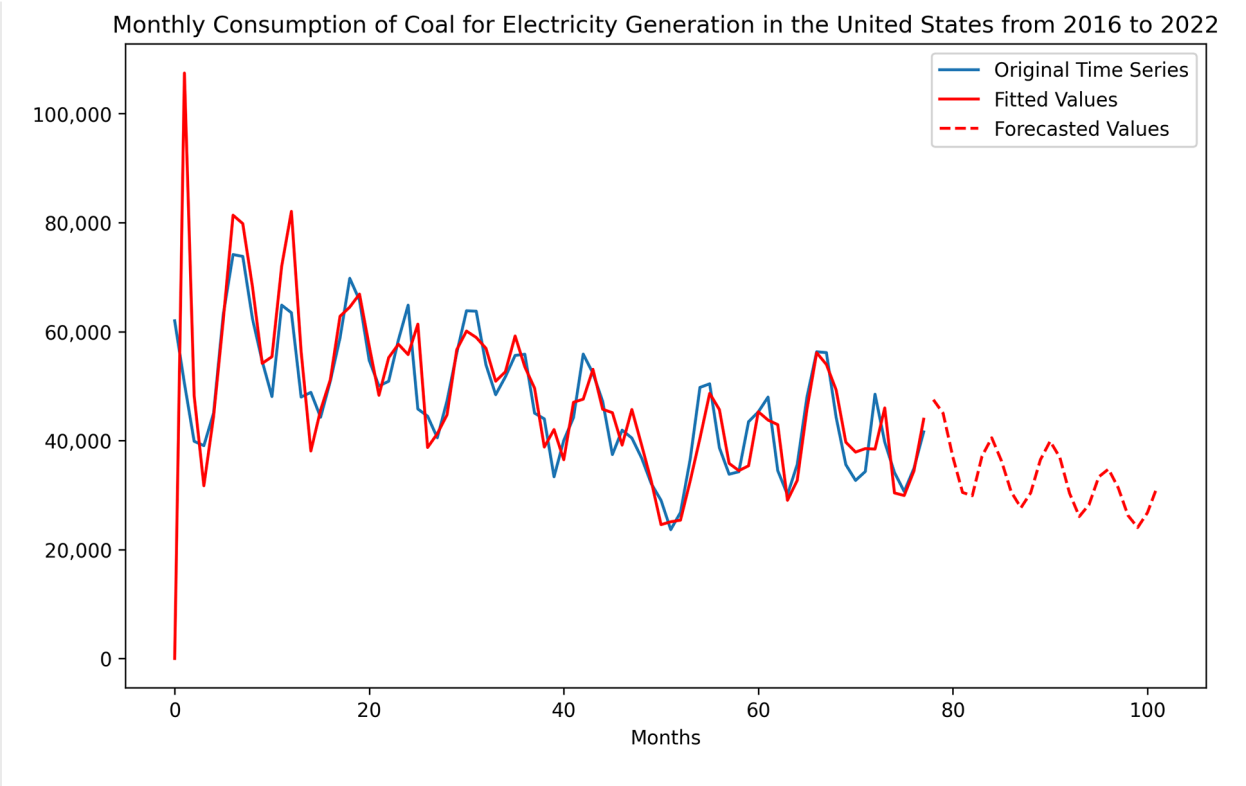


We see a peak at lag 12, corresponding to the expected yearly seasonal variation, so we will set $p = 12$ in the ARIMA model. For this time series, positive values of q did not change the accuracy of the model at all, so we set $q = 0$ (MA model component unnecessary). The model was used to forecast 24 time steps (2 years) into the future. We are using the Python library statsmodels.tsa.arima.model (https://openstax.org/r/statsmodel) to create the ARIMA model.

PYTHON CODE

```python
# Import ARIMA model from statsmodels.tsa library
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt ## for plotting graphs
from matplotlib.ticker import MaxNLocator ## for graph formatting
from matplotlib.ticker import FuncFormatter ## for formatting y-axis

# Fit an ARIMA model
p = 12; d = 2; q = 0 # ARIMA component orders
model = ARIMA(df['Value'], order=(p, d, q))
results = model.fit()

# Make forecasts
fc = results.forecast(steps=24)

# Plot the results
plt.figure(figsize=(10, 6))

# Plot original time series
plt.plot(df['Value'], label='Original Time Series')

# Plot fitted and forecasted values
plt.plot(results.fittedvalues, color='red', label='Fitted Values')
plt.plot(fc, color='red', linestyle='dashed', label='Forecasted Values') ##
Forecasted values

# Set labels and legend
plt.xlabel('Months')
plt.title('Monthly Consumption of Coal for Electricity Generation in the United
States from 2016 to 2022')
plt.legend()

# Function to format the Y-axis values
def y_format(value, tick_number):
  return f'{value:,.0f}'

# Apply the formatter to the Y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(y_format))
plt.show()
```

The resulting output will look like this:



Here, the blue curve is the original time series, and the red curve represents the ARIMA model, with forecasted values indicated by the dashed part of the curve. Although there is considerable error near the beginning of the time period, the model seems to fit the known values of the data fairly well, and so we expect that the future forecasts are reasonable.

5.4 Forecast Evaluation Methods

Learning Outcomes

By the end of this section, you should be able to:

- 5.4.1 Explain the nature of error in forecasting a time series.
- 5.4.2 Compute common error measures for time series models.
- 5.4.3 Produce prediction intervals in a forecasting example.

A time series forecast essentially predicts the *middle of the road* values for future terms of the series. For the purposes of prediction, the model considers only the trend and any cyclical or seasonal variation that could be detected within the known data. Although noise and random variation certainly do influence future values of the time series, their precise effects cannot be predicted (by their very nature). Thus, a forecasted value is the best guess of the model in the absence of an error term. On the other hand, error does affect how certain we can be of the model's forecasts. In this section, we focus on quantifying the error in forecasting using statistical tools such as prediction intervals.

Forecasting Error

Suppose that you have a model $(\hat{x}_n)$ for a given time series (x_n). Recall from Components of Time Series Analysis that the residuals quantify the error between the time series the model and serve as an estimate of the noise or random variation.

$$\varepsilon_n = x_n - \hat{x}_n$$

The better the fit of the model to the observed data, the smaller the values of ε_n will be. However, even very

good models can turn out to be poor predictors of future values of the time series. Ironically, the better a model does at fitting to known data, the worse it may be at predicting future data. There is a danger of *overfitting*. (This term is defined in detail in Decision-Making Using Machine Learning Basics, but for now we do not need to go deeply into this topic.) A common technique to avoid overfitting is to build the model using only a portion of the known data, and then the model's accuracy can be tested on the remaining data that was held out.

To discuss the accuracy of a model, we should ask the complementary question: *How far away are the predictions from the true values?* In other words, we should try to quantify the total error. There are several **measures of error**, or **measures of fit**, the metrics used to assess how well a model's predictions align with the observed data. We will only discuss a handful of them here that are most useful for time series. In each formula, x_i refers to the ith term of the time series, and ε_i is the error between the actual i^{th} term and the predicted i^{th} term of the series.

a. **Mean absolute error (MAE)**: $\frac{1}{n}\sum_{i=1}^{n}|\varepsilon_i|$. A measure of the average magnitude of errors.
b. **Root mean squared error (RMSE)**: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\varepsilon_i{}^2\right)}$. A measure of the standard deviation of errors, penalizing larger errors more heavily than MAE does.
c. **Mean absolute percentage error (MAPE)**: $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{\varepsilon_i}{x_i}\right|$. A measure of the average relative errors—that is, a percentage between the predicted values and the actual values (on average).
d. **Symmetric mean absolute percentage error (sMAPE)**: $\frac{1}{n}\sum_{i=1}^{n}\frac{2|\varepsilon_i|}{|x_i|+|\hat{x}_i|}$. Similar to MAPE, a measure of the average relative errors, but scaled so that errors are measured in relation to both actual values and predicted values.

Of these error measures, MAE and RMSE are **scale-dependent**, meaning that the error is in direct proportion to the data itself. In other words, if all terms of the time series and the model were scaled by a factor of k, then the MAE and RMSE would both be multiplied by k as well. On the other hand, MAPE and sMAPE are not scale-dependent. These measures of errors are often expressed as percentages (by multiplying the result of the formula by 100%). However, neither MAPE nor sMAPE should be used for data that is measured on a scale containing 0 and negative numbers. For example, it would not be wise to use MAPE or sMAPE as a measure of error for a time series model of Celsius temperature readings. For all of these measures of error, lower values indicate less error and hence more accuracy of the model. This is useful when comparing two or more models.

EXAMPLE 5.9

Problem

Compute the MAE, RMSE, MAPE, and sMAPE for the EMA smoothing model for the S&P Index time series, shown in Table 5.3.

Solution

The first step is to find the residuals, ε_n, and take their absolute values. For RMSE, we will need the squared residuals, included in Table 5.7. We also include the terms $\left|\frac{\varepsilon_i}{x_i}\right|$ and $\frac{2|\varepsilon_i|}{|x_i|+|\hat{x}_i|}$, which are used for computing MAPE and sMAPE, respectively. Notice all formulas are an average of some kind, but RMSE requires an additional step of taking a square root.

Year	S&P Index at Year-End	EMA Estimate	Abs. Residuals $\lvert\varepsilon_n\rvert = \lvert x_n - \widehat{x}_n\rvert$	$(\varepsilon_n)^2$	$\lvert\frac{\varepsilon_i}{x_i}\rvert$	$\frac{2\lvert\varepsilon_i\rvert}{\lvert x_i\rvert+\lvert\widehat{x}_i\rvert}$
2013	1848.36	1848.36	0	0	0	0
2014	2058.9	1848.36	210.54	44327.09	0.102	0.108
2015	2043.94	2006.27	37.67	1419.03	0.018	0.019
2016	2238.83	2034.52	204.31	41742.58	0.091	0.096
2017	2673.61	2187.75	485.86	236059.94	0.182	0.2
2018	2506.85	2552.15	45.3	2052.09	0.018	0.018
2019	3230.78	2518.17	712.61	507813.01	0.221	0.248
2020	3756.07	3052.63	703.44	494827.83	0.187	0.207
2021	4766.18	3580.21	1185.97	1406524.84	0.249	0.284
2022	3839.5	4469.69	630.19	397139.44	0.164	0.152
2023	4769.83	3997.05	772.78	597188.93	0.162	0.176
N/A	N/A	**Average:**	**453.52**	**339008.62**	**0.127**	**0.137**

Table 5.7 Data Needed to Compute the MAE, RMSE, MAPE, and SMAPE for EMA Smoothing Model

$\text{MAE} = 453.52$. (Predicted values are [on average] about 450 units away from true values.)

$\text{RMSE} = \sqrt{339008.62} = 582.24$. (The errors between predicted and average values have a standard deviation of about 580 from the ideal error of 0.)

$\text{MAPE} = 0.127, \text{or } 12.7\%$. (Predicted values are [on average] within about 13% of their true values.)

$\text{sMAPE} = 0.137, \text{or } 13.7\%$. (When looking at both predicted values and true values, their differences are on average about 14% from one another.)

These results may be used to compare the EMA estimate to some other estimation/prediction method to find out which one is more accurate. The lower the errors, the more accurate the method.

Prediction Intervals

A forecast is often accompanied by a **prediction interval** giving a range of values the variable could take with some level of probability. For example, if the prediction interval of a forecast is indicated to be 80%, then the prediction interval contains a range of values that should include the actual future value with a probability of 0.8. Prediction intervals were introduced in Analysis of Variance (ANOVA) in the context of linear regression, so we won't get into the details of the mathematics here. The key point is that we want a measure of *margin of error*, E_n (depending on n), such that future observations of the data will be within the interval $\hat{x}_n \pm E_n$ with probability α, where α is a chosen level of confidence.

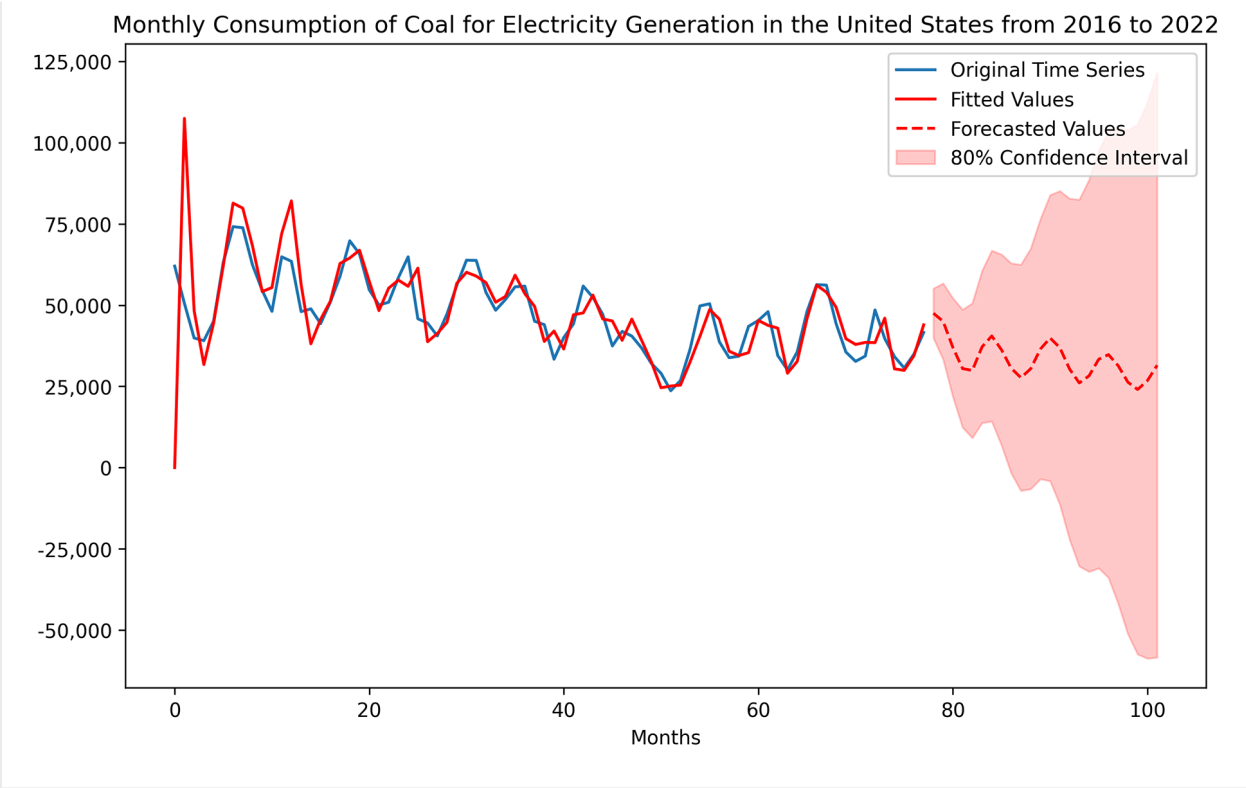
Here, we will demonstrate how to use Python to obtain prediction intervals.

The Python library `statsmodels.tsa.arima.model` (https://openstax.org/r/statsmodel) contains functions for finding confidence intervals as well. Note that the command `get_forecast()` is used here rather than `forecast()`, as the former contains more functionality.

PYTHON CODE

```
### Please run all code from previous section before running this ###

# Set alpha to 0.2 for 80% confidence interval
forecast_steps = 24
forecast_results = results.get_forecast(steps=forecast_steps, alpha=0.2)

# Extract forecast values and confidence intervals
forecast_values = forecast_results.predicted_mean
confidence_intervals = forecast_results.conf_int()

# Plot the results
plt.figure(figsize=(10, 6))

# Plot original time series
plt.plot(df['Value'], label='Original Time Series')

# Plot fitted values
plt.plot(results.fittedvalues, color='red', label='Fitted Values')

# Plot forecasted values with confidence intervals
plt.plot(forecast_values, color='red', linestyle='dashed', label='Forecasted
Values')
plt.fill_between(
  range(len(df), len(df) + forecast_steps),
  confidence_intervals.iloc[:, 0],
  confidence_intervals.iloc[:, 1],
  color='red', alpha=0.2,
  label='80% Confidence Interval'
)

# Set labels and legend
plt.xlabel('Months')
plt.title('Monthly Consumption of Coal for Electricity Generation in the United
States from 2016 to 2022')
plt.legend()

# Apply the formatter to the Y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(y_format))
plt.show()
```

The resulting output will look like this:



The forecast data (dashed curve) is now surrounded by a shaded region. With 80% probability, all future observations should fit into the shaded region. Of course, the further into the future we try to go, the more uncertain our forecasts will be, which is indicated by the wider and wider confidence interval region.

Datasets

Note: The primary datasets referenced in the chapter code may also be [downloaded here (https://openstax.org/r/drive1)](https://openstax.org/r/drive1).

Key Terms

additive decomposition time series decomposition into the sum of its components
augmented Dickey-Fuller (ADF) test statistical test for stationarity of a time series
autocorrelation measure of correlation between a time series and a shifted copy of itself
autoregressive (AR) model model or component of a model that captures how the time series depends on its own previous values
cyclic component large variations in the data that recur over longer time periods than seasonal fluctuations, having no fixed frequency
detrending one of two complementary operations that separate the trend component from a time series
differencing found by taking differences of consecutive terms, that is, $x_{n+1} - x_n$
error extent to which predictions differ from actual observations
exponential moving average (EMA) type of weighted moving average in which the most recent values are given larger weights and the weights decay exponentially the further back in time the values are in the series
forecasting making predictions about future, unknown values of a time series
integrative (I) component component of the ARIMA model that represents the differencing operation
lag number of time steps that a time series is shifted for the purpose of computing autocorrelation
level the mean of all the time series data
mean absolute error (MAE) measure of error; $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\varepsilon_i|$
mean absolute percentage error (MAPE) measure of relative error; $\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} |\frac{\varepsilon_i}{x_i}|$
measures of error (or measures of fit) the metrics used to assess how well a model's predictions align with observed data
multiplicative decomposition time series decomposition into the product of its components
naïve (or flat) forecasting method using only the last observed value of a time series to predicting the next term
noise (or random error) random, unpredictable variations that occur in the time series that cannot be attributed to the trend, cycles, or seasonal components
normalized (weights) weights or coefficients that add up to a total of 1
order (or degree) the number of terms or lags used in a model to describe the time series; for a sequence that is modeled by a polynomial formula, the value of the exponent on the term of highest order of the polynomial formula that models the sequence
peak when computing ACF, a correlation value that is significantly larger than the previous values
period smallest time interval over which a seasonal variation pattern repeats
prediction interval range of values the variable could take with some level of probability in a forecast
residuals the difference between the given time series and the model $(x_n - \hat{x}_n)$ quantifying the error of the model, or noise (random variation) of the time series
root mean squared error (RMSE) measure of error; $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\varepsilon_i{}^2\right)}$
scale-dependent regarding a measure or error that is in direct proportion with the data
seasonal component (seasonality) variation in the data that occurs at fixed time periods
Seasonal-Trend decomposition using LOESS (STL) a powerful tool that decomposes a time series into the trend-cycle, seasonal, and noise components
sequence ordered list of numbers, measurements, or other data
simple moving average (SMA) average (arithmetic mean) of a fixed number of consecutive data points in a time series
stationary characterizing a time series in which the variance is relatively constant over time, an overall upward or downward trend cannot be found, and no seasonal patterns exist
symmetric mean absolute percentage error (sMAPE) measure of relative error;

$$\text{sMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{2|\varepsilon_i|}{|x_i|+|\hat{x}_i|}$$

terms in mathematics, the individual values of a sequence

time series data that has a time component, or an ordered sequence of data points

time series analysis the examination of data points collected at specific time intervals, enabling the identification of trends, patterns, and seasonal variations crucial for making informed predictions and decisions

time series model function, algorithm, or method for finding, approximating, or predicting the values of a given time series

trend long-term direction of time series data in the absence of any other variation

trend curve (or trendline) line or curve that models the trend of a time series

trend-cycle component component of a time series that combines the trend and long-term cycles

volatility displaying significant fluctuations from the mean, typically due to factors that are difficult to analyze or predict

weighted moving average (WMA) moving average in which terms are given weights according to some formula or rule

white noise time series data that has constant mean value close to zero, constant variance, and no correlation from one part of the series to another

window term used to describe a fixed number of consecutive terms of a time series

Group Project

Project A – Smoothing out the Stock Market

Predicting the ups and downs of the stock market is a notoriously difficult task. On the other hand, if anyone could obtain accurate forecasts of future stock prices, then they could make a lot of money by buying up stocks at low prices and selling them later at higher prices. It is often useful just to predict whether the trend of a stock is upward or downward in the near future. In this project, you will use standard time series smoothing techniques to estimate trends in the S&P 500 Index.

Using the data set SP500.csv (https://openstax.org/r/spreadsheetsd1j), perform multiple techniques to smooth the time series including simple moving averages with various window lengths and exponential moving averages.

a. Produce SMA trendlines with window sizes 10, 30, 60, and 120. Graph the trendlines on the same set of axes as the original graph and describe the results. What happens to the trendline as the window size increases?
b. Produce EMA trendlines with $\alpha = 0.25, 0.5$ and 0.75. Graph the trendlines on the same set of axes as the original graph and describe the results. What happens to the trendline as the value of α increases?
c. Now compare your results from part a and part b. As a group, discuss the advantages and disadvantages of using SMA vs. EMA. What characteristics of the data are emphasized (or de-emphasized) by each method as the parameters (window size and α, respectively) change.

Project B – ARIMA Analysis of Coal Consumption

As a group, explore the time series in the data set MonthlyCoalConsumption.xlsx (https://openstax.org/r/spreadsheetsd1j).

a. Find the smallest order of differencing so that the time series becomes stationary.
b. Then run an ACF diagram and look for any spikes that might indicate periodicity of a seasonal component.
c. Run ARIMA analysis of the time series using the parameters for p and d established by your findings about periodicity and stationarity, trying different values of q. Evaluate the model's accuracy by computing the AIC score for each combination of parameters.
d. As a group, decide which parameters (p, d, q) you think will give the most accurate forecast. Then forecast

up to 30 time units into the future and produce confidence intervals for your forecasts. Interpret your results by indicating how likely your forecasts are to be within a range of values.

Critical Thinking

1.

a. What characteristics define a time series?

b. Which of the following are examples of time series data?

i. The vital statistics of a cancer patient taken right before a major surgery

ii. The monthly expenses of a small company recorded over a period of five years

iii. Daily temperature, rainfall, humidity, and wind speeds measured at a particular location over a few months

iv. Student final grades in all sections of a course at a university

2. Consider the graph in the figure as shown below.

a. Is the trend curve generally increasing, decreasing, or staying level over the entire time period?

b. The data set USATemps1961-2023.csv (https://openstax.org/r/spreadsheetsd1j) contains the time series from which the graph was created. Find the moving average trendline with period 10 for the data.

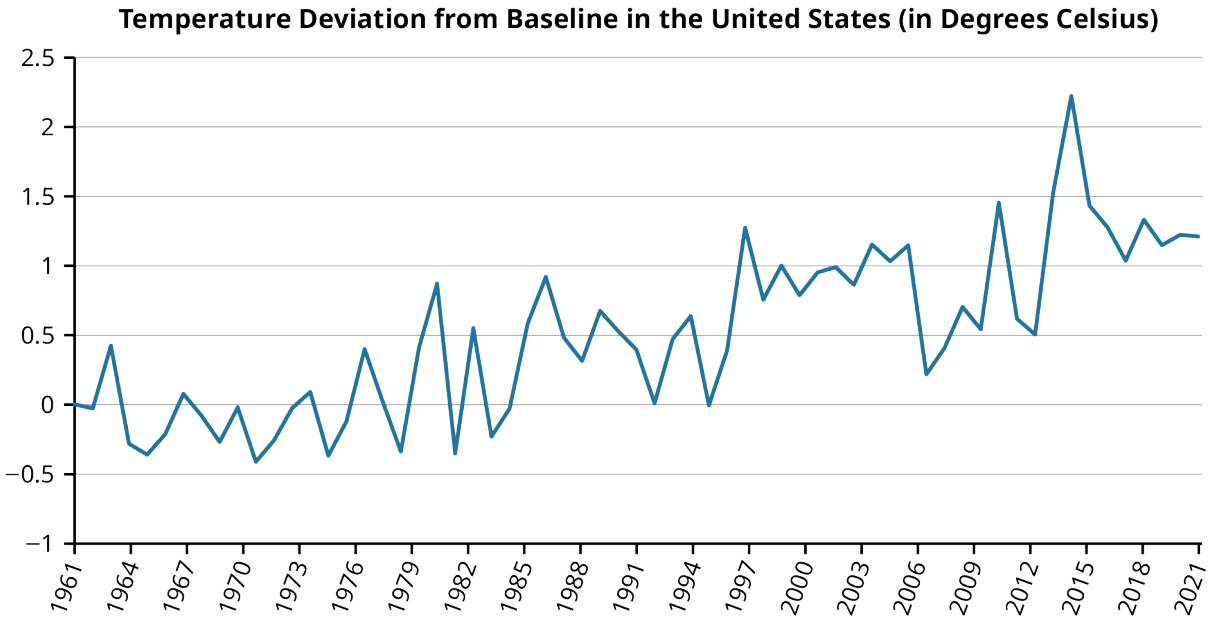
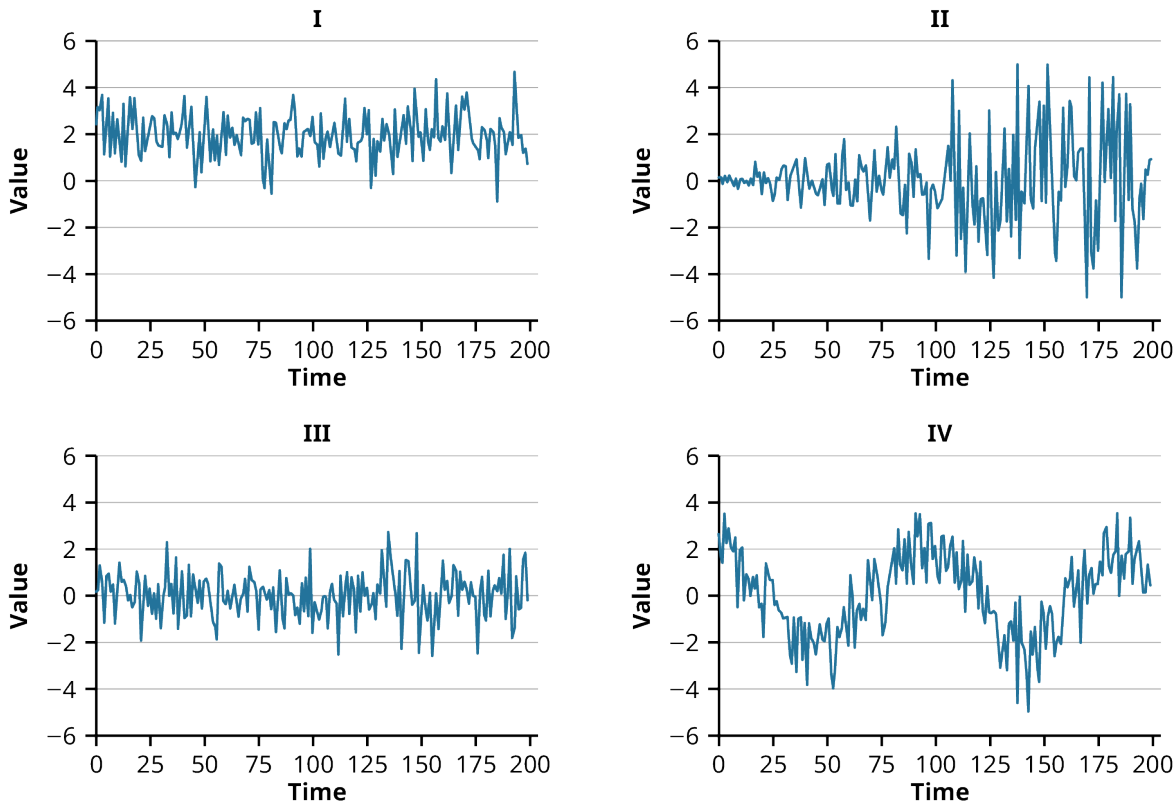


Figure 5.20

3.

a. What are the characteristics of white noise? Why is it important that the residuals of a time series model be white noise?

b. Determine which of the following graphs most likely represents white noise.



4. Four time series models were produced to model a given dataset. Various measures of error were run on each of the models, the results of which are shown in the table.

	MAE	RMSE	MAPE
Model 1	103.1	131.2	0.097
Model 2	251.7	207.9	0.136
Model 3	110.7	129.0	0.115
Model 4	89.8	125.3	0.191

Table 5.8 Measures of Error

a. Identify the most accurate model based only on:
i. MAE
ii. RMSE
iii. MAPE

b. Which model is likely to produce the least accurate forecasts overall, and why?
c. Which model is likely to produce the most accurate forecasts overall, and why?

Quantitative Problems

1. A U.S. company located in New Hampshire records weekly sales of its main product for a period of 3 years. If a seasonal variation in sales exists, what would be the most likely period of the seasonal variation?

2. Consider the time series given in the table as shown.

Month	Value
1	880.7
2	727.2
3	798.5
4	504.1
5	888.4
6	725.8
7	793.4
8	499.0
9	891.7
10	722.0
11	789.1
12	501.6

Table 5.9 Time Series

a. Identify the period of the seasonal component.
b. Using a centered simple moving average (SMA) of window size equal to the period you found in part a, identify a trend-cycle component, t_n, in the data.
c. Detrend the data by subtracting the SMA you found in part b.
d. Identify the seasonal component, $\hat{s}_n$, using the method of averaging.
e. Find the seasonally adjusted time series.

3. In **Collecting and Preparing Data**, the following data was given, as provided in the table, showing daily new cases of COVID-19. Since new cases were not reported on Saturdays and Sundays, those weekend cases were added to the Monday cases. One way to deal with the missing data is to use a centered simple moving average to smooth the time series.

Date	Weekday	New Case
10/18/2021	Monday	3115
10/19/2021	Tuesday	4849

Table 5.10 Sample of COVID-19 Data Cases within 23 Days (source: https://data.cdc.gov/Case-Surveillance)

10/20/2021	Wednesday	3940
10/21/2021	Thursday	4821
10/22/2021	Friday	4357
10/23/2021	Saturday	0
10/24/2021	Sunday	0
10/25/2021	Monday	8572
10/26/2021	Tuesday	4463
10/27/2021	Wednesday	5323
10/28/2021	Thursday	5012
10/29/2021	Friday	4710
10/30/2021	Saturday	0
10/31/2021	Sunday	0
11/1/2021	Monday	10415
11/2/2021	Tuesday	5096
11/3/2021	Wednesday	6882
11/4/2021	Thursday	5400
11/5/2021	Friday	6759
11/6/2021	Saturday	0
11/7/2021	Sunday	0
11/8/2021	Monday	10069
11/9/2021	Tuesday	5297

Table 5.10 Sample of COVID-19 Data Cases within 23 Days (source: https://data.cdc.gov/Case-Surveillance)

a. What is the most appropriate window size to use for centered SMA to address the issue of missing data in their analysis of COVID-19 data from the CDC?

b. Perform the SMA and illustrate the results by a graph.

4. Consider the data set USATemps1961-2023.csv (https://openstax.org/r/spreadsheetsd1j).
   a. Use an ACF plot to determine the period of a potential seasonal component.
   b. Use STL to decompose the time series into trend-cycle, seasonal, and noise components.

5. Use the recursive EMA formula with $\alpha = 0.6$ to smooth the time series found in USATemps1961-2023.csv (https://openstax.org/r/spreadsheetsd1j), and then use the EMA model to forecast the next value of the series.

6. A very small time series is given in the table, along with a model.

x_n	$\hat{x}_n$
122	117
108	135
172	152
190	169
167	186

Table 5.12 Time Series

   a. Find error measures MAE, RMSE, MAPE, and sMAPE for the model.
   b. Which of the error measures are scale-dependent? What would this imply about a similar time series and model in which all values are multiplied by 10?

6 Decision-Making Using Machine Learning Basics

Figure 6.1 Machine learning may be used to classify all kinds of data. Facial recognition technology relies heavily on machine learning algorithms, such as random forest models. (credit: modification of work "Humanoid Robot Uprising" by Steve Jurvetson/Flickr, CC BY 2.0)

Chapter Outline

6.1 What Is Machine Learning?
6.2 Classification Using Machine Learning
6.3 Machine Learning in Regression Analysis
6.4 Decision Trees
6.5 Other Machine Learning Techniques

Introduction

Imagine a world in which your face is your passport. No need for keys, cards, or passwords; your identity is simply your unique appearance. In this world, machine learning plays the role of gatekeeper, using the visible features of your face to identify you and grant you access. Machine learning has a broad range of applications that go beyond facial recognition, transforming the way we interact with technology. As we'll see in this chapter and Deep Learning and AI Basics, machine learning is also used in medical image analysis, satellite image interpretation, and augmented reality (e.g., in gaming, education/training, and architecture/design). It is used to power recommendation systems and for predictive analytics. In this chapter, we explore the fundamentals of classification, clustering, and regression algorithms that allow machines to learn how to label data accurately and make useful decisions that can be put to use in these many applications.

6.1 What Is Machine Learning?

Learning Outcomes

By the end of this section, you should be able to:

- 6.1.1 Summarize the differences between supervised and unsupervised learning in machine learning.
- 6.1.2 Describe the roles of training and testing datasets.
- 6.1.3 Use common measures of accuracy and error to determine the fitness of a model.
- 6.1.4 Explain the concepts of overfitting and underfitting and how these problems can adversely affect the model.

A **machine learning (ML) model** is a mathematical and computational model that attempts to find a relationship between input variables and output (response) variables of a dataset. The way that the model "learns" is by adjusting internal parameters until the model meets a certain level of **accuracy** (defined as a measure of how correct the model is when used to predict. We will have more to say on accuracy later). What sets apart a machine learning model from a typical mathematical model is that the user does not have full control over all the parameters used by the model. Moreover, there may be hundreds, thousands, or even millions of parameters that are used and adjusted within a given ML model—way more than a human could reasonably track.

The part of that dataset that is used for initial learning is the **training set (or data)**. There is often a **testing set (or data)** as well, which (as the name implies) can be used to determine whether the model is accurate enough. As a general rule of thumb, about 60–80% of available data will be used for training and the remaining data used for testing. After a machine learning model is trained, it can be used to make predictions as new input data is fed in and the responses produced. There may be multiple rounds of training and testing before the ML model is ready to make predictions or decisions.

The machine learning life cycle typically follows these steps:

1. **Problem formulation/identification:** Clearly state the problem you want the model to solve.
2. **Data collection and preparation:** Collect relevant data and clean it for analysis.
3. **Feature selection/engineering:** Choose important aspects of the data for the model.
4. **Model/algorithm selection:** Select a suitable machine learning algorithm for the task.
5. **Model/algorithm training:** Feed in the training data to teach the model patterns in the data.
6. **Model validation:** Test how well the model predicts outcomes.
7. **Model implementation:** Put the model to work in real-world scenarios.
8. **Performance monitoring and enhancement:** Keep an eye on the model's accuracy and update as needed.
9. **Continuous improvement and refinement:** Repeat and refine steps based on feedback and changes.

Throughout the process, be sure to keep ethical considerations in mind to ensure fairness, privacy, and transparency. For example, were the data collected with the proper permissions? Does the training data accurately reflect the same diversity of data that the algorithm will ultimately be used on? Currently there are serious questions as to the ethical implications of using facial recognition algorithms in law enforcement, as training images may have been gathered from people without their explicit consent, and the models created often reflect biases that can result in discriminatory outcomes, disproportionately affecting certain demographic groups, such as people of color or women.

Bias—error introduced by overall simplistic or overly rigid models that do not capture important features of the data—can be hard to identify! One example that is famous in data science circles (though it may be apocryphal) is the "Russian tank problem" (Emspak, 2016). As the story goes, the United States military trained an early ML model to distinguish between Russian and US tanks. The training set consisted of photos of many

different kinds of tanks used by both countries. Although the ML model produced very accurate results on the training and test sets, it performed terribly out in the field. Apparently, most of the pictures of Russian tanks were low quality and blurry, while those of the US tanks were high quality. They had inadvertently trained their ML model to distinguish between low- and high-quality photos regardless which county's tanks were depicted in them!

Supervised vs. Unsupervised Learning

Data may be labeled or unlabeled. **Labeled data** has specific labels, categories, or classes associated with each data point. For example, a dataset containing the height, weight, cholesterol levels, and various other health-related measures for patients at a clinic, along with an indication as to whether each is susceptible to heart disease or not, would be labeled data. The label is the value (yes or no) of the variable "susceptible to heart disease." **Unlabeled data** does not have a specific label, category, or class associated with its data points. For example, in product marketing, we might collect data representing customer attributes such as age, income, and purchase history, and we may want to identify distinct groups of customers with similar characteristics without knowing ahead of time what those groups (labels) will be.

Supervised learning uses labeled data in its training and testing sets. Parameters are adjusted in the model based on the correctness of the results as compared with the known labels. **Unsupervised learning** trains on unlabeled data. Often, both methods will be used in the same project. Unsupervised learning might be employed as an initial step to find patterns in the dataset, allowing the data to be labeled in some way. Then supervised learning techniques may be applied to the labeled data as a training step. The model created in this way may then be used to determine the best labels for future (unlabeled) data.

Supervised Learning

Supervised learning is analogous to a student learning from an instructor. At first, the student may answer many questions incorrectly, but the instructor is there to correct the student each time. When the student gives (mostly) correct responses, there is evidence that the student has sufficiently learned the material and would be able to answer similar questions correctly out in the real world where it really matters.

The goal of supervised learning is to produce a model that gives a mapping from the set of inputs or features to a set of output values or labels (see Figure 6.2). The model is function $y = f(X)$, where X is the input array and y is the output value or label. The way that it does this is through training and testing. When training, the algorithm creates a correspondence between input features and the output value or label, often going through many iterations of trial and error until a desired level of accuracy is achieved. The model is then tested on additional data, and if the accuracy remains high enough, then the model may be deployed for use in real-world applications. Otherwise, the model may need to be adjusted substantially or abandoned altogether.

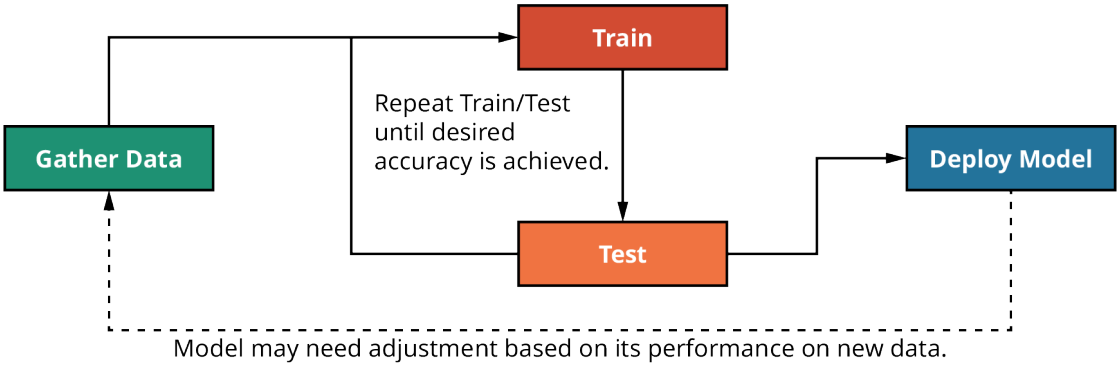


Figure 6.2 The Supervised Learning Cycle. The supervised learning cycle consists of gathering data, training on some part of the data, testing on another part of the data, and deployment to solve problems about new data.

Examples of supervised learning models include linear regression (discussed in Correlation and Linear Regression Analysis), logistic regression, naïve Bayes classification, decision trees, random forests, and many

kinds of neural networks. Linear regression may not seem like a machine learning algorithm because there are no *correction* steps. A formula simply produces the line of best fit. However, the regression formula itself represents an optimization that reduces the error in the values predicted by the regression line versus the actual data. If the linear regression model is not accurate enough, additional data could be collected to improve the accuracy of the model, and once the regression line is found, it may then be used to predict values that were not in the initial dataset. Linear regression has been developed in previous chapters, and you will see logistic regression, Bayes methods, decision trees, and random forests later in this chapter.

Unsupervised Learning

Unsupervised learning is like building a puzzle without having the benefit of a picture to compare with. It is not impossible to build a puzzle without the picture, but it may be more difficult and time-consuming.

The main advantage of unsupervised learning is that the training data does not need to be labeled. All the data is considered input, and the output would typically be information about the inherent structure of the data. Common tasks are to determine clusters, patterns, and shapes within the dataset. For example, if the current locations in latitude and longitude of every person in the world were somehow known and collected into a dataset, then an algorithm may discover locations of high densities of people—hence *learn* of the existence of cities!

Unsupervised learning algorithms include k-means clustering, DBScan, and some kinds of neural networks. More generally, most clustering algorithms and dimension-reduction methods such as principal component analysis (PCA) and topological data analysis (TDA) fall into the category of unsupervised learning. You will encounter the k-means and DBScan algorithms in this chapter. The more advanced topics of PCA and TDA are outside the scope of this text.

Variations and Hybrid Models

Although we will only cover supervised and unsupervised models, it is important to note that there are some additional models of machine learning.

When some of the data in a dataset has labels and some does not, a *semi-supervised learning algorithm* may be appropriate. The labeled data can be run through a supervised learning algorithm to generate a predictive model that can then be used to label the unlabeled data, called *pseudo-data*. From this point, the dataset can be regarded as completely labeled data, which can then be analyzed using supervised learning.

Reinforcement learning introduces rewards for accuracy while also penalizing incorrectness. This model most closely follows the analogy of a student-teacher relationship. Many algorithms in artificial intelligence use reinforcement learning. While reinforcement generally improves accuracy, there can be issues in locking the model into a rigid state. For example, have you ever noticed that when you search online for a type of product, you begin to see advertisements for those kinds of products everywhere—even after you've purchased it? It's as if the algorithm has decided that the only product you ever wish to purchase is what you have just searched for!

Training and Testing a Model

How do we know how good a model is? The key is to compare its output against known output. In supervised learning models, there is always a metric that measures efficacy of the model. During the training phase, this metric helps to determine how to adjust the model to make it more accurate. During the testing phase, the metric will be used to evaluate the model and determine if it is good enough to use on other datasets to make predictions. In this section, we will explore the supervised learning cycle in more detail.

Model Building

Suppose we are given a dataset with inputs or features X and labels or values y. Our goal is to produce a function that maps $y = f(X)$. In practice, we do not expect that our model can capture the function f with

100% accuracy. Instead, we should find a model $\widehat{f}$ such that predicted values $\hat{y} = \widehat{f}(X)$ are *close enough to* y. In most cases, the metric we use to determine the fitness of our model $\widehat{f}$ is some measure of error between $\hat{y}$ and y (e.g., see the calculation of residuals of a linear regression in Correlation and Linear Regression Analysis). The way that error is measured is specific to the type of model that we are building and the nature of the outputs. When the outputs are labels, we may use a count or percentage of misclassifications. When the outputs are numerical, we may find the total distances (absolute, squared, etc.) between predicted and actual output values.

First, the data are separated into a training set and testing set. Usually about 60–80% of the data is used for training with the remaining amount used for testing. While training, parameters may be adjusted to reduce error. The result of the training phase will be the model $\widehat{f}$. Then, the model is tested on the testing set and error computed between predicted values and known outputs. If the model meets some predefined threshold of accuracy (low enough error), then the model can be used with some confidence to make predictions about new data.

Measures of Accuracy

As mentioned previously, it is important to measure how good our machine learning models are. The choice of a specific measure of accuracy depends on the nature of the problem. Here are some common accuracy measures for different machine learning tasks:

1. Classification
   - Accuracy is the ratio of correct predictions to the total number of predictions.
   - **Precision** (p) is the ratio of true positive predictions (TP) to the total number of positive (true positive plus false positive) predictions $(TP + FP)$. This is useful when the response variable is true/false and we want to minimize false positives. $p = \frac{TP}{TP+FP}$
   - **Recall** (r) is the ratio of true positive predictions (TP) to the total number of actual positives (true positive plus false negative $(TP + FN)$ predictions. This is useful when the response variable is true/false and we want to minimize false negatives. $r = \frac{TP}{TP+FN}$
   - **F1 Score** is a combination of precision (p) and recall (r). $F1 = \frac{2(p)(r)}{p+r} = \frac{2TP}{2TP+FP+FN}$
2. Regression (comparison of the predicted values, $\hat{y}_i$, and the actual values, y_i)
   - **Mean absolute error (MAE)**: $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
   - **Mean absolute percentage error (MAPE)**: $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|$
   - **Mean squared error (MSE)**: $\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$
   - **Root mean squared error (RMSE)**: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$
   - *R-squared* is the proportion of the variance in the response variable that is predictable from the input variable(s).

Note that MAE and RMSE, also discussed in Forecast Evaluation Methods, have the same units as the data itself. MSE and RMSE can be very sensitive to outliers because the differences are squared. MAPE (also introduced in Forecast Evaluation Methods) is often represented in percent form, which you can easily find by multiplying the raw MAPE score by 100%. The R-squared measure, which is discussed in Inferential Statistics and Regression Analysis in the context of linear regression, can be computed by statistical software packages, though we will not make significant use of it in this chapter. In all cases, the smaller the value of the error term, the more accurate the modeling would be.

Due to the tedious calculations involved in these measures, statistical software packages are typically used to compute them in practice. But it is important for every data scientist to understand how the formulas work so that results can be evaluated and interpreted correctly. Next, we will work out a small example by hand so you can have experience with the formulas.

EXAMPLE 6.1

Problem

A test for COVID-19 was applied to 1,000 patients. Two hundred thirty-five tested positive for COVID-19, while the remaining 765 tested negative. Of those that tested positive, 198 turned out to carry the COVID-19 virus, and 37 did not have the virus. Of those that tested negative, 63 patients did in fact have the virus, while the remaining 702 did not. Compute the accuracy, precision, recall, and F1 scores for this experiment.

Solution

First, the total number of correct predictions were: 198 true positive predictions + 702 true negative predictions, for a total of $TP + TN = 900$. So the accuracy is $\frac{900}{1{,}000} = 0.9(90\%)$.

For precision, we use $TP = 198$ and FP (false positives) $= 37$, and we have: $p = \frac{TP}{TP+FP} = \frac{198}{198+37} = 0.843$ (84.3%).

For recall, we use $TP = 198$ and FN (false negatives) $= 63$, and we have: $r = \frac{TP}{TP+FN} = \frac{198}{198+63} = 0.759$ (75.9%).

Finally, $F1 = \frac{2(p)(r)}{p+r} = \frac{2(0.843)(0.759)}{0.843+0.759} = 0.799$ (79.9%).

EXAMPLE 6.2

Problem

The score on an exam is positively correlated with the time spent studying for the exam. You have collected the following data (pairs of study time in hours and exam scores):

(9.8, 83.0), (8.2, 87.7), (5.2, 61.6), (8.0, 77.8), (2.1, 42.2), (6.8, 62.1), (2.3, 30.9), (9.5, 94.4), (6.6, 76.2), (9.5, 93.1)

Use 80% of the dataset to create a linear model and discuss the accuracy of the model on the remaining testing set.

Solution

Since there are 10 data points, we use 8 of them for training, as 8 is 80% of 10.

Training data: (9.8, 83.0), (8.2, 87.7), (5.2, 61.6), (8.0, 77.8), (2.1, 42.2), (6.8, 62.1), (2.3, 30.9), (9.5, 94.4)

The linear regression model that best fits the training data is $y = 21.7 + 7.1x$. (See Correlation and Linear Regression Analysis for details on linear regression.) We will compute MAE, MAPE, MSE, and RMSE for the remaining two data points (test set) as in Table 6.1.

x	Prediction	Actual Value	Difference (Error)
2.3	$21.7 + 7.1\,(2.3) = 38.03$	30.9	-7.13
9.5	$21.7 + 7.1(9.5) = 89.15$	94.4	5.25

Table 6.1 Test Set

- $\text{MAE} = \frac{1}{2}(|-7.13| + |5.25|) = 6.19$

- $\text{MAPE} = \frac{1}{2}\left(\left|\frac{-7.13}{30.9}\right| + \left|\frac{5.25}{94.4}\right|\right) = 0.143$, or 14.3%
- MSE for cubic model: $\frac{1}{2}\left((-7.13)^2 + 5.25^2\right) = 39.2$
- RMSE for cubic model: $\sqrt{39.2} = 6.26$

The MAE and RMSE both show that the model's predictions are off by an average of about 6.2 points from actual values. The MAPE suggests that the model has an error of about 14.3% on average.

Overfitting and Underfitting

Can 100% accuracy be attained from a machine learning algorithm? In all but the most trivial exercises, the answer is a resounding ***no***. Real-world data is noisy, often incomplete, and rarely amenable to a simple model. (Recall from Collecting and Preparing Data that we defined noisy data as data that contains errors, inconsistencies, or random fluctuations that can negatively impact the accuracy and reliability of data analysis and interpretation.) Two of the most common issues with machine learning algorithms are overfitting and underfitting.

Overfitting happens when the model is adjusted to fit the training data too closely, resulting in a complex model that is too specific to the training data. When such a model is given data outside of the training set, it may perform worse, a phenomenon known as high **variance**.

Underfitting occurs when the model is not robust enough to capture essential features of the data. This could be because of too small a dataset to train on, choosing a machine learning algorithm that is too rigid, or a combination of both. A model that suffers from underfitting may be said to have *high bias*.

Often, a model that has low variance will have high bias, and a model that has low bias will have high variance. This is known as the *bias-variance trade-off*.

Overfitting

It might at first seem desirable to produce a model that makes no mistakes on the training data; however, such "perfect" models are usually terrible at making predictions about data not in the training set. It is best to explain this idea by example.

Suppose we want to create a model for the relationship between *x* and *y* based on the given dataset,

$$S = (x, y) = (2, 10), (4, 9), (6, 6), (8, 7), (10, 4), (12, 5)$$

How would we go about creating the model? We might first split S into a training set, $S_{\text{train}} = \{(2, 10), (4, 9), (6, 6), (8, 7)\}$, and testing set, $S_{\text{test}} = \{(10, 4), (12, 5)\}$. It can be shown that the cubic equation, $\hat{y} = \frac{1}{8}x^3 - \frac{7}{4}x^2 + \frac{13}{2}x + 3$, fits the training data perfectly. However, there is no reason to expect that the relationship between *x* and *y* is truly cubic. Indeed, the predictions of the cubic model on S_{test} are terrible! See Table 6.2.

x	Prediction, Cubic Model	Actual Value, *y*
10	18	4
12	45	5

Table 6.2 Cubic Model Predictions

The linear model (linear regression) that represents a best fit for the first four data points is $\hat{y} = -0.6x + 11$. Figure 6.3 shows the data points along with the two models.

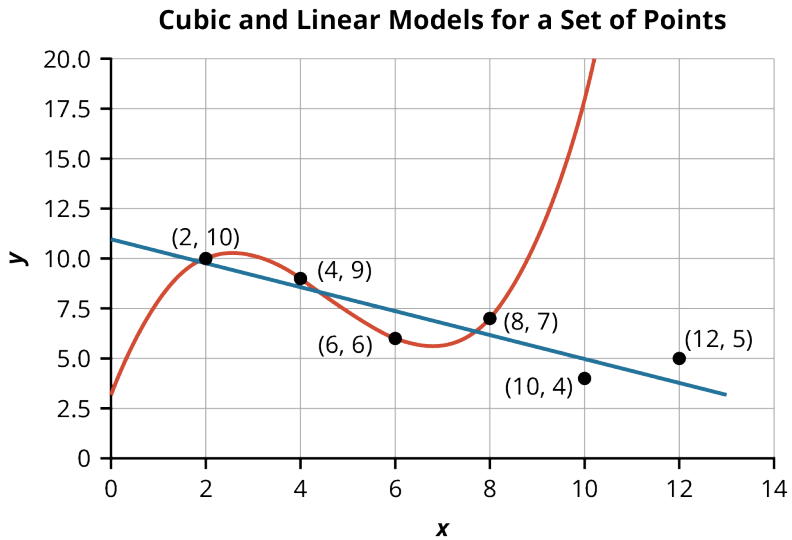


Figure 6.3 Data Points along a Cubic Model and a Linear Model. The four data points in S_{train} can be fitted onto a cubic curve (0% error), but it does poorly on additional data from the testing data, S_{test}. The cubic curve overfits the training data. The regression line does not fit the training data exactly; however, it does a better job predicting additional data.

To get a good sense of how much better the linear model is compared to the cubic model in this example, we will compute the MAPE and RMSE for each model, as shown in Table 6.3.

x_i	Actual Value, y_i	Prediction, y_i, Cubic Model	$\lvert y_i - \hat{y}_i \rvert$, Cubic Model	Prediction, $\hat{y}_i$, Linear Model	$\lvert y_i - \hat{y}_i \rvert$, Linear Model
2	**10**	10	0	9.8	0.2
4	**9**	9	0	8.6	0.4
6	**6**	6	0	7.4	1.4
8	**7**	7	0	6.2	0.8
10	**4**	18	14	5	1
12	**5**	45	40	3.8	1.2

Table 6.3 MAPE and RMSE Calculated for Each Model

- MAPE for cubic model: $\frac{1}{6}(0/10 + 0/9 + 0/6 + 0/7 + 14/4 + 40/5) = 1.917$, or 191.7%
- RMSE for cubic model: $\sqrt{\frac{1}{6}\left(0^2 + 0^2 + 0^2 + 0^2 + 14^2 + 40^2\right)} = 17.3$
- MAPE for linear model: $\frac{1}{6}(0.2/10 + 0.4/9 + 1.4/6 + 0.8/7 + 1/4 + 1.2/5) = 0.15$, or 15%
- RMSE for linear model: $\sqrt{\frac{1}{6}\left(0.2^2 + 0.4^2 + 1.4^2 + 0.8^2 + 1^2 + 1.2^2\right)} = 0.93$

The MAPE and RMSE values for the linear model are much lower than their respective values for the cubic model even though the latter predicts four values with 100% accuracy.

Underfitting

A model that is underfit may perform well on some input data and poorly on other data. This is because an underfit model fails to detect some important feature(s) of the dataset.

EXAMPLE 6.3

Problem

A student recorded their time spent on each homework set in their statistics course. Each HW set was roughly similar in length, but the student noticed that they spent less and less time completing them as the semester went along. Find the linear regression model for the data in Table 6.4 and discuss its effectiveness in making further predictions.

HW Set	1	2	3	4	5	6
Time (hr)	9.8	5.3	3.4	2.6	1.9	1.7

Table 6.4 Homework Set Completion Times

Solution

There is a definitive downward trend in the data. A simple linear regression model for this data is:

$$\hat{y} = -1.47143x + 9.26667$$

However, the line of best fit does not seem to fit the data all that well. There is no way that a linear function can *bend* the way the data points seem to do in this example (see Figure 6.4).

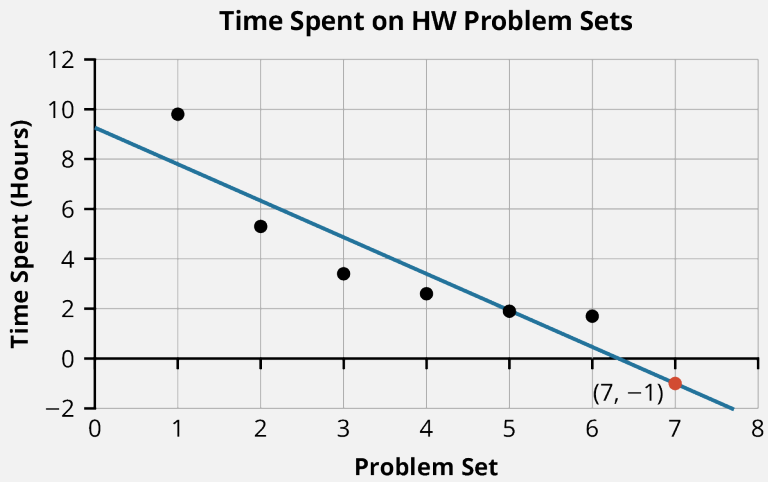


Figure 6.4 Linear Model That Best Fits the Data in Example 6.3. This graph contains a scatterplot of the six data points together with the linear model that best fits the data. This is a clear case of underfitting.

Another issue with using the linear model for the data in Table 6.4 is that eventually the values of $\hat{y}$ become negative! According to this model, it will take about -1 hours to complete the 7th homework set, which is absurd. Perhaps a decreasing exponential model, shown in Figure 6.5, would work better here. Proper model selection is the most effective way of dealing with underfitting.

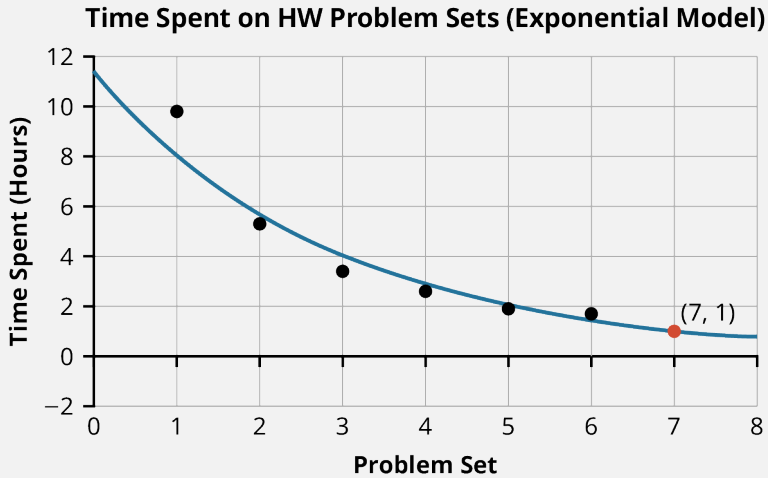


Figure 6.5 Exponential Model That Best Fits the Data in Example 6.3. This graph contains a scatterplot of the six data points together with the exponential model that best fits the data. Under this model, the prediction of $\hat{y} = 1$ when $x = 7$ seems much more reasonable.

6.2 Classification Using Machine Learning

Learning Outcomes

By the end of this section, you should be able to:

- 6.2.1 Perform logistic regression on datasets and interpret the results.
- 6.2.2 Perform k-means clustering on datasets.
- 6.2.3 Define the concept of density-based clustering and use DBScan on datasets.
- 6.2.4 Interpret the confusion matrix in clustering or classifying data.

Classification problems come in many flavors. For example, suppose you are tasked with creating an algorithm that diagnoses heart disease. The input features may include biological sex, age, blood pressure data, and cholesterol levels. The output would be either: yes (diagnose heart disease) or no (do not diagnose heart disease). That is, your algorithm should classify patients as “yes” or “no” based on an array of features, or *symptoms* in medical terminology. Logistic regression is one tool for classification when there are only two possible outputs. This is often called a **binary (binomial) classification** problem. If the goal is to classify data into more than two classes or categories, then the problem is referred to as **multiclass (multinomial) classification**.

Other kinds of classification problems involve finding two or more clusters in the data. A **cluster** is a collection of data points that are closer to one another than to other data points, according to some definition of *closeness*. For example, certain characteristics of music, such as tempo, instrumentation, overall length, and types of chord patterns used, can be used as features to classify the music into various genres, like rock, pop, country, and hip-hop. A supervised machine learning algorithm such as a decision tree (see Decision Trees) or random forest (see Other Machine Learning Techniques) may be trained and used to classify music into the various predefined genres. Alternatively, an unsupervised model such as k-means clustering could be used to group similar-sounding songs together without necessarily adhering to a concept of *genre.*

This section will focus on logistic regression techniques and clustering algorithms such as k-means and DBScan. We’ll use measures of accuracy including the confusion matrix.

Logistic Regression

A **logistic regression** model $L(X)$ takes input vectors X (features) and produces an output of “yes” or “no.” Much like linear regression, the logistic regression model first fits a continuous function based on known,

labeled data. However, instead of fitting a line to the data, a sigmoid function is used. A **sigmoid function** is a specific type of function that maps any real-valued number to a value between 0 and 1. The term *sigmoid* comes from the fact that the graph has a characteristic "S" shape (as shown in Figure 6.6), and *sigma* (σ) is the Greek letter that corresponds to our letter *S*. We often use the notation $\sigma(x)$ to denote a sigmoid function. The basic formula for the sigmoid is:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

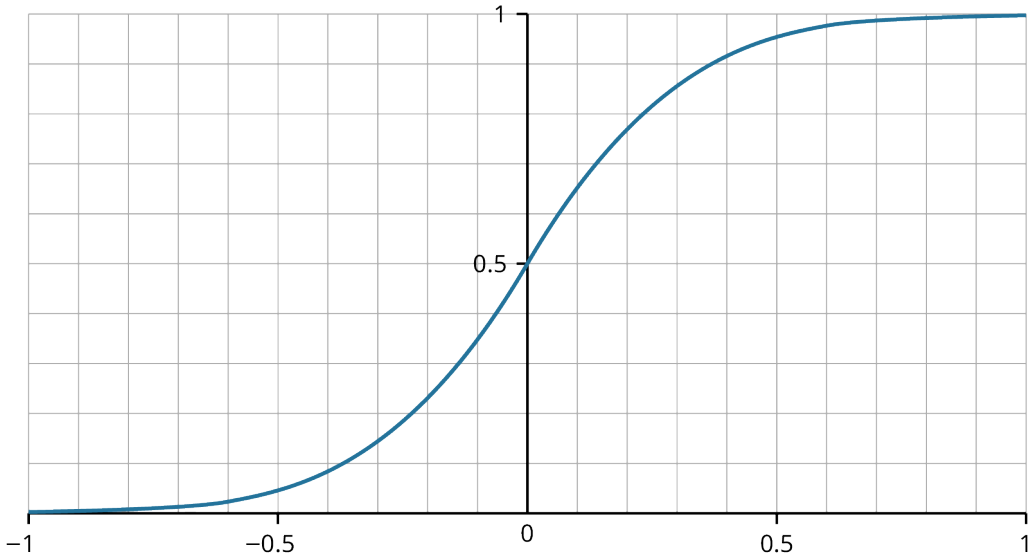


Figure 6.6 Graph of a Typical Sigmoid Function

One advantage of the sigmoid over a linear function is that the values of the sigmoid stay close to 0 when x is negative and close to 1 when x is positive. This makes it ideal for classifying arbitrary inputs x into one of two classes, "yes" (1) or "no" (0). In fact, we can interpret the *y*-axis as probability that an event occurs. The sigmoid function can be shifted right or left and compressed or expanded horizontally as necessary to fit the data. In the following formula, the parameters a and b control the position and shape of the sigmoid function, respectively.

$$\sigma(a+bx) = \frac{1}{1+e^{-(a+bx)}}$$

As complicated as this formula may seem at first, it is built up from simple elements. The presence of $a+bx$ suggests that there may be a linear regression $y = a+bx$ lurking in the background. Indeed, if we solve to find the inverse function of $\sigma(x)$, we can use it to isolate $a+bx$, which in turns allows us to use standard linear regression techniques to build the logistic regression. Recall that the natural logarithm function, ln n, is the inverse function for the exponential, e^x. In what follows, let p stand for $\sigma(x)$.

$$p = \sigma(x) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned}
\frac{1}{p} &= 1+e^{-x} \\
\frac{1}{p} - 1 &= e^{-x} \\
e^{-x} &= \frac{1}{p} - 1 = \frac{1-p}{p} \\
-x &= \ln\left(\frac{1-p}{p}\right) \\
x &= -\ln\left(\frac{1-p}{p}\right) = \ln\left[\left(\frac{1-p}{p}\right)^{-1}\right] = \ln\left(\frac{p}{1-p}\right)
\end{aligned}$$

In the final line, we used the power property of logarithms, $\ln A^n = n \ln A$, with power $n = -1$. The function we've just obtained is called the logit function.

$$\text{logit}\,(p) = \ln\left(\frac{p}{1-p}\right)$$

The key property of the **logit function** is that it is the inverse function of $\sigma(x)$. Recall from algebra, if $g(x)$ is the inverse function of $f(x)$, then we have $g\,(f(x)) = x$ for all x. In our case, that means $\text{logit}\,(\sigma(x)) = x$, for all x. Now applying the logit function to the general sigmoid function, we can write:

$$\text{logit}\,(\sigma(a+bx)) = a + bx$$

Thus, the logit function *linearizes* the sigmoid. Let's dig a little deeper into the logit function and what it measures.

Log-Odds and Discrete Logistic Regression

Logistic regression is a prediction model based on the idea of odds. The **odds** that an event *E* occurs is the ratio of the probability of *E* occurring over the probability of *E* not occurring. Let $p = P(E)$.

$$\text{Odds}\,(E) = \frac{P(E)}{P(E')} = \frac{P(E)}{1-P(E)} = \frac{p}{1-p}$$

Thus, the logit function is simply the logarithm of the odds of an event with a given probability p. This is why logit is often called *log-odds*. While probabilities (p) range from 0 to 1, log-odds will range from $-\infty$ (minus infinity) to ∞ (infinity).

Suppose we would like to predict the likelihood of an event A, given that another event B either occurs or does not occur. We will create a logistic regression model, $f\,(x) = \frac{1}{1+e^{-(a+bx)}}$, such that:

- $f(0)$ is the likelihood that A occurs when B does not occur, and
- $f(1)$ is the likelihood that A occurs when B does occur.

We obtain a and b using the logit function. If p_1 is the probability that A occurs given that B does not occur (i.e., $p_1 = P(A|B')$), using the notation of conditional probability (see Normal Continuous Probability Distributions), and if p_2 is the probability that A occurs given that B does occur (i.e., $p_2 = P(A|B)$), then

$a = \text{logit}\,(p_1)$, and $b = \text{logit}\,(p_2) - \text{logit}\,(p_1)$

This model is called a discrete logistic regression because the feature variable is either 1 or 0 (whether or not event B occurred).

EXAMPLE 6.4

Problem

At a particular college, it has been found that in-state students are more likely to graduate than out-of-state students. Roughly 75% of in-state students end up graduating, while only about 30% of out-of-state students do so. Build a discrete logistic regression model based on this data.

Solution

Let $p_1 = 0.3$, which is the probability of graduating if the variable "in-state" is false, or 0. Let $p_2 = 0.75$, which is the probability of graduating if the variable "in-state" is true, or 1.

$$\begin{aligned}\text{logit}\,(p_1) &= \ln\left(\frac{0.3}{1-0.3}\right) = \ln 0.4286 = -0.847\\ \text{logit}\,(p_2) &= \ln\left(\frac{0.75}{1-0.75}\right) = \ln 3.000 = 1.099\end{aligned}$$

So $a = \text{logit}(p_1) = -0.847$, and $b = \text{logit}(p_2) - \text{logit}(p_1) = 1.099 - (-0.847) = 1.946$. Therefore, the model is:

$$f(x) = \frac{1}{1 + e^{-(-0.847+1.946x)}} = \frac{1}{1 + e^{0.847-1.946x}}$$

You can verify that $f(0) = 0.3$ and $f(1) = 0.75$.

The previous example used only a single feature variable, "in-state," and so the usefulness of the model is very limited. We shall see how to expand the model to include multiple inputs or features in Multiple Regression Techniques.

Maximum Likelihood and Continuous Logistic Regression

The method of logistic regression can also be used to predict a yes/no response based on continuous input. Building the model requires finding values for the parameters of the sigmoid function that produce the most accurate results. To explain how this works, let's first talk about likelihood. The **likelihood** for a model $f(x)$ is computed as follows. For each data point x_k, if the label of x_k is 1 (or yes), then calculate $f(x_k)$; if the label is 0 (or no), then calculate $1 - f(x_k)$. The product of these individual results is the likelihood score for the model. We want to find the model that has the maximum likelihood.

EXAMPLE 6.5

Problem

Consider the models

$$f(x) = \frac{1}{1 + e^{1.6-1.1x}}, g(x) = \frac{1}{1 + e^{2.1-0.8x}}$$

Suppose there are four data points, $A = 1$, $B = 2$, $C = 3$, $D = 4$, and both A and B are known to have label 0, while C and D have label 1. Find the likelihood scores of each model and determine which model is a better fit for the given data.

Solution

For model f, the predicted likelihoods are $f(1) = 0.378$, $f(2) = 0.646$, $f(3) = 0.846$, and $f(4) = 0.943$. Since $x = 1$ and 2 have label 0, while $x = 3$ and 4 have label 1, we compute:

$$(1 - 0.378)(1 - 0.646)(0.846)(0.943) = 0.176$$

For model g, the predicted likelihoods are $g(1) = 0.214$, $g(2) = 0.378$, $g(3) = 0.574$, and $g(4) = 0.75$.

$$(1 - 0.214)(1 - 0.378)(0.574)(0.75) = 0.211$$

Thus, model g is the better of the two models on this data.

How does one find the model that has maximum likelihood? The exact method for finding the coefficients a and b for the model $\sigma(a + bx)$ falls outside the scope of this text. Fortunately, there are software packages available that can be used to do the work for you.

For example, the same university ran a study on the effect of first-year GPA on completion of a college degree. A small sample of the data is shown in Table 6.5. What is the model that best fits this data?

GPA	Completed College?
1.5	N
2.4	Y
3.4	Y
2.1	N
2.5	Y
0.8	N
2.9	N
4.0	Y
2.3	Y
2.1	N
3.6	Y
0.5	N

Table 6.5 GPA vs. College Completion Data

Figure 6.7 shows the data in graphical form.

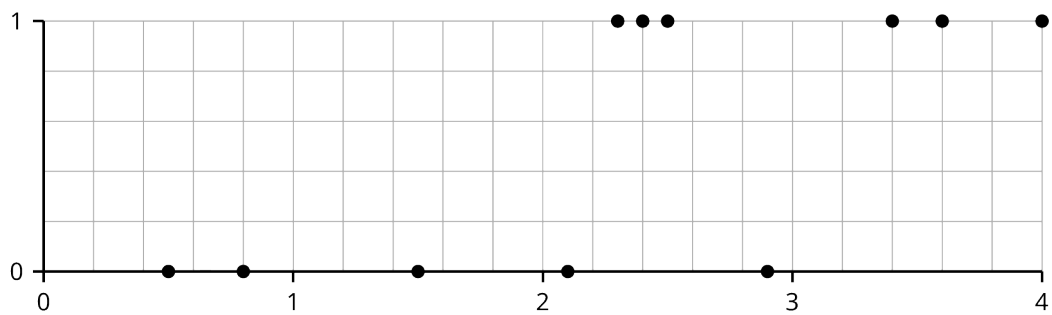


Figure 6.7 Graph of GPA (*x*-axis) and College Completion (*y*-axis). This uses values 0 for non-completion and 1 for completion. The graph suggests that higher GPAs tend to predict completion of a degree.

The logistics model (found using computer software) that maximizes the likelihood is:

$$\frac{1}{1 + e^{2.826 - 1.197x}}$$

The logistics regression model that best fits the data is shown in Figure 6.8 along with the data points colored green for predicted completion and red for predicted non-completion.

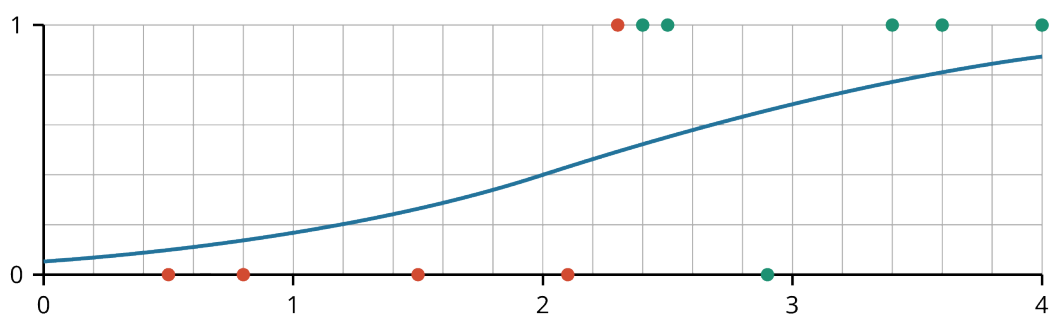


Figure 6.8 Logistic Regression Model Fitting the Data in Table 6.5

Logistic Regression in Python

A sample of Python code for logistic regression, using the data from Table 6.5, appears next. We use the `LogisticRegression` function found in the Python library sklearn.linear_model (https://openstax.org/r/scikit).

PYTHON CODE

```python
# Import libraries
from sklearn.linear_model import LogisticRegression

# Define input and output data
data = [[1.5, 0], [2.4, 1], [3.4, 1 ], [2.1, 0 ], [2.5, 1 ],
[0.8, 0 ], [2.9, 0 ], [4.0, 1], [2.3, 1], [2.1, 0], [3.6, 1 ], [0.5, 0]]

# Separate features (X) and labels (y)
X = [[row[0]] for row in data]  # Feature
y = [row[1] for row in data]    # Label

# Build the logistic model
Lmodel = LogisticRegression()
Lmodel.fit(X,y)

# Display the coefficients of the logistic regression
(a,b) = (Lmodel.intercept_[0], Lmodel.coef_[0,0])
print("Coefficients: ", (round(a,3),round(b,3)))

# Display the accuracy of the model
s = Lmodel.score(X,y)
print("Accuracy: ", round(s, 3))
```

The resulting output will look like this:

```
Coefficients: (-2.826, 1.197)
Accuracy: 0.833
```

The code produces the coefficients that can be plugged in for a and b in the model:

$$\sigma(-2.826 + 1.197x) = \frac{1}{1 + e^{-(-2.826+1.197x)}} = \frac{1}{1 + e^{2.826-1.197x}}$$

k-Means Clustering

Clustering algorithms perform like digital detectives, uncovering patterns and structure in data. The goal is to find and label groups of data points that are close to one another. This section develops methods for grouping data (clustering) that incorporate machine learning.

The **k-means clustering algorithm** can classify or group similar data points into clusters or categories without prior knowledge of what those categories might be (i.e., unsupervised learning). However, the user must provide a guess as to the value of k, the number of clusters to expect. (Note: You may get around this

issue by iterating the algorithm through many values of k and evaluating which k-value produced the best results. The trade-off is the time it takes to run the algorithm using multiple values of k and interpreting the results using. The so-called elbow method is often used to find the best value of k, but it is beyond the scope of this text to explain this method.) The user must also provide initial guesses as to where the centroids would be. We shall see by example how these choices may affect the algorithm.

k-Means Clustering Fundamentals

The basic idea behind k-means is quite intuitive: Each cluster must be centered around its centroid in such a way that minimizes distance from the data in the cluster to its centroid. The trick is in finding those centroids! Briefly, a **centroid** of a set of data points is defined as the arithmetic mean of the data, regarded as vectors.

$$\text{Centroid } = \frac{1}{N}\sum_{i=1}^{N} X_i$$

The basic steps of the k-means clustering algorithm are as follows:

1. Guess the locations of k cluster centers.
2. Compute distances from each data point to each cluster center. Data points are assigned to the cluster whose center is closest to them.
3. Find the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids stabilize—that is, the new centroids are the same (or approximately the same, within some given tolerance) as the centroids from the previous iteration.

If the clustering was done well, then clusters should be clearly separated. The **silhouette score** is a measure of how well-separated the clusters are and is defined as follows:

Let $a(i)$ be the mean distance between point i and all other data points in the same cluster as point i.

For each cluster J not containing point i, let $b_J(i)$ be the mean distance between point i and all data points in cluster J. Then, let $b(i)$ be the minimum of all the values of $b_J(i)$.

Then the silhouette score is found by the formula

$$S = \frac{1}{N}\sum_{i=1}^{N} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Values of S lie between −1 and 1, with values close to 1 indicating well-separated clusters, values close to 0 indicating clusters are ambiguous, and values close to −1 indicating poor clustering with points assigned to clusters arbitrarily. Fortunately, statistical software packages that can do k-means clustering will be able to compute the silhouette score for you.

EXAMPLE 6.6

Problem

An outbreak of fungus has affected a small garden area. The fungus originated underground, but it produced mushrooms that can easily be spotted. From a photo of the area, it is surmised that there are three clusters of mushrooms (see Figure 6.9). Use k-means to classify the mushrooms into three groups.

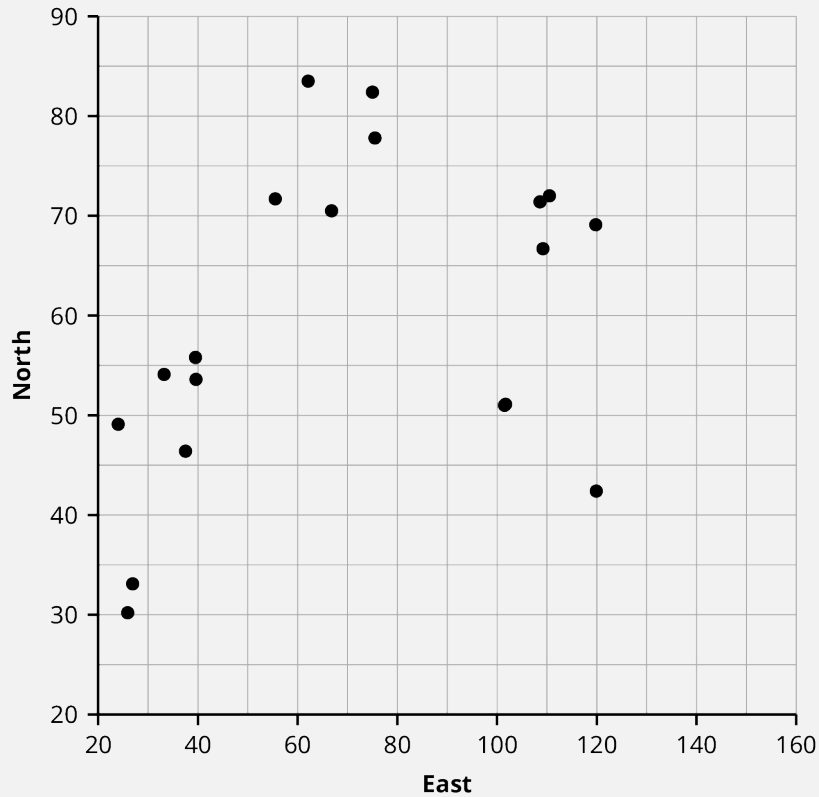


Figure 6.9 The Locations of Mushrooms in a Garden. There are three apparent clusters.

Solution

The location data consists of 19 points indicating position in feet east and north of a fixed point.

(101.7, 51.1), (109.2, 66.7), (33.2, 54.1), (39.6, 53.6), (66.8, 70.5), (119.9, 42.4),

(26.9, 33.1), (55.5, 71.7), (75, 82.4), (108.6, 71.4), (62.1, 83.5), (119.8, 69.1),

(25.9, 30.2), (37.5, 46.4), (24, 49.1), (75.5, 77.8), (101.5, 51), (110.5, 72),

(39.5, 55.8)

First, choose three potential centroids. To make things simple, we will just pick the first three data points (101.7, 51.1), (109.2, 66.7), and (33.2, 54.1), which we will color red, green, and blue, respectively. Now color all the data points red, green, and blue according to which of these three points it lies closest to. The initial grouping is not perfect, as you can see in Figure 6.10.

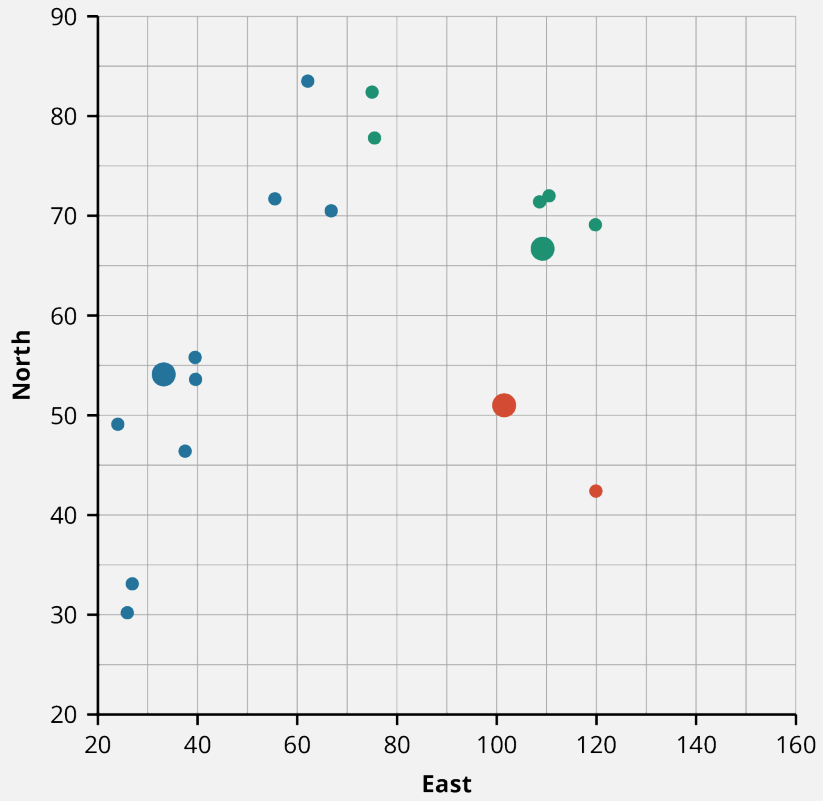


Figure 6.10 Graph with Large Dots Indicating the Initial Choices of Centroids. Data points are colored according to which centroid they are closest to.

Next, we find the centroids of each of the colored groups. In practice, this is all done automatically by software, but we will work out one of the centroids explicitly here. Let's work out the green centroid.

Current green points: (75, 82.4), (75.5, 77.8), (108.6, 71.4), (109.2, 66.7), (110.5, 72), (119.8, 69.1).

Green centroid *x*-value: $\bar{x} = \frac{1}{6}(75 + 75.5 + 108.6 + 109.2 + 110.5 + 119.8) = 99.8$

Green centroid *y*-value: $\bar{y} = \frac{1}{6}(82.4 + 77.8 + 71.4 + 66.7 + 72 + 69.1) = 73.2$

Thus, the green centroid is located at (99.8, 73.2).

Now plot all the centroids. You might imagine the previous potential centroids have moved to positions that are closer to the true centroids of the visible groups. The red point moves to (107.7, 48.2); the green point moves to (99.8, 73.2); and the blue point moves to (41.1, 54.8). Note that (aside from the initial choices) the centroids are not typically points in the dataset. Recolor the points of the dataset with respect to these new centroids, as shown in Figure 6.11.

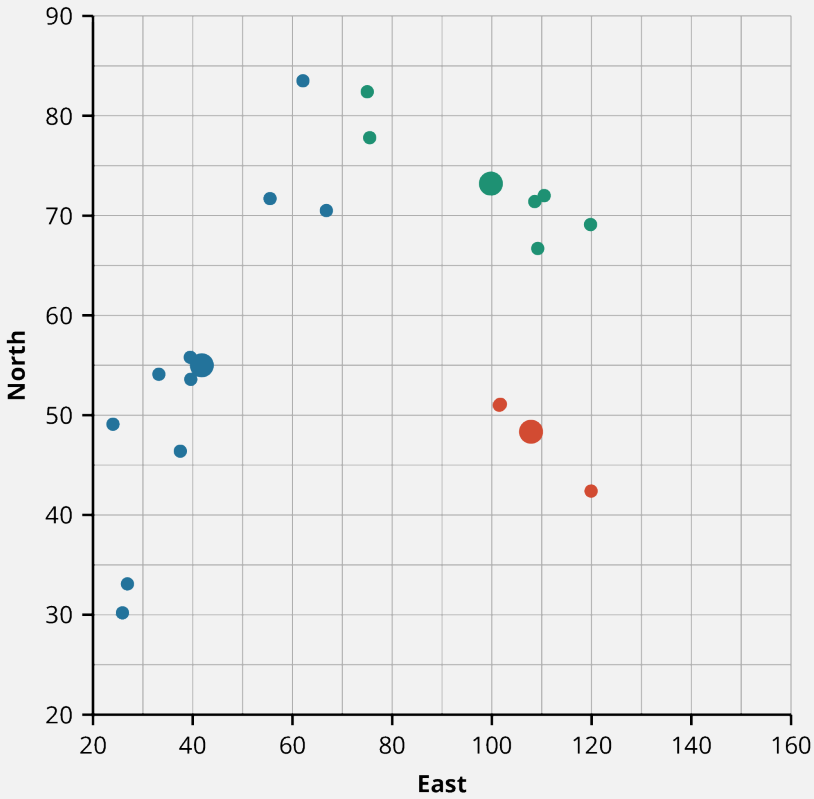


Figure 6.11 Graph with New Centroids. After the second iteration of the k-means algorithm, the centroids have moved closer to the centers of the main clusters. However, no points changed location, so the algorithm stabilizes at this step.

Run the algorithm a number of times until the coloring stabilizes. In this case, the next iteration is stable. Are the clusters where you thought they might end up? This example highlights the fact that k-means may be quite sensitive to initial choice of centroids.

Running the algorithm again with different initial centroids may produce different clusters, as Figure 6.12 shows.

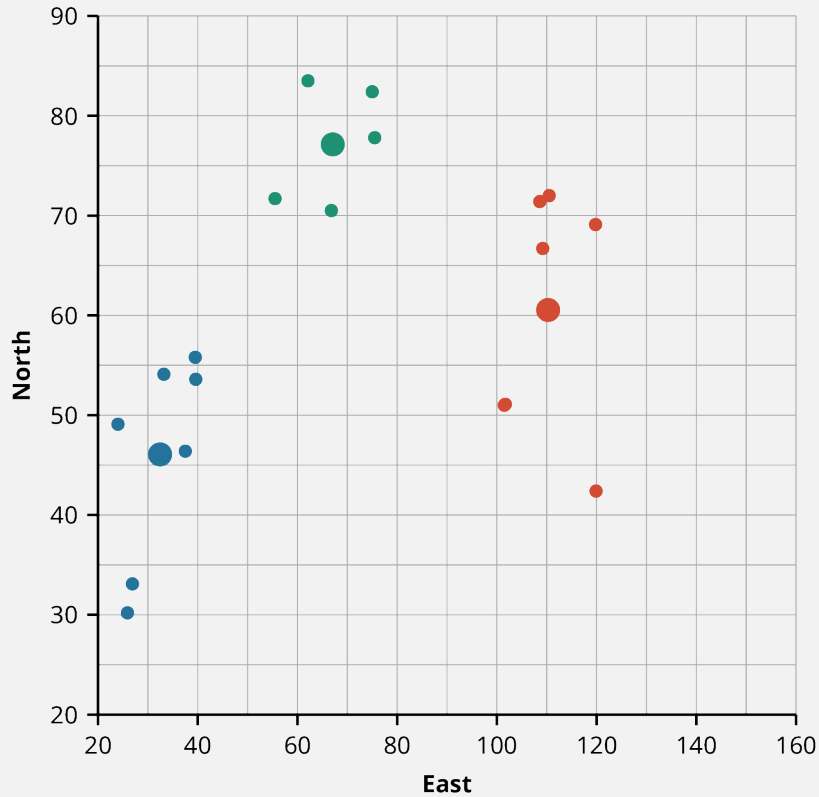


Figure 6.12 Graph with Different Initial Centroids. Using initial centroids of (110, 70), (60, 80), and (40, 40), the k-means algorithm produces a different clustering. The final position of the centroids are (110.2, 60.5), (67.0, 77.2), and (32.4, 46.0).

Clustering algorithms such as k-means also allow one to make predictions about new data. For example, suppose a new mushroom sprouted up at location (90, 50). To classify this point as red, green, or blue, simply find out which centroid is closest to the point.

Recall that the distance between two points (x_1, y_1) and (x_2, y_2) is found using the formula

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Distance to red centroid: $\sqrt{(90 - 110.2)^2 + (50 - 60.5)^2} = 22.77$

Distance to green centroid: $\sqrt{(90 - 67)^2 + (50 - 77.2)^2} = 35.62$

Distance to blue: $\sqrt{(90 - 32.4)^2 + (50 - 46)^2} = 57.74$

Thus, the new mushroom should be included in the red group.

Among clustering algorithms, k-means is rather simple, easy to implement, and very fast. However, it suffers from many drawbacks, including sensitivity to initial choice of centroids. Furthermore, k-means does detect clusters that are not simply "blobs." In the next section, we will discuss another clustering algorithm that generally performs better on a wider range of data.

k-Means Clustering in Python

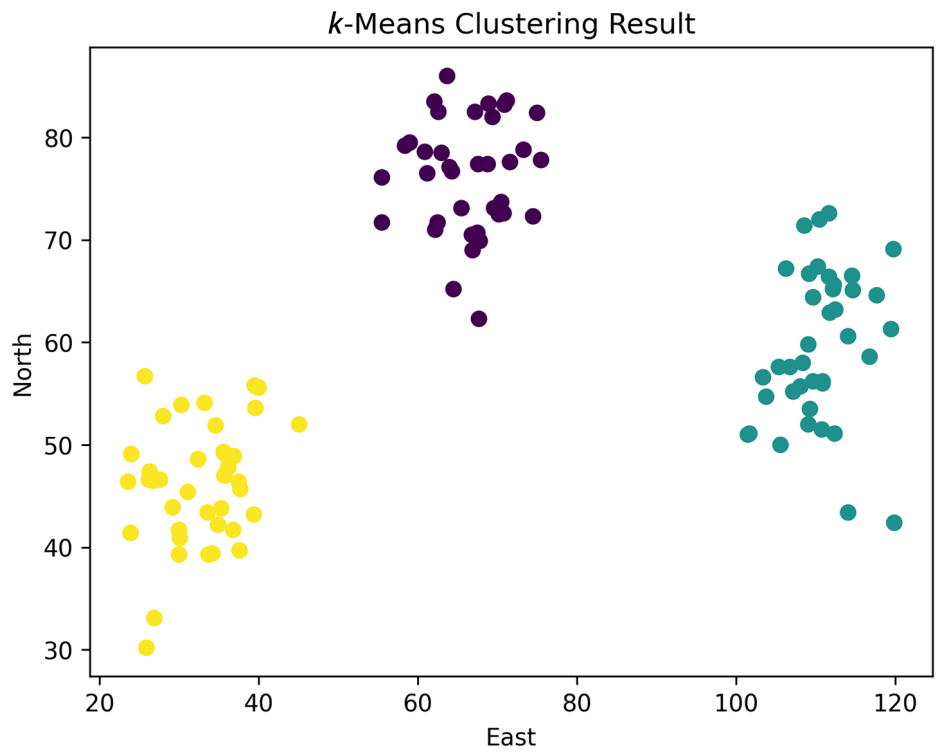
Our previous example used a sample of only 19 data points selected from the 115 data points found in the file FungusLocations.csv (https://openstax.org/r/FungusLocations). Here is the Python code to produce a k-means clustering from a dataset:

PYTHON CODE

```python
# Import libraries
import pandas as pd  ## for dataset management
import matplotlib.pyplot as plt ## for data visualization
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Read data
data = pd.read_csv('FungusLocations.csv').dropna()

# Build K-means model with 3 clusters
km = KMeans(n_clusters=3, n_init='auto')
km.fit(data)

# Calculate silhouette score
silhouette_avg = silhouette_score(data, km.labels_)
print("Silhouette Score:", round(silhouette_avg,2))

# Visualize the result of k-Means clustering using matplotlib
plt.scatter(data['East'], data['North'], c=km.labels_, cmap='viridis')
plt.xlabel('East')
plt.ylabel('North')
plt.title('k-Means Clustering Result')
plt.show()
```

The resulting output will look like this:

```
Silhouette Score: 0.77
```



With a silhouette score of about 0.77, the separation of data points into three clusters seems appropriate.

EXPLORING FURTHER

Centroids versus Other Means

Instead of using *centroids* (which are computed using arithmetic means, or averages), other means may be employed, including medians, trimmed averages, geometric or harmonic means, and many others. These algorithms may have different names such as *k-medians*, *k-medoids*, etc. More information can be found at Neptune (https://openstax.org/r/neptune) and Tidymodels articles (https://openstax.org/r/tidymodels)..

Density-Based Clustering (DBScan)

Suppose you need to classify the data in Figure 6.13 into two clusters.

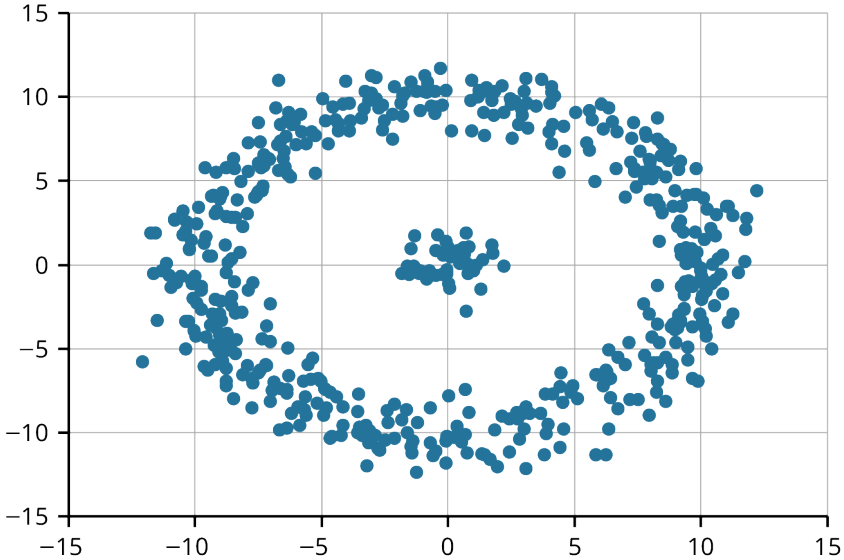


Figure 6.13 Dataset with an Inner and Outer Ring

How might you do this? There seems to be a clear distinction between points on the inner ring and points on the outer ring. While k-means clustering would not do well on this dataset, a **density-based clustering algorithm** such as DBScan might perform better.

DBScan Fundamentals

The **DBScan algorithm** (DBScan is an acronym that stands for *density-based spatial clustering of applications with noise*) works by locating core data points that are part of dense regions of the dataset and expanding from those cores by adding neighboring points according to certain closeness criteria.

First, we define two parameters: K, and r. A point in the dataset is called a core point if it has at least K neighbors, counting itself, within a distance of r from itself.

The steps of DBScan are as follows:

1. Choose a starting point that is a core point and assign it to a cluster. (Note: If there are no core points in the dataset at all, then the parameters K and r will need to be adjusted.)
2. Add all core points that are close (less than a distance of r) to the first point into the first cluster. Keep adding to the cluster in this way until there are no more core points close enough to the first cluster.
3. If there are any core points not assigned to a cluster, choose one to start a new cluster and repeat step 2 until all core points are assigned to clusters.
4. For each non-core point, check to see if it is close to any core points. If so, then add it to the cluster of the closest core point. If a non-core point is not close enough to any core points, it will not be added to any cluster. Such a point is regarded as an outlier or noise.

Since the values of K and r are so important to the performance of DBScan, the algorithm is typically run multiple times on the same dataset using a range of K values and r values. If there are a large number of outliers and/or very few clusters, then the model may be underfitting. Conversely, a very small number of outliers (or zero outliers) and/or many clusters present in the model may be a sign of overfitting.

Let's work through a small example using DBScan with $K = 3$ and $r = 2.1$ to classify the following dataset into clusters. (Points are labeled by letters for convenience.)

A: (3.5, 5.8) B: (4.5, 3.5) C: (5.0, 5.0) D: (6.0, 6.0) E: (6.0, 3.0) F: (6.9, 4.7)

G: (9.0, 4.0) H: (11.0, 5.0) J: (12.0, 4.0) K: (12.4, 5.4) L: (13.2, 2.5)

Note: In practice, DBScan would never be done by hand, but the steps shown as follows illustrate how the

algorithm works. Figure 6.14 displays the data.

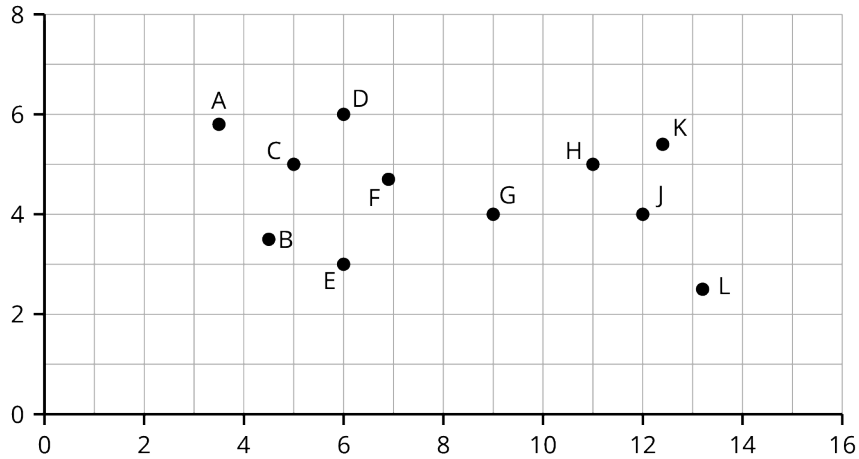


Figure 6.14 Original Dataset with 11 Unlabeled Points.

The core points are those that have at least K neighbors (or $K - 1$, not counting the point itself) within a distance of *r*. The easiest way to determine core points would be to draw circles of radius 2.1 centered at each point, as shown in Figure 6.15, and then count the number of points within each circle.

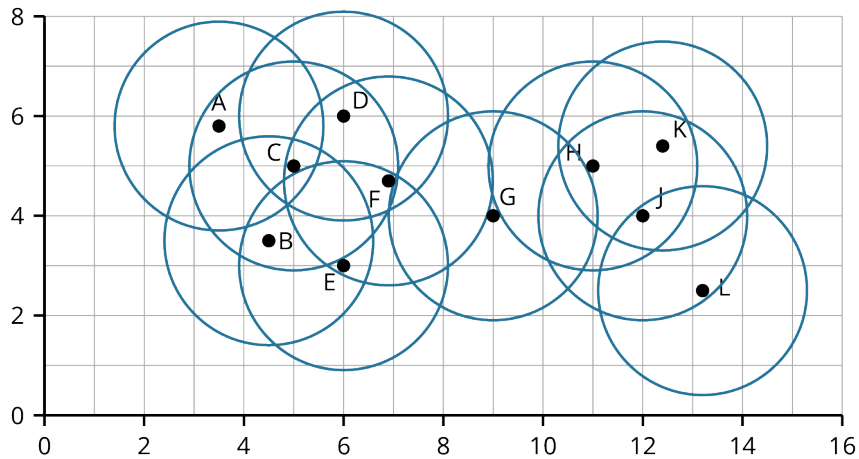


Figure 6.15 Circles of Radius Centered at Each Point. Drawing circles of radius *r* around each data point, it is easier to see which points are core and which are not core.

Based on Figure 6.15, point *A* has only two neighbors, *A* and *C*, so *A* is not a core point. On the other hand, point *B* has three neighbors, *B*, *C*, and *E*, so *B* is core. The full list of core points is *B*, *C*, *D*, *E*, *F*, *H*, *J*, and *K*. So *B* may be chosen to start the first cluster. Then, building from *B*, points *C* and *E* are added next. Next, *D*, *E*, and *F* are added to the growing first cluster. Neither *A* nor *G* are added at this stage since they are not core points. Having found all core points that are in the first cluster, let *H* start a second cluster. The only core points left are close enough to *H* that the second cluster is *H*, *J*, and *K*.

Finally, we try to place the non-core points into clusters. Point *A* is close enough to *C* to make it into the first cluster. Point *L* is close enough to *J* to become a part of the second cluster. Point *G* does not have any core points near enough, so *G* will be un-clustered and considered noise. Thus, cluster 1 consists of *{A, B, C, D, E, F}*, and cluster 2 consists of *{H, J, K, L}*. The results of DBScan clustering are shown in Figure 6.16.

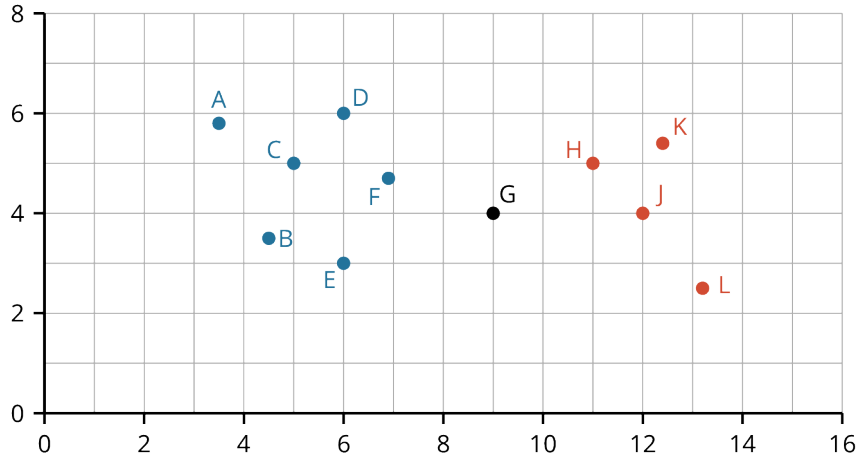


Figure 6.16 Graph Showing Results of DBScan Clustering. Cluster 1 is in blue and Cluster 2 is in red.

DBScan in Python

The Python library sklearn.cluster (https://openstax.org/r/stable) has a module named `DBSCAN`. Here is how it works on the dataset DBScanExample.csv (https://openstax.org/r/DBScanExample).

PYTHON CODE

```
# Import libraries
import pandas as pd ## for dataset management
from sklearn.cluster import DBSCAN

# Read data
data = pd.read_csv('DBScanExample.csv').dropna()

# Run DBScan
db = DBSCAN(eps=1.3, min_samples=5).fit(data)
labels = db.labels_

# Find the number of clusters and points of noise
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_noise = list(labels).count(-1)
print("Number of clusters: ", n_clusters)
print("Points of noise:", n_noise)
```

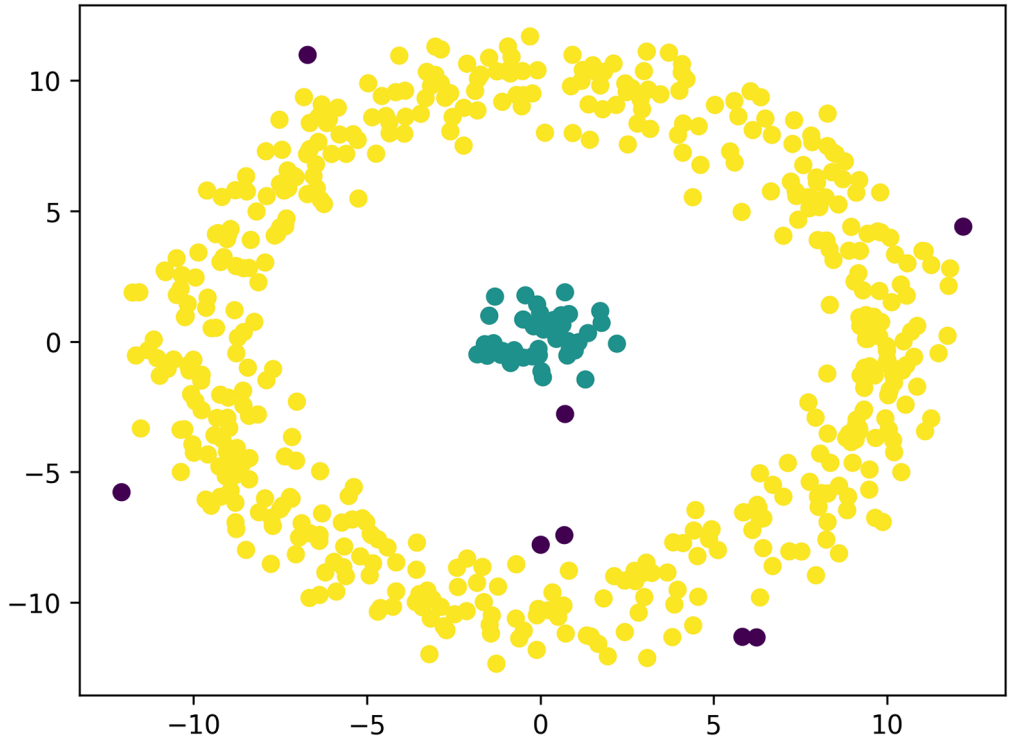
The resulting output will look like this:

```
Number of clusters: 2
Points of noise: 8
```

PYTHON CODE

```
# Visualize clusters
import matplotlib.pyplot as plt ## for data visualization

plt.scatter(data['X'], data['Y'], c=db.labels_)
```

The resulting output will look like this:



The Confusion Matrix

When training and testing any algorithm that performs classification, such as logistic regression models, k-means clustering, or DBScan, it is important to measure accuracy and identify error. The **confusion matrix** is one way of quantifying and visualizing the effectiveness of a classification model. This concept will be illustrated by the following example.

Suppose you have trained a model that classifies images of plants into one of three categories: flowers, trees, and grasses. When you test the model on 100 additional images of plants, you discover that it correctly identified flowers, trees, and grasses the majority of the time, but there were also quite a few mistakes. Table 6.6 displays the results in a confusion matrix:

	Identified as a Flower	Identified as a Tree	Identified as a Grass
Is a flower	23	3	9
Is a tree	2	32	0
Is a grass	12	1	18

Table 6.6 A Confusion Matrix for Flowers, Trees, and Grasses

Thus, we can see that 23 flowers, 32 trees, and 18 grasses were identified correctly, giving the model an accuracy of $\frac{23+32+18}{100} = 0.73$, or 73%. Note: The terms TP, TN, FP, and FN introduced in What Is Machine Learning? and defined again in the next example do not apply when classifying data into three or more categories; however, the confusion matrix gives more detailed information about the mistakes. Where there are relatively low numbers off the main diagonal, the model performed well. Conversely, higher numbers off the main diagonal indicated greater rates of misidentification, or confusion. In this example, when presented with a picture of a flower, the model did a pretty good job of identifying it as a flower but would mistake flowers for grasses at a higher rate than mistaking them for trees. The model did very well identifying trees as trees, never mistaking them for grasses and only twice mistaking them for flowers. On the other hand, the model did not do so well identifying grasses. Almost 40% of the images of grasses were mistaken for flowers! With this information in hand, you could go back to your model and adjust parameters in a way that may address these specific issues. There may be a way to tweak the model that helps with identifying grasses in particular. Or you may decide to take extra steps whenever an image is identified as a flower, perhaps running the image through additional analysis to be sure that it truly is a flower and not just a mislabeled grass.

When there are only two classes (binary classification), the confusion matrix of course would have only four entries. There are special terms that apply to this case when the two classes are “Positive” and “Negative.” Think of diagnosing a disease. Either the patient has the disease or they do not. A doctor can perform a test, and the hope is that the test will determine for sure whether the patient has that disease. Unfortunately, no test is 100% accurate. The two cases in which the test fails are called false positive and false negative. A false positive (also called type I error) occurs when the true state is negative, but the model or test predicts positive. A false negative (also called type II error) is just the opposite: the state is positive, but the model or test predicts a negative. Both errors can be dangerous in fields such as medicine. Some terminology for the four possibilities are shown in Table 6.7.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP) – hit	False Negative (FN) – miss
Actual Negative	False Positive (FP) – false alarm	True Negative (TN) – correctly rejected

Table 6.7 Terminology Summarizing the Four Possibilities from a Confusion Matrix

Adjustments to a model that reduce one type of error generally increases the rate of the other type of error. That is, if you want your model to have a very low rate of false negatives, then it may become positively biased (more sensitive), predicting more positives whether they are true or false positives. On the other hand, if the rate of false positives needs to be reduced, then the model may become negative biased (less sensitive) and yield more negative predictions overall, both true and false negatives.

Visualizing Confusion Matrices

Confusion matrices are often shaded or colored in a way to show contrast of high and low values, which is helpful in locating any abnormal behavior in the model, often called a **heatmap**. Figure 6.17 provides a heatmap for the flower/tree/grass example.

	Flower	Tree	Grass
Flower	23	3	9
Tree	2	32	0
Grass	12	1	18

Figure 6.17 Heatmap for Flower/Tree/Grass Example

The darker shades indicate higher values. The main diagonal stands out with darker cells. This is to be expected if our classifier is doing its job. However, darker shades in cells that are off the main diagonal indicate higher rates of misclassification. The numbers 12 in the lower left and 9 in the upper right are prominent. The model clearly has more trouble with flowers and grasses than it does with trees.

Generating a Confusion Matrix in Python

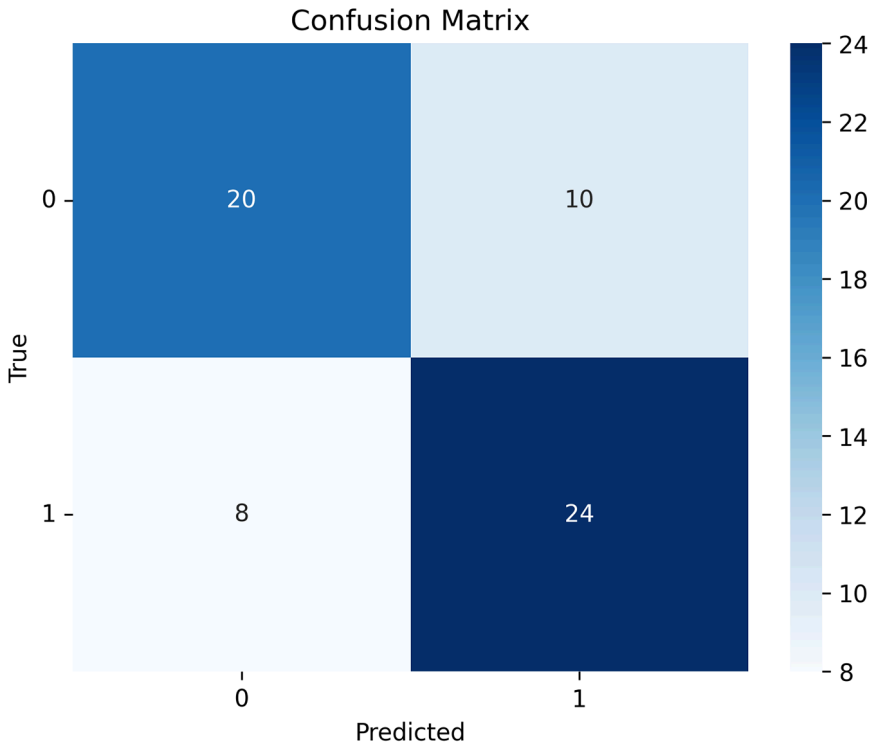
The dataset CollegeCompletionData.csv (https://openstax.org/r/CollegeCompletionData) contains 62 data points (from which only 12 points of data were used in Table 6.6). The following code produces a logistic regression based on all 62 points. Then, the confusion matrix is generated. Finally, a visual display of the confusion matrix as a heatmap is generated using another visualization library called seaborn (https://openstax.org/r/seaborn). If you do not need the heatmap, then just type print(cf) directly after finding the confusion matrix to print a text version of the matrix.

PYTHON CODE

```python
# Import libraries
import pandas as pd  ## for dataset management
import matplotlib.pyplot as plt ## for data visualization
import seaborn as sns ## for heatmap visualization
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Read data
data = pd.read_csv('CollegeCompletionData.csv').dropna()
x = data[['GPA']]
y = data['Completion']

# Build the logistic model
model = LogisticRegression()
model.fit(x,y)

# Generate model predictions
y_pred = model.predict(x)

# Generate the confusion matrix
cf = confusion_matrix(y, y_pred)

# Plot the heatmap using seaborn and matplotlib
sns.heatmap(cf, annot=True, fmt='d', cmap='Blues', cbar=True)
```

```
plt.xlabel('Predicted')
plt.ylabel('True')
plt.yticks(rotation=0)
plt.title('Confusion Matrix')
plt.show()
```

The resulting output will look like this:



From the preceding analysis, we find 20 true positives and 24 true negatives, representing an accuracy of $44/62 = 71\%$. There were 10 false negatives and 8 false positives.

6.3 Machine Learning in Regression Analysis

Learning Outcomes

By the end of this section, you should be able to:

- 6.3.1 Use bootstrapping to analyze variation in linear regression.
- 6.3.2 Outline assumptions relevant to a multiple linear regression.
- 6.3.3 Perform multiple linear regressions and analyze significance of coefficients.

Regression is a term that applies to many different techniques in data analysis and machine learning. The main idea is to model a relationship between one or more input (or independent) variables X and an output (or dependent) variable *y* by creating an explicit functional equation $y = f(X)$. Once the equation is found, new inputs can be plugged into it and the results would serve as predictions. The types of regressions are based on the form of the equations that are generated: linear, polynomial, logistic, exponential, etc. Most of these types of regression analyses may be done on single inputs or using multiple input variables. Some of these topics have already been discussed in this textbook, and so in this chapter, we delve into a more advanced analysis. First we explore *bootstrapping* as a way to find more information about the reliability and variability of the parameters of a linear regression. Then we discuss multiple linear and logistic regressions,

including how to perform these tasks in Python.

Linear Regression and Bootstrapping

The fundamentals of linear regression were presented in Inferential Statistics and Regression Analysis. While linear regression is a very powerful tool, it may suffer from inaccuracies due to noisy data. How confident can we be in the parameters of a linear model? In this section we use a resampling technique called **bootstrapping**, which is aimed at estimating variation to build a linear regression model with parameters within a specified confidence interval. Recall that we defined bootstrapping in Bootstrapping Methods as a technique in which repeated samples are taken hundreds or thousands of times and then the sample mean is calculated for each sample.

Bootstrapping Fundamentals

Recall, for a set of bivariate data—that is, a set of points $X = \{(x_i, y_i)\}$ in the plane—the linear regression or line of best fit is a line with equation $y = a + bx$, in which the values of a and b are chosen to minimize the squared vertical distance from each y_i to the predicted value, $\hat{y}_i = a + bx_i$. The values of a and b clearly depend on the points of the dataset X, and so these parameters are just our best guess estimates for the "true" values of a and b. Perhaps a different sampling of X would yield slightly different values of a and b. We may not have the luxury of resampling from the population, and so we do the next best thing: Resample from the set X itself! This process is called bootstrapping because you are attempting to "pull yourself up by your own bootstraps," as the old saying goes. Here is an outline of the process.

1. Choose a large number N of samples taken from the dataset X, with replacement (so repetitions of data are allowed).
2. Find the linear regression for this set of N samples. Record the a (intercept) and b (slope) values into lists A and B, respectively.
3. Repeat steps 1 and 2 many times.
4. The sets A and B represent distributions of possible values of the parameters a and b for the true linear fit, $y = a + bx$.
5. Let $\bar{a} = MEAN(A)$ and $\bar{b} = MEAN(B)$. These are the parameters we will chose for our (bootstrapped) linear regression, $y = \bar{a} + \bar{b}x$.
6. Find the standard deviation of A and B as well in order to produce a confidence interval for each parameter.

These steps will next be illustrated with the help of Python.

Linear Regression with Bootstrapping in Python

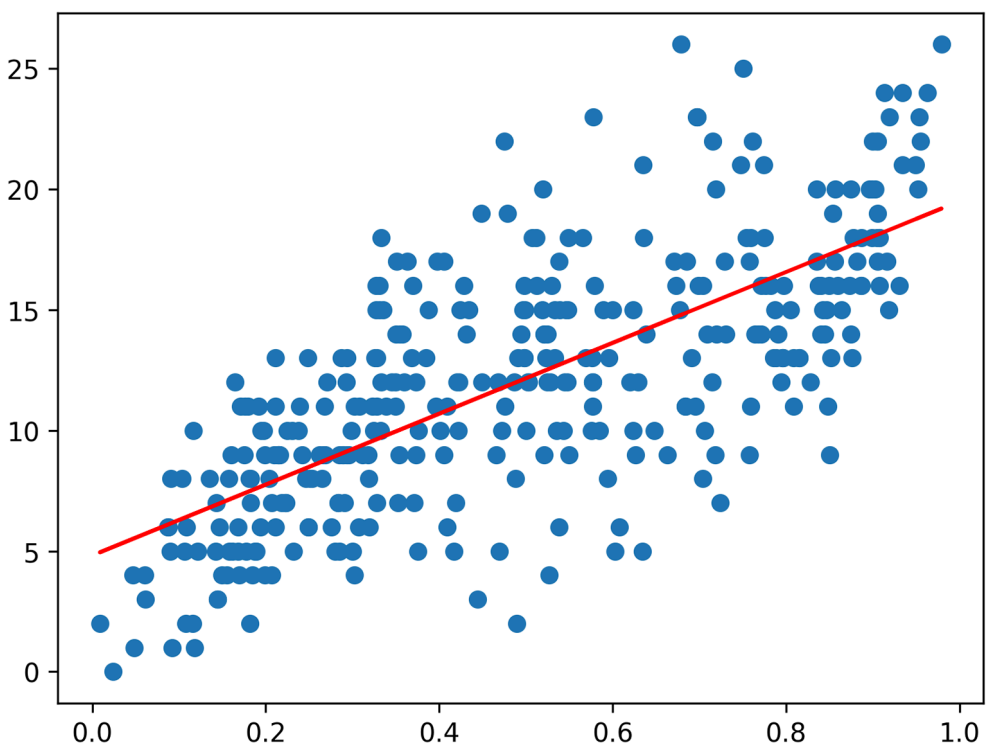
In NCAA Division I basketball, a statistic called BARTHAG provides a measure of many different factors that affect a team's chances of winning against an average team in the same division. This statistic (along with many others) can be found in the dataset NCAA-2021-stats.csv (https://openstax.org/r/NCAA). Let X be the dataset consisting of pairs (BARTHAG, W), where W stands for the number of wins in 2021. There are 346 data points in this set. Here is the code that produces the scatterplot and regression line.

PYTHON CODE

```
# Import libraries
import pandas as pd  ## for dataset management
import matplotlib.pyplot as plt ## for data visualization
from sklearn.linear_model import LinearRegression
```

```
# Read data
data = pd.read_csv('NCAA-2021-stats.csv')
data = data[['BARTHAG', 'W']].dropna()

# Find the linear regression model
model = LinearRegression()
x = data['BARTHAG'].values.reshape(-1,1)
y = data['W'].values.reshape(-1,1)
model.fit(x,y)
y_pred = model.predict(x)

# Visualizing the linear regression on the scatterplot
plt.scatter(x,y)
plt.plot(x, y_pred, color='red')
plt.show()

a = model.intercept_
b = model.coef_
print("Linear model: y = %1.3f + %2.3f x" % (a, b))
```

The resulting output will look like this:




```
Linear model: y = 4.820 + 14.682 x
```

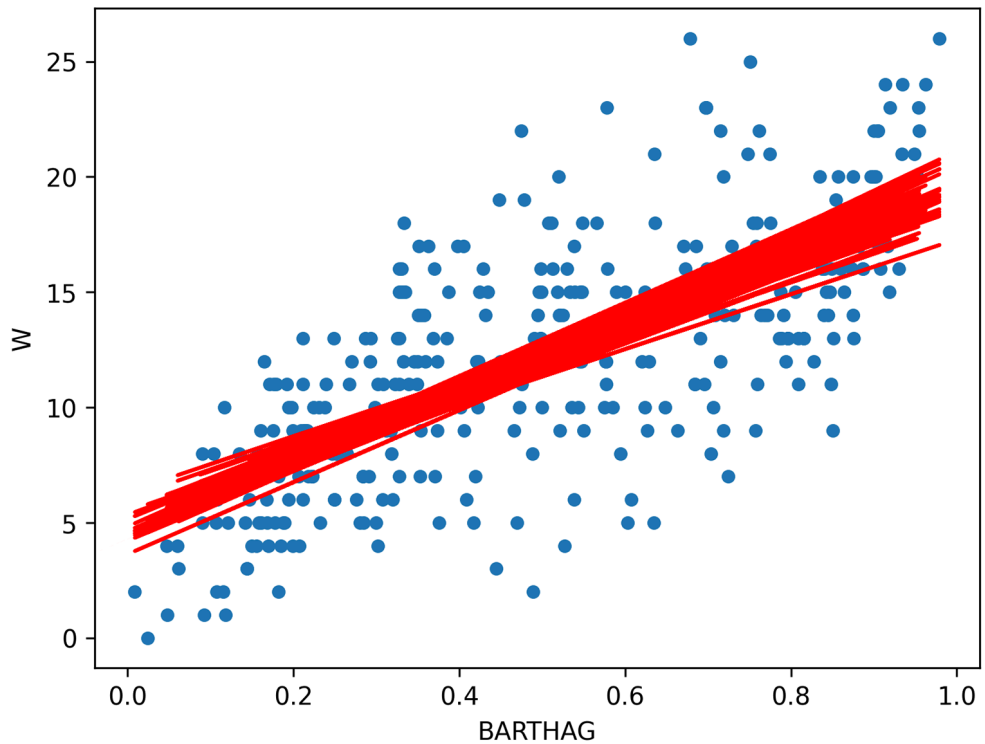
Linear model: y = 4.820 + 14.682 x

Then, the set is resampled 50 times, using 100 data points each time. A linear regression is run each time, producing a set of 50 values of a and 50 values of b. All 50 regression lines are shown on the same graph.

PYTHON CODE

```
from sklearn.utils import resample

data.plot.scatter(x='BARTHAG', y='W')

# Build arrays for slopes and intercepts
slopes = []
intercepts = []

# Resampling 100 points 50 times
for i in range(50):
  data1 = resample(data, n_samples=100, replace=True)

  x = data1['BARTHAG'].values.reshape(-1,1)
  y = data1['W'].values.reshape(-1,1)
  model.fit(x,y)
  y_pred = model.predict(x)

  slopes.append(model.coef_[0][0])
  intercepts.append(model.intercept_[0])

  plt.plot(x, y_pred, color='red')

plt.show()
```

The resulting output will look like this (results for bootstrapping models may slightly vary):



Note how all the lines are very close to one another, but there is some variation. The mean and standard deviation of intercepts and slopes may be found using the [statistics (https://openstax.org/r/docspython1)](https://openstax.org/r/docspython1) library in Python.

PYTHON CODE

```
import statistics as stats # for mean and standard deviation

a_bar = stats.mean(intercepts)
a_std = stats.stdev(intercepts)
b_bar = stats.mean(slopes)
b_std = stats.stdev(slopes)

print("Intercept: Mean = %1.3f, St.Dev. = %1.3f " % (a_bar, a_std) )
print("Slope: Mean = %1.3f, St.Dev. = %1.3f " % (b_bar, b_std) )
```

The resulting output will look like this:

```
Intercept: Mean = 4.947, St.Dev. = 0.645
Slope: Mean = 14.489, St.Dev. = 1.267
```

Recall that, before bootstrapping, we obtained a linear model with intercept 4.82 and slope 14.682. These are very close to the intercept and slope obtained from the bootstrapping method. The advantage to bootstrapping, though, is that we now have measures of spread (standard deviation), and so we can produce a

confidence interval for both parameters. Recall that the 95% confidence interval is found at plus or minus $(1.96)\,\sigma/\sqrt{N}$, where σ is the standard deviation (See Bootstrapping Methods). This can be done in Python using the `scipy.stats` package.

PYTHON CODE

```
import scipy.stats as st
import numpy as np

a_min, a_max = st.t.interval(0.95, df=len(intercepts)-1, loc=np.mean(intercepts),
scale=st.sem(intercepts))
b_min, b_max = st.t.interval(0.95, df=len(slopes)-1, loc=np.mean(slopes),
scale=st.sem(slopes))

print("95%% confidence interval for intercept: %1.2f < a < %1.2f " % (a_min, a_max)
)
print("95%% confidence interval for slope: %1.2f < b < %1.2f " % (b_min, b_max) )
```

The resulting output will look like this:

```
95% confidence interval for intercept: 4.76 < a < 5.13
95% confidence interval for slope: 14.13 < b < 14.85
```

Thus, we expect that the true value of the intercept is between 4.76 and 5.13 with 95% probability, and the true value of the slope is between 14.13 and 14.85 with 95% probability.

Multiple Regression Techniques

In a simple linear regression, one variable (x) serves as the predictor for the output variable (y). Many problems in data science, however, go well beyond a single predictor. There may be dozens of features in the data that work together to predict the output. In this context, we may use **multiple regression** techniques, which use more than one input variable. In this section, we focus on two main concepts: multiple linear regression and multiple logistic regression.

Multiple Linear Regression

A model for multiple linear regression involves some number of predictor variables, which we will label $X_1, X_2, \ldots X_r$. Here, we are using capital letters to distinguish these feature variables from data values like x_i. The multiple regression model takes the form

$$y = a + b_1 X_1 + b_2 X_2 + \ldots + b_r X_r$$

In the equation, a is the intercept (value of y when all $X_k = 0$), and the values of b_k are the coefficients of the predictors, which function much like the slope b in the simple linear regression model. A positive b_k indicates that the feature measured by X_k has a positive correlation on values of y, while a negative b_k indicates a negative correlation. More sophisticated models also introduce an error term, E, to account for unexplained variability in y.

The goal of multiple linear regression is to determine (or estimate) values of the parameters a and b_k (for $k = 1, 2, \ldots, r$) that minimize the squared vertical distance between the actual and predicted outputs. There are well-known formulas for this task, and typically the regression is handled by a computer.

There are certain assumptions that need to be checked in order to use multiple regression. The first two are about the nature of the input and output data itself. Generally, these data must be continuous.

1. The output y should be continuous, not discrete. However, regression could still be used if the output is a discrete variable with a very large, regular range of values (such as the number of people who live in a given country, which could be any positive integer).
2. The input variables, X_k, should be continuous, not discrete. However, the same comment as in (1) applies here: If the variable has a very large, regular range of possible values, then regression should work fine.

The next two assumptions ensure that multiple regression would provide the best model. If these assumptions are not met, then a more flexible model would be required.

3. There should be a linear relationship between X_k and for each k, regarding the other variables as constant. This can be checked by running simple linear regressions on the pair (X_k, y) for each k and verifying linearity.
4. The data should not have multicollinearity. Multicollinearity means that at least two of the input variables X_j, X_k are highly correlated with one another, in which case one (or more) of the variables can be dropped.

Finally, there are a number of technical assumptions that should be verified, including checking for outliers, making sure that the variance is more or less uniform at all parts of the domain (this is known as *homoscedasticity*), and analyzing the residuals to determine if they are fairly independent and normally distributed. More advanced analysis might result in the determination that some variables are not significant in the model. These further considerations fall outside the scope of this text, however.

Multiple Linear Regression in Python

In this section, we work through the Python code for multiple linear regression using the dataset NCAA-2021-stats.csv (https://openstax.org/r/NCAA). As in Machine Learning in Regression Analysis, we would like to predict the number of wins (W), but now we will include more than one predictor feature. In the first code block, we will use BARTHAG and WAB (which is known as *wins above bubble*, a measure of how many more wins this team has in comparison to a typical team that just makes it into the tournament on a similar schedule) to predict W.

PYTHON CODE

```python
# Import libraries
import pandas as pd  ## for dataset management
import matplotlib.pyplot as plt ## for data visualization
from sklearn.linear_model import LinearRegression

# Read data
data = pd.read_csv('NCAA-2021-stats.csv')
data = data.drop(['SEED'], axis=1).dropna() ## Remove the SEED column

X = data[['BARTHAG', 'WAB']]  # Feature list
y = data[['W']]  # Response variable

# Create the multiple linear regression model
model = LinearRegression()
```

```
model.fit(X, y)

b = model.intercept_[0]
a1, a2 = model.coef_[0]

print("Intercept: ", round(b,3))
print("Coefficients: ", [round(a1,3), round(a2,3)])
print("Score: ", round(model.score(X,y),2))
```

The resulting output will look like this:

```
Intercept: 12.868
Coefficients:  [4.794, 0.545]
Score:  0.58
```

According to the output, the multiple regression equation for this model is:

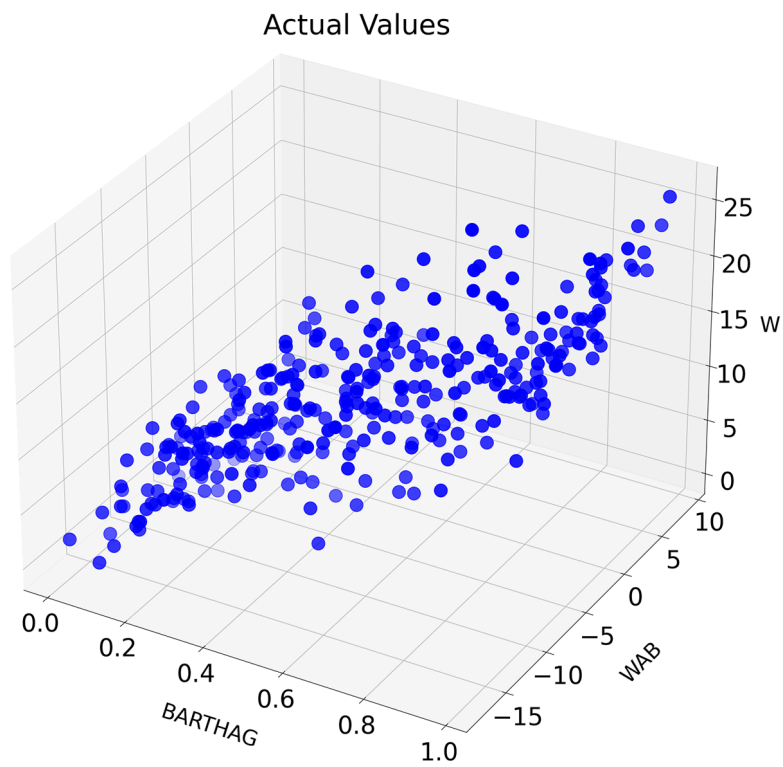
$$W = 12.868 + 4.794 \times (\text{BARTHAG}) + 0.545 \times (\text{WAB})$$

We can see that both BARTHAG and WAB have positive coefficients, and so as either variable increases, there is a corresponding increase in W. The R-squared score indicates that about 58% of the total variance in W is due to BARTHAG and WAB jointly. Let's take a look at the data and see how well the points conform to the regression plane. First, here is the code to generate the scatterplot of points (BARTHAG, WAB, W) in three dimensions. (Make sure that you have already run the previous code beforehand.)

PYTHON CODE

```
# import libraries for graphing
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

X1 = X[['BARTHAG']]
X2 = X[['WAB']]

# Create a 3D figure
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatterplot of actual data points
ax.set_xlabel('BARTHAG')
ax.set_ylabel('WAB', rotation=45)
ax.set_zlabel('W')
ax.set_title('Actual Values')
ax.set_box_aspect(aspect=None, zoom=0.85)
ax.scatter(X1, X2, y, c='b', marker='o')
```

The resulting output will look like this:



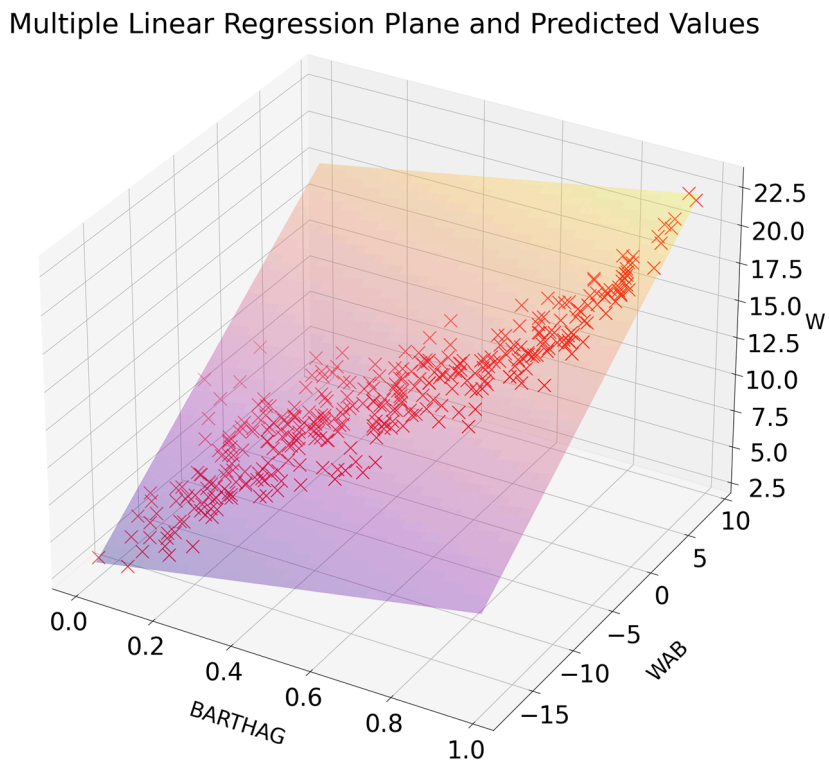
Next, let's take a look at the predicted values found by plugging BARTHAG and WAB into the multiple regression formula discussed previously in the chapter. We also include the regression plane. The Python code required to create the regression plane is somewhat involved, requiring another package, numpy (https://openstax.org/r/numpy), to help create a mesh grid of (x, y) points, so don't worry too much if you don't understand that part of the code.

PYTHON CODE

```
import numpy as np

# Create a 3D figure
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Generate prediction values using the model
y_pred = model.predict(X)

# Create a grid of points for the regression plane
X1min = min(X1.values)[0]
X1max = max(X1.values)[0]
X2min = min(X2.values)[0]
X2max = max(X2.values)[0]
X1_space = np.linspace(X1min, X1max, 100)
```

```
X2_space = np.linspace(X2min, X2max, 100)
x1, x2 = np.meshgrid(X1_space, X2_space)

# Use the regression equation to find z-values
z = b + a1 * x1 + a2 * x2

# Plot the regression plane
ax.set_xlabel('BARTHAG')
ax.set_ylabel('WAB', rotation=45)
ax.set_zlabel('W')
ax.set_title('Multiple Linear Regression Plane and Predicted Values')
ax.set_box_aspect(aspect=None, zoom=0.85)
ax.scatter(X1, X2, y_pred, c='r', marker='x')
ax.plot_surface(x1, x2, z, alpha=0.3, cmap='plasma')
```

The resulting output will look like this:



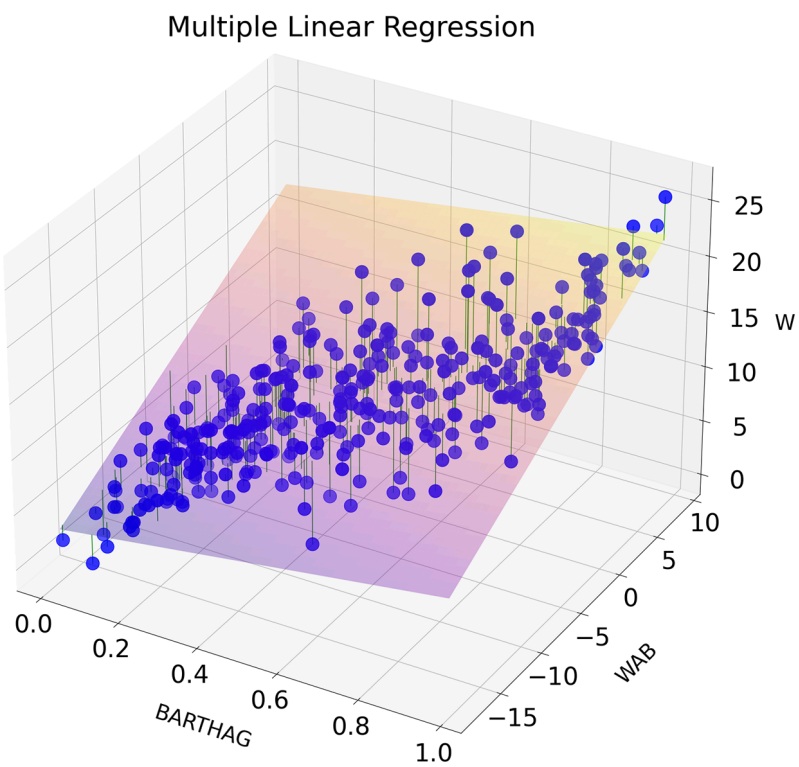
It may not be entirely clear from the graph, but each red “x” is a point on the regression plane. To see how well the regression formula does at modeling the actual data, we will remove the predicted values and put in the actual values together with line segments showing how close those values are to the plane.

PYTHON CODE

```
# Create a 3D figure
```

```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Add line segments from actual data points to predicted points
for i in range(len(X1)):
    ax.plot([X1.values[i,0], X1.values[i,0]],
     [X2.values[i,0], X2.values[i,0]],
      [y.values[i,0], y_pred[i,0]], c='g', linewidth=1)

# Plot the regression plane
ax.set_xlabel('BARTHAG')
ax.set_ylabel('WAB', rotation=45)
ax.set_zlabel('W')
ax.set_title('Multiple Linear Regression')
ax.set_box_aspect(aspect=None, zoom=0.85)
ax.scatter(X1, X2, y, c='b', marker='o')
ax.plot_surface(x1, x2, z, alpha=0.3, cmap='plasma')
```

The resulting output will look like this:



Now, let's see if we can improve the model by adding more features. Let's add TOR (turnover percentage), 2P_O (two-pointer shooting percentage), and 3P_O (three-pointer shooting percentage).

PYTHON CODE

```
X = data[['BARTHAG', 'WAB', 'TOR']]  # Feature list
y = data[['W']]  # Response variable

# Create the multiple linear regression model
model = LinearRegression()
model.fit(X, y)

b = model.intercept_[0]
a1, a2, a3 = model.coef_[0]

print("Intercept: ", round(b,3))
print("Coefficients: ", [round(a1,3), round(a2,3), round(a3,3)])
print("Score: ", round(model.score(X,y),2))
```

The resulting output will look like this:

```
Intercept:  16.116
Coefficients:  [4.502, 0.526, -0.169]
Score:  0.59
```

The formula now expands to:

$$W = 16.116 + 4.502 \times (\text{BARTHAG}) + 0.526 \times (\text{WAB}) - 0.169 \times (\text{TOR})$$

The R-squared measure has increased slightly to 0.59, meaning that the new model explains a bit more of the variance. However, the small increase in prediction accuracy comes at a cost of adding complexity to the model. Simpler models tend to do better at avoiding overfitting in practice, so we will stick to the original two feature variables for this example.

Multiple Logistic Regression

As the name suggests, multiple logistic regression is an extension of the single-variable logistic regression method to two or more input variables. The model estimates the probability of the dependent variable (y) taking on a particular category such as yes or no. Thus, the model is a binary classifier with multiple independent inputs. The standard multiple logistic regression model takes the form:

$$\text{logit}\,(y) = a + b_1 X_1 + b_2 X_2 + \ldots + b_r X_r$$

Here, the logit function is the same as described in section 6.2.1. Multiple logistic regression works just like single logistic regression in the sense that the logit function is used to transform the data, a (multiple) linear regression is performed, and the results are transformed back by a logistic function. Note that the features may be continuous numerical data or discrete data, including Boolean (1 or 0 for True/False or Yes/No).

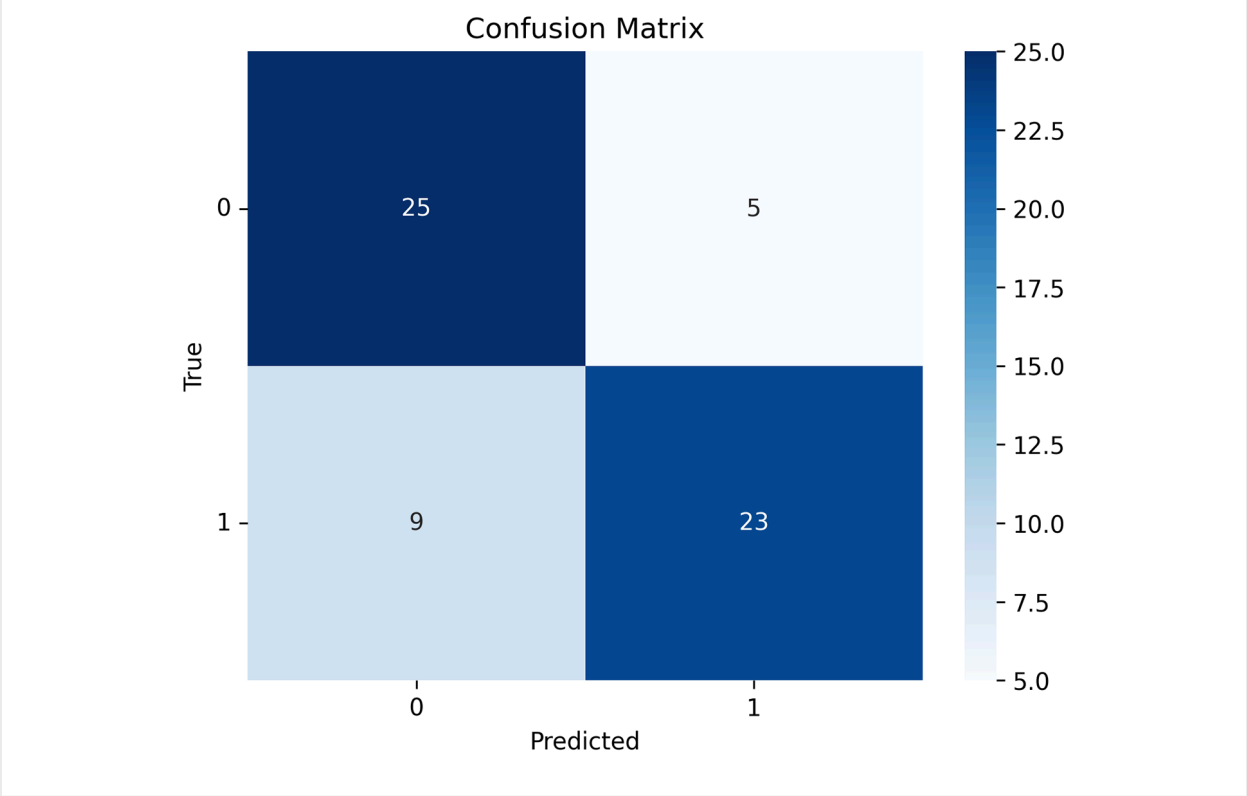
Multiple Logistic Regression in Python

Revisit the dataset CollegeCompletionData.csv (https://openstax.org/r/CollegeCompletionData) from The Confusion Matrix. It has two feature columns, “GPA” and “In-State.” Last time, we only used one feature to predict completion. It is just as easy to use multiple features!

PYTHON CODE

```python
# Import libraries
import pandas as pd  ## for dataset management
import matplotlib.pyplot as plt ## for data visualization
import seaborn as sns ## for heatmap visualization
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Read data
data = pd.read_csv('CollegeCompletionData.csv').dropna()
X = data[['GPA','In_State']]
y = data['Completion']

# Build the logistic model
model = LogisticRegression()
model.fit(X,y)

# Generate model predictions
y_pred = model.predict(X)

# Generate the confusion matrix
cf = confusion_matrix(y, y_pred)

# Plot the heatmap using seaborn and matplotlib
sns.heatmap(cf, annot=True, fmt='d', cmap='Blues', cbar=True)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.yticks(rotation=0)
plt.title('Confusion Matrix')
plt.show()
```

The resulting output will look like:



Recall that the rows of the confusion matrix are labeled by actual values while the columns are labeled by predicted values. Thus, there are 25 true positives (TP) and 23 true negatives (TN), increasing the accuracy to $48/62 = 77\%$. The number of false negatives (FN) dropped from 10 down to 5, but the number of false positives (FP) increased by 1, which is not a bad trade-off.

6.4 Decision Trees

Learning Outcomes

By the end of this section, you should be able to:

- 6.4.1 Measure information using entropy.
- 6.4.2 Understand how entropy can be used to classify data.
- 6.4.3 Build a top-down decision tree classifier.
- 6.4.4 Use decision trees to classify data and make predictions.

How do you make decisions? Often, we choose one product over another based on a combination of features. For instance, imagine you are considering purchasing a new smartphone. You might compare different models based on criteria such as camera quality, battery life, processing speed, and price. If there are many options available, you might organize your decision process into a *flow chart* or *tree*.

As a simple example, consider the game of 20 Questions. One person thinks up a person, place, or thing, and the others ask questions in turn to try to narrow down the space of possible answers until there is enough information to identify the object. Suppose that there are only five objects to choose from (a very simple game of 20 Questions indeed!): Tree, Dog, Human, Rock, and Diamond. The diagram in Figure 6.18, which may be called a *decision tree*, shows the questions and responses that would identify all of these items.

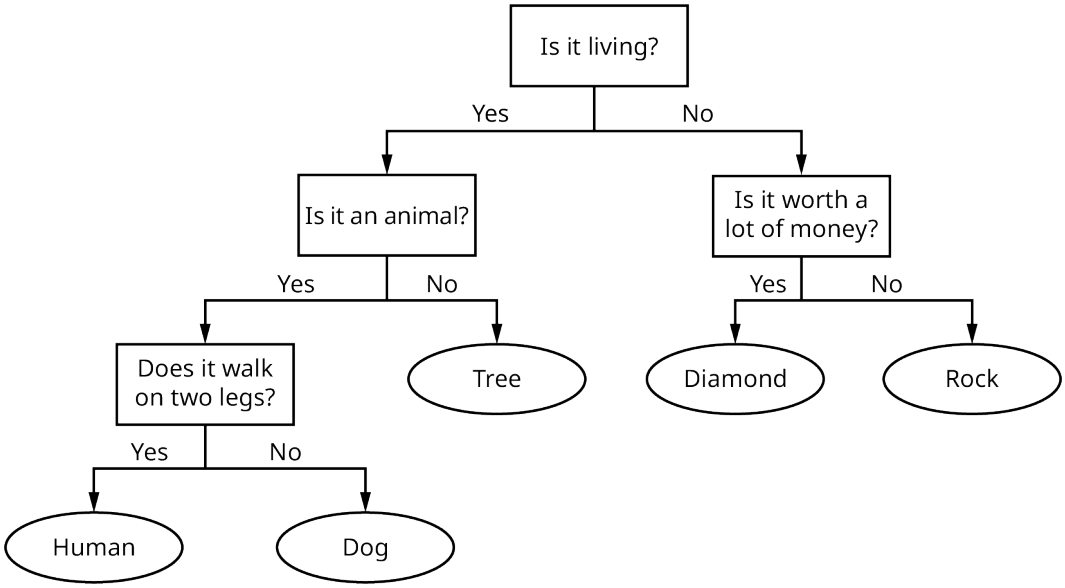


Figure 6.18 A Simple Decision Tree

In machine learning, a **decision tree** is an algorithm used for both classification and regression tasks, offering a visual and intuitive approach to solving complex problems using treelike structures to keep track of decisions based on the features of the dataset. Decision trees combine simplicity and flexibility in data analysis. They are simple structures to understand and create, inspired by the way our minds make choices. Decision trees break down decisions into a series of straightforward choices, much like following a trail of breadcrumbs through a dense forest. Their flexibility lies mainly in their nonlinear nature. Instead of trying to fit data to a linear model or some other rigid structure, the decision tree classifies data using a series of choices that depend on other choices, which may in turn depend on other choices, etc.

In this section, we will introduce information theory and entropy—a measure of information that is useful in constructing and using decision trees, illustrating their remarkable power while also drawing attention to potential pitfalls.

Information and Entropy

Information theory provides a framework for measuring and managing the uniqueness of data, or the degree of surprise or uncertainty associated with an event or message. For example, if I want to send a message consisting entirely of the letter "A" repeated 1,000 times, I do not have to send 1,000 "A"s. In fact, the phrase "1,000A" does the trick in only 5 characters (assuming the recipient of the message interprets it correctly). A related concept, *entropy*, is a measure of the *amount* of information in a message, structure, or system, as we will explore in the next section.

Information

Consider an event that has a certain probability p. The higher p is, the more likely the event is, and hence there is not much information (or uncertainty) contained in the occurrence of the event. For example, the event that my car starts when I get into it in the morning has a fairly low amount of information because my car is reliable, and I expect it to start. If, on the other hand, it does not start one morning, then that low-probability event would be surprising and would contain much more information.

Information is typically measured in bits, where one bit represents the amount of information needed to distinguish between two equally likely events or messages. For example, when you flip a fair coin, you gain one bit of information because you go from being uncertain about the outcome (heads or tails) to knowing the outcome (heads, for instance). To measure the number of bits of a whole number, you would use the base-2 logarithm ($\log_2 x$), so it may not come as a surprise to learn that the base-2 logarithm plays a huge role in

measuring information.

The information of an event with probability p is $-\log_2 p$. Recall that probabilities p are always between 0 and 1, and if $0 < p < 1$, then by properties of logarithms, we would have $\log_2 p < 0$. This explains the negative sign that appears in the formula, as it makes sense to measure information with a positive number.

Note that the information contained in an event that is certain to occur $(p = 1)$ is $-\log_2 1 = 0$, while the information contained in knowing that a coin was flipped and heads came up $(p = 1/2)$, would be $-\log_2\left(\frac{1}{2}\right) = -(-1) = 1$, or one bit of information.

Let's say the probability that my car will start on any given day is $p = 255/256$, and so the probability that it will not start is $1 - p = 1/256$. The information of the event that it starts is $-\log_2\left(\frac{255}{256}\right) = 0.0056$, whereas the information of the event that it does not start is $-\log_2\left(\frac{1}{256}\right) = 8$. Essentially, there are 8 bits of information contained in the knowledge that my car does not start because this would only have happened about once in every $2^8 = 256$ days.

Entropy

Entropy is a measure that quantifies the average amount of information, or uncertainty, in a random variable or a probability distribution. It is closely related to information. Think of entropy as a measure of the disorder, randomness, or unpredictability in a system.

For a discrete random variable X, with n values x_i and associated probabilities p_i the entropy of X is:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

EXAMPLE 6.7

Problem

Compute the entropy of the system described in the previous example about the probability of a car starting on any given day.

Solution

Recall, the probabilities are: $\frac{255}{256}$ for "car starts" and $\frac{1}{256}$ for "car does not start."

$$H(X) = -\left[\frac{255}{256}\log_2\left(\frac{255}{256}\right) + \frac{1}{256}\log_2\left(\frac{1}{256}\right)\right] = -(-0.00562 - 0.03125) = 0.03687$$

The low value of entropy, 0.03687, indicates that the outcome of the event is highly predictable. This makes sense since the probability of the car starting is $\frac{255}{256} = 0.996$, or 99.6%, which implies that we could predict that the car starts and be correct 99.6% of the time.

The idea of entropy is crucial in creating decision trees, as we shall see in the next section.

Building a Decision Tree

Suppose that you wish to classify data into some number of categories based on values of its features (inputs). We will assume that you have plenty of labeled data to train on. For the purposes of illustration, let's say that your data consists of a set S of points in the plane with labels Red and Blue. The format of each point is (X, Y, Label). Here is your training set.

(18.4, 3.3, Red) (10.1, 6.3, Blue) (9.1, 14.7, Blue) (16.3, 5, Red)

(12.8, 1.9, Blue) (6.9, 2.8, Blue) (3.2, 11, Blue) (17.7, 1.3, Red)

(8.6, 18.3, Blue) (16.9, 5, Red) (1.9, 13, Red) (12.6, 6.7, Blue)

(7.9, 10.6, Red) (14.6, 14.5, Blue) (9.4, 12.8, Blue) (7.6, 10.2, Red)

(19.6, 16.4, Blue) (19.5, 4, Red) (5.2, 12.2, Red)

A colored scatterplot of the dataset S is shown in Figure 6.19.

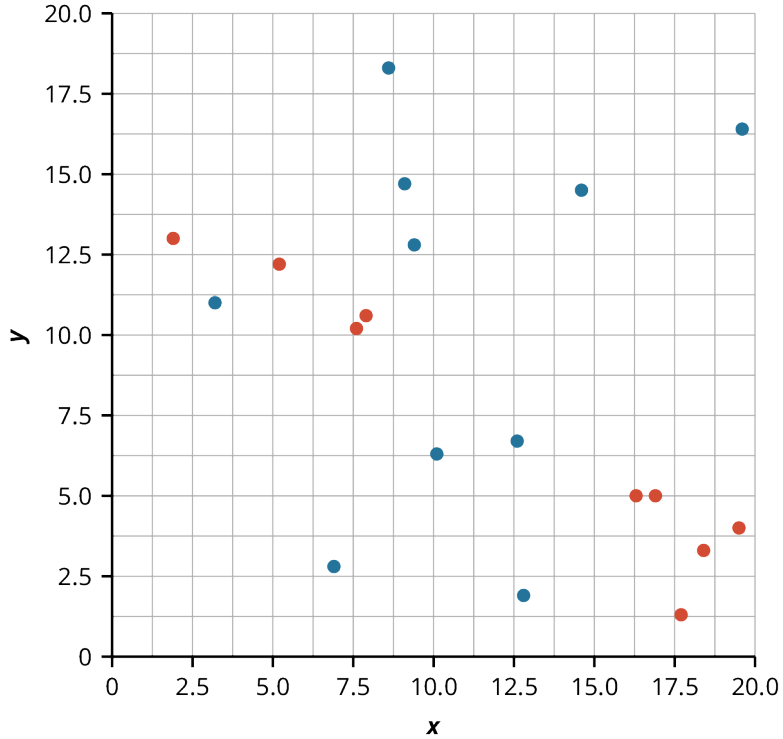


Figure 6.19 Labeled Scatterplot of Dataset S. Note the pattern of red and blue that would be impossible to predict from a typical linear or logistic regression.

Notice in Figure 6.19 that the red points fall into roughly two clusters that are bordered by blue points. In order to capture this feature, you will need a more flexible tool than linear or logistic regression. We will set up a decision tree!

As discussed earlier, a decision tree is a classification algorithm that builds a hierarchical structure where each internal node represents a decision based on a feature of the data and each leaf node represents a final decision, label, or prediction. The potential number of decision trees for any given dataset is vast, and so part of the process is to create the tree in such a way so that each choice provides the greatest information gain.

Entropy is often used as a measure to determine the best feature to split the data at each node. The feature that maximizes information gain (i.e., minimizes entropy) is chosen as the splitting criteria. Each node represents a choice that splits the data into subsets until the entropy is minimized (or below a predefined threshold) in each subset. Any subset that consists of data with only a single label or category is called pure, and no further splitting would be necessary. Such a subset becomes a leaf node. However, not all leaf nodes must be pure. Sometimes a leaf node just has a majority of a given label. This may be done to avoid overfitting the model, as we shall see in Using a Decision Tree to Predict.

Note that information gain is not the only measure that could be used to choose optimal splits. There are other measures, such as the Gini index, which is used in the CART (Classification and Regression Tree) algorithm. Although we will not cover these alternatives, it is important to realize that they exist.

So, how do we construct the nodes of the tree? Start with the root node consisting of the entire dataset. In our

example, there are 10 red points and 9 blue points. Consider this as a set with uncertainty of containing Red (r) or Blue (b) points. The probabilities of each case are $P(r) = \frac{9}{19} = 0.47$ and $P(b) = \frac{10}{19} = 0.53$. The total entropy of the root note is thus:

$$H(\text{root}) = -\left[0.47\log_2(0.47) + 0.53\log_2(0.53)\right] = 0.9974$$

This is very close to the entropy of a coin toss. We need to separate points based on some feature of the dataset. We will use an inequality such as $X \leq a$ or $Y \leq a$, where a is an appropriately chosen constant. The choice is based on maximizing **information gain**, which will be defined later in the chapter.

The type of decision tree that we are building is also called a **regression tree**, as the features are numeric, and we are going to use number comparisons to make decisions. Let's start with an arbitrary choice, say $X \leq 7.75$. (This point is halfway between two adjacent values of X, 7.6 and 7.9. This provides a bit of a buffer between points, which often produces better predictions on noisy data.) The inequality splits the data into two subsets: Left = L = {3 Red, 2 Blue} and Right = R = {6 Red, 8 Blue}. Note that L has 5 points, so its weight with respect to its parent node is $w_L = \frac{5}{19}$. Similarly, the weight of R is $w_L = \frac{14}{19}$. The information gain, or *IG*, of a split is defined by:

$$IG = H(\text{parent}) - w_L H(L) - w_R H(R)$$

Let's first find the entropy measures of L and R.

$$H(L) = -\left[\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right] = 0.9710$$

$$H(R) = -\left[\frac{6}{14}\log_2\left(\frac{6}{14}\right) + \frac{8}{14}\log_2\left(\frac{8}{14}\right)\right] = 0.9852$$

Thus, the information gain of this split is $IG = 0.9974 - \frac{5}{19}(0.9710) - \frac{14}{19}(0.9852) = 0.016$. The information gain is positive, which is a good thing. It is possible to have negative information gain if the split is chosen in such a way that raises the total entropy. But how good is this split? Intuitively, we could have done better by splitting in a way that groups more of the red or more of the blue into one of the two subsets. However, if we want the absolute best split, then we would need to try every possibility and pick the one that has the greatest *IG*. This is not difficult for a computer to do, as there are only finitely many points, hence only finitely many ways to split into subsets. For comparison, we will do one more, using the other feature (Y).

Split on $Y \leq 13.75$ (which is halfway between two adjacent *Y* values, 13 and 14.5). Now *L* = {9 Red, 6 Blue}, and *R* = {0 Red, 4 Blue}. The entropy of *L* is $-\left[\frac{9}{15}\log_2\left(\frac{9}{15}\right) + \frac{6}{15}\log_2\left(\frac{6}{15}\right)\right] = 0.9710$. Note that *R* is a pure node, which has entropy 0. Entropy equal to 0 implies that the outcome is known, that is, there is no uncertainty.

$$IG = 0.9974 - \frac{15}{19}(0.9710) - \frac{4}{19}\left(0\right) = 0.231$$

Since the value of *IG* is greater for this split compared to the previous one, it is better to split on $Y \leq 13.75$. It turns out that this is indeed the split with the greatest information gain, so we will use it to start building our decision tree. Each new node has the potential to become a new parent node, or if it is a pure node, then we do not branch from it.

Figure 6.20 shows a flowchart version of the completed decision tree, which completely classifies all of the data points so that all leaf nodes are pure. You can also see the way that each inequality divides up the plane in Figure 6.21.

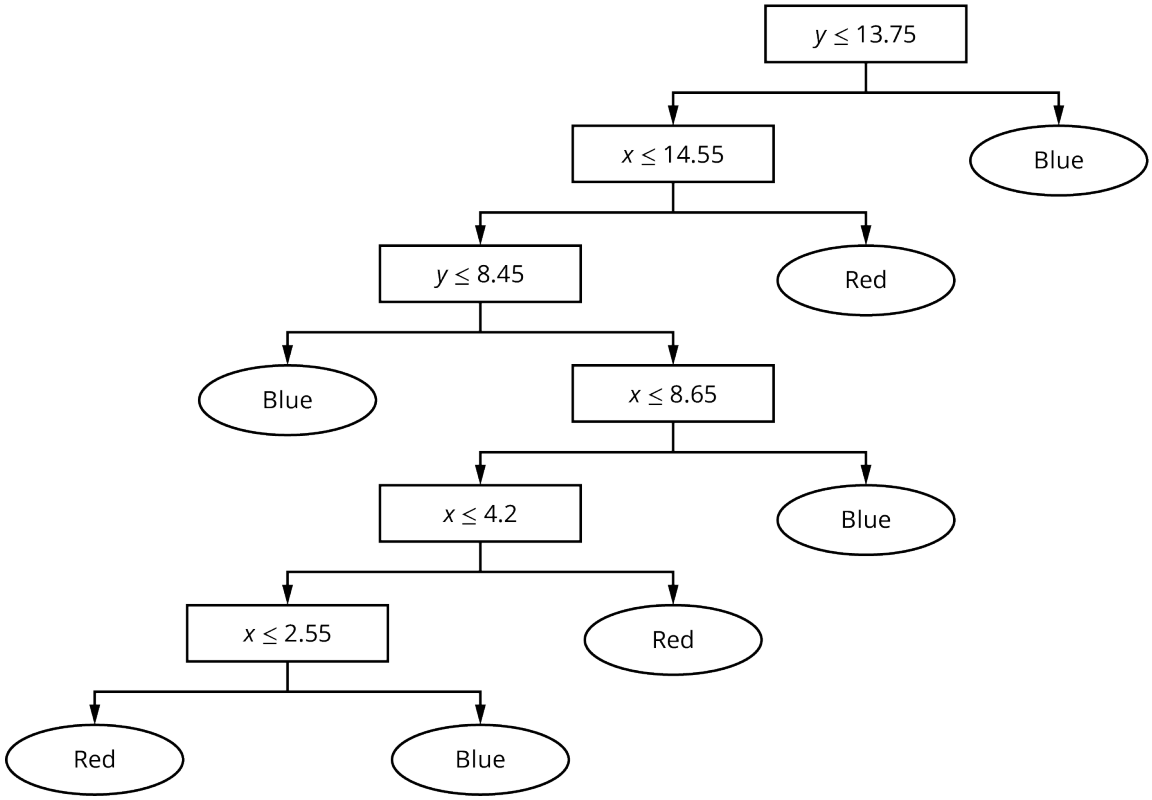


Figure 6.20 Flowchart for a Decision Tree. The root node is at the top. Each internal node asks a question about the data and routes accordingly. Leaf nodes indicate the label for the data.

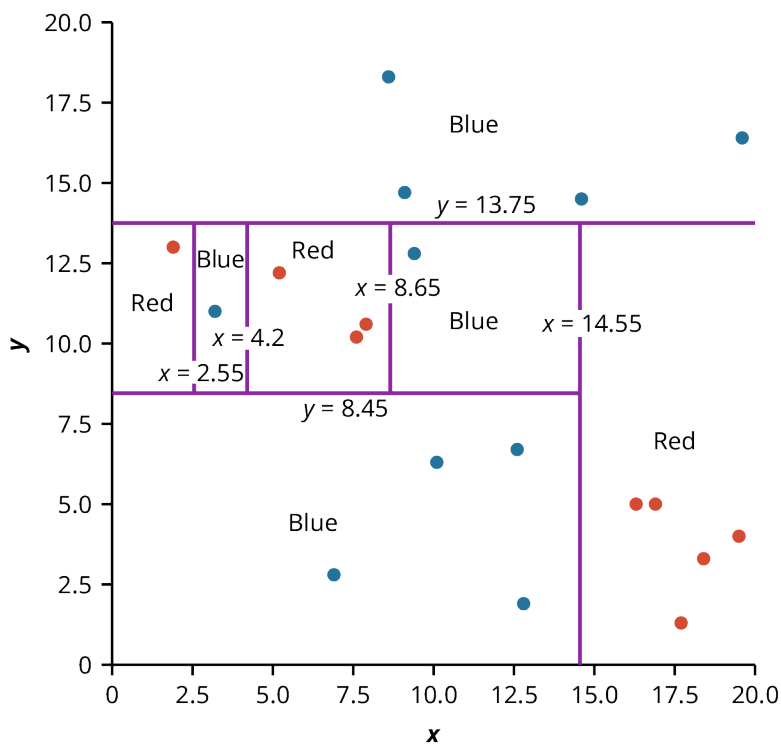


Figure 6.21 Regions of the Plane Corresponding to Pure Nodes. The dataset is completely categorized into red and blue regions separated by lines that are defined by inequalities.

In the preceding example, the maximum number of comparisons, or decisions, that could be made on any input data would be the length of the longest branch of the tree, which is 7 nodes. The number of levels, or length of the longest branch in a decision tree, is known as its **depth**. So this tree has a depth of 7. However,

the last few levels of the tree may in fact be overfitting on the training data. In the next section we will find out how to test and use decision trees and explore ways to reduce their overfitting variance by pruning and other methods.

Using a Decision Tree to Predict

Once you have created a decision tree model, it can be used to classify new data. Simply follow the flowchart to find out which leaf node your data points falls into.

EXAMPLE 6.8

Problem

Using the decision tree model *M*, classify the following points as Red or Blue: (9.9, 2.5), (15.1, 12.6), (3.5, 10.2). (Hint: Follow the flowchart shown in Figure 6.20.)

Solution

$(X, Y) = (9.9, 2.5)$. Starting at the root node, branch left $(Y = 2.5 \leq 13.75)$. Branch left again $(X = 9.9 \leq 14.55)$. Branch left once again $(Y = 2.5 \leq 8.45)$. This puts us into a leaf node marked Blue. So (9.9, 2.5) should be labeled Blue.

$(X, Y) = (15.1, 12.6)$. Starting at the root node, branch left $(Y = 12.6 \leq 13.75)$. Then branch right $(X = 15.1 > 14.55)$. This is a Red leaf node, and so the point (15.1, 12.6) gets the label Red.

$(X, Y) = (3.5, 10.2)$. Starting at the root node, branch left $(Y = 10.2 \leq 13.75)$. Branch left again $(X = 3.5 \leq 14.55)$. Then branch right $(Y = 10.2 > 8.45)$, then left $(X = 3.5 \leq 8.65)$, and then left again $(X = 3.5 \leq 4.2)$. Finally, take the right branch $(X = 3.5 > 2.55)$, which gives the label Blue to the point (3.5, 10.2).

Training and Testing

The decision tree model *M* we constructed in the previous section classifies the original dataset *S* with 100% accuracy. The reason is quite simple; we forced it to! Starting from the root node containing all the points, the data was split into subsets and smaller subsets until every leaf node contained a pure subset (all one label). However, there is no guarantee that *M* accurately represents the underlying structure of the data. Think of our dataset *S* as a random sample of points taken from a larger population of points. There could have been noise, outliers, and other factors present in our sample that cause *M* to give spurious predictions. In order to evaluate the predictive power of a decision tree, we should try it out on a testing set of additional labeled data. We measure the accuracy of the model using:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions}}$$

EXAMPLE 6.9

Problem

Suppose that the three points from Example 6.8 have known labels, as indicated in Table 6.8. How accurate is our decision tree model based on these test points?

X	*Y*	Label	Predicted Label
9.9	2.5	Blue	Blue
15.1	12.6	Red	Red
3.5	10.2	Red	Blue

Table 6.8 Test Points

Solution

According to this table, *M* was correct 2 out of 3 times, so the accuracy is $2/3 = 66.6\%$.

Pruning, Depth-Limiting, and Other Methods for Reducing Variance

Decision trees often suffer from overfitting (high variance), which adversely affects their accuracy. Fortunately, there are a number of methods that can be used to reduce overfitting.

A decision tree that has a lot of internal nodes is prone to overfitting because it may be making decisions based on only a very small amount of data or very small difference in the data's features. "Splitting hairs" is a common term for emphasizing small differences that may not actually matter much. We generally do not want our models to split hairs. **Pruning** is the process of reducing the size of a decision tree by removing branches that split the data too finely.

For example, in creating the model *M*, the subset of *S* defined by the sequence, left, left, right—that is: $Y \leq 13.75$, $X \leq 14.55$, and $Y > 8.45$—consists of the points (7.6, 10.2, Red), (7.9, 10.6, Red), (3.2, 11, Blue), (5.2, 12.2, Red), (9.4, 12.8, Blue), and (1.9, 13, Red). As you can see in Figure 6.20, it takes three additional inequalities to separate the four red and two blue points into regions of the same color. At the end of the process, each of the two blue points and one of the red points became the sole members of their subsets.

Splitting a set into singletons (one-element subsets) is generally not a great idea in machine learning, as this puts too much predictive weight on a single observation. One method of pruning is to remove branches that split the data into sets *L* and *R* in which one of *L* or *R* is a singleton (or below a predefined proportion of the original set). Then the node becomes a leaf consisting of data having multiple labels. The label of this new leaf is determined by majority vote. That is, the label that occurs most often becomes the predicted label of any data that finds itself to this node. This method is commonly called **error-based (reduced-error) pruning**, as it seeks to remove branches that do not significantly increase the classification error rate.

EXAMPLE 6.10

Problem

Use pruning to reduce the complexity of the decision tree model *M* described in this section. Then reclassify the data points given in Example 6.8.

Solution

From the node whose pathway is $Y \leq 13.75$, $X \leq 14.55$, $Y > 8.45$, the model *M* branched on the inequality $X \leq 8.65$. This splits the set into L = {(7.6, 10.2, Red), (7.9, 10.6, Red), (3.2, 11, Blue), (5.2, 12.2, Red), (1.9, 13, Red)} and R = {(9.4, 12.8, Blue)}. Because the split would produce a singleton, we simply make this node a leaf and prune away all branches and nodes underneath it. Now the node is a leaf with four red and two blue data points, and so this leaf is labeled Red. The flowchart and corresponding regions of the plane are

shown in Figure 6.22 and Figure 6.23. Note that the pruned tree has a depth of 4, which is quite a bit smaller than the original tree's depth of 7.

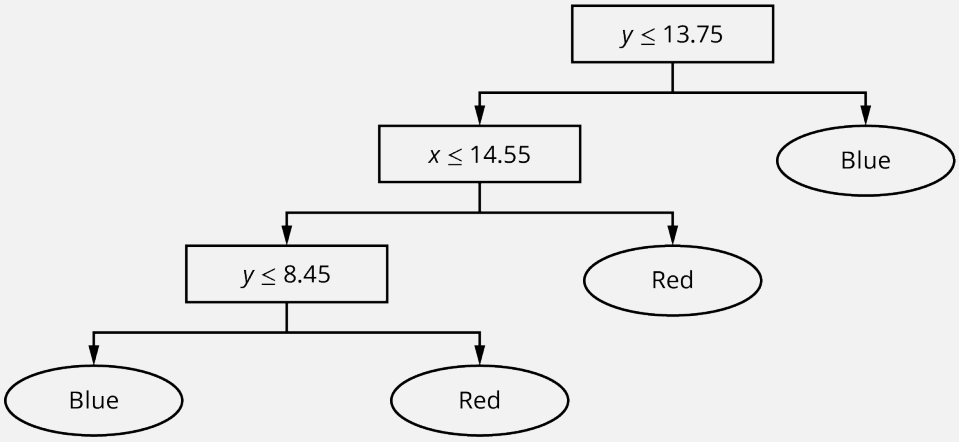


Figure 6.22 Flowchart for Pruned Decision Tree from Example 6.10

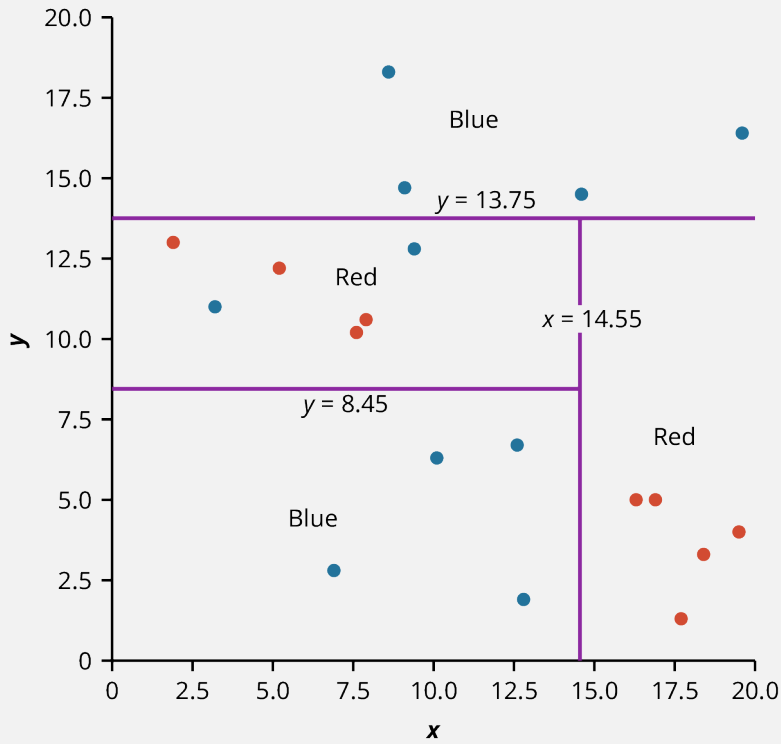


Figure 6.23 Regions of the Plane Corresponding to Pure Nodes for the Pruned Tree. The pruned model divides the plane into fewer regions than the original model did, which likely captures the true underlying pattern better without overfitting.

Now let's reclassify the points (9.9, 2.5), (15.1, 12.6), and (3.5, 10.2) as in Table 6.9.

$(X, Y) = (9.9, 2.5)$. Starting at the root node, branch left ($Y = 2.5 \leq 13.75$). Branch left again ($X = 9.9 \leq 14.55$). Branch left once again ($Y = 2.5 \leq 8.45$), which is a blue leaf.

$(X, Y) = (15.1, 12.6)$. Starting at the root node, branch left ($Y = 12.6 \leq 13.75$). Then branch right ($X = 15.1 > 14.55$), which is a red leaf.

$(X, Y) = (3.5, 10.2)$. Starting at the root node, branch left ($Y = 10.2 \leq 13.75$). Branch left again ($X = 3.5 \leq 14.55$). Then branch right ($Y = 10.2 > 8.45$), which is now a red leaf.

X	*Y*	Label	Predicted Label
9.9	2.5	Blue	Blue
15.1	12.6	Red	Red
3.5	10.2	Red	Red

Table 6.9 Reclassified Test Points

Now we find the model to be 100% accurate on the testing data. (Of course, a model trained on real-world data with many more data points cannot possibly attain 100% accuracy, but this is possible on very small datasets.) Note that the model's accuracy on the original training set *S* is no longer 100%. On *S*, the pruned tree misclassifies two points, (3.2, 11), and (9.4, 12.8) as red when they are in fact blue, and so the accuracy on the training set is now $17/19 = 89.5\%$. That's still pretty good!

Other methods exist for pruning a decision tree. There are two main types: pre-pruning and post-pruning. Pre-pruning refers to setting parameters or criteria for the complexity of the decision tree at the outset. New branches would be added only if the criteria are met. Post-pruning happens after the entire tree is created without restriction. In post-pruning, branches and nodes are removed according to user-defined rules, turning the previously internal nodes into leaf nodes. Error-based pruning may be implemented as either a pre- or post-pruning algorithm.

- **Depth-limiting pruning**, as the name suggests, means that a decision tree cannot grow beyond a predefined depth (or number of levels). This pre-pruning technique helps to reduce the number of small splits, but care must be taken so as not to set the depth so low that the model fails to distinguish important details.
- **Leaf-limiting pruning** puts a maximum cap on the total number of leaves. Like depth-limiting, leaf-limiting is used to restrict the overall complexity of the decision tree model but could lead to underfitting and reduce the predictive power of the model when the data itself possesses a great deal of complexity that needs to be captured.
- **Minimum description length (MDL) pruning** is a post-pruning method that seeks to find the least complex form of a decision tree that meets an acceptable measure of accuracy. In general, there is always a trade-off between simplicity and accuracy in any machine learning algorithm. What MDL does is to err on the side of simplicity. A full description of the MDL algorithm falls outside the scope of this text.

Decision Trees in Python

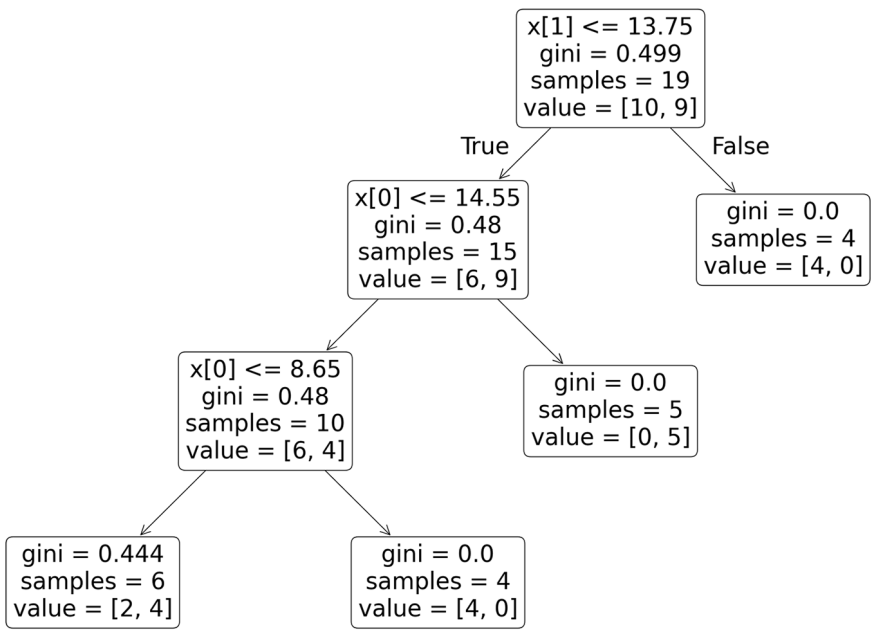
The Python package `sklearn` includes powerful algorithms for creating and working with decision trees. Note that the decision tree classifier is limited to a maximum depth of 3 and the Gini index is used rather than information gain. The data is contained in the dataset [RedBlue.csv (https://openstax.org/r/redblue)](https://openstax.org/r/redblue).

PYTHON CODE

```
# Import libraries
import pandas as pd  ## for dataset management
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

```
import matplotlib.pyplot as plt

# Read data file
data = pd.read_csv('RedBlue.csv')

inputs = data[['X', 'Y']]
output = data['Label']

# Train Decision Tree Classifier
clf = DecisionTreeClassifier(max_depth=3)
clf = clf.fit(inputs,output)

# Plot tree with customization
plt.figure(figsize=(12, 8))  # Set the size of the plot
tree.plot_tree(clf,
               rounded=True,       # Round the edges of the box
               fontsize=10)        # Adjust font size for clarity
plt.show()  # Display the plot
```

The resulting output will look like this:



6.5 Other Machine Learning Techniques

Learning Outcomes

By the end of this section, you should be able to:

- 6.5.1 Discuss the concept of a random forest as a bootstrapping method for decision trees.
- 6.5.2 Create a random forest model and use it to classify data.
- 6.5.3 Define conditional probability and explain prior probabilities for training datasets.
- 6.5.4 Produce a (multinomial) naïve Bayes classifier and use it to classify data.
- 6.5.5 Discuss the concept of a Gaussian naïve Bayes classifier.
- 6.5.6 Describe some methods for working with big data efficiently and effectively.

So far, we have encountered a number of different machine learning algorithms suited to particular tasks. Naturally, the realm of machine learning extends well beyond this small survey of methods. This section describes several more machine learning techniques, though in less detail. Methods such as random forests, naïve Bayes classifiers, and a refinement of the latter called Gaussian naïve Bayes, offer distinctive approaches to solving complex problems. Additionally, this section provides insights into strategies for handling vast datasets, presenting a comprehensive survey of methods tailored for the unique challenges posed by Big Data.

Random Forests

In Decision Trees, we constructed decision trees and discussed some methods, such as pruning, that would improve the reliability of predictions of the decision tree. For each classification problem, one tree is created that serves as the model for all subsequent purposes. But no matter how much care is taken in creating and pruning a decision tree, at the end of the day, it is a single model that may have some bias built into its very structure due to variations in the training set. If we used a different training set, then our decision tree may have come out rather differently.

In the real world, obtaining a sufficient amount of data to train a machine learning model may be difficult, time-consuming, and expensive. It may not be practical to go out and find additional training sets just to create and evaluate a variety of decision trees. A **random forest**, a classification algorithm that uses multiple decision trees, serves as a way to get around this problem. Similar to what we did for linear regressions in Machine Learning in Regression Analysis, the technique of *bootstrap aggregating* is employed. In this context, **bootstrap aggregating (bagging)** involves resampling from the same testing dataset multiple times, creating a new decision tree each time. The individual decision trees that make up a random forest model are called **weak learners**. To increase the diversity of the weak learners, there is random feature selection, meaning that each tree is built using a randomly selected subset of the feature variables. For example, if we are trying to predict the presence of heart disease in a patient using age, weight, and cholesterol levels, then a random forest model will consist of various decision trees, some of which may use only age and weight, while others use age and cholesterol level, and still others may use weight alone. When classifying new data, the final prediction of the random forest is determined by "majority vote" among the individual decision trees. What's more, the fact that individual decision trees use only a subset of the features means that the importance of each feature can be inferred by the accuracy of decision trees that utilize that feature.

Because of the complexity of this method, random forests will be illustrated by way of the following Python code block.

Suppose you are building a model that predicts daily temperatures based on such factors as the temperature yesterday and the day before as well forecasts from multiple sources, using the dataset temps.csv (https://openstax.org/r/temps). Since there are a lot of input features, it will not be possible to visualize the data. Moreover, some features are likely not to contribute much to the predictions. How do we sort everything out? A random forest model may be just the right tool. The Python library sklearn (https://openstax.org/r/scikit1) has a module called `sklearn.ensemble` that is used to create the random forest model.

PYTHON CODE

```
# Import libraries
import pandas as pd  ## for dataset management
from sklearn.model_selection import train_test_split
import sklearn.ensemble as ens
```

```
# Read input file
features = pd.read_csv('temps.csv').dropna()

# Use 'actual' as the response variable
labels = features['actual']

# Convert text data into numerical values
# This is called "one-hot" encoding
features = pd.get_dummies(features)

# The other columns are used as features
features = features.drop('actual', axis=1)
feature_list = list(features.columns)

# Split data into training and testing sets
train_features, test_features, train_labels, test_labels =
train_test_split(features, labels, test_size=0.25)

# Create random forest model
rf = ens.RandomForestRegressor(n_estimators=1000)
rf.fit(train_features, train_labels)
```

The resulting output will look like this:

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=1000)
```

The preceding Python code reads in the dataset temps.csv (https://openstax.org/r/temps), identifying the column "actual" (actual temperatures each day) as the variable to be predicted. All the other columns are assumed to be features. Since the "week" column contains categorical (text) data, it needs to be converted to numerical data before any model can be set up. The method of one-hot encoding does the trick here. In **one-hot encoding**, each category is mapped onto a vector containing a single 1 corresponding to that category while all other categories are set to 0. In our example, each day of the week maps to a seven-dimensional vector as follows:

- Monday: [1, 0, 0, 0, 0, 0, 0]
- Tuesday: [0, 1, 0, 0, 0, 0, 0]
- Wednesday: [0, 0, 1, 0, 0, 0, 0]
- Thursday: [0, 0, 0, 1, 0, 0, 0]
- Friday: [0, 0, 0, 0, 1, 0, 0]
- Saturday: [0, 0, 0, 0, 0, 1, 0]
- Sunday: [0, 0, 0, 0, 0, 0, 1]

The dataset is split into training (75%) and testing (25%) sets, and the random forest model is trained on the training set, as seen in this Python code block:

PYTHON CODE

```
import statistics as stats # to compute the mean

# Get predictions and compute average error
predictions = rf.predict(test_features)
errors = abs(predictions - test_labels)
round(np.mean(errors),2)
```

The resulting output will look like this:

```
3.91
```

With a mean absolute error of only 3.91, predicted temperatures are only off by about 4°F on average, so the random forest seems to do a good job with the test set, given that it is notoriously difficult to predict weather data since it typically shows high variance. Now let's find out which features were most important in making predictions, which is stored in `rf.feature_importances_`. Note the number of Python commands used solely for sorting and formatting the results, which can be safely ignored in this snippet of code.

PYTHON CODE

```
# Find the importance of each feature
importances = list(rf.feature_importances_)
feature_importances = [(feature, round(importance,2)) for feature, importance in
zip(feature_list, importances)]
feature_importances = sorted(feature_importances, key=lambda x: x[1], reverse=True)
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances];
```

The resulting output will look like this:

```
Variable: temp_1               Importance: 0.62
Variable: average              Importance: 0.19
Variable: forecast_acc         Importance: 0.06
Variable: forecast_noaa        Importance: 0.04
Variable: day                  Importance: 0.02
Variable: temp_2               Importance: 0.02
Variable: forecast_under       Importance: 0.02
Variable: friend               Importance: 0.02
Variable: month                Importance: 0.01
Variable: year                 Importance: 0.0
Variable: week_Fri             Importance: 0.0
Variable: week_Mon             Importance: 0.0
Variable: week_Sat             Importance: 0.0
```

```
Variable: week_Sun             Importance: 0.0
Variable: week_Thurs           Importance: 0.0
Variable: week_Tues            Importance: 0.0
Variable: week_Wed             Importance: 0.0
```

As we can see, the single best predictor of daily temperatures is the temperature on the previous day ("temp_1"), followed by the average temperature on this day in previous years ("average"). The forecasts from NOAA and ACC showed only minor importance, while all other feature variables were relatively useless in predicting current temperatures. It should come as no surprise that the day of the week (Monday, Tuesday, etc.) played no significant part in predicting temperatures.

EXPLORING FURTHER

Using Random Forests for Facial Recognition

As noted in this chapter's introduction, facial recognition is an important and widely used application of machine learning—and an example of a classification task. **Facial recognition** involves categorizing or labeling images of faces based on the identities of individuals depicted in those images, which is a *multiclass classification task*. (In its simplest form, when the task is to determine whether a given image contains the face of a particular individual or *not*, this is considered a *binary classification task*.) When the data consist of many images of faces, each containing hundreds or thousands of features, using random forests would be appropriate. Check out this Colab Notebook, written by Michael Beyeler, a neuroscientist at the University of California, Santa Barbara, which steps through the process of setting up and using a random forest model for facial recognition in Python.

Multinomial Naïve Bayes Classifiers

One method for classifying data is to look for commonly occurring features that would distinguish one data point from another. For example, imagine you have a vast collection of news articles and you want to automatically categorize them into topics such as sports, politics, business, entertainment, and technology. You may have noticed certain patterns, such as sport articles tend to mention team names and give numerical scores, while articles about politics mention the words "Democrat" and "Republican" fairly often. Business articles might use words such as "outlook" and "downturn" much more often than entertainment or technology articles would. Multinomial naïve Bayes can be applied to classify these articles based on the frequency and distribution of words typically found in each kind of article.

The multinomial **naïve Bayes classification** algorithm is based on Bayes' Theorem, making use of prior probabilities and Bayes' Theorem to predict the class or label of new data. (See Normal Continuous Probability Distributions for more details.) The naïve Bayes classifier algorithm is a powerful tool for working out probabilities of events based on prior knowledge. Recall, the notation $P(A|B)$ stands for the conditional probability that event A occurs given that event B is known to occur (or has already occurred or is presumed to occur). In the simplest version, Bayes' Theorem takes the form

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Here, we regard A as an event that we would like to predict and B as information that has been given to us (perhaps as a feature of a dataset). The conditional probability $P(B|A)$ represents the known likelihood of seeing event B in the case that A occurs, which is typically found using the training data. The value of $P(A)$ would be computed or estimated from training data as well and is known as the **prior probability**.

EXAMPLE 6.11

Problem

Three companies supply batteries for a particular model of electric vehicle. Acme Electronics supplies 25% of the batteries, and 2.3% of them are defective. Batteries 'R' Us supplies 36% of the batteries, and 1.7% of them are defective. Current Trends supplies the remaining 39% of the batteries, and 2.1% of them are defective. A defective battery is delivered without information about its origin. What is the probability that it came from Acme Electronics?

Solution

First, we find the probability that a delivered battery is defective. Let A, B, and C stand for the proportions of batteries from each of the companies, Acme Electronics, Batteries 'R' Us, and Current Trends, respectively. Then $P(A) = 0.25$, $P(B) = 0.36$, $P(C) = 0.39$. We also have the probabilities of receiving a defective battery (event d) from each company, $P(d|A) = 0.023$, $P(d|B) = 0.017$, $P(d|C) = 0.021$. Now, the probability of receiving a defective battery from any of the companies is the sum

$$\begin{aligned} P(d) &= P(A)P(d|A) + P(B)P(d|B) + P(C)P(d|C) \\ &= (0.25)(0.023) + (0.36)(0.017) + (0.39)(0.021) = 0.02 \end{aligned}$$

Using Bayes' Theorem, the probability that the defective battery came from Acme is:

$$P(A|d) = \frac{P(A)P(d|A)}{P(d)} = \frac{(0.25)(0.023)}{0.02} = 0.29 = 29\%$$

In practice, there are often a multitude of features, $B_1, B_2, B_3, \ldots B_n$, where each feature could be the event that a particular word occurs in a message, and we would like to classify the message into some type or another based on those events. In other words, we would like to know something about $P(A|B_1, B_2, B_3, \ldots, B_n)$, the probability of A occurring if we know that events $B_1, B_2, B_3, \ldots B_n$ occurred. In fact, we are just interested in distinguishing event A from A' rather than precise computations of probabilities, and so we will do simple comparisons to find out this information. The process is best illustrated by example.

EXAMPLE 6.12

Problem

A survey of 1,000 news articles was conducted. It is known that 600 were authentic articles and 400 were fake. In the real articles, 432 contained the word *today*, 87 contained the word *disaster*, and 303 contained the word *police*. In the fake articles, 124 contained the word *today*, 320 contained the word *disaster*, and 230 contained the word *police*. Predict whether a new article is real or fake if it (a) contains the word "disaster" or (b) contains the words "today" and "police."

Solution

First, find the prior probabilities that a news article is real or fake based on the proportion of such articles in the training set. $P(\text{Real}) = \frac{600}{1{,}000} = 0.6$ and $P(\text{Fake}) = \frac{400}{1000} = 0.4$. Next, using the word counts found in the training data, find the conditional probabilities:

$P(\text{today}|\text{Real}) = \frac{432}{600} = 0.72$, $P(\text{disaster}|\text{Real}) = \frac{87}{600} = 0.145$, $P(\text{police}|\text{Real}) = \frac{303}{600} = 0.505$

$P(\text{today}|\text{Fake}) = \frac{124}{400} = 0.31$, $P(\text{disaster}|\text{Fake}) = \frac{320}{400} = 0.8$, $P(\text{disaster}|\text{Fake}) = \frac{230}{400} = 0.575$

Now we will only use the numerator of Bayes' formula, which is proportional to the exact probability. This

will produce scores that we can compare. For part (a):

$P(\text{Real}|\text{disaster})$ score: $P(\text{Real})P(\text{disaster}|\text{Real}) = (0.6)(0.145) = 0.087$

$P(\text{Fake}|\text{disaster})$ score: $P(\text{Fake})P(\text{disaster}|\text{Fake}) = (0.4)(0.8) = 0.32$

Since the score for *Fake* is greater than the score for *Real*, we would classify the article as fake.

For part (b), the process is analogous. Note that probabilities are multiplied together when there are multiple features present.

$$P(\text{Real}|\text{today, police}) \text{ score: } P(\text{Real})P(\text{today}|\text{Real})P(\text{police}|\text{Real}) = (0.6)(0.72)(.505) = 0.22$$

$$P(\text{Fake}|\text{today, police}) \text{ score: } P(\text{Fake})P(\text{today}|\text{Fake})P(\text{police}|\text{Fake}) = (0.4)(0.31)(.575) = 0.071$$

Here the score for *Real* is greater than that of *Fake*, so we conclude the article is real.

Note: Naïve Bayes classification assumes that all features of the data are independent. While this condition may not necessarily be true in real-world datasets, naïve Bayes may still perform fairly well in real-world situations. Moreover, naïve Bayes is a very simple and efficient algorithm, and so it is often the first method used in a classification problem before more sophisticated models are employed.

Gaussian Naïve Bayes for Continuous Probabilities

Gaussian naïve Bayes is a variant of the naïve Bayes classification algorithm that is specifically designed for data with continuous features. Unlike the standard (multinomial) naïve Bayes, which is commonly used to classify text data with discrete features, Gaussian naïve Bayes is suitable for datasets with features that follow a Gaussian (normal) distribution (You may recall the discussion of the normal distribution in Discrete and Continuous Probability Distributions). It's particularly useful for problems involving real-valued, continuous data in which the feature variables are relatively independent.

In this example, we will work with a large dataset, cirrhosis.csv (https://openstax.org/r/cirrhosis), that has many features, both categorical and numerical. This dataset (along with many more!) is available for free download from Kaggle.com. We will only use four of the numerical columns as features. The response variables is "Status," which may take the values D for death, C for censored (meaning that the patient did not die during the observation period), and *CL* for censored due to liver transplantation.

PYTHON CODE

```
# Import libraries
import pandas as pd  ## for dataset management
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load the dataset
data = pd.read_csv('cirrhosis.csv').dropna()
```

```
# Choose feature columns and target labels
X = data[['Bilirubin','Cholesterol','Albumin','Copper']]
y = data['Status']

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Initialize and train the Gaussian naive Bayes classifier
gnb_classifier = GaussianNB()
gnb_classifier.fit(X_train, y_train)

# Predict labels for the test set
y_pred = gnb_classifier.predict(X_test)

# Evaluate the classifier's performance
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred, labels=['D','C','CL'])
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Confusion matrix with label order D, C, CL:\n", confusion)
```

The resulting output will look like this:

```
Accuracy: 0.71
Confusion matrix with label order D, C, CL:
[[10 14  2]
[ 0 30  0]
[ 0  0  0]]
```

The accuracy score of 0.71 means that correct labels were assigned 71% of the time. We can see that 10 deaths (D) and 30 non-deaths (C or CL) were corrected predicted. However, 16 predictions of death were ultimately non-deaths. This is a much more desirable error than the reverse (prediction of non-death when in fact the patient will die), so the model seems to be appropriate for the task of predicting cirrhosis fatalities.

Working with Big Data

In the rapidly evolving landscape of machine learning, the concept of big data looms large. As we saw in Handling Large Datasets, large datasets, often termed **big data**, refers to extremely large and complex datasets that are beyond the capacity of traditional data processing and analysis tools to efficiently handle, manage, and analyze. Big data is characterized by the "three Vs":

- **Volume:** Big data involves massive amounts of data, often ranging from terabytes ($2^{40} \approx 10^{12}$ bytes) to as large as petabytes ($2^{50} \approx 10^{15}$ bytes) and beyond. Where does such vast amounts of data originate? One example is the Common Crawl dataset (https://openstax.org/r/commoncrawl), which contains petabytes of data consisting of the raw web page data, metadata, and other information available from the internet. It is likely that there will be datasets containing exabytes ($2^{60} \approx 10^{18}$ bytes) of information in the near future.
- **Velocity:** In some important real-world applications, data is generated and collected at an incredibly high speed. This continuous flow of data may be streamed in real time or stored in huge databases for later analysis. One example is in the area of very high-definition video, which can be captured, stored, and

analyzed at a rate of gigabits per hour.

- **Variety:** Big data also encompasses diverse types of data, including structured data (e.g., databases), unstructured data (e.g., text, images, videos), and anything in between. This variety adds considerable complexity to data management and analysis. Consider all the various sources of data and how different each may be in transmission and storage. From social media interactions and e-commerce transactions to sensor data from Internet of Things (IoT) devices, data streams in at a pace, scale, and variety never before seen. Moreover, few will be able to predict the brand-new ways that data will be used and gathered in the future!

Data Cleaning and Mining

The most time-consuming aspect of data science has traditionally been **data cleaning**, a process we discussed in Data Cleaning and Preprocessing. Datasets are often messy, missing data, and likely to contain errors or typos. These issues are vastly compounded when considering big data. While it may take a person an hour or so to look through a dataset with a few hundred entries to spot and fix errors or deal with incomplete data, such work becomes impossible to do by hand when there are millions or billions of entries. The task of data cleaning at scale must be handled by technology; however, traditional data cleaning tools may be ill-equipped to handle big data because of the sheer volume. Fortunately, there are tools that can process large amounts of data in parallel.

Data Mining

Data mining is the process of discovering patterns, trends, and insights from large datasets. It involves using various techniques from machine learning, statistics, and database systems to uncover valuable knowledge hidden within data. Data mining aims to transform raw data into actionable information that can be used for decision-making, prediction, and problem-solving and is often the next step after data cleaning. Common data mining tasks include clustering, classification, regression, and anomaly detection. For unlabeled data, unsupervised machine learning algorithms may be used to discover clusters in smaller samples of the data, which can then be assigned labels that would be used for more far-ranging classification or regression analysis.

EXPLORING FURTHER

Tools for Data Mining

Tools for large-scale data mining include Apache Spark (https://openstax.org/r/spark), which uses a distributed file system (HDFS) to store and process data across clusters and also relies on in-memory processing, and Apache Flink (https://openstax.org/r/flink), which enables real-time data analytics. See this Macrometa article (https://openstax.org/r/macrometa) for a comparison of the two programs.

Methods for Working with Big Data

Machine learning in the era of big data calls for scalability and adaptability. Traditional analysis methods, like linear regression, logistic regression, decision trees, and Bayesian classifiers, remain valuable tools. However, they often benefit from enhancements and new strategies tailored for big data environments:

- **Parallel processing:** Techniques that distribute computation across multiple nodes or cores, like parallelized decision trees and ensemble methods, can significantly speed up model training.
- **Feature selection:** Not all features of a huge dataset may be used at once. Advanced feature selection algorithms are necessary to isolate only the most significant features useful for predictions.
- **Progressive learning:** Some big data scenarios require models that can learn incrementally from data streams. Machine learning algorithms will need to adapt to changing data in real time.
- **Deep learning:** Deep neural networks, with their capacity to process vast volumes of data and learn

intricate patterns, have become increasingly important in big data applications. We will delve into this topic in Deep Learning and AI Basics.

- **Dimensionality reduction:** Techniques like principal component analysis (PCA) help reduce the dimensionality of data, making it more manageable while retaining essential information. (Note: Dimension reduction falls outside the scope of this text.)

When the volume of data is large, the simplest and most efficient algorithms would typically be chosen, in which speed is more important than accuracy. For example, spam filtering for email often utilizes multinomial Bayes' classification. In the age of instant digital communication, your email inbox can quickly become flooded with hundreds of meaningless or even malicious messages (i.e., spam) in a day, and so it is very important to have a method of classifying and removing the spam. Algorithms easily scale to work on big data, while other methods such as decision trees and random forests require significant overhead and would not scale up as easily.

Datasets

Note: The primary datasets referenced in the chapter code may also be downloaded here (https://openstax.org/r/drive6).

Key Terms

accuracy for machine learning in general, a measure of the correctness of a machine learning model with respect to predictions.

bias error introduced by overly-simplistic or overly-rigid models that do not capture important features of the data

big data Extremely large and complex datasets that require special methods to handle

binary (binomial) classification classification of data into one of two categories

bootstrap aggregating (bagging) resampling from the same testing data multiple times to create a number of models (for example, decision trees) that all contribute to the overall model

bootstrapping resampling portions of the data multiple times in order to generate a distribution that determines a confidence interval for parameters in a model

centroid geometric center of a subset of points

cluster a subset of a dataset consisting of points that have similar characteristics or are near one another

confusion matrix table of values indicating how data was classified correctly or incorrectly by a given model. Entry in row *i* and column *j* gives the number of times (or percentage) that data with label *i* was classified by the model as label *j*

data cleaning process of identifying and correcting errors, typos, inconsistencies, missing data, and other anomalies in a dataset

data mining process of discovering patterns, trends, and insights from large datasets

DBScan algorithm common density-based clustering algorithm

decision tree classifier algorithm that builds a hierarchical structure where each internal node represents a decision based on a feature of the data, and each leaf node represents a final decision, label, or prediction

density-based clustering algorithm clustering algorithm that builds clusters of relatively dense subsets

depth number of levels of a decision tree, or equivalently, the length of the longest branch of the tree

depth-limiting pruning pre-pruning method that restricts the total depth (number of levels) of a decision tree

entropy measure of the average amount of information or uncertainty

error-based (reduced-error) pruning pruning method that removes branches that do not significantly improve the overall accuracy of the decision tree

F1 Score combination of precision (p) and recall (r). $F1 = \frac{2(p)(r)}{p+r} = \frac{2TP}{2TP+FP+FN}$

facial recognition application of machine learning that involves categorizing or labeling images of faces based on the identities of individuals depicted in those images

Gaussian naïve Bayes classification algorithm that is useful when variables are assumed to come from normal distributions

heatmap shading or coloring of a table to show contrasts in low versus high values

information gain comparison of entropy change due to adding child nodes to a parent node in a decision tree

information theory framework for measuring and managing the uniqueness of data, or the degree of surprise or uncertainty associated with an event or message

k-means clustering algorithm clustering algorithm that iteratively locates centroids of clusters

labeled data data that has classification labels

leaf-limiting pruning pre-pruning method that restricts the total number of leaf nodes of a decision tree

likelihood measure of accuracy of a classifier algorithm, useful for setting up logistic regression models

logistic regression modeling method that fits data to a logistic (sigmoid) function and typically performs binary classification

logit function function of the form $\ln\left(\frac{p}{1-p}\right)$ used to compute log-odds and transform data when performing logistic regression

machine learning (ML) model any algorithm that trains on data to determine or adjust parameters of a

model for use in classification, clustering, decision making, prediction, or pattern recognition

mean absolute error (MAE) measure of error: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

mean absolute percentage error (MAPE) measure of relative error: $\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$

mean squared error (MSE) measure of error: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$

minimum description length (MDL) pruning post-pruning method that seeks to find the least complex form of a decision tree that meets an acceptable measure of accuracy

multiclass (multinomial) classification classification of data into more than two categories

multiple regression regression techniques that use more than one input variable

naïve Bayes classification also known as multinomial naïve Bayes classification, a classification algorithm that makes use of prior probabilities and Bayes' Theorem to predict the class or label of new data

odds probability of an event E occurring divided by the probability of E not occurring

one-hot encoding replacing categorical/text values in a dataset with vectors that contain a single 1 and all other entries being 0; each category vector has the 1 in a distinct place

overfitting modeling using a method that yields high variance; the model captures too much of the noise and so may perform well on training data but very poorly on testing data

precision ratio of true positive predictions to the total number of positive predictions: $p = \frac{TP}{TP+FP}$

prior probability estimate of a probability, which may be updated or corrected based on Bayes' Theorem

pruning reducing the size of a decision tree by removing branches that split the data too finely

random forest classifier algorithm that uses multiple decision trees and bootstrap aggregating

recall ratio of true positive predictions to the total number of actual positives: $r = \frac{TP}{TP+FN}$

regression tree type of decision tree in which the decisions are based on numerical comparisons of continuous data

root mean squared error (RMSE) measure of error: $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$

sigmoid function function useful in logistic regression: $\sigma(x) = \frac{1}{1+e^{-x}}$

silhouette score a measure of how well-separated the clusters are when using a clustering algorithm

supervised learning machine learning methods that train on labeled data

testing set (or data) portion of the dataset that is set aside and used after the training of the algorithm to test for accuracy of the model

training set (or data) portion of the dataset that is used to train a machine learning algorithm

underfitting modeling using a method that yields high bias; the model does not capture important features of the data

unlabeled data data that has not been classified or for which classification data is not known yet

unsupervised learning machine learning methods that do not require data to be labeled in order to learn; often, unsupervised learning is a first step in discovering meaningful clusters that will be used to define labels

variance error due to an overly sensitive model that reacts to small changes in the data

weak learners individual models that are trained on parts of the dataset and then combined in a bootstrap aggregating method such as random forest

Group Project

Project A: Comparing k-Means and DBScan

In this chapter, there were two datasets used to illustrate k-means clustering and DBScan, FungusLocations.csv (https://openstax.org/r/FungusLocations) and DBScanExample.csv (https://openstax.org/r/DBScanExample). Run each algorithm on the other dataset. In other words, use DBScan to classify the points of FungusLocations.csv (https://openstax.org/r/FungusLocations) and k-means to cluster the points of DBScanExample.csv (https://openstax.org/r/DBScanExample). Discuss the results, comparing the performance

of both algorithms on each dataset. If there were any issues with the classifications, discuss possible reasons why those issues came up.

Project B: Building a Decision Tree to Predict College Completion

Build a decision tree classifier based on the dataset CollegeCompletionData.csv (https://openstax.org/r/CollegeCompletionData) to classify students as likely to complete college (or not) based on their GPA and in-state status. Experiment with different ratios of training to testing data and different pruning techniques to find a model with the best accuracy. Generate some data points with random GPAs and in-state statuses and use your model to predict college completion on your new data.

Project C: Predicting Outcomes in Liver Disease Patients

Analyze the dataset cirrhosis.csv (https://openstax.org/r/cirrhosis) to predict labels D, C, and CL based on various combinations of the feature columns, using (a) random forest and (b) multiple logistic regression. Compare the accuracy of your models with the Gaussian naïve Bayes classifier that was produced in Gaussian Naive Bayes for Continuous Probabilities.

Chapter Review

1. You are working with a dataset containing information about customer purchases at an online retail store. Each data point represents a customer and includes features such as age, gender, location, browsing history, and purchase history. Your task is to segment the customers into distinct groups based on their purchasing behavior in order to personalize marketing strategies. Which of the following machine learning techniques is best suited for this scenario?
   a. linear or multiple linear regression
   b. logistic or multiple logistic regression
   c. k-means clustering
   d. naïve Bayes classification

Critical Thinking

1. Discuss how different ratios of training versus testing data can affect the model in terms of underfitting and overfitting. How does the testing set provide a means to identify issues with underfitting and overfitting?

2. A university admissions office would like to use a multiple linear regression model with students' high school GPA and scores on both the SAT and ACT (standardized tests) as input variables to predict whether the student would eventually graduate university if admitted. Assuming the following statements are all accurate, which statement would be a reason not to use multiple linear regression?
   a. Students' SAT and ACT scores are often highly correlated with one another.
   b. GPA scores are measured on a different scale than either SAT or ACT scores.
   c. Scores of 0 are impossible to obtain on the SAT or ACT.
   d. Students can have high GPA but sometimes not do well on standardized tests like the SAT or ACT.

3. Using the data about words found in news articles from **Example 6.12**, classify an article that contains all three words, *today*, *disaster*, and *police*, as real or fake.

Quantitative Problems

1.
   a. Consider the data in the following table representing world record fastest times for the 100 m sprint and the year in which each occurred, from 1912 through 2002:

Year	Time
1912	10.6
1921	10.4
1930	10.3
1936	10.2
1956	10.1
1960	10
1968	9.95
1983	9.93
1988	9.92
1991	9.9
1991	9.86
1994	9.85
1996	9.84
1999	9.79

Table 6.10
100-Meter Spring Records

Using software such as Excel, Python, or similar tools, the regression line can be found. For this data, the linear model would be $\hat{y} = -0.00769x + 10.55$, where x is years since 1900. Compute the MAE, MAPE, MSE, and RMSE for this model.

b. Use the model to predict the world record fastest time for the 100 m spring in 2023. In August of 2023, Noah Lyles ran the 100 m sprint in 9.83 seconds. How does your prediction compare to this value?

2. If today is a cloudy day, then there is 68% chance of rain tomorrow. If today is not cloudy, then there is only 15% chance of rain tomorrow. Build a discrete logistic regression model based on this data.

3. Compute the information contained in the events.

a. Obtaining a head and a tail on a flip of two fair coins

b. Obtaining all heads on the flip of three fair coins

c. Rolling 10 fair 6-sided dice and obtaining all 1s

4. Compute the entropy of the following discrete random variables.

a. The distribution *X* whose values are given in the following table.

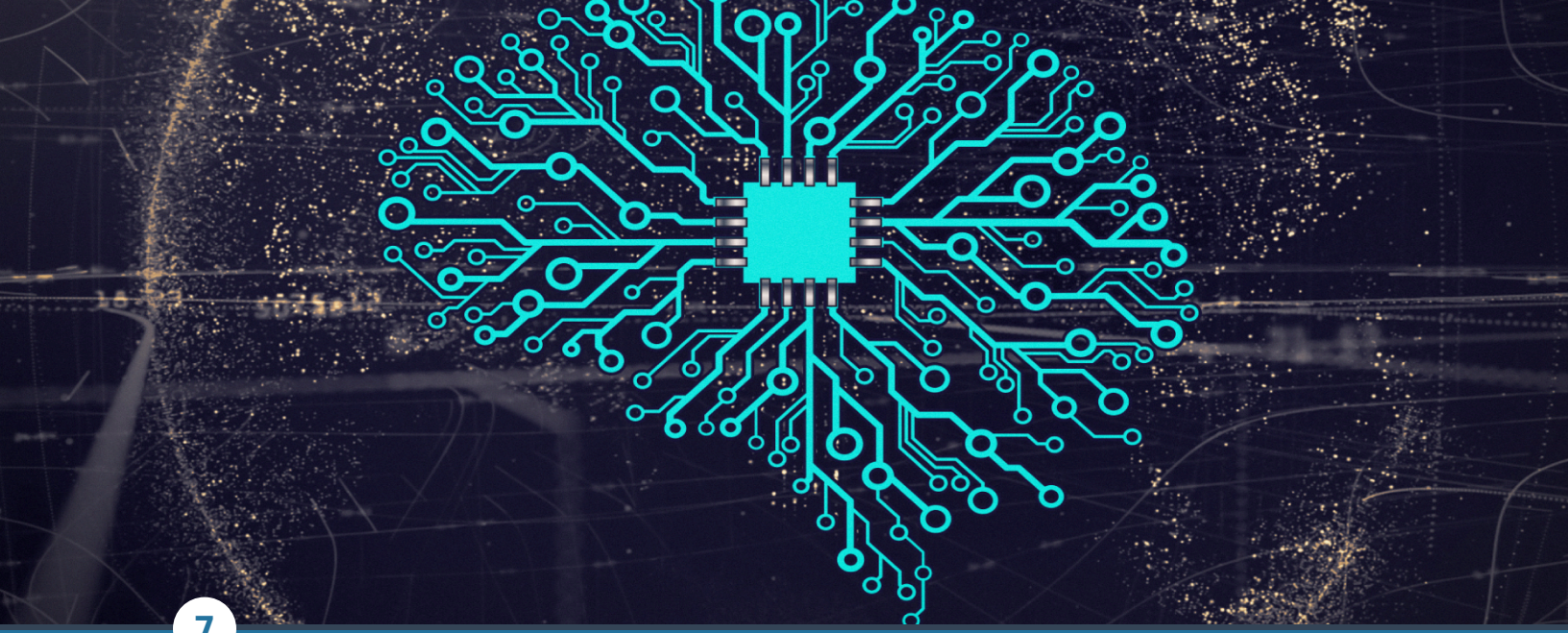
X	1	2	3	4
$p(x)$	0.7	0.15	0.05	0.05

Table 6.11 Distribution *X* Values

b. The binomial distribution with $p = 1/3$ and $n = 4$. Recall, the binomial distribution is defined by:
$P(x) = \binom{n}{x} p^x (1-p)^{n-x}$

References

1. Emspak, J. (2016, December 29). How a machine learns prejudice. *Scientific American*. https://www.scientificamerican.com/article/how-a-machine-learns-prejudice/



7 Deep Learning and AI Basics

Figure 7.1 Neural networks are central to modern AI, especially in the fields of deep learning and machine learning. Inspired by the human brain's structure and functioning, neural networks consist of interconnected layers of artificial neurons designed to recognize patterns, learn from data, and make decisions or predictions. (credit: modification of work "Machine Learning & Artificial Intelligence" by www.vpnsrus.com/mikemacmarketing, CC BY 2.0)

Chapter Outline

7.1 Introduction to Neural Networks
7.2 Backpropagation
7.3 Introduction to Deep Learning
7.4 Convolutional Neural Networks
7.5 Natural Language Processing

Introduction

How does the human brain learn new information? This question has puzzled scientists for ages. The field of neuroscience is dedicated to finding out how the brain functions, but there are still so many things that we do not know. One major advance was the discovery of how neurons link up with one another inside the brain to make vast networks. Unlike manufactured electronic circuits that either send electronic signals or not, neurons in the brain fire at various strengths. The receiving neurons then transmit information by converting stimuli into electrical impulses, thereby *deciding* how important the incoming signals are and sending out signals in turn. It is thought that information is carried by the pattern of impulses in time over many groups of neurons. The net result is a complex “machine” whose function relies on trillions of connections, each with the ability to send signals along a continuum of intensity and with the capability of changing its structure when new information is encountered! This chapter will delve into the topics of deep learning and artificial intelligence.

Artificial intelligence (AI) is a branch of computer science and data science that aims to create intelligent systems capable of simulating humanlike cognitive abilities, including learning, reasoning, perception, and decision-making. Neural networks lie at the heart of many sophisticated AI systems, using models such as deep learning (covered in Introduction to Deep Learning). Neural networks and related AI algorithms use this inspiration to solve real-world problems, including image recognition and language generation.

In the early days of artificial intelligence research, efforts were significantly hindered by limited computational

power. For example, in the 1950s and 1960s, early AI programs like the General Problem Solver (GPS) and the Logic Theorist could only solve relatively simple problems due to the constraints of the hardware available at the time. These programs ran on machines that had limited memory and processing speed, which meant that even basic tasks could take an inordinate amount of time to complete. The inability to process large datasets or perform complex calculations in a reasonable time frame greatly restricted the potential of AI.

Fast forward to the present—advancements in computational power and technology have dramatically transformed the landscape of AI. One prominent example of this transformation is the development of deep learning models, such as those used in natural language processing (NLP). Modern AI systems, like ChatGPT developed by OpenAI, are powered by state-of-the-art hardware, which enables the training of large neural networks on vast datasets.

We discuss many of these applications of neural networks and AI as well as some of the issues surrounding the ethics of artificial intelligence and machine learning, especially regarding fair use of artistic materials.

7.1 Introduction to Neural Networks

Learning Outcomes

By the end of this section, you should be able to:

- 7.1.1 Define neural networks and discuss the types of problems for which they may be useful.
- 7.1.2 Summarize the roles of weights and biases in a neural network.
- 7.1.3 Construct a simple neural network.

Just imagine what must go on inside the human brain when tasked with recognizing digits, given the varying ways the digits 0-9 can be hand-drawn. Human readers can instantly classify these digits, even from an early age. The goal of neural networks is for a computer algorithm to be able to classify these digits as well as a human.

EXPLORING FURTHER

MNIST Database

The MNIST (Modified National Institute of Standards and Technology) database is a large dataset of handwritten digits that is frequently used for training various image processing and machine learning systems. This video presentation by Michael Garris (https://openstax.org/r/michae), senior scientist, provides an overview. You may download the MNIST dataset files (https://openstax.org/r/yann) directly for further exploration.

In this section, we introduce simple models of neural networks and discover how they work. Many technical details are deferred to later sections or are outside the scope of this text.

What Is a Neural Network?

A **neural network** is a structure made up of components called **neurons**, which are individual decision-making units that take some number of inputs $x_1, x_2, \ldots, x_n$ and produce an output y, as in Figure 7.2. How the output is determined by the inputs could be quite complex. Moreover, any two *neurons* may behave quite differently from each other on the same inputs.

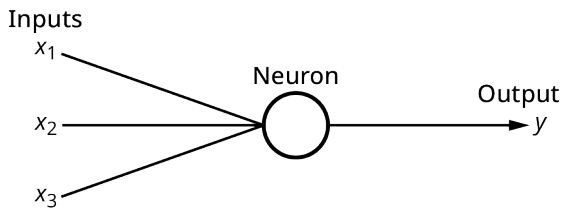


Figure 7.2 Single Neuron in a Neural Network

The neural network itself may consist of hundreds, thousands, or even millions of neurons connected to each other in *layers*, or groups of neurons that all receive the same inputs from previous layers and forward signals in aggregate to the next layer. There are always at least two layers, the **input layer** (containing neurons that accept the initial input data) and **output layer** (containing the neurons that are used to interpret the answer or give classification information), together with some number of **hidden layers** (layers between the input and output layers). Figure 7.3 shows a simple neural network diagram for reference.

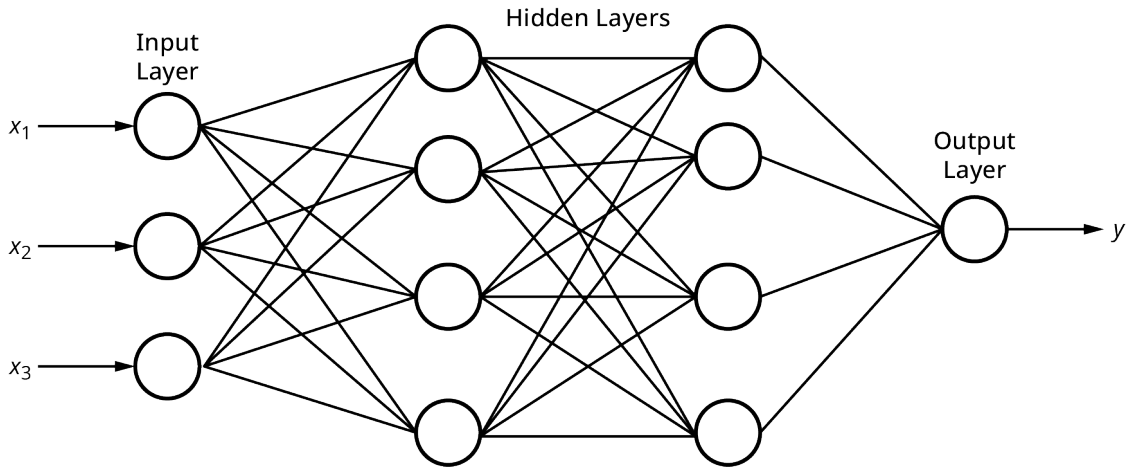


Figure 7.3 Neural Network Diagram. This neural network has four layers, two of which are hidden layers. There are three input neurons and one output neuron. In this example, all nodes in adjacent layers are connected, but some neural network models may not include all such connections (see, for example, convolutional neural networks in Introduction to Deep Learning).

The main purpose of a neural network is to classify complex data. Problems for which neural networks are especially well suited include the following:

- Image recognition, including facial recognition, identifying handwritten letters and symbols, and classifying parts of images. This is a huge area of innovation, with powerful tools such as TensorFlow developed by Google and PyTorch (https://openstax.org/r/pytorch) developed by Meta.
- Speech recognition, such as Google's Cloud Speech-to-Text (https://openstax.org/r/speech) service, offers accurate transcription of speech and translation for various languages.
- Recommendation systems, which are used to serve ads online based on each user's browsing habits, have been developed and used by Amazon, Meta, Netflix, and many other large companies to reach target markets.
- Anomaly detection has been developed to aid in fraud detection, data security, and error analysis/correction by finding outliers in large, complex datasets. An example is Microsoft's Azure Anomaly Detector (https://openstax.org/r/azure), which can detect anomalies in time series data.
- Autonomous vehicles and robotics, including Tesla's Autopilot technology, are becoming more and more prominent as automation alleviates some of the routine aspects of daily life and leads to increased efficiencies in business, manufacturing, and transportation.
- Generative art, examples of which include visual art, music, video, and poetry, leverage vast stores of human creative output to produce novel variations.
- Predictive text, including natural language processing models such as ChatGPT (see Convolutional Neural Networks).

EXPLORING FURTHER

TensorFlow

If you want to get your feet wet with neural networks, check out this interactive web-based neural network tool, called TensorFlow Playground (https://openstax.org/r/playground), which uses TensorFlow to train and update outputs in real time. There, you can choose a dataset from a list, adjust the number of hidden layers and neurons per layer by clicking the plus (+) and minus (−) buttons, and adjust the learning rate, choice of activation function, and other parameters (topics that we will learn more about in the rest of the chapter). Then click the "play" button to start training the model and watch as it learns how to classify the points in your chosen dataset! If you want to start over, just click "reset" and start from scratch!

Neurons, Weights, Biases, and Activation Functions

The way neurons work in a neural network is conjectured to be similar, or *analogous*, to how neurons work in the brain. The input of the neural network is fed into the neurons of the input layer. Each neuron then processes it and produces an output, which is in turn pushed to the neurons in the next layer. Individual neurons only send signals, or *activate*, if they receive the appropriate input required to activate. (**Activation** is the process of sending an output signal after having received appropriate input signals.) After passing through some number of hidden layers, the output of the last hidden layer feeds into the output layer. Lastly, the output of this final layer is interpreted based on the nature of the problem. If there is a single output neuron, then the interpretation could be *true* if that neuron is activated or *false* if not. For neural networks used to classify input into various classes (e.g., number recognition), there is usually one output neuron per class. The one that is *most* activated would indicate the classification, as shown in Figure 7.4.

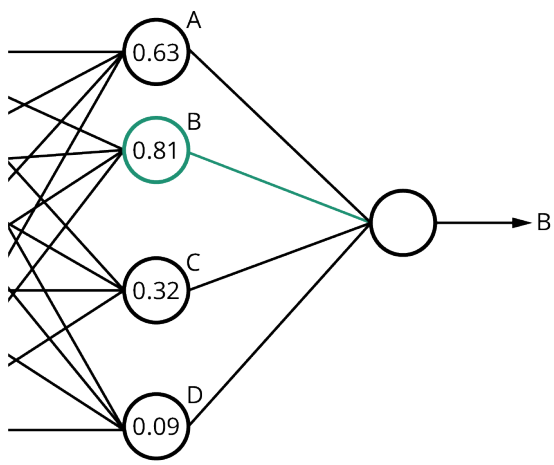


Figure 7.4 Output Neurons. In this figure, there are four output neurons, labeled A, B, C, and D. Since B has the highest activation level, the output of the neural network is B.

Each connection from one neuron to another has two parameters associated with it. The **weight** is a value w that is multiplied to the incoming signal, essentially determining the strength of the connection. The **bias** is a value b that is added to the weighted signal, making the neuron more likely (or less likely, if b is negative) to activate on any given input. The values of w and b may be positive, negative, or zero. Once the weight and bias are applied, the result is run through an **activation function**, which is a non-decreasing function f that determines whether the neuron activates and, if so, how strongly. (You'll see this later in Figure 7.6.)

Consider the simplest case of all, a neuron with single input x and output y. Then the flow of signal through the neuron follows the formula $y = f(wx + b)$. For example, suppose that the input value is $x = 0.87$, with weight $w = 0.53$ and bias $b = -0.12$. Then the signal would first be combined as $wx + b = (0.53)(0.87) + (-0.12) = 0.3411$. This value would be fed as input to the activation function to

produce $y = f(0.3411)$. In the next section, we will discuss various activation functions used in neural networks.

When there are multiple inputs, $x_1, x_2, \ldots, x_n$, each will be affected by its own weight. Typically, bias is thought of as a property of the neuron itself and so bias affects all the inputs in the same way. To be more precise, first, the inputs are multiplied by their individual weights, the result is summed, and then the bias is added. Finally, the activation function is applied to obtain the output signal.

$$y = f\left(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b\right) = f\left(\left(\sum_{i=1}^{n} w_i x_i\right) + b\right)$$

The weights and biases are parameters that the neural network learns during the training process, a topic we will explain in Backpropagation.

A typical neural network may have hundreds or thousands of inputs for each neuron, and so the equation can be difficult to work with. It would be more convenient to regard all the inputs $x_1, x_2, \ldots, x_n$ as parts of a single mathematical structure, called a **vector**. A **vector** is simply an ordered list of numbers, that is, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. (Note: In this text we use boldface letters to represent vectors. Also, the components of a vector are listed within a set of parentheses. Some texts vary on these notational details.) The number of components in a vector is called its **dimension**, so for example the vector $(5, 0, -2, 1.2, \pi)$ has dimension 5. Certain arithmetic operations are defined on vectors.

- Vectors of the same dimension can be added: If $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, then $\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$.
- Any real number can be multiplied to a vector: $k\boldsymbol{x} = k(x_1, x_2, \ldots, x_n) = (kx_1, kx_2, \ldots, kx_n)$.
- The **dot product** of two vectors of the same dimension results in a real number (not a vector) and is defined by:

$$\begin{aligned} \boldsymbol{x} \cdot \boldsymbol{y} &= (x_1, x_2, \ldots, x_n) \cdot (y_1, y_2, \ldots, y_n) = \textstyle\sum_{i=1}^{n} x_i y_i \\ &= x_1 y_1 + x_2 y_2 + x_3 y_3 + \ldots + x_n y_n \end{aligned}$$

If the inputs and weights are regarded as vectors, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, respectively, then the formula may be re-expressed more concisely as:

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

For example, if $\boldsymbol{w} = (0.3, -0.1, 0.9)$, $\boldsymbol{x} = (1.2, 0.4, -0.6)$, and $b = 0.1$, then

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = (0.3)(1.2) + (-0.1)(0.4) + (0.9)(-0.6) + 0.1 = -0.12$$

So in this example, the output would be $y = f(-0.12)$, the exact value of which depends on which activation function $f(x)$ is chosen.

Types of Activation Functions

Activation functions come in many types. Here are just a few of the most common activation functions.

1. **Step function**, $f(x) = \begin{cases} 0, \text{if } x < c \\ 1, \text{if } x \geq c \end{cases}$. The value of c serves as a threshold. The neuron only activates when the input is at least equal to a parameter c.
2. **Sigmoid function**, $\sigma(x) = \frac{1}{1+e^{-x}}$. Note, this is the same sigmoid function used in logistic regression (see Classification Using Machine Learning). Output values tend to be close to 0 when x is negative and close to 1 when x is positive, with a smooth transition in between.
3. **Hyperbolic tangent (tanh) function**, $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Output values have the same sign as x, with a smooth transition through 0.
4. **Rectified linear unit (ReLU) function**, $\text{ReLU}(x) = \max(0, x)$. The ReLU function is 0 for negative x values and equal to the input when x is positive.

5. **Leaky ReLU function**, $\text{LReLU}(x) = \max(cx, x)$, for some small positive parameter c. Leaky ReLU acts much like ReLU except that the values get progressively more negative when x gets more negative. Often, the optimal "leakiness" parameter c is determined during the training phase.
6. **Softplus function**, $f(x) = \ln(1 + e^x)$, which is a smoothed version of the ReLU function.

Figure 7.5 shows the graphs of the listed functions. A key feature of activation functions is that they are **nonlinear**, meaning that they are not just straight lines, $f(x) = mx + b$.

Another activation function that is important in neural networks, **softmax**, takes a vector of real number values and yields a vector of values scaled into the interval between 0 and 1, which can be interpreted as discrete probability distribution. (Recall from Discrete and Continuous Probability Distributions that discrete probability distributions provide measures of probability or likelihood that each of a finite number of values might occur.) The formula for softmax is:

$$\sigma(x_1, x_2, \ldots, x_n) = \left(\frac{e^{x_1}}{D}, \frac{e^{x_2}}{D}, \ldots, \frac{e^{x_n}}{D}\right)$$

where $D = e^{x_1} + e^{x_2} + \ldots + e^{x_n}$. We will encounter softmax activation in an example in Backpropagation.

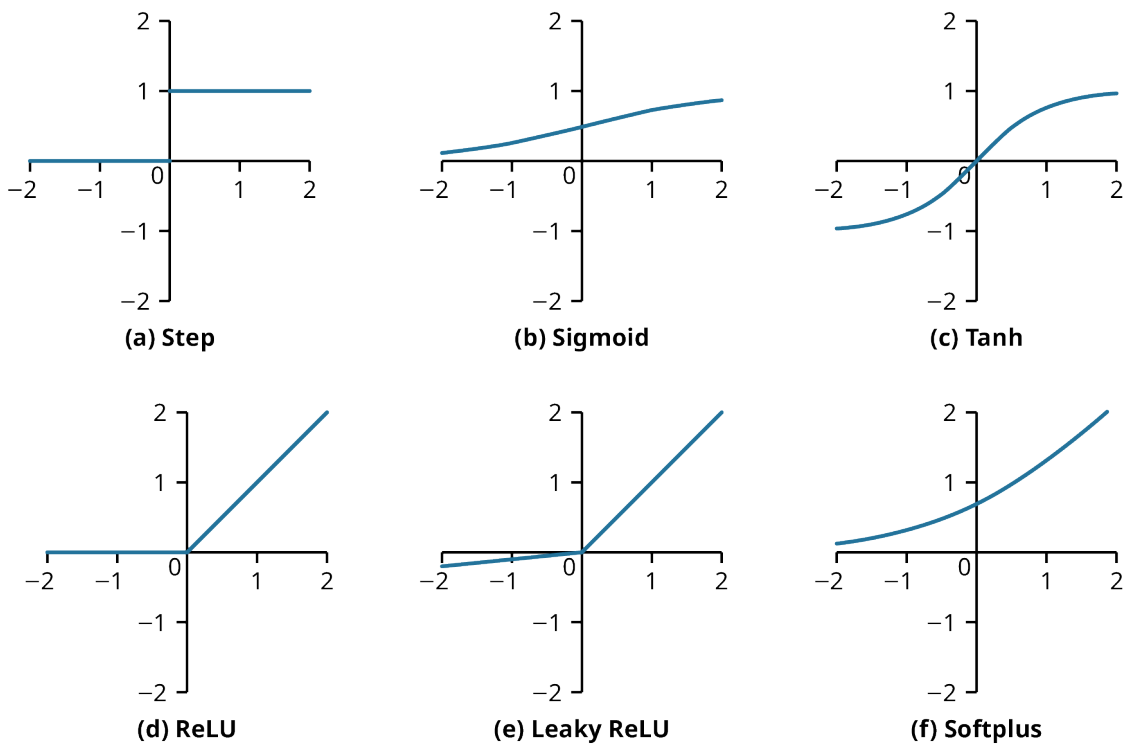


Figure 7.5 Graphs of Activation Functions. Top, from left to right, (a) step, (b) sigmoid, and (c) hyperbolic tangent (tanh) functions. Bottom, from left to right, (d) ReLU, (e) LReLU, and (f) softplus functions.

EXAMPLE 7.1

Problem

A simple neuron has four inputs, x_1, x_2, x_3, x_4, and one output, y. The weights of the four inputs are $w_1 = 0.34$, $w_2 = 0.07$, $w_3 = 0.59$, and $w_4 = -0.21$. The bias of the neuron is $b = 0.19$. Find the output in each of the following cases.

a. $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$. Activation function: ReLU.
b. $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$. Activation function: sigmoid.

c. $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$. Activation function: step, $f(x) = \begin{cases} 0, \text{if } x < 0.5 \\ 1, \text{if } x \geq 0.5 \end{cases}$.

d. $(x_1, x_2, x_3, x_4) = (0, 2.3, -1.6, 0.8)$. Activation function: softplus.

Solution

Setup for each part:

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b) = f(w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b) = f(0.34x_1 + 0.07x_2 + 0.59x_3 - 0.21x_4 + 0.19)$$

a. $\text{ReLU}(0.34(1) + 0.07(0) + 0.59(0) - 0.21(1) + 0.19) = \text{ReLU}(0.32) = 0.32$, since $0.32 > 0$

b. $\sigma(0.32) = \frac{1}{1+e^{-0.32}} = 0.58$ (Note, 0.32 was already computed in part a.)

c. $f(0.32) = 0$, since $0.32 < 0.5$ (Note, 0.32 was already computed in part a.)

d. $f(0.34(0) + 0.07(2.3) + 0.59(-1.6) - 0.21(0.8) + 0.19) = f(-0.761) = \ln\left(1 + e^{-0.761}\right) = -0.761$

In the next section, we explore neural networks that use the simplest of the activation functions, the step function.

Perceptrons

Although the idea of using structures modeled after biological neurons had been around prior to the development of what we would now consider neural networks, a significant breakthrough occurred in the middle of the 20th century. In 1957, Frank Rosenblatt developed the **perceptron**, a type of single-layer neural network designed for binary classification tasks—in other words, a neural network whose output could be "true" or "false" (see Figure 7.7). After initial successes, progress in AI faced challenges due to limited capabilities of AI programs and the lack of computing power, leading to a period of minimal progress, called the "Winter of AI." Despite these setbacks, new developments occurred in the 1980s leading to the **multilayer perceptron (MLP)**, which serves as the basic paradigm for neural networks having multiple hidden layers.

The single-layer perceptron learns weights through a simple learning rule, known as the perceptron learning rule.

1. Initialize the weights ($\boldsymbol{w}$) and bias (b) to small random values.
2. For each input ($\boldsymbol{x}$) in the training set,
   a. Compute predictions $\hat{y} = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$, where f is the step function,
   $$f(x) = \begin{cases} 0, \text{if } x < 0 \\ 1, \text{if } x \geq 0 \end{cases}$$
   b. Find the error $E = y - \hat{y}$, where y is the true or desired value corresponding to $\boldsymbol{x}$. Note that E can be positive or negative. Positive E implies the weights and/or bias is too low and needs to be raised. Negative E implies the opposite.
   c. Update the weights and biases according to the following formulas, where h is a small positive constant that controls the learning rate. Often the change of weights and biases will be small, not immediately causing a large effect on input. However, with repeated training, the perceptron will *learn* the appropriate values of $\boldsymbol{w}$ and b to achieve the desired output.
   $$\boldsymbol{w} \leftarrow \boldsymbol{w} + hE\boldsymbol{x}$$
   $$b \leftarrow b + hE$$

For example, suppose a perception with three neurons currently has weights $\boldsymbol{w} = (0.5, 0.7, 0.1)$ and bias $b = -0.9$. On input $\boldsymbol{x} = (0.6, 1, 0)$, we get:

$$\hat{y} = f((0.5)(0.6) + (0.7)(1) + (0.1)(0) - 0.9) = f(0.1) = 1$$

Suppose that the true output should have been $y = 0$. So there's an error of $E = 0 - 1 = -1$. If the learning rate is $h = 0.1$, then the weights and bias will be updated as follows:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + hE\boldsymbol{x} = (0.5 + (0.1)(-1)(0.6), 0.7 + (0.1)(-1)(1), 0.1 + (0.1)(-1)(0)) = (0.44, 0.6, 0.1)$$
$$b \leftarrow b + hE = -0.9 + (0.1)(-1) = -1.0$$

On the same input, the perceptron now has a value of:

$$\hat{y} = f((0.44)(0.6) + (0.6)(1) + (0.1)(0) - 1.0) = f(-0.136) = 0$$

In this simple example, the value $\hat{y}$ changed from 1 to 0, eliminating the error. However, there is no guarantee that the perceptron will classify all input without error, regardless of the number of training steps that are taken.

EXAMPLE 7.2

Problem

Suppose the next data point in the training set is $\boldsymbol{x} = (0, 0.5, 1)$, and suppose that the correct classification for this point is $y = 1$. With current weights $\boldsymbol{w} = (0.44, 0.6, 0.1)$ and bias $b = -1$, use the perceptron learning rule with $h = 0.1$ to update the weights and bias. Is there an improvement in classifying this data point?

Solution

$$\hat{y} = f((0.44)(0) + (0.6)(0.5) + (0.1)(1) - 1.0) = f(-0.6) = 0$$

Error: $E = 1 - 0 = 1$.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + hE\boldsymbol{x} = (0.44 + (0.1)(1)(0), 0.6 + (0.1)(1)(0.5), 0.1 + (0.1)(1)(1)) = (0.44, 0.65, 0.2)$$
$$b \leftarrow b + hE = -1 + (0.1)(1) = -0.9$$

Now we obtain $\hat{y} = f((0.44)(0) + (0.65)(0.5) + (0.2)(1) - 0.9) = f(-0.375)$, which still misclassifies the point as 0, so no improvement in accuracy. (However, at least the argument has increased from -0.6 to -0.375, so some progress has been made in training. In practice, many rounds of training may be necessary to achieve accurate results, with only incremental progress in each round.)

Building a Simple Neural Network

In What Is Data Science?, **Example 1.7**, we looked at the Iris Flower dataset to become familiar with dataset formats and structures. In this section, we will use the Iris dataset, iris.csv (https://openstax.org/r/datach7), to create a neural network with four inputs and one output neuron (essentially, a simple perceptron) to classify irises as either *setosa* or *not setosa*, according to four features: sepal length (SL), sepal width (SW), petal length (PL), and petal width (PW), all of which are measured in centimeters. The Iris dataset `iris.csv` has 150 samples of irises, consisting of 50 of each species, *setosa*, *versicolor*, and *virginica*. For the purposes of this example, the first 100 rows of data were selected for training and testing, using 75% of the data for training and the remaining 25% for testing. (Recall from Decision-Making Using Machine Learning Basics that it is good practice to use about 70%–80% of the data for training, with the rest used for testing.) The network will be trained using the perceptron learning rule, but instead of using the step function, the activation function will be the hyperbolic tangent, $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, which is chosen simply for illustration purposes.

THE IRIS FLOWER DATASET

The iris database was collected and used by Sir R. A. Fisher for his 1936 paper "The Use of Multiple Measurements in Taxonomic Problems." This work became a landmark study in the use of multivariate data in classification problems and frequently makes an appearance in data science as a convenient test case for

machine learning and neural network algorithms.

The model for the neural network is shown in Figure 7.6.

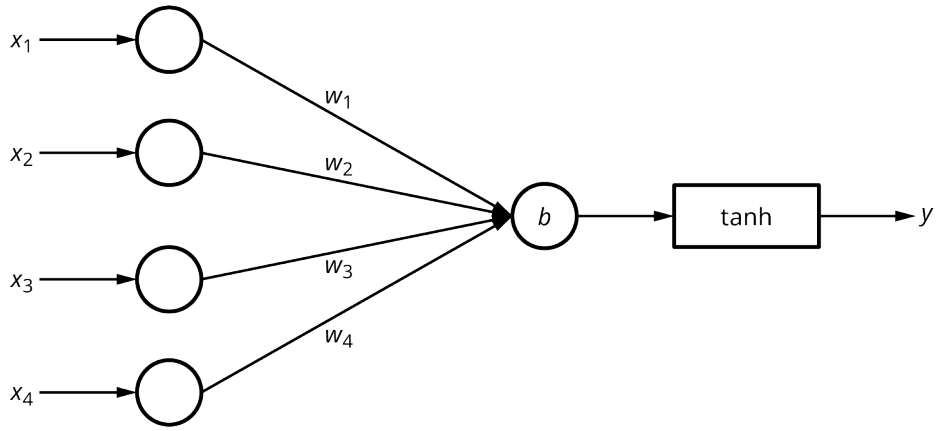


Figure 7.6 Simple Perceptron Neural Network Model. Here, x_1, x_2, x_3, x_4 are the inputs, w_1, w_2, w_3, w_4 represent the weights, and b represents the bias. The output, or response, is y.

The inputs (x_1, x_2, x_3, x_4) will take the values of SL, SW, PL, and PW, respectively. There are four corresponding weights (w_1, w_2, w_3, w_4) and one bias value b, which are applied to the input layer values, and then the result is fed into the tanh function to obtain the output y. Thus, the formula used to produce the predicted y values, denoted by $\hat{y}$, is:

$$\hat{y} = \tanh\left(w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b\right)$$

In this very simplistic model, we will interpret the value of y as follows:

- If $\hat{y} > 0$, classify the input as *setosa*.
- If $\hat{y} < 0$, classify the input as *not setosa*.

As for classifying error, we need to pick concrete target values to compare our outputs with. Let's consider the target value for classifying *setosa* as 1, while the target for classifying not *setosa* is -1. This is reasonable as the values of $\tanh(x)$ approach 1 as x increases to positive infinity and -1 as x decreases to negative infinity.

For example, if after training, the ideal weights were found to be $(w_1, w_2, w_3, w_4) = (-0.2, 0.6, -0.9, 0.1)$, and the bias is $b = 0.5$, then the response on an input $(x_1, x_2, x_3, x_4) = (5.1, 3.5, 1.4, 0.2)$ would be:

$$\hat{y} = \tanh\left((-0.2)(5.1) + (0.6)(3.5) + (-0.9)(1.4) + (0.1)(0.2) + 0.5\right) = \tanh(0.34) = 0.327$$

Since $\hat{y} > 0$, classify the input as *setosa*. Of course, the natural question would be, how do we obtain those particular weights and bias values in the first place? This complicated task is best suited for a computer. We will use Python to train the perceptron.

The Python library sklearn.datasets (https://openstax.org/r/scikit2) has a function, `load_iris`, that automatically loads the Iris dataset, so you should not need to import the dataset iris.csv (https://openstax.org/r/datach7). Other modules that are imported from `sklearn` are `Perceptron`, used to build the perceptron model, `train_test_split`, used to randomly split a dataset into a testing set and training set. Here, we will use a split of 75% train and 25% test, which is handled by the parameter `test_size=0.25`. We will also renormalize the data so that all features contribute equally to the model. The way to do this in Python is by using `StandardScalar`, which renormalizes the features of the dataset so that the mean is 0 and standard deviation is 1. Finally, `accuracy_score` is used to compute the accuracy of the classification.

PYTHON CODE

```python
# import the sklearn library (specific modules)
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data[:100] # Only taking first 100 samples for binary classification
y = iris.target[:100] # "0 for setosa", 1 for "not setosa"

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the perceptron
perceptron = Perceptron()
perceptron.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = perceptron.predict(X_test_scaled)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

The resulting output will look like this:

```
Accuracy: 1.0
```

Because the dataset was relatively small (100 rows) and well-structured, the perceptron model was able to classify the data with 100% accuracy (“Accuracy: 1.0”). In the real world, when datasets are larger and less regular, we do not expect 100% accuracy. There could also be significant overfitting in the model, which would lead to high variance when classifying data not found in the original dataset.

Here is how to use the perceptron model we just created to classify new data (outside of the original dataset):

PYTHON CODE

```
# Assume you have new inputs stored in a variable called 'new_inputs'
new_inputs = [[5.1, 3.5, 1.4, 0.2], # Example input 1
      [6.3, 2.9, 5.6, 1.8] # Example input 2
      ]

# Standardize the new inputs using the same scaler used for training
new_inputs_scaled = scaler.transform(new_inputs)

# Use the trained perceptron model to make predictions on the new inputs
predictions = perceptron.predict(new_inputs_scaled)

# Display the predictions
for i, prediction in enumerate(predictions):
print(f"Input {i+1} prediction: {'Setosa' if prediction == 0 else 'Not Setosa'}")
```

The resulting output will look like this:

```
Input 1 prediction: Setosa
Input 2 prediction: Not Setosa
```

7.2 Backpropagation

Learning Outcomes

By the end of this section, you should be able to:

- 7.2.1 Discuss the goal of adjusting weights and bias to reduce loss/error in a neural network.
- 7.2.2 Use static backpropagation to train a neural network.
- 7.2.3 Define recurrent neural networks and discuss their advantages/disadvantages.

In the previous section, we discussed the single-layer perceptron. Training the perceptron is straightforward as it only requires adjusting a set of weights and biases that directly affect the output. Neural networks that have more than one layer, such as multilayer perceptrons (MLPs), on the other hand, must be trained using methods that can change the weights and biases in the hidden layers as well.

Backpropagation is an algorithm used to train neural networks by determining how the errors depend on changes in the weights and biases of all the neurons, starting from the output layer and working backward through the layers, recursively updating the parameters based on how the error changes in each layer. In this section, we'll explore how neural networks adjust their weights and biases to minimize error (or loss), ultimately improving their ability to make accurate predictions. Fundamentally, backpropagation is a huge optimization problem. In this text, we focus on the intuitive ideas of this optimization problem rather than going into a deep dive of the methods and theory that are required to perform the optimization steps of backpropagation.

EXPLORING FURTHER

Backpropagation

A full treatment of backpropagation requires familiarity with matrix operations, calculus, and numerical analysis, among other things. Such topics fall well outside the scope of this text, there are many resources online that go into more depth, such as Neural Networks and Deep Learning (https://openstax.org/r/sanddeeplearning) by Michael Nielsen (2019) and What Is Backpropagation Really Doing? (https://openstax.org/r/backpropagation) by 3Blue1Brown.

Static Backpropagation

Backpropagation is a supervised learning algorithm, meaning that it trains on data that has already been classified (see What Is Machine Learning? for more about supervised learning in general). The goal is to iteratively adjust the weights of the connections between neurons in the network and biases associated with each neuron to minimize the difference between the predicted output and the actual target values. The term **static backpropagation** refers to adjustment of parameters (weights and biases) only, in contrast to **dynamic backpropagation**, which may also change the underlying structure (neurons, layers, connections, etc.).

Here's a high-level overview of the static backpropagation algorithm:

1. **Forward pass.** During the forward pass, input data is propagated through the network layer by layer, using the current values of weights and biases along with the chosen activation function, f. If the input vector is denoted by $\boldsymbol{x}^{(0)}$, and hidden layers by $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, etc., until we reach the output layer, $\boldsymbol{x}^{(t)}$, then each vector is obtained from the previous using a formula of the form $\boldsymbol{x}^{(k+1)} = f\left(\boldsymbol{w}\boldsymbol{x}^{(k)} + b\right)$. (Note: This formula hides a lot of complex number crunching. The values of the $\boldsymbol{w}$ and $\boldsymbol{b}$ are specific to each layer and so contain the weight and bias information for all the neurons in that layer. Indeed, $\boldsymbol{w}$ is no longer just a vector in this context, but a *matrix,* which may be regarded as a table of numbers. Finally, the activation function is applied to all entries of the vector input, resulting in a vector output. These details are not essential to the fundamental understanding of the process, though.)
2. **Error calculation.** Once the network produces an output, the error between the predicted output and the actual target value is computed. Error may be computed in many ways. The function used to measure error in a neural network is called a **loss or cost function**.
3. **Backward pass (backpropagation).** The error at each layer is recursively propagated backward through the network.
4. **Update parameters.** The weights and biases of the network are updated in such a way that reduces the error, typically using an optimization algorithm such as gradient descent (which we will discuss later).
5. **Repeat.** The steps are repeated on training data until a sufficient level of accuracy is achieved.

Error and Loss

Error generally refers to the difference between the predicted output of the neural network and the actual target values for a single data point. It is a quantitative measure of the mistakes made by the network in its predictions. The loss function is a mathematical function that takes the predicted output and the actual target values as inputs and outputs a single value representing the error. Learning is essentially minimizing the loss function over all training data. Suppose that the neural network outputs $\hat{\boldsymbol{y}} = \left(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\right)$ and the target output is $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$. Common loss functions that would measure the error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ include:

- **Mean squared error (MSE):** $\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$. The formula for MSE (introduced in Training and Testing a Model), is important due to its connection with the concept of variance in statistics. It is a commonly used loss function in regression tasks, where the goal is to predict continuous values.
- **Binary cross entropy loss**: $-\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \hat{y}_i + \left(1 - y_i\right) \ln\left(1 - \hat{y}_i\right)\right]$. This loss function is commonly

used in binary classification tasks—that is, when the output is either 0 or 1. (Note: The minus sign in front of the formula is there to make the overall value positive, as the values of the logarithms will generally be negative. The term *entropy* is related to the measures of information defined in Decision Trees.)

- **Hinge loss:** $\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\hat{y}_i\right)$. This function is also commonly useful in binary classification.
- **Sparse categorical cross entropy:** A generalization of binary cross entropy, useful when the target labels are integers.

Average loss is computed as an average (mean) of the loss over all the data points. The closer to zero the average loss is, the smaller the error in predictions. The choice of loss function depends on the specific task, the nature of the inputs and outputs, and many other factors, which will be discussed in more detail in Introduction to Deep Learning.

EXAMPLE 7.3

Problem

The outputs of a neural network on two data points are provided in Table 7.1:

Data Point	Predicted $\hat{y}$	Actual (Target) y
1	$(0.135, -0.252, 0.873, 1.297)$	$(1, 0, 1, 1)$
2	$(0.729, 0.458, 0.975, -0.025)$	$(1, 1, 1, 0)$

Table 7.1 Neural Network Output

Compute the average loss using the following loss functions.

a. MSE
b. Binary cross entropy loss
c. Hinge loss

Solution

Here, the number of terms is $n = 4$.

a. MSE

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 = \frac{1}{4}\left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \left(y_3 - \hat{y}_3\right)^2 + (y_4 - \hat{y}_4)^2\right]$$

Data point 1:

$$\text{Loss} = \frac{1}{4}\left[(1 - 0.135)^2 + (0 - (-0.252))^2 + (1 - 0.873)^2 + (1 - 1.297)^2\right] = 0.229$$

Data point 2:

$$\text{Loss} = \frac{1}{4}\left[(1 - 0.729)^2 + (0 - (-0.252))^2 + (1 - 0.873)^2 + (1 - 1.297)^2\right] = 0.092$$

$$\text{Average loss} : \tfrac{1}{2}(0.229 + 0.092) = 0.161$$

b. Binary cross entropy loss

$$-\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)\right]$$

Shortcut: If $y_i = 0$, the first term is zero, $y_i \ln \hat{y}_i = 0$, and the second term reduces to $\ln (1 - \hat{y}_i)$. If $y_i = 1$, the second term is zero, $(1 - y_i) \ln (1 - \hat{y}_i) = 0$, and the first term reduces to $\ln \hat{y}_i$.

Data point 1:

$$\text{Loss } = -\frac{1}{4}[\ln(0.135) + \ln(1-(-0.252)) + \ln(0.873) + \ln(1.297)] = 0.413$$

Data point 2:

$$\text{Loss } = -\frac{1}{4}[\ln(0.729) + \ln(0.458) + \ln(0.975) + \ln(1-(-0.025))] = 0.274$$

Average loss: $\frac{1}{2}(0.413 + 0.274) = 0.344$

c. Hinge loss

$$\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i \hat{y}_i\right)$$

First, compute the values of $1 - y_i \hat{y}_i$. Then if any of these are negative, you would use 0 instead in the sum.

Data point 1: $1 - y_1 \hat{y}_1 = 1 - (1)(0.135) = 0.865$. $1 - y_2 \hat{y}_2 = 1 - (0)(-0.252) = 1$.
$1 - y_3 \hat{y}_3 = 1 - (1)(0.873) = 0.127$. $1 - y_4 \hat{y}_4 = 1 - (1)(1.297) = -0.297$.

$$\text{Loss } = \frac{1}{4}[0.865 + 1 + 0.127 + 0] = 0.498$$

Data point 2: $1 - y_1 \hat{y}_1 = 1 - (1)(0.729) = 0.271$. $1 - y_2 \hat{y}_2 = 1 - (1)(0.458) = 0.542$.
$1 - y_3 \hat{y}_3 = 1 - (1)(0.975) = 0.025$. $1 - y_4 \hat{y}_4 = 1 - (0)(-0.025) = 1$.

$$\text{Loss } = \frac{1}{4}[0.271 + 0.542 + 0.025 + 1] = 0.460$$

Average loss: $\frac{1}{2}(0.498 + 0.460) = 0.48$

Note: The fact that each of the three loss functions gives a different result may signify the nature of the errors with respect to the model's performance, but with only two data points in this example, it is not advisable to place much meaning in these numbers.

Gradient Descent and Backpropagation

Since the goal of training a neural network is to reduce the value of error or loss, we essentially need to solve a (very large) optimization problem. Suppose that you are training a neural network. Consider all the weights and biases—of which there may be thousands, millions, or even more!—as parameters currently in the network that may be changed to improve the model. On a forward pass with an arbitrary input vector $\boldsymbol{x}^{(0)}$, the output $\hat{\boldsymbol{y}}$ is obtained and compared against the actual output $\boldsymbol{y}$ via a loss function, C. How would you be able to use this information to change the parameters (weights and biases) so that on the next pass with the same input $\boldsymbol{x}^{(0)}$, the output $\hat{\boldsymbol{y}}$ provides a more accurate representation of the actual output $\boldsymbol{y}$, in the sense that the value of C reduces? The key is to regard the whole neural network together with the loss calculation as a function F of the weights $\boldsymbol{W}$ and biases $\boldsymbol{B}$ and use then techniques such as gradient descent (which we will briefly discuss in a following section) to find a minimum value. Here, $\boldsymbol{W}$ and $\boldsymbol{B}$ are capitalized because they represent the multitude of weights and biases over the entire network, not just from a single neuron or layer. Figure 7.7 shows a schematic diagram illustrating the function F as a composite process.

As a very simple example, suppose that a neural network has only one neuron with weight w and bias b. Input x is fed into the neuron, the weight and bias are applied to x, and then an activation function is applied to the result to produce an output $\hat{y}$. We'll use the sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$ as activation function for simplicity. Now, the result, $\hat{y} = \sigma(wx + b)$, is compared to the true value y, using a cost function, which for this example will be MSE. The composite function F looks like this:

$$z = F(w, b) = \text{MSE}(\sigma(wx + b)) = \left(y - \frac{1}{1 + e^{-(wx+b)}}\right)^2$$

If there were more neurons, weights, and biases, then the formula would look much more complicated as we would need to keep track of many more variables and parameters, but the basic concept remains the same. We want to find values of w and b that minimize the value of $F(w, b)$. It takes many repeated cycles of adjusting weights and biases based on the values of the loss function. Each complete training cycle is called an **epoch**.

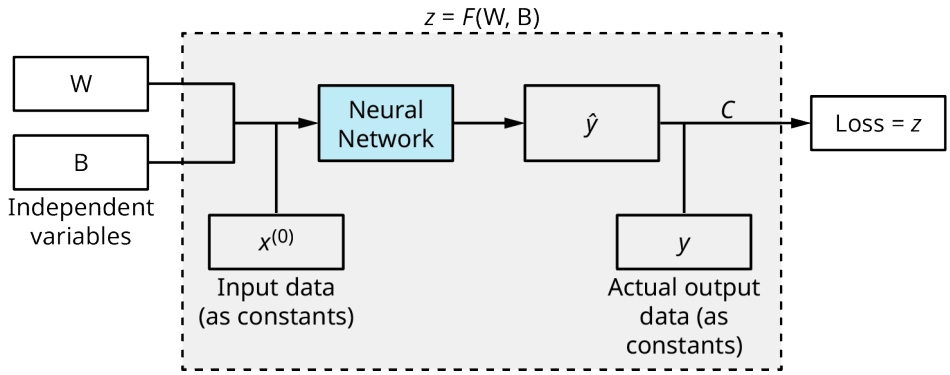


Figure 7.7 Neural Network with Loss Function C, Regarded as a Function of Weights and Biases. Given a set of weights $\boldsymbol{W}$ and biases $\boldsymbol{B}$, the neural network is run on a set of input data $\boldsymbol{x}^{(0)}$, the output $\hat{y}$ is obtained and compared against the actual output y via a loss function, C, providing a single number: the loss, z. The goal of training is to adjust $\boldsymbol{W}$ and $\boldsymbol{B}$ such that z is minimized.

Now suppose that $w = 0.5$ and $b = 0.25$. Consider a single point from the set of training data—for example, $(x, y) = (1, 0)$. The neural network computes a prediction, $\hat{y}$, as follows:

$$\hat{y} = \sigma(wx + b) = \frac{1}{1 + e^{-(0.5(1)+0.25)}} = 0.679$$

The loss with respect to the given actual output $(y = 0)$ is:

$$z = (y - \hat{y})^2 = (0 - 0.679)^2 = 0.46$$

Thus, we have the value $F(0.5, 0.25) = 0.46$, which indicates that there is some nonzero amount of loss on the prediction just made, and so the values of w and b should be adjusted. Instead of trying to adjust w and b individually, let's find the *direction* from the point (0.5, 0.25) that would result in the fastest decrease of the error z. If we label the direction of steepest decrease (dx, dy), then we could move from the point (0.5, 0.25) to a new point, $(0.5, 0.25) + h \cdot (dw, db) = (0.5 + h \cdot dw, 0.25 + h \cdot db)$, where h is a small number that affects the learning rate of the neural network. This is the main idea behind **gradient descent**. In our example, we could find the direction of steepest descent by simply looking at how the surface $z = F(w, b)$ is slanted near the point (0.5, 0.25). Figure 7.8 indicates the best direction with an arrow. (However, in practice, there will be no way to visualize the graph of loss with respect to weights $(\boldsymbol{W})$ and biases $(\boldsymbol{B})$ because the points $(\boldsymbol{W}, \boldsymbol{B}, z)$ exist in very high-dimensional spaces. The details of gradient descent and how it is used in neural networks falls outside the scope of this textbook. (See this IBM article (https://openstax.org/r/ibm) for a great introduction to this topic.)

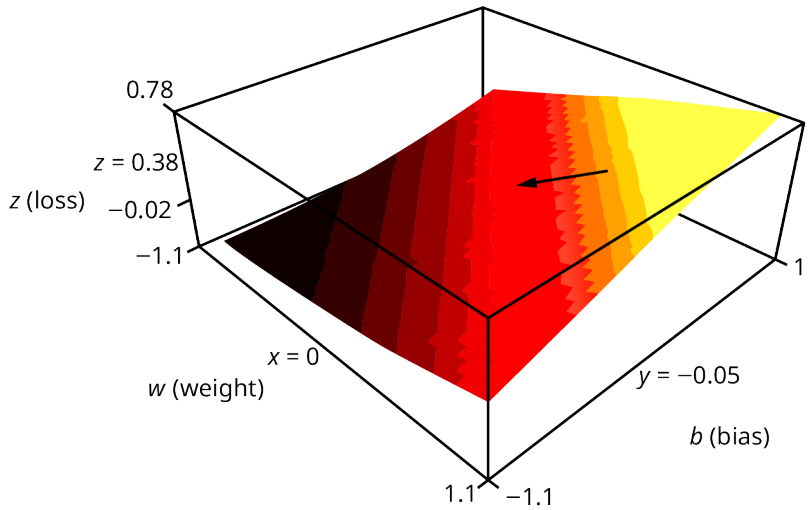


Figure 7.8 Simple Application of Gradient Descent. The point $(w, b) = (0.5, 0.25)$ has loss of $z = 0.46$. The arrow indicates the direction of steepest descent from the point $(0.5, 0.25, 0.46)$ on the surface.

One final point to mention is that gradient descent only works when the function F is *differentiable*, meaning that there are no corners, cusps, or points of discontinuity on the graph of $z = F(\boldsymbol{W}, \boldsymbol{B})$. Since F is a composite of many component functions, each component needs to be differentiable. Activation functions such as step, ReLU, and LReLU cannot be used with gradient descent. There are other methods available to backpropagate and train those kinds of networks. Alternatively, the non-smooth activation functions could be approximated by smooth versions—for example, sigmoid or tanh in place of step, or softplus in place of ReLU.

Training a Neural Network Using Backpropagation in Python using TensorFlow

The Python library TensorFlow (https://openstax.org/r/tensorflow), originally developed by the Google Brain team in 2015, can be used to build a neural network with backpropagation. (A **tensor** is a multidimensional array, generalizing the concept of vector.)

Let's set up a neural network and train it to recognize the handwritten letters in the MNIST database. This dataset, called mnist_784, can be loaded automatically using the `fetch_openml` command. The target labels must be converted to integers explicitly. Then the data is split into training (80%) and testing (20%) sets. The features are renormalized using `StandardScaler`. Next, the neural network model is created using `tf.keras.Sequential`. The model can be quite complex, but we will stick to the simplest case—one input layer with exactly as many neurons as features (784) and an output layer with 10 neurons, one for each digit. There are no hidden layers in this model. The command `tf.keras.layers.Dense` builds the output layer of 10 neurons that is fully connected to the input layer of 784 layers (the value of `X_train_scaled.shape[1]`). The softmax activation function is used, but other functions like ReLU are also available. The parameter `optimizer='adam'` in the function `model.compile` refers to an algorithm, *Adaptive Moment Estimation*, used to update the weights and biases of the model during training, the details of which are beyond the scope of this text.

Note: This code takes a while to execute—up to 5 minutes! Even with no hidden layers, the backpropagation takes some time to complete. When you run the code, you should see progress bars for each epoch (not shown in the Python output in the feature box).

PYTHON CODE

```
# import the libraries
import tensorflow as tf
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load MNIST Digits dataset
mnist = fetch_openml('mnist_784', version=1)

# Split the dataset into training and testing sets 80/20
X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target,
test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Cast the target variable as an integer (int32)
y_train = y_train.astype('int32')
y_test = y_test.astype('int32')

# Define the neural network architecture
model = tf.keras.Sequential([
tf.keras.layers.Dense(10, activation='softmax',
input_shape=(X_train_scaled.shape[1],))
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(X_train_scaled, y_train, epochs=10, batch_size=32,
validation_split=0.2)

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test_scaled, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

The resulting output will look like this:

```
Test Loss: 0.4051963686943054
Test Accuracy: 0.9189285635948181
```

Recurrent Neural Networks

Imagine if you could predict the stock market! What if you had a model that could take previous days' prices of

a stock and forecast the price of the stock tomorrow? Then you could maximize your profits by buying low and selling high with full knowledge of when those high and low points in price would occur. Given the extremely unpredictable nature of the stock market, it is unlikely that a simple forecasting model would be useful, so let's consider a neural network. You might set it up so that the input vector $\boldsymbol{x}^{(0)}$ holds 14 days of previous price data and the output y would be the predicted price of the stock on the next day. After a while, you decide to modify the design of the neural network to take in 30 days of price data. This would involve starting over, training and testing the new model from scratch. In fact, any time you decided to change the number of inputs, you would have to create a new model and go through the training and testing phases again and again. It would be much more convenient to have a model that can take different amounts of input data. Moreover, if the model first trains on 14 days of previous price data, then it would be desirable for the network to "remember" that training so when more data becomes available (30 days, 60 days, a year, etc.), it can simply improve its predictions from the baseline already established.

This flexibility is available in **recurrent neural networks (RNNs)**. An RNN is a neural network that incorporates **feedback loops**, which are internal connections from one neuron to itself or among multiple neurons in a cycle. Essentially, the output of a neuron is sent back as input to itself as well as going forward to the neurons in the next layer.

In the simplest such model, one feedback loop connects a single neuron with itself. The output z of the neuron is modified by a **connecting weight** w_c and the result included in the sum, making up the input of the same neuron. So when a new signal comes into the neuron, it gets the extra signal of $w_c z$ added to it. If the connecting weight is positive, then this generally causes the neuron to become more active over time. On the other hand, if the connecting weight is negative, then a *negative feedback loop* exists, which generally dampens the activity of the neuron over time. The connecting weights of an RNN are trained alongside all the weights and biases of the network using a variation of backpropagation called *backpropagation through time* (or *BPTT*). See Figure 7.9 for a simple diagram illustrating an RNN model.

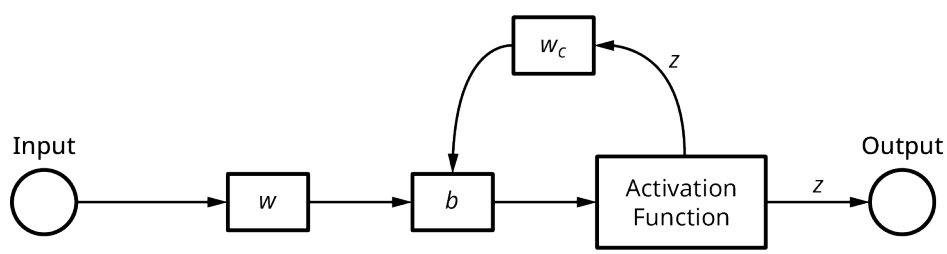


Figure 7.9 RNN Model, Simplified. The feedback loop gives the model a rudimentary *memory* of prior training.

The RNN model's feedback loops provide a simple memory, but how does it allow for different amounts of input values? The key is to feed in single (or small batches of) datapoints *sequentially*. For example, if $\boldsymbol{x}^{(0)} = \left(x_1, x_2, x_3, \ldots, x_{(n-2)}, x_{(n-1)}\right)$ represents the prices of a stock on day 1, 2, 3, up to n, and the actual value of the stock on day n is y, the RNN will take in just one element x_{n-1} (typically, the most recent data being used first) and adjust parameters based on the accuracy or loss of the prediction $\hat{y}_1$. On the next pass, it takes in $x_{(n-2)}$ and produces a new prediction $\hat{y}_2$. Keep in mind, the feedback loops allow the RNN to remember (in some capacity) how it performed on the first input, and so the prediction $\hat{y}_2$ is based on the two data points, $\left(x_{n-2}, x_{n-1}\right)$. Similarly, when x_{n-3} is fed into the model on the third pass, the RNN will produce a new prediction $\hat{y}_3$ that is based on the three data points, $\left(x_{n-3}, x_{n-2}, x_{n-1}\right)$. Thus, the feedback loops of an RRN provide a mechanism for the input to consist of any number of sequential input data points. This makes RNNs especially useful for time series data (See Time Series and Forecasting for an introduction to time series.)

Unrolling an RNN

The effect of feedback loops may be visualized by "unrolling" the RNN. Consider the simplest case of a single feedback loop from one neuron to itself. The effect of the connecting weight is equivalent to connecting to a copy of the same RNN (with identical weights and biases). Of course, since the second copy of the RNN has the

same feedback loop, it can be unrolled to a third, fourth, fifth copy, etc. An "unrolled" RNN with n copies of the neural net has n inputs. The effect of feeding the entire vector $\boldsymbol{x}^{(0)}$ into the unrolled model is equivalent to feeding the data points $(x_1, x_2, x_3, \ldots, x_n)$ sequentially into the RNN. Figure 7.10 shows how this works.

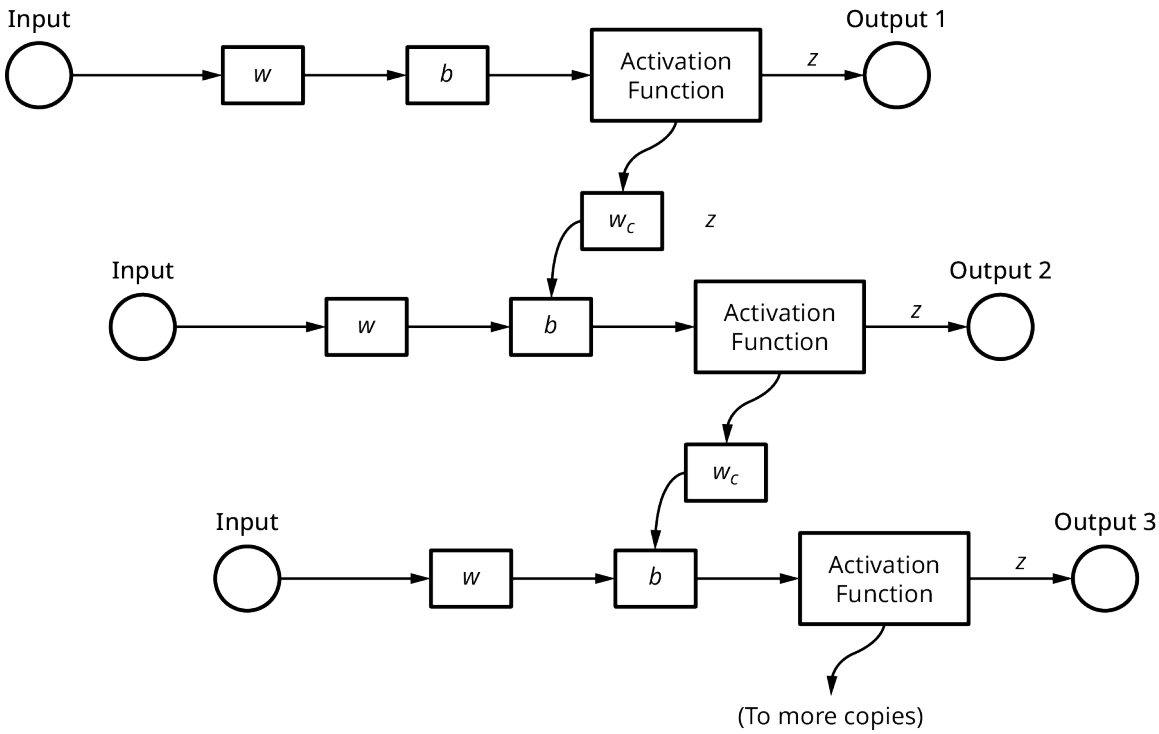


Figure 7.10 Unrolling an RNN

Vanishing/Exploding Gradient Problem

It is harder to train an RNN because the model can be very sensitive to changes in the connecting weights. During the training phase, gradient descent and a modified version of backpropagation (BPTT, as mentioned earlier) would be used to adjust all the weights and biases. However, because of the feedback loops in RNN, connecting weights can become compounded many times until they become very large. This causes algorithms like gradient descent to perform very poorly because the high weights cause proportionally high changes in parameters, as opposed to the tiny changes required to find minimum points. This is known as the **exploding gradient problem**. On the other hand, if connecting weights are too small to begin with, then training can cause them to quickly approach zero, which is called the **vanishing gradient problem**. Both issues are important to be aware of and addressed when working with RNNs; otherwise, the accuracy of your model may be compromised.

Long Short-Term Memory Networks

A **long short-term memory (LSTM) network** is a type of RNN designed to overcome the problems of exploding or vanishing gradients by incorporating **memory cells** that can capture long-term dependencies better than simple feedback loops can. LSTMs were introduced in 1997 and have since become widely used in various applications, including natural language processing, speech recognition, and time series forecasting.

LSTMs are generally more stable and easier to train than traditional RNNs, making them particularly well-suited for forecasting time series data with long-range trends and cyclic behaviors as well as working with sequential data that may seem much more unpredictable like natural language modeling, machine translation, and sentiment analysis. RNNs and LSTMs are a steppingstone to very sophisticated AI models, which we will discuss in the next section.

RNNs in Python

Consider the dataset MonthlyCoalConsumption.csv (https://openstax.org/r/datach7), which we analyzed using basic time series methods in Time Series and Forecasting. The dataset contains observations up to the end of 2022. Suppose you want to predict the monthly coal consumption in each month of the next year using a recurrent neural network. First, we will set up a simple RNN in Python. The library `TensorFlow` can be used to build an RNN. We will use `pandas` to load in the data and `numpy` to put the data in a more usable form (`np.array`) for the model. The RNN model is defined by the command `tf.keras.layers.SimpleRNN`, where `units` specifies the number of neurons in the layer and `activation='tanh'` is the activation function.

PYTHON CODE

```python
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler

# Load and preprocess the data
data = pd.read_csv('MonthlyCoalConsumption.csv')
values = np.array(data['Value'])
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_values = scaler.fit_transform(values.reshape(-1, 1))

# Prepare input sequences and target values
window_size = 12 # Number of months in each input sequence
X = []
y = []
for i in range(len(scaled_values) - window_size):
  X.append(scaled_values[i:i+window_size])
  y.append(scaled_values[i+window_size])
X = np.array(X)
y = np.array(y)

# Split data into training and testing sets
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Define the RNN architecture
model = tf.keras.Sequential([
  tf.keras.layers.SimpleRNN(units=32, activation='tanh',
                input_shape=(window_size, 1)),
  tf.keras.layers.Dense(1)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=16)

# Evaluate the model
loss = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
```

The resulting output will look like this:

```
Test Loss: 0.018054470419883728
```

The test loss is very low, and so we expect the model to be very accurate. Next, the model can be used to predict the unknown values in the next 12 months. Note: Since the original data was rescaled, the predictions made by the RNN need to be converted back into unnormalized data using the command `scaler.inverse_transform`.

PYTHON CODE

```
# Predict future values
future_months = 12
last_window = scaled_values[-window_size:].reshape(1, window_size, 1)
predicted_values = []
for _ in range(future_months):
prediction = model.predict(last_window)
predicted_values.append(prediction[0, 0])
last_window = np.append(last_window[:, 1:, :], prediction.reshape(1, 1, 1), axis=1)

# Inverse transform the predicted births to get actual values
predicted_values = scaler.inverse_transform(np.array(predicted_values).reshape(-1,
1))
print("Predicted Values for the Next 12 Months:\n", predicted_values.flatten())
```

The resulting output will look like this:

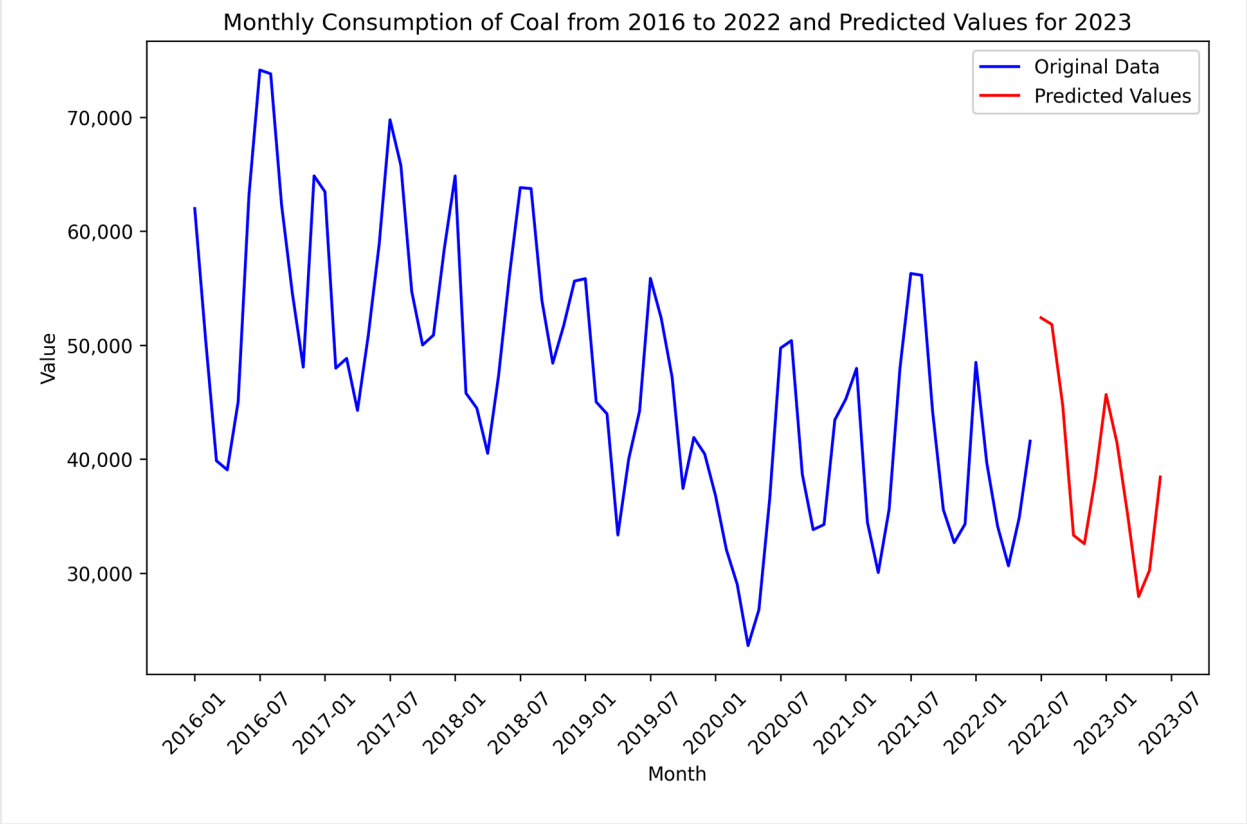
```
Predicted Values for the Next 12 Months:
[49227.05 49190.258 41204.16 30520.389 29591.129 34516.586 42183.016
38737.39 32026.324 23482.027 23381.914 30206.467]
```

Finally, here is the plot containing the original data together with the predicted values. Note: There are a lot of lines of code devoted to formatting the data and axis labels before graphing. This is, unfortunately, the nature of the beast. The full details of what each piece of code does will not be explained thoroughly in this text.

PYTHON CODE

```
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator ## for graph formatting
from matplotlib.ticker import FuncFormatter ## for formatting y-axis

# Function to format the Y-axis values
def y_format(value, tick_number):
return f'{value:,.0f}'

# Plot original time series data and predicted values
plt.figure(figsize=(10, 6))
plt.plot(data['Month'], data['Value'], label='Original Data', color='blue')
plt.plot(range(len(data['Value']), len(data['Value']) + future_months),
predicted_values,
    label='Predicted Values', color='red')

# Create a list for the months
pd_months = pd.date_range(start='2016-01-01', end='2023-07-01', freq='MS')

xticks_positions = range(0, len(pd_months), 6) # Positions to display ticks (e.g.,
every 12 months)
xticks_labels = [pd_months[pos].strftime('%Y-%m') for pos in xticks_positions] #
Labels corresponding to positions

plt.xlabel('Month')
plt.ylabel('Value')
plt.title('Monthly Consumption of Coal from 2016 to 2022 and Predicted Values for
2023')
plt.legend()
plt.xticks(ticks=xticks_positions, labels=xticks_labels, rotation=45)
# Apply the formatter to the Y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(y_format))

plt.show()
```

The resulting output will look like this:



7.3 Introduction to Deep Learning

Learning Outcomes

By the end of this section, you should be able to:

- 7.3.1 Discuss the role of hidden layers in a neural network.
- 7.3.2 Describe loss/error functions and their role in training and testing a neural network.
- 7.3.3 Set up, test, and train a deep learning neural network that can classify real-world data.

Deep learning is a general term for the training and implementation of neural networks with many layers to learn the relationships of structured representations of data. A subset of machine learning, deep learning has revolutionized various fields, ranging from computer vision and natural language processing to robotics and health care. Led by research from organizations such as Google DeepMind, Meta, IBM, MosaicML, Databricks, OpenAI, and others, deep learning has achieved remarkable outcomes, including AlphaGo Zero's unprecedented mastery of the ancient and complex game of Go. With its ability to automatically learn hierarchical representations from raw data, deep learning has unlocked new frontiers in pattern recognition, decision-making, and creativity. This chapter has already laid out the foundations of neural networks in previous sections, and the following section will focus on developing deep learning neural network models that can classify hand-drawn numbers and simple images using the `TensorFlow` library within the Python language.

Neural Networks with Many Layers

The hidden layers in a neural network play a crucial role in transforming input data into useful representations that can be used to make predictions or decisions. Suppose that we are trying to train a neural network to recognize handwritten digits (as in Figure 7.2). Ideally, each hidden layer should pick up on identifiable features—for example, whether a digit has a loop/circle (like 0, 6, 8, and 9) or not (like 1, 2, 3, 4, 5, 7). Other hidden layers may then look for presence of vertical or horizontal strokes to further assist in classifying the

numeral. However, hidden layers do not necessarily pick up on the same kinds of patterns that humans would. A recent direction in deep learning is in developing models that are capable of extracting higher-level features (e.g., loops, edges, textures) and reporting *reasons* why they made the decisions they made. Sophisticated image classification models can locate patterns, parts of images, and even spatial relationships between different parts of an image, all within the deeper layers of the network.

What makes deep learning *deep*? The **depth** of a neural network is the number of hidden layers it contains and is a defining characteristic of deep learning. Deeper networks have more capacity to learn complex patterns and relationships in the data. By stacking multiple layers of hidden units, deep learning models can learn to extract features at multiple levels of granularity, in a hierarchical (structured) manner, leading to more powerful and expressive representations of the input data. This hierarchical feature learning is key to the success of deep learning in various domains, including computer vision, natural language processing, and speech recognition, where complex patterns and relationships need to be captured from very high-dimensional data—that is, datasets that have many feature variables.

Loss Functions and Their Role in Training Neural Networks

You encountered many of the most common loss functions in Introduction to Neural Networks and Backpropagation. This section presents situations in which one loss function may be preferable over another. For instance, *MSE (mean squared-error)* is often used when the network is trained to predict continuous values, such as in regression analysis. For example, think of a neural network that is trained to provide a probability P that a patient who displays a certain set of symptoms has cirrhosis. Because MSE measures the square of differences between the predicted and actual values, MSE penalizes larger errors heavily and will be very sensitive to outliers. Moreover, MSE is not scale-independent, meaning that the units of MSE are not the same as the units of the data. The following example illustrates this last point.

EXAMPLE 7.4

Problem

A neural network is being trained to predict the maximum height of a variety of corn plant based on a number of features, such as soil acidity, temperatures during the growing season, etc. A sample of the predicted values along with the corresponding actual values are given in Table 7.2 in units of meters and centimeters. Note: The data is the same in both rows, but expressed in different units. Find the MSE for both rows of predictions. By what factor does MSE change?

	Predicted $\hat{y}$	Actual (Target) y
Units: m	$(1.18, 1.67, 1.12, 1.38, 1.42)$	$(1.07, 1.46, 1.05, 1.43, 1.45)$
Units: cm	$(118, 167, 112, 138, 142)$	$(107, 146, 105, 143, 145)$

Table 7.2 Predicted vs. Actual Values of Corn Height

Solution

Here, the number of terms is $n = 5$. For the data measured in m:

$$\begin{aligned} \text{MSE} &= \tfrac{1}{5}\left[(1.07-1.18)^2 + (1.46-1.67)^2 + (1.05-1.12)^2 + (1.43-1.38)^2 + (1.45-1.42)^2\right] \\ &= 0.0128 \end{aligned}$$

For the data measured in cm:

$$\text{MSE} = \frac{1}{5}\left[(107-118)^2 + (146-167)^2 + (105-112)^2 + (143-138)^2 + (145-142)^2\right] = 128$$

We find that the MSE changes by a factor of $\frac{128}{0.0128} = 10{,}000$, while the numeric values of the data only scaled by a factor of 100. Given that the dataset represents the same exact measurements, this example shows one of the major pitfalls in using MSE as a loss function.

Other commonly used loss functions include binary cross entropy, hinge loss, and (sparse) categorical cross entropy.

Binary cross entropy is more suitable for binary classification tasks—that is, deciding *True* or *False* based on the given input vector. More generally, if the output of a neural network is a probability of belonging to one of two classes (let's say 0 and 1), then binary cross entropy tends to do a better job as a loss function for training the network. The main reasons are that binary cross entropy heavily penalizes misclassification, works well even on **imbalanced data** (i.e., datasets that have significantly more of one class than another), and is well-suited to handle output that may be interpreted on a probability scale from 0 to 1.

Hinge loss is another common loss function for binary classification tasks. It is suitable for scenarios where the goal is to maximize the margin between classes while minimizing classification errors. We define the **margin** between classes as a measure of the separation of data points belonging to different classifications. Larger margins typically suggest more well-defined classifications. For example, if the set of scores on a quiz are (71, 73, 78, 80, 81, 89, 90, 92), then there seems to be a more natural separation between the subsets of (71, 73), (78, 80, 81), and (89, 90, 92) compared to the usual binning of scores by their tens places into (71, 73, 78), (80, 81, 89), and (90, 92). In the first case, there is margin[1] of 5 units between (71, 73) and (78, 80, 81) and a margin of 8 units between (78, 80, 81) and (89, 90, 92), compared to the margins of 2 and 1 between the subsets (71, 73, 78), (80, 81, 89), and (80, 81, 89), (90, 92), respectively. Thus, it seems more reasonable (based on the scores and their margins) to assign a grade of A to students who scored in the subset (89, 90, 92), B to those in the subset (78, 80, 81), and C to those in the subset (71, 73).

Sparse categorical cross entropy is a generalization of binary cross entropy useful for multi-class classification tasks, where the network predicts the probability distribution over multiple classes. It measures the similarity between the predicted class probabilities and the true class labels. The term *sparse* refers to the nature of the output as integers versus one-hot encoded vectors (recall *one-hot encoding* from Other Machine Learning Techniques). (Sparse) categorical cross entropy is often paired with softmax activation in the output layer of the network to ensure that the predicted probabilities sum to 1 and represent a valid probability distribution.

Example: Using Deep Learning to Classify Handwritten Numerals

`TensorFlow` has robust functionality for neural networks that incorporate hidden layers using various activation functions. It is very easy to add hidden layers, as you can see in the following code. Compare the `tf.keras.Sequential` function from the similar example from Backpropagation. The model still begins with an input layer of size equal to the number of features. The input neurons are densely connected to a hidden layer of 128 neurons using the `ReLU` activation function. The next hidden layer has 64 neurons, again using `ReLU` as activation function. Finally, the output layer has 10 neurons (one for each digit) and softmax activation. It is a common practice to use a simpler activation function (like `ReLU`) in hidden layers that have many connections, while more sophisticated activation functions are appropriate at the output layer.

Note that the accuracy of this model improves to 0.967, as compared to 0.919 from our previous model that lacked hidden layers.

1 Technically, margin refers to the distance between the closest data point in a class and a *separation hyperplane* that would serve as a boundary with respect to some other class of data point. These topics are important in *support vector machines (SVMs)*, another type of supervised learning model for classification and regression tasks, which we do not cover in this text.

PYTHON CODE

```python
# import the libraries
import tensorflow as tf
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load MNIST Digits dataset
mnist = fetch_openml('mnist_784', version=1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target,
test_size=0.2, random_state=42)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Cast the target variable as an integer (int32)
y_train = y_train.astype('int32')
y_test = y_test.astype('int32')

# Define the neural network architecture
model = tf.keras.Sequential([
tf.keras.layers.Dense(128, activation='relu',
input_shape=(X_train_scaled.shape[1],)),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(X_train_scaled, y_train, epochs=10, batch_size=32,
validation_split=0.2)

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test_scaled, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

The resulting output will look like this:

```
Test Loss: 0.20183834433555603
Test Accuracy: 0.968999981880188
```

7.4 Convolutional Neural Networks

Learning Outcomes

By the end of this section, you should be able to:

- 7.4.1 Define convolutional neural networks and describe some problems where they may perform better than standard neural networks.
- 7.4.2 Describe feature maps.
- 7.4.3 Train and test a convolutional neural network on image data and discuss the results.

Convolutional neural networks (CNNs) are a powerful class of neural network models developed to process structured, grid-like data, such as images, making use of the mathematical operation of *convolution* (which is similar to applying a filter or mask to an image). CNNs are useful for image classification, locating objects within images, edge detection, and capturing spatial relationships that exist within an image. We'll discuss these applications, and we'll explore the concept of **feature maps**, which are the output of convolutional layers in CNNs and represent the learned features of the input data. By the end of this section, you should see how to train and test a CNN to classify image data using Python.

The Components of Convolutional Neural Networks

CNNs consist of multiple hidden layers, including convolutional layers, pooling layers, and fully connected layers. The key components of a CNN are as follows:

- **Convolutional layers**: These layers apply "filters" to the input data that are capable of picking up on distinct structural features, such as vertical or horizontal lines/edges in an image. This is where the mathematical convolution operations are applied and feature maps produced, topics we shall discuss further later in the chapter.
- **Pooling layers**: Pooling layers *down-sample* the feature maps produced by the convolutional layers, reducing the spatial dimensions of the data while retaining important information. To illustrate, consider the list of data [0, 0, 0, 1, 1, 1, 0, 0, 0, 2, 2, 2]. This data can be compressed to [0, 1, 0, 2]. (In real applications, some kind of averaging or voting procedure is used to determine the down-sampled values.)
- **Fully connected layers**: These layers are typically used at the end of the network to perform classification or regression tasks based on the learned features extracted by the convolutional layers. Fully connected layers connect every neuron in one layer to every neuron in the next layer, as in a traditional neural network.

CNNs excel in tasks that include capturing spatial patterns and finding hierarchical features in images. In addition to image classification and object detection, CNNs can be used for **semantic segmentation**, which involves classifying each pixel of an image into a specific category or class. For example, in Figure 7.11, the parts of an image are analyzed and marked as different types, such as *buildings*, *vehicles*, and *road*. This kind of classification is an important aspect of developing safe self-driving cars.

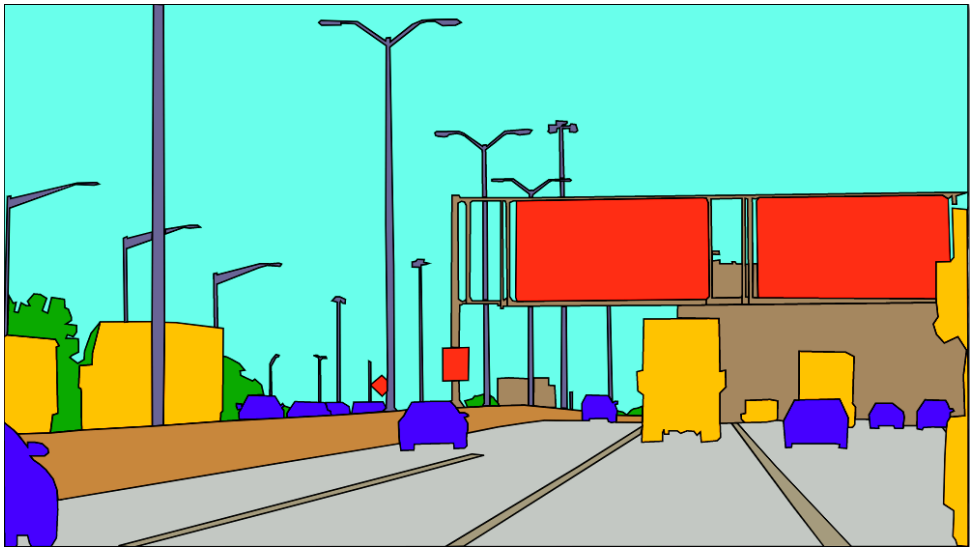


Figure 7.11 Semantic Segmentation of an Image. Small cars appear in blue, large vans/trucks in yellow, traffic signs in red, and buildings and other barriers in brown. (credit: "Semantic Segmentation Image Annotation Kotwel" by Kotwel Inc./Flickr, CC BY 2.0)

Feature Maps

In a typical CNN, after the data has been sent through convolutional and pooling layers, the CNN can then start to extract various features from the input. This step relies on filters or kernels that are specifically designed to pick up on specific patterns or features, such as edges, textures, or shapes, in different regions of the input data. The output of applying these filters to the input data results in **feature maps**, which are essentially two-dimensional arrays of values representing the activations of the filters across the spatial dimensions of the input. Each element of a feature map corresponds to the activation of a specific filter at a particular location in the input data.

Feature maps play a crucial role in the representation learning process of CNNs. They capture hierarchical features at different levels of abstraction, with early layers capturing low-level features (e.g., edges, corners) and deeper layers capturing more complex and abstract features (e.g., object parts, semantic concepts). By stacking multiple convolutional layers, CNNs learn to progressively extract (and abstract) features from raw input data, enabling them to effectively model complex patterns and relationships in the data.

Examples of some common features that may be captured in feature maps include the following:

- Edges and textures. These are the fundamental features useful in recognizing digits or letters as well as well-defined boundaries between parts of images.
- Color and intensity variations. Determining actual color information from a digital image is often very difficult. The same color may appear drastically different depending on factors such as lighting, surrounding colors, and intensity.
- Shapes and structures. Building upon edge-detection features, shapes and larger structures can be built up as higher-level features.
- Object parts and high-level patterns. Even higher in the hierarchy of image features, objects may be analyzed into parts, and non-obvious patterns may be discovered by the CNN.
- Semantic concepts (see *semantic segmentation*, mentioned previously). Outside of the realm of image analysis, feature maps may be synthesized to detect semantic concepts—that is, parts of the data that work together to form meaningful objects or ideas. Learning to recognize semantic concepts is key to the successful operation of natural language learning models.
- Temporal dynamics. In sequential data, such as audio signals or time series data, feature maps may capture temporal dynamics and patterns over time. These features help the network understand temporal dependencies and make predictions based on sequential input.

Image Analysis Using CNNs

Image analysis using semantic segmentation is a computer vision task that involves assigning semantic labels (meanings) to each pixel in an image, thus dividing the image into semantically meaningful regions or segments. Unlike traditional image classification tasks, where the goal is to assign a single label to the entire image, semantic segmentation provides a pixel-level understanding of the scene by labeling each pixel with the corresponding class or category it belongs to.

In semantic segmentation, each pixel in the image is assigned a label that represents the object or category it belongs to, such as "person," "vehicle," "tree," or "road." The output is a segmented image where different regions or objects are delineated by distinct colors or labels (as shown in Figure 7.11). This fine-grained understanding of the image allows for more detailed analysis and interpretation of visual scenes, making semantic segmentation a crucial task in various computer vision applications, including autonomous driving, medical image analysis, scene understanding, and augmented reality.

Semantic segmentation is typically performed using deep learning techniques, particularly convolutional neural networks, which have shown remarkable success in capturing spatial dependencies and learning complex visual patterns. By training CNNs on annotated datasets containing images and corresponding pixel-level labels, semantic segmentation models can learn to accurately segment objects and scenes in unseen images, enabling a wide range of advanced visual perception tasks.

The process of training a CNN to perform sematic segmentation is more involved than in image classification tasks. It is a supervised learning task, in that the training set must have labels assigned to each region of each image. The tedious work of labeling regions of images falls to human researchers.

7.5 Natural Language Processing

Learning Outcomes

By the end of this section, you should be able to:

- 7.5.1 Provide a brief history of the significant developments of natural language processing.
- 7.5.2 Discuss the importance of speech recognition and text-to-speech algorithms in everyday life and provide some examples.
- 7.5.3 Discuss how ChatGPT may be used effectively to speed certain kinds of tasks while also recognizing ethical issues in the use of ChatGPT and similar technologies.

AI and deep learning have progressed rapidly in the last decade. Some of the most significant breakthroughs have come from **natural language processing**, or **NLP**, which is an area of AI concerned with recognizing written or spoken language and generating new language content. From its humble beginnings in the 1950s, marked by rudimentary language translation programs and rule-based systems, through the development of neural network models, NLP has undergone a remarkable evolution. At the time of this writing, powerful NLP platforms such as OpenAI's **ChatGPT** and Microsoft's *Copilot* have captured everyone's interest with their uncanny ability to understand and respond to complex prompts, engaging users in natural and coherent conversations that simulate humanlike interactions. ChatGPT and Copilot are also known as a **large language model**, or **LLM**, which is a type of NLP model that is trained on massive datasets containing billions or even trillions of words. For an example of what ChatGPT looks like in action, see Figure 7.12. You can also go to https://chat.openai.com/ (https://openstax.org/r/chat) to use ChatGPT for free!

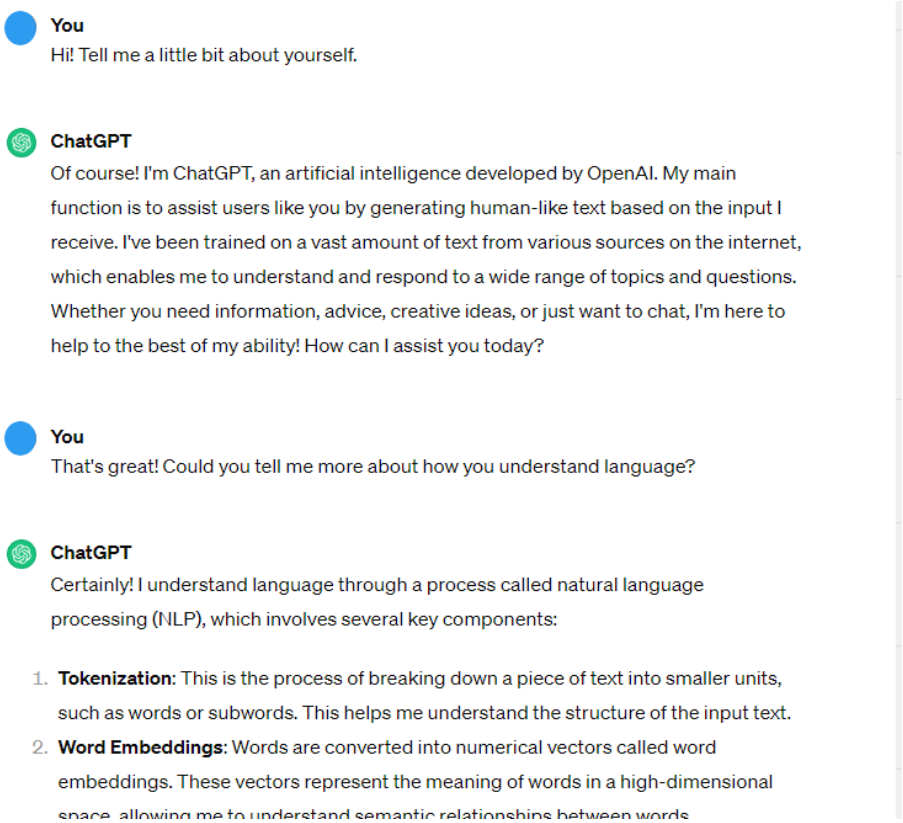


Figure 7.12 ChatGPT Conversation
OpenAI. (2024). *ChatGPT* (June 16 version) [Large language model]. https://chat.openai.com/chat (Created at: https://chat.openai.com/ (https://openstax.org/r/chat) using the prompt: "Hi! Tell me a little about yourself," followed by the prompt: "That's great! Could you tell me more about how you understand language?" Note: Each use of this prompt will produce varying output.)

NLP models may act as *force multipliers*, enhancing human capabilities by automating routine tasks and augmenting human intelligence. For example, they can assist in preparing form letters, summarizing information, analyzing sentiment in large volumes of text, extracting key insights from documents, translating from one language to another, and even generating creative content such as articles, stories, or poetry. You may be wondering whether NLP was used to help write this text. While almost every aspect of the writing was indeed handled by humans, ChatGPT has often been helpful in brainstorming ideas, finding out basic information, and providing a starting point for some of the content writing. You are currently experiencing the future of AI, where more and more of the tasks that humans consider mundane or routine can be offloaded to computers!

A Brief History of NLP

Key milestones, from the creation of ELIZA (an early "chatbot," or conversational agent—see Figure 7.13) in the 1960s to the advent of statistical language models in the 1990s, have paved the way for the emergence of the so-called *transformer models* and state-of-the-art language models like ChatGPT in the 2010s and beyond. **Transformer models** are a type of neural network architecture that use a very sophisticated internal memory system to capture long-range and large-scale dependencies in the input data. Today, NLPs power a diverse array of real-world applications.

Figure 7.13 A Screenshot of Eliza (credit: "ELIZA mostly restates content entered by the user to elicit further information. This tactic can help people feel like they're having a meaningful conversation." by Rosenfeld Media/Flickr, CC BY 2.0)

Here is a timeline of significant milestones in the development of NLPs. This topic is too vast to cover in this text, and so we will only mention some of the important models without going into detail. Table 7.3 highlights several key developments.

Year	Development
Mid-1960s	*ELIZA*, a computer program capable of simulating conversation, was created by Joseph Weizenbaum at MIT.
Late 1980s to 1990s	Statistical language models such as *hidden Markov Models (HMMs)* (a type of statistical model useful for predicting sequential data, such as speech and handwriting) and *n-gram* models (models based on predicting the next, or *n*th, term of a sequence based on the previous $n - 1$ terms), gained popularity for tasks like speech recognition and machine translation.
2001	The introduction of the *BLEU* metric for evaluating machine translation systems by Kishore Papineni and colleagues.
2003	*Moses*, an open-source statistical machine translation system, was introduced by Philipp Koehn and colleagues.
2013	*Word2vec*, a word embedding technique, was introduced by Tomas Mikolov and colleagues at Google. *Word embedding* refers to translating words into vector representations in high-dimensional vector spaces in such a way that preserves certain relationships among the words.
2017	The *Transformer* architecture, introduced in the paper "Attention Is All You Need" by Ashish Vaswani et al., revolutionized the world of NLPs. Instead of considering words one by one, a transformer model looks at sentences, paragraphs, and larger groups of words using a paradigm called *self-attention*, which allows the model to evaluate the importance of different words or phrases within a larger context.
2018	OpenAI introduced the first *GPT* (Generative Pretrained Transformer), a large-scale unsupervised language model.
2018	*BERT (Bidirectional Encoder Representations from Transformers)*, a transformer-based language model, was introduced by researchers at Google.

Table 7.3 Brief History of NLP

Year	Development
2020	OpenAI released *GPT-3* (Generative Pretrained Transformer 3), an even larger and more powerful language model, with up to 175 billion parameters.
2022	*ChatGPT*, a variant of GPT-3 fine-tuned for conversational tasks, was released, gaining widespread attention for its impressive performance in generating humanlike responses.
2023	Enhancements focused on improving LLM efficiency, accuracy, and scalability. Release of *GPT-4*, which incorporates more training with human feedback, continuous improvement from real-world data, and more robust quality control.

Table 7.3 Brief History of NLP

While most of the development has been in areas of text recognition and generation, significant innovations have also been made in areas of art, music, video editing, and even computer coding.

NLPs in Generative Art

Generative art is the use of AI tools to enhance or create new artistic and graphic works. In the past, people thought that one defining feature that set humans apart from animals or even from computers is our ability to make and appreciate art. Surveying the artistic genius of Leonardo Da Vinci, Vincent van Gogh, Pablo Picasso, Salvador Dalí, Frida Kahlo, and Jean-Michel Basquiat, it is difficult to imagine a machine producing anything like the masterworks created by human artists. However, we are beginning to witness a revolution in art, led by artificial intelligence! All judgements of quality are left up to the reader, but here are a few tools available now for generative art.

OpenArt: OpenArt (https://openstax.org/r/openart) utilizes neural style transfer techniques to transform photos into artworks inspired by famous artists' styles, or create your own images and animations based on prompts.

DALL-E and Craiyon (formerly DALL-E mini): Developed by OpenAI, Craiyon (https://openstax.org/r/craiyon) generates images from textual descriptions, demonstrating the ability to create diverse and imaginative visual content. Figure 7.14 shows some examples of artistic works created by Craiyon.

Figure 7.14 AI Art Created by Craiyon. (These nine images generated using Craiyon [link to: https://www.craiyon.com/ (https://openstax.org/r/craiyon) from the prompt "A surreal painting of a horse sitting at a bar, drinking a cherry milkshake." Note: Each use of this prompt will produce varying output.)

Magenta Project: Magenta (https://openstax.org/r/magenta) is an open-source research project by Google that explores the intersection of AI and creativity, including music and art generation. It offers various tools and models for generating music and visual art using deep learning techniques, such as neural style transfer and image generation models.

Music is another area of artistic expression that was until recently regarded as solely in the domain of human minds. Nowadays, there are numerous AI tools for processing, analyzing, and creating music.

Jukebox: An OpenAI project, Jukebox (https://openstax.org/r/jukebox) employs machine learning algorithms to automatically compose personalized music tracks for videos and other content.

Lyria model by Google Deepmind: Lyria (https://openstax.org/r/deepmind) is described as the most advanced AI music generation model to date. It excels at generating high-quality music with both instrumentals and vocals, tackling challenges such as maintaining musical continuity across phrases and handling multiple voices and instruments simultaneously.

Amadeus Code: Amadeus Code (https://openstax.org/r/amadeuscode) is an AI-powered music composition platform that uses deep learning algorithms to generate melodies and chord progressions. It allows users to input musical ideas and preferences and generates original compositions based on their input.

Video Editing and Captioning

There are a number of AI-assisted technologies to aid in video editing and captioning of videos.

Descript: Descript (https://openstax.org/r/descript) integrates NLP-based transcription and editing tools to facilitate easy editing of audio and video content through text-based commands and manipulation.

Kapwing: Kapwing (https://openstax.org/r/kapwing) utilizes NLP for automatic video captioning, enabling users to quickly generate accurate captions for their videos.

Simon Says: Simon Says (https://openstax.org/r/simonsaysai) is an AI-powered transcription and translation

platform that uses NLP to transcribe audio and video files in multiple languages. It offers features such as speaker identification, timecode alignment, and captioning, helping video editors streamline the post-production workflow.

Computer Coding

At the time of this writing, it is common knowledge among researchers and programmers that NLPs like ChatGPT can generate code in any common coding language based on user prompts. Moreover, online coding platforms such as Google Colab (which is the platform used throughout this text for Python code examples) offer AI assistance in creating and debugging code. Here are a few more resources that use AI to produce computer code.

GitHub Copilot: Not to be confused with Microsoft Copilot, GitHub Copilot (https://openstax.org/r/copilot) was developed by GitHub in collaboration with OpenAI to be an AI-powered code completion tool that uses NLP to suggest code snippets and provide context-aware code suggestions directly within integrated development environments (IDEs) like Visual Studio Code.

TabNine: TabNine (https://openstax.org/r/tabnine) is an AI-powered code completion tool that uses a combination of deep learning and NLP to provide intelligent code suggestions as developers type. It supports various programming languages and integrates with popular code editors.

Sonar: Sonar (https://openstax.org/r/sonarsource) is an AI-powered static code analysis platform that uses machine learning and NLP techniques to identify potential bugs, security vulnerabilities, and performance issues in code. It provides intelligent suggestions for code improvements, cleaning code, and best practices in programming.

CoNaLa: CoNaLa (https://openstax.org/r/conala) (Code/Natural Language) is a dataset and research project that aims to bridge the gap between natural language descriptions and code snippets. It uses NLP techniques to generate code snippets from natural language descriptions of programming tasks, helping automate code synthesis.

Speech Recognition and Text-to-Speech

NLP models play a crucial role in both speech recognition and text-to-speech (TTS) applications, enabling computers to understand and generate human speech.

In speech recognition, NLP models process spoken language and convert it into text. This involves several steps:

1. Acoustic Analysis: The speech signal is converted into a sequence of feature vectors using deep learning techniques.
2. Language Modeling: Further processing allows the NLP model to analyze and understand the linguistic structure of the spoken words, considering factors like grammar, vocabulary, and context. This helps in recognizing words and phrases accurately, even in noisy or ambiguous environments.
3. Decoding: Once the speech signal has been analyzed acoustically and linguistically, decoding algorithms are used to match the observed features with the most likely sequence of words or phrases.

In text-to-speech (TTS) applications, NLP models perform the reverse process: they convert text into spoken language. This involves the following steps:

1. Text Analysis: Using sophisticated deep learning models, the input text is analyzed and processed to extract linguistic features such as words, sentences, and punctuation.
2. Prosody Modeling: To avoid sounding “like a robot,” NLP models consider linguistic features like stress, intonation, and rhythm to generate natural-sounding speech.
3. Speech Synthesis: The synthesized speech is generated using digital signal processing techniques to produce sound waves that closely resemble human speech. This can involve concatenating prerecorded

speech segments, generating speech from scratch using parametric synthesis, or using neural network–based generative models.

Future Directions for NLP and LLM

Where will natural language processing and large language models be in the future? Given the rapid pace of development over the past few decades, it is hard to predict. Some of the more recent tools such as ChatGPT and Microsoft Copilot are already making a huge impact in the workplace, schools, and elsewhere.

A number of programs have AI embedded in their applications, a feature that is expected to increase as AI technology advances. AI has also been integrated into search engines such as Bing and Microsoft Edge. Both Microsoft and Google have introduced AI into their applications. For example, Copilot will provide enhanced abilities to summarize documents, add animations to slides in PowerPoint, and create stories in Word. It is being marketed as a tool to improve employee efficiency by automating time-consuming tasks such as composing or reading long emails or summarizing datasets in Excel.

In a comparable way, Google is testing incorporating AI into its Workspace applications. Google's tool, called Gemini, is designed to do many of the same things as Microsoft's Copilot. The program uses machine learning to generate emails, draft documents, and summarize data trends, among many other functions. Keep in mind that as AI continues to evolve, governments will seek to regulate its use and application.

For both Microsoft and Google, the goal is to use AI technology to enhance worker productivity. Adding automation to routine, time-consuming tasks can free up employees to focus on more pressing and impactful tasks that will contribute to their companies in a greater way. We can expect to see more AI in these programs and others with time. There will certainly be advances in deep learning models, leading to better image recognition and analysis; more realistic content generation, including art, music, speech, etc.; and more sophisticated interaction with users via natural language models. There are already news channels (https://openstax.org/r/ecHioH8fawE) that are completely AI-generated!

Moreover, NLP applications will have a huge impact on education. Colleges and universities have already been considering what to do about tools such as ChatGPT in the classrooms. Much the same way as pocket calculators revolutionized the mathematics curriculum, pushing educators to steer away from teaching tedious calculations by hand in favor of more conceptual knowledge and problem-solving skills, large language models like ChatGPT will force educators to rethink the purpose of writing papers. Will students in a literature class use an NLP model to summarize the plot, theme, and tropes of Shakespeare's *Hamlet* without the need to even read the play? If so, then educators must find alternative ways to engage their students. Perhaps the assignment will morph into using an NLP to create a contemporary music video based on the mysterious death of Ophelia (a character in *Hamlet*); a thorough understanding of the play would be essential for the student to be able to evaluate whether AI did a good job or not.

AI could also be used to train student teachers. Imagine interacting with an NLP model that simulates a sixth-grader with attention-deficit/hyperactivity disorder (ADHD). The student teacher would gain valuable experience working with different personality types even before entering a real classroom.

We are already seeing AI tools being used as advanced tutors in science, mathematics, computer science, and engineering. For example, Socratic (https://openstax.org/r/socratic) (by Google) is a mobile app that provides step-by-step explanations and solutions to homework problems in various subjects, leveraging powerful AI models. The use of AI in both helping students solve problems and generating educational content (including training future data scientists, who will use AI themselves!) can offer several advantages. Natural language processing models can efficiently generate high-quality content, covering a wide range of topics and providing explanations that are clear and accessible to learners. Think of AI as a *force multiplier*, freeing the human authors and educators to focus on the big picture. However, it's essential to recognize that while NLPs can generate content, they are merely tools created by humans. Therefore, the responsibility lies with educators and content creators to ensure that the generated material is accurate, interesting, up-to-date, and aligned

with the learning objectives of the course or text.

ChatGPT and Ethical Considerations of Using NLP

In the preceding sections, we introduced the powerful natural language processing model ChatGPT. While ethical issues must be considered throughout the entire data science process (and are discussed in depth in Ethics Throughout the Data Science Cycle), NLPs and large language models themselves present some unique ethical concerns.

As noted earlier, the current version of ChatGPT is primarily based on the *transformer model*, a deep learning architecture that incorporates a self-attention mechanism. Whereas simpler models like recurrent neural networks and long short-term memory networks have rudimentary *memory* structures, the transformer model utilizes a very sophisticated internal memory system that can capture long-range and large-scale dependencies in the input data. These models are trained on vast amounts of text data using unsupervised learning techniques, allowing them to generate humanlike text and perform various natural language processing tasks.

The main components of NLP models are as follows:

- Tokenization: The process of breaking down a piece of text into smaller units, such as words or subwords, so that the structure and meaning of the input text can be analyzed.
- Word embeddings: Words are converted into numerical vectors called word embeddings. These vectors represent the meaning of words in a high-dimensional space, allowing a deep understanding of semantic relationships between words.
- Contextual understanding: the meaning of individual words and phrases are analyzed within the larger context of a whole conversation rather than relying on only the immediate surrounding words.
- Feedback loop: As users interact with an LLM and give feedback on its responses, the model can continually learn and improve over time.

Protections for Artists and Their Intellectual Property

Advanced NLP models such as ChatGPT, DALL-E, Lyria, and the many others currently available present both remarkable opportunities and serious ethical issues. The use of these models raises concerns about appropriate protections for human artists and their creations. One major concern is how the models are trained. Essentially, anything that is available on the internet could be used as input to train an NLP model. This includes original text, art, and music that humans have created. Whether the original content is copyrighted or not, an NLP model may have access to it and be trained to mimic the real thing.

With the ability to produce vast amounts of content quickly, there is a huge risk of devaluing the work of human creators and perpetuating issues related to copyright infringement and intellectual property rights. As such, it's imperative to establish clear guidelines and regulations to safeguard the rights of human artists and ensure fair compensation for their work. Additionally, the deployment of ChatGPT and similar models in content creation should be accompanied by transparent disclosure and attribution practices to distinguish between AI-generated content and human-authored works.

The film industry especially has been both transformed and disrupted by generative AI, offering new opportunities for innovative filmmaking and major concerns about how AI usage might lead to infringement of copyright, job displacement, and breaches of data privacy. In the summer of 2023, both the Screen Actors Guild-American Federation of Television and Radio Artists (SAG-AFTRA) and the Writers Guild went on strike against the Alliance of Motion Pictures and Television Producers (AMPTP). The AMPTP includes the five big U.S. movie studios plus Amazon, Netflix, and Apple. The strikes, lasting four and five months, respectively, highlighted concerns about safeguarding union members against the potential threats to writers' livelihoods if studios were to replace writing roles with generative AI (GAI) and protection of actors' and extras' identities and "likenesses" from being digitally recreated without their consent. The landmark agreements negotiated

between unions and studios established what one observer termed “the most progressive AI protections in industry history” (Luna & Draper, 2023).

Disclosure and Attribution

The deployment of ChatGPT and similar AI models in content creation necessitates transparent disclosure and attribution practices to ensure clarity and accountability regarding the origin of generated content. Transparent disclosure involves clearly indicating when content has been generated by an AI model rather than authored by a human. This disclosure helps users understand the nature of the content they are interacting with and manage their expectations accordingly.

Attribution practices involve giving credit to the appropriate sources or creators of content, whether human or AI-generated. In the context of ChatGPT, attribution practices may involve indicating the use of AI assistance in content creation and acknowledging the contributions of the underlying model. This helps maintain transparency and integrity in content creation processes as well as respect for the efforts of human creators.

Currently, disclosure and attribution practices in ChatGPT and similar models vary depending on the specific use case and platform where the model is deployed. Some platforms may include disclaimers or labels indicating that content has been generated by an AI model, while others may not provide explicit disclosure. Additionally, attributing AI-generated content to specific models or algorithms may be challenging, as the content generation process often involves complex interactions between multiple components and datasets.

Moving forward, there is a need for standardized guidelines and best practices regarding disclosure and attribution in AI-generated content. This includes establishing clear standards for labeling AI-generated content, defining appropriate attribution mechanisms, and promoting transparency and accountability in AI-based content creation. More about this topic will be discussed in Ethics Throughout the Data Science Cyle.

Limitations of NLPs

Take a look at the “photo” of a musician playing an upright bass in Figure 7.15. Do you notice anything strange? First of all, the bass seems to be floating in front of the woman (or is she balancing it by some sort of handle sticking out of the neck of the instrument?). There are too few strings (there should be four). Even the machine heads (tuners, on the headstock, or very top, of the instrument) are a bit off, with apparently three of them showing on one side and with missing or malformed keys. Another physiological detail that AI-generated images often don’t depict accurately are hands, with missing or misshapen fingers (such as in the musician’s left hand in Figure 7.15).

Figure 7.15 Image of a Musician Created by OpenArt. While the picture resembles a real photo at first, there are numerous details that do not make sense. (Image generated using OpenArt [link to: https://openart.ai/create (https://openstax.org/r/openart1)] from the prompt "A woman playing upright bass." Note: Each use of this prompt will produce varying output.)

These kinds of problems with detail are very common in AI-generated material. There are various reasons for this, including lack of adequate training data, inherent bias, lack of diversity in training, complexity of the objects in the image, and prioritization of general style and composition over attention to detail by the AI model. This is sometimes called the "fingers problem," as AI has trouble in getting the right number of fingers on a human hand (though, in Figure 7.15, there seems to be the correct number of fingers visible).

Another issue that plagues LLMs is that of *hallucinations*, which are generated responses that have no basis. AI **hallucinations** are essentially made-up responses that seem to answer the prompt but are incorrect, misleading, or lacking proper context. Since everything that an LLM produces is, in a sense, "made up," the LLM cannot easily detect when it is giving a hallucination. Imagine asking a five-year-old how a car runs. She might tell you that there's a loud troll that lives under the hood turning the wheels by hand, drinking the gasoline that her parents continually feed it. In the absence of a true definition, it sounds plausible to the child.

Finally, it should be mentioned that any AI model is only as good as the training data that is fed into it. Despite remarkable advancements in NLP models and LLMs, a major concern is the potential for bias and unfairness due to unforeseen problems in its training data. AI systems, including ChatGPT, can inadvertently perpetuate stereotypes or amplify existing societal biases present in the training data. Moreover, NLP models like ChatGPT may struggle with understanding and generating contextually appropriate responses, leading to instances of misinformation or inappropriate content generation.

There are also concerns about overreliance on LLMs in content generation, potentially leading to a lack of diversity in perspectives or overlooking nuances in the subject matter. In some sense, a fully trained LLM produces an *average*, taking in the wonderful, rare, creative, surprising, and eccentric creations of humans and mixing all that in with the vast amount of the routine, mundane input data to come up with a plain vanilla response.

EXPLORING FURTHER

Bias and Ethics in AI

The effects of bias in deep learning and AI models are critical to understand, especially in important areas like hiring, finance, and law enforcement. Bias in AI can lead to unfair outcomes, perpetuating existing social inequalities. The article "Using Python to Mitigate Bias and Discrimination in Machine Learning Models (https://openstax.org/r/holisticai)" provides a practical guide for addressing these issues. It demonstrates how Python libraries such as `HolisticAI`, `Scikit-Learn`, and `Matplotlib` can be used to identify and mitigate bias in machine learning models. This process involves assessing model performance across different demographic groups and applying techniques to ensure fair treatment and equitable outcomes.

"The Ethical AI Libraries That Are Critical for Every Data Scientist to Know (https://openstax.org/r/hackernoon) discusses various other libraries designed to enhance ethical practices in AI development, such as IBM's `AI Fairness 360`, Google's `What-If Tool`, and Microsoft's `Fairlearn`. Each tool offers unique features to identify, assess, and mitigate biases in datasets and models.

Malicious Uses of AI

User privacy, data security, potential copyright infringement, and devaluation of human-created content are just the tip of the iceberg; hallucinations, bias, and lack of diversity in training data often lead to inaccuracies. Much more concerning than these unintentional inaccuracies are the malicious uses of AI to deceive people.

Deepfakes are products of AI systems that seem realistic and are intended to mislead people. For example, in January 2024, a number of residents of New Hampshire received a robocall (automatic recorded message by phone) that sounded like President Biden discouraging voters from voting in the Democratic presidential primary election. It was later discovered that the recording was created by AI by a person associated with one of Biden's rivals in the primary race.

Deepfake videos have been used to create fake celebrity endorsements, political propaganda, and illicit pictures that look realistic, highlighting the need for legal and regulatory frameworks to address the misuse of AI-generated content.

In order to address concerns about the malicious use of AI, tech companies and organizations now promote the ideal of **responsible AI**, which refers to the ethical and socially conscious development and deployment of artificial intelligence systems. It encompasses a set of principles, practices, and guidelines aimed at ensuring that AI technologies are developed and used in ways that are fair, transparent, accountable, and beneficial to society.

Key aspects of responsible AI include ethical considerations, transparency, accountability, fairness and bias mitigation, data security, societal and environmental impacts, and design principles that are human-centric, topics that we will explore further in Ethics Throughout the Data Science Cyle. The main goal of responsible AI is to foster the development and deployment of AI technologies that align with human values, respect human rights, and contribute to a more equitable and sustainable future.

Datasets

Note: The primary datasets referenced in the chapter code may also be downloaded here (https://openstax.org/r/datach7).

Key Terms

activation for a neuron, the process of sending an output signal after having received appropriate input signals

activation function non-decreasing function *f* that determines whether the neuron activates

artificial intelligence (AI) branch of computer science that aims to create intelligent systems capable of simulating humanlike cognitive abilities, including learning, reasoning, perception, and decision-making

backpropagation an algorithm used to train neural networks by determining how the errors depend on changes in the weights and biases of all the neurons, starting from the output layer and working backward through the layers, recursively updating the parameters based on how the error changes in each layer

bias value *b* that is added to the weighted signal, making the neuron more likely (or less likely, if *b* is negative) to activate on any given input

binary cross entropy loss loss function commonly used in binary classification tasks: $-\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \hat{y}_i - \left(1 - y_i\right)\ln(1 - \hat{y}_i)\right]$

ChatGPT Powerful natural language processing platform, created by OpenAI

connecting weight in a recurrent neural network, a weight that exists on a connection from one neuron to itself or to a cycle of neurons in a feedback loop

convolutional layers layers of a CNN that applies convolution filters to the input data to produce a feature map

convolutional neural network (CNN) class of neural network models developed to process structured, grid-like data, such as images, making use of the mathematical operation of convolution

deep learning training and implementation of neural networks with many layers to learn hierarchical (structured) representations of data

deepfake product of an AI system that seem realistic, created with malicious intent to mislead people

depth number of hidden layers in a neural network

dimension the number of components in a vector

dynamic backpropagation adjustment of parameters (weights and biases) and the underlying structure (neurons, layers, connections, etc.) in the training of a neural network

epoch single round of training of a neural network using the entire training set (or a batch thereof)

exploding gradient problem failure to train an RNN due to instability introduced by having connecting weights at values larger than 1

feature map output of convolutional layers in a CNN, representing the learned features of the input data

feedback loop internal connection from one neuron to itself or among multiple neurons in a cycle

fully connected layers layers of a neural network in which every neuron in one layer is connected to every neuron in the next layer

generative art use of AI tools to enhance or create new artistic works

gradient descent method for locating minimum values of a multivariable function using small steps in the direction of greatest decrease from a given point

hallucinations in the context of NLP, AI-generated responses that have no basis in reality

hidden layers layers between the input and output layers

hinge loss loss function commonly used in binary classification tasks: $-\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\hat{y}_i\right)$

hyperbolic tangent (tanh) common activation function, $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

imbalanced data datasets that contain significantly more data points of one class than another class

input layer neurons that accept the initial input data

large language model (LLM) powerful natural language processing model designed to understand and generate humanlike text based on massive amounts of training data

leaky ReLU common activation function, $\mathrm{LReLU}(x) = \max(cx, x)$, for some small positive parameter c

long short-term memory (LSTM) network type of RNN incorporating *memory cells* that can capture long-term dependencies

loss (or cost) function measure of error between the predicted output and the actual target values for a neural network

margin measure of the separation of data points belonging to different classifications

memory cells internal structures that allow the network to store and access information over long time intervals

multilayer perceptron (MLP) basic paradigm for neural networks having multiple hidden layers

natural language processing (NLP) area of AI concerned with recognizing written or spoken language and generating new language content

neural network structure made up of neurons that takes in input and produces output that classifies the input information

neuron individual decision-making unit of a neural network that takes some number of inputs and produces an output

nonlinear not linear; that is, not of the form $f(x) = mx + b$

output layer neurons that are used to interpret the answer or give classification information

perceptron single-layer neural network using the step function as activation function, designed for binary classification tasks

pooling layers layers of a CNN that reduce the dimensions of data coming from the feature maps produced by the convolutional layers while retaining important information

rectified linear unit (ReLU) common activation function, $\text{ReLU}(x) = \max(0, x)$

recurrent neural network (RNN) neural network that incorporates feedback loops

responsible AI ethical and socially conscious development and deployment of artificial intelligence systems

semantic segmentation process of partitioning a digital image into multiple components by classifying each pixel of an image into a specific category or class

sigmoid function common activation function, $\sigma(x) = \frac{1}{1+e^{-x}}$

softmax activation function that takes a vector of real-number values and yields a vector of values scaled into the interval between 0 and 1, which can be interpreted as discrete probability distribution

softplus common activation function, $f(x) = \ln(1 + e^x)$

sparse categorical cross entropy generalization of binary cross entropy, useful when the target labels are integers

static backpropagation adjustment of parameters (weights and biases) only in the training of a neural network

step function function that returns 0 when input is below a threshold and returns 1 when input is above the threshold

tensor multidimensional array, generalizing the concept of vector

vanishing gradient problem failure to train an RNN due to very slow learning rates caused by having connecting weights at values smaller than 1

vector ordered list of numbers, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$

weight value w that is multiplied to the incoming signal, essentially determining the strength of the connection

Group Project

Project A: Diagnosing Disease

In this project, you will create a neural network that diagnoses patients for cirrhosis (severe scarring of the liver) based on numerical and categorical factors. You will use the dataset cirrhosis.csv (https://openstax.org/r/datach7).

a. The dataset has some missing data ("NA"). Remove all rows with missing data.
b. The input data `x` should consist of all the feature columns except "ID," "N_Days," "Status," and "Drug." Drop those columns.

c. Convert categorical data into numerical data using *one-hot* encoding (see Other Machine Learning Techniques). You may find the Python function `get_dummies()` rather useful.
d. The response variable (output) should be set to Status. Status is either "D" for death of the patient or "C" or "CL" for no recorded death of the patient during the trial. The following code should be used to transform the string data into numerical data: `y = df['Status'].apply(lambda x: 0 if x=='D' else 1)`.
e. You will also need to convert string data to numerical in the input dataframe, `X`. Use the following code to do this:

```
X = X.replace('F',1)
X = X.replace('M',0)
X = X.replace('Y',1)
X = X.replace('N',0)
```

f. Now you're ready to do the train/test split and build your model! Using `TensorFlow`, set the number of units in the hidden layers to 16 and 16 (two hidden layers), and set the number of epochs to 10. This model converges very quickly on the data, so there is no need to run many epochs.
g. Record the final loss and accuracy of the training step as well as the accuracy score of the model on the testing set. How did your model do? Based on the accuracy score, how confident would you be in using this model to diagnose cirrhosis?

Project B: Recognizing Digits

There are many real-world applications that rely on the ability of technology to recognize handwritten digits, including reading forms automatically, postal code recognition in mail sorting, automated grading, and medical record digitization.

a. With the help of `TensorFlow`, build a convolutional neural network (CNN) model in Python and train it to recognize the handwritten digits in the MNIST dataset. This article on machine learning (https://openstax.org/r/machinelearningmastery1) walks you through the process, including detailed discussions of variations and improvements on the model.
b. Try out different learning rates, adding or removing layers and other variations, and report on their effectiveness.
c. As a group, discuss the advantages and disadvantages of the CNN model as compared to other potential models for classification of digits.

Project C: Artistic License

In this project, you will use NLP/LLM models, such as ChatGPT and OpenArt, to create an "original" story with illustrations.

a. As a group, brainstorm themes and characters for a story. Then use appropriate prompts to guide the LLM in creating a coherent narrative, with a clear plot and realistic characters. Explore different genres and themes, such as adventure, fantasy, or educational content. The team should edit and refine the generated text to ensure coherence and appropriateness for their intended target audience.
b. Use AI-based art generation tools to create illustrations for the story. The team will create prompts based on the generated story to guide the art generation process, ensuring that the illustrations align with the narrative and experimenting with different artistic styles and techniques to create a visually appealing storybook. The group should refine and select the best illustrations for inclusion in the final product.
c. For added interest, use AI-based music composition tools to create background music for the storybook. The group will select appropriate music styles based on the story's mood and setting. The music should enhance the overall storytelling experience.

Chapter Review

1. What is the primary role of hidden layers in a neural network?

a. To directly interact with the input data and produce the final output
b. To provide a way for the network to learn and represent complex patterns and relationships within the data
c. To reduce the number of features of the input data
d. To store the final predictions of the model

2. What is a convolutional neural network (CNN), and in which scenarios might it perform better than standard neural networks?
a. A CNN is a type of neural network designed to process sequential data, and it is particularly effective for tasks like language translation and text generation.
b. A CNN is a type of neural network that includes recurrent layers, making it suitable for time series prediction and speech recognition.
c. A CNN is a type of neural network that uses convolutional layers to process grid-like data structures, such as images, and is particularly effective for tasks like image classification, object detection, and recognizing spatial relationships.
d. A CNN is a type of neural network that relies on decision trees, and it is particularly effective for classification tasks involving structured tabular data.

3. Why are speech recognition and text-to-speech algorithms important in everyday life, and what are some examples of their applications?
a. Speech recognition and text-to-speech algorithms are mainly used in scientific research and have limited everyday applications.
b. These algorithms are only relevant in the context of language translation and have no significant impact on everyday life.
c. Speech recognition is primarily used for recording audio, while text-to-speech is used mainly for entertainment purposes, such as audiobooks.
d. These algorithms are important because they enable hands-free interaction with devices, improve accessibility for individuals with disabilities, and enhance user experience in various applications.

4. Which of the following would be considered an unethical use of AI or NLP technology?
a. Developing a chatbot that provides customer service and technical support on a 24/7 basis
b. Creating a natural language processing system that assists visually impaired users in reading text aloud
c. Designing a virtual assistant that collects personal data without explicit user consent and sells it to third-party advertisers
d. Implementing an AI-based language model that translates content across different languages for educational purposes

Critical Thinking

1. Consider a single perceptron model that is being trained to diagnose the flu in patients based on three simple markers. Suppose the next data point in the training set is $\boldsymbol{x} = (0.68, 0.42, 0.99)$, and suppose that the correct classification for this combination of markers is $y = 1$ (flu diagnosed). With current weights $\boldsymbol{w} = (0.7, 0.2, -0.3)$ and bias $b = -0.3$, does the perceptron correctly diagnose the patient as having the flu? If not, use the perceptron learning rule with $h = 0.2$ to update the weights and bias. Is there an improvement in classifying this data point?

2. For each scenario, determine whether a standard neural network, a recurrent neural network (RNN), or a convolutional neural network (CNN) would be most suitable. Explain your choice, detailing the advantages and disadvantages of each type of network for the given scenario.
a. A company wants to analyze customer reviews to determine whether they are positive, negative, or

neutral. The reviews are provided as text data, and the company is interested in understanding the general sentiment of each review.

b. An investment firm is developing a model to predict future stock prices based on historical price data. The firm wants to account for trends, patterns, and time-based dependencies in the data.

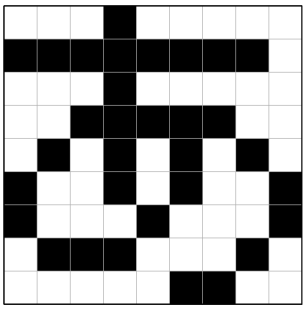
c. A technology company is building a model to classify images into various categories, such as "cat," "dog," and "car." The input data consists of individual images.

d. A company wants to predict whether a customer is likely to cancel their subscription based on a variety of features, such as customer demographics, subscription duration, customer interactions, and usage patterns.

3. The characters in some computer fonts are represented by arrays of color values or grayscale values, called *bitmaps*. See the following 9-by-9 bitmap representing the Japanese hiragana character "a" in which each number indicates one of two color values, 0 for white and 1 for black, for each pixel.

0	0	0	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	1	0	1	0	1	0	1	0
1	0	0	1	0	1	0	0	1
1	0	0	0	1	0	0	0	1
0	1	1	1	0	0	0	1	0
0	0	0	0	0	1	1	0	0

Here is how the character might look when rendered:



Partition the bitmap array into 2-by-2 blocks and use the pooling method of averaging to reduce the size of the bitmap. Note: Since the original array's dimensions are not divisible by 2, you should pad the array by appending a column of zeros to the right and a row of zeros on the bottom. How might the resulting bitmap be interpreted visually?

4. Which loss functions would be most appropriate to use in training a neural network for the following tasks? Why?

a. Predicting the temperature based on day of the year, recent wind speeds, precipitation, and prior temperatures

b. Classifying emails as spam or not spam based on their content

c. Classifying an image into one of several categories, such as animals, vehicles, or plants

d. Diagnosing cancer in patients based on multiple indicators

Quantitative Problems

1.

a. Suppose that a neural network has been trained to classify students as likely to graduate or not likely to graduate based on various input parameters. There is a single output neuron having four inputs,

x_1, x_2, x_3, and one output, y. The output y is used to make the classification. The weights of the three inputs of this neuron are $w_1 = 0.25$, $w_2 = 0.09$, and $w_3 = -0.31$. The bias of the neuron is $b = -0.23$. Find the output y if $(x_1, x_2, x_3) = (0.4, -1.2, 0.6)$ and if the activation function is:

i. ReLU
ii. Leaky ReLU defined by max (0.2x,x)
iii. Sigmoid
iv. Softplus

b. Which of the four activation functions, applied at the output layer, would be best suited for this classification task? Using this activation function, how would the neural network classify the student?

2. The output of a neural network on data points are given in the table.

Data point	**Predicted** $\hat{y}$	**Actual(target)** y
1	$(0.043, 1.169, 1.019)$	$(0, 1, 1)$
2	$(-0.149, 0.927, -0.136)$	$(0, 1, 0)$
3	$(0.96, 0.237, -0.109)$	$(1, 0, 0)$
4	$(0.842, -0.148, -0.218)$	$(1, 0, 0)$
5	$(0.942, 0.838, 1.011)$	$(1, 1, 1)$

Table 7.4

Compute the average loss using the following loss functions.

a. MSE
b. Hinge loss

References

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems *Annals of Eugenics, 7*(2), 179–188; also in (1950). *Contributions to Mathematical Statistics*. John Wiley.

Luna, A., & Draper, D. (2023, December 6). *Hollywood strikes back against generative AI disruption*. Bipartisan Policy Center. https://bipartisanpolicy.org/blog/hollywood-strikes-back-against-generative-ai-disruption/

8 Ethics Throughout the Data Science Cycle

Figure 8.1 Balancing the ethical aspects of the evolving practice of data science requires a careful approach throughout the data science cycle (i.e., during collection, analysis, and use of data, as well as its dissemination). (credit: modification of work "Data Security Breach" by https://www.blogtrepreneur.com, CC BY 2.0)

Chapter Outline

8.1 Ethics in Data Collection
8.2 Ethics in Data Analysis and Modeling
8.3 Ethics in Visualization and Reporting

Introduction

Data science is a rapidly growing field that has revolutionized numerous industries by providing an exceptional amount of data for analysis and interpretation. However, the existence of plentiful amounts of data, along with increased access to it, also raises many ethical considerations. **Ethics in data science** includes the responsible collection, analysis, use, and dissemination of data and the associated results from the use of the data.

Anyone working in data science must focus on making sure that data is used responsibly and ethically. **Data privacy**, or the assurance that individual data is collected, processed, and stored securely with respect for individuals' rights and preferences, is especially important, as the sheer amount of personal data readily available can put individuals at risk of having their personal information accessed and/or exposed. Data scientists must ensure they always adhere to ethical practices and data governance policies and regulations.

Another important aspect of ethics in data science involves fairness, transparency, and accountability. Recognizing and reducing bias, providing adequate and accurate attribution, and clearly documenting and communicating the processes, methods, and algorithms used to generate results are all essential components of ethical data science practice.

This chapter will provide an overview of ethical principles in data science with a focus on data privacy, fairness and bias, and responsible data governance. Understanding and applying these principles will ensure that data science projects are conducted in a way that respects individual rights, promotes fairness, and contributes positively to society.

8.1 Ethics in Data Collection

Learning Outcomes

By the end of this section, you should be able to:

- 8.1.1 Discuss data protection and regulatory compliance considerations in data science.
- 8.1.2 Explain the importance of privacy and informed consent in data science.
- 8.1.3 Identify data security and data sharing benefits and risks.

Data collection is an essential practice in many fields, and those performing the data collection are responsible for adhering to the highest ethical standards so that the best interest of every any party affected by the project is maintained. These standards include respecting individuals' privacy, accurately representing data, and disclosing collection intentions. First and foremost, it is important to ensure that data is used appropriately within the boundaries of the collected data's purpose and that individual **autonomy** is respected—in other words, that individuals maintain control over the decisions regarding the collection and use of their data.

For example, if someone is using a fitness tracking app, they may want to track their daily steps and heart rate but not their location or sleep patterns. They should have the autonomy to choose which data points they want to collect and which ones they do not. Similarly, individuals should also have the autonomy to choose the method of data collection they are most comfortable with. For some, manually inputting data may be preferable, while for others, using a smart device to collect data automatically may be more convenient. In addition, individuals should have control over these decisions—they should be able to manage their own personal data and make informed choices about what they are comfortable sharing. This promotes **transparency** and empowers individuals to fully participate in the data collection process.

To ensure ethical data collection, autonomy of personal data, and data security, it is essential to consult and adhere to the industry standards in data science as developed by organizations such as IADSS Initiative for Analytics and Data Science Standards (https://openstax.org/r/iadss)), DSA (Data Science Association (https://openstax.org/r/datascienceassn)), and ADaSci (Association of Data Scientists (https://openstax.org/r/adasci)). In addition, one must be fully aware of governmental regulations and industry standards.

Regulatory Compliance

Before beginning any data science project, it is important to understand the regulations and specific state and federal laws as well as industry and professional guidelines that apply to the type of data gathering required. Regulations protect the privacy and security of personal and confidential data as well as secure its accuracy, integrity, and completeness. Such regulations may vary from country to country or even from state to state within a country. In the United States, the US Privacy Act of 1974 regulates how federal agencies may collect, use, maintain, and disseminate personal information. This law was later supplemented by the E-Government Act of 2002, extending protections to data held digitally rather than physically. A good summary of these laws and other privacy regulations can be found at the GAO's Protecting Personal Privacy (https://openstax.org/r/gao) site.

Industry and Global Standards

Data privacy compliance requirements vary depending on the industry and the standards or laws that govern the organization's operations. Failure to comply with these requirements can result in penalties and legal consequences for the organization, so data scientists in these industries need to be aware of and adhere to these regulations to avoid any potential repercussions.

Data Security Regulations are based on data protection laws, such as the General Data Protection Regulation in the European Union (https://openstax.org/r/gdpr) (GDPR) and the 2018 California Consumer Privacy Act (https://openstax.org/r/privacy) (CCPA) in the United States, with its amendment, CPRA (2023). These regulations demand that organizations collect, process, and accumulate confidential data securely and

transparently. Figure 8.2 and Figure 8.3 summarize the GDPR and CCPA/CPRA principles. This GDPR Checklist for Data Controllers (https://openstax.org/r/gdpr1) on the EU website presents the principles in an actionable format for organizations.

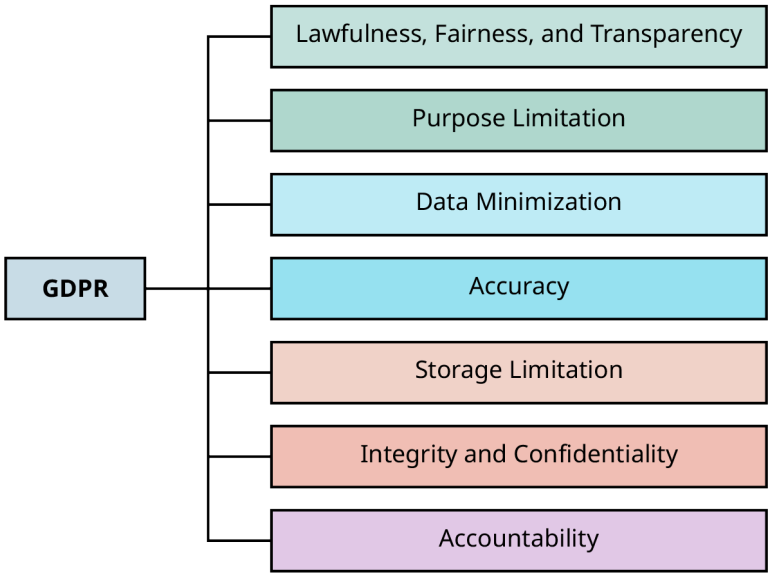


Figure 8.2 GDPR Principles

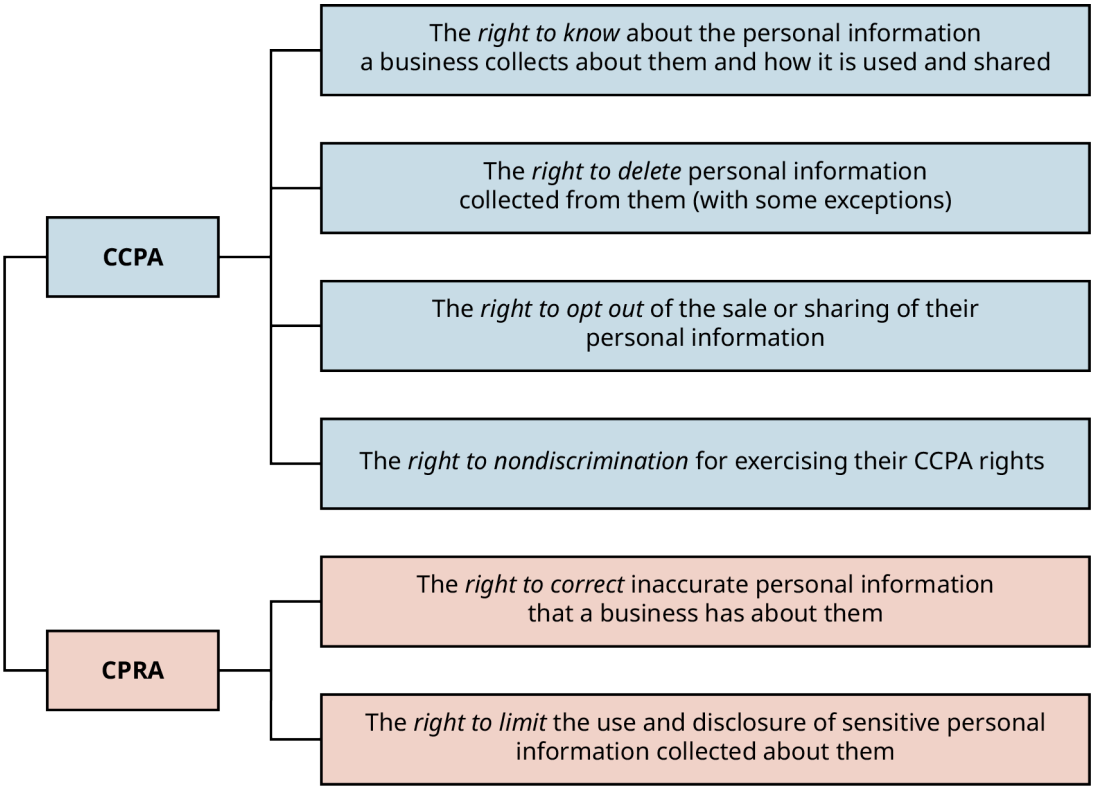


Figure 8.3 CCPA/CPRA Principles

Some countries have laws regarding **data sovereignty** that require data collected from their citizens to be stored and processed within their borders, making it important to understand where data is being collected and stored. Many countries have also enacted laws and regulations involving **data retention**, or how long personal data may be stored (e.g., the Sarbanes-Oxley Act [SOX] in the United States or the Data Protection Act [DPA] in the United Kingdom); **data breach** notification, or what to do when data is stolen by a malicious third party (Gramm-Leach-Bliley Act [GLBA] in the United States or the Personal Information Protection and

Electronic Documents Act [PIPEDA] in Canada); and many other issues of data privacy, security, and integrity.

Some regulations apply only to certain types of data or data related to certain industries. For example, in the United States, the **Health Insurance Portability and Accountability Act (HIPAA)** requires the safeguarding of sensitive information related to patient health. Financial institutions in the United States must adhere to the regulations set forth by the Financial Industry Regulatory Authority (FINRA) and the Securities and Exchange Commission (SEC), both of which regulate the use of sensitive financial information. The **Family Educational Rights and Privacy Act (FERPA)** provides protections for student educational records and defines certain rights for parents regarding their children's records.

Security and Privacy

Adhering to regulatory compliance standards from the beginning stages of the project ensures that all relevant data is gathered and preserved properly, abiding by legal and regulatory requirements. **Data security** measures consist of all the steps taken to protect digital information from unauthorized access, disclosure, alteration, or destruction, ensuring confidentiality, integrity, and availability. It is the responsibility of the data scientist to ensure that all appropriate data security measures are being taken.

Based on the project's purposes, data may need to be either anonymized or pseudonymized. **Anonymization** is the act of removing personal identifying information from datasets and other forms of data to make sensitive information usable for analysis without the risk of exposing personal information. This can involve techniques such as removing or encrypting identifying information. **Pseudonymization** is the act of replacing sensitive information in a dataset with artificial identifiers or codes while still maintaining its usefulness for analysis. These techniques may be required to comply with data protection laws.

To further comply with regulations, some data components may need to be isolated or removed from datasets; for example, industry-specific information may need to be undisclosable from other datasets, especially in the medical or financial domain. Ultimately, adhering to applicable regulatory provisions throughout the data science cycle ensures that all relevant information and data are collected and stored in a way that meets its specific needs without violating any laws or regulations.

The importance of adhering to regulatory compliance cannot be overstated when it comes to ethical business practices, especially when dealing with sensitive customer data such as personal health information and financial records. Firms are obligated to ensure all client information is secure and treated with the utmost care by following all laws and regulations related to data security and privacy. These policies typically include a written statement of data security and privacy requirements, clear definitions for handling customer data, a list of approved and unauthorized uses of data, and detailed procedures for handling, storing, and disposing of customer data. Companies should demonstrate a commitment to ethical practices and ensure customers feel safe by providing sensitive information to the business. These practices assist in building trust and strengthening the relationship between the business and its customers. In addition, following all applicable regulations and laws enables the protection of businesses from potential legal action from customers, regulators, or other third parties.

Regulatory Compliance Teams

Regulatory compliance is commonly controlled by legal and compliance teams or by external consultants for distinct industries. Compliance responsibilities are also often shared across departments such as finance, operations, sales, marketing, and IT. Firms ensuring successful compliance with laws can differ greatly depending on the industry and the size and complexity of the organization. In general, corporations develop procedures and policies designed to ensure that adherence conditions are met. These processes and policies may include frequent training, logging, and creating a system of checks and balances to ensure compliance. They may also include more refined measures, such as periodic reviews of business operations and processes or risk analysis and simulations.

Ultimately, an organization's regulatory compliance officer is responsible for ensuring that the data science project follows all relevant regulatory needs. A **regulatory compliance officer (RCO)** is a trained individual responsible for confirming that a company or organization follows the laws, regulations, and policies that rule its functions to avoid legal and financial risks. The RCO is entrusted with the responsibility of identifying potential compliance issues, devising and implementing effective policies and procedures to address them, and providing comprehensive training and guidance to employees. Additionally, they conduct thorough investigations to assess and monitor compliance levels and promptly report any instances of noncompliance to management. To that end, the RCO will typically use a combination of regulatory aids, internal and external assessments, and business process mapping to assess the project's compliance with applicable laws and regulations. If any deficiency in data privacy or security is found, the RCO will work with the project team to develop a plan for compliance.

EXAMPLE 8.1

Problem

A retail company, Gadget Galaxy, is launching a new customer study project to analyze customer purchasing behavior. The company has encountered several mistakes in its data collection process that violate regulatory compliance standards. Which of these action plans should this company follow if it is found to have violated regulatory compliance?

a. Gadget Galaxy can outsource its data collection process to a third-party company specializing in regulatory compliance.
b. The mistake made by Gadget Galaxy during its data collection process is not implementing any policies or procedures to protect the privacy and security of customer data. Therefore, Gadget Galaxy can continue with its project without implementing any policies or procedures as it is a small business and is not required to comply with regulations.
c. The project team has conducted simple training on regulatory compliance for handling customer data. Therefore, the project team can do their own research and create policies and procedures for data privacy and security.
d. Gadget Galaxy must halt the project and seek guidance from a regulatory compliance expert to develop appropriate policies and procedures.

Solution

The correct choice is D. Gadget Galaxy must halt the project and seek guidance from a regulatory compliance expert to develop appropriate policies and procedures. The project team should undergo training on regulatory compliance to ensure team members understand and observe these policies and procedures. An external review or risk analysis should also be conducted to identify any potential compliance gaps and address them accordingly.

Privacy and Informed Consent

Several important concepts are embedded in the regulations governing data science collection. These include transparency through informed consent, allowing individuals to review and update any personal information. They also rely on safeguarding secure storage and access protocols and regularly monitoring and revising any associated policies and procedures as necessary. Based on ethical principles of study and as a form of legal protection, informed consent needs to be obtained from project participants. **Informed consent** is a process of obtaining permission from a research subject indicating that they understand the scope of collecting data as well as the associated potential risks, benefits, and consequences, including how the data will be used and how it will be stored.

Informed consent consists of *disclosure*, or providing full and clear information about the activity, study, survey, or data collection process; *understanding*, or ensuring that the participant is able to comprehend the information; *voluntariness*, or absence of any coercion, pressure, or undue influence; and finally *consent*, or agreeing to or approving of the process.

For example, a data scientist, Sarah, is conducting research for a company that collects user data through a mobile app. The company requires informed consent from the app users before collecting any data. Sarah decides to conduct a survey to obtain consent from the users. The research aims to understand user behavior and usage patterns in order to improve the app experience. Informed consent is required to ensure transparency and compliance with regulations. By constructing an easy-to-read survey, Sarah is disclosing information and ensuring understanding. Participants take the survey voluntarily, and the results of the survey will indicate consent or non-consent.

COOKIES

After you have searched for a product on the internet, do you notice that you suddenly start receiving advertisements for that product or that a store website now displays that product on its front page? You can thank cookies for that. **Cookies** are small data files that are deposited on users' hard disks from websites they visit. They keep track of your browsing and search history, collecting information about your potential interests to tailor advertisements and product placement on websites. Cookies can be either blocked or accepted depending on a company's privacy policy. The potential risk of cookies is that they can store information about the user, user preferences, and user browsing habits. That said, they can also save time by storing users' log-in information and browsing preferences, allowing internet pages to load faster than if you had loaded them the first time. Regardless of convenience, it is a good idea to clear cookies from time to time and to restrict cookies on certain sites depending on your own preferences. Does the use of cookies by online websites violate any of the principles of informed consent? What about websites that will not load unless the user agrees to their cookie policy?

Confidentiality refers to the safeguarding of privacy and security of data by controlling access to it. The danger is that **personally identifiable information (PII)**, which is information that directly and unambiguously identifies an individual, may end up in the wrong hands. **Anonymous data** is data that has been stripped of personally identifiable information (or never contained such information in the first place) so that it cannot be linked back to any individual, even if it were improperly accessed. Confidentiality plays an important role in the informed consent process, as individuals providing informed consent must be assured that their data will be kept private. This means that any information disclosed during the process of obtaining informed consent should not be shared with unauthorized individuals or organizations. Furthermore, the right to confidentiality should be ongoing, and confidential data should be disposed of properly after the study has been concluded.

An example of a violation of the privacy of an individual or business in the data collection stage may occur when data scientists collect data without pursuing informed consent. Other examples of possible privacy violations during the data collection stage include collecting sensitive information such as health data without explicit consent (as per HIPAA introduced earlier in the chapter), data from minors without their parent's approval, and data from an individual or group that may be vulnerable or marginalized without proper notification or protection measures in place. Those who work in higher education need to be aware of FERPA (Family Educational Rights and Privacy Act), which restricts the sharing or obtaining of college students' academic information without prior consent, even if the requesting party is the student's parent or guardian.

EXAMPLE 8.2

Problem

A group of data scientists is contracted by Gadget Galaxy to conduct a study on consumer buying patterns. They begin by collecting private information from respective participants. The team has implemented an informed consent process to ensure ethical and transparent data collection. This process includes a consent form outlining the study's scope, potential risks and benefits, data handling and storage, and the ability for participants to review and update their personal information. Participants can review the terms at any point during the study, provide ongoing consent, and decide if their data can be transferred to a third party.

Which of the following is NOT a key aspect of the informed consent process described?

a. The consent form outlines the study's scope and potential risks and benefits.
b. Participants can review and update their personal information.
c. Participants' consent is limited to a single point in time.
d. Participants have the option to give or withhold their consent for data transfer to a third party.

Solution

The correct choice is D. This is not an aspect of informed consent. The informed consent process described emphasizes that participants' consent should be ongoing and not limited to a single point in time. All other options are key aspects of the informed consent process.

Data Security

Strong data security policies and protocols are critical parts of any organization in order to mitigate the risk of data breaches throughout the development process. In addition, institutions should have a strategy ready for responding to data breaches in an effective and timely manner. It is important for organizations to ensure that any data they collect or use has been obtained from secure and verified sources and that any data that is shared is done so securely.

Data breaches can have far-reaching consequences, from financial losses to diminished public trust in organizations. As an example, consider the 2017 Equifax data breach. Equifax, one of the three major credit reporting agencies in the United States, experienced a massive data breach in 2017 (https://openstax.org/r/oversight1). This breach exposed the personal information of over 145 million Americans (near half the total U.S. population) along with over 15 million British citizens and a smaller number of people from other countries, making it one of the largest and most severe data breaches in history. The data that was compromised included names, Social Security numbers, birth dates, addresses, driver's license numbers, and for a small percentage of individuals, credit card information. The exposed information put millions of people at risk of identity theft and fraud. Criminals could use the stolen data to open new credit accounts, file fraudulent tax returns, and commit other forms of identity fraud. Victims of the breach had to spend significant time and money to protect their identities, such as by freezing their credit reports, monitoring their credit, and dealing with fraudulent charges. This case highlights the importance of data security in our increasingly digital world, where large-scale data breaches can have widespread and severe consequences for individuals and organizations alike.

If collected data is particularly sensitive or confidential, then it may require **encryption**, which is the process of converting data into code that would be very difficult or impossible for a third party to decipher. There are many encryption algorithms, making use of sophisticated mathematical algorithms, that are routinely used in industries such as finance, health care, education, and government agencies; however, the details of encryption fall outside the scope of this text.

EXPLORING FURTHER

Cryptocurrencies

Bitcoin and other cryptocurrencies use encryption to secure transactions and maintain the integrity of the blockchain, the decentralized ledger that records all transactions. Encryption methods, such as public-private key cryptography, ensure that only the rightful owners can access and transfer their digital assets, creating a secure and verifiable system for storing value in a purely digital form. This cryptographic security underpins the trust and value associated with cryptocurrencies and provides a way to "store value" in the otherwise intangible medium of pure data. For more about bitcoin and other cryptocurrencies, check out Riverlearn's "How Bitcoin Uses Cryptography" (https://openstax.org/r/river) and Kraken's "Beginner's Guide to Cryptography" (https://openstax.org/r/kraken).

Intellectual Property

Another form of data protection that typically does not require encryption is that of legal protection of intellectual property. **Intellectual property**, broadly defined as original artistic works, trademarks and trade secrets, patents, and other creative output, is often protected from copying, modification, and unauthorized use by specific laws. For example, in the United States, **copyright** protects original creative works for the life of the creator plus 70 years. Any protected data under intellectual property rights can be accessed legitimately, if proper permission from the holder of the copyright is obtained and proper attribution given. More about copyright and attribution of data will be discussed in Ethics in Visualization and Reporting.

Some kinds of data are publicly available. Even when this is the case, data scientists need to be aware of the nature of the data to ensure it is used appropriately. For example, if publicly available data contains a person's medical information or financial information, this would still pose a data security risk. Organizations should always ensure that any data they make publicly available is secure and protected by their data security policies and protocols. If data is accidentally breached during the project development process (thereby inadvertently making it *public* data), this could have serious and damaging consequences. For example, in 2019, two datasets from Facebook apps were breached and exposed to the public internet, revealing phone numbers, account names, and Facebook IDs of over 500 million users. Now that the data has been released, there is no way to ensure its security going forward.

Security for Quantitative and Qualitative Data

The collection of q*uantitative* and *qualitative* data (defined in Data and Datasets) have different ethical data security conditions. Securing quantitative data typically involves encryption and anonymization among other techniques. It is essential to maintain the integrity and accuracy of quantitative data, as errors or tampering can lead to incorrect conclusions. Qualitative data collection may also be secured with encryption and anonymization, but there is often a greater emphasis on creating and maintaining trust between the data scientist and respondent, as qualitative data can be context-rich, containing identifiable information, confidential details, and personal data, even when names are removed.

Finally, training and awareness of team members who collect, use, and analyze sensitive data are crucial for ensuring data security. Anyone who works with data must be in compliance with legal and ethical standards and help to maintain the trust of individuals whose data is being handled. In a significant number of the cases of data breaches, the team members themselves were not thoroughly vetted or trained. For more on how businesses may be better prepared to handle data and respond to data breaches, see the Federal Trade Commission's Data Breach Response: A Guide for Business (https://openstax.org/r/ftc).

EXAMPLE 8.3

Problem

Consider a data science team working on a revolutionary data science project that would have a huge impact on the health care industry. The project involves gathering and analyzing sensitive medical information from patients, including full names, addresses, phone numbers, and medical histories. The team recognizes the critical importance of maintaining data security to ensure the success and ethical integrity of their work. One day, a request for access to the data is received from a researcher from an agency outside of the team who has heard about their work. What steps should be taken to ensure data security?

Solution

This data contains personally identifiable information (PII). Moreover, the team needs to ensure that the project complies with HIPAA regulations because there are medical records involved. The following measures should be taken at the beginning of the project and throughout the collection stage.

1. Review access control policies: The first step in granting access to any sensitive data is to review the access control policies in place. This includes understanding who has access to the data, what level of access they have, and how this access is granted. This will help determine if the researcher should be granted permission and what level of access they should have.
2. Authenticate the request: Before granting access, the team should verify the authenticity of the request. This can be done by confirming the identity of the requester and their authority to access the data. The researcher must provide proof of their affiliation with a legitimate organization and the purpose of their request.
3. Assess need-to-know: The team should evaluate whether the researcher has a legitimate need to know the data. If the data is not relevant to their research or can be obtained through alternative means, access may not be granted. This step ensures that only authorized parties have access to the sensitive data.
4. Obtain consent from data owner: As the data pertains to confidential medical information of patients, it is important to obtain consent from the data owner before granting access to a third party. This can be done through a formal process where the data owner signs off on granting access to the researcher.
5. Use secure methods for data transfer: If access is granted, the team must ensure that the data is transferred using secure methods to protect it from any potential breaches. This can include using encryption and secure file-sharing platforms.
6. Review for due diligence: Once access is granted, the team should continue to monitor and review the usage of the data to ensure that the researcher is only accessing the data for approved purposes. Any breaches or misuse of the data should be reported and addressed immediately.
7. Revoke access when necessary: Access to sensitive data should only be granted for a specified period of time and should be revoked when it is no longer needed. If the researcher's authorization or employment changes, their access should be revoked immediately to prevent any potential data breaches.

In general, data scientists should start by carefully considering the demand for data collection within the data science task. They should ensure that the data being collected is necessary and relevant to the project. Next, they should understand the types and quantity of data being collected to determine the level of security measures needed for different types of data.

EXAMPLE 8.4

Problem

A data scientist is leading a project for a construction company that involves extensive data collection. The team carefully assesses the data requirements and implements measures to encrypt the stored data during transmission to safeguard against potential breaches. As the project progresses, the team becomes aware of the importance of obtaining appropriate authorization from copyright holders for any data protected by intellectual property laws. The team prioritizes ethical responsibility by properly attributing all data used and seeking permission from third parties when necessary. However, despite proactive measures, the team acknowledges the ongoing risk of data breaches during the development process. What should the data scientists do if they encounter a data security breach during the development process?

Solution

A possible data security breach can only be handled if the team already has a response plan in place. Having a response plan also shows that data scientists are prepared to handle any potential issues that may arise, can minimize the damage caused by a breach, and can prevent further breaches from occurring. The steps to be taken should include the following:

1. Identify and contain the breach: The first step is to identify the type and scope of the breach. This may involve investigating any unauthorized access, stolen or lost devices, or other security incidents. Once identified, the breach must be contained to prevent further exposure of sensitive data.
2. Assess the damage: Once the breach is contained, the team must assess the extent of the damage caused. This may involve determining the type and amount of data that was compromised as well as any potential impact on affected individuals.
3. Notify authorities: Depending on the nature of the breach, it may be necessary to notify authorities such as law enforcement or regulatory agencies. This is especially important for breaches involving personally identifiable information (PII) or sensitive data.
4. Notify affected individuals: The team must also inform any individuals whose data was compromised in the breach. This notification should include details of the breach, the type of data exposed, and any steps the individual can take to protect themselves.
5. Secure affected accounts: If the breach involves compromised user accounts, those accounts must be secured immediately. This may involve resetting passwords or disabling accounts.
6. Review and update security protocols: The team should conduct a thorough review of their current security protocols and make any necessary updates or improvements to prevent future breaches.
7. Communicate with all project participants: It is important to keep customers, partners, and employees, informed about the breach and the steps being taken to address it. This will help maintain trust and mitigate any potential damage to the organization's reputation.
8. Monitor for further breaches: The team should continue to monitor for any further breaches or suspicious activity to prevent any additional data exposure.
9. Conduct a post-breach review: After the data breach has been resolved, the team should conduct a post-breach review to identify any weaknesses or vulnerabilities in their security protocols and take steps to address them.
10. Provide support for affected individuals: The team should also provide support for affected individuals, such as offering credit monitoring services or assistance with any financial or identity theft issues that may arise from the breach.

Data Sharing

Data sharing is the process of allowing access to or transferring data from one entity (individual, organization,

or system) to another. It is done for a variety of reasons, including collaboration, research, providing services, improved forecasting, and jumpstarting innovation. Data sharing can be carried out in a variety of ways and may involve different types of data. Organizations and individuals may use public datasets, open data standards, or data pools to share data. But most importantly, they must establish **data governance protocols**—a set of rules, policies, and procedures that enable precise control over data access while ensuring that it is safeguarded. For data sharing to be successful, organizations and individuals are required to establish strong infrastructure management and protocols. It is crucial to consider all perspectives of the project participants and how they will operate the data. Consider a policy that outlines the steps for data classification. The policy defines what types of data are sensitive or confidential and how they should be handled. It also specifies who is responsible for managing and accessing different types of data and what security measures need to be in place to protect it.

What and How Much Should Be Shared?

The scope of the data science project, along with associated requirements, will determine the type and extent of data that can be shared. This data may encompass quantitative, qualitative, tabular, text-based, image-based, and audio-based formats, among others, each of which may require unique methods of sharing. For example, tools like Tableau (https://openstax.org/r/tableau) or Power BI (https://openstax.org/r/microsoft) can be used for sharing tabular data and visualizations, while GitHub (https://openstax.org/r/github) has traditionally been useful for sharing programming code, datasets, and related materials (though today, anything can be shared on GitHub). As introduced in Python Basics for Data Science and discussed through this book, Google Colab (https://openstax.org/r/colab), which is connected to Google Drive, allows users to collaborate remotely on coding tasks. Files may be hosted on Google Drive or Microsoft OneDrive, allowing lead data scientists to set different levels of permissions relevant to each team member or outside entity.

Those directly involved in the project, including team members, customers, researchers, and the project's eventual audience, should be granted access to the data in accordance with the intended purpose. If appropriate, and with no privacy-related implications, the data may be disseminated to the broader public through platforms like Kaggle (https://openstax.org/r/kaggle), which is designed for open data sharing in a research context. To protect data privacy and integrity when sharing information with outside parties, ethical policies should be implemented, and appropriate security measures should be put in place. Clear guidelines and procedures must also be provided to ensure that the data is used correctly and only for its intended purpose.

Who Should Have Access to Data?

Access to data in a data science project should be given to individuals who are directly involved in the project and have a legitimate need for the data. This may include the following:

1. **Data scientists and analysts.** These are the core team members who are responsible for analyzing and interpreting the data to draw insights and make decisions.
2. **Data engineers.** These individuals are responsible for managing and maintaining the data infrastructure and ensuring that the data is accessible and stored securely.
3. **Project managers.** They need access to the data to monitor the progress of the project, make informed decisions, and allocate resources effectively.
4. **Data owners.** These are the individuals or teams responsible for collecting, processing, and storing the data. They have a deep understanding of the data and can provide valuable insights.
5. **Interested parties.** These are individuals who have a business need for the data and will use the insights to make decisions and drive business growth.
6. **Legal and compliance teams.** They ensure that the data is accessed and used in compliance with laws, regulations, and company policies.
7. **Data privacy specialists.** They ensure that sensitive data is appropriately handled and protected to maintain the privacy of individuals.

8. **External partners or clients.** In some cases, access to the data may be necessary for external partners or clients who are involved in the project or have a business need for the data.

It is important to have measures in place to control and monitor data access, such as role-based access controls, to ensure that only authorized individuals have access to the data and that it is used appropriately. Data usage agreements and confidentiality agreements may also be necessary to protect the data and the interests of all involved parties.

EXAMPLE 8.5

Problem

John is working as a data analyst for a large retail company. He has been assigned to a data science project to improve forecasting and increase sales. His team has been collecting various types of data for the project, including sales data, customer demographics, and market trends. As they delve further into the project, they realize that sharing the data with other departments and collaborating with external organizations could greatly benefit their research efforts. However, before sharing the data, they need to consider certain factors. John and his team discuss the different ways they can share the data. What is the most important factor that needs to be in place for secure data sharing? What tool(s) would be most effective in sharing the data in this situation?

Solution

In their discussions, data governance protocols stand out as the most important factor in guaranteeing that the data is shared and used responsibly and ethically. This includes setting guidelines and procedures for data usage, implementing security measures to protect data privacy and integrity, and providing access only to authorized individuals or organizations. For example, data governance policy might dictate that all personally identifiable information (PII) be anonymized or excluded when sharing with entities outside of the company. Since the data being shared is a mix of qualitative and quantitative data, an online file sharing app such as Google Drive or Microsoft OneDrive is most appropriate. Permissions should be granted to various team members based on their need to know.

8.2 Ethics in Data Analysis and Modeling

Learning Outcomes

By the end of this section, you should be able to:

- 8.2.1 Define bias and fairness in the context of data science and machine learning.
- 8.2.2 Identify sensitive data elements and implement data anonymization techniques.
- 8.2.3 Apply data validation techniques, such as cross-validation and outlier detection.

In today's technology, industries produce massive amounts of data every day, and with the improvement in digital resources, analyzing and modeling the delivered data has evolved into an essential part of decision-making in many fields, such as business, health care, education, and government. However, with the control and influence that data analysis and modeling hold, it is crucial to consider the ethical implications of this practice.

Data analysis, defined in Data Science in Practice, is the process of examining, organizing, and transforming data to gather knowledge and make well-informed decisions. *Modeling*, on the other hand, as defined in Statistical Inference and Confidence Intervals and applied in Statistical Inference and Confidence Intervals and Introduction to Time Series Analysis, involves employing mathematical and statistical methods to replicate real-life situations and anticipate outcomes. Both processes rely heavily on data, and the accuracy of the results depends on the quality of the data used. Ethical considerations arise when using data that has

systematic bias (intentional or unintentional) or when sensitive or private data is used in the analysis and modeling process, thereby compromising the data. This section will discuss these issues along with some tools that data scientists can use to help mitigate them, including anonymization, data validation, and outlier detection.

Bias and Fairness

Data scientists need to be aware of potential sources of bias and strive for fairness. (See What Is Machine Learning? for a formal definition of bias.) Bias, whether intentional or unintentional, causes unethical outcomes and may even lead to legal concerns. It is important to proactively address any potential concerns before concluding the project and posting the results. Data science teams must plan and build models addressing all possible outcomes with fairness and equity at the forefront of their minds to ensure the best possible results.

Biases in data can lead to biased analysis and modeling, ultimately resulting in unreasonable decisions. For example, if a company's dataset only represents a specific demographic, the insights and predictions produced will only apply to that demographic. This could lead not only to exclusion and discrimination toward marginalized groups, but also to inaccurate modeling when given data from those groups. It is important to regularly check for bias in the data and address it appropriately.

When developing a model, it is important to estimate how the model might introduce bias or unfairness into its decision-making process to avoid its motivations. **Fairness** in data science and machine learning is described as the absence of bias in the models and algorithms used to process data. To ensure models are unbiased and fair, developers need to evaluate a variety of characteristics, including data quality and representation, how to break data into training and test groups, how models are trained and validated, and how the forecasted results are monitored and adjusted. In practice, data bias may be due to poor sampling techniques, small datasets, or differences in measurement protocols among different data gatherers, but it could also be due to intentional practices, such as “cooking the books” to make financial data look more favorable for a company, data fabrication or falsification, “cherry-picking” (choosing only data that supports a predefined conclusion and ignoring the rest), and model manipulation to achieve a desired outcome. Motivations behind these practices include financial gain, prestige, and intent to discriminate against other demographic groups.

When data is biased, the resulting models and algorithms may lead to misinformed decisions, inaccurate predictions, or unfair outcomes. A real-world example of how bias in data science methods or data collection can ruin an analysis is seen in the case of the COMPAS algorithm used in the U.S. criminal justice system. COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is an algorithm designed to predict the likelihood of a defendant reoffending (Berk et al., 2021). However, an investigation in 2016 found that the algorithm was biased against Black defendants. The analysis revealed that the algorithm incorrectly flagged Black defendants as high risk at nearly twice the rate as White defendants. Conversely, White defendants were more likely to be incorrectly flagged as low risk compared to Black defendants. This bias in the data collection and algorithm led to unjust sentencing and parole decisions, disproportionately affecting Black individuals and undermining the fairness and accuracy of the judicial process. This case underscores the critical need for rigorous evaluation and mitigation of bias in data science applications to ensure fair and equitable outcomes. An interesting use of AI using the open-source Python library FairLens (https://openstax.org/r/fairlens) to further analyze this dataset can be found at this OECD policy website (https://openstax.org/r/oecd), though any such application must be viewed with caution.

The data scientist’s primary goal is to construct models that can accurately and fairly depict the population of interest by measuring and mitigating potential bias as much as they can in the datasets used to prepare the models and implementing fairness into the algorithms used to analyze the data. This includes objectively testing the algorithm design, using standard protocols for data collection, ensuring proper handling of data, and making sure that any decisions made in the modeling process are based on improving the model, not

manipulating the model to achieve a desired outcome. The use of available training datasets (see Decision-Making Using Machine Learning Basics) is crucial in detecting discrepancies or inequalities that could lead to unfair results. Models and algorithms should be tested frequently for potential and any expected bias to ensure continuous fairness throughout the model's serviceability.

Factors That Might Lead to Bias

The following factors might cause bias in the process of conducting data science projects:

1. **Outliers.** An outlier might cause an algorithm to drastically change its outcomes due to its abnormality compared to the rest of the data.
2. **Unrepresentative samples.** If the data employed to train an algorithm lacks significant representations of a particular group, then that algorithm will not be qualified to properly serve that group.
3. **Imbalance of classes.** If the data factors are weighted disproportionately to one class group, then the algorithm may be biased toward that particular group.
4. **Limited features.** If there are not enough features to represent the data accurately or completely, the algorithm will likely have an incomplete understanding of the data, leading to inaccurate outcomes or biased decisions.
5. **Overfitting.** Overfitted models are trained on specific data models that cause them to be unable to generalize the data that they have not seen before, causing bias.
6. **Data quality.** Poorly formatted data, inaccurately labeled data, or misplaced data can cause an algorithm to base its decisions on irrelevant or invalid information, resulting in a bias.

Factors That Generally Lead to Fairness

The factors that yield fairness throughout a data science project process include the following:

1. **Representative data.** The data used to train and test models should represent the entire population it seeks to address to ensure fairness.
2. **Feature selection.** Careful consideration of the features utilized by a model will ensure that aspects unrelated to the outcome, such as gender or race are not used for prediction.
3. **Interpretability.** Understanding how outcomes are related to the features of the data can help determine if bias is present and lead to corrections. This may be difficult if the algorithm is very complex, especially when the model involves neural networks and AI. **Explainable AI (XAI)**—a set of processes, methodologies, and techniques designed to make artificial intelligence (AI) models, particularly complex ones like deep learning models, more understandable and interpretable to humans—facilitates interpretability in this case.
4. **Human oversight.** Whenever practical, humans should observe and evaluate how the model is performing. If the dataset is very large, then samples can be pulled and examined for consistency.
5. **Metrics.** Certain measurements or tests can be performed on the data and the model to help ensure that expectations are met and any bias is identified and handled.

EXAMPLE 8.6

Problem

Consider a loan approval system that uses an algorithm to determine an individual's creditworthiness based on their personal and financial information. The algorithm is trained to analyze various attributes such as income, credit score, employment history, and debt-to-income ratio to make a decision on whether or not to approve a loan for each individual based on their likelihood of paying it back. Two datasets are available for training. Which one would be most appropriate in reducing potential bias in the model?

a. Dataset 1, containing information from 5,000 college graduates

b. Dataset 2, containing information provided by the first 200 people who passed by the bank during the previous week
c. Dataset 3, containing information from 3,500 individuals who were selected from different socioeconomic backgrounds, genders, ages, and demographic groups

Solution

The correct answer is C. The most appropriate training set for this system would be Dataset 3 (C), as it includes a broadly representative range of individuals. While Dataset 2 (B) may also represent a diverse range of individuals, the dataset is likely too small to be of much use. Dataset 1 (A) is not appropriate, as the system would then be biased to perform well only on college graduates to the exclusion of other groups of individuals.

Potential Misuse of Data Analysis and Modeling

The potential for misuse of data analysis and modeling is another ethical concern. If data is misapplied or used inaccurately for something other than the project purposes, the predictions produced through data analysis and modeling may manipulate or hurt individuals or groups.

Consider a health care startup that develops a predictive model to identify patients at high risk of hospitalization due to chronic illnesses. To create the model, the company collects a wide range of sensitive patient data, including medical history, medication records, and biometric information. The company fails to implement robust data protection measures, leading to a breach in which hackers access the patient data. The sensitive nature of this information makes its exposure highly concerning, potentially leading to identity theft or discrimination against affected individuals. This example highlights the risks associated with using private or sensitive data in analysis and modeling, emphasizing the importance of strong data security practices and compliance with data protection regulations.

As mentioned earlier in this section, it is essential to have ethical guidelines and regulations in place to prevent such misuse. In addition, there is a growing concern about the ethical implications of automated decision-making systems. These systems use data analysis and modeling to make decisions without human intervention. While this can save time and resources, it also raises questions about fairness and accountability. Who is responsible if the system produces biased or discriminatory results? It is crucial to have ethical guidelines and regulations in place, including human oversight at all stages of the project, to ensure these systems are transparent, fair, and accountable.

Moreover, there is a responsibility to continuously monitor and reassess the ethical implications of data analysis and modeling. As technology and data practices evolve, so do ethical concerns. It is important to regularly review and adapt ethical standards to ensure the protection of individuals' rights and promote responsible and ethical rules of data usage.

While data analysis and modeling have the power to provide valuable insights and facilitate decision-making, it is crucial to consider the ethical implications surrounding this practice. Transparency, accountability, fairness, and the protection of individuals' rights should be at the forefront of any data analysis and modeling process.

Data Anonymization

As discussed in Ethics in Data Collection, the process of removing or modifying personally identifiable information is called data anonymization or pseudonymization, respectively. Anonymized data can usually still be utilized for analysis without compromising the individual's identity. Pseudonymization of data involves masking, encoding, or encrypting the values of specific features of the data. Features such as names or Social Security numbers can be exchanged with artificial values. If needed, these values can be replaced later using a secure lookup table or decryption algorithm. A common method of pseudonymization is called **hashing**, which

is the process of transforming data into a fixed-length value or string (called a hash), typically using an algorithm called a hash function. The key property of hashing is that it produces a unique output for each unique input (within practical constraints) while making it infeasible to reverse the transformation and retrieve the original input from the hash.

As an example, at a particular university, student records are stored securely on a private server. From time to time, student workers need to pull grade information from the server to perform analysis; however, there is a strict policy in place mandating that students may not have access to personally identifiable information of fellow students. Therefore, the university's information technology division has devised a coding scheme in which the last name and student ID are combined into a codeword, which may be used to access that student's grade information from the server. This is an example of pseudonymization, as the PII is encoded rather than removed from the dataset.

Robust data anonymization should follow a policy of **k-anonymization**, which is the principle of ensuring that each record within a dataset is indistinguishable from at least $k - 1$ other records with respect to a specified set of identifying attributes or features. This approach helps reduce the risk of re-identifying individuals in a dataset, allowing organizations to protect clients and their information without restricting their ability to use the data to gain insight and make decisions.

Data anonymization policy (including the determination of *k* in k-anonymization if needed) should be set by the regulatory compliance officer (RCO) or other compliance professionals with specialized knowledge of data privacy regulations. These professionals must be familiar with prevailing data privacy regulations and the data masking process to ensure that all possible security measures comply with these regulations. It is also important that all data masking processes use protected techniques to ensure that data can't be accessed or operated in an unauthorized manner.

EXAMPLE 8.7

Problem

Mike is a data privacy officer for his local school district. The district is looking to collect data from parents to gain insight into their opinions on the district's curriculum. However, Mike is concerned about protecting the privacy of the parents and their children. He knows that by including too much identifying information in the survey, it could potentially allow someone to re-identify the respondents. What steps can Mike take to protect the privacy of the respondents while still collecting useful data for analysis?

Solution

Mike decides to implement k-anonymization. To determine the appropriate value for *k*, Mike looks at the total number of respondents and the information collected in the survey. He knows that the more identifying characteristics he removes, the higher the level of anonymity will be. However, he also wants to make sure that the data is still useful for analysis. After some research, he determines that a minimum of 5 respondents should have the same characteristics in order to achieve a satisfactory level of anonymization.

After reviewing the survey questions, Mike sees that there are 10 potential identifying characteristics, such as the number of children, their school district, and their ages. To calculate the k-anonymization, he takes the total number of respondents and divides it by 5. For example, if there are 100 respondents, the k-anonymization would be $100/5 = 20$. This means that for each set of characteristics, there must be at least 20 respondents who share the same values. Mike implements this rule by removing or encrypting some of the identifying characteristics in the survey. For example, instead of asking for the exact age of the children, the survey will only ask for their age range. He also removes the specific names of schools and instead groups them by district. By doing so, he ensures that there are at least 20 respondents who share the same characteristics, thus achieving the k-anonymization.

Data that is under intellectual property rights, such as copyright, can also be anonymized as long as the intellectual property rights holder has given permission to do so. Anonymization of data is useful for maintaining privacy in sensitive data while also preserving the intellectual property rights of the original data. However, certain precautions should be taken to ensure that the data is not used for any unethical or illegal purposes.

The process of data anonymization generally follows a few common steps.

1. First, it is important to determine which data features need to be anonymized and which do not.
2. Once the data elements have been identified, the next step is to develop a technique to mask the sensitive data and remove the identifying elements.
3. Tools such as hashing and encryption can be used to anonymize the data.
4. It is important to ensure that all techniques used for data masking are secure and compliant with current privacy regulations.
5. The data should then be tested and observed to ensure that masking processes are implemented accurately.
6. For added security, data anonymization should also involve k-anonymization.
7. Finally, access management must be in place to ensure that only those with the right privileges can access the data.

An example of masked data:

Luke Smith, birthdate 1997, Social Security 567-89-1234, and address 123 Main Street becomes "LS97, XXX-XX-XXXX, XXX Main Street."

Here we see that the name and birthdate are hashed into a four-character code, “LS97,” while the Social Security number and street address number are masked completely, effectively removing those features completely.

Data Validation

Data validation (as introduced in Collecting and Preparing Data) is the process of checking, verifying, and validating the accuracy and reliability of data before it is used in decision-making. When both data validation and ethical rules are applied, the models and analyses that use the data are more reliable and effective.

Data validation provides a set of checks and controls that help ensure that data meets certain standards for accuracy and reliability. It also helps ensure that data is handled properly and follows ethical policies and principles. By establishing processes and procedures for data validation, companies can demonstrate a commitment to ethical data usage, which is an important component of any data governance policy. Effective data validation will secure data from the influence of bias, prejudices, or preconceptions. As mentioned earlier, unbiased data is essential for ethical modeling, as it gives an accurate illustration of a real-world situation rather than being influenced by personal intentions or opinions held by certain individuals and ensures that decisions are made on solid facts and evidence instead of subjective opinion.

The process of data validation can be divided into three key areas: outlier detection, cross-validation, and ethical guidelines.

Outlier detection concerns the identification of observations that are significantly different from the rest of the data and ensures that the data is free from outliers that can affect the results of the models and analysis. If outliers are detected, they may either be removed or imputed to have more reasonable values.

Cross-validation is a process used to evaluate a set of data, and it involves the comparison of the results with different subsets of the data or with the entire dataset by repeatedly breaking the data into training and testing sets and evaluating the model's performance on different subsets of the data. If the results vary too much on different subsets, then the model may have been over-fit, leading to biases and inaccurate predictions when used on new data.

Ethical guidelines for data validation cover a broad range of protocol and policy intended to inform when and how such tools as cross-validation and outlier detection and handling should be used.

There are overlaps and interactions between these three key areas of data validation. Ethical rules may influence the choice and implementation of outlier detection methods to ensure fairness and avoid discrimination. Cross-validation techniques can be used to assess the robustness of outlier detection algorithms and evaluate their performance across different subsets of the data, and ethical considerations may guide the interpretation of cross-validation results, especially when assessing the impact of outliers on model performance and fairness.

While ethical rules or guidelines, cross-validation, and outlier detection serve distinct purposes in data validation, they work together to ensure the reliability, fairness, and robustness of data-driven models and analyses. The intersections among ethical rules, cross-validation, and outlier detection is illustrated in Figure 8.4.

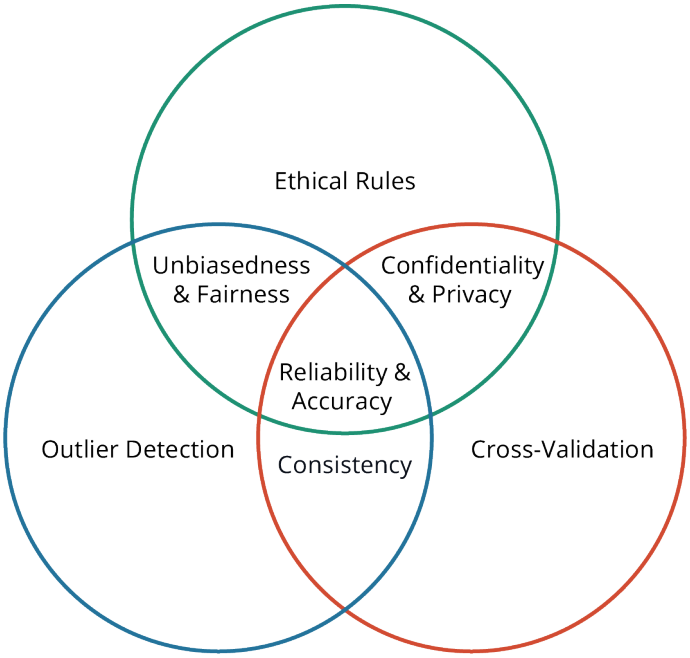


Figure 8.4 Venn Diagram Illustrating the Relationship Between Ethical Rules, Cross-Validation, and Outlier Detection for Data Validation and Analysis

EXPLORING FURTHER

The Environmental Impact of Data Science

This section has focused on the ethics of data and how it is used in modeling and data science. But did you know that data science also carries with it a substantial environmental impact? As datasets grow larger and modeling techniques, including AI, get more and more complex, vast arrays of computer servers are required to do the analysis and number crunching. The sheer number of computer servers, along with all the technology and devices that support them, greatly raises the carbon footprint of data science. The increased use of energy directly affects greenhouse gas emissions, which contributes to climate change and depletes valuable resources, leading to increased inequalities around the world. Global electricity consumption related to information and communication technology (of which significant support goes to data science tasks) is estimated to account for between 1 and 1.5% of all global electricity demand (https://openstax.org/r/iea), and greenhouse gas emissions from these sources are expected to exceed 14% of global emissions of greenhouse gasses by 2040 (Nordgren, 2023). The environmental effects of data

science must be considered alongside other ethical issues if we hope to maintain a sustainable relationship with the Earth.

8.3 Ethics in Visualization and Reporting

Learning Outcomes

By the end of this section, you should be able to:

- 8.3.1 Recognize the importance of visualizing data in a way that accurately reflects the underlying information.
- 8.3.2 Define data source attribution and its significance in data science and research.
- 8.3.3 Identify barriers to accessibility and inclusivity and apply universal design principles.

After data is collected and stored securely, it should be analyzed to extract insights from the raw data using authorized tools and approved procedures, including data validation techniques. The insights are then visualized using charts, tables, and graphs, which are designed to communicate the findings clearly and concisely. The final report and conclusion should be prepared, ensuring that the visualizations used adhere to ethical guidelines such as avoiding misleading information or making unsubstantiated claims. Any assumptions and unavoidable biases should be clearly articulated when interpreting and reporting the data.

It is essential to ensure that data is presented with fairness and in a way that accurately reflects the underlying research and understanding. All data should be accompanied by appropriate and factual documentation. Additionally, both data and results should be safeguarded to prevent misinterpretation and to avoid manipulation by other parties. Moreover, barriers to accessibility need to be identified and addressed, following guidelines of inclusivity and universal design principles. These ethical principles should be adhered to throughout the data analysis process, particularly when interpreting and reporting findings.

To maintain ethical principles throughout the data analysis process and in reporting, the data scientist should adhere to these practices:

1. Exercise objectivity when drafting and presenting any reports or findings.
2. Acknowledge all third-party data sources appropriately.
3. Ensure that all visualizations are unambiguous and do not focus on any sensationalized data points.
4. Construct visualizations in a meaningful way, utilizing appropriate titles, labels, scales, and legends.
5. Present all data in a complete picture, avoiding masking or omitting portions of graphs.
6. Ensure that the scales on all axes in a chart are consistent and proportionate.
7. Exercise caution when implying causality between connected data points, providing supporting evidence if needed.
8. Utilize representative datasets of the population of interest.

Accurate Representation

Accurate representation is a crucial aspect of data science and reporting, and it refers to presenting data in a way that authentically reflects the underlying information. This includes ensuring that the data is not misrepresented or manipulated in any way and that the findings are based on reliable and valid data. From a data scientist perspective, accurate representation involves this sequence of steps:

1. **Understanding the data.** Before visualizing or reporting on data, a data scientist must have a thorough understanding of the data, including its sources, limitations, and potential biases.
2. **Choosing appropriate visualizations.** Data can be presented in a variety of ways, such as graphs, charts, tables, or maps. A data scientist must select the most suitable visualization method that accurately represents the data and effectively communicates the findings.

3. **Avoiding bias and manipulation.** Data scientists must avoid manipulating or cherry-picking data to support a specific narrative or agenda. This can lead to biased results and misinterpretations, which can have serious consequences.
4. **Fact-checking and verifying data.** Data scientists must ensure the accuracy and validity of the data they are working with. This involves cross-checking data from multiple sources and verifying its authenticity.
5. **Proper data attribution.** Giving credit to the sources of data and properly citing them is an important aspect of accurate representation. This allows for transparency and accountability in data reporting.

When reporting results, data scientists must be transparent about the data they have collected and how it was utilized. Did they obtain informed consent from individuals before collecting their data? What measures were taken to ensure the quality and reliability of the data? Moreover, as data and algorithms become more complex, it might be challenging to understand the sense behind the results. The absence of transparency may lead to skepticism in the results and decision-making process. Data analysts and modelers must demonstrate their strategies and results clearly and legibly.

EXAMPLE 8.8

Problem

A team of data scientists is tasked with conducting a comprehensive analysis of a company's sales performance. To provide an accurate assessment of the trends and patterns, the team collects data from a variety of sources. The first data scientist presents a graph illustrating the company's sales trend over the course of the past year, including all identified outliers as shown in Figure 8.5. However, a second data scientist chooses to hide certain data points with lower sales figures from the graph as shown in Figure 8.6. Select and discuss the more appropriate representation of the data. Which data scientist is showing a more precise and representative visualization of the company's sales performance?

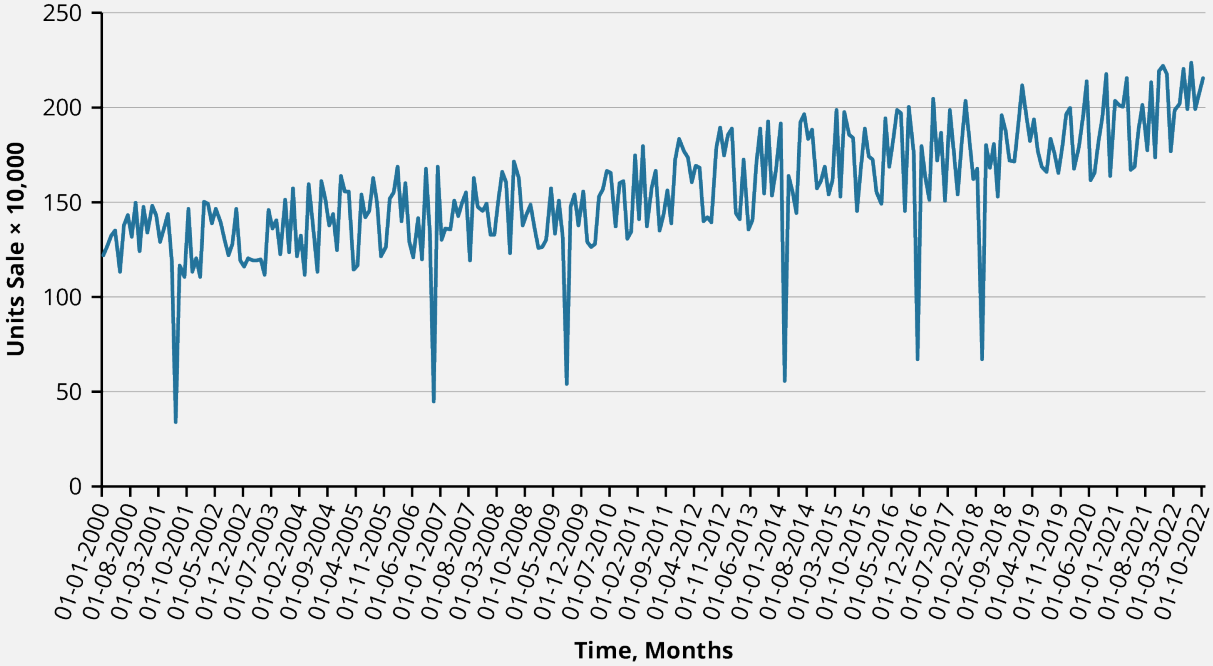


Figure 8.5 First Data Scientist's Presentation

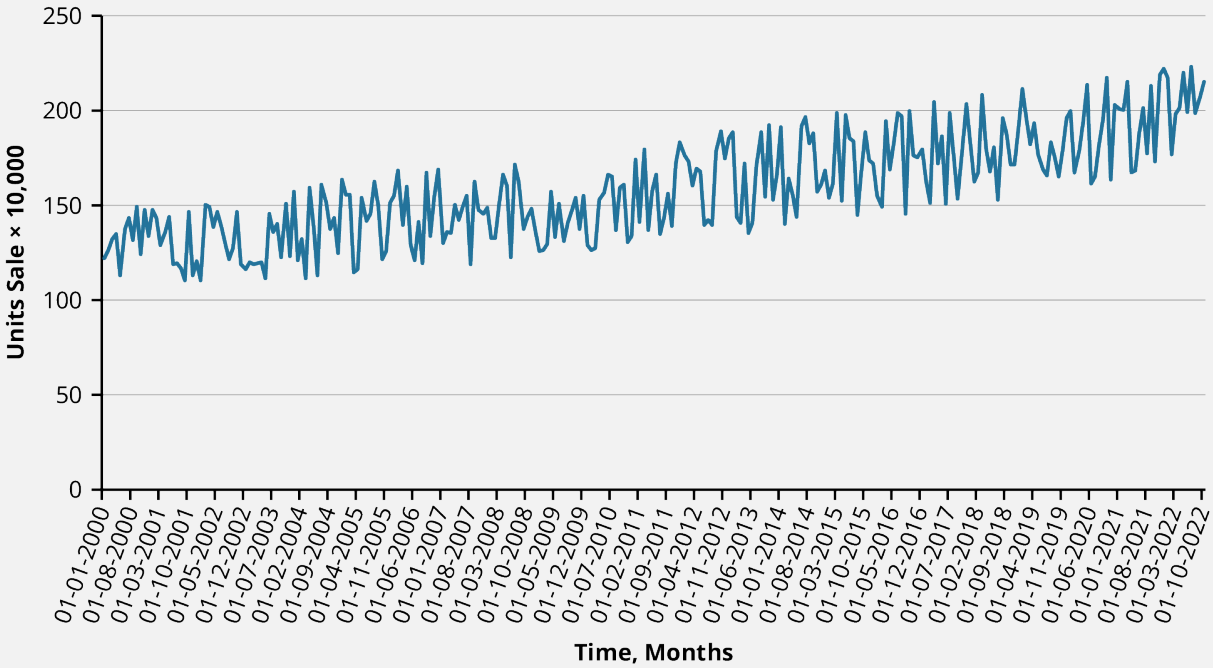


Figure 8.6 Second Data Scientist's Presentation

a. The second data scientist chooses to remove the low sales outliers from the graph. This approach can be useful in certain cases, such as when the data is extremely skewed and the outliers are significantly impacting the overall trend of the data. The first data scientist chooses to include all data points, including the outliers, in their graph. This means that the graph of the first data scientist includes unusual or extreme values that may lead to misleading insights and decisions.
b. The more accurate presentation would be the first one, which includes all the data points without any exclusion. By excluding the low sales points, the second data scientist is not presenting the complete picture, and it can cause misleading insights and decisions being made based on incomplete information. Additionally, the assumption is that the low sales points are genuine measurements and so may not in fact be random outliers.
c. In terms of accuracy, it is difficult to say which presentation is more accurate. In general, it is important to carefully consider the impact of outliers before deciding whether or not to include them in the data analysis. However, if there are clear errors or irregularities in the data, it may be appropriate to remove them to prevent them from skewing the results. Ultimately, both approaches have their merits and limitations.
d. The second data scientist chooses to remove the outliers from the data and only includes the data points that fall within a certain range. This results in a cleaner and more consistent trend in the graph, as the outliers are not shown to skew the data. However, removing the outliers may cause important information about the sales performance to not be accurately represented.

Solution

The correct answer is B. The presentation delivered by the first data scientist is deemed more precise as it offers a holistic and unbiased depiction of the sales trend. This enables a comprehensive understanding of the sales performance and any discernible patterns. On the contrary, the second data scientist's presentation may give the illusion of a positive trend in sales, but it fails to accurately portray the complete dataset. Omitting the lower sales points can distort the data and potentially lead to erroneous conclusions. Moreover, the act of disregarding data points goes against the principle of transparency in data analysis. Data scientists must present the entire dataset and accurately exhibit existing trends and patterns rather

than manipulating the data to support a desired narrative. Additionally, the removal of data points without disclosure in the graph violates ethical principles in data analysis.

EXAMPLE 8.9

Problem

Two expert data analysts have constructed bar graphs shown in Figure 8.7 and Figure 8.8 depicting the market share proportions of various soft drink brands presented in Table 8.1. Each bar is clearly labeled with the respective brand name and corresponding percentage, providing a straightforward and accurate representation of the data. Which of the two expert data analysts has accurately depicted the market share proportions of soft drink brands in their bar graph? Explain the difference between the two figures and potential consequences of each presentation.

Soft Drink	Share of Volume (%)
Coca-Cola	18.6%
Pepsi	9.1%
Diet Coke	8.0%
Dr. Pepper	7.5%
Sprite	7.2%
Mountain Dew	7.0%
Diet Pepsi	3.5%
Coke Zero	2.8%
Fanta	2.4%
Diet Mountain Dew	2.1%

Table 8.1 Global Soft Drinks Industry, Top 10 Brands in 2019, by Share of Volume (source: https://www.wm-strategy.com/news/soft-drinks-industry-trends-and-size)

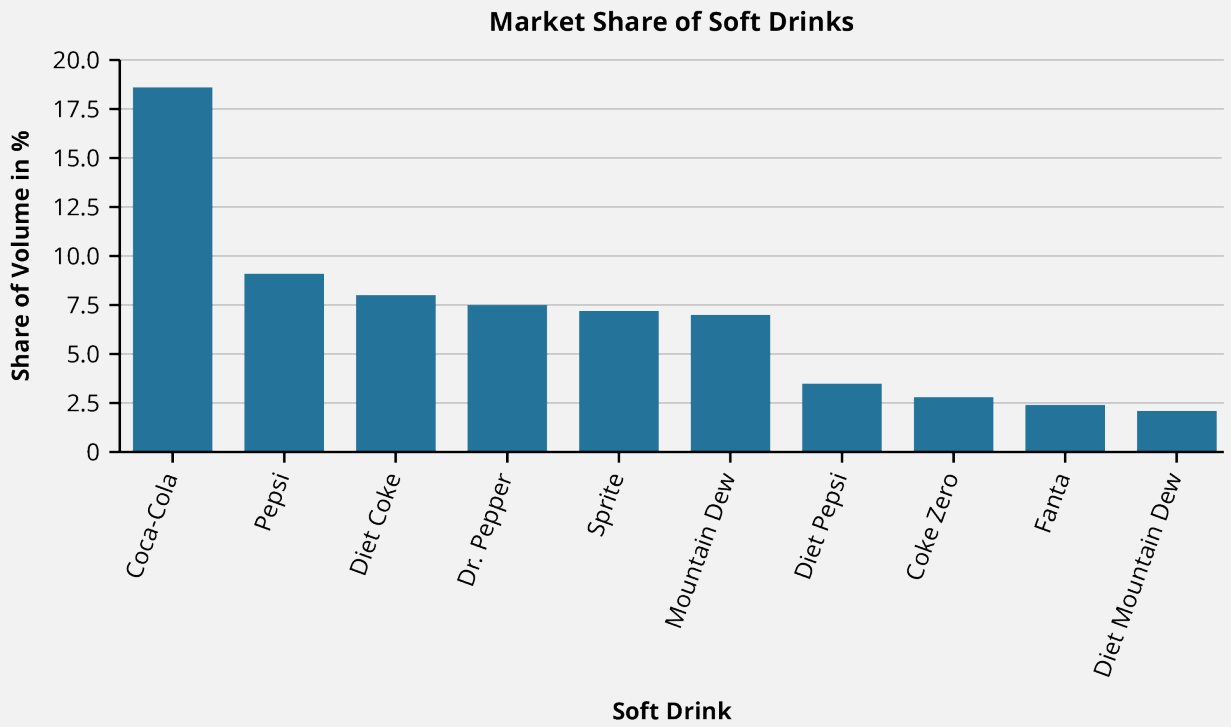


Figure 8.7 Graph A: Representation of First Data Scientist's Analysis

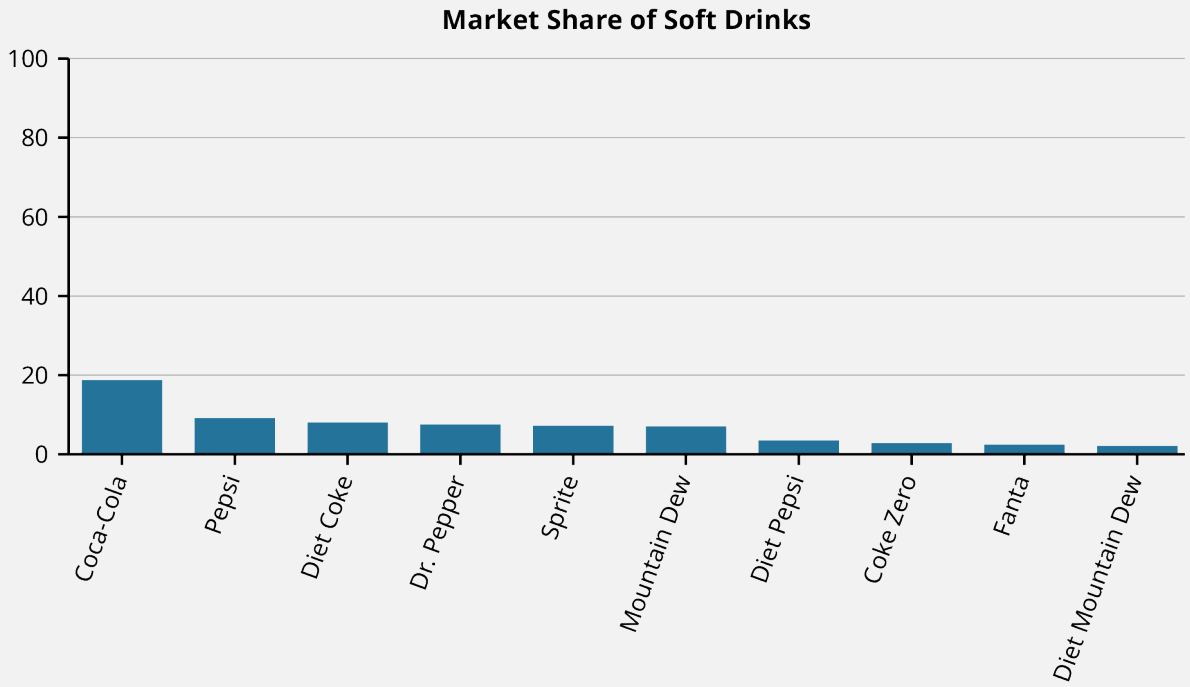


Figure 8.8 Graph B: Representation of Second Data Scientist's Analysis (Market Share in %)

Solution

The first data scientist, who generated Graph A in Figure 8.7, has effectively represented the brand names and their corresponding market share percentages, using correct labeling and scaling of axes. This facilitates easy interpretation and comparison of the data. The second data scientist, who produced Graph B as shown in Figure 8.8, scaled the vertical axis from 0 to 100, making it more difficult to observe small differences in percentages. Moreover, Graph B fails to label the horizontal or vertical axes in equivalent

detail, which may cause undue confusion, even if the intended labels may be clear from context.

EXAMPLE 8.10

Problem

A data scientist at a health care research institution is analyzing the effects of a new medication on a specific disease. After collecting data from multiple clinical trials, the scientist summarized the results in Table 8.2 and created a histogram to show the distribution of participants across different age groups as shown in Figure 8.9. The scientist included age intervals of 18–25, 26–36, 37–46, 47–56, 57–66, and 67+ in the chart. To provide a complete understanding of the data, what additional information should be included?

a. The percentage of participants in each group to accurately depict the proportion of participants in each age group
b. The total number of participants to demonstrate the scope of the results
c. The number of participants in each age group for a comprehensive overview of the data
d. Options A, B, and C should all be included in the chart

Age	Participants
18–26	100
27–36	120
37–46	107
47–56	71
57–66	89
67+	113

Table 8.2 Statistical Data about Clinical Trials Participants

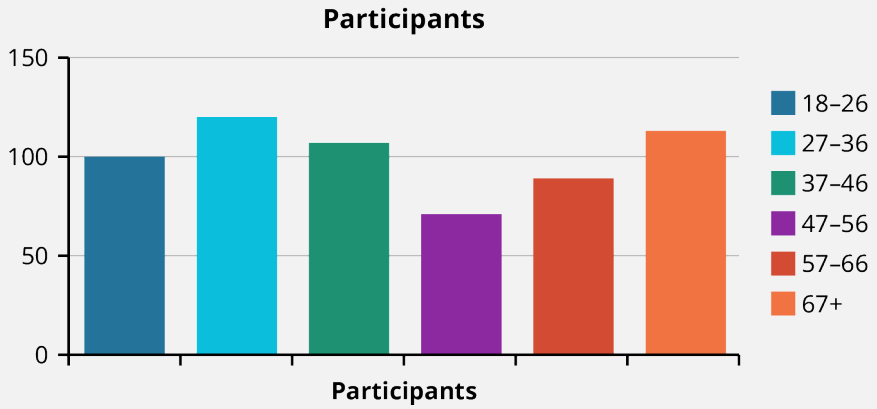


Figure 8.9 Insufficient Information on Participants in Clinical Trials

Solution

The correct answer is D. This is the most appropriate choice because the presentation of results needs to be improved to accurately depict the data. This includes clearly indicating the percentage and total number of participants in each group and the overall number of participants as given in Figure 8.10. See Statistical

Inference and Confidence Intervals for a review of the significance and implications of accuracy and confidence in relation to data reliability and decision-making processes.

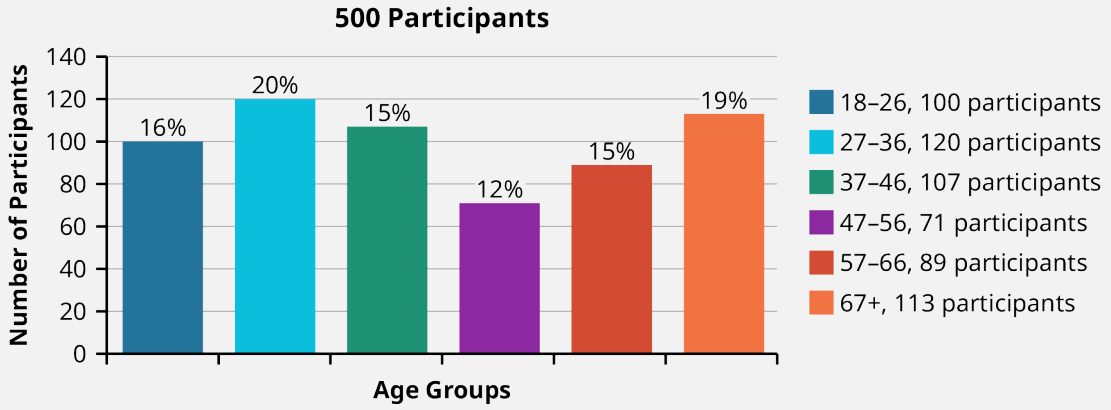


Figure 8.10 Data-based Clinical Trials Participants

Data Source Attribution

Data source attribution is the important practice of clearly identifying and acknowledging the sources employed in the visualizations and reporting of data. It is the data scientist's responsibility to uphold these principles and present data in a manner that is both transparent and ethical. Data source attribution is demanded for several reasons:

1. **Accuracy and credibility.** Attribution ensures that the data utilized in the visualizations and reporting is accurate and reliable. By clearly stating the sources, any potential errors or biases in the data can be identified and corrected, increasing the credibility of the information presented.
2. **Trust and transparency.** By disclosing the sources of data, it promotes trust and transparency between data producers and consumers. This is especially important when data is used to inform decision-making or to shape public perceptions.
3. **Accountability.** Featuring full attribution in the available results allows the rest of the research community to validate and further research the results provided. This safeguard enforces data integrity and holds developers accountable for the claims and conclusions put forth.
4. **Privacy considerations.** In some circumstances, the data employed in visualizations may retain sensitive or personal information. Attributing the source of the data can help protect individuals' privacy and avoid potential harm or misuse of their information.

It is important to note that if the data utilized in graphs or reports are not publicly available, permission must be obtained from the source before using it in new research or publication. This would also require proper attribution and citation to the source of the data.

Data source attribution is also closely tied to the principles of accessibility and inclusivity. By clearly stating the sources of data, it enables people with different backgrounds and abilities to access and understand the information presented.

Proper attribution of sources also builds trust. Without explicit citations and permissions in place, the data scientist can be accused of plagiarism, whether this is done intentionally or not. Consider the use of AI in education. Some instructors allow their students to use resources such as ChatGPT to assist them in their writing and research. Similarly to citing textual references, these instructors might require their students to include a statement on the use of AI, including what prompts they used and exactly what the AI or chatbot responded. The student may then modify the responses as needed for their project. Other instructors disallow the use of AI altogether and would consider it to be cheating if a student used this resource on an assignment.

Questions of academic integrity abound, and similar ethical issues must also be considered when planning and conducting a data science project.

Accessibility and Inclusivity

One of the ethical responsibilities of data scientists and researchers is to ensure that the data they present is accessible and inclusive to all individuals regardless of their capabilities or experiences. This includes considering the needs of individuals with disabilities, individuals from different artistic or linguistic environments, and individuals with different levels of education or literacy. Indeed, incorporating universal design principles when reporting results will aid not only those with different abilities, but every individual who reads or views the report.

Universal design principles refer to a set of guidelines aimed at creating products, environments, and systems that are accessible and usable by all people regardless of age, ability, or disability. The goal of universal design is to ensure inclusivity and promote equal access, allowing everyone to participate in everyday activities without additional adaptations or accommodations. In data science, this may involve creating visualizations and reports that utilize accessible fonts, colors, and formats, and providing alternative versions for individuals who may have difficulty accessing or understanding the data in its original form. Barriers to accessibility and inclusivity include physical barriers, such as inaccessible visualizations for individuals with visual impairments. Also, linguistic and cultural barriers may prevent people from outside the researchers' cultural group, or those with limited literacy and non-native speakers, from fully understanding complex data visualizations and reports. By applying universal design principles, data scientists and researchers can help mitigate these barriers and ensure that the presented data is accessible and inclusive to a wide range of individuals.

In addition to differences of ability, there are also significant issues in the ability of some individuals to access and use digital technologies because of socioeconomic differences. The **digital divide** refers to the gap between those who have access to digital technologies, such as the internet and computers, and those who do not. Factors such as geographic location, age, education level, and income level contribute to the digital divide, which can lead to data inequality, where certain groups are underrepresented or excluded from data-driven decision-making. Addressing the digital divide through investments in infrastructure (e.g., developing reliable internet access in underserved areas), digital literacy education, and inclusive data collection (methods that do not rely solely on participants having access to technology) will ultimately narrow this divide and foster greater social and economic equity.

Last but not least, data scientists should be cognizant of the role that their field plays in opening up opportunities to historically underrepresented and marginalized communities. The field of data science is becoming more inclusive over time, with greater emphasis on diversity, equity, and inclusion (DEI). In recent years, there has been a growing recognition of the need to increase diversity and inclusion in data science. Efforts to address this imbalance include creating mentorship programs, supporting underrepresented groups in STEM education, and promoting equitable hiring practices in tech companies. As the field evolves, there's a concerted effort to welcome more women and people from underrepresented backgrounds into data science roles.

Initiatives aimed at promoting representation in data science careers, addressing bias in data and algorithms, and supporting equitable access to educational resources are increasingly common. By fostering a culture of inclusion, data scientists can not only drive innovation but also ensure that the insights and technologies they develop benefit all segments of society.

Data science is becoming a standard part of the educational landscape, reaching diverse learners and providing pathways to employment in the field. The transition of data science from a specialized, highly technical, and exclusive subject to one that can be approached by a wider audience helps to bridge gaps in educational opportunities and fosters a more inclusive pipeline into data science careers. It also reflects the

increasing demand for data science skills in the workforce, encouraging broader adoption in varied educational settings.

Key Terms

anonymization act of removing personal identifying information from datasets and other forms of data to make sensitive information usable for analysis without the risk of exposing personal information

anonymous data data that has been stripped of personally identifiable information (or never contained such information in the first place)

autonomy in data science, the ideal that individuals maintain control over the decisions regarding the collection and use of their data

confidentiality safeguarding of privacy and security of data by controlling access to it

cookies small data files from websites that are deposited on users' hard disk to keep track of browsing and search history and to collect information about potential interests to tailor advertisements and product placement on websites

copyright protection under the law for original creative work

cross-validation comparison of the results of a model with different subsets of the data or with the entire dataset by repeatedly breaking the data into training and testing sets and evaluating the model's performance on different subsets of the data

data breach the act of data being stolen by a malicious third party

data governance protocols set of rules, policies, and procedures that enable precise control over data access while ensuring that it is safeguarded

data privacy the assurance that individual data is collected, processed, and stored securely with respect for individuals' rights and preferences

data retention how long personal data may be stored

data security steps taken to keep data secure from unauthorized access or manipulation

data sharing processes of allowing access to or transferring data from one entity (individual, organization, or system) to another

data source attribution the practice of clearly identifying and acknowledging the sources employed in the visualizations and reporting of data

data sovereignty laws that require data collected from a country's citizens to be stored and processed within its borders

digital divide gap between those who have access to digital technologies, such as the internet and computers, and those who do not

encryption the process of converting sensitive or confidential data into a code in order to protect it from unauthorized access or interception

ethics in data science responsible collection, analysis, use, and dissemination of data

explainable AI (XAI) set of processes, methodologies, and techniques designed to make artificial intelligence (AI) models, particularly complex ones like deep learning models, more understandable and interpretable to humans

fairness absence of bias in the models and algorithms used to process data

Family Educational Rights and Privacy Act (FERPA) legislation providing protections for student educational records and defining certain rights for parents regarding their children's records

hashing process of transforming data into a fixed-length value or string (called a hash), typically using an algorithm called a hash function

Health Insurance Portability and Accountability Act (HIPAA) U.S. legislation requiring the safeguarding of sensitive information related to patient health

informed consent the process of obtaining permission from a research subject indicating that they understand the scope of collecting data

intellectual property original artistic works, trademarks and trade secrets, patents, and other creative output

k-anonymization principle of ensuring that each record within a dataset is indistinguishable from at least $k - 1$ other records with respect to a specified set of identifying attributes or features

outlier detection identification of observations that are significantly different from the rest of the data

personally identifiable information (PII) information that directly and unambiguously identifies an individual

pseudonymization act of replacing sensitive information in a dataset with artificial identifiers or codes while still maintaining its usefulness for analysis

regulatory compliance officer (RCO) a trained individual responsible for confirming that a company or organization follows the laws, regulations, and policies that rule its functions to avoid legal and financial risks

transparency being open and honest about how data is collected, stored, and used

universal design principles set of guidelines aimed at creating products, environments, and systems that are accessible and usable by all people regardless of age, ability, or disability

Group Project

Project A: Analysis of One-Year Temperatures

The World Wildlife Fund (WWF), the largest privately supported international conservation organization, recently advertised for a data scientist to investigate temperature variations. After receiving applications from ten highly qualified candidates, the organization selected four applicants and asked them to complete the following project. Utilizing temperature data collected by the NOAA National Centers for Environmental Information (NOAA, 2021), applicants were asked to create a graphical representation of the temperature trend throughout the year 2020, using maximum, average, and minimum monthly temperatures. As an organization committed to upholding ethical and professional standards, the World Wildlife Fund carefully evaluated the submitted graphs and selected one successful candidate for the data scientist position. The selected applicant was chosen based on their ability to present the data in a transparent manner, closely adhering to ethical principles.

For your group assignment, please conduct a search on a trustworthy website for a dataset containing temperature recordings for the year 2020 in your state. Specifically, gather the datasets for the maximum, average, and minimum temperatures.

As a group, complete these six steps:

1. Discuss the ethical principles that should be applied in graphical presentation of temperature variation.
2. Determine the appropriate *x*-axis label and *y*-axis label and their units.
3. Come up with a meaningful graph title.
4. Identify the fundamental features that should be available in the graph.
5. Write a description explaining the temperature variation, comparing the minimum, average, and maximum temperatures, and how this graph can be useful or not useful in predicting the temperatures on any day of the next year.
6. Finally, produce the graph containing all the elements described in Steps 1 to 5.

Project B: Assess the Quality of the Food Services

The school cafeteria plays a crucial role in providing students with sustenance during their academic day.

The objective of this project is to obtain a comprehensive understanding of students' opinions and suggestions regarding the quality of food services provided in the cafeteria. The gathered information will be used to enhance the food quality at the school, thereby potentially improving the academic performance of students.

Group 1 will be responsible for creating a questionnaire survey on the food services quality in the school cafeteria.

Group 2 will be responsible for analyzing the data and visualizing the outcomes of the food service quality in the school cafeteria.

Group 1: Questionnaire Survey

Group Formation: Each group will consist of two teams, with distinct roles:

- The first team will be responsible for creating a questionnaire survey regarding the food services in the school cafeteria.
- The second team will analyze the survey questions based on ethical principles, specifically considering privacy, data security, and data sharing practices as well as evaluating for bias and fairness.

Questionnaire Survey: The survey will focus on gathering feedback and opinions about the food services in the school cafeteria. It should include questions that cover various aspects such as food quality, variety, pricing, customer service, and overall satisfaction.

Ethical Considerations: The second team will carefully analyze each question in the survey, taking into account ethical principles such as privacy, data security, and data sharing. They will also evaluate for bias and fairness in the questions to ensure inclusivity and objectivity.

Modifications and Informed Consent: As a group, discuss any potential modifications that can be made to improve the survey's effectiveness and ethical practices. Before distributing the survey, a detailed Informed Consent document needs to be prepared, outlining the purpose of the survey, the use of the collected data, and the participants' rights. This document will be required to be read and signed by all participants before they can complete the survey.

Group 2: Data Visualization

Group Formation: Each group will consist of two teams, with distinct roles:

- The first team will be responsible for analyzing and reporting the outcomes from Group 1 by manipulating graphs, tables, bar charts, and pie charts.
- The second team will analyze the visual object from the first team based on ethical principles.

Ethical Considerations: The second team will carefully analyze each visual object in the report, taking into account ethical principles such as presentation accuracy, data source attribution, accessibility, and inclusivity.

Modifications and Conclusion: As a group, discuss any potential modifications that can be made to improve the reported findings based on ethical practices. Before publishing the final report, a clear and unbiased conclusion should summarize all aspects of the findings and review based on ethical principles to achieve the objective of the project.

Project C: Using Data to Predict Ransomware Attacks

Ransomware attacks have become an increasingly worrisome challenge for organizations. A variety of companies and government agencies monitor this trend. Go to a reputable analyst website, such as this Comparitech publication (https://openstax.org/r/comparitech). Their primary objective is to compare the frequency and severity of attacks across four major sectors: business, health care, government, and education. A team of security experts gathers information from a variety of sources, such as news reports, government agencies, and victim reports. The team's analysis reveals a concerning increase in ransomware attacks over the past decade and huge spikes in the business sector in particular, sparking concerns about the effectiveness of current cybersecurity measures in these industries.

Discuss the findings presented in the online Comparitech visual graphs (https://openstax.org/r/blog1) (or from a similar data source, such as DNI government publication (https://openstax.org/r/dni)).

What can you conclude from these findings? Are ransomware attacks predictable? How should any outliers in the data be handled? What are some reasonable options for policy makers who want to protect organizations from attack? Research one to two specific examples of an organizational target of a ransomware attack to support your case.

Chapter Review

1. Which option best explains the significance of informed consent during the data collection phase?
   a. To ensure individuals are aware of the potential risks and benefits of the research
   b. To bypass legal regulations and guidelines
   c. To collect data without the knowledge of individuals
   d. To speed up the data collection process

2. Who is responsible for obtaining informed consent from individuals or companies before collecting their data?
   a. Data analysts
   b. Government agencies
   c. Medical professionals
   d. Data scientists

3. Which of the following is an important ethical principle to consider during the data collection phase?
   a. Maximizing collected data
   b. Minimizing informed consent
   c. Ensuring secure storage and access protocols
   d. Avoiding compliance with regulations and guidelines

4. What could be a serious consequence of a data security breach during the development process of a data science project?
   a. Unauthorized access to sensitive information
   b. Alteration or destruction of data
   c. Risk of legal issues
   d. All of the above

5. How can organizations mitigate the risk of data breaches during the development process?
   a. Have strong data security policies and protocols in place
   b. Respond to data breaches in a timely manner
   c. Only use publicly available data
   d. Share data with third parties without permission

6. Which of the following are valid sources of data for a data science project? Select all that apply.
   a. Public datasets
   b. Private databases from unaffiliated sources
   c. Open repositories such as Kaggle.com and OpenML
   d. All of the above

7. What type of data can be shared in a data science project?
   a. Quantitative and qualitative data only
   b. Tabular, text-based, and image-based data only
   c. Audio-based data only
   d. Quantitative, qualitative, tabular, text-based, and audio-based data

8. Who should be given access to the data in a data science project?
   a. Anyone who is directly involved in the project
   b. Shareholders and customers only
   c. Anyone who is directly involved in the project and has a legitimate need for the data

d. Anyone with a valid email address

9. Which department is responsible for handling regulatory compliance in an organization?
a. Human resources
b. Marketing
c. Legal and compliance teams
d. Manufacturing

10. What is the main purpose of considering fairness and equity in machine learning models?
a. To increase the accuracy of the model by avoiding bias
b. To avoid legal trouble
c. To improve the interpretability of the model
d. To focus on one group of data used in training

11. A new social media platform is looking for ways to improve its content-delivery algorithm so that it appeals to a broader audience. Currently, the platform is being used primarily by college-aged students. Which of these datasets would be the most appropriate data to collect and analyze to improve the algorithm and reduce potential bias in the content-delivery model?
a. Dataset 1, consisting of all the past browsing data on the platform
b. Dataset 2, the results of an extensive survey of people from various backgrounds and demographic segments who may or may not be familiar with the new social media platform
c. Dataset 3, the results of an internal company questionnaire

12. What type of data would be considered an outlier in machine learning models?
a. Data points that are well-represented in the training set
b. Data points that are rare or abnormal in comparison to the rest of the data
c. Data points that are equally distributed among all classes
d. Data points that are consistently labeled incorrectly

13. Why is it important to monitor and regularly test algorithms for any potential bias?
a. To ensure models are not overfitting
b. To maintain the accuracy of the model
c. To uphold fairness and equity in the decision-making process
d. To avoid legal consequences

14. The admissions team of a large university would like to conduct research on which factors contribute the most to student success, thereby improving their selection process for new students. The dataset that they plan to use has numerical features such as high school GPA and scores on standardized tests as well as categorical features such as name, address, email address, and whether they played sports or did other extracurricular activities. The analysis of the data will be done by a team that involves student workers. Which of these features should be anonymized before the student workers can get to work?
a. Name, address, and email address
b. High school GPA and scores on standardized tests
c. Sports and extracurricular activities
d. None of the data should be anonymized

15. In what situation can data be anonymized even if it falls under intellectual property rights?
a. When the individual has given consent for anonymization
b. When the data is used for research purposes
c. When there is a potential threat to national security

d. When the data will be used for marketing purposes

16. Data source attribution promotes trust and transparency by:
a. Identifying potential errors and biases in the data
b. Allowing others to replicate the results
c. Protecting the privacy and dignity of individuals
d. All of the above

17. Applying universal design principles in data visualization and data source attribution helps ensure that people in which group are able to view and use the data?
a. People with visual impairments
b. People with cognitive impairments
c. People with auditory impairments
d. Any person who will use the data

18. What is the term used for properly crediting and acknowledging the sources of data in data science and research?
a. Data attribution
b. Data manipulation
c. Data accuracy
d. Data extraction

19. What is one ethical responsibility of data scientists and researchers when presenting data?
a. Maximizing profits
b. Manipulating data to fit a desired narrative
c. Ensuring accessibility and inclusivity
d. Following the latest trends in data visualization techniques

20. There was a growing concern at the Environmental and Agriculture Department about the rising levels of CO_2 in the atmosphere. It was clear that human activities, especially the burning of fossil fuels, were the primary cause of this increase. The department assigned a team of data scientists to research the potential consequences of climate patterns in the United States. They studied the trends of CO_2 emissions over the years and noticed a significant uptick in recent times. The team simulated various scenarios using climate models and predicted that if the current trend continues, there will be disastrous consequences for the climate. The four graphs displayed in Figure 8.11 depict the identical dataset from four different researchers. Out of the four graphs, which one accurately represents the CO_2 data without any type of distortion or misrepresentation?

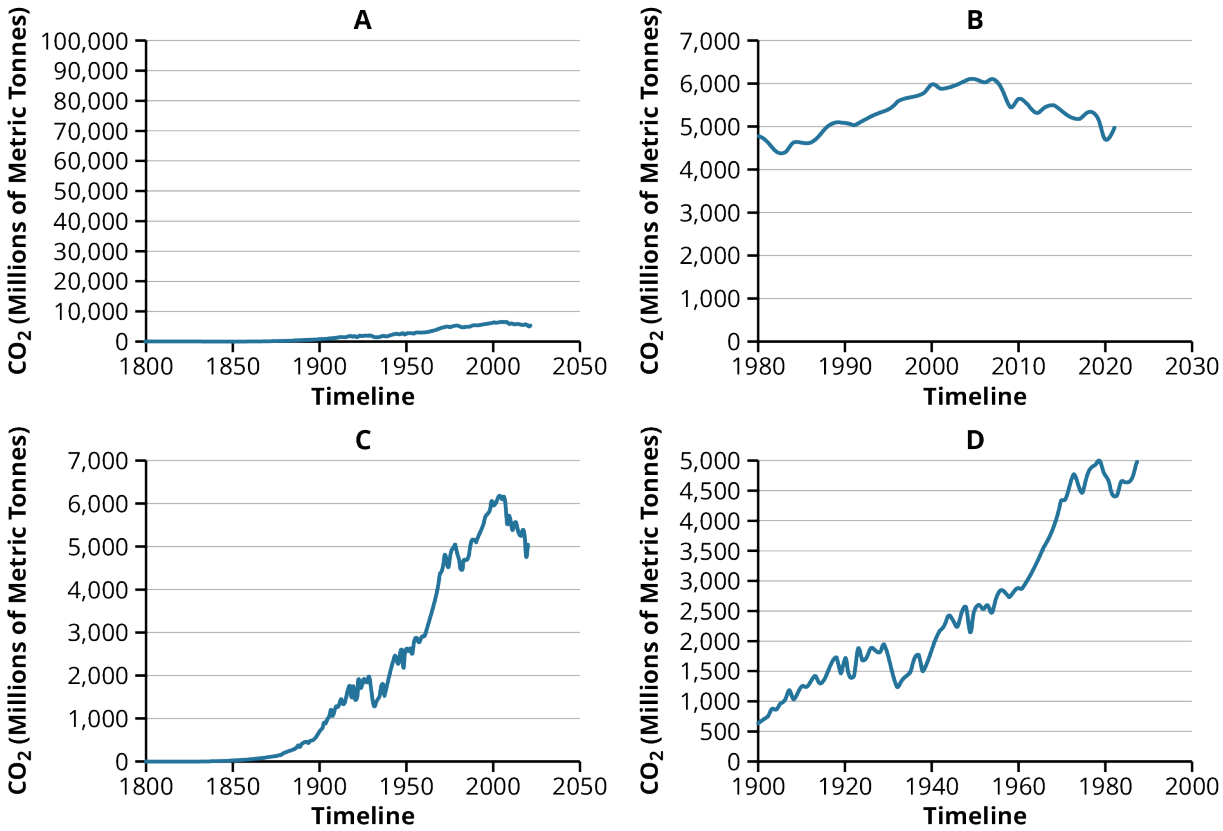


Figure 8.11 Source: Hannah Ritchie, Max Roser and Pablo Rosado (2020) - "CO_2 and Greenhouse Gas Emissions." Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/co2-and-greenhouse-gas-emissions' [Online Resource]

a. Graph A
b. Graph B
c. Graph C
d. Graph D

Critical Thinking

1. Why is it important to establish infrastructure management and protocols for data sharing?
2. What is the purpose of anonymizing or pseudonymizing data in a data science project?
3. How does data validation contribute to ethical data usage?

References

Berk, R., Heidari, H., Jabbari, S., Kearns, M., & Roth, A. (2021). Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research, 50*(1), 3-44. https://doi.org/10.1177/0049124118782533

NOAA National Centers for Environmental Information. (2021). *State of the climate: Global climate report for annual 2020*. https://www.ncdc.noaa.gov/sotc/global/202013

Nordgren, A. (2023). Artificial intelligence and climate change: Ethical issues. *Journal of Information, Communication, and Ethics in Society*, *21*(1), 1-15. https://doi.org/10.1108/JICES-11-2021-0106

U.S. House of Representatives, Committee on Oversight and Government Reform. (2018). *Majority staff report: The Equifax data breach*. https://oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf.

9 Visualizing Data

Figure 9.1 Data visualization techniques can involve collecting and analyzing geospatial data, such as measurements collected by the National Park Service during the High-Efficiency Trail Assessment Process (HETAP). (credit: modification of work "Trail Accessibility Assessments" by GlacierNPS/Flickr, Public Domain)

Chapter Outline

9.1 Encoding Univariate Data
9.2 Encoding Data That Change Over Time
9.3 Graphing Probability Distributions
9.4 Geospatial and Heatmap Data Visualization Using Python
9.5 Multivariate and Network Data Visualization Using Python

Introduction

Data visualization serves as an effective strategy for detecting patterns, trends, and relationships within complex datasets. By representing data graphically, analysts can deduce relationships, dependencies, and outlier behaviors that might otherwise remain hidden in the raw numbers or tables. This visual exploration not only aids in understanding the underlying structure of the data but also helps to detect trends and correlations and leads data scientists to more informed decision-making processes.

Data visualization can enhance communication and comprehension across diverse audiences, including decision-makers in organizations, clients, and fellow data scientists. Through intuitive charts, graphs, and interactive dashboards, complex insights can be conveyed in a clear, accessible, and interactive manner. Effective **data visualization** transforms raw data into easier-to-understand graphical representations, which then helps to facilitate and share insights into analytical conclusions. A data scientist can help generate data presentations for management teams or other decision-makers, and these visualizations serve as a tool to aid and support decision-making.

In addition, data visualizations are instrumental in examining models, evaluating hypotheses, and refining analytical methodologies. By visually inspecting model outputs, data scientists can assess the accuracy and robustness of predictive algorithms, identify areas for improvement, and iteratively refine and improve their models. Visualization techniques such as histogram, time series charts, heatmaps, scatterplots, and geospatial maps offer important insights into model performance and predictive ability, which allows data scientists to

refine and improve models with the result of improved predictive accuracy and reliability. Data visualization is a key component of the data science lifecycle and is an important tool used by data scientists and researchers.

Throughout this text we have discussed the concept of data visualization and have used technology (primarily Python) to help generate appropriate visual output such as graphs, charts, and maps. In Descriptive Statistics: Statistical Measurements and Probability Distributions, we introduced graphs of probability distributions such as binomial and normal distributions. Then, in Inferential Statistics and Regression Analysis, we introduced and worked with scatterplots for bivariate data. In Time and Series Forecasting, methods to visualize time series data and trend curves were reviewed. In this chapter, we review some of the basic visualization tools and extend the discussion to more advanced techniques such as heatmaps and visualizing geospatial and three-dimensional data. Reporting Results will provide strategies for including data visualization in executive reports and executive summaries.

In this chapter, we will review the basic techniques for data visualization and provide more detail on creating line charts and trend curves, with and without Python. We also explore techniques such as geospatial and multivariate data analysis that can uncover hidden dependencies or hidden relationships that are critical to researchers, data scientists, and statisticians.

9.1 Encoding Univariate Data

Learning Outcomes

By the end of this section, you should be able to:

- 9.1.1 Visualize data with properly labeled boxplots, histograms, and Pareto charts.
- 9.1.2 Use Python to generate various data visualizations for univariate data.

We've seen that data may originate from surveys, experiments, polls, questionnaires, sensors, or other sources. Data may be represented as numeric values, such as age or salary, or as categories, such as hair color, political affiliation, etc. Once data is collected and organized, a data scientist is interested in creating various visualizations, graphs, and charts to facilitate communication and assist in detecting trends and patterns.

Encoding text involves converting the text into a numerical representation for machine learning algorithms to process (see Data Cleaning and Preprocessing). **Univariate data** includes observations or measurements on a single characteristic or attribute, and the data need to be converted in a structured format that is suitable for analysis and visualization. **Bivariate data** refers to data collected on two (possibly related) variables such as "years of experience" and "salary." The data visualization method chosen will depend on whether the data is univariate or bivariate, as discussed later in the chapter. For example, boxplots might be used to graph univariate data, whereas a scatterplot might be used to graph bivariate data.

Univariate data can be visualized in many ways, including as bar graphs (or bar charts). **Bar graphs** present categorical data in a summarized form based on frequency or relative frequency. For example, a college administrator might be interested in creating a bar chart to show the geographic diversity of students enrolling in the college during a certain semester.

In this section, we discuss three commonly used visualization methods—boxplots, histograms, and Pareto charts.

Boxplots

A **boxplot** or **box-and-whisker plot** is a graph used to display a distribution of a dataset based on the **quartiles**, minimum, and maximum of the data. This is called the five-number summary, and thus a boxplot displays the minimum, Q_1 (first quartile), **median**, Q_3 (third quartile), and the maximum of a dataset. Note that a boxplot can be created using a horizontal format or vertical format. A boxplot can also be created with multiple plots, and this is helpful to compare two or more variables side by side. A boxplot is also useful to identify **outliers** and investigate skewness of a dataset.

For example, a human resources administrator might want to compare salaries for men versus women employees and thus create two side-by-side boxplots showing the distribution of men's salaries vs. women's salaries at the company. An example of a side-by-side boxplot created using Python appears in Using Python to Create Boxplots.

The steps needed to create a horizontal boxplot are given in the following list; keep in mind that typically we use a tool such as Python or Excel to create these boxplots.

1. Create a horizontal scale that extends from the minimum to the maximum of the dataset.
2. Draw a rectangle above the numerical scale where the left side of the rectangle is placed at the first quartile of the dataset and the right side of the triangle is placed at the third quartile of the dataset.
3. Draw a vertical line within the rectangle located at the median.
4. Draw a whisker to the left extending from the rectangle to the location of the minimum. Also, draw a whisker to the right extending from the rectangle to the location of the maximum.
5. An optional step is to plot outliers on the boxplot either to the left of the left whisker or to the right of the right whisker. (See Descriptive Statistics: Statistical Measurements and Probability Distributions for a further explanation regarding identification of outliers). Typically, an outlier is plotted using an asterisk or other symbol on the boxplot. A boxplot that includes outliers is sometimes referred to as a *modified boxplot*.

Note that boxplots can also be created in a vertical format; an example of a vertical boxplot created using Python is shown in Example 9.1.

Boxplots provide important information to the data scientist to assist with identifying the shape of the distribution for a dataset. After creating the boxplot, the researcher can visualize the distribution to identify its shape, whether there are outliers, and if any skewness is present.

The following guidelines may be useful in determining the shape of a distribution in a boxplot:

- **Variability:** Recall that the length of the box represents the interquartile range, IQR. Both the IQR and the length of the whiskers can provide information about the spread and variability of the data. A narrower box and shorter whiskers indicate lower variability, while a wider box and longer whiskers indicate higher variability.
- **Outliers:** Boxplots can provide a visual representation of outliers in the data. Outliers are data points that can be plotted outside the whiskers of the boxplot. Identifying outliers can provide insights into the tails of the distribution and help determine whether the distribution is skewed.
- **Skewness:** Boxplots can help identify skewness in the distribution. Recall from Collecting and Preparing Data that skewness refers to the lack of symmetry in a distribution of data. If one whisker (either the left or right whisker) is longer than the other, it indicates skewness toward that side. For instance, if the left whisker is longer, the distribution may be left-skewed, whereas if the right whisker is longer, the distribution may be right-skewed.
- **Symmetry:** The symmetry of the distribution can be determined by examining the position of the median line (the line inside the rectangle). If the median line is closer to one end of the box, the distribution may be skewed in that direction. If the median line is positioned close to the center of the box, the distribution is likely symmetric.

Using Python to Create Boxplots

To help with data visualization, Python includes many built-in graphing capabilities in a package called `Matplotlib` (https://openstax.org/r/matplotlib1), which we first introduced in Python Basics for Data Science.

`Matplotlib` is a library containing a variety of plotting routines used for creating high-quality visualizations and graphs to generate visualizations and plots for data exploration, analysis, and presentation. `Matplotlib` contains functions such as `bar`, `plot`, `boxplot`, and `scatter` that can be used to generate bar charts, time

series graphs, boxplots, and scatterplots, respectively. We've already seen how this package is used to generate a variety of visualizations, including scatterplots (in Inferential Statistics and Regression Analysis), time series graphs (in Time and Series Forecasting, and clustering results (in Decision-Making Using Machine Learning Basics). Here we'll show how it can be used to create boxplots, histograms, and Pareto charts.

The following Python command will import this library and label this as `plt` in the Python code:

```
import matplotlib.pyplot as plt
```

IMPORT COMMAND

The import commands allow the user to create a shortened reference to the command, so in the preceding example, the shortened reference `plt` can be used to refer to the `Matplotlib` package.

Once the import command is executed, a boxplot can be created in Python using the following syntax:

```
plt.boxplot([minimum, q1, median, q3, maximum], vert=False, labels = ['specific
label'], widths = 0.1)
```

The parameter `vert` controls whether the boxplot is to be displayed in vertical format or horizontal format.

The parameter `labels` specifies a label to be placed next to the boxplot.

The parameter `widths` specifies the width of the rectangle.

Consider a human resources manager who collects the following salary data for a sample of employees at a company (data in US dollars):

42000, 43500, 47900, 52375, 54000, 56125, 58350, 62429, 65000, 71325, 79428, 85984, 92000, 101860, 119432, 139450, 140000

The following is the **five-number summary** for this dataset and the corresponding boxplot showing the distribution of salaries at a company (refer to Descriptive Statistics: Statistical Measurements and Probability Distributions for details on how to calculate the median and quartiles of a dataset):

Five-number summary for salaries at a company:

$$\begin{aligned} \text{Minimum Salary} &= \$42000 \\ \text{First Quartile Salary} &= \$54000 \\ \text{Median Salary} &= \$65000 \\ \text{Third Quartile Salary} &= \$92000 \\ \text{Maximum Salary} &= \$140000 \end{aligned}$$

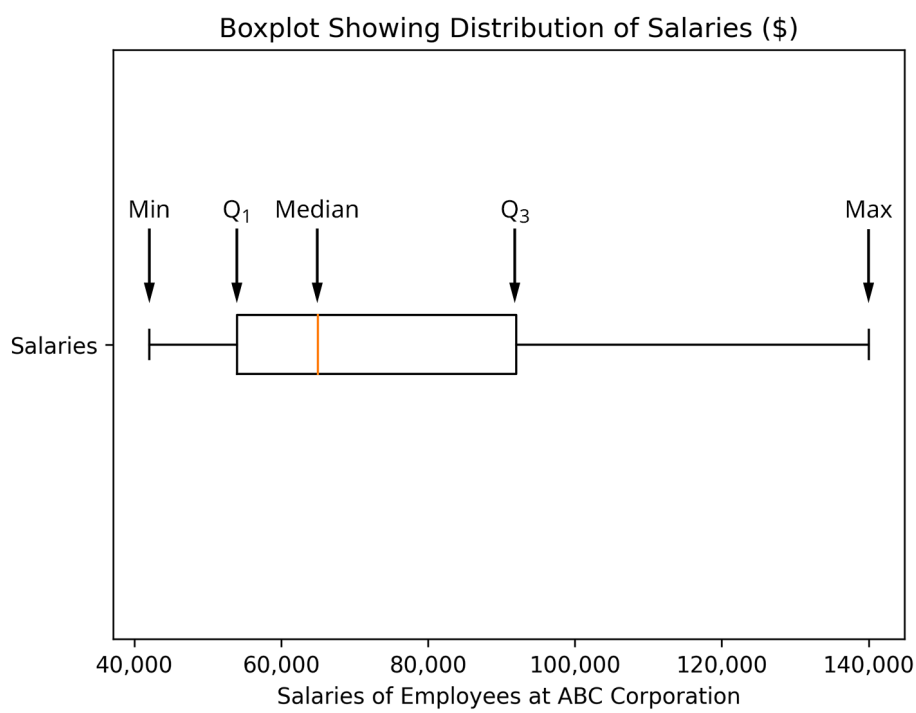
The following Python code will generate the boxplot of salaries in a horizontal format.

PYTHON CODE

```
# import matplotlib

import matplotlib.pyplot as plt
```

```
import matplotlib.ticker as ticker

# Given values for the five-number summary
minimum = 42000
q1 = 54000
median = 65000
q3 = 92000
maximum = 140000
# Define a function to format the ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'

# Create a boxplot using the given values
plt.boxplot([[minimum, q1, median, q3, maximum]], vert=False, labels=['Salaries'],
widths=0.1)

# Add labels and title
plt.xlabel('Salaries of Employees at ABC Corporation')
plt.title('Boxplot Showing Distribution of Salaries ($)')

# Apply the custom formatter to the x-axis
plt.gca().xaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))

# Show the plot
plt.show()
```

The resulting output will look like this:



Notice in the boxplot above the left edge and right edge of the rectangle represent the first quartile and third

quartile, respectively. Recall from Measurements of Position that the **interquartile range (IQR)** is calculated as the third quartile less the first quartile, and thus the IQR is represented by the width of the rectangle shown in the boxplot. In this example, the first quartile is $54,000 and the third quartile is $92,000, and the IQR is thus $\$92{,}000 - \$54{,}000$, which is $38,000.The vertical line shown within the rectangle indicates the median, and in this example, the median salary is $65,000. Finally, the whisker on the left side of the boxplot indicates the minimum of the dataset and the whisker on the right side of the boxplot represents the maximum of the dataset. In this example, the minimum salary at the company is $42,000 and the maximum salary is $140,000.

Notice from the boxplot that the vertical line within the box is not centered within the rectangle; this indicates that the distribution is not symmetric. Also, notice that the whisker on the right side of the rectangle that extends from Q_3 to the maximum is longer than the whisker on the left side of the rectangle that extends from Q_1 to the minimum; this gives an indication of a skewed distribution (right-skew).

Example 9.1 provides an example of this, using the same Python command to generate a boxplot in a vertical format.

EXAMPLE 9.1

Problem

Create Python code to generate a boxplot in a vertical format for ages of Major League baseball players based on the following dataset: SOCR-small.csv (https://openstax.org/r/socr), provided in Table 9.1. (This dataset was initially introduced in What Are Data and Data Science?.)

Name	Team	Position	Height (Inches)	Weight (Pounds)	Age (Years)
Paul_McAnulty	SD	Outfielder	70	220	26.01
Terrmel_Sledge	SD	Outfielder	72	185	29.95
Jack_Cust	SD	Outfielder	73	231	28.12
Jose_Cruz_Jr.	SD	Outfielder	72	210	32.87
Russell_Branyan	SD	Outfielder	75	195	31.2
Mike_Cameron	SD	Outfielder	74	200	34.14
Brian_Giles	SD	Outfielder	70	205	36.11
Mike_Thompson	SD	Pitcher	76	200	26.31
Clay_Hensley	SD	Pitcher	71	190	27.50
Chris_Young	SD	Pitcher	82	250	27.77
Greg_Maddux	SD	Pitcher	72	185	40.88
Jake_Peavy	SD	Pitcher	73	180	25.75

Table 9.1 Baseball Player Dataset (SOCR-small.csv)
source: http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights

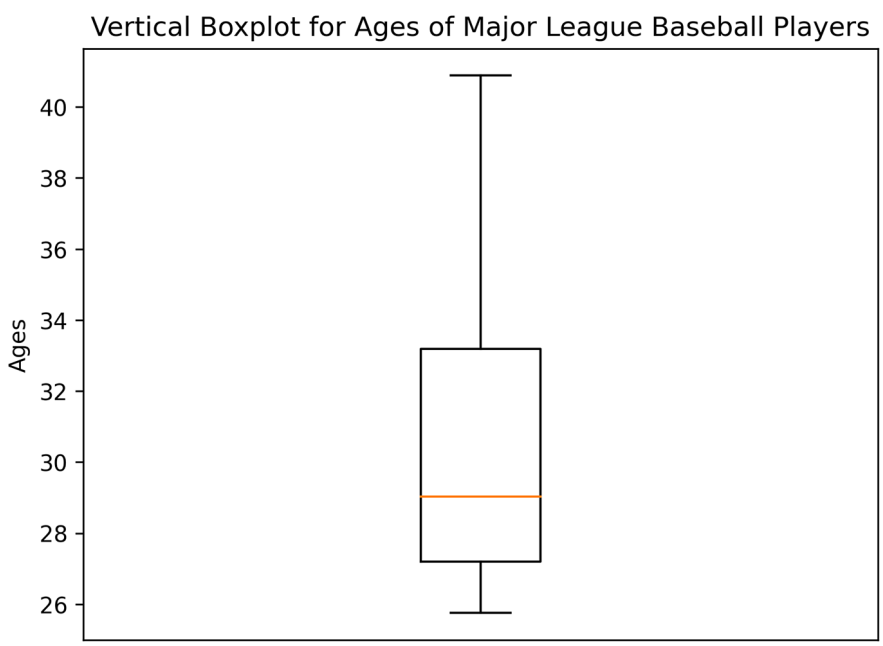
Solution

Here is the Python code to generate a vertical boxplot for the ages of a sample of baseball players. (Note: In order to generate a boxplot using a vertical format, set the parameter `vert` to be TRUE.)

PYTHON CODE

```
import matplotlib.pyplot as plt

# Sample data
ages = [26.01, 29.95, 28.12, 32.87, 31.2, 34.14, 36.11, 26.31, 27.5, 27.77,
40.88, 25.75]

# Creating the boxplot
plt.boxplot(ages, vert="TRUE")

# Adding title and labels
plt.title('Vertical Boxplot for Ages of Major League Baseball Players')
plt.ylabel('Ages')

# Display the plot
plt.show()
```

The resulting output will look like this:



Notice in the boxplot, the horizontal line representing the median is not centered within the rectangle, which indicates the distribution is not symmetric. Also, notice the whisker running vertically from the top of the rectangle is longer than the whisker running vertically from the bottom of the rectangle, and this gives an indication of a skewed distribution (right-skew).

The following example illustrates the generation of a side-by-side boxplot using Python.

EXAMPLE 9.2

Problem

A real estate agent would like to compare the distribution of home prices in San Francisco, California, versus San Jose, California. The agent collects data for a sample of recently sold homes as follows (housing prices are in U.S. $1000s).

San Francisco Housing Prices:

1250, 1050, 972, 1479, 1550, 1000, 1499, 1140, 1388, 1050, 1465

San Jose Housing Prices:

615, 712, 879, 992, 1125, 1300, 1305, 1322, 1498, 1510, 1623

Solution

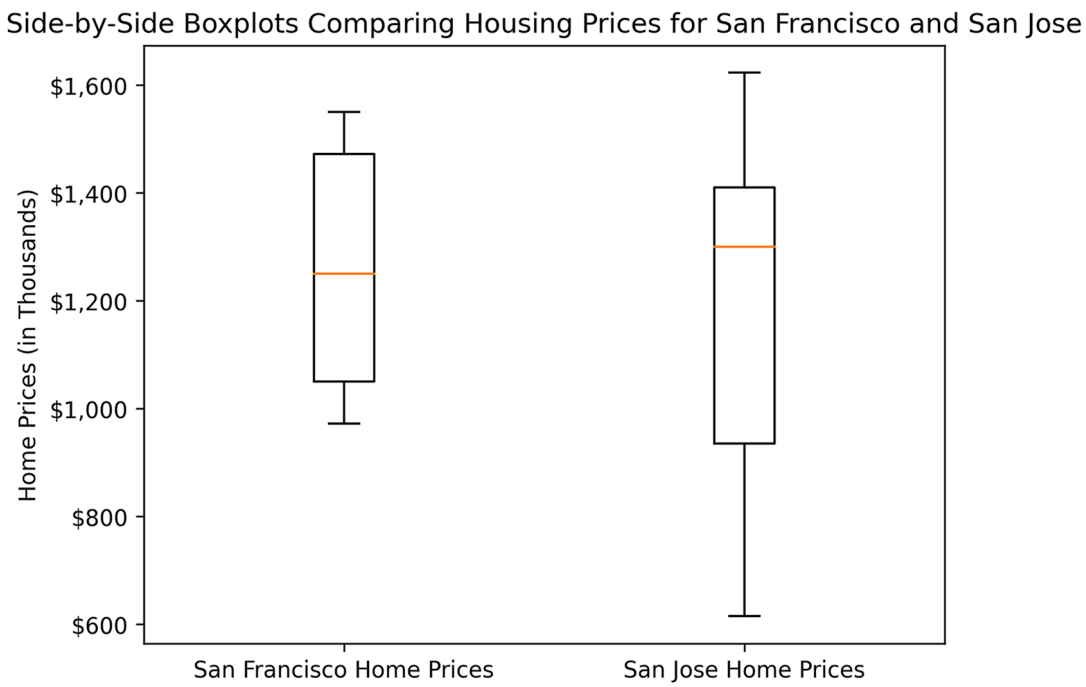
The Python code in this example uses the `boxplot` routine, which is part of the `Matplotlib` library:

PYTHON CODE

```python
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Housing prices for two cities (data in thousands)
SanFran_prices = [1250, 1050, 972, 1479, 1550, 1000, 1499, 1140, 1388, 1050,
1465]
SanJose_prices = [615, 712, 879, 992, 1125, 1300, 1305, 1322, 1498, 1510, 1623]

# Put data into an array
home_prices = [SanFran_prices, SanJose_prices]

# Create the figure and axis
fig, ax = plt.subplots()

# Create the boxplot using boxplot routine
ax.boxplot(home_prices, labels=['San Francisco Home Prices', 'San Jose Home
Prices'])

# Add titles and labels
plt.title('Side-by-Side Boxplots Comparing Housing Prices for San Francisco and
San Jose')
plt.ylabel('Home Prices (in Thousands)')

# Define a function to format the ticks for the y-axis to include dollar signs
def format_ticks(value, tick_number):
    return f'${value:,.0f}'
```

```
# Apply the custom formatter to the y-axis
ax.yaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))
ax.set_ylabel('Home Prices (in Thousands)')

# Show the plot
plt.show()
```

The resulting output will look like this:



Notice in the boxplot for San Franscico housing prices, the distribution appears symmetric, and the median housing price appears to be approximately $1,250. The median home price for San Jose appears to be approximately $1,200. The boxplot for San Jose home prices indicates more dispersion as compared to the boxplot for San Francisco home prices.

Histograms

A **histogram** is a graphical representation of the distribution of numerical data. Histograms are particularly useful for displaying the distribution of continuous data; they provide the visualization of a frequency (or relative frequency) distribution table.

For example, Figure 9.2 displays the distribution of heights (in inches to the nearest half-inch) of 100 semiprofessional male soccer players. In this histogram, heights are plotted on the horizontal axis and relative frequency of occurrence is posted on the vertical axis. Notice the tallest bar in the histogram corresponds to heights between 65.95 and 67.95 inches.

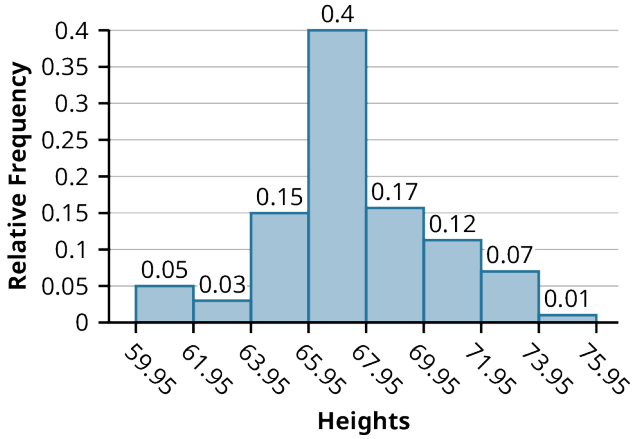


Figure 9.2 Histogram Displaying the Distribution of Heights (in inches to the nearest half-inch) of 100 Male Semiprofessional Soccer Players

Histograms are widely used in exploratory data analysis and descriptive statistics to gain insights into the distribution of numerical data, identify patterns, detect outliers, and make comparisons between different datasets.

A histogram is similar to a bar chart except that a histogram will not have any gaps between the bars whereas a bar chart typically has gaps between the bars. A histogram is constructed as a series of continuous intervals where the height of each bar represents a frequency or count of the number of data points falling within each interval. The vertical axis in a histogram is typically frequency (count) or relative frequency. and the horizontal axis is based on specific intervals for the data (called bins).

The range of values in the dataset is divided into intervals, also known as **bins**. These bins are usually of equal width and are created after an examination of the minimum and maximum of a dataset. There is no standard rule as to how many bins should be used, but a rule of thumb is to set up the number of bins to approximate the square root of the number of data points. For example, if there are 500 data values, then a researcher might decide to use approximately 22 bins to organize the data. Generally, the larger the dataset, the more bins are employed. A data scientist should experiment with different numbers of bins and then determine if the resulting histogram provides a reasonable visualization to represent the distribution of the variable.

The width of each bin can be determined using a simple formula based on the maximum and minimum of the dataset:

$$\text{Width of Each Bin } = \frac{\text{max } - \text{min}}{\text{number of bins}}$$

Like the boxplot, a histogram can provide a visual summary of the distribution of the data, including information about central tendency, spread, skewness, and presence of outliers.

The shape of a histogram can provide characteristics of the underlying data distribution. If the histogram shows the heights of bars where the tallest bar is in the center of the histogram and the bars decrease in height to the left and right of center, this is an indication of a bell-shaped distribution—also called a **normal distribution** (see Figure 9.3).

If the histogram shows taller bars on the left side or taller bars on the right side, this is indicative of a skewed distribution. Taller bars on the left side implies a longer tail on the right side of the histogram, which then indicates a *skewed right distribution*, whereas taller bars on the right side of the histogram implies a longer tail on the left side of the histogram, which then indicates a *skewed left distribution*.

Examples of skewed distributions are shown in Figure 9.4 and Figure 9.5.

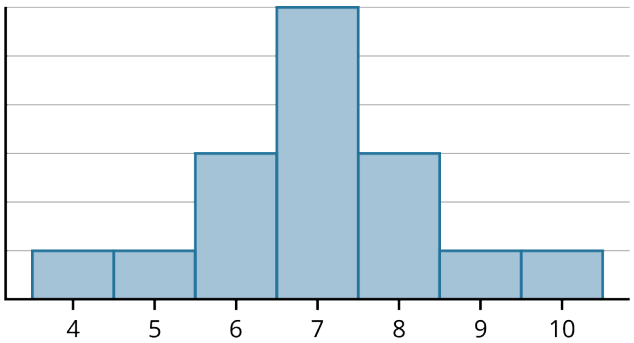


Figure 9.3 Bell-Shaped Histogram

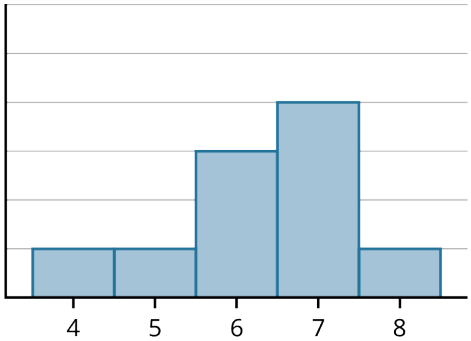


Figure 9.4 Skewed Left Histogram

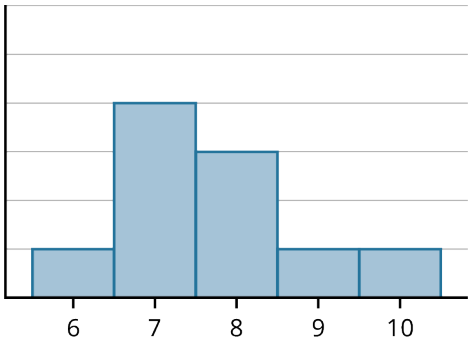


Figure 9.5 Skewed Right Histogram

EXPLORING FURTHER

Determining a Skewed or Symmetric Distribution with the Mean and Median

Another method to determine skewed versus bell-shaped (symmetric) distributions is through the use of the mean and median. In a symmetric distribution, we expect the mean and median to be approximately the same. In a right skewed distribution, the mean is typically greater than the median. In a left skewed distribution, the mean is typically less than the median. This Briefed by Data website (https://openstax.org/r/briefedbydata) provides animations and more detail on shapes of distributions, including examples of skewed distributions.

A histogram with multiple peaks is indicative of a bimodal or multimodal distribution—which indicates the possibility of subpopulations within the dataset.

A histogram showing bars that are all at the same approximate height is indicative of a uniform distribution, where the data points are evenly distributed across the range of values.

To create a histogram, follow these steps:

1. Decide on the number of bins.
2. Calculate the width of each bin.
3. Set up a frequency distribution table with two columns: the first column shows the interval for each bin and the second column is the frequency for that interval, which is the count of the number of data values that fall within the given interval for each row in the table.
4. Create the histogram by plotting the bin intervals on the *x*-axis and creating bars whose heights are determined by the frequency for each row in the table.

Using Python to Create Histograms

Typically, histograms are generated using some form of technology. Python includes a function called `hist()` as part of the `Matplotlib` library to generate histograms, as demonstrated in Example 9.3.

EXAMPLE 9.3

Problem

A corporate manager is analyzing the following monthly sales data for 30 salespeople (with sales amounts in U.S. dollars):

4969, 4092, 2277, 4381, 3675, 6134, 4490, 6381, 6849, 3134, 7174, 2809, 6057, 7501, 4745, 3415, 2174, 4570, 5957, 5999, 3452, 3390, 3872, 4491, 4670, 5288, 5210, 5934, 6832, 4933

Generate a histogram using Python for this data and comment on the shape of the distribution. Use 5 bins for the histogram.

Solution

The function `hist()`, which is part of the `Matplotlib` library, can be used to generate the histogram.

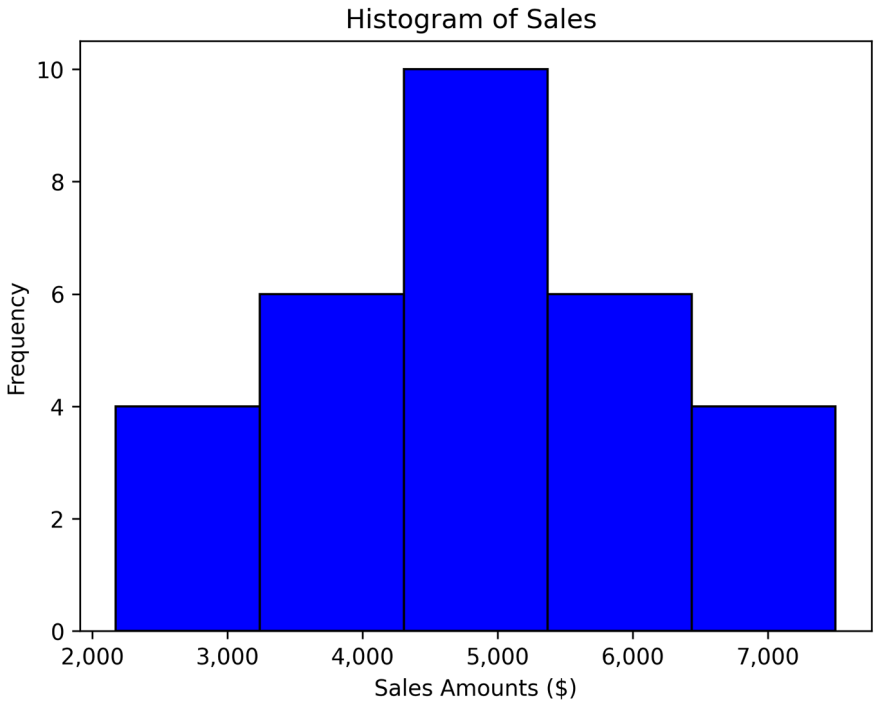
The Python program uses the `plot.hist()` function, and the data array is specified as well as the number of bins.

PYTHON CODE

```python
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Define a function to format the ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'


# Dataset of sales amounts
sales = [4969, 4092, 2277, 4381, 3675, 6134, 4490, 6381, 6849, 3134, 7174, 2809,
6057, 7501, 4745, 3415, 2174, 4570, 5957, 5999, 3452, 3390, 3872, 4491, 4670,
5288, 5210, 5934, 6832, 4933]
```

```
# Create histogram
# specify 5 bins to be used when creating the histogram
plt.hist(sales, bins=5, color='blue', edgecolor='black')

# Add labels and title
plt.xlabel('Sales Amounts ($)')
plt.ylabel('Frequency')
plt.title('Histogram of Sales ')

# Display the histogram
plt.show()
```

The resulting output will look like this:



Notice that the shape of the histogram appears bell-shaped and symmetric, in that the tallest bar is in the center of the histogram and the bars decrease in height to the left and right of center.

Pareto Charts

A **Pareto chart** is a bar graph for categorical data where the taller bars are on the left side of the chart and the bars are placed in a descending height from the left side of the chart to the right side of the chart.

A Pareto chart is unique in that it combines both bar and line graphs to represent data in descending order of frequency or importance, along with the cumulative percentage of the total. The chart is named after Vilfredo Pareto, an Italian economist who identified the 80/20 rule, which states that roughly 80% of effects come from 20% of causes. This type of chart is commonly used in quality improvement efforts where a researcher would like to quickly identify the major contributors to a problem.

When creating a Pareto chart, two vertical scales are typically employed. The left vertical scale is based on frequency, whereas the right vertical scale is based on cumulative percentage. The left vertical scale corresponds to the bar graph and the bars are then oriented in descending order according to frequency, so the height of the bars will decrease from left to right. The right vertical scale is used to show cumulative frequency for the various categories and corresponds to the line chart on the graph. The **line chart** allows the reader to quickly determine which categories contribute significantly to the overall percentage. For example, the line chart makes it easier to determine which categories make up 80% of the cumulative frequency, as shown in Example 9.4.

EXAMPLE 9.4

Problem

A manufacturing engineer is interested in analyzing a high defect level on a smartphone manufacturing line and collects the following defect data over a one-month period (see Table 9.2).

Failure Category	Frequency (Number of Defects)
Cracked screen	1348
Power button nonfunctional	739
Scratches on case	1543
Does not charge	1410
Volume controls nonfunctional	595

Table 9.2 Smartphone Defect Data

Create a Pareto chart for these failure categories using Python.

Solution

We can use pandas (https://openstax.org/r/pydata1) to create a DataFrame to store the failure categories and number of defects. Then we will calculate the cumulative frequency percentages for display on the Pareto chart.

The DataFrame is sorted in descending order of number of defects. The cumulative percentage is calculated using the Python `cusum()` function.

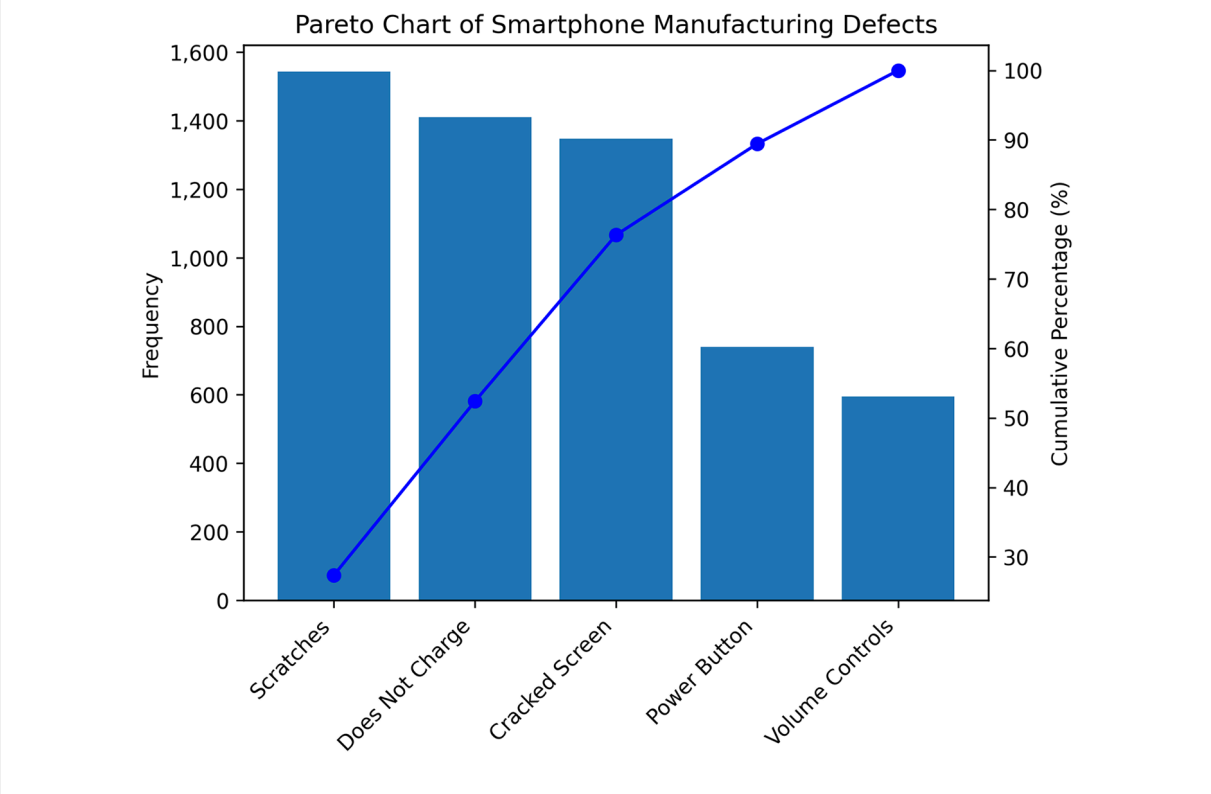
The reference to `ax1` is the primary *y*-axis for frequency (i.e., left-hand *y*-axis). The reference to `ax2` is the *y*-axis, which will plot the cumulative frequency (right-hand *y*-axis). The reference to the `twinx` function is a method in matplotlib that allows the two *y*-axis scales to share the same *x*-axis.

PYTHON CODE

```
import pandas as pd
import matplotlib.pyplot as plt

# Manufacturing data results for one-month time period
```

```
data = {
    'Failure_Category': ['Cracked Screen', 'Power Button', 'Scratches', 'Does Not
Charge', 'Volume Controls'],
    'Frequency': [1348, 739, 1543, 1410, 595]
}

# Create a DataFrame from the data
df = pd.DataFrame(data)

# Sort the DataFrame in descending order based on the frequency
df_sorted = df.sort_values(by='Frequency', ascending=False)

# Calculate the cumulative percentage
df_sorted['Cumulative Percentage'] = (df_sorted['Frequency'].cumsum() /
df_sorted['Frequency'].sum()) * 100

# Create the Pareto chart
fig, ax1 = plt.subplots()

# Plot the bars in descending order
ax1.bar(df_sorted['Failure_Category'], df_sorted['Frequency'])
ax1.set_ylabel('Frequency')

# Plot the cumulative frequency line chart
ax2 = ax1.twinx()
ax2.plot(df_sorted['Failure_Category'], df_sorted['Cumulative Percentage'],
color='blue', marker='o')
ax2.set_ylabel('Cumulative Percentage (%)')

# Set x-axis labels
ax1.set_xticklabels(df_sorted['Failure_Category'], rotation=45, ha='right')

# Title
plt.title('Pareto Chart of Smartphone Manufacturing Defects')

# Show the plot
plt.show()
```

The resulting output will look like this:



Notice in the Pareto output chart that the leftmost three bars account for approximately 75% of the overall defect level, so a quality engineer can quickly determine that the three categories of "Scratches," "Does Not Charge," and "Cracked Screen" account for the majority of the defects and would then be the defects to focus on for quality improvement efforts.

According to the Pareto chart that is created, the three defect categories of "Scratches," "Does Not Charge," and "Cracked Screen" are major contributors to the defect level and should be investigated with higher priority. The two defect categories of "Power Button" and "Volume Controls" are not major contributors and should be investigated with lower priority.

9.2 Encoding Data That Change Over Time

Learning Outcomes

By the end of this section, you should be able to:

- 9.2.1 Create and interpret labeled graphs to visually identify trends in data over time.
- 9.2.2 Use Python to generate various data visualizations for time series data.

Researchers are frequently interested in analyzing and interpreting data over time. Recall from Time and Series Forecasting that any set of data that consists of numerical measurements of the same variable collected and organized according to regular time intervals may be regarded as *time series* data. A time series graph allows data scientists to visually identify trends in data over time. For example, a college administrator is very interested in tracking enrollment data from semester to semester. A marketing manager is interested in tracking quarterly revenue for a business. An investor might be interested in tracking stock prices on a monthly basis.

In many cases, a time series graph can reveal seasonal patterns to cycles, which is important information to

help guide many business decisions, such as planning marketing campaigns or setting optimal inventory levels. Time series graphs also provide important data for forecasting methods, which attempts to predict future trends based on past data.

Refer to Examples of Time Series Data for the basics of visualizing time series data in Excel and Python. The Trend Curve and Trend-Cycle Component explores visualization of a trend curve. Here we'll explore line charts, trend curves, and the use of Python for time series data visualization in more detail.

Line Charts and Trend Curves

A time series graph is used in many fields such as finance, health care, economics, etc. to visualize data measurements over time. To construct a time series graph, the horizontal axis is used to plot the time increments, and the vertical axis is used to plot the values of the variable that we are measuring, such as revenue, enrollment, inventory level, etc. By doing this, we make each point on the graph correspond to a point in time and a measured quantity. The points on the graph are typically connected by straight lines in the order in which they occur. Once the chart is created, a researcher can identify trends or patterns in the data to help guide decision-making.

EXAMPLE 9.5

Problem

The following data plots the enrollment at a community college over an eight-year time period. Create a times series graph of the data and comment on any trends (Table 9.3).

Year	Enrollment
1	8641
2	9105
3	9361
4	9895
5	9230
6	8734
7	8104
8	7743

Table 9.3 Community College Enrollment

Solution

To create a time series graph for this dataset, start off by creating the horizontal axis by plotting the time increments, from years 1 to 8. For the vertical axis, used a numeric scale extending from 4,000 to 12,000 since this covers the span of the numeric enrollment data.

Then for each row in the table, plot the corresponding (x, y) point on the graph. By doing this, we make each point on the graph correspond to a point in time and a measured quantity. For example, the first row in the table shows the (x, y) datapoint of (1, 8641). To plot this point, identify the location that corresponds to Year 1 on the horizontal axis and an enrollment value of 8641 on the vertical axis. At the intersection of the horizontal location and vertical location, plot a single point to represent this (x, y) data value. Repeat this process for each row in the table. Since there are eight rows in the table, we expect to see eight points on the time series graph. Finally, the points on the graph are typically connected by straight lines in the

order in which they occur (see the Python output from Example 9.6).

Using Python for Times Series Data Visualization

Data visualization is a very important part of data science, and Python has many built-in graphing capabilities through a package called `matplotlib`. To import this package into a Python program, use the command:

```
import matplotlib.pyplot as plt
```

`Matplotlib` contains functions such as `plot` that can be used to generate time series graphs.

After defining the (x, y) data to be graphed, the Python function `plt.plot(x, y)` can be used to generate the time series chart.

EXAMPLE 9.6

Problem

Use the `plot` function in Python to create a time series graph for the dataset shown in Example 9.5.

Solution

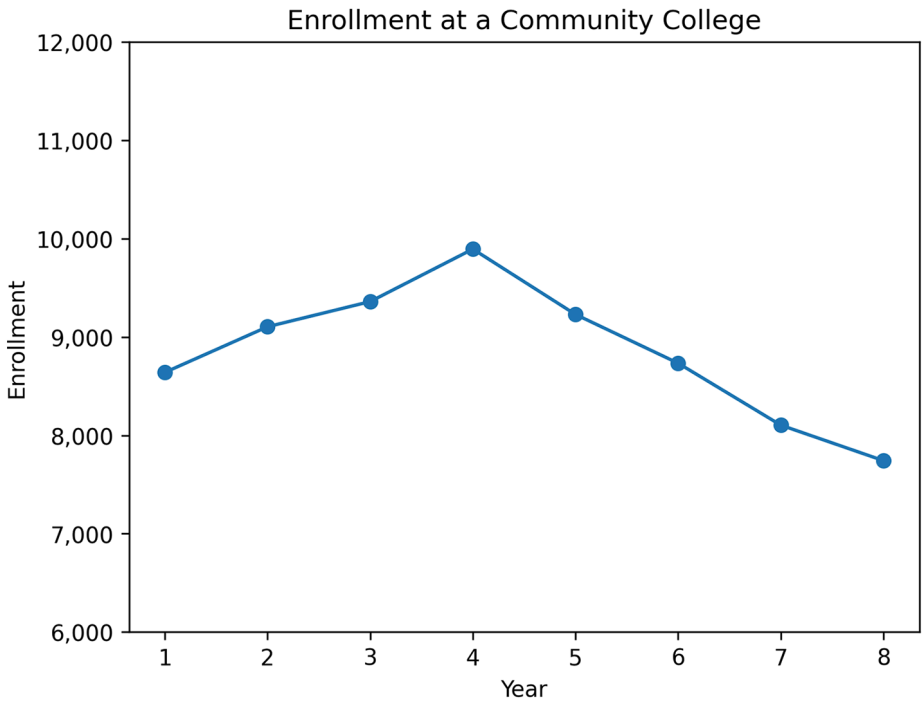
In the following Python code, note that the `plt.plot` with marker specifies that points should be plotted at the various (x, y) points. In addition, `plt.ylim` specifies the range for the *y*-axis.

PYTHON CODE

```python
# import the matplotlib library
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Define the x-data
x = [1, 2, 3, 4, 5, 6, 7, 8]

# Define the y-data
y = [8641, 9105, 9361, 9895, 9230, 8734, 8104, 7743]

# Use the plot function to generate a time series graph
plt.plot(x, y, marker='o', linestyle='-')

# Set the scale for the y-axis
plt.ylim(6000, 11000)  # This sets the y-axis range from 6000 to 11000

# Define a function to format the ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'
```

```
# Apply the custom formatter to the y-axis
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))

# Add labels to the x and y axes by using xlabel and ylabel functions
plt.xlabel("Year")
plt.ylabel("Enrollment")

# Add a title using the title function
plt.title("Enrollment at a Community College")

# Show the plot
plt.show()
```

The resulting output will look like this:



The graph indicates that the enrollment gradually increased over years 1 to 4 but then began to decrease from years 4 to 8.

Guidelines for Creating Effective Visualizations

You have likely come across poorly constructed graphs that are difficult to interpret or possibly misleading in some way. However, if you follow certain guidelines when creating your visualizations, you will be sure your visualizations are effective and represent the data fairly.

The first rule is to ensure your graph or chart is clear and easy to understand. You may be familiar with the type of data collected, but your audience may not be. Provide a title and label each axis, including units where appropriate. Provide a data source where appropriate. Use scales for the axes and avoid distortion (i.e., don't

intentionally adjust the scale to produce a desired effect in the graph).

The use of colors to differentiate data can be an important part of graphs and displays. Overall, the use of color can make the visualization more appealing and engaging for the viewer. However, there are some important things to keep in mind when using color. Be sure to apply color consistently to represent the same categories. Ensure that there is contrast between different colors to make them easily distinguishable. For example, if you used various shades of blue for the sectors in a pie chart, it might be difficult for the reader to distinguish among them.

Keep in mind that some viewers may be color-blind, so it's a good rule of thumb to use color palettes that are accessible to color-blind individuals. The most basic rule for a color-blind palette is to avoid combining green and red.

EXPLORING FURTHER

Resources for Creating Color-Blind Palettes

- The Dataviz Best Practices blog includes tips on creating The Best Charts for Colorblind Viewers (https://openstax.org/r/datylon).
- 10 Essential Guidelines for Colorblind Friendly Design (https://openstax.org/r/colorblindguide) offers more information on types of color-blindness and ways to make your visual design accessible for and appealing to color-blind users.

You should also be aware that colors can have different meanings in different cultures. Keep the number of colors used in your visual to a minimum. One rule of thumb is to limit the number of colors to three to five to avoid a color palette that is too confusing.

Create the graph with your audience in mind. If the graph is intended for a general audience, then the graph should be as simple as possible and easy to interpret, perhaps with pie charts or bar charts. Don't include too many categories in your pie chart; if you have more than about five or six categories, you may want to consider a different type of chart. Consider combining categories where appropriate for simplicity. (Sometimes an "other" category can be used to combine categories with very few counts.) If the graph is intended for a more technical audience or for more complicated types of analyses, then a more detailed display such as a scatterplot with regression line and confidence bounds may be appropriate. (Visualizing data with scatterplots is covered in Multivariate and Network Data Visualization Using Python.)

Selecting the Best Type of Visualization

When creating graphs and displays to visualize data, the data scientist must consider what type of graph will work best with the collected data and will best convey the message that the data reveals. This section provides some guidance on the optimal type of graph for different scenarios.

Before we discuss how to choose the best visualization, it is important to consider the type of data being presented. Recall that data can be divided into the broad categories of *qualitative data* (describes categories or groups) and *quantitative data* (numeric data) and that quantitative data is further classified as *discrete* (usually counts) and *continuous* (usually measurements that can take on any value within a range). You first need to be clear about the type of data being illustrated.

Then, carefully consider the goal of the visualization—that is, what are you trying to accomplish with the graph or display? What is your data telling you? How is it distributed, and what patterns or relationships do you want to highlight? You might be interested in comparing different groups of categories, or you may be interested in showing how data is distributed over some interval. Recall from the previous discussion of correlation and regression that we are often interested in showing correlations or relationships between quantitative

variables. Table 9.4 offers some suggestions for picking the most appropriate graphing method based on your goals and type of data.

Goal of Visualization	Type of Data	Visualization Methods to Consider
Show a comparison of different groups or categories	Qualitative or Quantitative	• Bar charts are useful for comparing different groups of categories • Column chart
Show correlations or relationships between quantitative variables	Quantitative	• Scatterplot • Time series graph • Heatmap
Show how a whole is divided into parts	Quantitative	• Pie chart
Show how data is distributed over some interval	Quantitative	• Histogram • Boxplot

Table 9.4 Summary of Visualization Methods Based on Goals and Data

9.3 Graphing Probability Distributions

Learning Outcomes

By the end of this section, you should be able to:

- 9.3.1 Create graphs to visualize the shape of various types of probability distributions.
- 9.3.2 Interpret probabilities as areas under probability distributions.
- 9.3.3 Use Python to generate various data visualizations for probability distributions

We introduced probability distributions in Discrete and Continuous Probability Distributions. Recall that the binomial distribution and the Poisson distribution are examples of *discrete probability distributions*, and the normal distribution is an example of a *continuous probability distribution*. In addition, a **discrete random variable** is a random variable where there is only a finite or countable infinite number of values that the variable can take on. A *continuous random variable* is a random variable where there is an infinite number of values that the variable can take on.

Very often, a data scientist or researcher is interested in graphing these probability distributions to visualize the shape of the distribution and gain insight into the behavior of the distribution for various values of the random variable. For continuous probability distributions, the area under the probability density function (PDF) is equivalent to probabilities associated with the normal probability distribution. Determining probabilities associated with probability distribution allows data scientists to measure and predict probabilities and to estimate the likelihood of achieving certain outcomes.These probabilities also form the basis for more advanced statistical analysis such as confidence interval determination and hypothesis testing.

Graphing the Binomial Distribution

The **binomial distribution** is used in applications where there are two possible outcomes for each trial in an experiment and the two possible outcomes can be considered as success or failure.

The necessary requirements to identify a binomial experiment and apply the binomial distribution include the following:

- The experiment of interest is repeated for a fixed number of trials, and each trial is independent of other trials.

- There are only two possible outcomes for each trial, which can be labeled as “success” or “failure.”
- The probability of success remains the same for each trial of the experiment.
- The random variable *x* counts the number of successes in the experiment. Notice that since *x* counts the number of successes, this implies that *x* is a discrete random variable.

When working with a binomial experiment, it is useful to identify two specific parameters in a binomial experiment:

1. The number of trials in the experiment. Label this as *n*.
2. The probability of success for each trial (which is a constant value). Label this as *p*.

We then count the number of successes of interest as the value of the discrete random variable. Label this as *x*.

Discrete and Continuous Probability Distributions introduced the binomial probability formula used to calculate binomial probabilities. However, most data scientists and researchers use technology such as Python to calculate probabilities associated with both discrete and continuous probability distributions such as the binomial distribution and the normal distribution.

Python provides a number of built-in functions for calculating these probabilities as part of the `scipy.stats` library.

The function `binom()` is the probability mass function used to calculate probabilities associated with the binomial distribution. The syntax for using this function is

```
binom.pmf(x, n, p)
```

Where:

n is the number of trials in the experiment,

p is the probability of success,

x is the number of successes in the experiment.

Data scientists are interested in graphing discrete distributions such as the binomial distribution to help to predict the probability of achieving a certain number of successes in a given number of trials with a specified probability of success. This also allows visualization of the shape of the binomial distribution as parameters such as *n* and *p* are varied. Changes to the value of the probability of success can affect the shape of the distribution, as shown in Example 9.7.

EXAMPLE 9.7

Problem

Use the Python function `binom` to graph the binomial distribution for $n = 20$ and three different values of p, namely $p = 0.10, 0.50$, and 0.85, and comment on the resulting shapes of the distributions.

Solution

To create a graph of the binomial distribution, we will import the `binom` function from the `scipy.stats` library. We will then use the `bar` function to create a bar chart of the binomial probabilities.

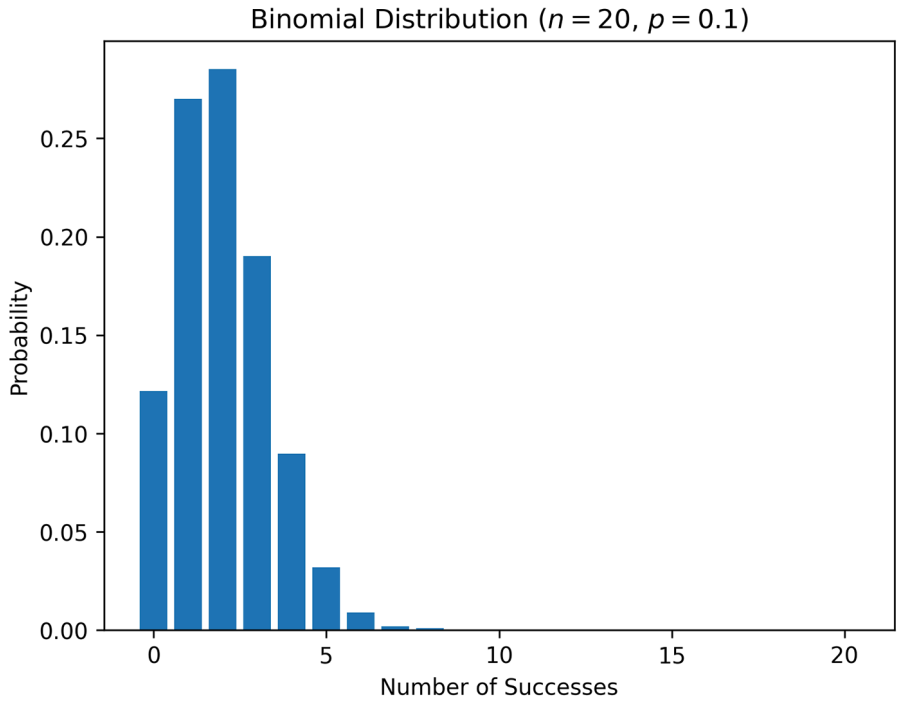
Here is the Python code and graph for $n = 20$ and $p = 0.10$.

Note: To generate the plots for $p = 0.5$ and $p = 0.85$, change the one line of Python code, $p = 0.1$, as needed.

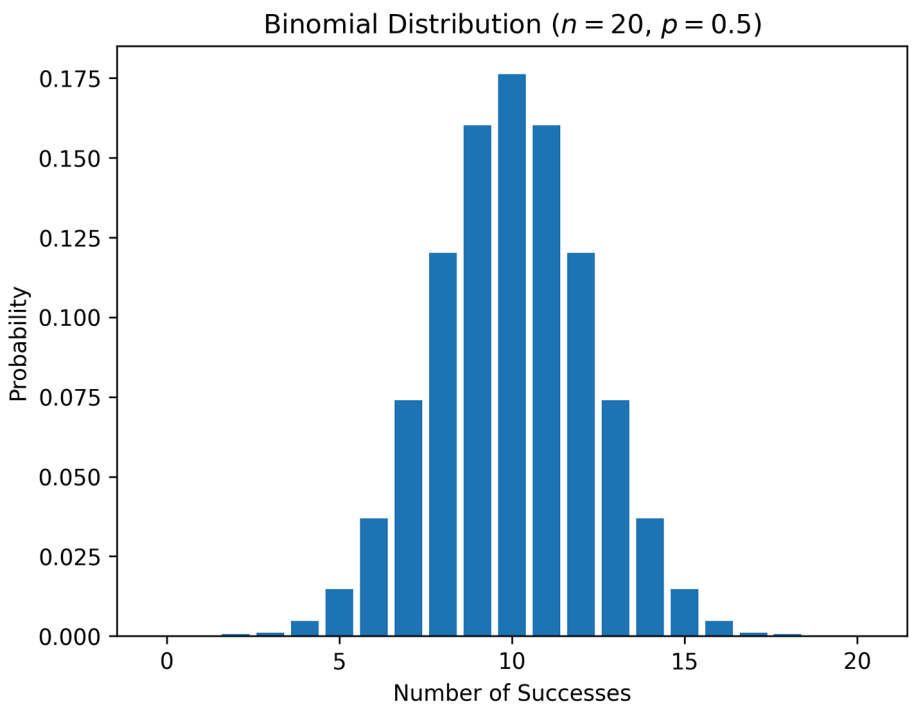
PYTHON CODE

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import binom

# Define parameters for the binomial distribution where:
# n is number of trials
# p is the probability of success
n = 20
p = 0.1

# Generate values for x, which is the number of successes
# Note that x can take on the value of zero since there can be
# zero successes in a typical binomial experiment
x = np.arange(0, n+1)

# Calculate the probability mass function (PMF) for the binomial distribution
probabilities = binom.pmf(x, n, p)

# Plot the binomial distribution
plt.bar(x, probabilities)

# Add labels and title
plt.xlabel('Number of Successes')
plt.ylabel('Probability')
plt.title('Binomial Distribution ($n = 20$, $p = 0.1$)')

# Show the plot
plt.show()
```

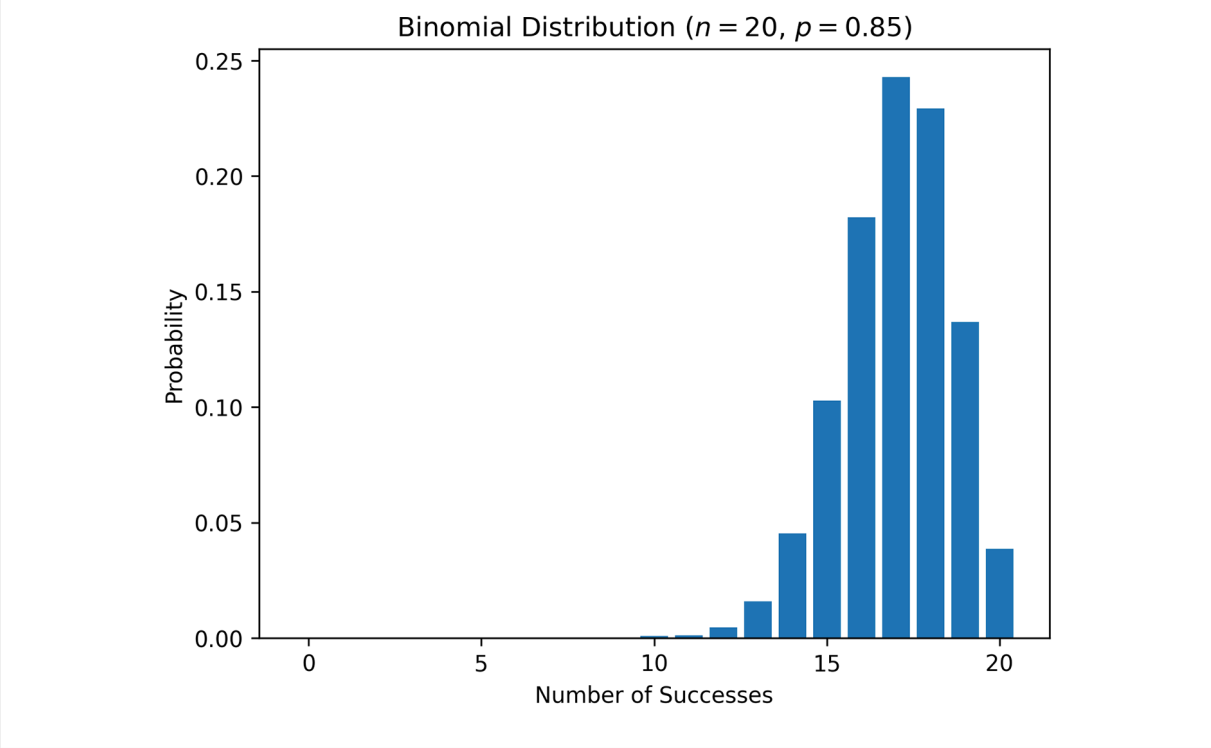
The resulting output will look like this:



Here is the graph for $n = 20$ and $p = 0.50$.



Here is the graph for $n = 20$ and $p = 0.85$.



Notice the impact of p on the shape of the graph: for $p = 0.1$, the graph is right skewed, for $p = 0.5$, the graph resembles a bell-shaped distribution, and for $p = 0.85$, the graph is left skewed.

Graphing the Poisson Distribution

The **Poisson distribution** is used when counting the number of occurrences in a certain interval. The random variable then counts the number of occurrences in that interval.

The Poisson distribution is used in the types of situations where the interest is in a specific certain number of occurrences for a random variable in a certain interval such as time or area.

Recall from Discrete and Continuous Probability Distributions that the Poisson distribution is used where the following conditions are met:

- The experiment is based on counting the number of occurrences in a specific interval where the interval could represent time, area, volume, etc.
- The number of occurrences in one specific interval is independent of the number of occurrences in a different interval.

Notice that when we count the number of occurrences of a random variable x in a specific interval, this will represent a discrete random variable.

Similar to the `binom()` function within the `scipy.stats` library, Python also provides the `poisson()` function to calculate probabilities associated with the Poisson distribution.

The syntax for using this function is:

```
poisson.pmf(x, mu)
```

Where:

mu is the mean of the Poisson distribution (*mu* refers to the Greek letter μ).

x is the number of successes in the specific interval of interest.

EXAMPLE 9.8

Problem

The Port of Miami is home to many cruise lines, and the port is typically heavily congested with arriving cruise ships. Delays are common since there are typically not enough ports to accept the number of arriving ships. A data scientist is interested in analyzing this situation and notices that on average, 3 ships arrive every hour at the port. Assume the arrival times are independent of one another.

Use the Python function `poisson` to graph the Poisson distribution for $n = 12$ hours and mean μ of 3 ships per hour.

Based on the graph, come to a conclusion regarding the number of ships arriving per hour.

Solution

To create a graph of the Poisson distribution, we will import the `poisson` function from the `scipy.stats` library. We will then use the `bar` function to create a bar chart of the corresponding Poisson probabilities.

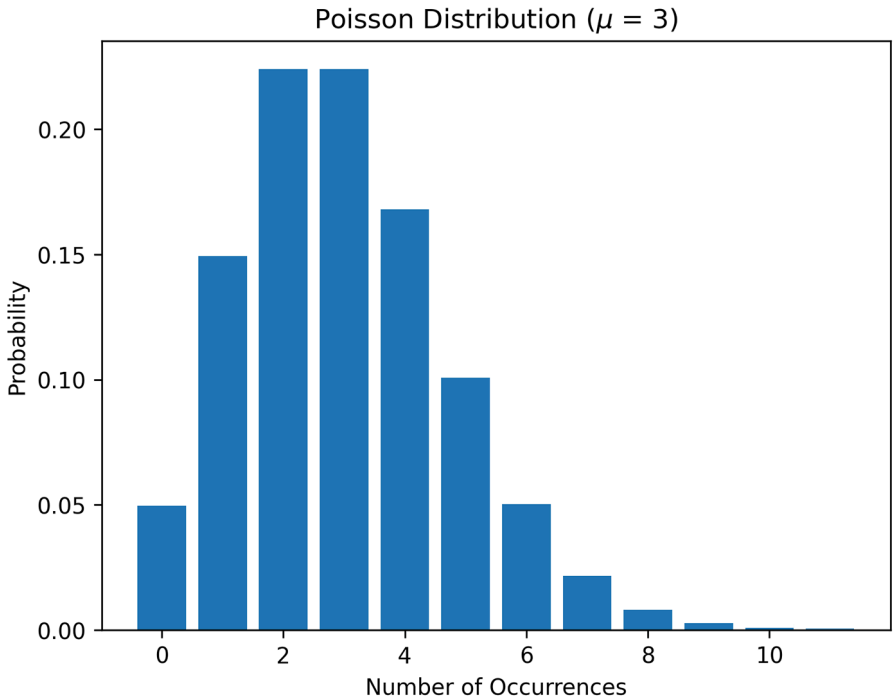
Here is the Python code and graph for $n = 12$ and mean (μ) of 3.

PYTHON CODE

```python
import matplotlib.pyplot as plt
from scipy.stats import poisson

# Define the mean for the Poisson distribution
# this is the average number of ships arriving per hour
mu = 3

# Generate x values (number of ships arriving per hour)
x = np.arange(0, 12)

# Calculate the probability mass function (PMF) for the Poisson distribution
probabilities = poisson.pmf(x, mu)

# Plot the Poisson distribution
plt.bar(x, probabilities)

# Add labels and title
plt.xlabel('Number of Occurrences')
plt.ylabel('Probability')
plt.title('Poisson Distribution ($\mu$ = 3)')

# Show the plot
```

```
plt.show()
```

The resulting output will look like this:



From the graph, there are lower probabilities for $x = 7$ and larger. So there is not much of a chance that 7 or more ships will arrive in a one-hour time period. The administrators for the Port of Miami should plan for 6 or fewer ships arriving per hour to cover the most likely scenarios.

Graphing the Normal Distribution

The normal distribution discussed in Discrete and Continuous Probability Distributions is one of the more important distributions in data science and statistical analysis. This continuous distribution is especially important in statistical analysis in that many measurements in nature, science, engineering, medicine, and business follow a normal distribution. The normal distribution also forms the basis for more advanced statistical analysis such as confidence intervals and hypothesis testing discussed in Hypothesis Testing.

The normal distribution is bell-shaped and has two parameters: the mean, μ, and the standard deviation, σ. The mean represents the center of the distribution, and the standard deviation measures the spread or dispersion of the distribution. The variable x represents the realization or observed value of the random variable X that follows a normal distribution.

The graph of the normal curve is symmetric about the mean μ. The shape of the graph of the normal distribution depends only on the mean and the standard deviation. Because the area under the curve must equal 1, a change in the standard deviation, σ, causes a change in the shape of the normal curve; the curve becomes fatter and wider or skinnier and taller depending on the standard deviation μ. A change in the mean σ causes the graph to shift to the left or right. To determine probabilities associated with the normal distribution, we find specific areas under the graph of the normal curve within certain intervals of the random variable.

To generate a graph of the normal distribution, we can plot many points that fall on the normal curve according to the formal definition of the probability density function (PDF) of the normal distribution, as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Of course, this is a job left to Python, and the Python code to generate a graph of the normal distribution is shown in Example 9.9.

EXAMPLE 9.9

Problem

A medical researcher is investigating blood pressure–lowering medications and wants to create a graph of systolic blood pressures. Assume that blood pressures follow a normal distribution with mean of 120 and standard deviation of 20. Use Python to create a graph of this normal distribution.

Solution

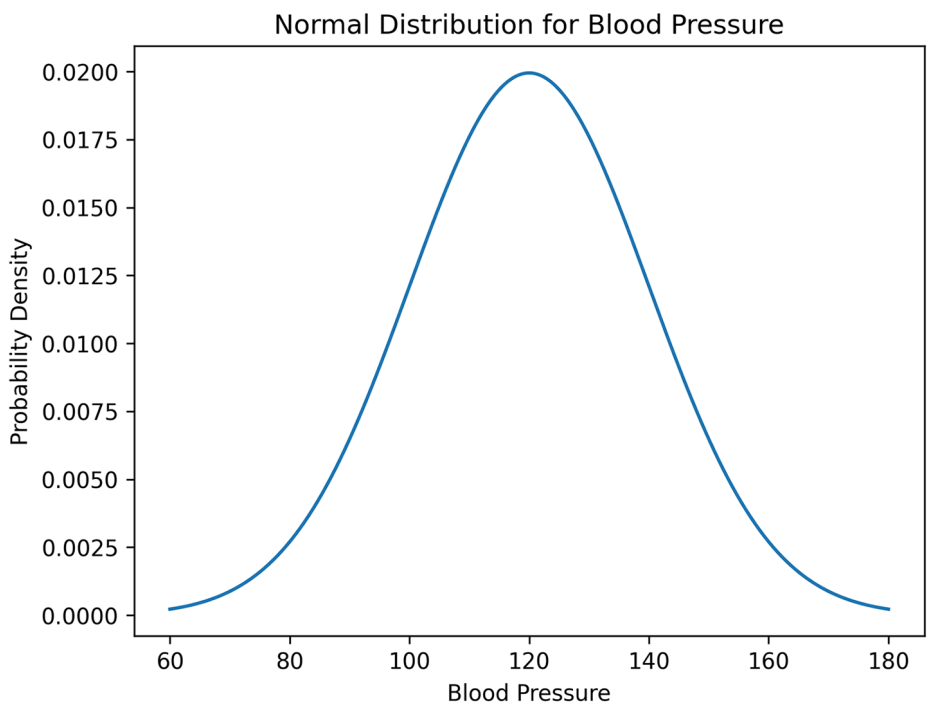
To create a graph of the normal distribution, we will plot a large number of data points (say 1,000 points), and the values of x will range from $\mu + 3\sigma$ and $\mu - 3\sigma$. Recall from the discussion in Discrete and Continuous Probability Distributions regarding the empirical rule that about 99.7% of the x-values lie between -3σ and $+3\sigma$ units from the mean μ (i.e., within three standard deviations of the mean). In Python, we can use the `np.linspace` function to create evenly spaced points between a lower bound and upper bound. Then we can use the `plot` function to plot the (x, y) points corresponding to the normal density function. Note: We also need the numpy (https://openstax.org/r/numpy) library to calculate numerical functions such as square root.

PYTHON CODE

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Define mean mu and standard deviation sigma for the normal distribution
mu = 120
sigma = 20

# Generate data points from the normal distribution

# Use np.linspace to generate 1,000 evenly spaced points
# These points will extend from mu - 3sigma to mu+3sigma bounds
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 1000)

# Calculate values of the normal density function using norm.pdf function
y = norm.pdf(x,mu,sigma)
```

```
# Plot the normal distribution
plt.plot(x, y)

# Add labels and title
plt.xlabel('Blood Pressure')
plt.ylabel('Probability Density')
plt.title('Normal Distribution for Blood Pressure')

# Show the plot
plt.show()
```

The resulting output will look like this:



9.4 Geospatial and Heatmap Data Visualization Using Python

Learning Outcomes

By the end of this section, you should be able to:

- 9.4.1 Describe the insights produced by spatial heatmaps based on geospatial data.
- 9.4.2 Discuss the common features of GIS mapping.
- 9.4.3 Use a GIS mapper to produce a visualization of geospatial data.

Geospatial data refers to data that describes the geographic location, shape, size, and other attributes relative to a location on the Earth's surface. Geospatial data can be captured, stored, manipulated, analyzed, and visualized using various technologies and tools.

Geospatial data visualization using Python involves the representation and analysis of data that has a

geographic component, such as latitude and longitude coordinates. Python offers several libraries and utilities for geospatial data visualization and analysis, including the `Pandas` library discussed in What Are Data and Data Science? `Pandas` is a Python library specialized for data manipulation and analysis. In a similar way, `Geopandas` extends the capabilities of `Pandas` to handle geospatial data. The `Geopandas` library provides the ability to read, write, analyze, and visualize geospatial data. `Geopandas` provides a GeoDataFrame object, which is similar to a `Pandas` DataFrame but includes geometry information for spatial operations.

Spatial and Grid Heatmaps

Spatial heatmaps are a data visualization method used to represent the density or intensity of data points within a geographical area using coloring and shading to represent densities of various attributes. These heatmaps provide a visual summary of the distribution and concentration of data points, highlighting areas of high and low density.

For example, a spatial heatmap of New York City might show crime rate information where higher crime rates in a certain location are represented by darker shades of red and lower crime rates are represented by lighter shades of red. The reader can quickly ascertain the differences in crime rates among various neighborhoods based on the shading in the heatmap.

Figure 9.6 provides an example of a heatmap showing differences in average temperature for January 2024 as compared to previous temperature data from 1991–2020 where the coloring scheme represents the differences (blue represents a reduction in temperature and red represents an increase in temperature).

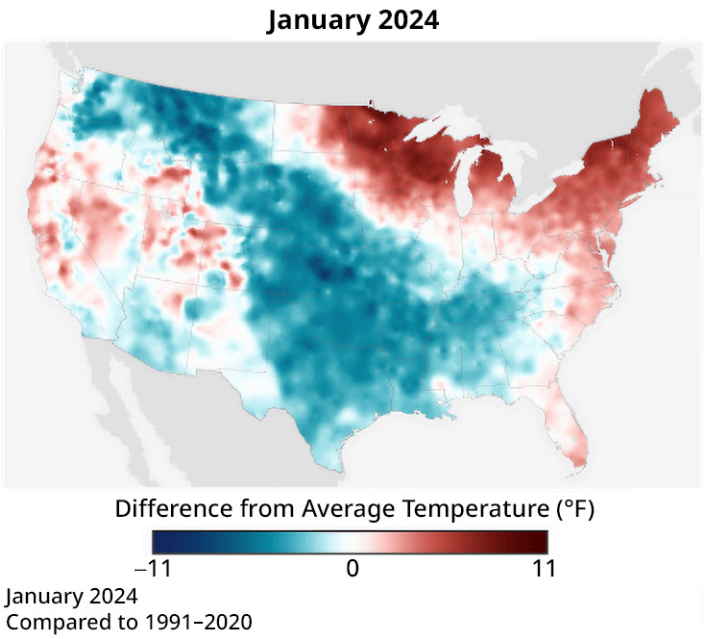


Figure 9.6 Example of Heatmap Showing Differences from Average Temperature for January 2024 by Colors (data source: NOAA Climate.gov maps based on data from NOAA National Centers for Environmental Information. [NCEI] https://www.climate.gov/media/15930, accessed June 23, 2024.)

Spatial heatmaps are generated by condensing the spatial coordinates of individual data points into a grid or a set of bins covering the geographic area of interest. The density of data points within each grid cell or bin is then calculated, and these density results are then mapped to specific colors using a color gradient map. Geographic areas with higher density values are assigned colors that represent higher intensity, while areas with lower density values are assigned colors that represent lower intensity. Finally, the colored grid cells can be overlaid on a map, and the result is a visual representation of the spatial distribution of data points.

Grid heatmaps display colors in a two-dimensional array where the x-axis and *y*-axis represent two differing characteristics and the color coding for a certain cell is based on the combined characteristics for that cell. For

example, in Figure 9.7, a matrix is created where the *x*-axis represents different local farmers and the *y*-axis represents harvest for various crops (in tons/year), and then each cell is coded based on the harvest value for that combination of farmer and specific crop. At a glance, the viewer can discern that the highest harvests appear to be for potatoes at BioGoods Ltd and for barley at Cornylee Corp. Note that in a grid heatmap, the grids are typically based on a fixed size since the intent is to detect clustering for the characteristics of interest. Also note that the color scheme used is where green and yellow shading represent higher harvest levels and blue and purple represent lower harvest levels.

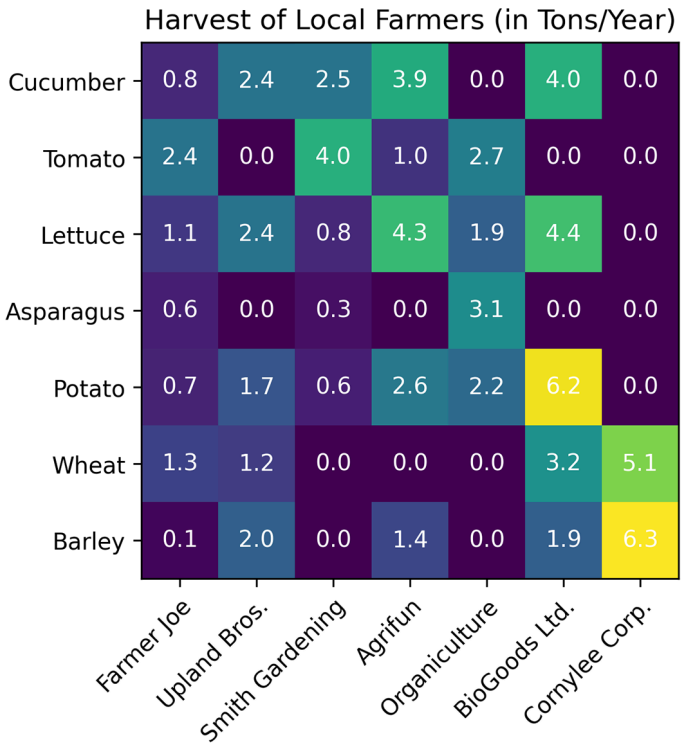


Figure 9.7 Grid Heatmap Showing Color Density for Crop Harvest by Local Farmer and Type of Crop (source: created by author using Matplotlib by D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007. DOI: https://zenodo.org/records/13308876)

EXPLORING FURTHER

Interactive Heatmaps

Various interactive heatmaps are available that allow the user to preselect variables of interest and then view the corresponding geospatial map or heatmap. For example, see these interactive heatmaps utilizing crash and demographic data as well as traffic safety improvement activities in California:

https://catsip.berkeley.edu/resources/crash-data/crash-data-tools-and-resources (https://openstax.org/r/berkeley)

https://safetyheatmap.berkeley.edu/map/// (https://openstax.org/r/safetyheatmap)

Using Python to Generate Heatmaps

There are several ways to generate heatmaps in Python. Two common methods include the following:

- `imshow()` function, which is part of `MatPlotlib` plotting library. The `imshow()` function can easily display heatmaps such as the two-dimensional heatmap shown in Figure 9.7.

- `heatmap()` function, which is part of the Seaborn library. Seaborn provides a high-level capability for statistical graphs.

EXPLORING FURTHER

Seaborn

Seaborn includes a wide variety of functions for visualizing statistical relationships. This user guide and tutorial (https://openstax.org/r/tutorial) provides an excellent introduction to its uses.

In the next example, Python is used to create a heatmap based on a heatmap function called `sns.heatmap()`, which is part of the Seaborn library. The example plots number of airline passengers for time-based data of month and year where month is plotted on the horizontal axis, year is plotted on the vertical axis, and the color coding of the heatmap represents the magnitude of the number of airline passengers.

EXAMPLE 9.10

Problem

Generate a heatmap of a dataset using the heatmap function that is part of the Seaborn library in Python.

Solution

As an example of generating a simple heatmap in Python using the heatmap function, we can make use of an existing dataset in the Seaborn library called "flights (https://openstax.org/r/flights)" that lists month, year, and number of passengers for historical airline data: Master Flights Seaborn Data (https://openstax.org/r/mwaskom).

Note: Many such datasets are available at Data Depository for Seaborn Examples (https://openstax.org/r/seaborn1).

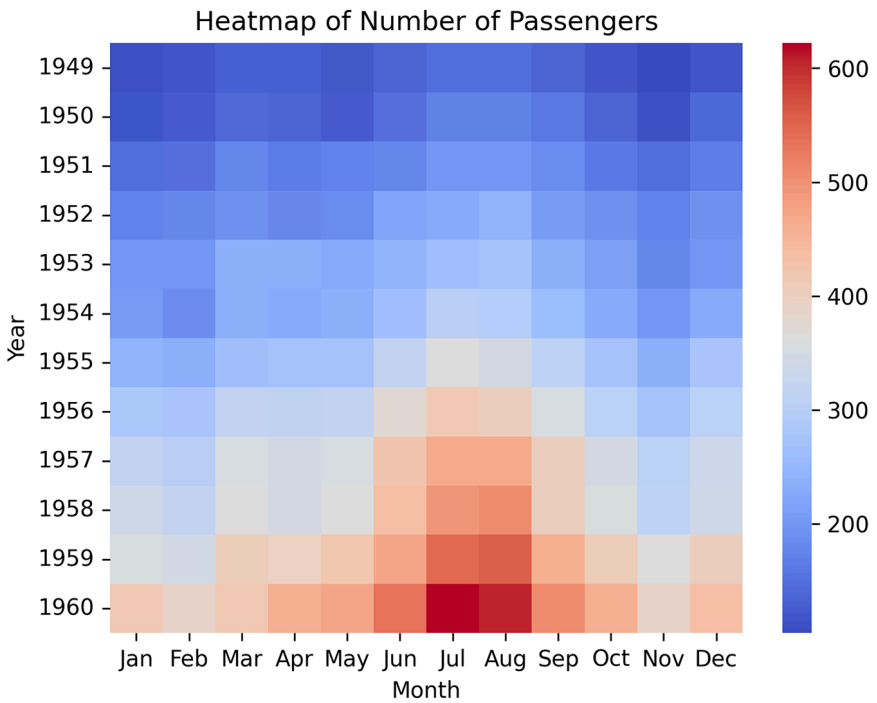
Once the array is created, the function `sns.heatmap()` can be used to generate the actual heatmap, which will color code the matrix based on the number of passengers. The `sns.heatmap` function also shows the color bar to provide the viewer with the scale of color gradient used in the heatmap.

PYTHON CODE

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset from seaborn library called flights
heatmap_data = sns.load_dataset("flights")

# Create an array with heatmap data from the flights dataset
heatmap_data_array = heatmap_data.pivot(index="year", columns="month",
values="passengers")
```

```
# Create the heatmap
sns.heatmap(heatmap_data_array, cmap = 'coolwarm')

# Set labels and title
plt.xlabel('Month')
plt.ylabel('Year')
plt.title('Heatmap of Number of Passengers')

# Show the plot
plt.show()
```

The resulting output will look like this:



By examining the heatmap the viewer can conclude that the number of passengers has increased year to year, and that the summer months of July and August tend to have a higher number of passengers.

Introduction to Geospatial Information System (GIS) Mapping

Geospatial Information System (GIS) mapping is a tool for visualizing, analyzing, and interpreting spatial data that makes use of various types of geographical data, such as maps and satellite images. The goal is to allow the user to visualize patterns and trends based on geographic attributes.

It is likely that you have already come across a heatmap overlaid on a geographic map in a news article or website. Heatmaps such as these are often referred to as a **choropleth graph**, which refers to the graphing of data according to geographic boundaries such as states, counties, etc. A choropleth graph is created by overlaying a grid on a geographic map and then typically coloring the map based on density or some other measurement for a specific grid location on the map. Choropleth graphs are commonly used to plot data overlaid on geographic maps for economic data, population data, housing data, medical-related data, etc.

These graphs provide a very natural and intuitive way to visualize data for certain geographic regions and thus identify patterns, trends, and correlations. These types of maps also allow the user to identify clusters and outliers to assist in data interpretation.

For example, a medical researcher might be interested in visualizing and tracking the spread of a virus over time across various counties, or a job seeker might be interested in visualizing the density of specific job openings in various cities.

Here are some other applications where the use of GIS mapping provides visualizations and insights, trends, and correlations:

- Telecommunications services and coverage
- Urban and transportation planning
- Environmental management
- Health care applications
- Disaster assessment and recovery
- Supply chain management

EXPLORING FURTHER

Interactive COVID-19 Choropleth Graph

Figure 9.8 provides an example of a choropleth graph for COVID-19 cases overlaid on a map of the world. Notice how higher numbers of cases of COVID-19 correlate to darker color shading on the map. Click on the "play" button in this interactive time-lapse map (https://openstax.org/r/ourworldindata1) to see the change in time basis for the graph. Note that the time scale on the bottom of the graph can be used to show changes in data over time.

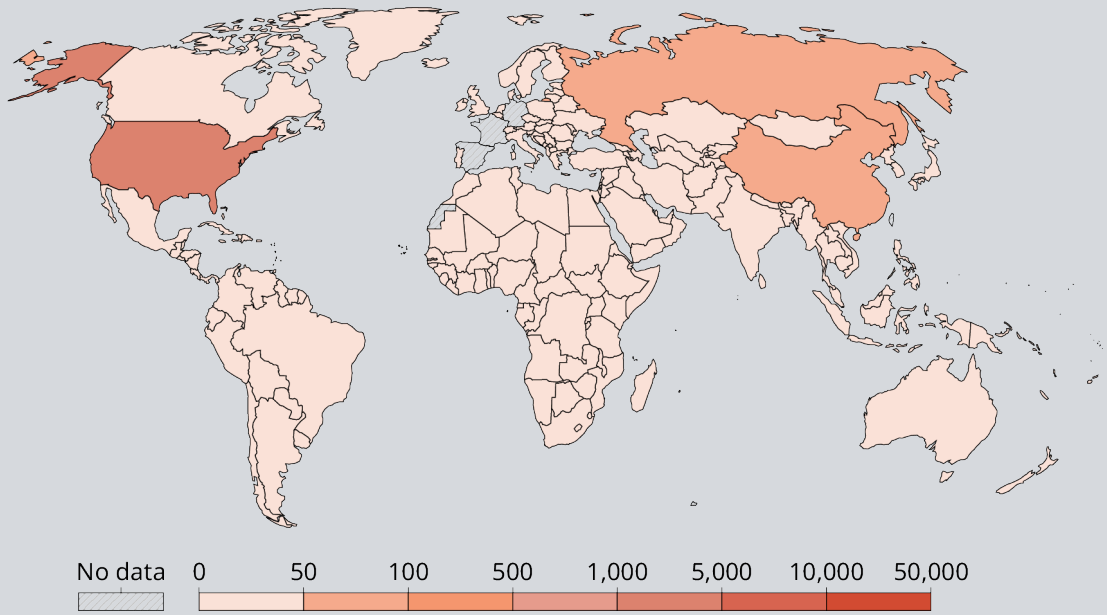


Figure 9.8 Choropleth Graph Showing Cases of COVID-19

(data source: Data from WHO COVID-19 Dashboard. Geneva: World Health Organization, 2020. Available online: https://data.who.int/dashboards/covid19/" via OurWorldinData.org. Retrieved from: https://ourworldindata.org/grapher/weekly-covid-deaths [Online Resource].)

Using Python for Geographic Mapping

As mentioned earlier, Python provides a number of libraries and tools to facilitate the generation of

geographic mapping. There are several steps involved in the process:

1. Establish and collect geospatial data: The first step is to collect the geospatial data of interest. This type of data is readily available from a number of sources such as open data repositories, government agencies such as the Census Bureau, for example, etc. The data can be found in various formats such as shapefiles, GeoJSON files, etc. (a shapefile is a data format used for geospatial data that represents geographic data in a vector format).
2. Data processing: Once the data is collected, data processing or transformation may be needed, as discussed in Collecting and Preparing Data. For example, some form of coding might be needed to transform addresses into geographic coordinates such as latitude and longitude to facilitate the geographic mapping. Additional data processing might be needed to estimate or interpolate missing data to generate continuous temperature maps, for example.
3. Generate the map to visualize the geographic data using libraries such as `matplotlib` and `geopandas`. Various tools also provide interactivity to the map so that the user can zoom in, zoom out, rotate, or view data when hovering over a map feature.

Python provides an excellent environment to allow the user to create, manipulate, and share geographic data to help visualize geospatial data and detect trends or arrive at conclusions based on the data analysis.

9.5 Multivariate and Network Data Visualization Using Python

Learning Outcomes

By the end of this section, you should be able to:

- 9.5.1 Produce labeled scatterplots and scatterplots with variable density points, different colors, etc. to indicate additional information.
- 9.5.2 Create and interpret correlation heatmaps from multidimensional data.
- 9.5.3 Create and interpret graphs of three-dimensional data using a variety of methods.

A data scientist or researcher is often interested in generating more advanced visual representations such as scatterplots, correlation maps, and three-dimensional (3D) representations of data. Visualizations or graphs produced in a 3D format are especially important to convey a realistic view of the world around us. For example, on a manufacturing line, a 3D visualization of a mechanical component can be much easier to interpret than a two-dimensional blueprint drawing. 3D graphs can offer insights and perspectives that might be hidden or unavailable in a two-dimensional view.

3D visualizations are especially useful when mapping large datasets such as population density maps or global supply chain routes. In these situations, Python tools such as `geopandas` or 3D plotting routines available as part of `matplotlib` can be used to visualize relationships, patterns, and connections in large datasets.

As we saw in Decision-Making Using Machine Learning Basics, data collection and analysis of very large datasets, or *big data,* has become more common as businesses, government agencies, and other organizations collect huge volumes of data on a real-time basis. Some examples of big data include social media data, which generates very large datasets every second, including images, videos, posts, likes, shares, comments, etc. Data visualization for social media data might include dashboards to visualize trends and engagement on hot topics of the day (see more on these in Reporting Results), or geospatial maps to show the geographic distribution of users and associated demographics.

In this section, we explore more advanced visualization techniques such as scatterplots with variable density points, correlation heatmaps, and three-dimensional type analysis.

Scatterplots, Revisited

A **scatterplot** is a graphing method for bivariate data, which is paired data in which each value of one variable is paired with a value of a second variable, which plots (x, y) data for two numeric quantities, and the goal is to

determine if there is a correlation or dependency between the two quantities. Scatterplots were discussed extensively in Inferential Statistics and Regression Analysis as a key element in correlation and regression analysis. Recall that in a scatterplot, the independent, or explanatory, quantity is labeled as the *x*-variable, and the dependent, or response, quantity is labeled as the *y*-variable.

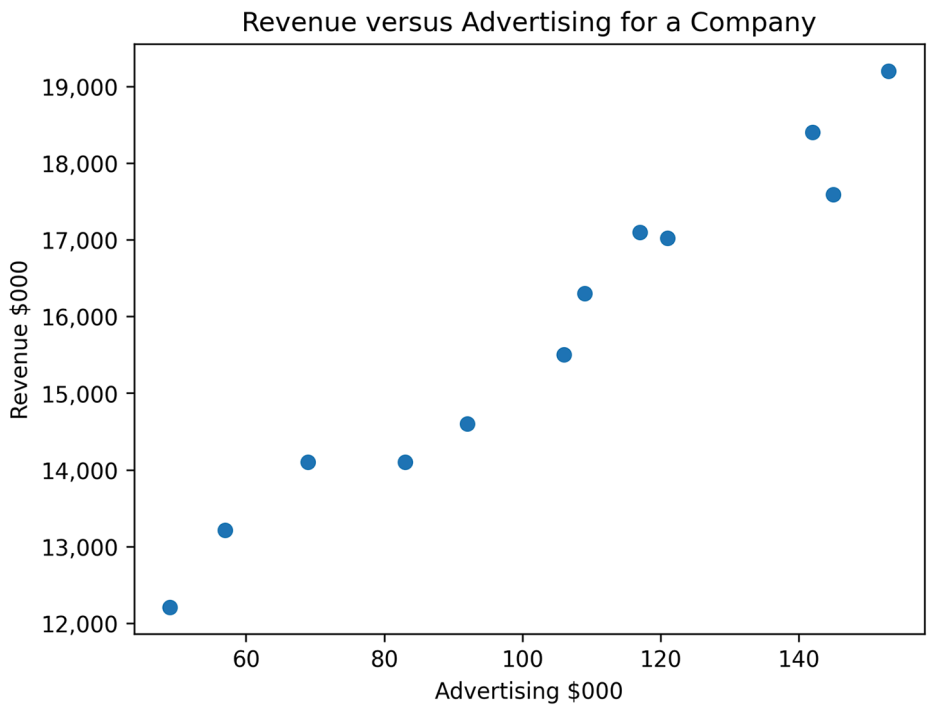
When generating scatterplots in Inferential Statistics and Regression Analysis, the `plt.scatter()` function was used, which is part of the Python `matplotlib` library. In Correlation and Linear Regression Analysis, an example scatterplot was generated to plot revenue vs. advertising spend for a company.

The Python code for that example is reproduced as follows:

PYTHON CODE

```python
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Define the x-data, which is the amount spent on advertising
x = [49, 145, 57, 153, 92, 83, 117, 142, 69, 106, 109, 121]

# Define the y-data, which is revenue
y = [12210, 17590, 13215, 19200, 14600, 14100, 17100, 18400, 14100, 15500, 16300,
17020]

# Use the scatter function to generate a time series graph
plt.scatter(x, y)

# Add a title using the title function
plt.title("Revenue versus Advertising for a Company")

# Add labels to the x and y axes by using xlabel and ylabel functions
plt.xlabel("Advertising $000")
plt.ylabel("Revenue $000")

# Define a function to format the ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'

# Apply the custom formatter to the y-axis
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))

# Show the plot
plt.show()
```

The resulting output will look like this:



We can augment this scatterplot by adding color to indicate the revenue level (*y*-value).

To do this, use the *c* parameter in the `plt.scatter` function to specify that the color of the points will be based on the *y*-value of the data point. The parameter `cmap` allows the user to specify the color palette to be used such as "hot," coolwarm," etc.

In addition, a color bar can be added to the scatterplot so the user can interpret the color scale.

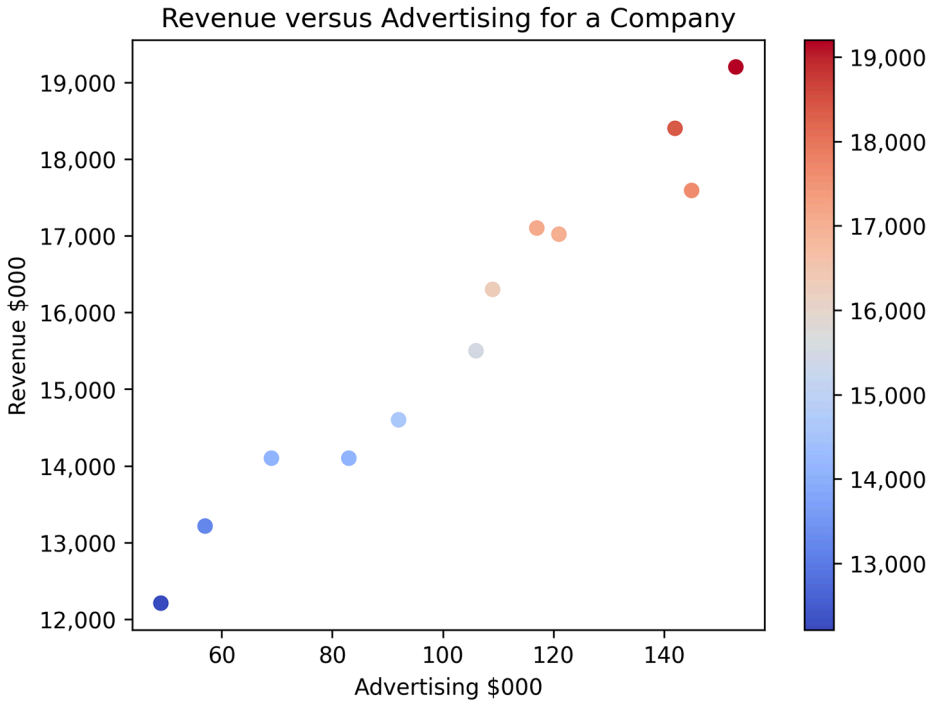
Here is a revised Python code and scatterplot that includes the color enhancement for the data points:

PYTHON CODE

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Define the x-data, which is the amount spent on advertising
x = [49, 145, 57, 153, 92, 83, 117, 142, 69, 106, 109, 121]

# Define the y-data, which is revenue
y = [12210, 17590, 13215, 19200, 14600, 14100, 17100, 18400, 14100, 15500, 16300,
17020]

# Use the scatter function to generate a scatterplot
# Specify that the color of the data point will be based on revenue (y)
# Specify the 'coolwarm' color palette
plt.scatter(x, y, c=y, cmap='coolwarm')
```

```
# Plot a color bar so the user can interpret the color scale
cbar = plt.colorbar()

# Define a function to format the color bar ticks with commas as thousands
separators
def format_colorbar_ticks(value, tick_number):
    return f'{value:,.0f}'

# Apply the custom formatter to the color bar ticks
cbar.ax.yaxis.set_major_formatter(ticker.FuncFormatter(format_colorbar_ticks))

# Add a title using the title function
plt.title("Revenue versus Advertising for a Company")

# Add labels to the x and y axes using xlabel and ylabel functions
plt.xlabel("Advertising $000")
plt.ylabel("Revenue $000")

# Define a function to format the y-axis ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'

# Apply the custom formatter to the y-axis
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))

# Show the plot
plt.show()
```

The resulting output will look like this:



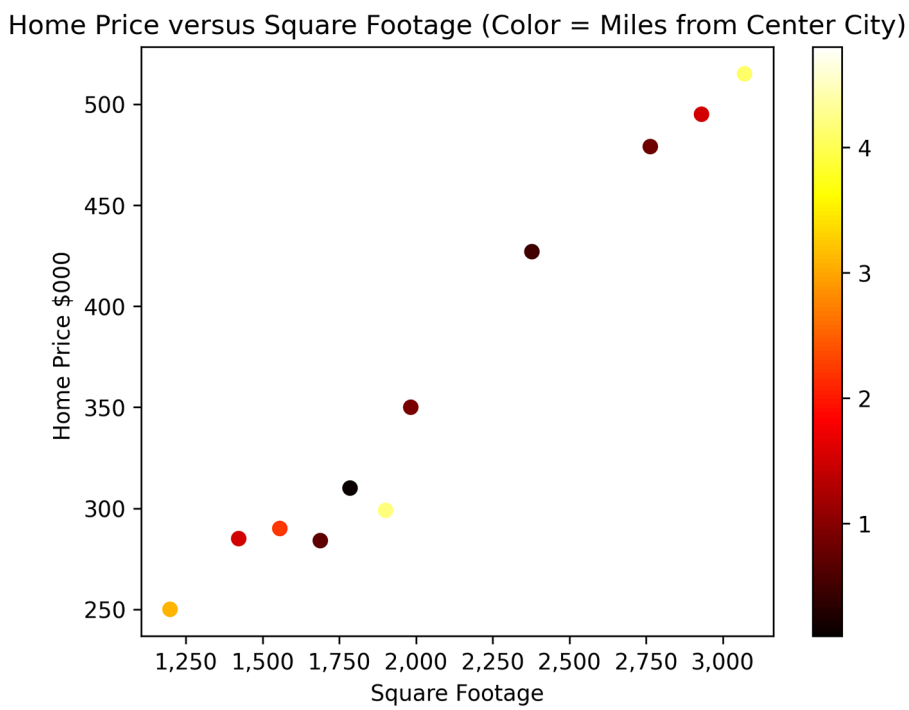
A typical application for this capability is to plot the color of the datapoint on a scatterplot based on the value of a third variable. For example, a real estate professional may be interested in the correlation between home price, square footage of the home, and location in terms of distance in miles from the center of a city. The scatterplot can be created based on (x, y) data, where *x* is the square footage of the home, *y* is the price of the home, and *z* is the distance from center of the city, represented by the color of the point.

PYTHON CODE

```
# import the matplotlib library
import matplotlib.pyplot as plt

# define the x-data, which is square footage of a home
x = [2100, 2378, 1983, 1422, 2764, 1901, 1198, 1785, 1556, 2931, 3071, 1688]

# define the y-data, which is the corresponding home price in 000
y = [390, 427, 350, 285, 479, 299, 250, 310, 290, 495, 515, 284]

# define the z-data, which is the corresponding distance from city center
z = [4.8, 0.5, 0.9, 1.5, 0.8, 4.2, 3.1, 0.1, 2.2, 1.5, 4.1, 0.7]

# use the scatter function to generate a time series graph
# specify that the color of the data point will be based on revenue (y)
# specify the hot color palette
plt.scatter(x, y, c = z, cmap='hot')
```

```
# also plot a color bar so user can interpret the color scale
plt.colorbar()

# Add a title using the title function
plt.title("Home Price versus Square Footage (Color = Miles from Center City)")

# Add labels to the x and y axes by using xlabel and ylabel functions
plt.xlabel("Square Footage")
plt.ylabel ("Home Price $000")

# Define a function to format the ticks with commas as thousands separators
def format_ticks(value, tick_number):
    return f'{value:,.0f}'

# Apply the custom formatter to the x-axis
plt.gca().xaxis.set_major_formatter(ticker.FuncFormatter(format_ticks))
```

The resulting output will look like this:



Correlation Heatmaps

Recall from Correlation and Linear Regression Analysis that correlation analysis is used to determine if one quantity is related to or correlates with another quantity. For example, you know that the value of a car is correlated with the age of the car where, in general, the older the car, the less its value. A numeric correlation coefficient (*r*) was calculated that provided information on both the strength and direction of the correlation.

The *value* of *r* gives us this information:

- The value of *r* is always between -1 and $+1$. The size of the correlation *r* indicates the strength of the linear relationship between the two variables. Values of *r* close to -1 or to $+1$ indicate a stronger linear

relationship. Values of *r* close to zero indicate weak correlation.

The *sign* of *r* gives us this information:

- A positive value of *r* means that when *x* increases, *y* tends to increase, and when *x* decreases, *y* tends to decrease (*positive correlation*).
- A negative value of *r* means that when *x* increases, *y* tends to decrease, and when *x* decreases, *y* tends to increase (*negative correlation*).

Data scientists often collect data on more than two variables and then are interested in the various correlations between the variables. Of particular interest are those pairs of variables with stronger correlations. A convenient way to visualize this data is with a **correlation heatmap**, which assigns a color palette to the various values of the correlation coefficient for the two variables.

Python provides a mechanism to calculate a correlation matrix, which shows the value of *r* for each pair of variables in a dataset using the `corr()`. This correlation matrix is typically used by researchers to identify those variables having stronger correlations. A correlation heatmap is a visual representation of the correlation matrix that implements color coding to visualize those variables with stronger correlations and those variables with weaker correlations.

Typically, darker colors are used to indicate variables with higher correlations and lighter colors are used to indicate variables with weaker correlations.

As an example of a correlation heatmap, we can use an available dataset from the Seaborn library related to car crash data found here at the car crashes dataset (https://openstax.org/r/seabornblob); the fields in the dataset are shown in Table 9.5.

Field Name	Field Description
abbrev	Abbreviation of the state name
total	Total number of car crashes reported in the state
speeding	Percentage of car crashes in which speeding was a contributing factor
alcohol	Percentage of car crashes in which alcohol was involved
not_distracted	Percentage of car crashes where the driver was not distracted
no_previous	Percentage of car crashes where the driver had no previous accidents
ins_premium	Insurance premium per insured vehicle
ins_losses	Insurance losses incurred per insured vehicle

Table 9.5 Fields and Descriptions for the "Car_Crashes" Dataset from the Seaborn Library

There are 50 records in the dataset (one record per state). The first five lines of the dataset appear in Table 9.6.

total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	abbrev
18.8	7.332	5.64	18.048	15.04	784.55	145.08	AL
18.1	7.421	4.525	16.29	17.014	1053.48	133.93	AK
18.6	6.51	5.208	15.624	17.856	899.47	110.35	AZ
22.4	4.032	5.824	21.056	21.28	827.34	142.39	AR
12	4.2	3.36	10.92	10.68	878.41	165.63	CA

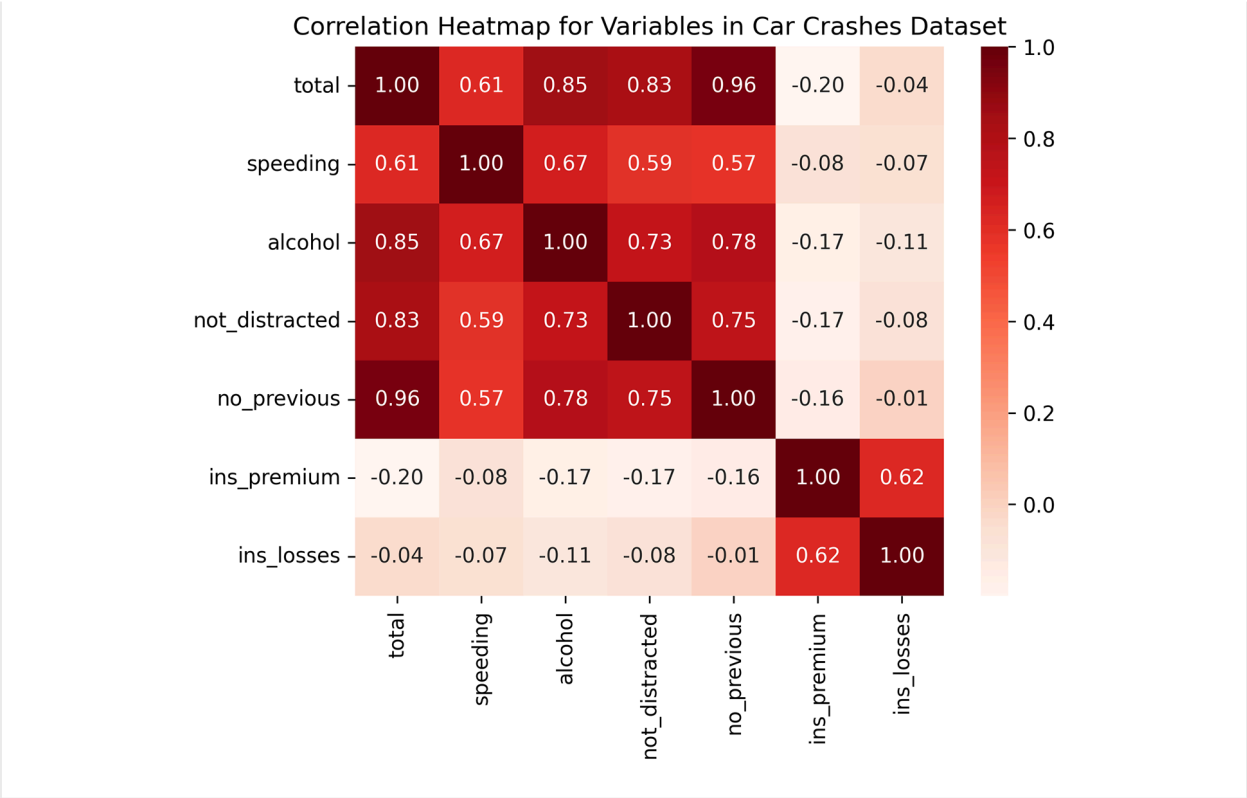
Table 9.6 "Car_Crashes" Dataset Excerpt

The short Python program that follows can be used to generate a correlation heatmap for the variables in the dataset; it makes use of the `heatmap` function available in the `Seaborn` library (see also Decision-Making Using Machine Learning Basics).

PYTHON CODE

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the car_crashes dataset from seaborn library
dataset = sns.load_dataset("car_crashes")

# Select only numeric columns
numeric_columns = dataset.select_dtypes(include=['number'])

# Calculate the correlation matrix
correlation_matrix = numeric_columns.corr()

# Generate the correlation heatmap using the Reds color palette
sns.heatmap(correlation_matrix, cmap='Reds', annot=True, fmt=".2f")

# Add a Title to the plot
plt.title('Correlation Heatmap for Variables in Car Crashes Dataset')

# Show the plot
plt.show()
```

The resulting output will look like this:



Notice in the correlation heatmap that dark-red coloring is visible on the diagonal from the upper left to the lower right corner of the matrix. This dark shading represents a correlation coefficient of 1.0 since any variable is considered to be perfectly correlated with itself.

Visualizing Three-Dimensional Data

Let's say you are taking an anatomy class, and an instructor attempts to verbally describe the human nervous system. Compare this to an interactive 3D model of the human nervous system, where a student can view the nervous system in three dimensions, rotate the 3D image, and view the nervous system from various angles and perspectives. Let's say an auto mechanic needs to repair a transmission on a car and attempts to read the written technical service manual to understand how to start the repair. Compare this to an interactive 3D model of the car's transmission where the mechanic can visualize the exact location for the repair and interact with the 3D model.

These examples show the advantages and power of three-dimensional perspectives. Visualizing three-dimensional data has many benefits such as spatial perception, depth perception, and an ability to better illustrate the clustering and patterns inherent in the dataset. **3D visualization** is also useful in simulation and modeling applications.

Three-dimensional visualization is also used as part of multiple linear regression analysis to view a scatterplot in three dimensions and view the corresponding regression plane (see Machine Learning in Regression Analysis).

Python includes several tools and libraries to facilitate 3D visualization and analysis; some tools allow for interactivity such as the ability to zoom in/zoom out, pan, and rotate three-dimensinoal images. Here are some Python modules that support 3D visualization:

- Matplotlib (https://openstax.org/r/matplotlib2) includes facilities for 3D visualization such as `mpl.toolkits.mplot3d`.
- Plotly (https://openstax.org/r/plotly) is a library that includes 3D plotting capability, surface plots, and

more.
- VTK (https://openstax.org/r/vtk) is a library for 3D graphics and image processing (VTK stands for Visualization Toolkit).

To illustrate an example of a 3D scatterplot in Python using `Matplotlib`, let's generate Python code to plot the Seaborn "Iris" data, which can be downloaded from the Seaborn repository (https://openstax.org/r/mwaskom1). (We used this dataset in What Are Data and Data Science? and Deep Learning and AI Basics as well.) The dataset contains 150 rows of data related to measurements for different varieties of iris flowers, but we will only be using a few rows (see Table 9.7).

Sepal Length	Sepal Width	Petal Length	Petal Width	Species of Iris
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
6.7	3.0	5.2	2.3	virginica
5.9	3.0	5.1	1.8	virginica

Table 9.7 "Iris" Dataset Excerpt

We will use the `mpl_toolkits.mplot3d()` function to generate the three-dimensional scatterplot. First, we will read in the "iris" dataset from the Seaborn library (https://openstax.org/r/seabornpydata). Next, the 3D scatterplot is created using the `fig.add.subplot()` function. The `ax.scatter()` function is then used to plot the data points.

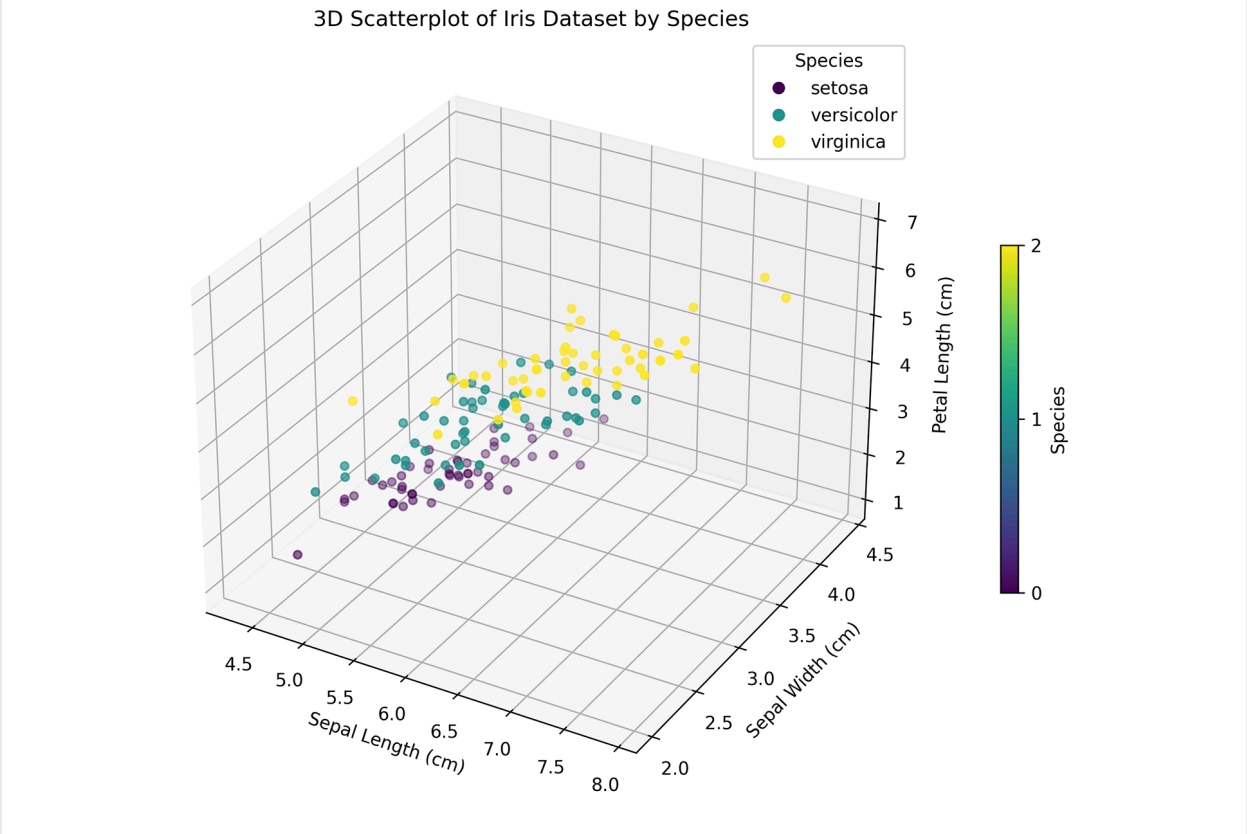
Here is the Python code:

PYTHON CODE

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import matplotlib.cm as cm
import matplotlib.colors as mcolors
from matplotlib.patches import Rectangle

# Load the iris dataset from Seaborn data repository
flower = sns.load_dataset('iris')

# Create a blank figure with a white background
fig = plt.figure(figsize=(9, 8), facecolor='white')

# Add a subplot to the figure
ax = fig.add_subplot(111, projection='3d')

# Set the background color of the axes to white
ax.set_facecolor('white')
```

```
# Plot data points
scatter = ax.scatter(flower['sepal_length'], flower['sepal_width'],
flower['petal_length'],
                     c=flower['species'].astype('category').cat.codes,
cmap='viridis', marker='o')

# Set labels and title
ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Sepal Width (cm)')
ax.set_zlabel('Petal Length (cm)')
ax.set_title('3D Scatterplot of Iris Dataset by Species')

# Map numerical codes to actual species names
species_names = {0: 'setosa', 1: 'versicolor', 2: 'virginica'}

# Create a ScalarMappable object
norm = plt.Normalize(flower['species'].astype('category').cat.codes.min(),
                     flower['species'].astype('category').cat.codes.max())
sm = plt.cm.ScalarMappable(cmap='viridis', norm=norm)
sm.set_array([])

# Create the colorbar and adjust its spacing
cbar = plt.colorbar(sm, ticks=[0, 1, 2], label='Species', fraction=0.02, pad=0.1)

# Adjust colorbar position to increase space between colorbar and plot
cbar.ax.tick_params(labelsize=10)  # Adjust the colorbar tick label size if needed

# Adjust figure layout to provide appropriate space around the plot
fig.subplots_adjust(left=0.1, right=0.85, top=0.9, bottom=0.1)  # Adjust margins as
needed

# Add legend
ax.legend(handles=scatter.legend_elements()[0],
          labels=[species_names[name] for name in
flower['species'].astype('category').cat.codes.unique()],
          title='Species')

plt.show()
```

The resulting output will look like this:



Datasets

Note: The primary datasets referenced in the chapter code may also be [downloaded here (https://openstax.org/r/folders14a)](https://openstax.org/r/folders14a).

Key Terms

3D visualization a graph or display that shows information plotted along three dimensions, typically referred to as the x-axis, y-axis, and z-axis

bar graph a chart that presents categorical data in a summarized form based on frequency or relative frequency

bin an interval or range into which data points are grouped; often used to create histograms

binomial distribution a probability distribution for discrete random variables where there are only two possible outcomes of an experiment

bivariate data paired data in which each value of one variable is paired with a value of a second variable

boxplot ("box-and-whisker plot") a graphical display showing the five-number summary for a dataset: the min, first quartile, median, third quartile, and the max

choropleth graph a graphical display where areas are shaded in proportion to the value of a variable being represented; choropleth maps are typically used to present spatial patterns in geographic regions

correlation heatmap a visual representation of the correlation matrix that implements color coding to visualize those variables with stronger correlations and those variables with weaker correlations.

data visualization the use of graphical displays, such as bar charts, histograms, and scatterplots, to help interpret patterns and trends in a dataset

discrete random variable a random variable where there is only a finite number of values that the variable can take on

five-number summary a summary of a dataset that includes the minimum, first quartile, median, third quartile, and maximum

geospatial data data that describes the geographic location, shape, size, and other attributes relative to a location on the Earth's surface

Geospatial Information System (GIS) mapping a tool for visualizing, analyzing, and interpreting spatial data that makes use of various types of geographical data, such as maps and satellite images

grid heatmap a graphical representation of data where values are depicted as colors within a grid such as an (x, y) mapping; a grid heatmap is typically used to show correlations between two quantities

histogram a graphical display of continuous data showing class intervals on the horizontal axis and frequency or relative frequency on the vertical axis

interquartile range (IQR) a number that indicates the spread of the middle half, or middle 50%, of the data; the difference between the third quartile (Q_3) and the first quartile (Q_1)

line chart a type of graph that uses lines to connect (x, y) data points

median the middle value in an ordered dataset

normal distribution a bell-shaped distribution curve that is used to model many measurements, including IQ scores, salaries, heights, weights, blood pressures, etc.

outliers data values that are significantly different from the other data values in a dataset

Pareto chart a type of bar chart where the bars are arranged in order of decreasing height

Poisson distribution a probability distribution for discrete random variables used to calculate probabilities for a certain number of successes in a specific interval

quartiles numbers that divide an ordered dataset into quarters; the second quartile is the same as the median

scatterplot (or scatter diagram) a graphical display that shows the relationship between a dependent variable and an independent variable

spatial heatmap a data visualization method used to represent the density or intensity of data points within a geographical area using coloring and shading to represent densities of various attributes

univariate data observations recorded for a single characteristic or attribute

Group Project

Data Visualization: Scatterplot with Color Shading

For your group, you will select at least 15 different models and types of vehicles and research the value of a used car for that model in your area.

You can search used car websites such as autotrader.com (https://openstax.org/r/autotrader) or cars.com (https://openstax.org/r/cars).

The result is you should have at least 15 records where you record the brand, model, type of vehicle, age, and value of the vehicle.

For example, a few rows in the dataset might look like (Table 9.8):

Brand	Model	Type_of_Vehicle	Age	Value
Toyota	Camry LE	Sedan	5	$22750
Ford	Expedition	SUV	7	$37640

Table 9.8 Used Car Dataset Template

As a group, complete the following steps:

a. Create a scatterplot of your (x, y) data points. Plot the age of the vehicle on the *x*-axis and plot the value on the *y*-axis.
b. Color code the data points according to type of vehicle. For example, color datapoints for sedans using one color, color datapoints for SUVs using a different color, color sports cars using a different color, etc.
c. Discuss the trend of the scatterplot—that is, do the plotted points follow an upward trend from left to right?
d. Discuss the information provided by the color coding of the datapoints on the scatterplot.
e. Could this scatterplot be useful if you were shopping for a used vehicle?

Critical Thinking

1.

a. For the following five-number summary of a dataset of textbook cost per student per semester, create a horizontal boxplot.

$$\begin{aligned} \text{Minimum} &= \$83 \\ \text{First Quartile} &= \$255 \\ \text{Median} &= \$481 \\ \text{Third Quartile} &= \$604 \\ \text{Maximum} &= \$793 \end{aligned}$$

b. Analyze the boxplot and comment on variability, symmetry, and skewness.

2.

a. Create a graph for the binomial distribution with $n = 50$ and $p = 0.75$.
b. Analyze the graph and comment on symmetry and skewness.

3.

a. Create a graph for the Poisson distribution with mean $\mu = 7$.
b. Analyze the graph and comment on symmetry and skewness.

4.

a. Create a graph of the normal distribution for weights of car engines with mean of 200 kilograms and standard deviation of 15 kilograms.

b. Analyze the graph and comment on symmetry and skewness.
c. Determine the probability that a random engine weighs more than 245 kilograms.

5.
a. Create a time series chart for dataset in the table as shown. The following data represents the number of subscribers to a social media site over a 7-year time period.

Year	Subscribers (in millions)
1	6.3
2	7.9
3	11.0
4	9.8
5	8.2
6	6.4
7	4.9

Table 9.9 Social Media Subscribers

b. Comment on any trends based on a review of the graph.

6. Access and download any dataset from the Seaborn repository at https://github.com/mwaskom/seaborn-data (https://github.com/mwaskom/seaborn-data).
a. Create a correlation heatmap for the selected dataset to investigate correlations among the variables.
b. From the heatmap, which two variables exhibit the strongest correlation?
c. From the heatmap, which two variables exhibit the weakest correlation?

7. Access and download any dataset from the Seaborn repository at https://github.com/mwaskom/seaborn-data (https://github.com/mwaskom/seaborn-data).
a. Create a 3D visualization for the dataset.
b. What conclusion can you draw from the 3D visualization?

10 Reporting Results

Figure 10.1 The report is key: When done well, data science reports provide evidence-based insights that promote transparency by documenting methods and assumptions used, help identify risks and opportunities, and support decision-making. (credit: "CC-BY-Mapbox-Uncharted-ERG_Mapbox-b041" by Mapbox/Flickr, CC BY 2.0)

Chapter Outline

10.1 Writing an Informative Report
10.2 Validating Your Model
10.3 Effective Executive Summaries

Introduction

Data science is not just about analyzing datasets. Communicating the findings effectively is an essential part of the process aimed at promoting understanding and retention of key information. The ability to convey complex information to various kinds of audiences in a clear, consumable, and accessible manner is a vital skill for any data scientist. This chapter is designed to equip you with the skills necessary to create informative reports that combine analytical rigor and clarity. You will learn how to identify and engage your audience, document the intricate methods that underpin your data science projects, and determine the most compelling ways to present your data, analysis, and findings.

Throughout the chapter, we will be utilizing a common dataset, diabetes_data.csv (https://openstax.org/r/spreadsheetsd), to demonstrate ways in which Python can help to create elements critical to a clear and informative report. These data are courtesy of Dr. John Schorling, Department of Medicine, University of Virginia School of Medicine (rdrr.io/cran/ROCit/man/Diabetes.html (https://openstax.org/r/rocit)). The dataset was used in a study published in 1997 to determine the prevalence of coronary heart disease (CHD) risk factors (including factors associated with diabetes) among a population-based sample of Black residents of Virginia. CHD is one of the most common causes of death among Black adults (Willems et al., 1997). The dataset contains information on 403 subjects out of a total of 1,046 diabetes patients interviewed, and it is readily accessible.

10.1 Writing an Informative Report

Learning Outcomes

By the end of this section, you should be able to:

- 10.1.1 Identify the audience for a technical report and tailor word choice and style appropriately.
- 10.1.2 Document and describe the methods used in a data science project.
- 10.1.3 Determine the most effective way to present data, analysis, and results.

Data scientists should be prepared to explain the research question, the sources of data, the steps in data processing, key analytical techniques and statistical methods employed and their purposes and justifications, and finally, the outcomes of the project. They must be prepared to create and present visual exhibits (e.g., tables, graphs, charts) to support their findings. Additionally, it is important to discuss any limitations or potential biases in the data and analysis and offer actionable insights as well as guidance for further work. Such comprehensive reporting not only builds trust with the project team, but also contributes to the advancement of knowledge by enabling peer review and constructive feedback. Report writers often use a *structured format*, in which the report is divided into various standard parts for ease of understanding and consistency.

The main elements of a data science report include the following:

- *Title page*, providing the title of the report, the names of the team members who worked on it, their affiliations, and the current date
- *Executive summary*, including a brief overview of the report, the team's key findings, and main conclusions and recommendations
- *Methodology*, which discusses the data sources, data collection and cleaning methods, algorithms and models used, and assumptions made during the analysis
- *Data exploration and analysis*, detailing the whole analysis process, including descriptive statistics of the data; visualizations; model building; training, validation, and testing of models; hyperparameter tuning; and model comparison and evaluation using performance metrics
- *Results and discussion*, conveying the finding and their interpretation, along with strengths and weaknesses of the models, potential biases (and how they were addressed), and areas for further exploration
- *Conclusion*, containing a summary of key findings, final thoughts, and recommendations

Additional parts may be included in the report, including a table of contents (TOC), introduction section, list of references, and appendices as needed. Often, those commissioning the project may require a certain format for the report, and so it is essential for data scientists to work closely with all parties to determine what format is preferred in order to create an effective and informative report that fully meets everyone's needs.

Writing to Your Audience

Writing an informative data science report requires, first and foremost, a full understanding of the **audience** for which you are writing—that is, the individuals who will be reading the report. For instance, does the audience consist of fellow data science experts who can understand complex details or corporate individuals who need concise information to help them make sound business decisions? Does your audience rely more on visuals for explanation, or do they require additional detailed explanations? Perhaps your audience includes a mixture of businesspeople, technical experts, and end users, necessitating multiple formats for conveying the same information in different ways.

Start by determining the audience member types. We often talk about four audience categories: experts, technicians, executives, and nonspecialists. Each category has distinct characteristics, perspectives, expectations, and objectives, as described below.

Experts

Experts are individuals with an advanced understanding of the subject matter, often with specialized knowledge or education in the topic at hand. They are comfortable with the **jargon**, or specialized and/or technical terms of the field. Communication aimed at experts can be highly technical and detailed, and it often assumes a high level of prior knowledge. The purpose of such communication is usually to share new findings, discuss advanced techniques, or delve into the nuances of a particular issue.

Technicians

Technicians are practical users of information who may apply the knowledge in a hands-on manner. They typically have good familiarity with the subject but may not have the theoretical knowledge of experts. Information presented to technicians should be accurate and technical but focused on application and procedures. It can include jargon but should explain how the information affects practical work.

Executives

Executives consist of decision-makers such as managers, chief executive officers (CEOs), or business owners who may not have detailed technical knowledge but need an understanding of the implications of the information for strategic decision-making. Communication with executives should be concise, focused more on summaries and outcomes and less on details and theory. The language should be less technical and oriented toward business implications, costs, benefits, risks, and opportunities.

Sometimes, data science summaries are conveyed to executives and senior management by way of an executive dashboard. The **executive (or data) dashboard** is a visual representation tool designed to provide a quick, real-time overview of an organization's key data points and metrics, often updated in real time or at regular intervals. The key metrics, often referred to as **key performance indicators (KPIs)**, allow the audience to view consolidated data from various sources in an intuitive and visually engaging format. The purpose of the dashboard is to build awareness of and highlight key patterns to support quick identification of trends, anomalies, and opportunities. A key advantage of an executive dashboard is the real-time access to critical and actionable data that allows an executive to make rapid decisions regarding business-critical KPIs. Data dashboards also allow for continuous monitoring of performance metrics and can help identify trends where early identification may result in a competitive advantage to a company. Executives can quickly scrutinize the dashboard metric to assess progress toward corporate goals and objectives.

A few examples of executive dashboards include the ability to monitor sales performance through real-time sales data, revenue trends, and customer engagement metrics, which can enable teams to adjust strategies in a timely manner. Data dashboards can also support operational efficiency by displaying key metrics such as production rates, inventory levels, and logistic statuses to assist teams in optimizing processes. Additionally, data dashboards can be leveraged by marketing teams to visualize campaign performance, audience engagement, and conversion rates to help refine their tactics and maximize returns. Overall, the dashboard can enhance agility and effectiveness in informed managerial decision-making (see Figure 10.2).

Figure 10.2 The executive dashboard is a visual representation of data to give executives and senior management a quick, real-time overview of an organization's key performance indicators (KPIs).
(credit: modification of work "Analyst analytics blur" by David Stewart/https://homegets.com/Flickr, CC BY 2.0)

Government dashboards, with their focus on transparency to the public, often provide good examples.

Examples of U.S. federal government dashboards from different economic sectors include the following:

- Data USA (https://openstax.org/r/datausa)
- Aviation Community Dashboard (https://openstax.org/r/swpc)
- IT Portfolio Dashboard (https://openstax.org/r/itdashboard)
- Transportation Performance Management (https://openstax.org/r/fhwa)

Nonspecialists

Nonspecialists, or “lay” audiences, do not have specialized knowledge of the subject but might be interested in the topic or need to understand the basics for a particular reason. Communication with this group should avoid technical language and jargon as much as possible, or at least explain it thoroughly using clear, straightforward language. The focus should be on the broader implications of the information and its relevance in a wider context.

Writing for a Diverse Audience

When writing technical data science reports, it is often necessary to connect with diverse audiences who may have different levels of expertise, interest, and background knowledge. One can build connections by tailoring communication to a specific audience category while ensuring that the explanations are clear enough to resonate with others. The presenter must consider not only the complexity of the language, but also the depth of detail, the format, and the presentation of the information. The writing must be accessible to everyone in your audience and should avoid nonessential jargon, cryptic acronyms, and excessive details (unless the level of audience requires a high level of technicality and theory). Consider the *human element* when creating an informative narrative. In other words, take into account the perspectives, emotions, motivations, and experiences of the audience to make the information more engaging and meaningful. It also helps to apply several illustrative examples and clear analogies to help make the case for the real-world utility of your

approach and findings. When writing for a nontechnical audience, it is important to articulate the tangible impact and pertinence of the data, ensuring that it connects with the reader's context and concerns. In the description of the various aspects of the report, be selective and avoid overwhelming the audience with data that is less significant and instead spotlight pivotal key insights. Complement the narrative with visualizations (e.g., graphs, charts, tables) that illuminate the findings. Be mindful of the diverse spectrum of your potential audience and adopt a writing style that is coherent, succinct, and interesting, avoiding excessive complexity. Employ an active voice and brief sentence construction and offer careful transitional cues to guide your readers through the report.

Writers and presenters of technical reports should also consider issues related to audience **neurodiversity**, or the normal variation of individual cognitive abilities, especially in realms of communication and processing information. In the same way that we must consider a range of audience from expert to nonspecialist, it is vital to recognize that cognitive and neurological differences among readers or other consumers of the report can significantly impact their comprehension and engagement with the material. Some of the same principles apply, including use of clear, concise language, structured organization, and the inclusion of visual aids such as charts and graphs to cater to diverse learning and processing styles. Additionally, providing summaries or abstracts at the beginning of each section can aid in comprehension for those who might struggle with large blocks of text or complex information. By adopting these strategies, report writers not only uphold the principles of inclusivity and accessibility but also enhance the overall effectiveness of their communication, ensuring that their findings and insights are accessible to a broader audience.

Explaining Your Methods

As described in What Is Data Science?, the steps of the data science cycle consist of defining the objective, acquiring and cleaning data from a variety of sources, analyzing data, and communicating relevant insights. In a technical report, the purpose of explaining data science methods is to provide a clear and rigorous account across the data science cycle as well as to justify the validity and reliability of the results and conclusions. However, as mentioned above, different audiences may have different roles and levels of expertise in data science. Effectively communicating with these varied audiences requires different strategies for explaining the data science methods that you employed.

One strategy used when there may be audiences of different levels or interests is a **layered approach**, in which the report is organized into sections or appendices that provide different levels of detail and complexity for different audiences. For example, the main body of the report could provide a high-level overview of the data science methods, focusing on the main objectives, steps, and outcomes, using simple and nontechnical language. This would be suitable for audiences who are interested in the general findings and implications of the data analysis, such as managers and policymakers. The appendices could provide more detailed and technical descriptions of the data science methods, such as the data sources, sampling methods, preprocessing techniques, algorithms, parameters, evaluation metrics, and limitations. These appendices would be suitable for audiences who are interested in the technical aspects and validity of the data analysis, such as data scientists, researchers, or reviewers.

It is important that such a report adheres to a recognized editorial style, such as one of these:

- American Medical Association (AMA) (https://openstax.org/r/amastyle) for medical and health fields
- American Psychological Association (APA) (https://openstax.org/r/apastyle) for scientific and psychological fields
- Chicago Manual of Style (CMOS) (https://openstax.org/r/chicago) for published works
- IEEE (https://openstax.org/r/ieee) for engineering and computing fields

Always consult the project sponsors to find out what formatting and style are required for the report, and make sure that the report contains appropriate references and citations for any third-party sources referenced.

Some reports may require the development of a **codebook** or *data dictionary* to detail the variables, their units, values, and labels within the data or datasets used in the project. This codebook must align with the final report and the code used during the project to maintain consistency. In addition, it may be important to implement a **version control system**, such as Git or SVN, to help keep track of changes to code or any digital content over time, especially if the project is complex and there are many team members working on it. *Git* is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency, while *SVN* (Subversion) is a centralized version control system that allows users to keep track of changes to files and directories over time. These information repositories are beneficial for tracking and managing changes to code and data throughout the lifecycle of a project. Such a system documents the project's history, assists in resolving conflicts or errors, and facilitates collaboration among researchers. Crucially, it promotes *reproducibility*, enabling others to access and execute the same code and data utilized in the project. In addition, individuals can utilize cloud-based systems (e.g., Google Docs, Microsoft Office 365) that provide document sharing and change tracking features that allow for advance version control and a repository of report backup options.

Should That Be a Graph or a Table?

Visualizing Data covered in depth the importance of incorporating visual aids such as graphs, charts, tables, diagrams, and flowcharts as an effective way to clarify data science methodologies. These tools can distill complex information and highlight significant patterns, trends, or relationships within the data. Data visuals can captivate and maintain audience engagement while fostering comprehension and memory retention. Data scientists should exercise sensible caution when utilizing visual aids; improper design or presentation can lead to bias or misinterpretation. Thus, each visual aid should be paired with succinct, clear explanations that clarify the purpose of the visual aid, its content, and possible limitations. The inclusion of **alt text**, or brief descriptions of images that accompany the image or may be embedded within the image data, is essential to aid readers who are not able to view the images and graphics due to disability or other limitations.

The art of data science communication entails presenting analytical results logically and compellingly. Graphs and tables offer distinct advantages for various data types, reporting goals, audiences, and intended messages. Graphs excel at depicting trends, patterns, connections, and anomalies, whereas tables are optimal for displaying exact values, comparisons, and classifications. Given the diversity of available graphs and tables, selecting the most fitting type for the data at hand is crucial.

A well-constructed data analysis report should include enough graphs and tables within the main body to adequately explain each of the central arguments or findings. These should be straightforward, self-explanatory, and identified with descriptive labels, titles, legends, scales, and units. Consistency in style and formatting is essential. Authors should avoid overloading the report with excessive or overly intricate visuals that might distract the reader or graphics that offer no additional insight. The author is also responsible for clarifying the significance of these visuals within the accompanying text (alt text, captions, etc.), spotlighting the principal observations or inferences. Table 10.1 provides a summary of the types of visual aids that can be used in data science reports and the types of information that they best convey (note that these displays were discussed in Inferential Statistics and Regression Analysis and Visualizing Data):

Visual Aid	Informational Purpose
Line graphs	Illustrate time series data or variations across continuous data
Bar graphs	Compare the distributions of discrete or categorical data
Pie charts	Represent parts of a whole in percentages or proportions
Scatterplots	Demonstrate correlations between data from two continuous variables
Histograms	Show the distribution frequencies of data from a single continuous variable

Table 10.1 Summary of Visuals Used in Data Science Reports

Visual Aid	Informational Purpose
Box plots	Provide summary statistics and outliers for a continuous variable or across groups
Contingency tables	Display the joint frequency or counts for the data from a combination of two or more categorical variables
Summary tables	Present descriptive statistics or model outputs for data from one or more continuous variables or across different categories

Table 10.1 Summary of Visuals Used in Data Science Reports

In addition, consider using interactive visualizations, which offer users the opportunity to engage dynamically with information through direct manipulation and exploration. This method leverages the power of graphical tools to transform static data into a visual dialogue, allowing users to uncover patterns, anomalies, and insights by interacting with the visualization elements themselves. By providing mechanisms such as zooming, filtering, and sorting, interactive visualizations empower users to personalize their data exploration journey, making complex data more accessible and understandable.

EXAMPLE 10.1

Problem

A large pharmaceutical company is seeking to identify key health factors that are associated with coronary heart disease and diabetes, with the potential goal of identifying markets for medications based on the findings about those key factors. One source of data is the dataset diabetes_data.csv (https://openstax.org/r/spreadsheetsd). Assuming that some preliminary analysis has already been completed, how would a data scientist address each of the following elements of the report? (Note: You may want to return to the chapter introduction to review the information about this dataset.)

- Title Page
- Executive Summary
- Methodology
- Data Exploration and Analysis
- Results and Discussion
- Conclusion

Solution

- **Title Page:**

Title: Identifying Potential Markets for Coronary Heart Disease and Diabetes Medications

Prepared for: [Pharmaceutical Company Name]

Prepared by: [Data Scientist's Name]

Date: [Report Date]

- **Executive Summary:** This section provides a concise overview of the entire report, summarizing the key findings, methodologies, and recommendations. It should be written last to encapsulate the most important points derived from the entire analysis.

Example:

This report identifies key health factors associated with coronary heart disease (CHD) and diabetes, utilizing

data from the [diabetes_data.csv (https://openstax.org/r/spreadsheetsd)](https://openstax.org/r/spreadsheetsd) dataset. Preliminary analysis indicates significant correlations between factors [X, Y, Z] and disease prevalence. Recommendations include targeted marketing strategies in high-prevalence areas as well as areas in which populations are known to have high levels of [X, Y, Z]. [Add further details as appropriate from your analysis].

- **Methodology:** This section outlines the methods and procedures used to conduct the analysis. It details the data sources, data cleaning processes, analytical techniques, and any assumptions made during the analysis.

Example:

The analysis began with data cleaning and preprocessing of the diabetes dataset to handle missing values and ensure data integrity. Missing numerical values were imputed using the median values, while missing categorical data were filled in with the string "MISSING." Exploratory data analysis, including plotting pairs of variables, was conducted to uncover initial patterns and correlations. Advanced statistical methods and machine learning algorithms, including multiple linear regression, logistic regression, and decision trees, were employed to model disease prevalence and identify significant predictors.

- **Data Exploration and Analysis:** This section provides a detailed examination of the dataset, including descriptive statistics, visualizations, and initial insights. It identifies key variables, patterns, and anomalies that influence the analysis.

Example:

The dataset comprises key demographic and health metrics for a population-based sample of Black residents of Virginia. Variables analyzed include cholesterol, glucose level, age, weight, height, and presence of diabetes, among others. Descriptive statistics revealed that higher age and BMI are strongly associated with increased CHD and diabetes prevalence. Visualizations such as histograms, scatterplots, and heatmaps highlighted these correlations. Anomalies such as outliers were addressed through data normalization techniques.

- **Results and Discussion:** This section presents the findings of the analysis, interpreting the results and discussing their implications. It should relate the findings back to the research questions and objectives outlined in the introduction. Strengths and weakness should be discussed.

Example:

The analysis identified several key features that were highly correlated with CHD and diabetes within the dataset. Significant predictors of disease prevalence included age, BMI, and smoking status. One strength of the model is that the analysis employed a variety of modeling techniques, which allowed for a comprehensive evaluation of the data from different statistical perspectives, increasing the robustness of the findings. One weakness is the limited scope of the dataset, given its focus on a specific demographic. This suggests areas for further research, including expanding the dataset to include more diverse populations and other geographic regions.

- **Conclusion:** This section summarizes the key points of the report, reiterates the main findings, and provides actionable recommendations based on the analysis.

Example:

The analysis highlights critical factors that correlate with CHD and diabetes, providing insight that could lead to market penetration for medication into areas whose populations are known to have greater risk based on those health factors. Our team recommends conducting further research to expand the dataset and validate findings across different demographics. Addressing these high-prevalence areas with tailored health care solutions will not only meet the specific needs of affected populations but also drive the

company's growth and market presence in the health care sector.

10.2 Validating Your Model

LEARNING OUTCOMES:

By the end of this section, you should be able to:

- 10.2.1 Identify meaningful assumptions, state how they are used in the modeling process, and provide justifications for each assumption.
- 10.2.2 Discuss various error measures in the context of a modeling problem.
- 10.2.3 Perform sensitivity analysis on models and interpret the results to determine how different variables and assumptions impact the outcomes of a model.
- 10.2.4 Write informative summaries of strengths and weaknesses of a model.

As discussed in Time Series and Forecasting, modeling is the process of creating a mathematical representation of real-world phenomena to allow for predictions and insights based on data. Modeling involves selecting appropriate algorithms, training the model on historical data, and validating its performance to ensure it generalizes well to new, unseen data. Prior to developing models, it is critical to identify and clearly state assumptions and provide justifications for the data collection and analysis strategies used. In this section we present the methods used for assessing how a model's predictions align with the actual observed data and introduce the concept of model validation. We also briefly discuss approaches used to understanding the robustness of different models and share practices to support documentation and feedback loops for a balanced and transparent analysis.

Stating Assumptions and Justifications

A key element in data science report writing is the ability to identify and document significant *assumptions* that emerge during the foundation of a project or task. In general, an **assumption** is a statement that is thought to be true without being verified or proven. Assumptions often stem from one's experience, accumulated knowledge, or even intuition. For data scientists, assumptions have a very specific meaning: they are foundational hypotheses or beliefs about the structure, relationships, or distribution of data that guide the analytical approach and model selection for a project. These assumptions play an important role in steering the decision-making process and shaping the solution's development. However, it is important to note that assumptions come with their own risks. If the assumptions turn out to be false or inaccurate, they can lead to bias, errors, delays, or outright failures in the project. Therefore, documenting assumptions in a clear and explicit manner is of utmost importance. Providing justifications for each assumption is equally crucial.

Assumptions vs. Constraints

Assumptions should be documented within the methodology section of the report. When it comes to writing the assumptions section, several best practices should be followed. First, it is important to distinguish between assumptions and constraints. Assumptions are things we believe to be true but have not yet confirmed, while **constraints** are limitations or restrictions that are imposed on a project or its solution. A common data science–related assumption is the belief that the historical data used to train a predictive model is representative of future conditions, implying that patterns observed in the past will persist in the future. This assumption underpins the use of time series analysis and forecasting models in predicting future trends based on historical data patterns. By contrast, a common *constraint* encountered is the limited availability of high-quality, relevant data, which can significantly restrict the depth and accuracy of analyses. Additionally, computational limitations, such as processing power and memory capacity, can pose significant challenges in handling large datasets or complex models, impacting the feasibility and scalability of data science projects. Yet other constraints are imposed from the project sponsors, including finding solutions that do not exceed

certain resource requirements or that must attain certain goals.

Using precise and unambiguous language and avoiding any vagueness or generalities helps in accurately stating the assumptions. Assumptions should be verifiable statements, meaning they should be capable of being evaluated or validated with evidence or data. This approach helps data scientists avoid subjective opinions or personal biases.

EXAMPLE 10.2

Problem

A manufacturing company commissioned a report on sales and revenue at different production levels. The company has no more than 13 machines that can be used to make the product, and there are 29 employees that are capable of operating the machines. List the constraints of the project along with any implied assumptions. In addition, formulate follow-up questions that you should ask the executives of the company before you begin any data analysis.

Solution

Constraints:

- The company has a maximum of 13 machines, limiting the number of products that can be produced simultaneously.
- There is also a limit on the number of employees, with at most 29 being capable of operating the machines.

Assumptions:

- All 13 machines can be used if there are enough employees to operate them. The 29 employees are sufficient to operate all 13 machines simultaneously. (Note: We do not know how many employees are required to operate a machine, so this needs to be verified.)
- Machines and operators are assumed to have uniform characteristics, meaning that the volume of production depends only on the number of machines and operators available.
- Production will not be interrupted by downtime of either the machines or the employees. The market demand is sufficient to absorb the production levels at various stages, meaning all produced goods can be sold.

(These are only some of the assumptions that could be listed.)

Follow-up questions for the company:

- What is the maximum output capacity of each machine per hour/day?
- How many employees are required to operate each machine?
- Do the machines have uniform performance, or are some capable of producing more of the product? If so, are additional employees required to work the higher-performing machines (e.g., some machines may be larger and require more operators, producing more of the product as a result)?
- What is the average uptime and downtime for each machine?
- How often are machines maintained? Are there any specific times when machines are not operational (e.g., scheduled maintenance, shift changes)?
- Regarding the employees, are there scheduling factors, differentiated expertise levels, or any other factors that must be considered?
- Are there any supply chain or market limitations?

(Many more questions may be formulated beyond this basic sample of inquiries.)

Nonconstant Assumptions

In addition to the above practices, it is essential to include relevant time frames for the validity of the assumptions. This involves specifying when the assumptions are applicable and noting any changes or updates that might impact them. For example, it may be assumed that sales for a new tech product will be very high only in the months leading up to the holiday season.

Providing rationales or context—that is, explaining the basis for each assumption and how it relates to the project's objectives, requirements, or scope—is also crucial. External factors that may influence the assumptions, such as market conditions, customer preferences, or legal regulations, should be considered. These factors could change over time, and so it is important to note that in the modeling process.

Moreover, acknowledging any resource limitations, like budget, staff, equipment, or materials, is vital as these may affect the project's execution or outcome. Technical limitations, including those related to computing, software, hardware, or network, should also be recognized as they can impact the project's functionality or performance and may also change in the future.

For each assumption made, the technical writer should offer a justification supported by evidence, data, or logical reasoning. For example, we might assume that a software product will be compatible with the Microsoft Windows 11 operating system because market research data shows that Windows 11 is the most widely used operating system among the target audience. Another assumption could be that users will have basic computer literacy and familiarity with similar software products based on customer feedback surveys indicating that they are professionals in the field who have used previous versions of the software or related products from competitors. We might also assume that users will have access to online help and support resources since the software product includes a built-in help feature linking to the company's website, where FAQs, tutorials, videos, and contact information are available. Finally, the assumption that the software product will receive regular updates to fix bugs and improve features can be justified by referring to the company's quality assurance policy and customer satisfaction strategy.

To assess whether technical and data science–related assumptions are being met, rigorous validation techniques such as cross-validation and sensitivity analysis are often employed. (This will be discussed further below.) Careful evaluation and validation of these assumptions help to prevent biased results and enhance the generalizability of the model to various datasets and real-world scenarios. Additionally, continuous monitoring and updating of models with new data help in adjusting to changes that were not anticipated by the initial assumptions, ensuring models remain accurate and relevant over time.

Implicit Bias

It is also important to examine ethical considerations related to assumptions and implicit bias in developing data science reports (see Ethics Throughout the Data Science Cycle), as these factors can significantly impact the fairness and accuracy of the findings. Assumptions, if not rigorously validated, can lead to skewed results that misrepresent the data and its implications. Implicit bias, whether in data collection, analysis, or interpretation, can introduce systemic errors that disproportionately affect certain groups, leading to inequitable outcomes. Therefore, it is necessary to examine and challenge assumptions, employ diverse and representative datasets, and implement techniques to detect and mitigate bias. Often, the best approach is to allow a third party to vet the methodology, identifying assumptions that you may have overlooked. Transparency in methodology and a commitment to ethical standards ensure that data science reports are not only scientifically sound but also socially responsible, promoting trust and integrity in data-informed suggestions and decision-making.

By thoroughly documenting these assumptions and their justifications, technical writers can ensure that the assumptions align with the project's scope and requirements. Additionally, this documentation provides a framework for these assumptions to be validated or revised as necessary throughout the project lifecycle.

Interpreting Measures of Fit

As we saw in Time Series and Forecasting, What Is Machine Learning?, and Deep Learning and AI Basics, models need to be evaluated using appropriate *measures of fit*—that is, the metrics that assess how well a model's predictions align with the actual observed data. Metrics such as R-squared in regression or accuracy in classification are integral in this process. However, it is not sufficient to merely state these values; a comprehensive technical report must explore their meanings and limitations within the context of the specific problem being addressed. High values for measures of fit are not always indicative of a good model. Conversely, low error values may not always mean that the model will do a good job on future data. There are further considerations to explore, such as the bias-variance trade-off, the impact of outliers on the results, and potential for small changes in input to produce relatively large deviations in output (sensitivity). When reporting on the metrics for model fit, it is vital to put these values into their proper context, include accounting for the model's significance, avoiding overfitting, and ensuring robustness.

Validation Assessment

A model of statistical validation can help strengthen the technical report and support consumer confidence with the inclusion of the information. However, in certain instances the results of the validity of fitness testing may not result in favorable outcomes. To maintain and convey integrity in the work, it is essential to report the results in an honest and accurate manner, even when those results are not favorable. In addition, one may consider bringing on an external evaluator to provide an extra level of validation assessment. The external practitioner may provide additional technical expertise to confirm practices or make recommendations for adjustment.

Recall from Correlation and Linear Regression Analysis that, r^2, or *R-squared*, also known as *the coefficient of determination*, measures the proportion of the variance in the dependent variable that is predictable from the independent variables. Consider an example of a linear model in which the factors (feature variables) are square footage of the house, number of bathrooms, proximity to a city center, local school district ranking, etc., while the response (dependent variable) is the market value of the house. After training, the linear regression model has an r^2 value of 0.75. In simple terms, this value suggests that 75% of the variation in market value can be explained by the features used in the model (e.g., size, number of bathrooms, location) and that 25% of the variation is due to noise and other factors not accounted for in the model. This seems to imply that the model is relatively effective at capturing the relationships between the independent variables and house prices, but it is important to understand what this means precisely. It does not mean that the model predicts house prices with 75% accuracy. Instead, it suggests that a significant portion (75%) of the variance in housing prices is captured by the model.

In the context of housing prices, an r^2 of 0.75 might be considered strong, given the high variability in real estate markets due to numerous factors, some of which may not be included in the model (e.g., economic trends, unquantifiable factors like scenic views). It is crucial to remember that a high r^2 value does not imply causation. In other words, we may not conclude that the local school district ranking necessarily *causes* house prices to go up. Any correlation between housing prices and school district ranking may be due to some other factor, such as median income of the residents in the neighborhood.

Additionally, r^2 cannot determine whether the coefficient estimates and predictions are unbiased, which is why we must also assess other diagnostic measures and look at residuals to check for patterns that might indicate issues with the model. Perhaps there are interactions between variables that should be included in the model, implying that a quadratic model might be more appropriate than a linear one, for example. Moreover, the features might explain some of the remaining variability in the response; however, this estimate may be impacted by the sample size, presence of outliers, and the extent to which model assumptions are met.

If the analysis had used a different set of features or a different kind of model (e.g., decision tree), the results may have ended up with a different degree of accuracy. Comparing different models for the same set of data

can help in appreciating the impact of different features on the prediction accuracy. By understanding that our model explains a significant portion of the variance, we can use it to make informed decisions about housing prices. However, recognizing its limitations and the factors it does not account for is crucial in avoiding overreliance on the model for price prediction. Future improvements could include adding more relevant features, using different modeling techniques, and/or collecting more data to improve the model's accuracy and reliability.

EXAMPLE 10.3

Problem

A data science team is examining the diabetes dataset diabetes_data.csv (https://openstax.org/r/spreadsheetsd) utilizing a classification decision tree (see Decision-Making Using Machine Learning Basics) to better understand how factors such as cholesterol level, sex, height, weight, BMI, and others correlate with the presence of diabetes in patients. Use modeling best practices to evaluate the model and discuss how overfitting will be avoided.

Solution

PYTHON CODE

```python
#load libraries for decision trees and visualization

import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

#load data. The code for your dataset will be dependent upon the file path you
#set up
df = pd.read_csv('diabetes_data.csv', index_col='Patient number')
```

When preparing data to run various analyses or models, it is recommended to examine the data characteristics to ensure that the data is properly formatted and has the appropriate data types for the analysis or modeling technique. The function `df.info()` provides the list of features of the data set (column names) and their types (Dtype).

PYTHON CODE

```python
#examine the dataset
df.info()
```

The resulting output will look like this:

```
Data columns (total 13 columns):
#    Column        Non-Null Count  Dtype
---   ------        --------------  -----
0    Cholesterol   390 non-null    int64
1    Glucose       390 non-null    int64
2    HDL Chol      390 non-null    int64
3    Age           390 non-null    int64
4    Gender        390 non-null    object
5    Height        390 non-null    int64
6    Weight        390 non-null    int64
7    BMI           390 non-null    float64
8    Systolic BP   390 non-null    int64
9    Diastolic BP  390 non-null    int64
10   waist         390 non-null    int64
11   hip           390 non-null    int64
12   Diabetes      390 non-null    object
dtypes: float64(1), int64(10), object(2)
memory usage: 42.7+ KBInt64Index: 390 entries, 1 to 390Data columns (total 13
columns):#    Column        Non-Null
Count  Dtype---   ------        --------------  -----0    Cholesterol   390 non-
null    int641    Glucose       390 non-null    int642    HDL Chol      390 non-
null    int643    Age           390 non-null    int644    Gender        390 non-
null    object5    Height        390 non-null    int646    Weight        390 non-
null    int647    BMI           390 non-null    float648    Systolic BP   390
non-null    int649    Diastolic BP  390 non-null    int6410   waist         390
non-null    int6411   hip           390 non-null    int6412   Diabetes      390
non-null    objectdtypes: float64(1), int64(10), object(2)memory usage: 42.7+ KB
```

To use the sklearn (https://openstax.org/r/scikitlearn) library, all data must be numeric. However, if you have columns with string data, there are several methods to convert them to numeric values. For “True”/“False” values, simply use `.astype ('int')`. For other strings, you can employ the `pandas` functions like `map`, `replace`, or `apply`.`map` is generally the fastest and most efficient of these, but it lacks flexibility, as it converts unmatched values to NaN. `replace` is more versatile, allowing partial or full data replacement, but it's slower than `map`. `map` is preferable for large datasets or in production due to its speed.

PYTHON CODE

```
#convert categorical data to numeric values
df['Gender'] = df['Gender'].replace({'male': 0, 'female': 1})
```

To prepare the data for modeling, you will set up features and targets. In addition, you will want to establish a set of training and testing data.

PYTHON CODE

```
#set up data features and targets and set up training and testing datasets
features = df.drop('Diabetes', axis=1)
targets = df['Diabetes']

x_train, x_test, y_train, y_test = train_test_split(features, targets,
stratify=targets, random_state=42)
```

The modeling technique used is a decision tree, which will operate similarly to logistic regression in sklearn as you create the class and then use the fit method. It has the same score method and other methods like predict.

PYTHON CODE

```
#run a decision tree
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

print(dt.score(x_train, y_train))
print(dt.score(x_test, y_test))
```

The resulting output will look like:

```
1.0
0.826530612244898
```

The accuracy observed on the training set is clean, registering at 100%, whereas the accuracy on the test set is significantly lower, recorded at 87.7%. This discrepancy is a signal of overfitting. Doing further analysis by examining the depth of the tree and visualizing it would be beneficial.

PYTHON CODE

```
#measure the depth of the tree
dt.get_depth()
```

The resulting output will look like:

```
7
```

Seeing how deep the tree got and the number of samples in the leaf nodes illustrates that the model is likely overfitting on the data. We can restrict the number of levels (depth) with max_depth. You can try a few values below and see how it changes. Here, we settled on two since that results in nearly equal train/test scores and looks like it reduces the overfitting. Pruning can also be applied to decision trees; as defined in Decision Trees, *pruning* involves the removal of sections of the tree that provide little predictive power to improve the model's accuracy and prevent overfitting.

PYTHON CODE

```
#adjust the depth of the decision tree to 2
dt = DecisionTreeClassifier(max_depth=2)
dt.fit(x_train, y_train)

print(dt.score(x_train, y_train))
print(dt.score(x_test, y_test))
```

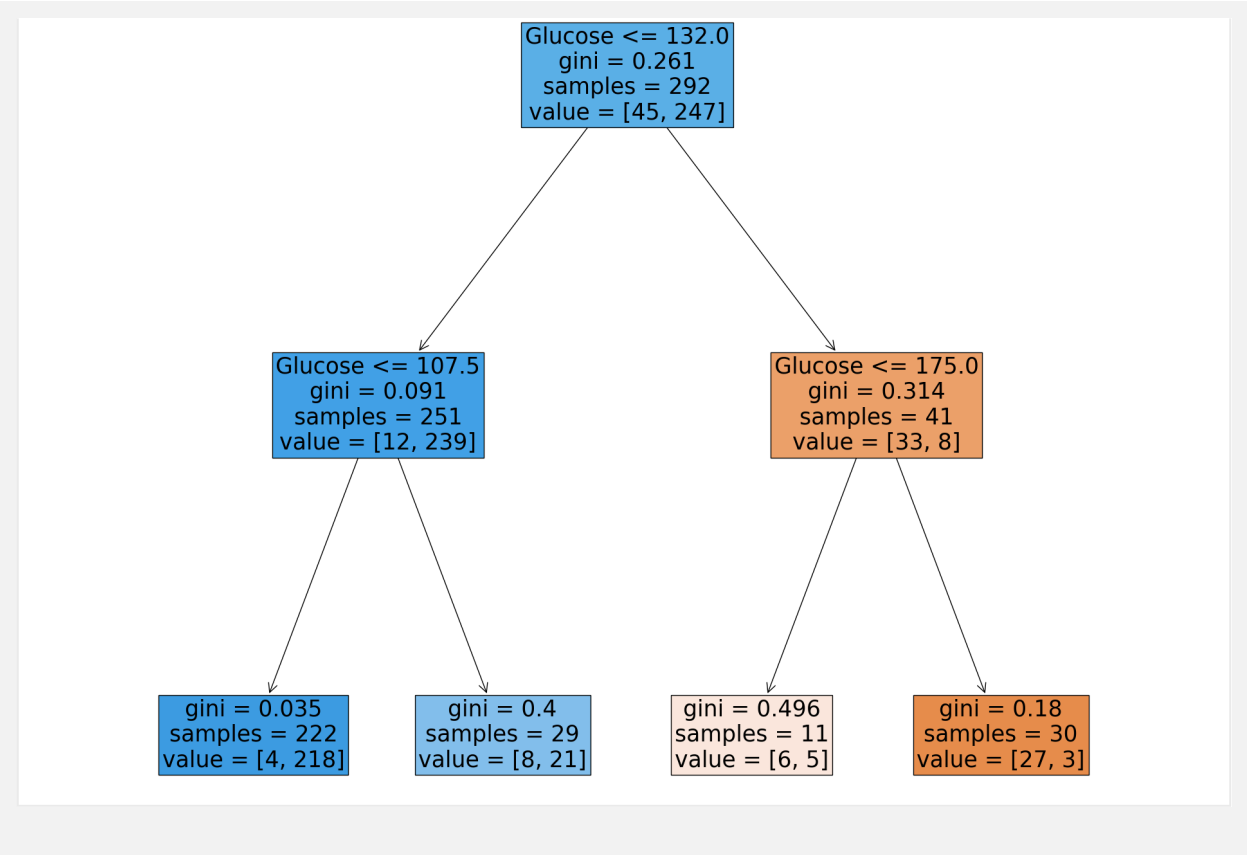
The resulting output will look like:

```
0.9315068493150684
0.9081632653061225
```

PYTHON CODE

```
#create a decision tree with a max depth of 2
f = plt.figure(figsize=(8, 8))
_ = plot_tree(dt, fontsize=10, feature_names=features.columns.tolist(),
filled=True)
```

The resulting output will look like:



Validation Techniques

Suppose that your data science team has come up with multiple models, each of which seems to do well on the training and testing sets. How might you choose between the various models? This is where *validation* plays a role. **Validation** is a broad term for the evaluation of multiple predictive models and/or fine-tuning of certain constants that affect the performance of a model (**hyperparameter tuning**). These techniques provide a means to rigorously test the model's performance on unseen data, thereby reducing overfitting and ensuring that the model captures as much of the true underlying patterns in the data as possible rather than noise or random patterns. Be sure to fully explain all validation methods used and how you selected the best models in the data exploration and analysis part of your report.

Validation typically involves setting aside a portion of the data that is used to test multiple different models after training. This is different than splitting the data into a training set and testing set (as covered in Decision-Making Using Machine Learning Basics), in which the model is trained only once on the training set and then evaluated on the test set. When *validation* is used, the dataset is split into three parts, *training*, *validation,* and *testing sets.* After a few rounds of training and validation, the best model is chosen, and then finally the testing set is used on the best model to find a measure of fit. Thorough validation processes are crucial for building robust models that perform well in real-world applications, as they allow for model selection and fine-tuning before deployment, ultimately leading to more reliable and effective predictive analytics.

Cross-validation, a variation on validation, divides the training set into multiple subsets and iteratively trains and evaluates the model on different combinations of these subsets, offering a more comprehensive evaluation by leveraging all available data. Generally, cross-validation will be done on each model under consideration, which may take considerable time and resources to accomplish. Although time-consuming, cross-validation helps to prevent fitting to random effects in the data.

One simple way to do cross-validation is the *bootstrap method* (discussed in some detail in Machine Learning

in Regression Analysis). Bootstrapping is a powerful statistical technique used to estimate the distribution of a sample by repeatedly resampling with replacement. In the context of model evaluation, the bootstrap method involves creating multiple new datasets, or **bootstrap samples**, by randomly sampling from the original dataset, allowing duplicates. Each bootstrap sample is used to train and evaluate the model, and the performance metrics are averaged over all samples to provide an estimate of model accuracy and variability. This may be done on multiple different models and values of hyperparameters to arrive at the best model. The bootstrap method is highly flexible and can be applied to a wide range of statistical problems, but it can be computationally intensive, especially with large datasets. Nevertheless, it is a valuable tool for obtaining robust estimates of model performance and understanding the variability in predictions.

Additionally, there are two common cross-validation strategies called *k-fold cross-validation* and *leave-one-out cross-validation*.

k-fold cross-validation works by dividing the dataset into k equally sized subsets, or **folds**. The model is trained on $k - 1$ of the folds and tested on the remaining fold, a process that is repeated k times with each fold serving as the test set once. This approach ensures that every data point is used for both training and validation, providing a more comprehensive assessment of the model's generalizability. This method is particularly advantageous when the dataset is limited in size, as it maximizes the use of available data for both training and testing.

Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k is set to the number of data points in the dataset, meaning that each fold contains only one observation. In LOOCV, the model is trained on all data points except one, which is used as the validation set, and this process is repeated for each data point in the dataset. This method provides an exhaustive validation mechanism, ensuring that every single data point is used for testing exactly once, thus offering an unbiased evaluation of the model's performance. However, LOOCV can be computationally expensive, especially for large datasets, because it requires training the model as many times as there are data points. Despite its computational intensity, LOOCV is particularly useful for small datasets where retaining as much training data as possible for each iteration is crucial for model accuracy.

Accuracy and Error

Once you have settled on a validation strategy, how do you evaluate the performance of your models so that you can select the best one? In previous chapters, we have introduced numerous measures of fit, some of which are useful in classification tasks, while others are more suitable for regression tasks, as we'll see below.

Classification Accuracy

Recall from What Is Machine Learning? that *accuracy* is calculated as the proportion of correctly predicted instances out of the total instances in a classification task. For instance, consider the question of using a classification model to predict whether an email is spam or not based on its content and metadata. If the model correctly predicts 90 out of 100 emails as spam or not spam, the accuracy is 90%. While 90% appears to be a high level of accuracy, it is necessary to delve deeper into the understanding of this metric. In cases where the data set is imbalanced, such as when 95% of emails are not spam, an accuracy of 90% might not be as impressive. This scenario could mean that the model is merely predicting the majority class (non-spam) most of the time.

To gain a more nuanced understanding of the model's performance, additional measures like *precision*, *recall*, and the F1 score are considered. We saw in What Is Machine Learning? that *precision* measures the proportion of correctly identified positive instances (true positives) among all instances that the model predicts as positive—that is, $TP/(TP + FP)$. On the other hand, *recall* measures the proportion of true positives among all positives in dataset—that is, $TP/(TP + FP)$. Finally, the F_1*-score* provides a balance between precision and recall, offering a single metric to assess the model's accuracy in cases where there is a trade-off between the two.

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

For example, a high precision with a low recall might indicate that the model is too conservative, only labeling an email as spam if it is very sure and leading to much spam being missed.

It is also vital to examine the confusion matrix, which provides a detailed breakdown of the model's predictions, showing the number of true positives, false positives, true negatives, and false negatives (see Classification Using Machine Learning). The *confusion matrix* helps to explain the nature of errors the model is making. For instance, a high number of false positives (non-spam emails classified as spam) might be more undesirable than false negatives in an email classification scenario, as it could lead to important emails being missed. Therefore, while accuracy gives an initial indication of the model's performance, a comprehensive evaluation requires analyzing additional metrics and understanding their implications in the specific context of the data and the problem at hand. This approach ensures that decisions based on the model's predictions are informed and reliable, considering not just the overall accuracy but also the nature and implications of the errors it makes.

Regression Error Measures

For regression models, measures of error are used to evaluate model performance, including those discussed in Time Series and Forecasting and Decision-Making Using Machine Learning Basics: *mean absolute error (MAE), mean square error (MSE), mean absolute percentage error (MAPE),* and *root mean square error (RMSE)*, in addition to statistical measures of fit such as *R-squared* or the *Pearson correlation coefficient (r)*, which are detailed in Statistical Inference and Confidence Intervals. Adding one more measure to this list, consider the **mean percentage error (MPE)**, which is a metric used to assess the accuracy of predictions in a regression model by calculating the average of the percentage errors between predicted $\left(\hat{y}_i\right)$ and actual (y_i) values.

$$\text{MPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i}$$

The mean percentage error (MPE) is like the MAPE except that the absolute value of the difference is not used. Instead, MPE will tend to be smaller than MAPE when the predictions are balanced between higher than actual and lower than actual. However, one drawback of MPE is its susceptibility to skewness from large percentage errors, especially when actual values are close to zero, potentially leading to misleading results. Additionally, MPE can yield negative values if predictions consistently overestimate or underestimate the actual values, complicating the interpretation. Despite these limitations, MPE can offer valuable insights when used in conjunction with other error metrics, contributing to a more nuanced understanding of model performance.

AIC and BIC

There are two related measures that are often used in validation, the *Akaike Information Criterion (AIC)* and *Bayesian Information Criterion (BIC)*. Most statistical software packages will compute AIC and BIC, but it is helpful to see the formulas. Let n be the number of data points, k the number of parameters (or degrees of freedom), and L the maximum likelihood of the model (see Classification Using Machine Learning).

$$\begin{aligned} \text{AIC} &= 2k - 2\ln(L) \\ \text{BIC} &= k\ln(n) - 2\ln(L) \end{aligned}$$

Both AIC and BIC are criteria used for model selection and validation in the context of statistical models. They help in choosing the best model among a set of candidate models by balancing goodness of fit and model complexity. BIC imposes a harsher penalty on the number of parameters compared to AIC and so will select simpler models overall. During the model selection process, various candidate models are fitted to the data. AIC and BIC values are computed for each model, and the model with the lowest criterion value is selected:

PYTHON CODE

```
 # cross validation with Bayesian Information Criterion

from sklearn.model_selection import cross_val_score, GridSearchCV
import numpy as np # Import numpy

# Define the decision tree classifier
dt = DecisionTreeClassifier()

# Perform cross-validation with 5 folds
scores = cross_val_score(dt, features, targets, cv=5)

# Print the average accuracy
print(f"Average accuracy over 5 folds: {scores.mean():.2f}")

# Define the parameter grid for grid search
param_grid = {
  'max_depth': range(1, 10),
  'min_samples_leaf': range(1, 10),
  'min_samples_split': range(2, 10)
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(dt, param_grid, cv=5)
grid_search.fit(features, targets)

# Print the best parameters and score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best score: {grid_search.best_score_:.2f}")

# Print the BIC score for the best model
best_model = grid_search.best_estimator_
bic = len(features) * np.log(len(targets)) - 2 * best_model.score(features,
targets) # Now np is defined
print(f"BIC score: {bic:.2f}")
```

The resulting output will look like this:

```
Average accuracy over 5 folds: 0.41
Best parameters: {'max_depth': 1, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best score: 0.72
BIC score: 2165.56
```

The goal of BIC is to balance the fit of the model with its complexity, helping to avoid overfitting. A lower BIC score indicates a better model, as it suggests that the model has a good fit to the data while being relatively simple. For instance, the BIC score of 2,165.56 would be considered good if it is lower than the BIC scores of

competing models, indicating it provides a better balance between goodness of fit and complexity.

Sensitivity Analysis

Sensitivity analysis is an analytical technique used to determine how different sources of uncertainty in the inputs of a model or system affect its output. Generally speaking, sensitivity analysis is accomplished by varying inputs or parameters slightly—by up to 5%, for example—and then examining how the output changes. If the output varies too much under small changes of input, then the model may be unstable, requiring different modeling techniques to improve stability.

Particularly in data science, where complex models with numerous variables and sources of uncertainty are common, sensitivity analysis is invaluable. For instance, it quantifies the relationship between dataset size and model performance in a specific prediction problem, aiding in determining the requisite data volume for achieving desired accuracy levels. Additionally, it identifies key input variables that significantly influence the model output, thereby focusing attention on the most pertinent features and parameters for the problem domain.

Sensitivity analysis in data science serves multiple purposes such as identifying influential input variables, assessing the model's robustness against input variations, and uncovering potential biases. Various methods are employed for conducting sensitivity analysis, tailored to the model's complexity and the required detail level. Below are a few examples of commonly used sensitivity analysis techniques.

One-way sensitivity analysis examines how changes in one input parameter at a time affect the outcome of a model. It is useful for identifying the impact of individual variables on the result, allowing decision-makers to understand which single factors are most critical.

Multi-way sensitivity analysis explores the effects of simultaneous changes in multiple input parameters on the outcome. This approach helps in understanding the interactions between different variables and how they collectively influence the model's results.

Scenario analysis evaluates the outcomes under different predefined sets of input parameters, representing possible future states or scenarios. It aids in assessing the resilience of a model's outcomes under various hypothetical conditions, offering insights into potential risks and opportunities.

Monte Carlo simulation uses random sampling and statistical modeling to estimate the probability of different outcomes under uncertainty. This method provides a comprehensive view of the potential variability in results, helping decision-makers gauge the range of possible outcomes and the likelihood of various scenarios.

EXAMPLE 10.4

Problem

To assess the sensitivity of outcomes of the diabetes models as mentioned above, a Monte Carlo simulation was applied to the data. What is the average accuracy over 1,000 simulations?

Solution

PYTHON CODE

```
import random
```

```
# Define the number of simulations
num_simulations = 1000

# Initialize a list to store the results
results = []

# Loop through the number of simulations
for i in range(num_simulations):
  # Randomly split the data into training and testing sets
  x_train, x_test, y_train, y_test = train_test_split(features, targets,
stratify=targets, random_state=random.randint(0, 1000))

  # Create a decision tree classifier
  model = DecisionTreeClassifier(random_state=random.randint(0, 1000))

  # Train the model on the training set
  model.fit(x_train, y_train)

  # Predict on the testing set
  y_pred = model.predict(x_test)

  # Calculate the accuracy
  accuracy = (y_pred == y_test).mean()

  # Store the accuracy in the results list
  results.append(accuracy)

# Calculate the average accuracy across all simulations
average_accuracy = sum(results) / len(results)

# Print the average accuracy
print(f"Average accuracy over {num_simulations} simulations:
{average_accuracy:.2f}")
```

The resulting output will look like:

```
Average accuracy over 1000 simulations: 0.75
```

The average accuracy across the simulations showed an accuracy of 75%. Monte Carlo simulations further enhance understanding by presenting a comprehensive view of the model's response to varied input distributions, highlighting the model's overall uncertainty and potential outliers. Through such analyses, valuable insights emerge about the housing price prediction model. These might include the predominant influence of location on price, sensitivity to data outliers, or diminishing returns from additional bedrooms beyond a certain threshold. This information becomes crucial in refining the model, guiding data collection strategies, and informing more accurate real estate pricing decisions.

Sensitivity analysis supports decision-making processes across various domains to build an understanding of how different inputs influence the outcomes of their models or projects. By carefully changing important factors and watching how results shift, sensitivity analysis makes it easier to see how different inputs are linked to outputs. This process gives a better understanding of which factors matter the most for the end results. Sensitivity analysis is an important component of the data analysis section of your technical report, demonstrating how robust the models are, thereby strengthening the trust that project sponsors have in your chosen modeling method.

Documenting Strengths and Weaknesses

Every model has strengths and weaknesses. When documenting strengths or weaknesses within the results and discussion portion of a data science report, it is crucial to be clear about the results in a manner that is supported by the data or information that led to the conclusion. The documentation should be in a standalone section in the technical report to draw attention to the matter.

Strengths

Strengths should be specific and meaningful. For example, if the dataset is very comprehensive, you might write: “The dataset includes detailed information on 50,000 housing transactions over the past decade, which provides a robust basis for understanding market trends.” If you want to highlight part of the modeling process that emphasizes accuracy, robustness, etc., you might write “Implementing a random forest model consisting of 1,000 decision trees. A 10-fold cross-validation approach was employed to validate the model performance, ensuring that the results are not overfitted to the training data.”

Overemphasizing or fabricating strengths should be avoided at all costs. Going back to the example of email spam detection, if the model shows 90% accuracy but the training dataset is very imbalanced, with over 95% of messages being non-spam, then it would be misleading to report the 90% accuracy as a strength of the model.

Weaknesses

No one likes to admit weaknesses; however, in the context of a data science project, listing real or potential weaknesses is an essential part of writing an effective and informative report. Weakness statements should be as specific as possible and should indicate steps taken to address the weaknesses as well as suggest ways in which future work might avoid the weakness. Weaknesses may arise from having limited or biased data, datasets that are missing many values, reliance on models that are prone to either underfitting or overfitting, limited scope, or assumptions and simplifications made that could potentially introduce bias and inaccuracy of the model.

An example of an informative weakness statement is: “Approximately 15% of the records had missing values for key variables such as square footage and number of bedrooms, which were imputed using median values, potentially introducing bias.” This provides detailed information (15% of records have missing values) and actions taken to try to address the problem (median values imputed), and there is an implied suggestion that if a more complete dataset could be found, then this weakness could be mitigated.

Using Peer Feedback

Peer feedback for data science reports can strengthen the deliverable by offering diverse perspectives and expert insights, identifying potential weaknesses, biases, or areas for improvement that the original author might have overlooked. For example, a peer might suggest that additional validation techniques be employed to confirm the robustness of a model's predictions, or individuals could point out a more effective method for visualizing complex data. Incorporating such feedback can lead to more accurate and reliable results, enhancing the overall credibility of the report. In addition, peers can help ensure that the report's language and structure are clear and accessible, making it easier for a broader audience to understand and apply the findings. An example of peer feedback might be “Consider using cross-validation to further validate your model's performance and include a comparison with alternative models to provide a comprehensive analysis.”

For those who are providing feedback, it is best to use constructive criticism, acknowledging limitations while proposing potential solutions or future research directions. Instead of outlining weaknesses, emphasize opportunities for improvement, suggesting alternative modeling approaches, data acquisition strategies, or bias mitigation techniques. Providing suggestions for future analyses can lead to advancement on the topic area being analyzed.

EXAMPLE 10.5

Problem

To better understand consumer behavior and retention, the marketing director of an online clothing retailer requested that the data science team develop a model to analyze customer churn. (*Customer churn* refers to the rate at which customers stop using a company's products or services over a specific period.) The data science team created a model using logistic regression based on the internal data of the organization. While the model achieved an accuracy of 85%, it exhibited a tendency to misclassify a significant portion of customers who were at high risk of churning based on the results of incoming data. As a supervisor of the data science team, what steps would you take to help the data science team increase accuracy and communicate the current findings to a marketing director?

Solution

Steps to provide constructive feedback among the data science team.

- Create a collaborative space and collective approach to examine the factors impacting model.
- Acknowledge the limitation by sharing the findings of the most recent data as it compares to the model.
- Focus the conversation on the model enhancement, not the team performance.
- Support the conversation for enhancement through the provision of suggestions (e.g., incorporate additional customer behavior data, external factors, data science techniques) or questions (e.g., data availability, sources).
- Review and agree on the options for next steps moving forward.

Steps to provide constructive feedback to the marketing director.

- Describe the dataset that was used to develop the model and also share why the selected data science technique was used without being overly technical.
- Share the limitations of the dataset and logistic regression technique.
- Present the results of the model with a data visual such as a confusion matrix.
- Invite opportunity for the marketing director to provide suggestions for additional data to be used.
- Provide suggestions on other data science techniques available based on the recommended data.
- Agree on next steps for model development.

10.3 Effective Executive Summaries

LEARNING OUTCOMES:

By the end of this section, you should be able to:

- 10.3.1 Define executive summaries, noting their important features.
- 10.3.2 Determine which details of the modeling process would be most important to include in an effectively written executive summary.
- 10.3.3 Write an executive summary that includes actionable advice based on the results of the model.

The executive summary serves as an essential first impression, succinctly transforming complex technical analyses into an accessible and engaging synopsis for an audience of varied backgrounds. The summary should begin with a clear introduction to the business problem or research question that the project addresses. This introduction must be crafted in a manner that is comprehensible to a nontechnical audience, aiming to capture their interest by emphasizing the potential impact of the findings.

An executive summary should provide an overview of the data sources utilized and the methodologies applied, focusing on the novelty or relevance of the approach. In the case of a technical data science report, the use of advanced machine learning techniques or unique data-gathering methods should be highlighted, avoiding overly technical language. The core of the summary is the presentation of key findings and insights, articulated in a manner that is both lucid and actionable. Instead of merely presenting statistics, the emphasis should be on their interpretation, relevance to the initial query, and their practical implications in the business or research domain. The conclusion should be a persuasive call to action, suggesting further research or the practical application of the model, designed to encourage the reader to engage with the full report and explore the depth of the project's contributions.

What Details Need to Be Included?

An **executive summary** is a concise, standalone document that encapsulates the essence of a data science report. This summary is the first element that readers encounter, and it plays a critical part in setting the tone for the entire document. The structure of the executive summary is designed to guide the reader through a logical progression of ideas. It typically begins with a clear statement of the problem or research question, setting the context for the report. This is followed by a brief description of the data sources and methodologies used, highlighting any innovative or advanced techniques employed in the analysis. The summary then succinctly presents the key findings and insights derived from the data, focusing on those that are most relevant and impactful. Complex data and technical details are distilled into understandable terms, ensuring that the summary remains engaging and informative without being overwhelming.

The executive summary concludes with recommendations or conclusions drawn from the analysis. These recommendations need to be presented clearly and persuasively, emphasizing their relevance and potential impact. The summary is also characterized by its brevity and clarity, typically spanning no more than a page or two, ensuring that it can be quickly read and understood. The language used must be straightforward and jargon-free, catering to the diverse backgrounds of the report's readership. Overall, the executive summary serves as an effective tool for communicating the value and implications of a data science report, bridging the gap between technical analysis and strategic decision-making.

Presenting Actionable Advice

Actionable advice in an executive summary provides specific recommendations and guidance to practitioners related to how the results can be used to share information to build knowledge or support decisions. The actionable advice should not be directive but presented as concepts to consider or even venues to share the information. Depending upon the objective(s), topic(s), and level of detail, one may recommend specific divisions, departments, or units in the organization where the report may be best utilized; the advice should be clearly linked to the data and analysis, offering concrete steps that can be implemented to achieve desired outcomes. Consider the order of the suggestions based on their perceived impact and feasibility within the organizational landscape (e.g., culture, politics, priorities).

For example, suppose that you have made a model for home prices. Your analysis may suggest the following actionable advice: “Homes with recent renovations show a significant increase in market value. Focus on updating kitchens, bathrooms, and other key areas to maximize return on investment.” By providing well-founded, practical suggestions, the executive summary not only informs but also empowers individuals to make informed decisions.

Executive Summary Example

Below is an example of an executive summary report in Python.

PYTHON CODE

```
!pip install fpdf
from fpdf import FPDF

# Define a custom PDF class that inherits from FPDF
class PDF(FPDF):
  def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.set_margins(25.4, 25.4, 25.4) # Set 1-inch margins (25.4 mm)

# Define the header method
  def header(self):
    self.set_font('Arial', 'B', 12) # Set the font for the header
    self.cell(0, 10, 'Executive Summary', 0, 1, 'C') # Add the header text

# Define a method to add chapter titles
  def chapter_title(self, title):
    self.set_font('Arial', 'B', 10) # Set the font for chapter titles
    self.cell(0, 10, title, 0, 1, 'L') # Add the chapter title
    self.ln(2) # Adjust the line break for single spacing

# Define a method to add chapter body text
  def chapter_body(self, body):
    self.set_font('Arial', '', 10) # Set the font for the body text
    self.multi_cell(0, 5, body) # Add the body text with decreased line spacing
    self.ln(2) # Adjust the line break for single spacing

# Define a function to create the executive summary
def create_executive_summary(title, sections):
  pdf = PDF() # Create a PDF object
  pdf.add_page() # Add a page
  pdf.chapter_title(title) # Add the main title
  for section_title, section_body in sections.items(): # Loop through sections
    pdf.chapter_title(section_title) # Add section title
    pdf.chapter_body(section_body) # Add section body text
  pdf.output('executive_summary.pdf') # Generate and save the PDF to current
directory

# Content of the executive summary
title = 'Applying Data Science Techniques to Detect Diabetes'
sections = {
  'Purpose of the Study': 'This research study aimed to analyze patient health data
```

```
and physical characteristics to detect diabetes using a classification decision
tree model. The primary goal is to identify key predictors of diabetes and develop
an accurate model for early detection and diagnosis. The study utilized a dataset
comprising health metrics (i.e., cholesterol, glucose, HDL cholesterol, systolic
BP, diastolic BP) and physical characteristics (i.e., sex, age, height, weight,
BMI, waist, hip) of patients. A classification decision tree model was employed to
analyze the data, leveraging its ability to handle both numerical and categorical
data, interpretability, and effectiveness in identifying significant predictors of
diabetes.',
  'Findings': 'The study found that glucose levels, BMI, and waist circumference
are the most significant predictors of diabetes using a classification decision
tree model. This model achieved high accuracy (91%) in detecting diabetes,
indicating its effectiveness in correctly identifying both diabetic and nondiabetic
patients. While sex and age were included, they had less impact compared to the
health metrics. The research highlights the potential of this model to improve
early diabetes detection and suggests further refinement and integration into
clinical practice, alongside public health initiatives, to manage obesity and
glucose levels for diabetes prevention.',
  'Recommendation': 'Based on the findings of the study, we recommend integrating
the classification decision tree model into clinical practice to enhance early
detection and diagnosis of diabetes. By utilizing this model, health care providers
can identify at-risk individuals more effectively, enabling timely intervention and
management. Additionally, further research is encouraged to refine the model and
explore the inclusion of other relevant health metrics and demographic factors to
improve its predictive accuracy. The findings should also inform public health
initiatives aimed at reducing obesity and managing glucose levels, thereby
contributing to diabetes prevention and overall population health improvement.'
}

create_executive_summary(title, sections)
```

The resulting output will look like:

Executive Summary

Applying Data Science Techniques to Detect Diabetes

Purpose of the Study

This research study aimed to analyze patient health data and physical characteristics to detect diabetes using a classification decision tree model. The primary goal is to identify key predictors of diabetes and develop an accurate model for early detection and diagnosis. The study utilized a dataset comprising health metrics (i.e., cholesterol, glucose, HDL cholesterol, systolic BP, diastolic BP) and physical characteristics (i.e., sex, age, height, weight, BMI, waist, hip) of patients. A classification decision tree model was employed to analyze the data, leveraging its ability to handle both numerical and categorical data, interpretability, and effectiveness in identifying significant predictors of diabetes.

Findings

The study found that glucose levels, BMI, and waist circumference are the most significant predictors of diabetes using a classification decision tree model. This model achieved high accuracy (91%) in detecting diabetes, indicating its effectiveness in correctly identifying both diabetic and nondiabetic patients. While sex and age were included, they had less impact compared to the health metrics. The research highlights the potential of this model to improve early diabetes detection and suggests further refinement and integration into clinical practice, alongside public health initiatives to manage obesity and glucose levels for diabetes prevention.

Recommendation

Based on the findings of the study, we recommend integrating the classification decision tree model into clinical practice to enhance early detection and diagnosis of diabetes. By utilizing this model, health care providers can identify at-risk individuals more effectively, enabling timely intervention and management. Additionally, further research is encouraged to refine the model and explore the inclusion of other relevant health metrics and demographic factors to improve its predictive accuracy. The findings should also inform public health initiatives aimed at reducing obesity and managing glucose levels, thereby contributing to diabetes prevention and overall population health improvement.

EXPLORING FURTHER

Additional Resources

Many useful articles and books have been written about technical writing and communicating data science findings. A brief list is included here, and we encourage readers to continue their learning about this important phase of the data science process:

Bettes, S. (2019). Audience. In M. Beilfus, S. Bettes, & K. Peterson (Eds.), *Technical and professional writing genres: A study in theory and practice*. OKState. https://open.library.okstate.edu/technicalandprofessionalwriting/chapter/chapter-2/ (https://openstax.org/r/okstate)

Brownlee, J. (2021, February 1). *Sensitivity analysis of dataset size vs. model performance*. Machine Learning Mastery. https://machinelearningmastery.com/sensitivity-analysis-of-dataset-size-vs-model-performance/ (https://openstax.org/r/machinelearningmastery)

Hotz, N. (2024, April 5). *15 data science documentation best practices*. Data Science and AI Project Management Blog. https://www.datascience-pm.com/documentation-best-practices (https://openstax.org/r/datasciencepm)

Intelligent Editing. (2021, June 9). *How to understand your audience in technical writing*. PerfectIt. https://intelligentediting.com/blog/how-to-understand-your-audience-in-technical-writing (https://openstax.org/r/intelligentediting)

Kazakoff, M. (2023, May). *Delivering the facts*. The Actuary. https://www.theactuary.com/2023/05/04/delivering-facts (https://openstax.org/r/theactuary)

Marino, V., & Dragan, D. (2018, July). Science communication with diverse audiences. *Adult Development &*

Aging News. https://www.apadivisions.org/division-20/publications/newsletters/adult-development/2018/07/sharing-research-effectively (https://openstax.org/r/apadivisions)

Datasets

Note: The primary datasets referenced in the chapter code may also be downloaded here (https://openstax.org/r/drivefolders).

Key Terms

actionable advice recommendations or guidance in a report, particularly in an executive summary, providing practical ways to apply the report's findings or insights

alt text brief descriptions of images that accompany the image or may be embedded within the image data, meant to aid readers who are not able to view the images and graphics due to disability or other limitations

assumption statement that is thought to be true without being verified or proven; foundational hypotheses or beliefs about the structure, relationships, or distribution of data that guide the analytical approach and model selection for a project

audience the person or group that will be reading the report

bootstrap samples multiple samples taken from a dataset with duplicates permitted

codebook data dictionary to detail the variables, their units, values, and labels within the dataset used in the project

constraint limitation or restriction that is imposed on a project or its solution

cross-validation validation method that divides the training set into multiple subsets and iteratively trains and evaluates the model on different combinations of these subsets

executive (or data) dashboard a visual representation tool designed to provide a quick, real-time overview of an organization's key performance indicators (KPIs) and metrics to senior management, often updated in real time or at regular intervals

executive summary concise, standalone document that encapsulates the essence of a data science report

executives decision-makers such as managers, CEOs, or others who may not have detailed technical knowledge but need an understanding of the implications of the information for strategic decision-making

experts individuals with an advanced understanding of the subject matter, often with specialized knowledge or education in the topic at hand

fold one of a set of equally sized subsets used in k-fold cross-validation

hyperparameter tuning fine-tuning of certain constants that affect the performance of a model

jargon specialized and/or technical terms in a given field

k-fold cross-validation cross-validation strategy that works by dividing the dataset into k equally sized subsets or folds, where the model is trained on $k-1$ of the folds and tested on the remaining fold, a process that is repeated k times with each fold serving as the test set once

key performance indicators (KPIs) quantifiable metrics used to evaluate the success of an organization, employee, or process in achieving specific objectives and goals

layered approach writing strategy in which a report is organized into sections or appendices that provide different levels of detail and complexity for different audiences

leave-one-out cross-validation (LOOCV) special case of k-fold cross-validation where k is set to the number of data points in the dataset so that the model is trained on all data points except one, which is used as the validation set, and this process is repeated for each data point in the dataset

mean percentage error (MPE) average of the percentage errors between predicted $\left(\hat{y}_i\right)$ and actual (y_i) value: $\text{MPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i-\hat{y}_i}{y_i}$

Monte Carlo simulation random sampling and statistical modeling to estimate the probability of different outcomes under uncertainty

multi-way sensitivity analysis explores the effects of simultaneous changes in multiple input parameters on the outcome

neurodiversity normal variation of individual cognitive abilities, especially in realms of communication and processing in formation

nonspecialists nonexpert audience without specialized knowledge of the subject but with some need to understand the basics of a data science report for a particular reason

one-way sensitivity analysis examines how changes in one input parameter at a time affect the outcome of a model

scenario analysis evaluates the outcomes under different predefined sets of input parameters, representing possible future states or scenarios

sensitivity analysis technique used to determine how different values of an independent variable affect a particular dependent variable under a given set of assumptions

technicians practical users of information in a data science report who may apply the knowledge in a hands-on manner

validation evaluation of multiple predictive models and/or hyperparameter tuning

version control system a software tool that helps keep track of changes to files, code, or any type of digital content over time

Group Project

You are working in a business analyst role on a team that provides analytical support for a marketing department at a large grocery retailer. Sales are currently decreasing, and the marketing director needs to provide a data-informed strategy to support sales growth. Your group's objective is to build a classification model that maximizes profit for the upcoming marketing campaign aimed at selling a new line of food items. The goal is to develop a classification model that can be applied to the entire customer base to identify and target customers most likely to purchase the new items, thus increasing the campaign's profitability. Additionally, the marketing director is interested in understanding the characteristics of customers who are inclined to buy these specific new items.

Part 1

Explore the ifood-data-business-analyst (https://openstax.org/r/spreadsheetsi) dataset and use summary tables and/or data visuals to provide insights to better understand the characteristics of the sample respondents and summarize the customer segmentation based on their behaviors. Collaborate as a group; for example, one team member could work on descriptive statistics while another member could create graphs and charts, and another member may write explanations to accompany the data and visuals.

Part 2

Create a classification model to identify approaches and factors that can maximize the profit of the upcoming marketing campaign. Be sure to validate and test your model. Report on the measures of fit and justify why your model is effective.

Part 3

Write a 1-page executive summary for the marketing director with your findings and recommendations. Include strengths and weaknesses of your model and modeling process.

Chapter Review

1. In a technical data science report, what type of communication is most suitable for executives?
   a. Highly technical and detailed
   b. Focused on practical application
   c. Concise with a focus on summaries and outcomes
   d. Broad and nontechnical for general understanding

2. What should a technical report for nonspecialists avoid?
   a. Complex statistical data
   b. Technical jargon
   c. Visual aids like graphs and charts
   d. Descriptions of practical applications

3. Which is a key consideration when writing technical data science reports?
   a. Using complex language to demonstrate expertise
   b. Connecting with diverse audiences through tailored communication
   c. Focusing only on the most advanced techniques
   d. Avoiding visual aids to prevent confusion

4. For technicians, a data science report should primarily focus on:
   a. Theoretical knowledge
   b. Practical application and procedures
   c. Advanced research and new findings
   d. Business implications and strategies

5. What is the purpose of a layered approach in a data science report?
   a. To provide different levels of detail for various audiences
   b. To simplify the report writing process
   c. To focus solely on the technical aspects
   d. To avoid including any technical language

6. What is the primary role of visual aids like graphs and tables in a data science report?
   a. To replace written content
   b. To add aesthetic appeal to the report
   c. To clarify and accentuate significant data insights
   d. To demonstrate the author's technical skills

7. In a technical report, how should graphs and tables be presented to the reader?
   a. In a complex and detailed manner
   b. Randomly placed throughout the report
   c. Without any accompanying explanations
   d. With clear labels, titles, and straightforward explanations

8. What is the primary purpose of documenting assumptions in a data science report?
   a. To prove the assumptions are correct
   b. To steer the decision-making process
   c. To increase the complexity of the report
   d. To show the expertise of the report writer

9. What is a key characteristic of a well-stated assumption in a technical report?
   a. It is based on personal opinion.
   b. It is specific and verifiable.
   c. It is vague and general.
   d. It is always proven to be true.

10. In the context of measuring house prices, what does an R-squared value of 0.75 indicate?
   a. The model predicts housing prices with 75% accuracy.
   b. Seventy-five percent of the variance in housing prices is explained by the model.
   c. The model is 75% reliable.
   d. The model will increase housing prices by 75%.

11. Why is it important to consider additional metrics beyond accuracy in a classification model?
   a. To comply with industry standards

b. To understand the model's performance more deeply
c. To make the report longer
d. To use up more computational resources

12. What is the purpose of conducting a sensitivity analysis in a data science project?
a. To test the robustness of results against input variability
b. To reduce the number of input variables
c. To increase the complexity of the model
d. To make the report more appealing

13. What is the primary purpose of an executive summary in a data science report?
a. To present complex technical analyses in a detailed manner
b. To provide a comprehensive background of the data scientists
c. To summarize complex analyses for an audience of varied backgrounds
d. To list all the data sources and methodologies in detail

14. 14. In a data science report, what should the executive summary ideally begin with?
a. A detailed description of the data
b. An introduction to the business problem or research question
c. Advanced statistical formulas
d. The names and credentials of the data scientists

15. What aspect of the methodologies should be highlighted in the executive summary of a technical data science report?
a. The complexity of the methodologies
b. The novelty or relevance of the approach
c. Every step in the methodological process
d. The technical language used in the methodologies

16. How should the executive summary in a data science report conclude?
a. With a persuasive call to action
b. By summarizing the data sources used
c. With a detailed technical explanation
d. By introducing new topics

Critical Thinking

1. How can a technical writer ensure that their data science report is accessible to both experts and nonspecialists?
2. What strategies should be employed to make complex data science findings understandable for an executive audience?
3. In the context of data science reporting, what are the benefits of using a version control system, and how does it contribute to the report's quality and reliability?
4. Describe how visual aids like graphs and tables can be effectively utilized in a data science report to communicate complex data insights to different audience types.
5. What are the key considerations a technical writer should keep in mind when documenting and describing methods in a data science project, and why are these considerations important?
6. How can a technical writer effectively document and justify the assumptions made in a data science

project?

7. What are the key considerations when interpreting the R-squared value in a regression analysis, and why are they important?
8. In what ways can sensitivity analysis enhance the understanding and robustness of a data science model?
9. How can a technical writer accurately convey the limitations and strengths of a data science model in a report, and what is the significance of this practice?
10. What role does a confusion matrix play in evaluating a classification model, and how should its findings be interpreted in a technical report?
11. How can a technical writer effectively balance technical detail and accessibility in an executive summary for a data science report?
12. What strategies should be employed in an executive summary to make the findings of a data science report actionable and relevant for strategic decision-making?
13. In what ways can the actionable advice in an executive summary be tailored to different audience types or departments to maximize the report's utility and impact?

References

Willems, J. P., Saunders, J. T., Hunt, D. E., & Schorling, J. B. (1997). Prevalence of coronary heart disease risk factors among rural blacks: A community-based study. *Southern Medical Journal*, *90*(8), 814–820. https://doi.org/10.1097/00007611-199708000-00008

A Appendix A: Review of Excel for Data Science

Microsoft® Excel® is a popular spreadsheet program developed by Microsoft Corporation. Excel is one of the first and simplest spreadsheet tools developed to help with the manipulation and analysis of data. It includes a range of functionalities that cater to the diverse needs of many data science applications. You are likely to have used Excel in some form or other—perhaps to organize the possible roommate combinations in your dorm room or to plan for a party or in some instructional context.

Excel was first released by Microsoft in 1987, and it has become the most popular choice for loading and analyzing tabular data.

Basic Data Analysis Using Excel

A screenshot of the main Excel menu when it is open is provided in Figure A1. As you see, Excel consists of a series of submenus such as Home, Insert, Draw, Page Layout, Formulas, etc. Clicking on each Menu item opens a submenu with additional functions. For example, clicking on the Formulas menu item opens a submenu as shown in Figure A2.

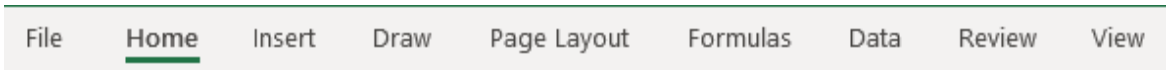


Figure A1 MS Excel Menu (Used with permission from Microsoft)

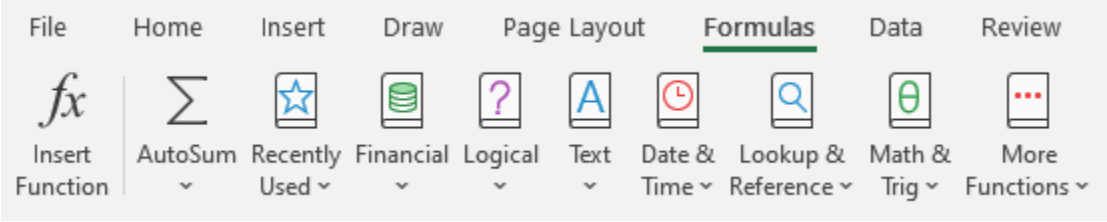


Figure A2 Formulas Menu in MS Excel (Used with permission from Microsoft)

Excel presents data in the form of table with rows and columns. We call each rectangle, defined by the combination of row number and column name (e.g., A1, B3), a **cell**. We can use a mouse pointer to select a specific cell and then edit the data inside. Once a cell is selected, *Namebox* shows the identifier of the selected cell Figure A3.

Formula Bar is used to read/write a formula for a cell, which is identified by Namebox.

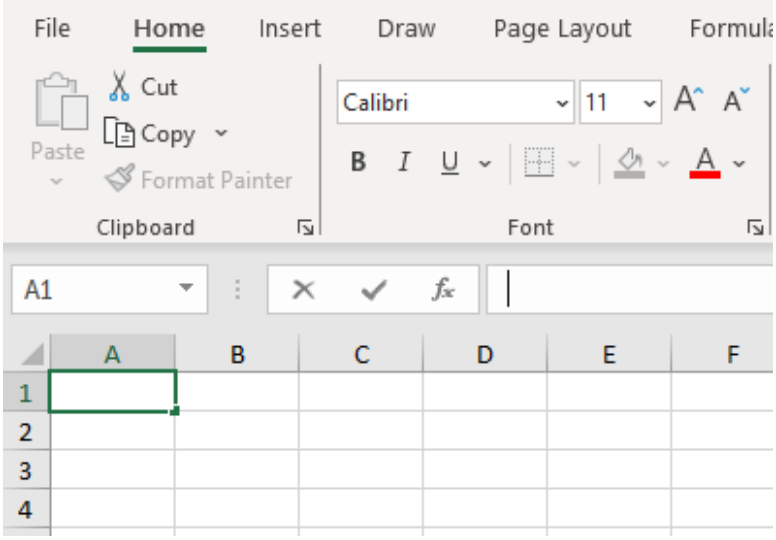
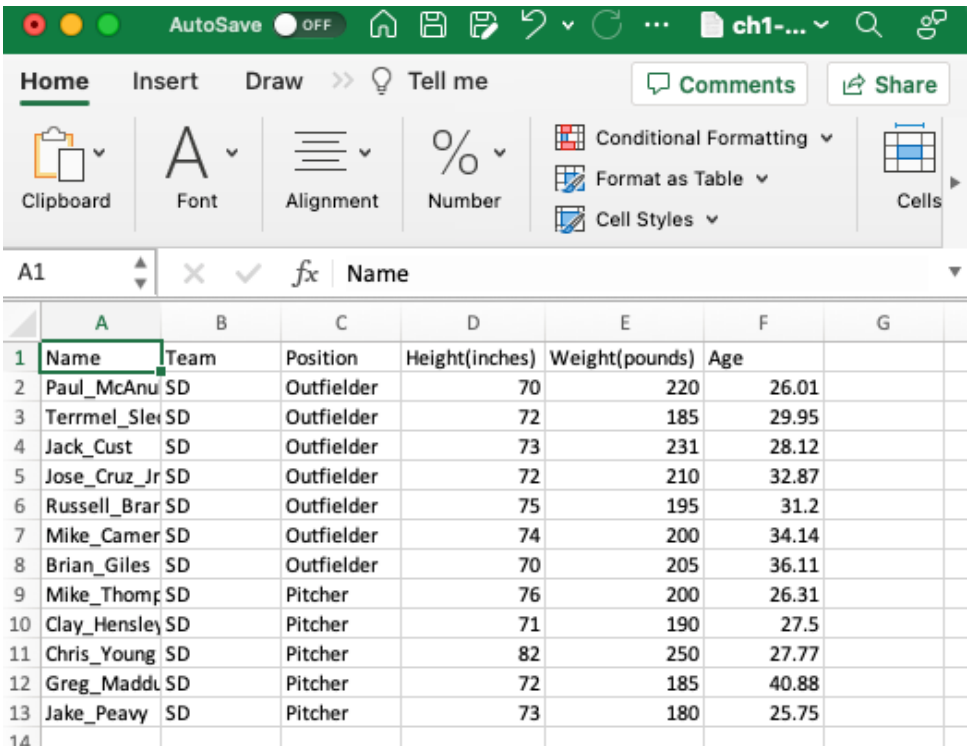


Figure A3 MS Excel Showing Rows and Columns (Used with permission from Microsoft)

Load Data Using Excel

Loading a dataset to Excel can be done simply by clicking File > Open File. That will open up a file explorer window, and you can simply select the CSV file you want to load. We are going to use a part of a public dataset, titled "SOCR- 1035 Records of Heights (in) and Weights (lbs) of Major League Baseball (MLB) Players" (ch1-SOCR-small.csv (https://openstax.org/r/folders14)).

Once loaded, the dataset will show up on the Worksheet Window, as shown in Figure A4.



	A	B	C	D	E	F	G
1	Name	Team	Position	Height(inches)	Weight(pounds)	Age	
2	Paul_McAnu	SD	Outfielder	70	220	26.01	
3	Terrmel_Sle	SD	Outfielder	72	185	29.95	
4	Jack_Cust	SD	Outfielder	73	231	28.12	
5	Jose_Cruz_Jr	SD	Outfielder	72	210	32.87	
6	Russell_Brar	SD	Outfielder	75	195	31.2	
7	Mike_Camer	SD	Outfielder	74	200	34.14	
8	Brian_Giles	SD	Outfielder	70	205	36.11	
9	Mike_Thomp	SD	Pitcher	76	200	26.31	
10	Clay_Hensley	SD	Pitcher	71	190	27.5	
11	Chris_Young	SD	Pitcher	82	250	27.77	
12	Greg_Maddu	SD	Pitcher	72	185	40.88	
13	Jake_Peavy	SD	Pitcher	73	180	25.75	
14							

Figure A4 SOCR-small Dataset Opened with MS Excel (Used with permission from Microsoft)

Summarize Data Using Excel

We will continue to refer to the same dataset about the MLB players. For simplicity, only 12 items of SD Outfielders and Pitchers are included in Figure A5.

Name	Team	Position	Height (inches)	Weight (pounds)	Age
Paul_McAnulty	SD	Outfielder	70	220	26.01
Terrmel_Sledge	SD	Outfielder	72	185	29.95
Jack_Cust	SD	Outfielder	73	231	28.12
Jose_Cruz_Jr.	SD	Outfielder	72	210	32.87
Russell_Branyan	SD	Outfielder	75	195	31.2
Mike_Cameron	SD	Outfielder	74	200	34.14
Brian_Giles	SD	Outfielder	70	205	36.11
Mike_Thompson	SD	Pitcher	76	200	26.31
Clay_Hensley	SD	Pitcher	71	190	27.5
Chris_Young	SD	Pitcher	82	250	27.77
Greg_Maddux	SD	Pitcher	72	185	40.88
Jake_Peavy	SD	Pitcher	73	180	25.75

Figure A5 ch1-SOCR-small.csv Opened with MS Excel

EXAMPLE A.1

Problem

On ch1-SOCR-small.csv (https://openstax.org/r/folders14), write a formula that counts the number of Outfielders.

Solution

The formula is

=COUNTIF(C2:C13, "Outfielder")”

This will count the rows whose position, located at column C, is “Outfielder.” Note that double quotation marks (i.e., “ ”) are used to indicate “Outfielder” is a string. The result is 7.

Search Data Using Excel

You can search data in a table in multiple ways with Excel. We will first explore ways without using Excel formulas, and then we’ll look at some useful formulas.

You can search for the cells containing a certain number or text by using “Find,” as in Figure A6. The Find pop-up window will have a text box in which you can type in any number or text, and then Excel will search and find all the cells that include number/text you typed in.

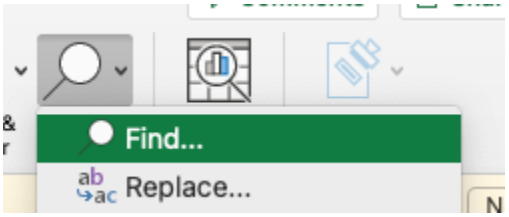


Figure A6 Find Feature in MS Excel (Used with permission from Microsoft)

If you are more interested in viewing all the rows that fulfill certain criteria, you can use "Filter," as in Figure A7. Once Filter is enabled, you will see drop-down arrows on each column of the table. The arrows allow you to select specific values that you are searching for and then presents only the rows with such values.

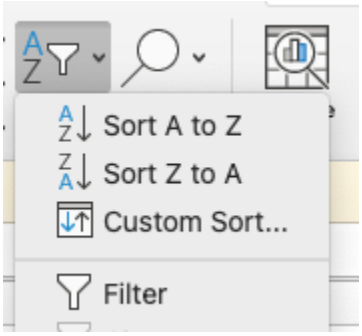


Figure A7 Filter Feature in MS Excel (Used with permission from Microsoft)

For example, if you click "Outfielder" in the Position window, the resulting table will only have the rows whose Position is Outfielder, as shown in Figure A8.

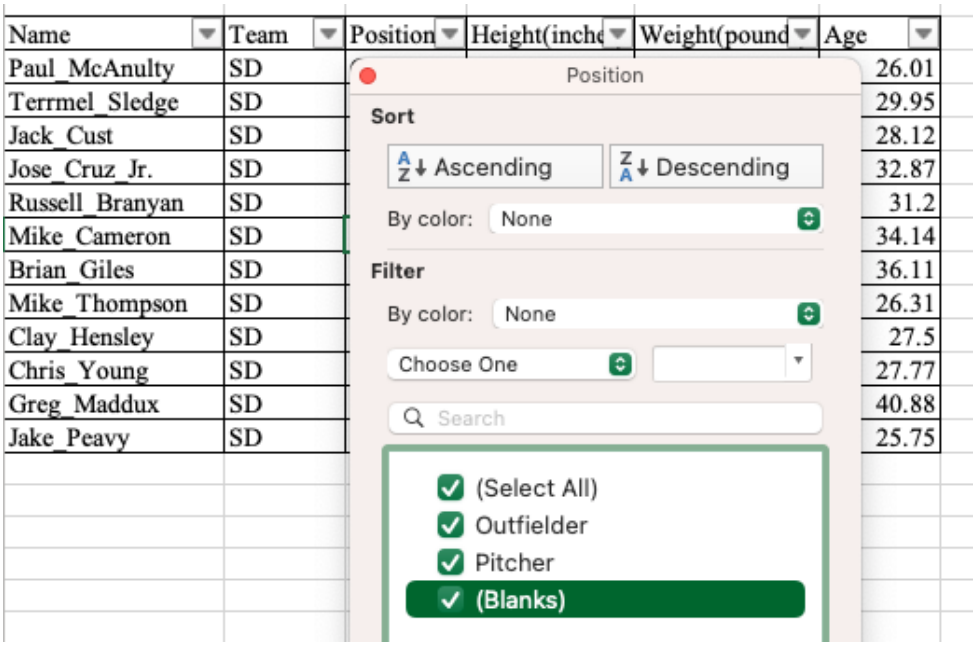


Figure A8 Filter by Position on MS Excel (Used with permission from Microsoft)

Now let's do some more sophisticated searches using some Excel formulas.

INDEX

Index returns a cell value at a specific location, given a range of cells. The location is identified as a relative row and column number within the range. For example, the following formula on ch1-SOCR-small.csv (https://openstax.org/r/folders14) will return 72 since the cell located at the fifth row and second column within the range of C8:E13 will refer to the row 12 and column D—so the cell D12. The value at D12 is 72.

=INDEX(C8:E13,5,2)

EXAMPLE A.2

Problem

On ch1-SOCR-small.csv (https://openstax.org/r/folders14), what value does the following formula return?

=INDEX(A3:D9,7,3)

Solution

This formula finds the seventh row and third column within the range of A3:D9, which indicates C9. Thus, the resulting value is "Outfielder."

MATCH

Given a range of single-column cells, MATCH returns the relative row number at which the cell value matches the value you specify. MATCH takes three arguments—the value to match, the range of search, and the match type. This example illustrates how to locate the value 75 within the cell D2, D3, ..., through D13. The third argument, match type, is set to 0 so that it seeks for an exact match (i.e., exactly 75, not 74 or 75.5).

=MATCH(75, D2:D13, 0)

The formula above returns 5, a relative row number. The relative row number 5 indicates the fifth row within the range, and this corresponds to the absolute row number 6. Notice that D6 has 75.

WHAT HAPPENS IF THERE ARE MULTIPLE MATCHING ROWS?

If there are multiple rows with a match, MATCH only returns the very first instance. For example, this formula will simply return 2 (i.e., the second row within D2:D13, which is D3) even though there are three occurrences of 72 at rows 3, 5, and 12.

=MATCH(72, D2:D13, 0)

EXAMPLE A.3

Problem

Using MATCH, find the *absolute* row number of a first player whose weight is 185 lb on ch1-SOCR-small.csv (https://openstax.org/r/folders14).

Solution

Weight is located in column E. The correct use of MATCH is provided next. It looks through E2, E3, ..., E13 and finds the first exact match of 185. The first occurrence of 185 is in the second row in the range E2:E13, which is E3, so this formula will return 2.

=MATCH(185, E2:E13, 0)

To get the absolute row number, you need to offset the output above with a constant number. Notice that the search range starts from E2, so you should offset the output by +1. Therefore, the final answer is:

=MATCH(185, E2:E13, 0) + 1

VLOOKUP

Given the search range, VLOOKUP locates a specific search key within the first column in terms of the row number and then returns a value at the column of your choice along that row. It requires four arguments—the search key, the range of search, the relative position of the column of your interest, and the match type. For example, the formula below locates the first pitcher along column C from C2 to C13 first. Similarly to MATCH, VLOOKUP supports multiple match types and FALSE indicates the exact match (i.e., not the "partial" match) for VLOOKUP.

=VLOOKUP("Pitcher", C2:E13, 3, FALSE)

Then, the formula reads a specific cell value in the same row. The column number is indicated as the third argument in the formula, called the relative position. It refers to how many columns the search key is away to the right from the column that is specified in the first argument. The first exact match is found at row 2, and the formula reads values in the third column to the right from column C: E. Thus, the value 220 in cell E2 is the result.

EXAMPLE A.4

Problem

Using VLOOKUP, find the age of Mike Thompson on ch1-SOCR-small.csv (https://openstax.org/r/folders14).

Solution

First you need to search for the player Mike Thompson. His row can be located by his name, "Mike_Thompson." Then, the relative position argument of VLOOKUP allows you to read his age. The age column, F, is 6 columns to the left of Name, so the relative position argument should be 6. The fourth argument remains FALSE as you are looking for an exact match.

=VLOOKUP("Mike_Thompson", A2:F13, 6, FALSE)

The formula above will return 26.31.

Basic Visualization and Graphing Using Excel

Excel provides many options for generating graphs and other visualizations to assist in data science investigations.

The menu in Figure A9 shows some of the many options available within Excel for graphing.

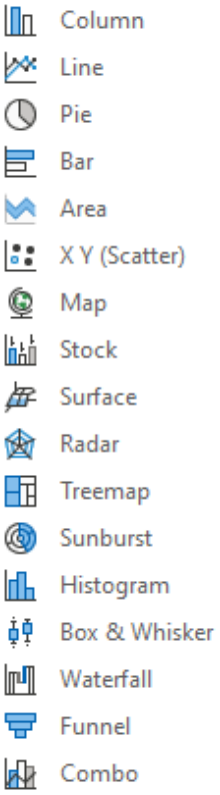


Figure A9 A Menu of Various Graphs Available within MS Excel (Used with permission from Microsoft)

Graphically Represent Data Using Excel

Graphical representation of data, or *data visualization*, can be done easily with Excel. Let's draw a bar chart as an example. You can start by simply clicking Insert > Bar chart > Clustered Column icon (Figure A10).

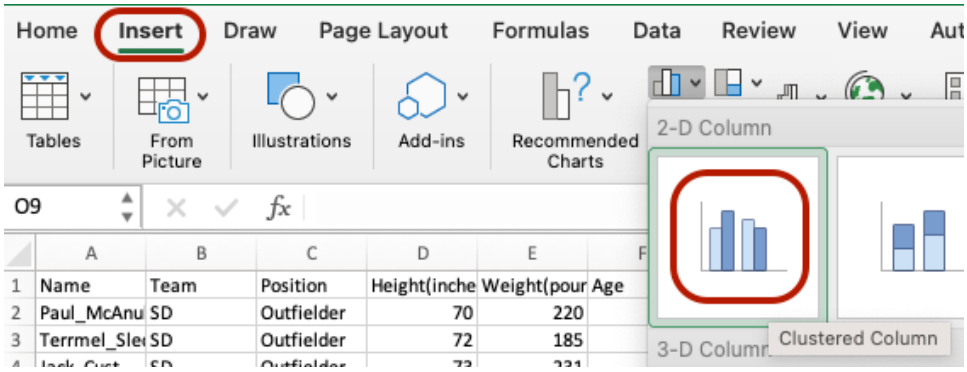


Figure A10 MS Excel's Bar Chart Icon (Used with permission from Microsoft)

Once the icon is clicked, you will see an arbitrary bar chart. It could be just a plain whitespace too—it really depends on what cell was selected when you clicked the icon.

Now click the chart area and it will reveal two new tabs—Chart Design and Format. Go to the Chart Design tab and click Select Data (Figure A11).

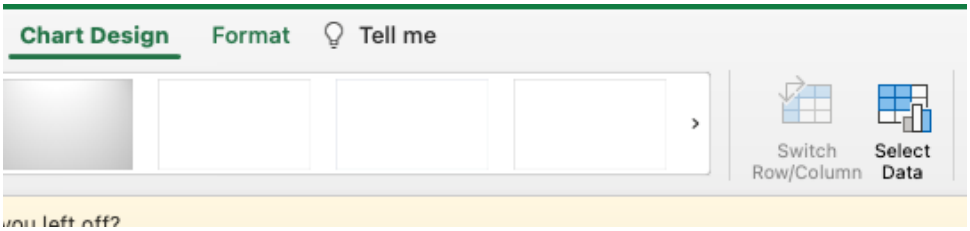


Figure A11 Select Data Menu on MS Excel (Used with permission from Microsoft)

In the Select Data window, define Chart data range by highlighting A1:F13 (i.e., the entire table). This will auto-fill the rest of the pop-up window (Figure A12).

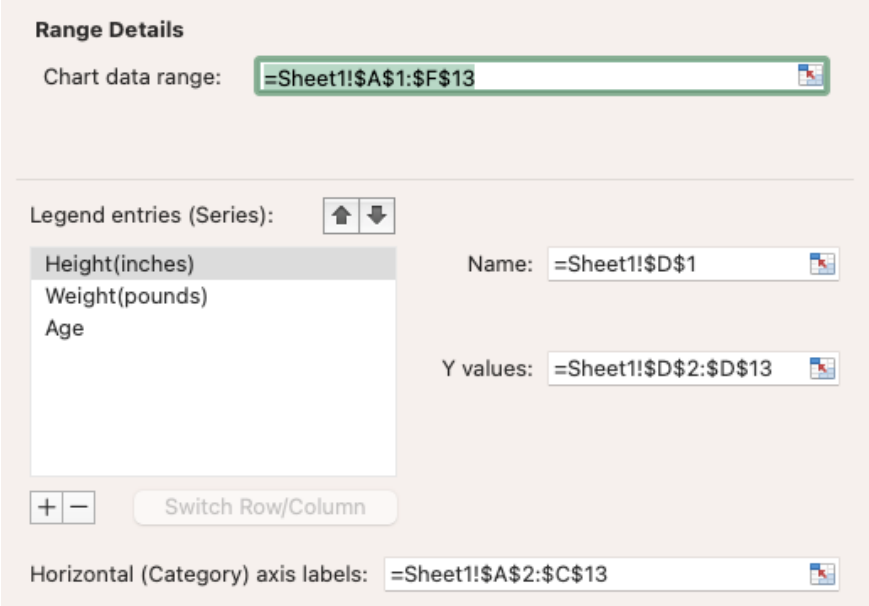


Figure A12 Select Data Pop-Up Window on MS Excel (Used with permission from Microsoft)

Three entries—Height, Weight, Age—refer to each group of bars. Each group will have 13 bars, each of which represents different baseball players. For now, let's plot *only* Height. You can do so by removing the other two by clicking the "-" button under the entry list.

Click Ok and the bar chart is generated as Figure A13. Notice that columns A through C are displayed as the *x*-axis labels, as indicated on the pop-up window (see the Horizontal [Category] axis labels part in Figure A12).

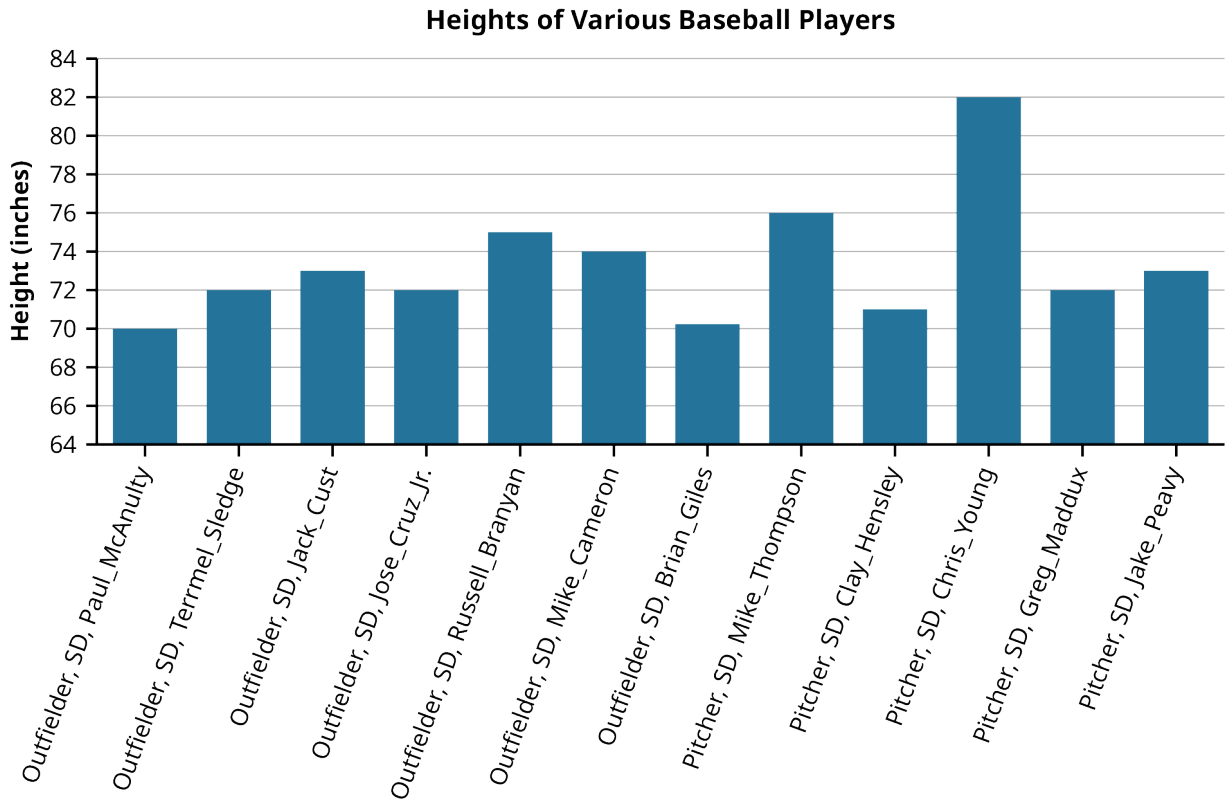


Figure A13 The First Bar Chart with Three Kinds of *x*-Axis Labels (Position, Team, and Name)

If you wish to show only the player names, go back to the Select Data window and update the Horizontal axis labels to be just A2:A13 (Figure A14).

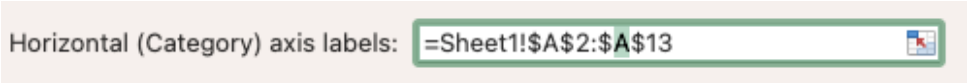


Figure A14 Keeping Only Name on the *x*-Axis Label (Used with permission from Microsoft)

The resulting plot will only have player names along the *x*-axis (Figure A15).

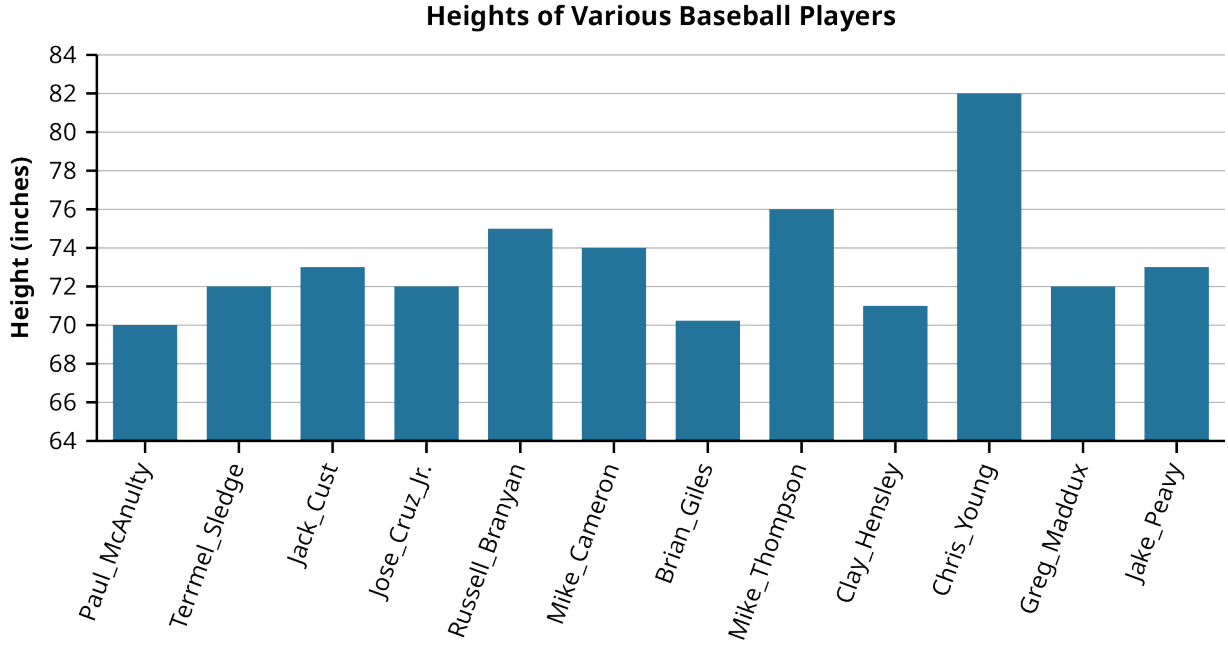


Figure A15 The Second Bar Chart with Only Name Displayed on the *x*-Axis

Changing the formatting of different chart elements is easy in Excel as well. You can 1) click the element to change the text or drag to move the position of the element or 2) double-click the element in the corresponding pop-up menu. Figure A16 shows the pop-up menu for the chart title element. Notice that you can change different aspects of the presentation with the title element, such as Fill and Border.

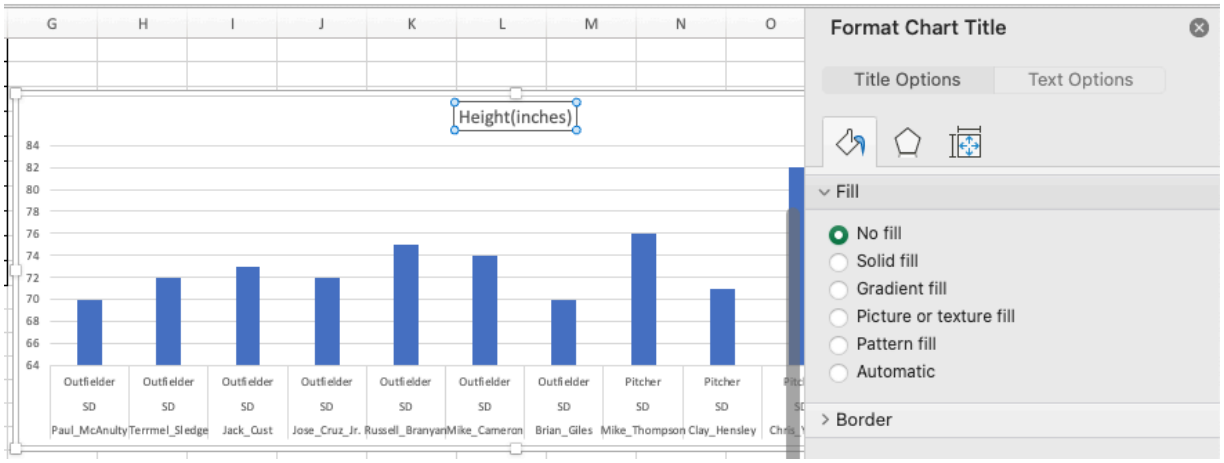


Figure A16 Changing the Formatting of the Bar Chart (Used with permission from Microsoft)

EXAMPLE A.5

Problem

Using MS Excel, draw a scatterplot between Weight (*x*-axis) and Height (*y*-axis). A scatterplot is a plot that depicts each data point as a dot. The location of a dot is typically determined by the x- and y-axis.

Solution

There is an icon for scatterplots under Insert (see Figure A17). On the Select Data pop-up window, select D1:E13 since the weights and heights are in column D and E, as shown in Figure A17 and Figure A18.

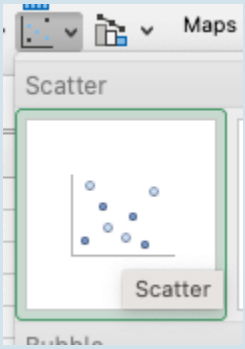


Figure A17 Scatterplot Icon (Used with permission from Microsoft)

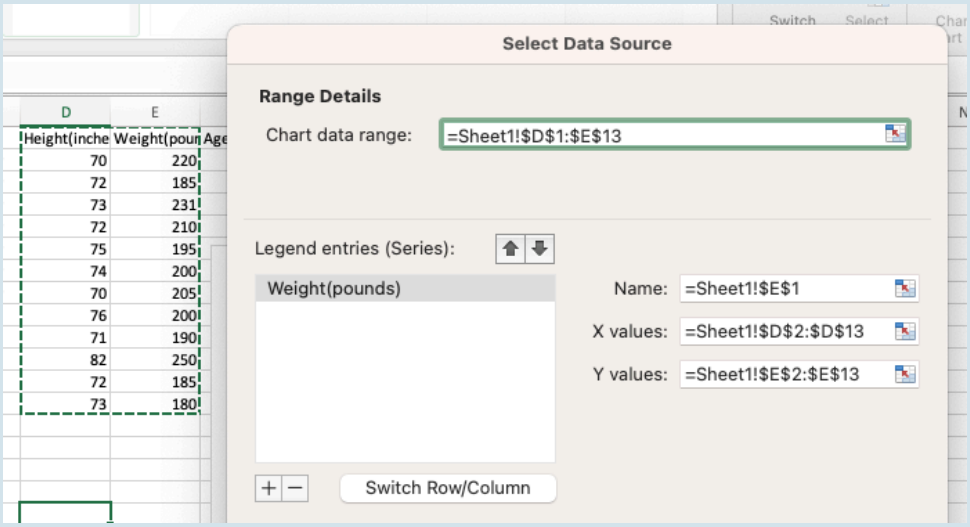


Figure A18 Select Data for Drawing a Scatterplot (Used with permission from Microsoft)

The resulting scatterplot is shown in Figure A19. Notice that the numbers along the *x*-axis are heights, while those along the *y*-axis are weights.

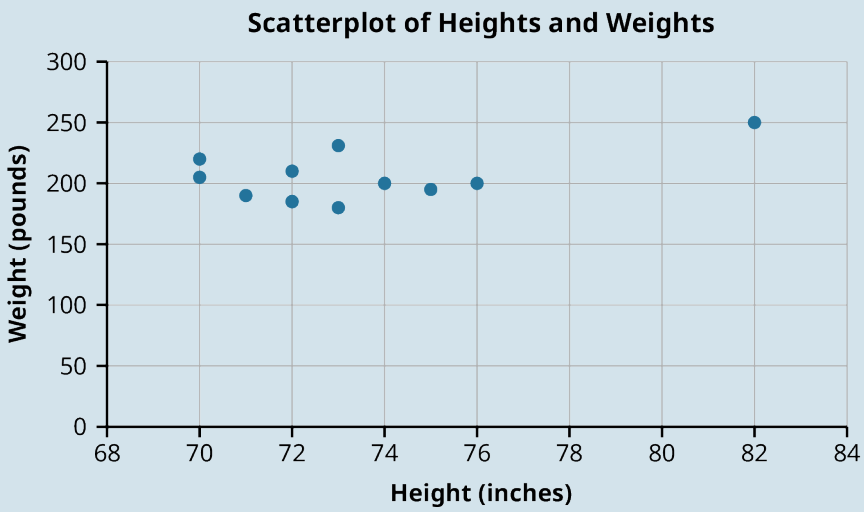


Figure A19 The Initial Scatterplot

You can change the chart title and the axis titles by double-clicking the title text and the axes themselves, as in Figure A20.

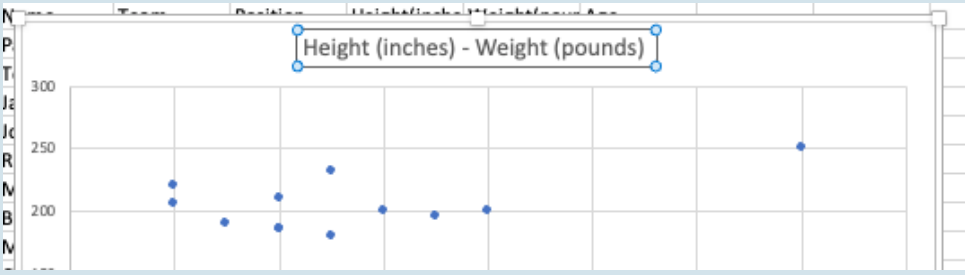


Figure A20 Changing the Title of the Scatterplot (Used with permission from Microsoft)

Once either the *x*- or *y*-axis is double-clicked, a different menu will show up and there you can add axis titles, as illustrated in Figure A21.

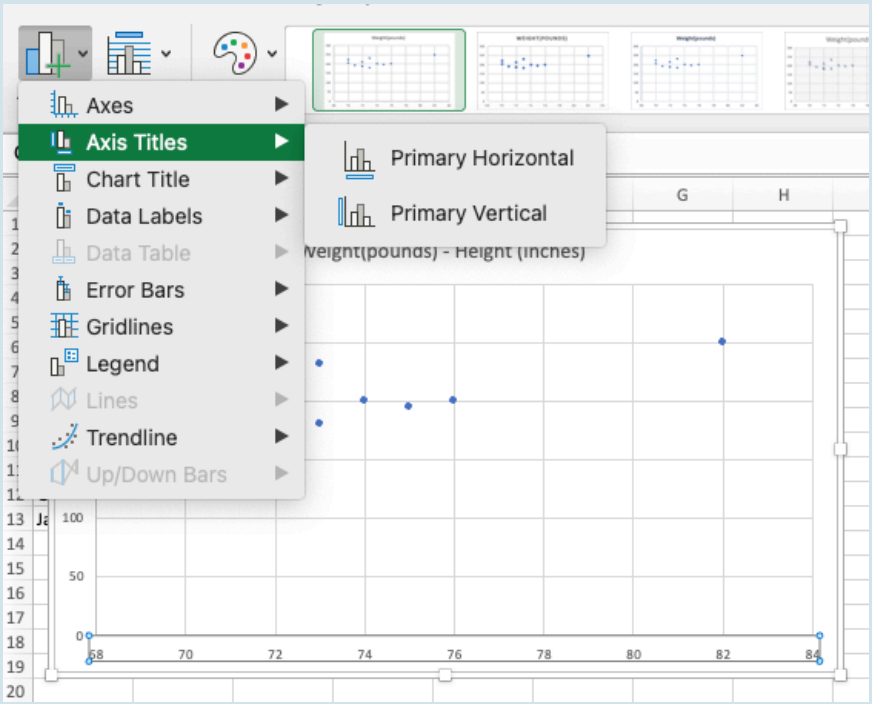


Figure A21 Changing the Axis Title of the Scatterplot (Used with permission from Microsoft)

Basic Statistical Analysis Using Excel

Excel provides many statistical functions and statistical distributions that are used in data science applications.

Excel Analysis for Measures of Center and Dispersion

Excel allows you to type a formula into a cell. Using a formula, you can refer to a value from another cell or produce an output based on a value in a cell (or values from multiple cells). A formula always starts with an equal sign (=). Following is an example formula, calculating an average of three numbers in cells A1, A2, and A3.

= AVERAGE(A1, A2, A3)

Suppose you have numbers in cells A1, A2, and A3 and want to show the average of the three numbers in cell B1. Figure A22 shows how to use AVERAGE in this scenario.

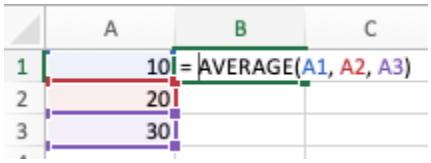


Figure A22 Use of AVERAGE in MS Excel (Used with permission from Microsoft)

Once you are done typing the formula in B1, press enter. The formula in B1 will change to the result of the formula – 20 (see Figure A23).

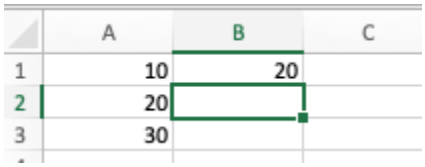


Figure A23 The Result of Using AVERAGE in MS Excel (Used with permission from Microsoft)

EXCEL DOES NOT SAVE THE FORMULA WHEN THE FILE IS SAVED AS CSV

You can use Excel formulas with no problem on a CSV file, but Excel does not save those when the file is saved in the CSV format. If you want to keep the formula you have used, make sure to save your file as XLSX (File > Save As... > Excel Workbook (XLSX)). XLSX stores more information beyond text, such as formulas and drawings.

EXAMPLE A.6

Problem

On ch1-SOCR-small.csv (https://openstax.org/r/folders14), calculate the average height of the MLB players in the dataset using AVERAGE.

Solution

On the worksheet where the MLB Players dataset is open, type this formula in any empty cell and press the enter key (Figure A24).

=AVERAGE(D2:D13)

This will fill the average across cells D2, D3, D4, ..., through D13. The resulting average height is 73.3333.

	A	B	C	D	E	F
1	Name	Team	Position	Height(inches)	Weight(pounds)	Age
2	Paul_McAnulty	SD	Outfielder	70	220	26.01
3	Terrmel_Sledge	SD	Outfielder	72	185	29.95
4	Jack_Cust	SD	Outfielder	73	231	28.12
5	Jose_Cruz_Jr.	SD	Outfielder	72	210	32.87
6	Russell_Branyan	SD	Outfielder	75	195	31.2
7	Mike_Cameron	SD	Outfielder	74	200	34.14
8	Brian_Giles	SD	Outfielder	70	205	36.11
9	Mike_Thompson	SD	Pitcher	76	200	26.31
10	Clay_Hensley	SD	Pitcher	71	190	27.5
11	Chris_Young	SD	Pitcher	82	250	27.77
12	Greg_Maddux	SD	Pitcher	72	185	40.88
13	Jake_Peavy	SD	Pitcher	73	180	25.75
14				=AVERAGE(D2:D13)		

Figure A24 Calculate the Average Height Using AVERAGE in MS Excel (Used with permission from Microsoft)

Other Useful Formulas for Summarizing Data

There are a lot of other ways to use simple Excel formulas to analyze data values. You can type each of the example formulas listed in Table A1 and see how it works.

Formula	Description
=MEDIAN(D2:D13)	Compute median of D2, D3, ..., and D13. Returns 72.5 as a result.
=STDEV(D2:D13)	Compute standard deviation of D2, D3, ..., and D13. Returns 3.28 as a result.

Table A1 Example MS Excel Formulas

Formula	Description
=MIN(D2:D13)	Find the minimum value among D2, D3, ..., and D13. Returns 70 as a result.
=MAX(D2:D13)	Find the maximum value among D2, D3, ..., and D13. Returns 82 as a result.
=SUM(D2:D13)	Sum the values of D2, D3, ..., through D13. Returns 880 as a result.
=COUNTIF(D2:D13, 72)	Count the number of occurrences of "72" within the range of D2, D3, ..., through D13. Returns 3 as a result (i.e., there are three 72s in D2:D13).

Table A1 Example MS Excel Formulas

EXAMPLE A.7

Problem

On ch1-SOCR-small.csv (https://openstax.org/r/folders14), write a formula that calculates the median weight of the players.

Solution

The formula is

=MEDIAN(E2:E13)

This will find the median across cell E2, E3, E4, ..., through E13, which is 200. If you do not know what median is, don't worry! You will learn more about it in Measures of Position.

Similar to the built-in function for the mean (=AVERAGE), Excel provides a function to calculate the sample standard deviation of a dataset =STDEV.S (for the sample standard deviation). To calculate these statistical results in Excel, enter the data values in a column. Let's assume the data values are placed in cells A2 through A11. In any cell, type the Excel command =AVERAGE(A2:A11) and press enter. Excel will calculate the arithmetic mean in this cell. Then, in any other cell, type the Excel command =STDEV.S(A2:A11) and press enter. Excel will calculate the sample standard deviation in this cell. Figure A25 shows the mean and standard deviation for the 10 ages.

	A	B	C	D	E
1	**Ages**				
2	40				
3	36		Excel command to calculate sample mean:		
4	44		**=AVERAGE(A2:A11)**		
5	51				
6	54		Result:	46	
7	55				
8	39		Excel command to calculate sample standard deviation:		
9	47		**=STDEV.S(A2:A11)**		
10	44				
11	50		Result:	6.50	

Figure A25 Mean and Standard Deviation in MS Excel (Used with permission from Microsoft)

Here is a more comprehensive listing of various statistical functions available within Excel. The reader is encouraged to go to the Formulas menu in Excel and select any of the submenus such as "Financial," "Math & Trig," Lookup & Reference," etc. to see various Excel formulas. Under the option for "More Functions," select STATISTICAL to see a menu of various statistical functions available within Excel. Table A2 shows a sample of some statistical functions available within Excel:

Function	Purpose
=AVERAGE(A1:A10)	Find the mean of a set of numbers.
=MEDIAN(A1:A10)	Find the median of a set of numbers.
=STDEV.S(A1:A10)	Find the standard deviation for a set of numbers representing a **sample**.
=STDEV.P(A1:A10)	Find the standard deviation for a set of numbers representing a **population**.
=VAR.S(A1:A10)	Find the variance for a set of numbers representing a sample.
=VAR.P(A1:A10)	Find the variance for a set of numbers representing a population.
=MIN(A1:A10)	Find the minimum of a set of numbers.
=MAX(A1:A10)	Find the maximum of a set of numbers.
=MAX(A1:A10) – MIN(A1:A10)	Find the range of a set of numbers.
=CORREL(A1:A10, B1:B10)	Find the correlation coefficient "r" for (x, y) data. x-data is in cells A1 to A10; y-data is in cells B1 to B10.
=QUARTILE (A1:A10, 1)	Find the first quartile for a set of numbers.
=QUARTILE (A1:A10, 2)	Find the second quartile for a set of numbers (note that the second quartile is the same as the median).
=QUARTILE (A1:A10, 3)	Find the third quartile for a set of numbers.

Table A2 Statistical Functions Available within MS Excel

Excel Analysis for Probability Distributions

As discussed in Measures of Position, data scientists are often interested in various probability distributions such as the normal distribution, binomial distribution, and Poisson distribution. Excel provides built-in functions to analyze many probability distributions.

Excel uses the command =NORM.DIST to find the area under the normal curve to the *left* of a specified value:

=NORM.DIST(XVALUE, MEAN, STANDARD_DEV, TRUE)

For example, suppose that at a financial consulting company, the mean employee salary is $60,000 with a standard deviation of $7,500. A normal curve can be drawn to represent this scenario, in which the mean of $60,000 would be plotted on the horizontal axis, corresponding to the peak of the curve. Then, to find the probability that an employee earns more than $75,000, you would calculate the area under the normal curve to the right of the data value $75,000.

For example, at the financial consulting company mentioned previously, the mean employee salary is $60,000 with a standard deviation of $7,500. To find the probability that a random employee's salary is less than $55,000 using Excel, this is the command you would use:

=NORM.DIST(55000, 60000, 7500, TRUE)

Result: 0.25249

Thus, there is a probability of about 25% that a random employee has a salary less than $55,000.

Excel also provides built-in functions for binomial and Poisson distributions as shown in Table A3:

Probability Distribution	Excel Command	Usage
Binomial	=BINOMIAL.DIST	=BINOMIAL.DIST(Number_s, Trials, Probability_S, Cumulative)
Poisson	=POISSON	=POSSON(X, Mean, Cumulative)

Table A3 Built-In Functions for Binomial and Poisson Distributions in MS Excel

Excel Analysis for Correlation and Regression

In Inferential Statistics and Regression Analysis, the concepts of correlation and regression were discussed.

In correlation analysis, the data scientist is often interested in calculating the correlation coefficient, which is a numerical measure of the direction and strength of the correlation between two numeric quantities.

The Excel command to calculate the correlation coefficient uses the following format:

=CORREL(A1:A10, B1:B10)

In regression analysis, the data scientist is interested in calculating the equation of the best-fit linear model for two numeric quantities (assuming the correlation is significant).

Recall the equation of the best fit line is

$$\hat{y} = a + bx$$

where m is the slope of the line and b is the y-intercept of the line.

To calculate the slope and y-intercept of the linear model using Excel, start by entering the (x, y) data in two columns in Excel. Then the Excel commands =SLOPE and =INTERCEPT can be used to calculate the slope and intercept, respectively.

The dataset in Table A4 will be used as an example: the monthly amount spent on advertising and the monthly revenue for a Fortune 500 company for 12 months.

Month	Amount Spent on Advertising	Revenue
Jan	49	12210
Feb	145	17590
Mar	57	13215
Apr	153	19200
May	92	14600
Jun	83	14100
Jul	117	17100
Aug	142	18400
Sep	69	14100
Oct	106	15500
Nov	109	16300
Dec	121	17020

Table A4 Revenue versus Advertising for a Fortune 500 Company ($000s) Monthly Amount Spent on Advertising and the Monthly Revenue for a Fortune 500 Company

To calculate the slope of the regression model, use the Excel command

=SLOPE(*y*-data range, *x*-data range)

It's important to note that this Excel command expects that the *y*-data range is entered first and the *x*-data range is entered second. Since revenue depends on amount spent on advertising, revenue is considered the *y*-variable and amount spent on advertising is considered the *x*-variable. Notice the *y*-data is contained in cells C2 through C13 and the *x*-data is contained in cells B2 through B13. Thus the Excel command for slope would be entered as

=SLOPE(C2:C13, B2:B13)

In the same way, the Excel command to calculate the *y*-intercept of the regression model is

=INTERCEPT(*y*-data range, *x*-data range)

For the dataset shown in the above table, the Excel command would be

=INTERCEPT(C2:C13, B2:B13)

The results are as follows (see Figure A26):

$$\begin{aligned} \text{slope } b &= 61.8 \\ \text{intercept } a &= 9{,}376.7 \end{aligned}$$

	A	B	C
1	**Month**	**Advertising Expenditure**	**Revenue**
2	Jan	49	12,210
3	Feb	145	17,590
4	Mar	57	13,215
5	Apr	153	19,200
6	May	92	14,600
7	Jun	83	14,100
8	Jul	117	17,100
9	Aug	142	18,400
10	Sep	69	14,100
11	Oct	106	15,500
12	Nov	109	16,300
13	Dec	121	17,020
14			
15		**=SLOPE(C2:C13,B2:B13)**	**=INTERCEPT(C2:C13,B2:B13)**
16		**61.8**	**9,376.7**

Figure A26 Revenue versus Advertising for a Fortune 500 Company ($000s) Showing Slope and *y*-Intercept Calculation in MS Excel (Used with permission from Microsoft)

Based on this, the regression equation can be written as

$$\begin{aligned} \hat{y} &= a + bx \\ \hat{y} &= 9{,}376.7 + 61.8x \end{aligned}$$

where *x* represents the amount spent on advertising (in thousands of dollars) and *y* represents the amount of revenue (in thousands of dollars).

EXPLORING FURTHER

Excel Resources

For additional resources, refer to the chapters "Advanced Excel Formulas, Functions, and Techniques (https://openstax.org/r/pages10)" and "Advanced Excel Spreadsheets: Statistical and Data Analysis (https://openstax.org/r/pages11)" in Bolling, T., Mitchell, A., Scott, T., & Wheeler, N. (2023). *Workplace Software and Skills*. OpenStax. Retrieved from https://openstax.org/books/workplace-software-skills/pages/1-chapter-scenario (https://openstax.org/r/pages1).

B Appendix B: Review of R Studio for Data Science

R is a statistical analysis tool that is widely used in the data science field. R provides many tools for data exploration, visualization, and statistical analysis. It is available as a free, open-source program and provides an integrated suite of functions for data analysis, graphing, and statistical programming. R is increasingly being used by data scientists as a data analysis and statistical tool in part because R is an open-source language and additional features are constantly being added by the user community. The tool can be used on many different computing platforms and can be downloaded at the R Project website (https://openstax.org/r/rproject).

Once you have installed and started R on your computer, at the bottom of the R console, you should see the symbol >, which indicates that R is ready to accept commands.

For a user new to R, typing `help.start()` at the R prompt provides a menu of Manuals and Reference materials as shown in Figure B1.

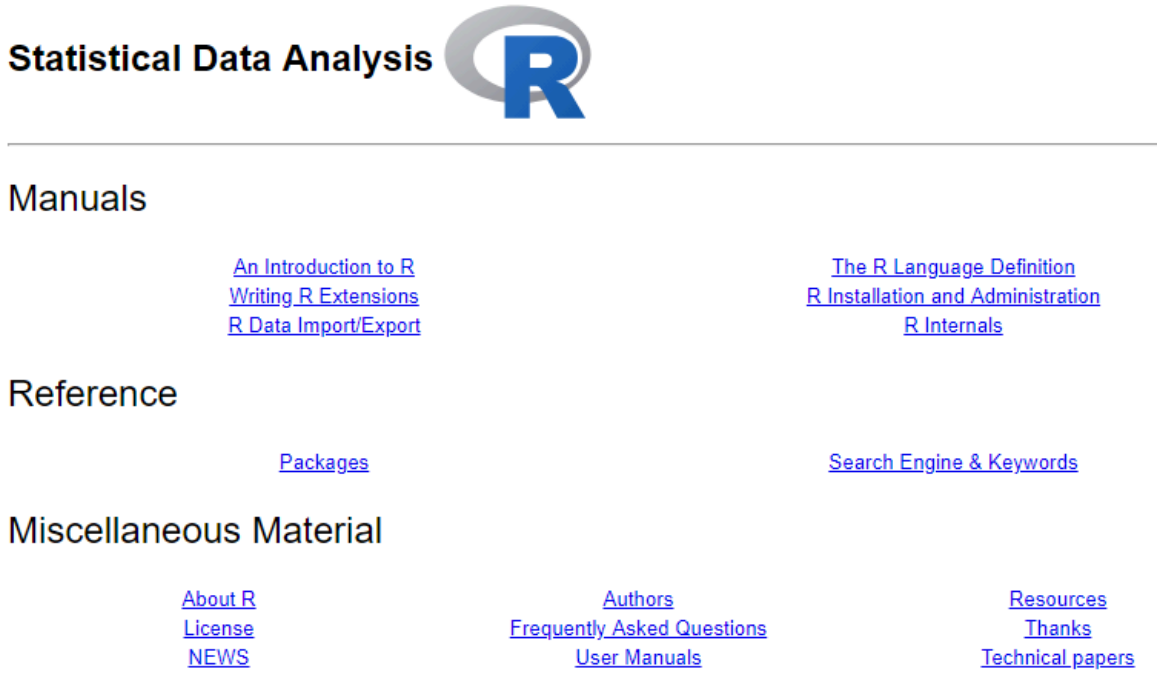


Figure B1 R Help Menu Based on `help.start()`

R provides many built-in help resources. When a user types `help()` at the R prompt, a listing of help resources are provided. For a specific example, typing `help(median)` will show various documentation on the built-in median function within R.

In addition, if a user types `demo()` at the R prompt, various demonstration options are shown. For a specific example, typing `demo(graphics)` will provide some examples of various graphics plots.

Basic Data Analysis Using R

R is a command-driven language, meaning that the user enters commands at the prompt, which R then executes one at a time. R can also execute a program containing multiple commands. There are ways to add a graphic user interface (GUI) to R. An example of a GUI tool for R is RStudio (https://openstax.org/r/posit).

The R command line can be used to perform any numeric calculation, similar to a handheld calculator. For example, to evaluate the expression $10 + 3 \cdot 7$, enter the following expression at the command line prompt

and press return. The numeric result of 31 is then shown:

```
> 10+3*7

[1] 31
```

Most calculations in R are handled via functions. For data science and statistical analysis, there are many pre-established functions in R to calculate mean, median, standard deviation, quartiles, and so on. Variables can be named and assigned values using the assignment operator `<-`. For example, the following R commands assign the value of 20 to the variable named *x* and assign the value of 30 to the variable named *y*:

```
> x <- 20

> y <- 30
```

These variable names can be used in any calculation, such as multiplying *x* by *y* to produce the result 600:

```
> x*y

[1] 600
```

The typical method for using functions in statistical applications is to first create a vector of data values. There are several ways to create vectors in R. For example, the `c` function is often used to combine values into a vector. The following R command will generate a vector called *salaries* that contains the data values 40000, 50000, 75000, and 92000:

```
> salaries <- c(40000, 50000, 75000, 92000)
```

This vector *salaries* can then be used in statistical functions such as mean, median, min, max, and so on, as shown:

```
> mean(salaries)

[1] 64250

> median(salaries)

[1] 62500

> min(salaries)

[1] 40000

> max(salaries)

[1] 92000
```

Another option for generating a vector in R is to use the `seq` function, which will automatically generate a sequence of numbers. For example, we can generate a sequence of numbers from 1 to 5, incremented by 0.5, and call this vector *example1*, as follows:

```
> example1 <- seq(1, 5, by=0.5)
```

If we then type the name of the vector and press enter, R will provide a listing of numeric values for that vector name.

```
> salaries

[1] 40000 50000 75000 92000

> example1

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Oftentimes, a data scientist is interested in generating a quick statistical summary of a dataset in the form of its mean, median, quartiles, min, and max. The R command called `summary` provides these results.

```
> summary(salaries)

Min. 1st Qu. Median Mean 3rd Qu. Max.

40000 47500 62500 64250 79250 92000
```

For measures of spread, R includes a command for standard deviation, called `sd()`, and a command for variance, called `var()`. The standard deviation and variance are calculated with the assumption that the dataset was collected from a sample.

```
> sd(salaries)

[1] 23641.42

> var(salaries)

[1] 558916667
```

To calculate a weighted mean in R, create two vectors, one of which contains the data values and the other of which contains the associated weights. Then enter the R command `weighted.mean(values, weights)`.

The following is an example of a weighted mean calculation in R:

EXAMPLE B.1

Problem

Assume a financial portfolio contains 1,000 shares of XYZ Corporation, purchased on three different dates, as shown in Table B1. Use R to calculate the weighted mean of the purchase price, where the weights are based on the number of shares in the portfolio.

Date Purchased	Purchase Price ($)	Number of Shares Purchased
January 17	78	200
February 10	122	300
March 23	131	500
Total		**1000**

Table B1 Portfolio of XYZ Shares

Solution

Here is how you would create two vectors in R: the *price* vector will contain the purchase price, and the *shares* vector will contain the number of shares. Then execute the R command `weighted.mean(price, shares)`, as follows:

```
> price <- c(78, 122, 131)

> shares <- c(200, 300, 500)

> weighted.mean(price, shares)

[1] 117.7
```

A list of common R statistical commands appears in Table B2.

R Command	Result
`mean()`	Calculates the arithmetic mean
`median()`	Calculates the median
`min()`	Calculates the minimum value
`max()`	Calculates the maximum value
`weighted.mean()`	Calculates the weighted mean

Table B2 List of Common R Statistical Commands

R Command	Result
`sum()`	Calculates the sum of values
`summary()`	Calculates the mean, median, quartiles, min, and max
`sd()`	Calculates the sample standard deviation
`var()`	Calculates the sample variance
`IQR()`	Calculates the interquartile range
`barplot()`	Plots a bar chart of non-numeric data
`boxplot()`	Plots a boxplot of numeric data
`hist()`	Plots a histogram of numeric data
`plot()`	Plots various graphs, including a scatter plot
`freq()`	Creates a frequency distribution table

Table B2 List of Common R Statistical Commands

Basic Visualization and Graphing Using R

R provides many built-in functions for data visualization and graphing and allows the data scientist significant flexibility and customization options for graphs and other data visualizations.

There are many statistical applications in R, and many graphical representations are possible, such as bar graphs, histograms, time series plots, scatter plots, and others.

As a simple example of a bar graph, assume a college instructor wants to create a bar graph to show enrollment in various courses such as statistics, history, physics, and chemistry courses.

Table B3 shows the enrollment data:

College Course	Student Enrollment
Statistics	375
History	302

Table B3 Enrollment Data

College Course	Student Enrollment
Physics	294
Chemistry	193

Table B3 Enrollment Data

The basic command to create a bar graph in R is the command `barchart()`.

First, create a dataframe called enrollment to hold the data (a dataframe can be considered a table or matrix to store the dataset).

```
enrollment <- data.frame(course=c("Statistics", "History", "Physics", "Chemistry"),
enrolled=c(375, 302, 294, 193))
```

Next, use the `barplot` function to create the bar graph and add labels for x-axis, y-axis, and overall title.

```
barplot(enrollment$enrolled, names.arg=enrollment$course,
main="Student Enrollment at the College", xlab="Course Name",
ylab = "Enrollment")
```

The resulting output is shown below in Figure B2:

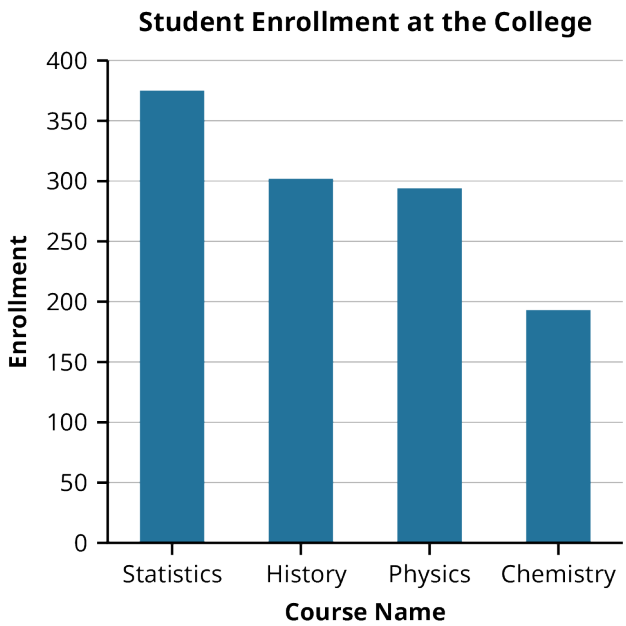


Figure B2 Bar Graph of Student Enrollment Data

The basic command to create a scatter plot in R is the plot command, `plot(x, y)`, where *x* is a vector containing the *x*-values of the dataset and *y* is a vector containing the *y*-values of the dataset.

The general format of the command is as follows:

```
>plot(x, y, main="text for title of graph",
xlab="text for x-axis label", ylab="text for y-axis label")
```

For example, we are interested in creating a scatter plot to examine the correlation between the value of the S&P 500 and Nike stock prices. Assume we have the data shown in Table B4, collected over a one-year time

period.

Date	S&P 500	Nike Stock Price ($)
4/1/2020	2912.43	87.18
5/1/2020	3044.31	98.58
6/1/2020	3100.29	98.05
7/1/2020	3271.12	97.61
8/1/2020	3500.31	111.89
9/1/2020	3363.00	125.54
10/1/2020	3269.96	120.08
11/1/2020	3621.63	134.70
12/1/2020	3756.07	141.47
1/1/2021	3714.24	133.59
2/1/2021	3811.15	134.78
3/1/2021	3943.34	140.45
3/12/2021	3943.34	140.45
(source: https://finance.yahoo.com/)		

Table B4 Data for S&P 500 and Nike Stock Price over a 12-Month Period

Note that data can be read into R from a text file or Excel file or from the clipboard by using various R commands. Assume the values of the S&P 500 have been loaded into the vector *SP500* and the values of Nike stock prices have been loaded into the vector *Nike*. Then, to generate the scatter plot, we can use the following R command:

```
>plot(SP500, Nike, main="Scatter Plot of Nike Stock Price vs. S&P 500",
xlab="S&P 500", ylab="Nike Stock Price")
```

As a result of these commands, R provides the scatter plot shown in Figure B3.

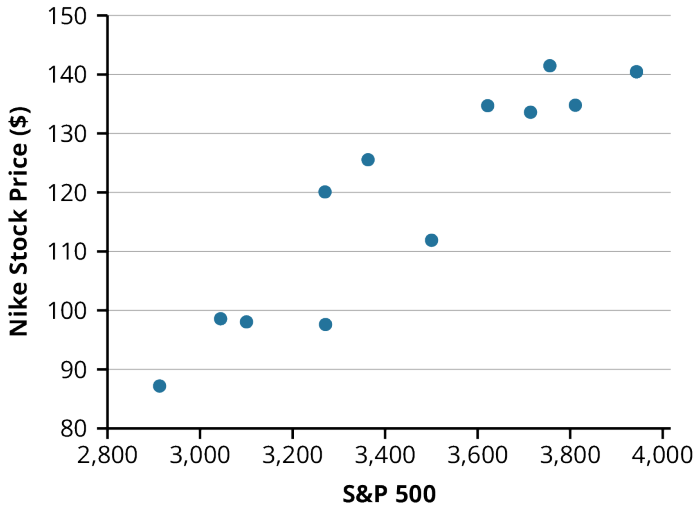


Figure B3 Scatter Plot Generated by R for Nike Stock Price vs. S&P 500

R Analysis for Probability Distributions

As mentioned earlier, a quick statistical summary can be generated using the `summary()` command in R, where the usage is `summary(data_vector)`.

R has extensive built-in libraries for calculating confidence intervals, hypothesis testing, and working with various probability distributions such as binomial, Poisson, normal, chi-square, and other probability distributions.

As discussed in Measures of Center, data scientists are often interested in various probability distributions such as the normal distribution, binomial distribution, and Poisson distribution. Excel provides built-in functions to analyze many probability distributions.

R uses the `pnorm` command to find the area under the normal curve to the left of a specified value:

Usage: `pnorm(x_value, mean, standard_deviation)`

where `pnorm` returns the probability that a random variable having a given mean and standard deviation is less than `x_value`.

EXAMPLE B.2

Problem

Birth weights for newborns in the United States are normally distributed with mean of 3,400 grams and standard deviation of 500 grams.

a. Use R to find the probability that a random newborn infant weighs less than 4,000 grams.
b. Use R to find the probability that a random newborn infant weighs more than 3,000 grams.

Solution

For part (a), use the `pnorm` command as follows:

```
pnorm(4000, 3400, 500)
```

which returns the probability result of:

```
[1] 0.8849303
```

For part (b), an option on the `pnorm` command "`lower.tail=FALSE`" can calculate the area to the right of a given *x*-value:

```
pnorm(3000, 3400, 500, lower.tail=FALSE)
```

which returns the probability result of:

```
[1] 0.7881446
```

R also provides a built-in function for the binomial distribution as follows:

Binomial distribution:

Usage: `pbinom(k, n, p)`

where *n* is the number of trials, *p* is the probability of success, and *k* is the number of successes for which

the probability is desired.

EXAMPLE B.3

Problem

A data scientist conducts a survey for a sample of 20 people and asks the survey question: "Did you find the website for ABC corporation easy to navigate?" From past data, the probability that a random person found the website easy to navigate was 65%. Use R to find the probability that 13 out of the 20 respond that they find the website easy to navigate.

Solution

Use the `pbinom` command as follows:

```
pbinom(13, 20, 0.65)
```

which returns the probability result of:

```
[1] 0.5833746
```

R also provides a built-in function for the binomial distribution as follows:

Binomial distribution:

```
Usage: dbinom(k, n, p)
```

where n is the number of trials, p is the probability of success, and k is the number of successes for which the probability is desired.

EXAMPLE B.4

Problem

A data scientist conducts a survey for a sample of 20 people and asks the survey question: "Did you find the website for ABC corporation easy to navigate?" From past data, the probability that a random person found the website easy to navigate was 65%. Use R to find the probability that 13 out of the 20 responds that they find the website easy to navigate.

Solution

Use the `dbinom` command as follows:

```
dbinom(13, 20, 0.65)
```

which returns the probability result of:

```
[1] 0.1844012
```

R also provides a built-in function for the Poisson distribution as follows:

Poisson distribution:

```
Usage: dpois(k, mu)
```

where mu is the mean of the Poisson distribution and k is the number of successes for which the probability is desired.

EXAMPLE B.5

Problem

A traffic engineer investigates a certain intersection that has an average of 3 accidents per month. Use R to find the probability of 5 accidents in a given month.

Solution

Use the `ppois` command as follows:

```
dpois(5, 3)
```

which returns the probability result of:

```
[1] 0.1008188
```

Basic Correlation and Regression Analysis Using R

Recall in Inferential Statistics and Regression Analysis the discussion on correlation and regression analysis. A first step in correlation analysis is to calculate the correlation coefficient (*r*) for (*x*, *y*) data. R provides a built-in function `cor()` to calculate the correlation coefficient for bivariate data.

As an example, consider the dataset in Table B5 that tracks the return on the S&P 500 versus return on Coca-Cola stock for a seven-month time period.

Month	S&P 500 Monthly Return (%)	Coca-Cola Monthly Return (%)
Jan	8	6
Feb	1	0
Mar	0	-2
Apr	2	1
May	-3	-1
Jun	7	8
Jul	4	2

Table B5 Monthly Returns of Coca-Cola Stock versus Monthly Returns for the S&P 500

To calculate the correlation coefficient for this dataset, first create two vectors in R, one vector for the S&P 500 returns and a second vector for Coca-Cola returns:

```
> SP500 <- c(8,1,0,2,-3,7,4)
```

```
> CocaCola <- c(6,0,-2,1,-1,8,2)
```

The R command called `cor` returns the correlation coefficient for the *x*-data vector and *y*-data vector:

```
> cor(SP500, CocaCola)

[1] 0.9123872
```

Thus the correlation coefficient for this dataset is approximately 0.912.

Linear Regression Models Using R

To create a linear model in R, assuming the correlation is significant, the command `lm()` (for linear model) will provide the slope and *y*-intercept for the linear regression equation.

The format of the R command is

```
lm(dependent_variable_vector ~ independent_variable_vector)
```

Notice the use of the tilde symbol as the separator between the dependent variable vector and the independent variable vector.

We use the returns on Coca-Cola stock as the dependent variable and the returns on the S&P 500 as the independent variable, and thus the R command would be

```
> lm(CocaCola ~ SP500)

Call:

lm(formula = CocaCola ~ SP500)

Coefficients:

(Intercept) SP500

-0.3453 0.8641
```

The R output provides the value of the *y*-intercept as -0.3453 and the value of the slope as 0.8641. Based on this, the linear model would be

$$\begin{aligned} \hat{y} &= a + bx \\ \hat{y} &= -0.3453 + 0.8641x \end{aligned}$$

where *x* represents the value of S&P 500 return and *y* represents the value of Coca-Cola stock return.

The results can also be saved as a formula and called "model" using the following R command. To obtain more detailed results for the linear regression, the `summary` command can be used, as follows:

```
> model <- lm(CocaCola ~ SP500)

> summary(model)
```

```
Call:

lm(formula = CocaCola ~ SP500)

Residuals:

1 2 3 4 5 6 7

-0.5672 -0.5188 -1.6547 -0.3828 1.9375 2.2969 -1.1109

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.3453 0.7836 -0.441 0.67783

SP500 0.8641 0.1734 4.984 0.00416 **
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.658 on 5 degrees of freedom

Multiple R-squared: 0.8325, Adjusted R-squared: 0.7989

F-statistic: 24.84 on 1 and 5 DF, p-value: 0.004161
```

In this output, the *y*-intercept and slope is given, as well as the residuals for each *x*-value. The output includes additional statistical details regarding the regression analysis.

Predicted values and prediction intervals can also be generated within R. First, we can create a structure in R called a dataframe to hold the values of the independent variable for which we want to generate a prediction. For example, we would like to generate the predicted return for Coca-Cola stock, given that the return for the S&P 500 is 6.

We use the R command called `predict()`.

To generate a prediction for the linear regression equation called *model*, using the dataframe where the value of the S&P 500 is 6, the R commands will be

```
> a <- data.frame(SP500=6)

> predict(model, a)

1

4.839062
```

The output from the `predict` command indicates that the predicted return for Coca-Cola stock will be 4.8% when the return for the S&P 500 is 6%.

We can extend this analysis to generate a 95% prediction interval for this result by using the following R

command, which adds an option to the `predict` command to generate a prediction interval:

```
> predict(model, a, interval="predict")

fit lwr upr

1 4.839062 0.05417466 9.62395
```

Thus the 95% prediction interval for Coca-Cola return is (0.05%, 9.62%) when the return for the S&P 500 is 6%.

Multiple Regression Models Using R

R also includes many tools to allow the data scientist to conduct multiple regression, where a dependent variable is predicted based on more than one independent variable. For example, we might arrive at a better prediction model for monthly return of Coca-Cola stock if we consider not only the S&P500 monthly return but also take into account the monthly sales of Coca-Cola products as well.

Here are several examples where a multiple regression model might provide an improved prediction model as compared to a regression model with only one independent variable:

a. Employee salaries can be predicted based on years of experience and education level.
b. Housing prices can be predicted based on square footage of a home, number of bedrooms, and number of bathrooms.

The general form of the multiple regression model is:

$$\hat{y} = a + b_1 x_1 + b_2 x_2 + b_3 x_3 + ? + b_n x_n$$

where:

$x_1, x_2, x_3, \ldots, x_n$ are the independent variables,
$b_1, b_2, b_3, \ldots, b_n$ are the coefficients where each coefficient is the amount of change in *y* when the independent variable x_i is changed by one unit and all other independent variables are held constant,
a is the *y*-intercept, which is the value of *y* when all $x_i = 0$.

Recall from an earlier example that the format for linear regression analysis in R when there is only one independent variable looked like the following:

```
> model <- lm(y ~ x)
```

where *y* is the dependent variable and *x* is the independent variable.

To "add in" additional independent variables for the multiple regression approach, we use a format as follows:

```
> model <- lm(y ~ x1 + x2 + x3)
```

where x_1, x_2, x_3 are the independent variables.

Example:

Use R to create a multiple regression model to predict the price of a home based on the independent variables of square footage and number of bedrooms based on the following dataset. Then use the multiple regression model to predict the price of a home with 3,000 square feet and 3 bedrooms (see Table B6).

Price of Home (y)	Square Footage x_1	Number of Bedrooms x_2
466000	4668	6
355000	3196	5
405900	3998	5
415000	4022	5
206000	1834	2
462000	4668	6
290000	2650	3

Table B6 Home Prices Based on Square Footage and Number of Bedrooms

First, create three vectors in R, one vector each for home price, square footage and number of bedrooms:

```
> price <- c(466000, 355000, 405900, 415000, 206000, 462000, 290000)
> square_footage <- c(4668, 3196, 3998, 4022, 1834, 4668, 2650)
> bedrooms <- c(6, 5, 5, 5, 2, 6, 3)
```

Next, run the multiple regression model using the `lm` command:

```
> model <- lm(price ~ square_footage + bedrooms)
> summary(model)

Call:
lm(formula = price ~ square_footage + bedrooms)

Residuals:
1 2 3 4 5 6 7
-1704.6 1438.4 -606.9 6908.7 -6747.4 -5704.6 6416.5

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 57739.887 9529.693 6.059 0.00375 **
square_footage 66.017 8.924 7.398 0.00178 **
bedrooms 16966.536 6283.183 2.700 0.05408 .
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6563 on 4 degrees of freedom
Multiple R-squared: 0.9968, Adjusted R-squared: 0.9953
F-statistic: 630.6 on 2 and 4 DF, p-value: 9.996e-06
```

In the R output, note the column called “Estimate” provides the estimates for the coefficients and the *y*-intercept.

The *y*-intercept is given as 57740 (rounding to nearest whole number).

The coefficient for the “square footage” variable is given as 66.

The coefficient for the "bedrooms" variance is given as 16967.

Based on these values, the multiple regression model is:

$$\hat{y} = 57740 + 66x_1 + 16967x_2$$

We can now use the multiple regression model to predict the price of a home with 3,000 square feet and 3 bedrooms by setting $x_1 = 3000$ and $x_2 = 3$, as follows:

$$\begin{aligned} \hat{y} &= 57740 + 66(3000) + 16967(3) \\ \hat{y} &= 306641 \end{aligned}$$

Thus, the predicted price of a home with 3,000 square feet and 3 bedrooms is $306,641.

C Appendix C: Review of Python Algorithms

This appendix provides a summary of various Python algorithms used in this textbook. The intent is to provide students with a cross-reference to the various algorithms used in the text and provide a general description of the algorithm and link to the section of the text.

For more details on Python functions, syntax, and usage, please refer to Appendix D: Review of Python Functions and/or the Python documentation (https://openstax.org/r/docs3) online.

Chapter Title	Topic	Description of Algorithm	First Reference
What Are Data and Data Science?	Loading and viewing data	Using Python `pandas` library to load a CSV file, describe its features, and explore its contents	Python Basics for Data Science
	Visualizing data using Python	Using Python to create a scatterplot for two numeric quantities	Python Basics for Data Science
Collecting and Preparing Data	Scraping data from a website	Using Python to extract a data table from a website	Web Scraping and Social Media Data Collection
	Using regular expressions in Python	Using Python to search for a selected word in a given string and output the number of times it appears	Web Scraping and Social Media Data Collection
	Processing and storing data	Using Python to process data and output to a CSV file	Web Scraping and Social Media Data Collection
	Parsing and extracting data	Using Python to parse and extract specific data from a given dataset	Web Scraping and Social Media Data Collection
Descriptive Statistics: Statistical Measurements and Probability Distributions	Calculate binomial probabilities	Using Python to calculate probabilities associated with the binomial distribution	Discrete and Continuous Probability Distributions
	Calculate probabilities from a normal distribution	Using Python to calculate probabilities associated with the normal distribution	Discrete and Continuous Probability Distributions
Inferential Statistics and Regression Analysis	Computing margin of error	Using Python library `scipy.stats` to compute a margin of error using the *t*-distribution	Statistical Inference and Confidence Intervals
	Confidence interval using bootstrapping	Using Python to calculate a confidence interval using a bootstrapping approach	Statistical Inference and Confidence Intervals

Table C1

Chapter Title	Topic	Description of Algorithm	First Reference
	Confidence interval for the mean	Using Python to calculate a confidence interval for the mean using the normal and *t*-distributions (two examples)	Statistical Inference and Confidence Intervals
	Hypothesis test for the mean (one sample)	Using Python to calculate a *p*-value associated with a hypothesis test for the mean	Hypothesis Testing
	Hypothesis test for a proportion	Using Python to calculate a *p*-value associated with a hypothesis test for a proportion	Hypothesis Testing
	Hypothesis test for the mean (one sample)	Using Python to calculate a test statistic and *p*-value associated with a hypothesis test for the mean	Hypothesis Testing
	Hypothesis test for the mean (two samples)	Using Python to calculate a test statistic and *p*-value associated with a hypothesis test for the difference between two means	Hypothesis Testing
	Correlation coefficient	Using Python to calculate the Pearson correlation coefficient for two numeric variables	Correlation and Linear Regression Analysis
	Creating a scatterplot	Using Python to create a scatterplot for two numeric quantities	Correlation and Linear Regression Analysis
	Creating a linear regression model	Using Python to calculate the slope and intercept for a linear regression model	Correlation and Linear Regression Analysis
	One-way analysis of variance (ANOVA)	Using Python to conduct a one-way analysis of variance hypothesis test	Analysis of Variance (ANOVA)
	Visualizing time series data	Using basic Python plot routine and `matplotlib.pyplot` to generate a time series graph based on a `pandas` DataFrame (two examples)	Analysis of Variance (ANOVA)
Time Series and Forecasting	Plotting a simple moving average (SMA) and differencing	Using Python to generate a simple moving average and first-order difference for time series data	Time Series Forecasting Methods

Table C1

Chapter Title	Topic	Description of Algorithm	First Reference
	Decomposing a time series into components	Using Python autocorrelation function and STL (seasonal and trend decomposition using LOESS) model to decompose time series data into its components	Time Series Forecasting Methods
	Exponential moving average (EMA)	Using Python to calculate exponential moving average for time series data	Time Series Forecasting Methods
	Autoregressive integrated moving average (ARIMA)	Using Python to calculate autoregressive integrated moving average model for time series data	Time Series Forecasting Methods
	Autoregressive integrated moving average (ARIMA)	Using Python to fit and plot an autoregressive integrated moving average and use it to make forecasts for time series data	Time Series Forecasting Methods
	Forecasting methods	Using Python to create and plot a forecasting model with confidence intervals for time series data	Forecast Evaluation Methods
Decision-Making Using Machine Learning Basics	Logistic regression	Using Python to fit a logistic regression model and assess the accuracy of the model	Classification Using Machine Learning
	K-means clustering	Using Python to produce a k-means clustering model from a dataset	Classification Using Machine Learning
	DBscan clustering	Using Python to generate a density-based spatial clustering of applications with noise (DBScan) model	Classification Using Machine Learning
	Confusion matrix	Using Python to generate a confusion matrix	Classification Using Machine Learning
	Linear regression with bootstrapping	Using Python to generate a linear regression model using a bootstrapping method	Machine Learning in Regression Analysis
	Multiple regression	Using Python to perform multiple regression analysis	Machine Learning in Regression Analysis

Table C1

Chapter Title	Topic	Description of Algorithm	First Reference
	Three-dimensional scatterplot	Using Python to generate a three-dimensional plot of a multiple regression model	Machine Learning in Regression Analysis
	Mesh grid	Using Python to generate a mesh grid plot of a multiple regression model	Machine Learning in Regression Analysis
	Multiple logistic regression	Using Python to perform multiple logistic regression analysis and generate corresponding confusion matrix	Machine Learning in Regression Analysis
	Decision trees	Using Python to generate decision trees	Decision Trees
	Random forests	Using Python to train a random forests model and analyze the importance of each feature	Other Machine Learning Techniques
	Gaussian naïve Bayes	Using Python to perform Gaussian naïve Bayes analysis	Other Machine Learning Techniques
Deep Learning and Artificial Intelligence (AI) Basics	Perceptrons	Using Python to train and test a perceptron classification model and assess the accuracy of the model	Introduction to Neural Networks
	Training a neural network with backpropagation	Using Python's `TensorFlow` library to train and test a neural network classification model using backpropagation and assess the accuracy of the model	Backpropagation
	Recurrent neural networks	Using Python's `TensorFlow` library to train and test a classification model using recurrent neural networks (RNN) and assess the accuracy of the model	Backpropagation
	Predict future values	Using Python to predict future values for a classification model using recurrent neural networks (RNN)	Backpropagation

Table C1

Chapter Title	Topic	Description of Algorithm	First Reference
	Plot predicted values	Using Python to plot future values versus original data for a classification model using recurrent neural networks	Backpropagation
	Deep learning	Using Python's `TensorFlow` library to train and test a classification model using deep learning and assess the accuracy of the model	Backpropagation
Visualizing Data	Data visualization using boxplots	Using Python to create boxplots	Encoding Univariate Data
	Data visualization using histograms	Using Python to create histograms	Encoding Univariate Data
	Data visualization using Pareto charts	Using Python to create Pareto charts	Encoding Univariate Data
	Data visualization using time series charts	Using Python to create time series charts	Encoding Data That Change over Time
	Data visualization for binomial probabilities	Using Python to create graphs associated with the binomial distribution	Graphing Probability Distributions
	Data visualization for Poisson probabilities	Using Python to create graphs associated with the Poisson distribution	Graphing Probability Distributions
	Data visualization for normal probabilities	Using Python to create graphs associated with the normal distribution	Graphing Probability Distributions
	Heatmaps	Using Python to create heatmaps	Geospatial and Heatmap Data Visualization Using Python
	Scatterplots with colormaps	Using Python to create scatterplots with colormaps	Multivariate and Network Data Visualization Using Python
	Correlation heatmaps	Using Python to create correlation heatmaps	Multivariate and Network Data Visualization Using Python
	Data visualization using three-dimensional plots	Using Python to create three-dimensional plots	Multivariate and Network Data Visualization Using Python

Table C1

Chapter Title	Topic	Description of Algorithm	First Reference
Reporting Results	Identify data characteristics	Using Python to identify data characteristics of a dataset	Validating Your Model
	Decision tree	Using Python to run a decision tree and generate a visualization of the result	Validating Your Model
	Model validation using Bayesian information criterion (BIC)	Using Python to perform cross validation with Bayesian information criterion	Validating Your Model
	Monte Carlo simulation	Using Python to perform Monte Carlo simulation	Validating Your Model
	Executive summary	Using Python to create an executive summary report	Effective Executive Summaries

Table C1

D Appendix D: Review of Python Functions

This appendix provides a summary of Python functions used in this textbook. The intent is to provide students with a cross-reference of Python commands that includes a description of the Python functions, general syntax for usage, and a link to the section where the function is first used in the text.

Please note this is a very high-level description of these functions. Many functions require specific libraries to be installed. For more details on Python functions, syntax, and usage, please refer to the Python documentation (https://openstax.org/r/python3) posted online.

Python Function	Description	Syntax	First Reference
What Are Data and Data Science?			
`print()`	Prints a specified message or specified values to the screen or other output device	`print("text")` `print(x, y)`	Python Basics for Data Science
`pd.read_csv()`	Loads data from a CSV (comma-separated values) file and stores in a DataFrame	`pd.read_csv (path_to_csv datafile)`	Python Basics for Data Science
`DataFrame.describe()`	Returns a table with basic statistics for a dataset including min, max, mean, count, and quartiles	`DataFrame.describe()` Where: `DataFrame` is the name of the DataFrame.	Python Basics for Data Science
`DataFrame.iloc[]`	Allows access to data in a DataFrame using row/column integer-based indexes.	`DataFrame.iloc[row, column]` Where: `DataFrame` is the name of the DataFrame.	Python Basics for Data Science

Table D1

Python Function	Description	Syntax	First Reference
`DataFrame.loc[]`	Used to access a group of rows and columns by labels or a Boolean array	`DataFrame.loc[criteria]` Where: `DataFrame` is the name of the DataFrame.	Python Basics for Data Science
`Plt.scatter()`	Generates a scatterplot for (*x*, *y*) data	`plt.scatter(x_data, y_data)`	Python Basics for Data Science
`Plt.title()`	Specifies a title for a chart	`plt.title("Title")`	Python Basics for Data Science
`Plt.xlabel()`	Specifies a label for the *x*-axis	`plt.xlabel("x-axis label")`	Python Basics for Data Science
`Plt.ylabel()`	Specifies a label for the *y*-axis	`plt.ylabel("y-axis label")`	Python Basics for Data Science
`Plt.xlim()`	Specifies limits to use for *x*-axis numbering	`plt.xlim(lower, upper)`	Python Basics for Data Science
`Plt.ylim()`	Specifies limits to use for *y*-axis numbering	`plt.ylim(lower, upper)`	Python Basics for Data Science
Collecting and Preparing Data			
`pd.read_html()`	Read HTML table from a web page and convert into a DataFrame	`pd.read_html(URL)`	Web Scraping and Social Media Data Collection
`pd.to_numeric()`	Converts strings or other data types to numeric values	`pd.to_numeric (column_name)`	Web Scraping and Social Media Data Collection
`len()`	Returns the length of an object	`len(object)`	Web Scraping and Social Media Data Collection

Table D1

Python Function	Description	Syntax	First Reference
`re.findall()`	Returns all non-overlapping matches of a specified pattern in a string	`re.findall(pattern, string)`	Web Scraping and Social Media Data Collection
`re.search()`	Checks if a specified pattern appears in a string	`re.search(pattern, string)`	Web Scraping and Social Media Data Collection
Descriptive Statistics: Statistical Measurements and Probability Distributions			
`binom.pmf()`	Calculates the probability mass function (PMF) for a binomial distribution. It gives the probability of having exactly *x* successes in *n* trials with success probability *p*.	`binom.pmf(x, n, p)` Where: *x* is the number of successes in the experiment, *n* is the number of trials in the experiment, *p* is the probability of success.	Discrete and Continuous Probability Distributions
`round()`	Rounds a numeric result to a specified level of precision	`round(number, digits)`	Discrete and Continuous Probability Distributions
`poisson.pmf()`	Calculates probabilities associated with the Poisson distribution	`poisson.pmf(x, mu)` Where: *x* is the number of events of interest, *mu* is the mean of the Poisson distribution.	Discrete and Continuous Probability Distributions

Table D1

Python Function	Description	Syntax	First Reference
`norm.cdf()`	Calculates probabilities associated with the normal distribution (returns the area under the normal probability density function to the left of a specified measurement)	`norm.cdf(x, mu, std)` Where: *x* is the measurement of interest, *mu* is the mean of the normal distribution, *std* is the standard deviation of the normal distribution.	Discrete and Continuous Probability Distributions
Inferential Statistics and Regression Analysis			
`t.ppf()`	Generates the value of the *t*-distribution corresponding to a specified area under the *t*-distribution curve and specified degrees of freedom	`t.ppf` `(area to left, degrees of freedom)`	Statistical Inference and Confidence Intervals
`bootstrap()`	Performs bootstrap process to generate confidence interval	`bootstrap` `(data, statistic, confidence_level, number_resamples)`	Statistical Inference and Confidence Intervals

Table D1

Python Function	Description	Syntax	First Reference
`norm.interval()`	Calculates confidence interval for the mean when population standard deviation is known, given sample mean, population standard deviation, and sample size (uses normal distribution). Note: Standard error is the standard deviation divided by the square root of the sample size.	`norm.interval` (*conf_level, sample_mean, standard_error*)	Statistical Inference and Confidence Intervals
`t.interval()`	Calculates confidence interval for the mean when population standard deviation is unknown, given sample mean, sample standard deviation, and sample size (uses *t*-distribution). Note, standard error is the standard deviation divided by the square root of the sample size.	`t.interval` (*conf_level, degrees_freedom, sample_mean, standard_error*)	Statistical Inference and Confidence Intervals

Table D1

Python Function	Description	Syntax	First Reference
`proportion_confint()`	Calculates confidence interval for a proportion (uses normal distribution)	`proportion_confint (success, sample_size, alpha)`	Statistical Inference and Confidence Intervals
`ttest_1samp()`	Returns the value of the test statistic and the two-tailed *p*-value for a one-sample hypothesis test using the *t*-distribution	`ttest_1samp (data_array, null_hypothesis_mean)`	Hypothesis Testing
`ttest_ind_from_stats()`	Returns the value of the test statistic and the two-tailed *p*-value for a two-sample hypothesis test using the *t*-distribution	`ttest_ind_from_stats (sample_mean1, sample_standard_deviation1, sample_size1, sample_mean2, sample_standard_deviation2, sample_size2)`	Hypothesis Testing
`np.array()`	Creates a numerical array from a list-like object	`np.array(object)`	Correlation and Linear Regression Analysis
`pearsonr()`	Calculates the value of the Pearson correlation coefficient *r*	`pearsonr (x_data, y_data)`	Correlation and Linear Regression Analysis

Table D1

Python Function	Description	Syntax	First Reference
`linregress()`	Generates a linear regression model and provides slope, *y*-intercept, and other regression-related output	`linregress (x_data, y_data)`	Correlation and Linear Regression Analysis
`f_oneway()`	Returns both the *F* test statistic and the *p*-value for the one-way ANOVA hypothesis test	`f_oneway (Array1, Array2, Array3, …)`	Analysis of Variance (ANOVA)
Time Series and Forecasting			
`plot()`	Generates a time series plot	`plot(dataframe)`	Introduction to Time Series Analysis
`rolling()`	Provides rolling window calculations	`rolling (window=window)`	Time Series Forecasting Methods
`mean()`	Computes the average of a dataset	`mean(dataset)`	Time Series Forecasting Methods
`diff()`	Computes the first-order difference of data in a window	`diff(dataframe)`	Time Series Forecasting Methods
`plot_acf()`	Plots the ACF (autocorrelation function) for a time series, up to lag *L*	`Plot_acf (time_series_data, lags=L)`	Time Series Forecasting Methods

Table D1

Python Function	Description	Syntax	First Reference
`STL()`	Decomposes a time series with known period *P* into its components	`STL (time_series_data, period=P)`	Time Series Forecasting Methods
`ewm()`	Performs exponential moving average (EMA) smoothing	`ewm(dataframe)`	Time Series Forecasting Methods
`adfuller()`	Performs the Augmented Dickey-Fuller (ADF) test, which is a statistical test for checking the stationarity of a time series	`adfuller (time_series_data)`	Time Series Forecasting Methods
`ARIMA()`	Fits an ARIMA(*p, d, q*) (AutoRegressive Integrated Moving Average) model to time series data	`ARIMA (time_series_data, order=(p, d, q))`	Time Series Forecasting Methods
Decision-Making Using Machine Learning Basics			
`LogisticRegression()`	Creates a logistic regression model	`LogisticRegression()`	Classification Using Machine Learning
`model.fit()`	Trains a machine learning model on a given dataset	`model.fit (feature_matrix, target_vector)`	Classification Using Machine Learning

Table D1

Python Function	Description	Syntax	First Reference
`KMeans()`	Sets up a k-means clustering model (Use model.fit() to fit the model to a dataset.)	`KMeans(n_clusters=k)`	Classification Using Machine Learning
`DBSCAN()`	Sets up a DBSCAN (Density-Based Spatial Clustering of Applications with Noise) model (Use model.fit() to fit the model to a dataset.)	`DBSCAN(options)`	Classification Using Machine Learning
`confusion_matrix()`	Used to visualize the performance of a model by comparing actual and predicted values	`confusion_matrix (target_values, predicted_values)`	Classification Using Machine Learning
`LinearRegression()`	Fits a linear regression model to data	`LinearRegression() .fit(feature_matrix, target_vector)`	Machine Learning in Regression Analysis
`predict()`	Used on trained machine learning models to generate predictions for new data points	`predict(feature_matrix)`	Machine Learning in Regression Analysis

Table D1

Python Function	Description	Syntax	First Reference
`DecisionTreeClassifier()`	Sets up a decision tree model (Use model.fit() to fit the model to a dataset.)	`DecisionTreeClassifier(`*`options`*`)`	Decision Trees
`ens.RandomForestRegressor()`	Sets up a random forest model (Use model.fit() to fit the model to a dataset.)	`ens.RandomForestRegressor(`*`options`*`)`	Other Machine Learning Techniques
`GaussianNB()`	Set up a Naïve Bayes classification model (Use model.fit() to fit the model to a dataset.)	`GaussianNB()`	Other Machine Learning Techniques
Deep Learning and Artificial Intelligence (AI) Basics			
`Perceptron()`	Sets up a perceptron model (Use model.fit() to fit the model to a dataset.)	`Perceptron()`	Introduction to Neural Networks
`train_test_split()`	Splits dataset randomly into train and test subsets, using a proportion of *P* of the data for the test set	`train_test_split(`*`input_data_arrays, target_data,`* `test_size=`*`P`*`)`	Introduction to Neural Networks
`StandardScaler()`	Used to standardize features by removing the mean and scaling to unit variance	`StandardScaler()`	Introduction to Neural Networks

Table D1

Python Function	Description	Syntax	First Reference
`accuracy_score()`	Calculates the accuracy of a classification model as the ratio of the number of correct predictions to the total number of predictions	`accuracy_score(y_true, y_predicted)`	Introduction to Neural Networks
`scaler.fit_transform()`	Fits a scaler to the data and then transforms the data according to the fitted scaler	`scaler.fit_transform(array)`	Introduction to Neural Networks
`scaler.transform()`	Applies a previously fitted scaler to new data	`scaler.transform(array)`	Introduction to Neural Networks
`tf.keras.Sequential()`	Creates a linear stack of layers for building a neural network model	`tf.keras.Sequential(layers, additional options)`	Backpropagation
`model.compile()`	Used to configure the learning process of a neural network model before training	`model.compile(optimizer, loss, metrics)`	Backpropagation
Visualizing Data			
`boxplot()`	Creates a box-and-whisker plot	`plt.boxplot(array)`	Encoding Univariate Data

Table D1

Python Function	Description	Syntax	First Reference
`hist()`	Creates a histogram	`plt.hist (array)`	Encoding Univariate Data
`plot()`	Creates 2D line plots such as a time series graph	`plt.plot (x_data, y_data)`	Graphing Probability Distributions
`bar()`	Creates a bar chart	`plt.bar (x_array, heights)`	Graphing Probability Distributions
`imshow()`	Displays an image on a 2D regular raster, such as a heatmap	`plt.imshow(array)`	Geospatial and Heatmap Data Visualization Using Python
`heatmap()`	Creates a heatmap visualization	`sns.heatmap(array)`	Geospatial and Heatmap Data Visualization Using Python
`colorbar()`	Adds a colormap to a figure	`plt.colorbar()`	Multivariate and Network Data Visualization Using Python
`corr()`	Calculates the pairwise correlations of columns in a DataFrame	`dataframe.corr()`	Multivariate and Network Data Visualization Using Python
`add.subplot()`	Adds a subplot to a figure stored in fig	`fig.add.subplot (position)`	Multivariate and Network Data Visualization Using Python
`ax.scatter()`	Creates a scatterplot	`ax.scatter (x_data, y_data)`	Multivariate and Network Data Visualization Using Python
Reporting Results			

Table D1

Python Function	Description	Syntax	First Reference
`plot_tree()`	Creates a visualization of a decision tree	`plot_tree` (*estimator, feature_names*)	Validating Your Model
`DataFrame.info()`	Provides a concise summary of a DataFrame's structure and content	`DataFrame.info()`	Validating Your Model
`DataFrame.drop()`	Removes rows or columns from a DataFrame	`DataFrame.drop` (*labels*, axis=*rows_columns*)	Validating Your Model
`score()`	Evaluates the performance of a trained model on a given dataset	`model.score` (*feature_matrix, true_labels*)	Validating Your Model
`dt.get_depth()`	Retrieves the depth of the decision tree, dt	`dt.get_depth()`	Validating Your Model
`cross_val_score()`	Evaluates a model's performance using cross-validation	`cross_val_score` (*estimator, feature_matrix, target_variable*)	Validating Your Model
`GridSearchCV ()`	Search for the best parameters for a specified estimator, with *k*-fold cross-validation	`GridSearchCV` (*estimator, parameters, k*)	Validating Your Model

Table D1

Answer Key

Chapter 1

Chapter Review

1. **b.** Data preparation: have the collected data stored in a server as is so that you can start the analysis.
2. **a.** Initially, data was stored locally on individual computers, but with the advent of cloud-based systems, data is now stored on designated servers outside of local storage.
3. **c.** A biologist analyzing a large dataset of genetic sequences to gain insights about the genetic basis of diseases

Chapter 2

Chapter 3

Quantitative Problems

1. **a.** $\text{Mean} = 16.4,\ 10\%\ \text{Trimmed mean} = 12.9$
   b. See Solution Manual
2. **a.** $\text{Mean} = 5.7,\ \text{Median} = 4.1$
   b. See Solution Manual
   c. See Solution Manual
3. **a.** $\text{Range} = 8.7,\ \text{Standard Deviation} = 2.5,\ \text{Variance} = 6.5$
   b. See Solution Manual
   c. See Solution Manual
4. **a.** $Q_1 = 386.45,\ Q_2 = 878.32, Q_3 = 1182.42$
   b. $\text{IQR} = 795.97$
   c. $\text{Outlier data value} = 4303.76$
5. $\text{Percentile} = 50\%$
6. **a.** $z = -3.04$
   b. See Solution Manual
   c. $z = 3.48$
   d. See Solution Manual
7. $\text{Probability} = 0.005$
8. **a.** 0.074
   b. 0.195
   c. 0.805
9. **a.** See Solution Manual
   b. 0.63
10. **a.** See Solution Manual
    b. 0.035
    c. 0. 272
11. 0.272
12. 0.026
13. 0.085
14. **a.** 0.112
    b. 0.691

Chapter 4

Chapter 5

Quantitative Problems

1. Most likely, the period would be 52, as sales probably go through a cycle each year, and there are 52 weeks in the year.
2. **a.** There are peaks in the data at times 1, 5, and 9 as well as minimum points at times 4, 8, and 12, suggesting a period of 4 months.
   b. (728.6, 729.4, 728.6, 727.3, 727.1, 727.0, 726.0, 725.8)
   There will be 8 terms in the centered SMA, as shown. These values form the trend-cycle component.
   c. (69.9, −225.3, 159.8, −1.5, 66.3, −228.0, 165.7, −3.8)
   d. (68.1, −226.65, 162.75, −2.65), repeating in blocks of 4
   e. (812.6, 953.85, 635.75, 506.75, 820.3, 952.45, 630.65, 501.65, 823.6, 948.65, 626.35, 504.25)
3. **a.** Since the missing data is due to different reporting policies on different days of the week, an SMA window of 7 days is most appropriate.
   b. See Solution Manual
4. **a.** $\text{Period} = 3$ from the ACF plot.
   b. See Solution Manual
5. Forecasted value for 2024: 1.2124
6. **a.** $\text{MAE} = 421.6$. $\text{RMSE} = \sqrt{313,168.55} = 559.6$. $\text{MAPE} = 0.123$, or 12.3%. $\text{sMAPE} = .0133$, or 13.3%.
   b. See Solution Manual

Chapter 6

Chapter Review

1. **c.** k-means clustering

Quantitative Problems

1. **a.** MAE = 0.0415. MAPE = 0.41%. MSE = 0.003126. RMSE = 0.0559.
   b. 9.60. This is only about 2.3% off from the actual value of 9.83.
2. $\frac{1}{1+e^{1.735-2.488x}}$
3. **a.** 1
   b. 3
   c. 25.85
4. **a.** 1.203
   b. 1.919

Chapter 7

Chapter Review

1. **b.** To provide a way for the network to learn and represent complex patterns and relationships within the data
2. **c.** A CNN is a type of neural network that uses convolutional layers to process grid-like data structures, such as images, and is particularly effective for tasks like image classification, object detection, and recognizing spatial relationships.
3. **d.** These algorithms are important because they enable hands-free interaction with devices, improve accessibility for individuals with disabilities, and enhance user experience in various applications.
4. **c.** Designing a virtual assistant that collects personal data without explicit user consent and sells it to third-

party advertisers

Quantitative Problems

1. **a.**
   i. 0
   ii. -0.0848
   iii. 0.396
   iv. 0.503

   b. See Solution Manual
2. **a.** 0.018
   b. 0.499

Chapter 8

Chapter Review

1. **a.** To ensure individuals are aware of the potential risks and benefits of the research
2. **d.** Data scientists
3. **c.** Ensuring secure storage and access protocols
4. **d.** All of the above
5. **a.** Have strong data security policies and protocols in place
6. **d.** All of the above
7. **d.** Quantitative, qualitative, tabular, text-based, and audio-based data
8. **c.** Anyone who is directly involved in the project and has a legitimate need for the data
9. **c.** Legal and compliance teams
10. **a.** To increase the accuracy of the model by avoiding bias
11. **b.** Dataset 2, the results of an extensive survey of people from various backgrounds and demographic segments who may or may not be familiar with the new social media platform
12. **b.** Data points that are rare or abnormal in comparison to the rest of the data
13. **a.** To ensure models are not overfitting
14. **a.** Name, address, and email address
15. **a.** When the individual has given consent for anonymization
16. **d.** All of the above
17. **d.** Any person who will use the data
18. **a.** Data attribution
19. **c.** Ensuring accessibility and inclusivity
20. **c.** Graph C

Chapter 9

Chapter 10

Chapter Review

1. **c.** Concise with a focus on summaries and outcomes
2. **b.** Technical jargon
3. **b.** Connecting with diverse audiences through tailored communication
4. **b.** Practical application and procedures
5. **a.** To provide different levels of detail for various audiences
6. **c.** To clarify and accentuate significant data insights
7. **d.** With clear labels, titles, and straightforward explanations
8. **b.** To steer the decision-making process

9. **b.** It is specific and verifiable.
10. **b.** Seventy-five percent of the variance in housing prices is explained by the model.
11. **b.** To understand the model's performance more deeply
12. **a.** To test the robustness of results against input variability
13. **c.** To summarize complex analyses for an audience of varied backgrounds
14. **b.** An introduction to the business problem or research question
15. **b.** The novelty or relevance of the approach
16. **a.** With a persuasive call to action

Index

Symbols

3D visualization 457

A

A prime 123
accuracy 270
Accurate representation 399
ACF (autocorrelation function) 244
Actionable advice 489
Activation 338
activation function 338
adaptive learning 14
additive decomposition 224
Advanced NLP models 370
advertising 195
`AI Fairness 360` 373
alt text 470
alternative hypothesis 167
Amadeus Code 367
Amazon 13
Analysis of variance (ANOVA) 205
Anonymization 384
Anonymous data 386
application programming interface (API) 68
`apply.map` 478
arithmetic mean 106
Artificial intelligence (AI) 335
Association of Data Scientists 382
Association of Data Scientists (ADaSci) 31
assumption 473
at least one occurrence 127
attributes 19
audience 466
Augmented Dickey-Fuller (ADF) test 252
Autocorrelation 241
autonomy 382
Autoregressive (AR) model 251
autoregressive integrated moving average (ARIMA) model 219
`ax.scatter()` 458

B

Backpropagation 345
`bar` 417, 440
Bar graphs 416
Bayes' Theorem 128
BCNF (Boyce-Codd Normal Form) 82
Beautiful Soup 72
Bernoulli trials 132
best-fit linear equation 197
Bias 270, 338, 393
big data 90, 327, 449
bimodal 110
binary (binomial) classification 278
Binary cross entropy loss 346
`binom()` function 139
binomial distribution 131, 156, 435
binomial experiment 131
bins 424
bivariate data 190, 416
bootstrap aggregating (bagging) 321
bootstrap samples 482
bootstrapping 161, 298
box-and-whisker plot 416
boxplot 416, 417
Bureau of Labor Statistics (BLS) 22, 54
business 13

C

Careers in Data Science 16
Categorical data 19
cell 32
central limit theorem 150
central tendency 106
centroid 284
ChatGPT 363
choropleth graph 447
cirrhosis.csv 326, 332
Closed-ended questions 64
Cloud computing 93
Cloud storage 91
cluster 278
cluster sampling 65
codebook 470
coefficient of variation (CV) 116
Colab 36
CollegeCompletionData.csv 296, 308, 332
comma-separated values (CSV) 21
complement of an event 123
CoNaLa 368
conditional probability 124
confidence interval 148
confidence intervals 136
confidence level 149
Confidentiality 386
confusion matrix 294
connecting weight 352
constraints 473
consumer analysis 105
continuous 434
continuous data 18
Continuous probability distributions 130
Continuous random variable 130
Convenience sampling 65
Convolutional layers 361
Convolutional neural networks (CNNs) 361
Cookies 386
copyright 388
Correctional Offender Management Profiling for Alternative Sanctions 393
correlation 147
correlation analysis 147, 189, 454
correlation coefficient 191
correlation heatmap 455
critical value 152
Cross-validation 397, 481
cryptocurrencies 388
CSV library 76
customer surveys 111
`cusum()` 428
Cyclic component 224

D

DALL-E and Craiyon 366
dashboards 449

data 17
Data aggregation 86
data analysis 11
data breach 383
Data chunking 92
data cleaning 328
Data collection 11, 60, 382
Data collection and preparation 59
Data compression 90
data dictionary 76
Data discretization 78
data governance protocols 391
Data indexing 91
Data integration 78
data irregularities 82
data management 90
Data mining 328
data preparation 11
Data privacy 381
Data processing 75
Data reporting 12
data retention 383
Data science 9
Data Science Association 382
Data Science Association (DSA) 31
data science cycle 11
data security 382, 384
Data sharing 390
Data source attribution 405
data sovereignty 383
Data standardization 81
Data transformation 82, 84
Data validation 82, 85, 397
Data visualization 12, 415
data volumes 59
data warehouse 91
data warehousing 12
Data.gov 22, 54
Database management 92
DataFrame 38, 70
`DataFrame.describe()` 116
dataset 19
`DBSCAN` 293
DBScan algorithm 291
DBScanExample.csv 293, 331
decision tree 311
decision-making 105
Deep learning 357
Deepfake 373
degree 234
density-based clustering algorithm 291
Dependent events 124
dependent samples 181
dependent variable 189
depth 315, 358
Depth-limiting pruning 319
Descript 367
descriptive statistics 105
Detrending 229
`df.info()` 477
differencing 230, 233
digital divide 406
dimension 339
discrete 434
discrete data 18
Discrete probability distributions 130
Discrete random variable 129, 435
dynamic backpropagation 346

E

Education 14, 369
ELIZA 364
empirical probability 123
empirical rule 137
encryption 387
energy consumption 14
engineering 13
engineering analysis 105
Entropy 312
epoch 349
Equifax 387
error 218
error-based (reduced-error) pruning 317
ethical data collection 382
Ethics in data science 381
event 122
Excel 29, 106, 133, 136, 161, 219, 220, 226, 235, 369, 417
executive (or data) dashboard 467
executive summary 489
Executives 467
experimental research 60
Experts 467
Explainable AI (XAI) 394
exploding gradient problem 353
expon 138
exponential moving average (EMA) 230
Exponential transformations 84
Extensible Markup Language (XML) 21

F

`f_oneway()` 207
F-distribution 205
F1 Score 273
Facial recognition 324
`Fairlearn` 373
Fairness 393
false positive 128
Family Educational Rights and Privacy Act (FERPA) 384
feature maps 361, 362
feedback loops 352
`fig.add.subplot()` 458
film industry 370
five-number summary 418
flat forecasting method 218
folds 482
forecasting 218
forecasting models 148
frequency 107
frequency distribution 107
Fully connected layers 361
FungusLocations.csv 288, 331

G

Gaussian naïve Bayes 326
Gemini 369
Generative art 366
Geospatial data 443
Geospatial Information System (GIS) 447
Gini index 313
GitHub 391
GitHub Copilot 368
Google 369
Google Colab 368, 391
Google Colaboratory (Colab) 31

Google Sheets 29
gradient descent 349
graphical displays 106
Grid heatmaps 444

H

hallucinations 372
hashing 395
Health Insurance Portability and Accountability Act (HIPAA) 384
heatmap 295, 456
hidden layers 337
Hinge loss 347
`hist()` 426
histogram 423
`HolisticAI` 373
Huffman coding 90
Hyperbolic tangent (tanh) function 339
hyperparameter tuning 481
hypothesis testing 136, 167

I

imbalanced data 359
Implicit bias 475
imputation 79
inconsistency 82
Independent events 124
independent samples 181
independent variable 189
indexing 70
Inferential statistics 147
information 17
information gain 314
Information theory 311
Informed consent 385
Initiative for Analytics and Data Science Standards 382
Initiative for Analytics and Data Science Standards (IADSS) 31
input layer 337
Integrative (I) component 251
Intellectual property 388
Internet of Things (IoT) 13
interquartile range (IQR) 119, 420
items 19

J

jargon 467
JavaScript Object Notation (JSON) 21
Jukebox 367
Jupyter Notebook 31

K

k-anonymization 396
k-fold cross-validation 482
k-means clustering algorithm 283
Kaggle 22, 54, 391
Kapwing: 367
key performance indicators (KPIs) 467

L

Labeled data 271
lag 241
large language model 363
layered approach 469
Leaf-limiting pruning 319
Leaky ReLU function 340
learning management systems 14
least squares method 195
Leave-one-out cross-validation (LOOCV) 482
level 225
level of significance 170
likelihood 281
line chart 428
linear correlation 191
linear regression 148
`linregress()` 199, 202
log transformation 84
logistic regression 278
`LogisticRegression` 283
logit function 280
long short-term memory (LSTM) network 353
loss or cost function 346
Lossless compression 90
Lossy compression 90
lower bound 119
Lyria model 367

M

machine learning (ML) model 270
machine learning life cycle 270
Magenta Project 367
`map` 478
MAR (missing at random) data 79
margin 359
margin of error 149
margin of error formula 158
matched pairs 185
Matplotlib 30, 373, 417, 418, 457
matplotlib.pyplot 222
MCAR (missing completely at random) data 79
mean 106
Mean absolute error (MAE) 257, 273
Mean absolute percentage error (MAPE) 257, 273
mean percentage error (MPE) 483
Mean squared error (MSE) 273
Measurement errors 67
measures of error 257
measures of fit 257, 476
median 109, 416
memory cells 353
metadata 92
method of least squares 197
Microsoft 369
Minimum description length (MDL) pruning 319
Missing data 78
MNAR (missing not at random) data 79
mode 110
modeling 473
Monte Carlo simulation 485
MonthlyCoalConsumption.xlsx 225, 235, 238, 253, 262
`mpl_toolkits.mplot3d()` 458
`mpl.toolkits.mplot3d` 457
Multi-way sensitivity analysis 485
multiclass (multinomial)

classification 278
multilayer perceptron (MLP) 341
multiple regression 302
multiplicative decomposition 224

N

naïve 218
naïve Bayes classification 324
National Oceanic and Atmospheric Administration (NOAA) 22, 54
natural language processing 363
Natural Language Toolkit (NLTK) 72
NCAA-2021-stats.csv 298, 303
Network analysis 69
neural network 336
neurodiversity 469
neurons 336, 338
NLP models 364
Noise 224, 228
Noisy data 84
nominal data 19
non-response bias 67
nonlinear 340
nonparametric method 161
Nonspecialists 468
`norm.cdf()` 140
`norm()` 140
normal distribution 136, 424
normal form (NF) 82
Normalization 82
normalized 248
NoSQL databases 91
`np.array()` function 192
null hypothesis 167
Numeric data 18
numpy 305
NumPy 30

O

object storage 91
Observational data 60
odds 280
one-hot encoding 322
one-way ANOVA 205
One-way sensitivity analysis 485
Open-ended questions 64
OpenArt 366
Order 234
ordinal data 19
outcome 122
outlier 78
Outlier detection 397
outliers 59, 106, 416
output layer 337
Overfitting 275

P

p-value 171
$P(A|B)$ 124
paired samples 181
Panda 30
pandas 70, 76, 428, 478
pandas library 221
Parametric methods 161
Pareto 427
Pareto chart 427
peak 241
`Pearsonr()` 199, 201
Percentiles 117
perceptron 341
period 227
personally identifiable information (PII) 386, 389
Pew Research Center 22, 54
`plot` 417
`plot.hist()` 426
Plotly 457
`plt.scatter()` 199
point estimate 148
point-in-time data 217
poisson 138, 440
Poisson distribution 134, 439
Pooling layers 361
Population data 107
population mean 107, 148
population proportion 148
population size 107
population variance 114
potential for bias 372
Power BI 391
Precision 273
Precision Medicine Initiative 15
prediction interval 258
predictions 193
predictive analytics 15
principal component analysis (PCA) 272
prior probability 324
Probability 121
Probability analysis 106
probability density function (PDF) 135, 435, 442
Probability distribution 129
probability experiment 123
probability mass function (PMF) 132
programming language 29
proportion 156
Pruning 317
Pseudonymization 384
Public Policy 14
purposive sampling 66
Python 29, 136, 164, 221, 222, 241, 258, 417

Q

qualitative 111
Qualitative data 18, 388, 434
quantitative 111
Quantitative data 18, 388, 434
quartiles 118, 416
Quota sampling 66

R

R 29, 106, 133, 136
Random Error 224
random forest 321
Random variable 129
Range 113
`re` 72
`read_html()` 70
Recall 273
recommendation systems 16
Rectified linear unit (ReLU) function 339
recurrent neural networks (RNNs) 352
RedBlue.csv 319
redundancy 82
regex 72
Regression analysis 147

regression models 483
regression tree 314
Regular expressions 72
regulatory compliance officer (RCO) 385
Relational databases 91
Relative frequency probability 122
`replace` 478
reproducibility, 470
residual 197
residuals 228
responsible AI 373
revenue 196
Root mean squared error (RMSE) 257, 273
`round()` function 139
Russian tank problem 270

S

S&P 500 190
sabermetrics 15
Sample data 107
sample mean 107, 148
sample proportion 149, 176
sample size 107
sample size requirement 158
sample space 122
sample statistic 152
sample variance 114
Sampling 63, 150
sampling (or sample) distribution 150
Sampling Bias 66
Sampling error 66
sampling frame 65
sampling techniques 64
sampling variability 67
Sampling with replacement 160
Sarbanes-Oxley Act [SOX] 383
scale-dependent 257
scaling 78
`scatter` 417
scatterplot 449
scatterplot, or scatter diagram 190
Scenario analysis 485
science 13
scientific experiments 63
`Scikit-Learn` 373
`scipy` library 207
`SciPy library` 164
scipy.stats 138, 302
scipy.stats library 155, 163, 192
Seaborn 30, 296
Seasonal component 224
seasonal variations 227
Seasonal-Trend decomposition using LOESS (STL) 230
seasonality 224, 227, 240
self-selection bias 67
semantic segmentation 361
Sensitivity analysis 485
sigmoid function 279, 339
silhouette score 284
Simon Says 367
simple moving average (SMA) 230
Simple random selection 64
`sklearn` 319, 321, 478
sklearn.cluster 293
sklearn.datasets 343
`sklearn.ensemble` 321
sklearn.linear_model 283
Slicing a string 74
smart city 14
Snowball sampling 66
`sns.heatmap()` 446
Social listening 69
Social media data collection 69
softmax 340
Softplus function 340
Sonar 368
SP500.csv 221, 262
SpaCy 72
Sparse categorical cross entropy 347
Spatial heatmaps 444
speech recognition 368
Splitting a string 74
sports analytics 15
Spreadsheet programs 29
square root transformation 84
standard deviation 114
standard error of the mean 152
standard normal distribution 136
standardized test statistic 170
standardizing data 82
static backpropagation 346
Statista 22, 54
statistical analysis 61, 105
statistical validation 476
statistics 301
statsmodels 244
statsmodels.tsa.arima.model 254, 258
Step function 339
Stratified sampling 65
structured dataset 20
sum of squares 114
Supervised learning 271
supervised learning cycle 271
survey 60
Symmetric mean absolute percentage error (sMAPE) 257
Systematic sampling 65

T

t-distribution 154
`t.ppf()` 155
Tableau 391
TabNine 368
Technicians 467
temps.csv 321, 322
tensor 350
terms 217
test statistic 170
`test_size=0.25` 343
testing set (or data) 270
Text preprocessing 87
text-to-speech (TTS) 368
TextBlob 72
Theoretical probability 123
time series 216
time series model 218
`to_datatime()` 222
`to_numeric()` 70
Tokenizing 72
topological data analysis (TDA) 272
training set (or data) 270
Transactional data 60
Transformer models 364
transparency 382
Trend 224
trend curve 225

trend-cycle component 224
trendline 225
trimmed mean 106, 108
`ttest_1samp()` 178
`twinx` 428
Type I error 170
Type II error 170

U

unbiased estimator 149
Underfitting 275
Univariate data 416
Universal design principles 406
Unlabeled data 271
unstructured data 59
unstructured dataset 20
Unsupervised learning 271
upper bound 119

V

Validation 481
vanishing gradient problem 353
variable 147
variance 114, 275
vector 339
version control system 470
volatility 220
Volunteer sampling 66
VTK 458

W

Walmart 13
weak learners 321
weather forecasting 13
Web scraping 68
weight 338
weighted moving average (WMA) 230
`What-If Tool` 373
white noise 228
window 230
World Health Organization (WHO) 22

X

XML tag 24

Z

z-score 120