FUNDAMENTALS OF DATA SCIENCE

Sanjeev J. Wagh, Manisha S. Bhende, and Anuradha D. Thakare

A **Chapman & Hall** Book

CRC Press
Taylor & Francis Group

Fundamentals of Data Science

Fundamentals of Data Science

Sanjeev J. Wagh, Manisha S. Bhende, and Anuradha D. Thakare

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

First edition published 2022
by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742

and by CRC Press
2 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 978-1-138-33618-6 (hbk)
ISBN: 978-1-032-07986-8 (pbk)
ISBN: 978-0-429-44323-7 (ebk)

DOI: 10.1201/9780429443237

Typeset in Palatino
by codeMantra

Contents

Preface

The best way to teach data science is to focus on the work culture in the data analytics business world. Data science is touted as the sexiest job of the 21st century. Everyone – from companies to individuals – is trying to understand it and adopt it. If you're a programmer, you most definitely are experiencing FOMO (fear of missing out)! Data science helps you be data-driven. Data-driven decisions help companies understand their customers better and build great businesses. Companies collect different kinds of data depending upon the type of business they run. For example, for a retail store, the data could be about the kinds of products that its customers buy over time and their spending amounts. For Netflix, it could be about what shows most of their users watch or like and their demographics.

This book consists of 11 chapters, encompassing data science concepts to hands-on tools to practice, which will support aspirants to the data science domain. Chapter 1 introduces the importance of data science and its impact on society. The process, prerequisites, components, and tools are discussed to enhance the business strategies. Chapter 2 defines the concepts of statistics and probability, which are the basis to understand data science strategies. To understand the background of data science, traditional SQL practices are discussed with the recent popular tool, i.e., NoSQL, with its importance and the techniques to handle more complex databases in Chapter 3. Chapter 4 elaborates on data science methodology with data analysis and life cycle management. The analytics for data science with some examples is discussed. The data analytics life cycle with all phases is discussed to in line with understanding the data science techniques. Chapter 5 outlines data science methods and machine learning approach with various techniques for regression analysis and standard machine learning methods. Data analytics and text mining are precisely elaborated with natural language processing in Chapter 6. Chapters 7–11 cover various platforms and tools to practice the data science techniques with a practical approach. This section covers Python, R, MATLAB, GNU Octave, and Tableau tools precisely with sample examples to practice and apply for implementing desired applications in the theme domain.

This will be a textbook tool that provides students, academicians, and practitioners with a complete walk-through right from the foundation required, outlining the database concepts and techniques required to understand data science. This book isn't meant to be read cover to cover or page to a chapter that looks like something you're trying to accomplish or that simply ignites your interest in data analytics for implementation of real project applications.

MATLAB® is a registered trademark of The MathWorks, Inc. For product information,

please contact:
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

Authors

Sanjeev Wagh is working as **Professor and Head** in the Department of Information Technology at **Govt. College of Engineering, Karad**. He has completed his BE (1996), ME (2000), and PhD (2009) in Computer Science and Engineering from Govt. College of Engineering in Pune and Nanded. He was a full-time **postdoctoral fellow** at Center for TeleInFrastructure, **Aalborg University, Denmark,** during 2013–14. He has also completed MBA (IT) from NIBM (2015), Chennai. He has a total of 24 years of experience in academics and research. His research interest areas are natural science computing, Internet technologies and wireless sensor networks, and data sciences and analytics. He has published **100+ research papers** to his credit, published in international/national journals and conferences. Four research scholars completed PhD under his supervision from Pune University. Currently, three research scholars are pursuing PhD under his supervision in various Indian universities. He is a fellow member of ISTE and IETE and a member of IEEE, ACM, and CSI. He is co-editor for international journals in engineering and technology. He has visited Denmark (Aalborg University in Aalborg and Copenhagen), Sweden (Gothenburg University, Gothenburg), Germany (Hamburg University, Hamburg), Norway (University of Oslo), France (University of London Institute in Paris), China (Shanghai Technology Innovation Center, Shanghai – delegation visit), Thailand (Kasetsart University, Bangkok), and Mauritius (University of Technology, Port Louis) **for academic and research purposes.**

Manisha S. Bhende is working as Associate Professor in Dr. D.Y. Patil Institute of Engineering Management and Research, Pune. She has completed BE (1998), ME (2007), and PhD (2017) in Computer Engineering from the University of Pune and a bachelor's degree from Government College of Engineering, Amravati, India. Her research interests are IoT and wireless networks, network security, cloud computing, data science, machine learning and data analytics. She has published 39 research papers/book chapters in international/national conferences and journals. She delivered expert talks on various domains such as wireless communication, wireless sensor networks, data science, cybersecurity, IoT, embedded and real-time Operating systems, and IPR and innovation. She has four patents and three copyrights to her credit. She is a reviewer/examiner for PhD theses and ME dissertations

for state/national universities. She is associated with PhD research centers. She is working as an editor/reviewer for various reputed national/international journals and conferences. She is the coordinator of IQAC, IPR Cell, IIP Cell, and Research Cell at the institute level. She is working as Subject Chairman for various computer engineering subjects under Savitribai Phule Pune University (SPPU). She contributed to SPPU syllabus content design and revision. She received Regional Young IT Professional Award from CSI in 2006. She is a member of ISTE, ACM, CSI, IAENG, Internet Society, etc.

Anuradha D. Thakare received her PhD in Computer Science and Engineering from Sant Gadge Baba Amravati University, ME degree in Computer Engineering from Savitribai Phule Pune University (SPPU), and BE degree in Computer Science and Engineering from Sant Gadge Baba Amravati University, Amravati, India. She is working as a Professor in the Computer Engineering Department of Pimpri Chinchwad College of Engineering, Pune. Dr. Anuradha is Secretary of Institution of Engineering & Technology Pune LN and a member of IEEE and ACM. She is a PhD guide in Computer Engineering in SPPU, Pune. She has been a General Chair of IEEE International Conference ICCUBEA 2018 and an advisory member for international conferences. She worked as reviewer for the Journal of International Blood Research, IEEE Transactions, and Scopus journals. She is reviewer and examiner for PhD defense for state/national universities.

She has published 78 research papers in reputed conferences and journals with indexing in Scopus, IEEE, Web of Science, Elsevier, Springer, etc. She received research project grants and workshop grants from AICTE-AQIS, QIP-SPPU, BCUD-SPPU Pune, and Maharashtra State Commission for Women. She received the Best Women Researcher Award and Best Faculty Award from International Forum on Science, Health & Engineering. She received the best paper award in international conferences. She delivered 20 expert talks on machine learning, evolutionary algorithms, outcome-based education, etc. She worked with industries such as DRDO and NCL for research projects.

She is working as Subject Chairman for various computer engineering subjects under Savitribai Phule Pune University (SPPU). She contributed to SPPU syllabus content design and revision.

Part I

Introduction to Data Science

1

Importance of Data Science

1.1 Need for Data Science

Today lots of data are generated via various electronic gadgets. Earlier lots of data were stored in unstructured as well as structured format and lots of data flow happen. If you look today, data is created faster than it is imagined. While travelling by road, we can recognize lots of data being created, for example, vehicles speed, traffic light switching, Google map, etc. which get captured through satellites and transmitted to handheld devices in real time. It guides by showing several alternative paths with traffic intensity and a favorable path to take. We also get a suggestion on which route to take and which route to avoid. So there is a lot of data being created as we do our day-to-day activity. The problem is that we are not doing anything with the data. We are not able to analyze due to inefficient scientific insights.

We are creating data, but we are not utilizing it for behavioral analysis or predictions. If we look at the corporate world, we find that lots of data reports are generated. But what about utilization? How much time are we spending on preparation? We are not drawing any scientific insights; e.g., we are not using data for forecasting. It can be forecasted for sale in the coming months, seasonal time, or years. We can conclude from available data for decision making and prediction. From the data, we can visualize what the actual status of the data is and what decision we have to take. It is possible for shop owner to predict that if person wearing a red shirt and have age between 25 and 30, will be interested in the product. We can also see pattern discovery. There are lots of sales happening in November and December every year, that is, in the festival season, and also the changes in weather increase the sale, so actually we need to draw lots of pattern from data and do discovery. It needs to recognize the process of data analysis, data patterns and behaviors patterns using data science for increasing sale.

The upcoming automobile technology is an autonomous car. It is exciting to have a car driving automatically which will take you from home to office and office to home, and data science uses the coordinates to take a lot of decisions in this whole process. Decisions of whether to speed up, whether to apply the brake, whether to take left or right turn, and whether to slow down are a part

DOI: 10.1201/9780429443237-2

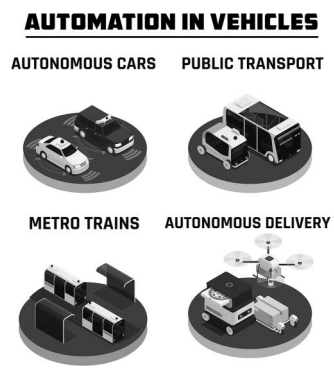


FIGURE 1.1
Autonomous vehicles.

of data science. The case study predicts that self-driving cars will minimize accidents, and in fact, it will save more than two million deaths caused by car accidents annually. Self-driving cars have a lot of scope for research design and testing in force to update knowledge of cars. Every automotive company is investing in self-driving cars. Studies say that in about 10–15 years, most of the cars will be autonomous self-driving cars [1]. The possible environment for autonomous vehicles is shown in Figure 1.1.

Another example, while booking travel plan, passenger are unaware about the weather condition as per decided travel schedule, hence need data science for predicting and alerting passenger about weather condition for taking decision of travel. Using data science, proper route planning can be done if in advance weather condition is predicted. Also air traffic managements system can plan the alternate route by rescheduling the flights as per environmental conditions. If the route is not planned properly, then you might end up in a situation where the flight is not available. Planning alternate flights between cities are challenging in last minutes due to preschedule air traffics. The roles of data science in airline industries are shown in Figures 1.2 and 1.3. If we use data science properly, most of these problems can be avoided and it will help in reducing the pain both for the airlines and for the passengers.

Few more applications can be possible in the airline industry such as better route planning so that there will be fewer cancellations and avoid the frustration of passengers. Predictive analytics can be used for predictions of any delays so that some flights can be rescheduled ahead of time and there will be no last-minute change. Data science can also be used to make promotional offers, select flights, and explain different classes of flight and different routes for better performance, as shown in Figure 1.3.

The logistics industry is another example where data science can be used. Companies such as FedEx use data science models to increase their efficiency by optimizing the routes and save cost. For shipping the materials, company

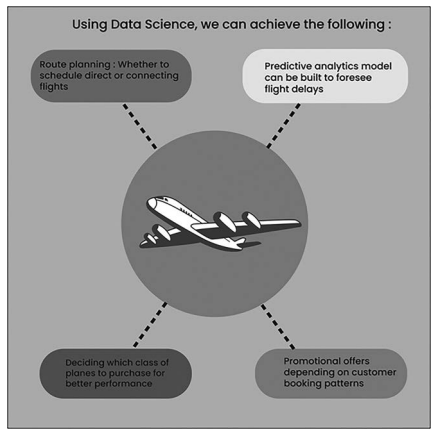


FIGURE 1.2
Data science and airline industries.

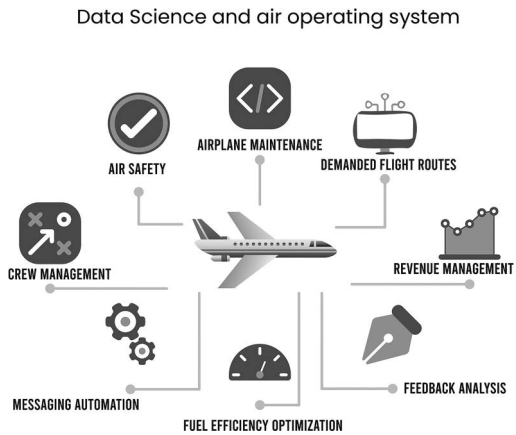


FIGURE 1.3
Use of data science in an airline operating system.

can use data science for prediction of possible routes, suitable time and best mode of transport for the delivery [4, 5].

The main purpose of data science is to compute better decision making. In some situations, there are always tricky decisions to be made, for example, on which prediction techniques are suitable for specific conditions, as shown in Figure 1.4. For example, can we predict delays in the case of an airline? Can we protect demand for a certain product? Another area is business in wearable sales. The apparels sales can be possible to schedule appropriately, if wearable pattern and demand recognized in time to increase the business. Also, selection patterns of apparels as per data of buying apparels in certain months,

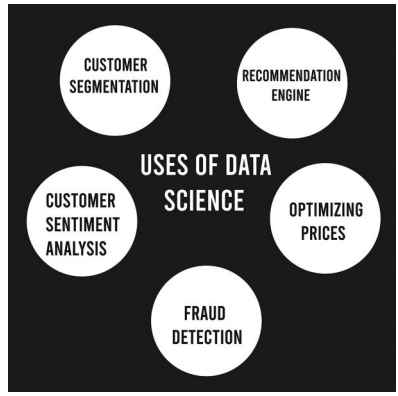


FIGURE 1.4
Uses of data science.

seasons, and so on help to decide sale time [2]. Another area of data science is predictive maintenance. For example, we need to find out whether our car or refrigerator will break down in the next year or two if we prepare to buy a new car or refrigerator. We can potentially apply data science here as well.

Currently, we find the use of data science effectively in politics. Most of the advertisements about election candidatures we see on television may be about elections in the USA or in the UK or even in India. Nowadays everybody is applying data science in elections and trying to capture the votes, or rather the voters influence the voters by personalizing messages or providing personalized messages. The voters use data science to predict who is going to win the elections and with what margin. Probably not all predictions come out to be true, as it depends on the quality of data and prediction techniques used. There are other domains where data science plays an active role as shown in Figure 1.5.

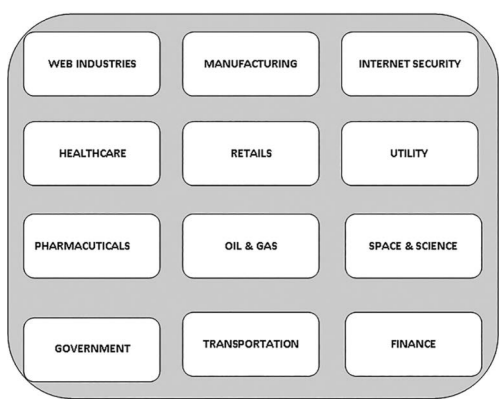


FIGURE 1.5
Data science domains.

1.2 What Is Data Science?

Jonathan Goldman joined LinkedIn in 2006. When he joined, LinkedIn was not as popular as it is today. There was no fancy activity, and people were started joining, and by adding their profiles and sharing other stuff. There was no much activity to do. But it was growing, that is, from several thousand to several million accounts, and they were having the problems of scaling. Jonathan Goldman found that the data gathered is very abnormal and there is something wrong in the structure. The LinkedIn management was just trying to get more people in to connect by searching relative data of peoples. There were no activities, discussion in network as per the interest of members, hence they had started leaving LinkedIn. He concluded something is missing in their data. The engineers of LinkedIn did not take his suggestion seriously. Then, the engineers gave Jonathan a free hand and asked to implement any idea if he has in mind. Jonathan Goldman introduced the idea known as LinkedIn recommendation system. He applied recommendations, and then, actually traffic started growing. Through the recommendation system, they target advertisements for various products. He had several thoughts in his mind for applications to LinkedIn such as co-relation with people. There is a chance that if A knows B and B knows C, then there is a good chance for A to know C. He had many such theories and applied his several ideas to LinkedIn. He is a good statistician and also a good programmer. He was sure that if he applies such concepts, traffic will increase definitely. By adding multiple facilities and interest domains, the LinkedIn management system can realize that traffic got increased since people started surfing, finding interest partners, jobs and business through this system and then LinkedIn started flourishing. So we can guess the power of data. The job executed by Jonathan Goldman in the way he utilized the data is the job of a data scientist. It is about an increase in revenue and about getting insights into the data, so the data can recognize the type of data and domains of expertize required. The LinkedIn story is the best example for the role of a data scientist [6]. Today we find the status of LinkedIn. For searching prospective candidate as per emergent requirements, the best possible option is LinkedIn recommendation system

You have the algorithm to deal with data, but it doesn't mean that the algorithm is perfect to analyze the data. If you have more and more data, there is a chance of more and more information and more kinds of scenarios. One of the simple and popular examples of a recommendation system is movies on your past references. If you try to search on the Internet, you will find lots of search spirits to give more information about movies. The increased data if not managed perfectly, it loose the performance of the algorithm, no matter your algorithm is strongly efficient. They can often be beaten simply by having more or exploded data [7].

So what is data science? We take number of decisions in day-to-day life, for example, buying furniture for setting new office. So we start scrutinizing possibilities with different options for searching materials through websites or other survey. The activity is to search for a proper and reliable website for furniture selection from several websites. So, which website to use is the first decision you need to take. We may surf multiple websites, and we can't discard the websites that don't sell furniture and stick to those websites that sell furniture. Now within that, we try to find out what are the ratings of the selected website. If the rating is more, that means they are reliable and the quality is probably good, and only then you decide to buy from that website as shown in Figure 1.6 [1].

While surfing website pages for choosing furniture as per requirements, feedback ratings, discounts, promotions, etc. get checked. Some of them will probably not be providing any discount, and some websites will probably be providing discounts. We may go ahead and select the furniture and purchase it [3]. Another example can be booking a CAB. For travelling from location A to location B, need to find best route and optimum travel time it will take. There could be several factors while choosing routes, such as traffic, road conditions, and weather.

Another example is TV shows. Netflix and maybe other TV channels have to perform an analysis to find out what kinds of shows people are watching, what kinds of shows people like, and so on. The data is analyzed and then this information is sent to advertisers because their main source of revenue is advertising. So this is again a major application of data science as shown in Figure 1.7.

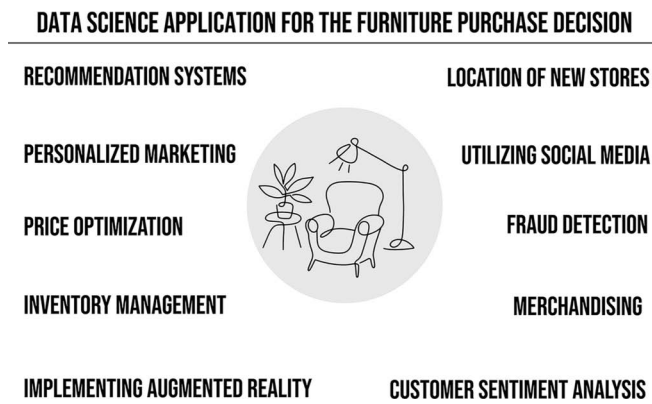


FIGURE 1.6
The furniture purchase decision using data science.

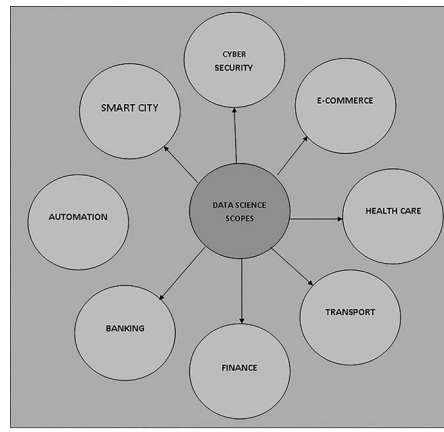


FIGURE 1.7
Scopes of data science.

1.3 Data Science Process

There are various steps in data science. The first step is asking the right question and exploring the data. You must know exactly the problem that needs to be solved. The problem can be understood by asking the right questions. As per the available data for input, the data may be incomplete for which, exploratory analysis can be done. Or data cleansing can be done on raw data for accurate data input which is a part of exploratory analysis. And then, you need to do modeling as shown in Figure 1.8.

Let us say that if you have to perform machine learning, you need to decide which algorithm to use, and through which model, and then you need to train the model accordingly. This is the modeling process, which then runs your data through the model. Then through this process, you come out to the final

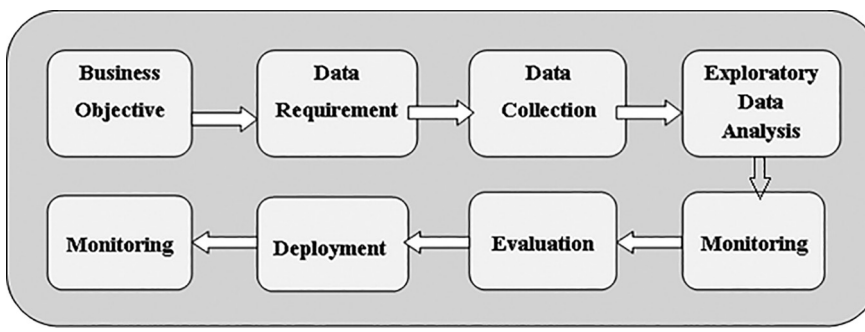


FIGURE 1.8
Data science process.

results of this exercise which includes visualizing the results and preparing a way to communicate the results to the people concerned. So it could be in the form of PowerPoint slides, or it could be in the form of dashboard for proper visualization, so appropriate communication for easily understanding can be possible for intended users.

1.4 Business Intelligence and Data Science

Business intelligence is one of the initial phases where people want to make some sense out of data. Earlier the manufacturing and selling was managed through enterprise resource planning ERP system or customer relationship management (CRM) systems. But later people started taking interest in understanding data for information and that's how business intelligence started affecting.

In view of data source, we can compare criteria concerns to methods, skills and focus of the data. For creating business intelligence through data science, structured data were used the ERP or CRM system works on relational database system, RDBMS such as Oracle, SQL and MySQL which use structured data inputs. The ERP or CRM system works on relational database system, RDBMS such as Oracle, SQL and MySQL which use structured data inputs.

All these data are structured in good form of a database table, rows and columns, and they were all brought into a centralized place because it remembers these were still different applications, so they were working on a different database in silos. So, if you want to get a combined view, you need to create what is known as a data warehouse and bring all the data together and then look at it in a uniform way for the required application. The business intelligence uses structured data that reports on dashboards. Data science in addition to structured data also uses a lot of unstructured data, for example, Web or comments. If we are talking about customer feedback, it may be structured if gathered in formatted rubrics or may be unstructured if scripted for performance analysis. That is, data sources may be in different formats.

Business intelligence is analytical in the sense that you have some data, and you are just trying to present the truth out of it. In the case of data science, we go beyond that. Business intelligence uses many statistics such as correlations and regressions to predict what will be the sales maybe in the future. In the case of data science, skills are many more as compared to business intelligence.

The focus of business intelligence is pretty much on historical data. In data science, you take historical data, but you also combine it with some other required information and you also try to predict the future.

1.5 Prerequisites for a Data Scientist

Now moving on to the life of a data scientist, what does a data scientist do during his work? Typically a data scientist is given a problem, i.e., business problem, and he needs to solve it. To do that, he asks the question as to what is the problem that needs to be solved, which is the first thing he gets from a problem. The next thing is to gather the data that is required to solve the problem, so he searches for data from anywhere. In the enterprise, very often the data is not provided in the structured format that he would like to have.

The first step is to get possible raw data in any format. The data may be enterprise data or public data. The raw data collected and converted into required format for analysis. The cleaned and formatted data used as an input to statistical model through machine learning mechanisms for communicating analytics results to stakeholders. This is the very high-level view of a day in the life of a data scientist. The role of data scientist is to gather data, process data in required format, manipulates as per requirements and feed for analysis. The analysis helps to build mathematical models or machine learning models which are used to present system behavior to concern stakeholders. There are three essential things required to be a data scientist. The first one is *curiosity*. You should be able to ask the right questions: The first step in data science is asking the question what is the problem you are trying to solve; if you ask the right question, then you will get the right answer. This is very crucial step, since it directly impact on project success. Asking incorrect questions will be lead to incorrect response and may impact on correct analysis. You must ask the right question. A second essential thing that a data scientist should have is the *common sense*. The data scientist needs to be creative in solving problems at hand. In many cases, you may not have sufficient data to conclude; hence, you need to fulfill this gap intelligently, finally, a data scientist should have *communication skills*. After doing all the analysis, if you are unable to communicate the results in the right way, the whole exercise will fail. Communication is the key trait for data scientists. You may be technically brilliant, but if you are unable to communicate the result in the proper way, it will not help. So, these are the three main traits, i.e., curiosity, common sense, and communication skills, as shown in Figure 1.9.

1.6 Components of Data Science

The different components for data science are as follows:

1. **Machine Learning** – Machine learning is the backbone of data science. Data science involves quite a bit of learning the basics of statistics to design a learning algorithm.

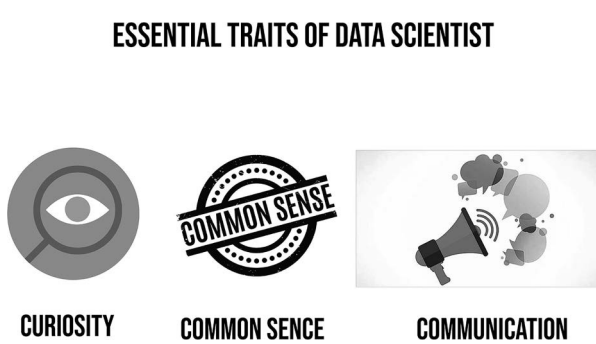


FIGURE 1.9
Essential traits of a data scientist.

2. **Modeling** – Modeling is also a part of machine learning in a way, but you need to be good at identifying what are the algorithms that are more suitable to solve the given problems, which model can be used, and how to train this model.
3. **Statistics and Probabilities** – The statistics and probabilities are like the core foundation of data science. The data scientist must be prominent in statistics and probability theories, so that he can formulate the problems for processing the required results. For the data science projects, basic programming skills and some fundamental knowledge of databases is required. The most common programming languages are Python, R, MATLAB, and Octave. In particular, Python is becoming a very popular programming language in data science because of its ease of learning and the multiple libraries that it supports. Python is the most popular language in data science. To begin for data science career, can start python programming since its simplest language and easy to understand for beginners. Then fundamentals of database is another important aspect to learn for handling databases and extract required information from raw data. So, these are some of the key components of data science.

1.7 Tools and Skills Needed

The tools required for data science algorithm executions are Python or R in addition to some other programming languages. The good knowledge or understanding of statistics and the tools that are used in data analysis helps to execute good applications in this domain. Statistical Analysis System (**SAS**) has been one of the most popular tools for a very long time; however, it is a proprietary software package, whereas Python and R are mostly open source. Other tools such as Jupyter and RStudio are development tools.

Jupyter Notebook is an interactive development environment. The RStudio initially was average due to its interactive facility for performing data analysis or implementing machine learning activities. The new edition of RStudio facilities interactive user interface and become popular.

There are tools such as MATLAB, and of course, some people do with Excel as well.

As far as data is concerned, some of the skills that are required are ETL, i.e., Extract, Transform, and Load. So if you have data in a database like your ERP system and you need to extract that, then you have some processing and then you load it into your warehouse so that all the data from various sources will look like a uniform structure. Then, you need some skills such as querying the data and writing SQL queries. HADOOP is another important skill especially if you are handling large amounts of data. One of the specialties of HADOOP is that it can be used for handling unstructured data as well. So it can be used for large amounts of structured and unstructured data. The **SPARK** is an excellent computing engine for performing data analysis on machine learning in a distributed model. If you have a large amount of data, the combination of SPARK and HADOOP can be extremely powerful. So, you can store your data in HADOOP HDFS and use SPARK as a computation engine. It works in a distributed mode similar to HADOOP-like clusters. These are excellent skills for data warehousing, and some standard tools are available, such as Xplenty, Amazon RedShift, Teradata, Oracle 12c, Informatica, IBM Infosphere, Cloudera, etc. If you want to implement some applications on the cloud, AWS Redshift is again good for data visualization.

R provides very good visualization capabilities especially during development. Python libraries provides powerful visualization capabilities for skill perspectives. The TABLEAU is an another proprietary visualization tool that has excellent capabilities for information presentations. There are tools such as **COGNOS** analytics, which is an IBM product that provides very good visualization capabilities as well.

In the machine learning part, the skills required are Python, which is more for the programming part; then some mathematical skills such as algebra, especially linear algebra; and then statistics, especially calculus. The tools that are used for machine learning are SPARK MLlib and Apache Mahout, and if you want to use the cloud, you can use Microsoft Azure ML Studio as well. So, these are by no means an exhaustive list. There are many tools available apart from these, and also probably few more skills are also required as shown in Figure 1.10.

1.8 Summary

This chapter focuses on the need of data science and its impact on daily life. The concept of data science is elaborated in great detail with its applications in autonomous cars, airline industries, logistics, digital marketing, and other

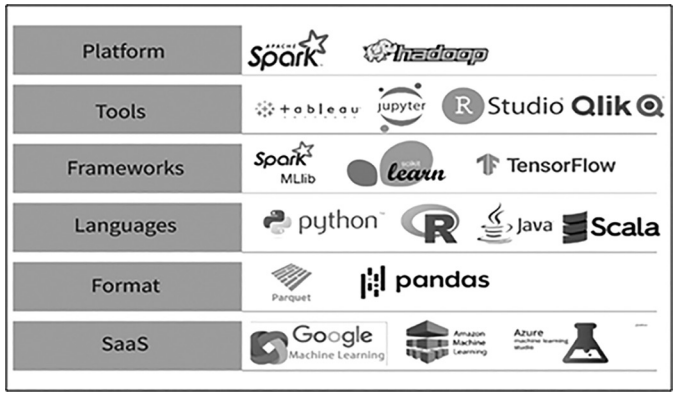


FIGURE 1.10
Tools and skills required in data science.

possible data science domains. The data science process is defined precisely with an illustration of the role of data science in business intelligence. The roles and responsibilities of data scientists, components of data science, and the tools and skills needed to execute data science-based applications are explored.

Exercise

1. What is data science? Why is data science required?
2. How data science is used in airline systems to enhance business and management?
3. What is an autonomous car? What is the role of data science in autonomous cars?
4. Explain the different domains of data science.
5. How does data science help in decision making?
6. Define data science process.
7. What is the difference between business intelligence and data science?
8. What is the role of a data scientist? What are the prerequisites to become a data scientist?
9. Explain different components of data science?
10. List and explain the various tools and skills required for data science.

References

1. http://houseofbots.com/news-detail/3195--explore-more-about-data-science-and-its-significant-importance
2. http://bigdguru.com/13-amazing-applications-uses-of-data-science-today/
3. htps://www.edureka.co/blog/data-science-applications/
4. http://www.elogicsquare.com/top-7-data-science-use-cases-in-healthcare/?sg_popup_id=8
5. https://medium.com/activewizards-machine-learning-company/top-7-data-science-use-cases-in-healthcare-cddfa82fd9e3
6. https://www.simplilearn.com/tutorials/data-science-tutorial/what-is-data-science
7. Jonathan Goldman. Founding advisor – Change research | LinkedIn stories.

2

Statistics and Probability

Statistics and probability are essential as these branches of science lead the basic foundation of all data science domains, i.e., artificial intelligence, machine learning algorithms, and deep learning. Statistics and probability are foundations to execute the concepts of trending technologies. Mathematics is embedded in every aspect of our lives, i.e., from shapes, patterns, and colors to the count of petals in flowers [1].

2.1 Data Types

Data is a set of numbers or some sort of documents that are stored on a computer. Data founds everything if you look around. Each click on your computer or mobile systems generates data. The data is so important that the generated data provides size for analysis and support to make better decisions in business. The formal definition of data is "facts and statistics connected for reference or analyses". This is the definition of data in terms of statistics and probability so as we know, data can be collected, measured, and analyzed and can be visualized by using statistical models and graphs [1].

Data is divided into two major subcategories, that is, qualitative data and quantitative data. Under qualitative data, we have nominal and ordinal data, and under quantitative data, we have discrete and continuous data.

Qualitative data: This type of data can be observed subjectively and deals with characteristics and descriptors that cannot be easily measured. It is further divided into nominal and ordinal data. *The nominal data* is any sort of data that doesn't have any order or ranking. An example of nominal data is gender. Now, there is no ranking in gender, and there is only male, female, or other. There is no 1, 2, 3, 4, or any sort of ordering in gender. The race is another sample of nominal data. The *ordinal data* is an ordered series of information. Let's say you went to a restaurant, and your information is stored in the form of a customer ID, which means you are represented by the customer ID. Now, you would have rated their services as good or average, and that's how the ordinal data is; similarly, there have a record of other customers who visit the restaurant along with their ratings. So, any data which has some sort of sequence or some sort of order to rate is known as ordinal data.

DOI: 10.1201/9780429443237-3

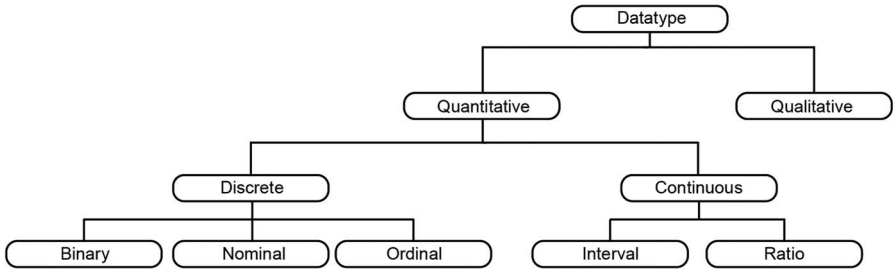


FIGURE 2.1
Types of data.

Quantitative data: Quantitative data concerns measurable things with numbers. We can understand that by the word quantitative itself. Quantitative is a quantity that is given numbers with anything that you can measure objectively [1]. The quantitative data can further be divided into discrete and continuous data. *Discrete data* is also known as unqualified data, and it can hold a limited number of possible values. The student strength in a class is a limited number, which means you can't have an unlimited number of students in a class. Let's say in a fifth grader 100 students in your class so this is not an unlimited number but a definite limited number of students in a class. So, this is the discrete data. Another type of data is continuous data. *The continuous data* concerns an unlimited number of possible values, e.g., the weight of a person. For example, the weight can be 50 kg, or 50.1 kg, or 50.001 kg or 50.0001 or 50.023 kg, and so on. There are an unlimited number of possible values, so this is an example of continuous data. The types of data are summarized in Figure 2.1.

2.2 Variable Types

There are also different types of variables, i.e., discrete variables and continuous valuables. *The discrete variable* is also known as a categorical variable which can hold values of different categories. Let's say that you have a variable called a message. There are two types of values that this variable can hold. Let's say that your message can be either a spam message or a non-spam message. While calling variables, can be distinguished as categorical variable, since it holds value that represents different categories of data. The continuous variable is a variable that can store an unlimited number of values; for example, the weight of a person can be continuous and is variable, and also it can store an unlimited number of possible values, which is the reason we call it a continuous variable. So, the variable is anything that can store. If you associate any sort of data with the variable, then it will become

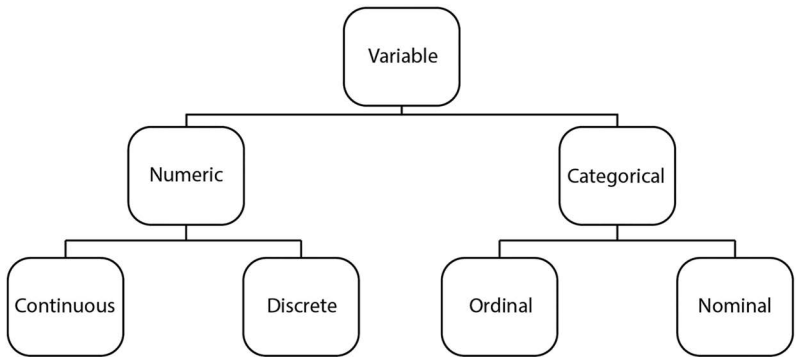


FIGURE 2.2
Types of variables.

either a discrete variable or continuous variable that is also a dependent and independent type of variable. The other types of variables are the independent variables and dependent variables. The dependent variable is any variable whose value depends on any other independent variable. The types of variables are summarized in Figure 2.2.

2.3 Statistics

The formal definition of statistics is as follows: "Statistics is an area of Applied Mathematics which is concerned with data collection, analysis, interpretation, and presentation" [1]. Other than these parameters, the statistics are just not all about analysis but also have other parts to do, including data collection, data interpretation, presentation, and visualization. The statistical methods are used to visualize data and collect data for interpretation of information. The methods to solve complex problems using data are statistical methods.

For example, to treat cancer patients the organization has created a new drug. What will be the process to carry out the test to satisfy the effectiveness of drug? This can be solved using a statistical approach that you have to create a test that can confirm the effectiveness of the drug. This is a common problem that can be solved using statistics. Another example is as follows: The latest sales data is made available to you and your team leader has to work on the strategy to improve business of the company and needs to identify the places where scopes of improvement are better and prepare a report for management. This problem involves a lot of data analysis. You have to look at the different variables that are causing your business to go down, or you have to look at the variables that are increasing the performance of your models and

grow your business. So, this involves a lot of data analysis, and the basic idea behind data analysis is to use statistical techniques to figure out the relationship between different variables or different components in your business.

The statistics are of two types, descriptive statistics and inferential statistics. The method that is used to describe and understand the features of specific datasets, by giving a summary of data, is called descriptive statistics. It mainly focuses on the characteristics of data and graphical summary of the data. For example, an institute wants to procure and distribute uniforms to the students. It needs to study the average size of the student's uniform in the institute. Here, it needs to take measurements of all the students for the uniform. From all measurement data, we need to conclude the small, average, and large size of the measurement, so that the vendor can supply uniforms of three variable sizes instead of considering each student's uniform separately. This describes the descriptive statistics concept [2, 3].

The method that makes inferences and predictions based on the sample datasets about the population is nothing but inferential statistics. In general words, it generalizes large datasets and applies probability to conclude. So based on the statistical model, it allows you to infer data parameters using sample data. Consider the example of finalizing the measurement of uniform size for the students; in inferential statistics, we can take the sample size of the class, which means a few students from the entire class. Students are already categorized into measurement classes of small, average, and large. In this method, you build the statistical model and expand it for the entire population in the class [1].

The methods that describe the basic features of the data are descriptive statistics. They provide simple summaries about the sample and the measures. Descriptive statistics are typically distinguished from inferential statistics. Descriptive statistics are used to provide basic information about variables in a dataset and highlight potential relationships between variables. Also it can be displayed graphically or pictorially using graphical or pictorial measures [4].

There are four major types of descriptive statistics:

- **Measures of Frequency** – Count, percent, frequency, etc.
- **Measures of Central Tendency** – Mean, median, and mode.
- **Measures of Dispersion or Variation** – Range, variance, standard deviation, etc.
- **Measures of Position** – Percentile ranks and quartile ranks.

The process to deduce the properties using data analysis of an underlying distribution of probability is the statistical inference. The properties of population are inferred by inferential statistics analysis. The inferential statistics describes situation or phenomenon, which may possible to conclude information based on some hypothetical situation. The inferential statistics types are extensively used and relatively easy to interpret. The types of statistics are

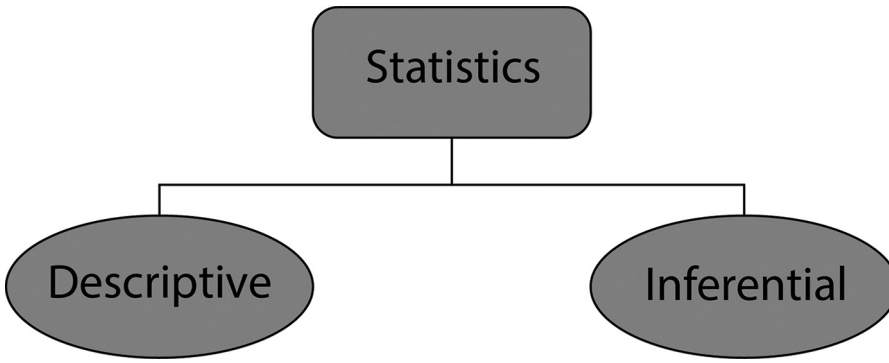


FIGURE 2.3
Types of statistics.

shown in Figure 2.3. The primary difference between descriptive and inferential statistics is that descriptive statistics measure for definitive measurement while inferential statistics note the margin of error of research performed.

Before you dive deep into statistics, you need to understand the basic terminology used in statistics. The two most important terminologies in statistics are population and sample [1]. Throughout this chapter, for any problem that you are discussing with statistics, you will come across the two words population and sample. *The population* is a collection of a set of individual objects or events whose properties are to be analyzed. So basically you can refer to population as a subject that you are trying to analyze. Now the sample is just like the word suggests; i.e., *sample* is a subset of the population. So, you have to be sure that you choose the sample in such a way that it presents the entire population. Sample should not focus on part of the population, but represent the entire.

The information about particular population parameters is based in elegant samples. So choosing elegant sample space is important since it deals with the selection of observation individually within a population.

Sampling is performed to deduce statistical knowledge about a population. Let's understand the different types of statistics of a population such as the mean, the median, the mode, or the standard deviation, or the variance of a population; then, we need to perform sampling. It is not reasonable to study a large population and find out the mean, the median, the variance, and all. Now let's see why sampling is performed. If we need to know what the point of sampling is, we can just study the entire population. Now think of a scenario where you have been asked to perform a survey on eating habit of pizza in the USA. As the population of USA is 42 million, and number is still growing, it is difficult to survey each of these 42 million individuals about their health. It might be possible, but this will take it ever to do. So it's not reasonable to go around each row and ask for what your pizza eating status is. That's why sampling is used as a method in a sample study to draw

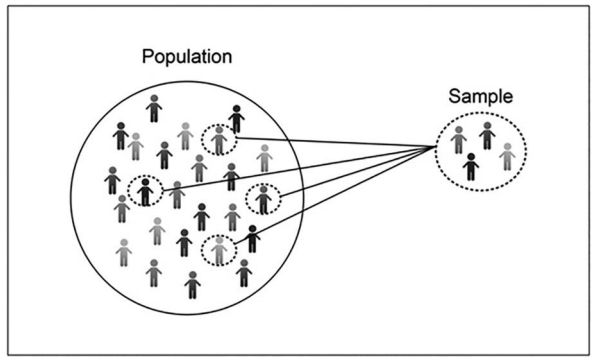


FIGURE 2.4
Population and sample.

inference about the entire population. Sampling method helps to study the entire population instead of considering entire individual population and trying to find out solutions. The statistical analysis represents inference of total population [1]. The population and sample space is graphically demonstrated in Figure 2.4.

2.4 Sampling Techniques and Probability

The sampling techniques are of two types, that is, probability sampling and non-probability sampling. Figure 2.5 shows sampling techniques focusing on probability sampling.

The samples chosen from a large population using theory of probability are known as probability sampling. The types of probability sampling are random sampling, systematic sampling, and stratified sampling.

Random sampling: In the random sampling method, each member of the population has an equal chance of being selected in the sample. Every individual

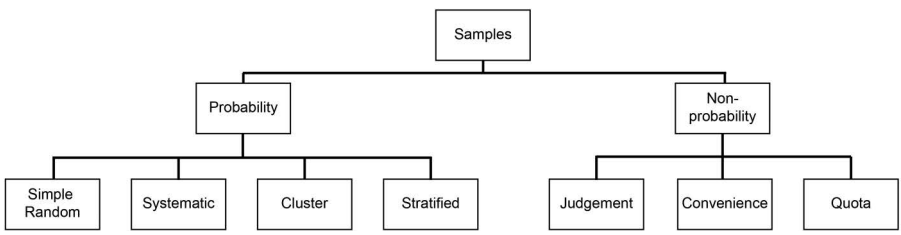


FIGURE 2.5
Sampling techniques.

FIGURE 2.6
Random sampling.

or every object in the population has an equal chance of being a part of the sample. You are randomly going to select any individual or any object, so this way each individual has an equal chance of being selected (Figure 2.6).

Systematic sampling: In systematic sampling, every n^{th} record is chosen from the population to be a part of the sample. Now as shown in Figure 2.7, out of these groups every second person is chosen as a sample [1].

Stratified sampling: In the stratified sampling technique, striatum is used to form samples from a large population. A stratum is a subset of the population that shares at least one common characteristic [1]. For example the population of men and women can be identified by gender and that is striatum. It is a subset of the population that shares at least one common characteristic. The random sample can be possible after creating striatum for choosing final sample as shown in Figure 2.8.

Cluster sampling: In cluster sampling, researchers divide a population into smaller groups known as clusters. They then randomly select among these clusters to form a sample. Cluster sampling is a method of probability sampling

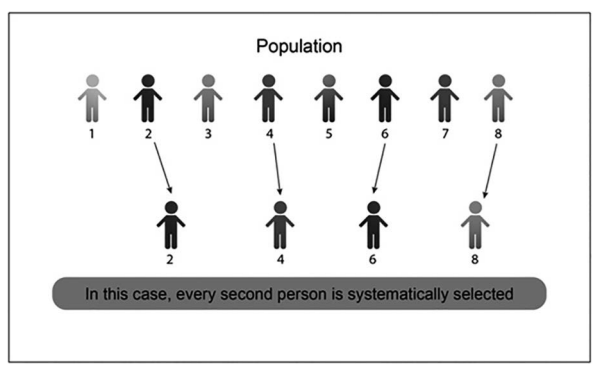


FIGURE 2.7
Systematic sampling.

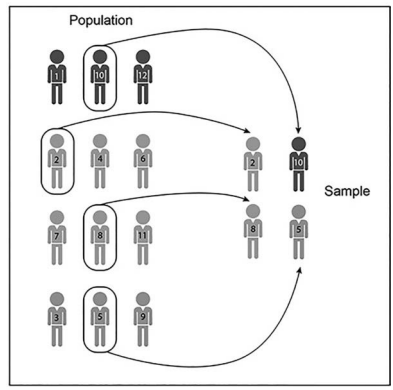


FIGURE 2.8
Stratified sampling.

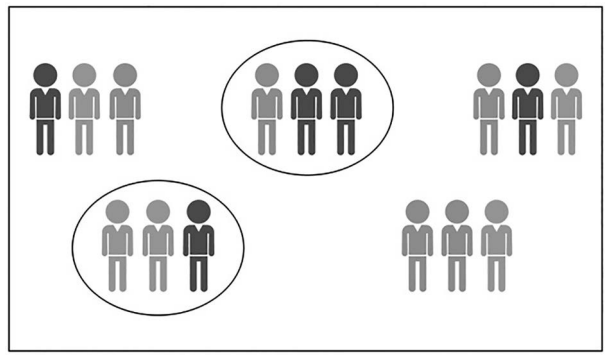


FIGURE 2.9
Cluster sampling.

that is often used to study large populations, particularly those that are widely geographically dispersed. For example, an organization intends to survey to analyze the performance of smartphones across India. They can divide the entire country's population into cities (clusters) and select cities with the highest population and also filter those using mobile devices as shown in Figure 2.9.

2.5 Information Gain and Entropy

Information gain and entropy are involved in many topics of machine learning such as decision tree and random forest. We need to learn the role of information gain and entropy in building a machine learning model. We will focus on the statistical part of information gain and entropy and what exactly they are.

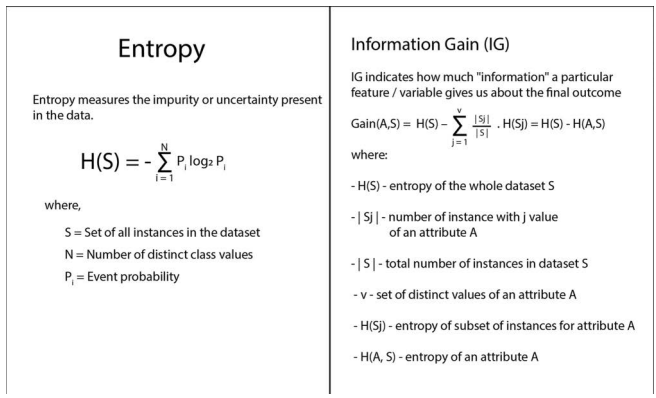


FIGURE 2.10
Information gain and entropy.

Entropy: Entropy is a measure of any sort of uncertainty that is present in data. It can be measured by using the formula shown in Figure 2.10.

Information gain: As the word suggests, information gain indicates how much information a particular feature or a particular variable gives us about final outcomes. It can be measured by using the formula shown in Figure 2.10.

For example, you have to predict whether the match can be played or not, stating the weather conditions [1].

So the predictor variables here are outlook, humidity, and wind, and the target variable is playing. So the target variable is the variable we need to predict. Now the value of the target variable will be decided, whether or not the game can be played. The No prediction means that the weather condition is not good and therefore you cannot play the game, and the Yes prediction means the game can be played, so the play has a value that is "yes" or "no". Now to solve such a problem, we make use of a decision tree. Consider an inverted tree and each branch of the tree denotes some decision. Each branch is known as a branch node, and at each branch, we need to decide in such a manner that you can get an outcome at the end of the branch.

Figure 2.11 shows that of the 14 observations, 9 observations result in a yes, meaning that out of 14 days, the match can be played on only 9 days. So here if you see on days 1, 2, 8, 9, and 11, the outlook has been sunny. So basically we are trying to cluster dataset depending on the outlook. When it is sunny, we have two "Yes's" and three "No's"; when the outlook is overcast, we have all four as Yes's, meaning that on the 4 days when the outlook was overcast, we can play the game. Now when it comes to rainy, we have three "Yes's" and two "No's". The decision can be made as per outlook variable at the root node. So, the root node is the topmost node in a decision tree. Now what we have done here is that we have created a decision tree that starts with the outlook node and then split the decision tree further depending on other parameters such as sunny, overcast, and rainy.

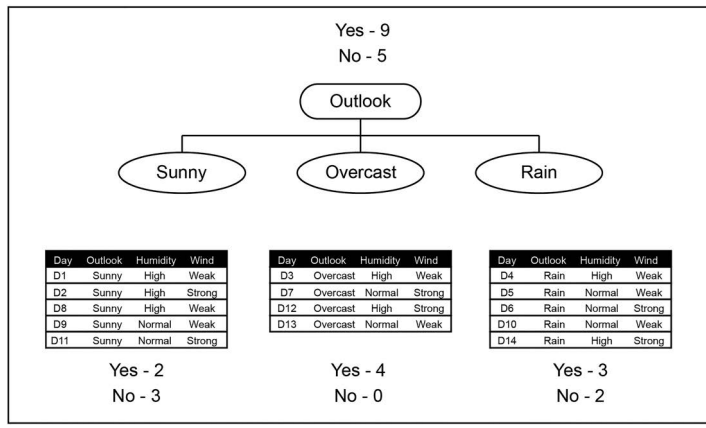


FIGURE 2.11
Decision tree for prediction.

So what you are doing here is you are making the decision tree by choosing the outlook variable at the root node. The root node is the topmost node in a decision tree. The outlook node has three branches coming out of it, that is, sunny, overcast, and rainy. So basically outlook can have three values that are sunny, overcast, and rainy. These three values are assigned to the intermediate branch node and computed for the possibility of play equal to "yes". For sunny and rainy branches if it is mix of yes and no, will give impure output. But when it comes to the overcast variable, it results in a 100% pure subject. This shows that the overcast variable will result in a definite and certain output. This is exactly what the entropy is used to measure. It calculates the impurity or the uncertainty. So the lesser the uncertainty or the entropy of a variable, the most significant is that variable [1]. So when it comes to overcast that has no impurity in a dataset, it is a perfect pure subset. We don't always get lucky and we don't always find variables that will result in subset to measures the entropy. Thus, the lesser the entropy of a particular variable, the most significant that variable will be. So in the decision tree, the root node assigned attribute considered for the precise outcome. This means that the root node should have the most significant variable, which is why we chose outlook [1].

The overcast node is not a variable but the subset of outlook root node. Now the question is how to decide which variable or attribute best splits the data. When it comes to decision trees, information gain, and entropy, it will help you to understand which variable will best split the dataset, or which variable is used to assign to the root node. The variables assigned to root node will split as per dataset with the most significant variables.

As shown in Figure 2.12, we can find information gain and entropy. Of the total 14 instances that we saw, 9 said yes, and 5 said no, which means you cannot play on that particular day. So we can calculate the entropy using the

From the total of 14 instances we have

- 9 instances "yes"
- 5 instances "no"

The entropy is :

$$H(S) = -\sum_{i=1}^{N} P_i \log_2 P_i$$

$$H(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

FIGURE 2.12
Computing information gain and entropy.

formula shown in Figure 2.12. You just substitute the values in the formula. When you substitute the values in the formula, you get a value of 0.940. This is the entropy and this is the uncertainty of the data present in the sample. Now to ensure that we chose the best variable for the root node, let's look at the possible combinations that you can use on the root node [1].

All the possible combinations that you can use at the root node are shown in Figure 2.13. The possible combination can be outlook, windy, humidity, or temperature. These are four variables and you can have any one of these variables as your root node. But how do you select the variable that best fits the root node? Here, we can use information gain and entropy. Thus, the task is to find the information gain for each of these attributes, i.e., for outlook, windy, humidity, and temperature. The variable that results in the highest information gain must be chosen because it give most precise output information.

The information gain for the attribute windy will calculate that attribute first; here, we have six instances of true and eight instances of false as shown

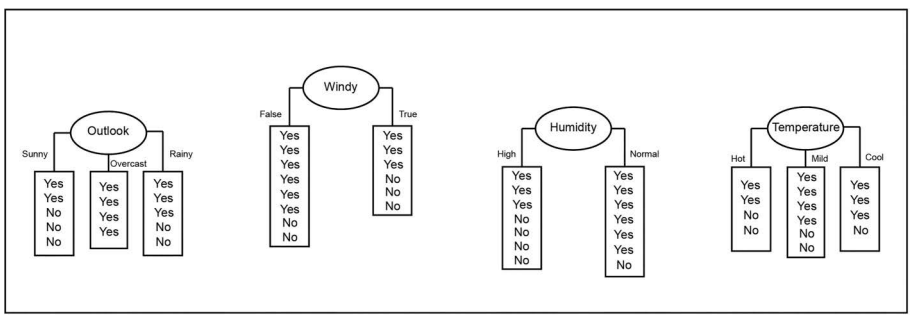


FIGURE 2.13
Selecting the root variable.

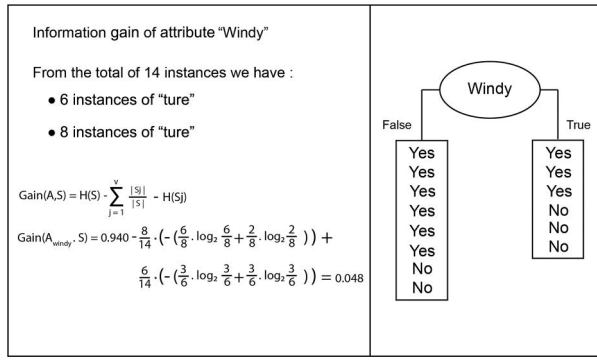


FIGURE 2.14
Information gain for the windy attribute.

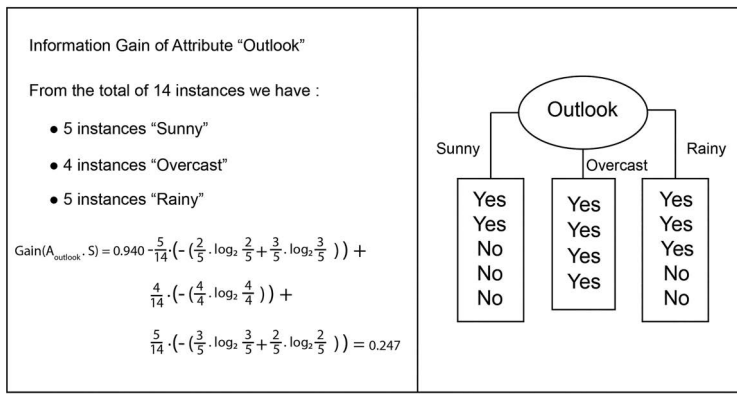


Figure 2.15

Information gain for the outlook attribute.

in in Figure 2.14. When you substitute all the values in the formula, you will get a value of 0.048. The value is less for information gain as the information get from the windy attribute is low.

Now we will calculate the information gain of the attribute outlook. As shown in Figure 2.15, from a total of 14 instances, we have 5 instances of sunny, 4 instances of overcast, and 5 instances of rainy. For sunny, we have three yes's and two no's; for overcast, we have all the four as Yes; and for rainy, we have three yes's and two no's. So when you calculate the information gain of the outlook variable, you will get a value of 0.247. Comparing to this value information gain of the windy attribute is good, i.e. 0.247 for information gain.

Now, let's look at the information gain of the attribute humidity. Here we find seven instances which see "high" and seven instances see "normal". The "high" value in branch node have seven instances and rest of "normal" value. Similarly on the normal branch, we have seven instances that say yes and one

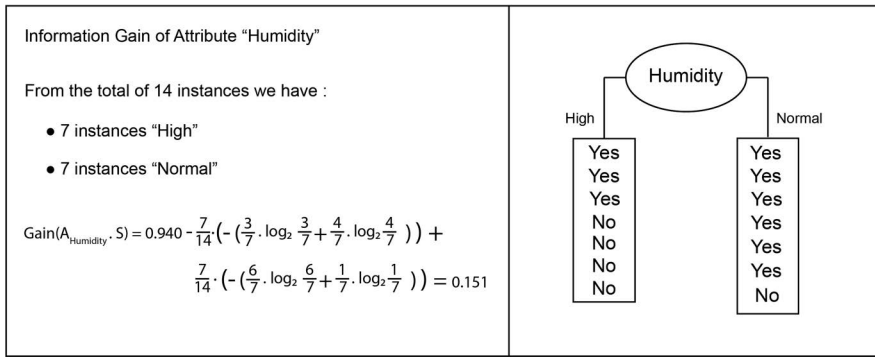


FIGURE 2.16
Information gain for the humidity attribute.

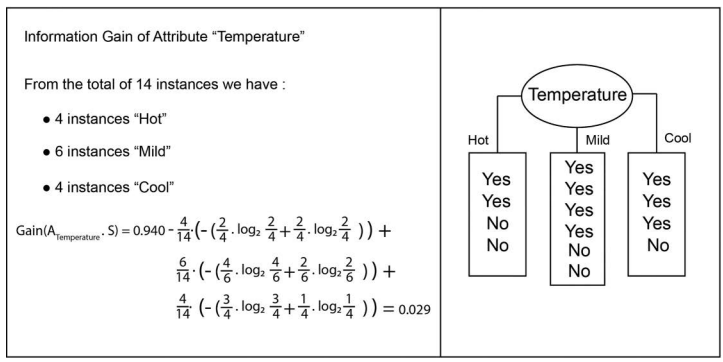


FIGURE 2.17
Information gain for the temperature attribute.

instance that says no, as shown in Figure 2.16. When calculating the information gain for the humidity variable, you are going to get a value of 0.151. This is also a very decent value, but when compared to the information gain of the attribute outlook, it is less._

Now, look at the information gain of attributes as temperature. The temperature can hold three basic attributes: hot, mild, and cold. Under hot, we have two instances of yes and two instances of no; under mild, we have four instances of yes and two instances of no; and under cold, we have three instances of yes and one instance of no. If we calculate the information gain for this attribute, we will get a value of 0.029, which is less as shown in Figure 2.17.

To summarize, if you look at the information gain for each of these variables, you will see that for outlook, we have the maximum gain. We have 0.247, which is the highest information gain value, and you must always choose the variable with the highest information gain to split the data at the root node that's why we assign the outlook variable at the root node as shown in Figure 2.18.

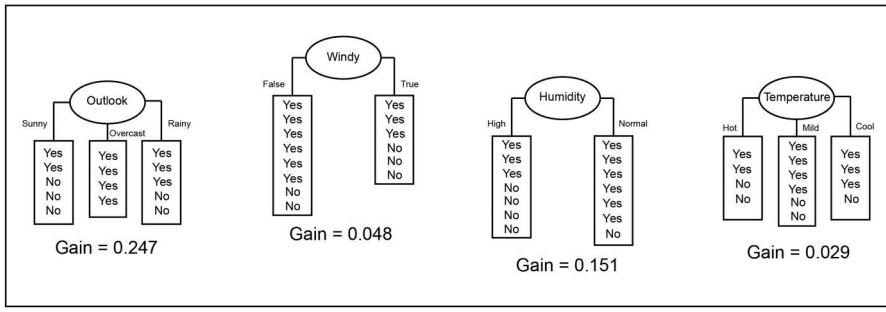


FIGURE 2.18
Information gain for all attributes.

The confusion matrix is a matrix that is often used to describe the performance of a model. This is used as classifier or classification models which calculate the accuracy and the performance of the classifier by comparing actual results and protected results.

$$\frac{\text{True positives} + \text{True negatives}}{\text{True positives} + \text{True negatives} + \text{False positives} + \text{False negatives}}$$

Example

Consider that you give data about 165 patients, out of which 105 patients have a disease and the remaining 60 patients don't have a disease [1]. We can build a classifier that can calculate the performance of the model. There are two possible predicted classes. The classifier made a total of 165 predictions. Out of these 165 observations, the classifier predicts the output for new patients' details. Now out of these 165 cases, the classifier predicted "yes" 110 times and "no" 55 times. So "yes" basically stands for that the person has a disease and "no" stands for that the person doesn't have a disease. It's clear that as per prediction, 110 patients have disease and 55 patients do not have any disease, however in reality only 105 patients in the sample have the disease and 60 patients do not have the disease. We will see how to calculate the accuracy of the model. We build the confusion matrix as shown in Figure 2.19.

Figure 2.19a and b denotes the total number of observations that you have, which is 165 in our case. Actual denotes the actual values in the dataset, and predicted denotes the predicted values by the classifier. Here, the actual value is "no" and predicted values is "no", so the classifier was correctly able to classify 50 cases as "no". Similarly, it is wrongly predicted that five patients do not have the disease while they actually did have the disease, and it is correctly predicted that 100 patients have the disease. The actual value "no" and predicted value "no" means that it is corrected predicted as 50 value". Value "no" and predicted "yes" means it is wrongly predicted "yes" for the value you suppose to predict "no". Let's discuss what true positive and true negative is. True positives are the cases in which we predicted a yes and the patients do not have

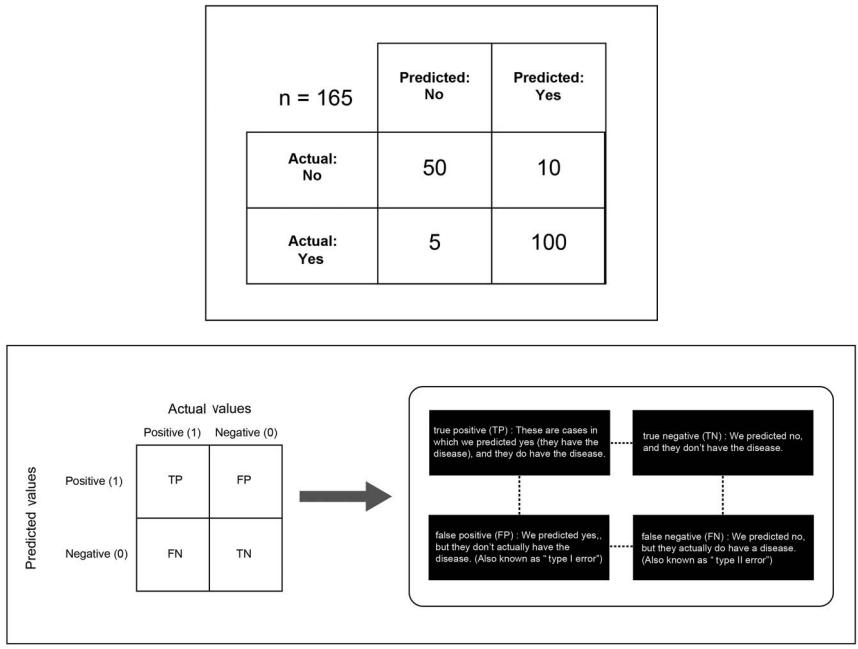


FIGURE 2.19
(a) Confusion matrix. (b) Example of confusion matrix with TP, TN, FP, and FN.

the disease. If the predicted value is "yes" and they do not have any disease, it can be confirmed as true positives. The true negative is when the predicted value is "no" and they do not have any disease. False positive means predicted value is "yes" and no actual disease, known as type 1 error, and false negative means predicted value is "no" and have disease". This classification is call confusion matrix.

2.6 Probability Theory

So before we understand what exactly probability is, let us be clear about a very common misconception about the relationship between statistics and probability. Probability is a mathematical method used for statistical analysis; therefore, we can say that probability and statistics are interconnected branches of mathematics that deal with analyzing the relative frequency of events. The probability makes use of statistics and statistics makes use of probability [1].

Probability is the measure of how likely an event will occur. To be more precise, it is the ratio of the desired outcome to the total outcomes [1]. Now the probability of all outcomes always has a sum up to one. The probability will always have a sum up to one, and it cannot go decimals such as 0.52 or 0.55 or in the form

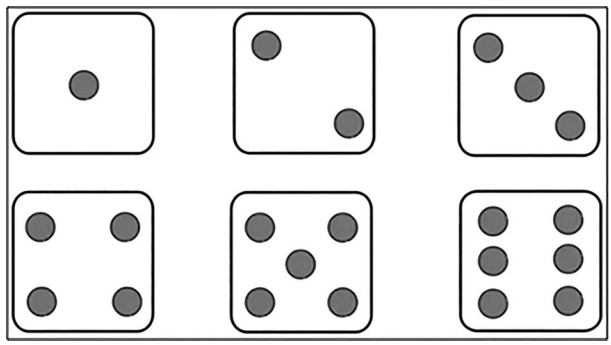


FIGURE 2.20
Sample space for rolling die.

of 0.50, 0.7, or 0.9, which is valuable as it always stays between zero and one. Let's discuss example of probability of 6 face rolling die. As shown in Figure 2.20, while rolling die, we will get "6" possible outcomes, i.e. 1, 2, 3, 4, 5, 6 face of a face dice. Now each face have probability of getting 1/6. Similarly, the probability of getting 5 is 1/6. So the entire sum of the probability is 1. Probability is a very simple concept that we have learned in high school and beyond.

Now let's discuss terminology in probability theory. There are three terminologies that you often come across when we talk about probability.

A random experiment is an experiment or a process for which the outcomes cannot be predicted with certainty. We are going to use probability to predict the outcome with some sort of certainty. *A sample space* is the entire possible set of outcomes of a random experiment, and an event is one or more outcomes of an experiment. If you consider the example of rolling a dice, finding the probability of getting a 2 is the random experiment. The sample space is your entire possibility. That is, 1, 2, 3, 4, 5, and 6 faces are there, and out of those, you need to find the probability of getting a 2. All the possible outcomes will represent your sample space. So 1–6 are all possible outcomes, and this presents your sample space. *The event* is one or more outcomes of an experiment. In this case, the event is to get a 2 when we roll a dice. The probability of getting the event 2 in 6 face die means, getting 2 is the event in sample space is random experiment. There are different types of events, and the two types of events that we should know are disjoint and non-disjoint events. *Disjoint* events are events that do not have any common outcome; for example, if you draw a single card from a deck of cards, it cannot be exactly "king" and a "queen", but it will be either "king" or "queen".

Non-disjoint events are events that have common outcomes; for example, a student who can get 100 marks in statistics and 100 marks in probability, and also the outcome of a ball delivered that can be a no-ball or can be a 6. This is what non-disjoint events are, and these are very simple to understand. The summary information about disjoint and non-disjoint events is shown in Figure 2.21.

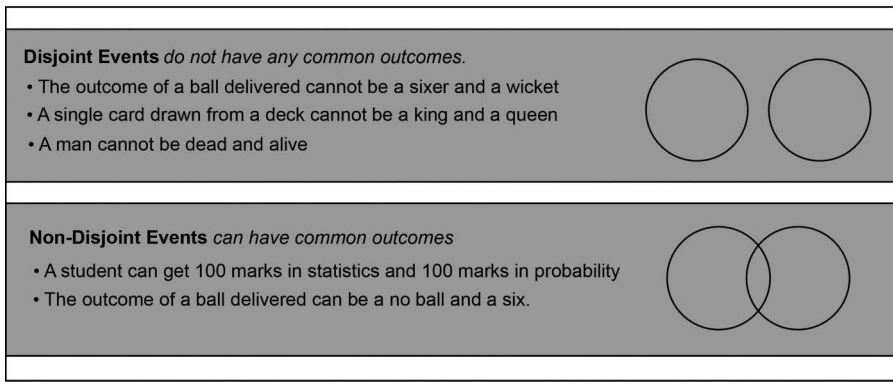


FIGURE 2.21
Disjoint and non-disjoint events.

2.7 Probability Types

Most of the problems can be solved by understanding which type of probability we should use to solve the problem. There are three important types of probability; they are marginal, joint, and conditional probability.

The probability of an event occurring unconditional on any other event is known as *marginal probability* or *unconditional probability*. Let's consider that you want to find the probability that the card drawn is a heart. Then, the probability is 13/52, since there are 52 cards in a deck and there are 13 hearts in a deck of cards, as shown in Figure 2.22.

Now let's understand joint probability. *The joint probability* is a measure of two events happening at the same time [1]. Let's say that the two events are A and B. Then, the probability of events A and B occurring is the intersection of

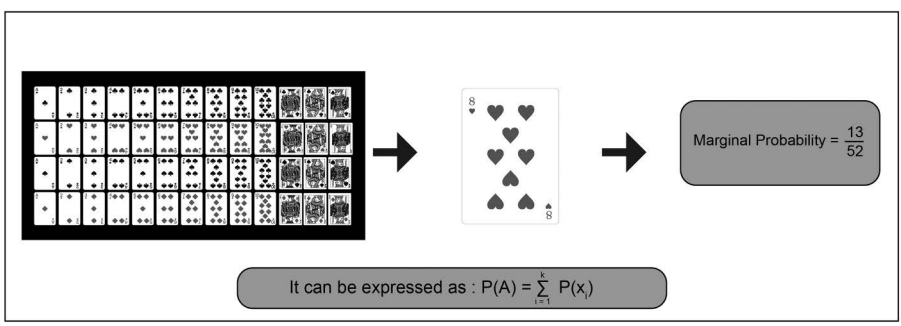


FIGURE 2.22
Examples of marginal probability.

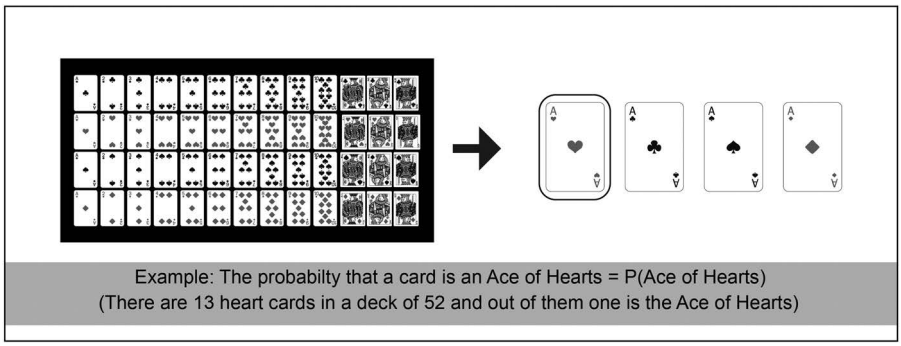


FIGURE 2.23
Examples of joint probability.

A and B. For example, if you want to find the probability that a card is 4 and red, that would be the joint probability. Because you are finding a card that is 4 and is red. This will be 2/52 because we have one 2 in heart and one 2 in diamonds. So both these are red therefore the probability is 2 by 52 and if you further down, it will be 1 by 26, as shown in Figure 2.23.

Let's look at what exactly conditional probability is. If the probability of an event or an outcome is based on the occurrence of a previous event or an outcome, then it is called *conditional probability*. The conditional probability of an event is the probability that the event will occur given that an event A has already occurred. If A and B are dependent events, then the expression for conditional probability is given by $P(B|A) = P(A \text{ and } B)/P(A)$. Now, the first term on the left-hand side, which is $P(B|A)$, is the probability of event B occurring given that the event A has already occurred. If A and B are dependent event then the expression as above, But A and B are independent events, then the expression for conditional probability is $P(B|A) = P(B)$, $P(A)$, and $P(B)$ is the probability of A, while B is already occurred as shown in Figure 2.24.

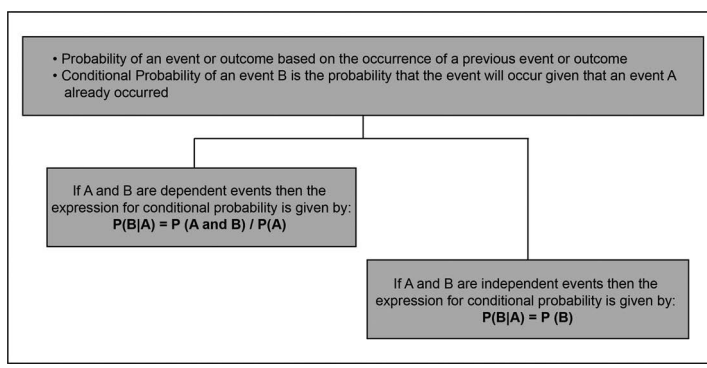


FIGURE 2.24
Conditional probability.

Example

A new project assignment order is received by an IT company and the authority wants to assign the projects according to the salary package.

There are 60 employees without training and 45 employees who are trained with required professional courses. Now the task here is that you have to assign the job according to the salary package.

In total, we have 105 employees, out of which 60 have not completed professional training and 45 have completed professional training. This is a small survey that was conducted. The rating of the salary packages is as follows. As shown in the dataset in Figure 2.26, five employees who are without professional training got a very low salary package. Similarly, there are 30 candidates with professional training who got a good salary package. Here, we are comparing the salary packages of a person depending on the status of professional training completion. The problem statement is to "find the probability of the employee that they have undergone professional training" (Table 2.1).

As per the problem statement, we can recognize that this is a type of marginal probability and organized dataset as shown in Figure 2.28. The probability that an employee has undergone professional training is 45/105 (i.e., 0.42), since 45 is the number of employees with professional training and 105 is the total number of employees [1].

The next problem statement is to "find the probability that an employee has attended professional training and also has a good salary package". Now, this is a joint probability problem. As per the dataset table in Figure 2.28, we can directly find that employees who have got a good salary package along with professional training are 30. Out of the 105 people, 30 people have professional training and a good salary package. Specifically asking for employees with professional training, remember that the question is to find the probability that an employee has attended professional training and also has a good salary package. Here, we need to consider two factors, that is, an employee who attended professional training and who has a good salary package. Here, the number is 30; hence, the answer is 30/105 = 0.28.

TABLE 2.1

Dataset for Employees

Results		Training		Total
		Without Professional Training	With Professional Training	
Salary package obtained by employees	Poor salary	05	00	05
	Below-average salary	10	00	10
	Average salary	40	10	50
	Good salary	05	30	35
	Excellent salary	00	05	05
	Total	60	45	105

Find the probability that an employee has a good salary package given that the employee has not undergone professional training. Now, this is conditional probability because here we are defining a condition that we want to find the probability of an employee who has a good salary package given that the employee has not undergone any training. The condition is that the employee has not undergone any training. As per the dataset, the employees who have not undergone professional training are 60, and out of the 60, five have got a good salary package. Hence, the answer is 5/60, and not 5/105. We have to consider only the people who have not undergone training. Only five people who have not undergone training have got a good salary package. Therefore, you get a probability of around 0.08, which is very low.

2.8 Probability Distribution Functions

Here, we will be discussing the three main probability distribution functions, i.e., density function, normal distribution, and central limit theorem.

Probability density function (PDF) is concerned with the relative likelihood for a continuous random variable to take on a given value [1]. So, the period gives the probability of a variable that lies between the range a and b. We are trying to find the probability of a continuous random variable over a specified range, as shown in Figure 2.25.

Now, the graph shown denotes the PDF of a continuous variable. This graph is also known as the bell curve, because of its shape. The three important properties that we need to know about a PDF are as follows:

Property 1 – The graph of a PDF will be continuous over a range, because you are finding the probability that a continuous variable lies between A and B.

Property 2 – The area bounded by the curve of a density function and the x-axis is equal to 1. The area below the curve is equal to one because it denotes the probability; it has to be between 0 and 1.

Property 3 – The probability that a random variable assumes a value between **a** and **b** is equal to the area under the PDF bounded by a and b. That means the probability value is denoted by the area of the graph, so whatever value that you get is one, which is the probability that a random variable will lie between **a** and **b**. It's the probability of finding the value of a continuous random variable between **a** and **b**.

The normal distribution is a probability distribution that denotes the symmetric property of the mean. In normal distribution, the data near the mean occurs more frequently than the data away from the mean. This means that

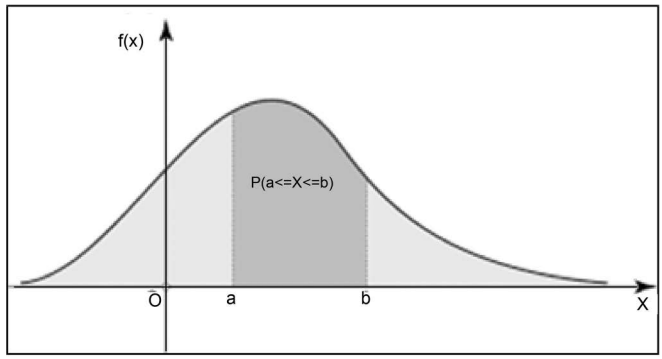


FIGURE 2.25
Probability density function (PDF).

the data around the mean represents the data of the entire dataset; if you just take a sample of data around the mean, it can represent the entire dataset. Similar to the PDF, the normal distribution appears as a bell curve. When it comes to the normal distribution, there are two important factors, the *mean* of the population and the *standard deviation*. The standard deviation finds the height of the graph, while the mean finds the location of the center of the graph. If the standard deviation is large, the curve is going to look as shown in Figure 2.26 that it will be short and wide, and if the standard deviation is small, the standard deviation is tall and narrow [1].

The graph shown in Figure 2.27 clarifies the central "limit theorem".

The central limit theory states that the sampling distribution of the mean of any independent random variables will be normal or nearly normal if the sample size is large in number. In simple term if we had large population and we divide it into many samples, then the mean of all the samples from the population will be almost equal to the mean of the entire population. Meaning the each of the sample is normally distributed. So if you compared

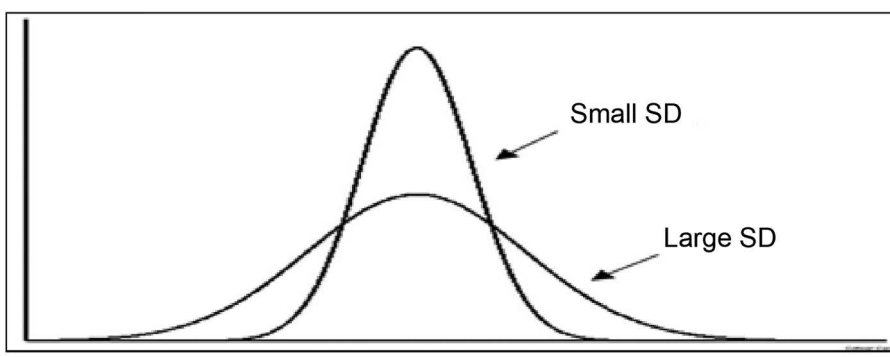


FIGURE 2.26
Standard deviation.

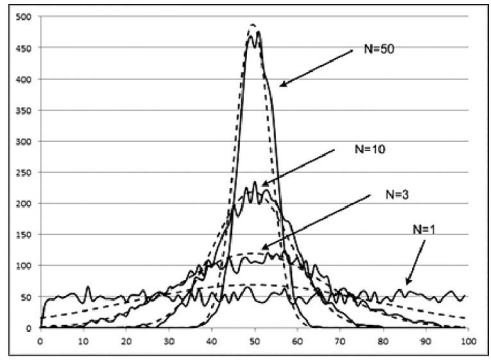


FIGURE 2.27
Sampling distributions and central limit theorem.

the mean of each of the sample, it will be almost be equal to mean of the population. Figure 2.28 represents central limit theorem graph.

We can see each sample for and the mean of each sample is almost along the same line. The accuracy or the resemblance to the normal distribution depends on two main factors: The first is the number of sample points that you consider, and the second is the shape of the underlined population. Now, the shape depends on the standard deviation and the mean of a sample, so the central limit theorem states that each sample will be normally distributed in such a way that the mean of the sample will coincide with the mean of the actual population. That means central limit theorem states true value only for large data set. For large data set, there are more deviations as compared to small data because of scaling factor.

2.9 Bayes' Theorem

The Bayes theorem is a very important concept when it comes to statistics and probability [1]. The naive Bayes algorithm is majorly used. The naive Bayes algorithm is precisely used in a supervised learning classification algorithm used for applications such as Gmails spam filtering. If you open up Gmail, you see that you have a folder called Spam. That is carried out through machine learning, and the algorithm used there is naive Bayes.

The Bayes theorem is used to show the relation between conditional probability and its inverse. It is nothing but the probability of an event occurring based on prior knowledge of conditions that might be related to the same event. Mathematically, the naive Bayes is represented as shown in the following equation:

$$P(A \mid B) = P(B \mid A) * P(A)/P(B).$$

The term on the left-hand side is referred to as the likelihood ratio, which measures the probability of occurrence of event B given event A. The term on the left-hand side is known as the posterior, which means the probability of occurrence of event A given event B. The second term is referred to as the likelihood ratio. This measures the probability of occurrence of event B given that event A occurs. P(A) is also known as the prior that refers to the actual probability distribution of A and P(B) is the given probability of B. This is the Bayes theorem.

To understand practically how the Bayes theorem works, let's look at a small example. We have three baskets, basket A, basket B, and basket C. Basket A contains two blue balls and four red balls, basket B contains eight blue balls and four red balls, and basket C contains one blue ball and three red balls. If we draw one ball from each basket, what will be probability to draw blue from basket A, if we know exactly two blue balls in A?

Let A be the event of picking a blue ball from basket A and X be the event of picking exactly two blue balls. These are the two events for which we need to calculate the probability. These two events are represented by A and X, respectively, so we need to compute the probability of occurrence of event A given X. That means for picking exactly two blue balls, what will be probability of picking a blue ball from basket A. By the definition of conditional probability, the equation is $Pr(A|X) = Pr(A \cap X)/Pr(X)$.

2.10 Inferential Statistics

Now we will discuss inferential statistics, which is the second type of statistics, and we have discussed descriptive statistics earlier. The inferential statistics deals with forming inferences and predictions about a population based on a sample of data taken from the population [1]. So how does one form inferences of predictions from samples? The inference can be drawn from prediction sample using point estimation. Point estimation focuses on the sample data to measure a single value that serves as an approximate value of the best estimate of an unknown population parameter. For example, in order to calculate the mean of a huge population, draw the sample of the population and then find the sample mean. The sample mean is used to estimate the population mean. This is the point estimate [1]. Estimating the value of one of the parameters of the population. This is important in calculating the value of the mean to estimate the value of the mean. This is the point estimation. The two main terms in point estimation are the estimator and the estimate. The estimate is the function of the sample that is used to find out the estimate, e.g., the sample mean. A function that calculates the sample mean is known as an estimator, and to realize the value of the estimator is the estimate. There are four common ways to find the estimates. The first one is the method of moments. Here, what you need to do is form an equation

in the sample dataset and then analyze a similar equation in the population dataset as well, like the population mean, population variance, and so on. In simple terms, you are taking down some known facts about the population and extending those ideas to the sample. Once you do that, we can analyze a sample and estimate more essential or more complex values. Next, we have a maximum likelihood. This method uses a model to estimate a value. The maximum likelihood is based on the probability involved in this method. Next, we have the base estimator. This works by minimizing the error order of the average risk. The best estimator has a lot to do with the Bayes theorem. Finally, we have the Bayes unbiased estimator. In this method, several unbiased estimators can be used to approximate a parameter. These are a couple of methods that are used to find the estimate, but the most well-known method to find the estimate is known as interval estimation. This is one of the most important estimation methods. The confidence interval also comes into the picture. The method of estimation is summarized in Figure 2.28.

Apart from interval estimation, we must know as the margin of error. Now, we will be discuss about interval estimate. The estimate interval is the range of values that can be used to estimate a population parameter, i.e. estimating the value of a parameter that find means of the population. We are going to build a range and your value will lie in that range or in that interval. In this way, your output is going to be more accurate because you have not predicted the point estimation, but instead you have estimated an interval within which your value might occur.

Figure 2.29 clearly shows the difference between the point estimate and interval estimate. The interval estimate is more accurate as not focusing on a particular value or a particular point to predict the probability, instead the value lies between the limits of lower confidence and the upper confidence. This denotes the range or the interval. For example, if we say that it takes 30 minutes to reach the school, this is point estimation, but if we say that it will

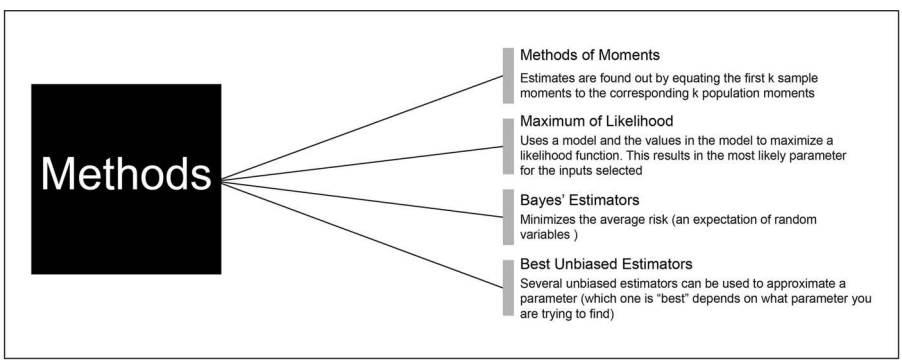


FIGURE 2.28
Methods for estimation.

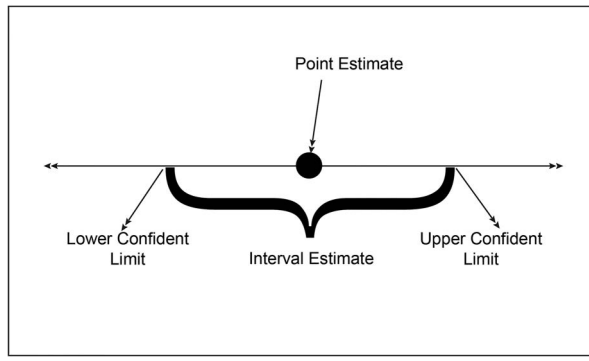


FIGURE 2.29
Point estimations.

take between 50 minutes and an hour to reach the school, this is an example of interval estimation [1].

The estimation gives rise to two important statistical terminologies: One is known as confidence interval and the other is known as the margin of error. The confidence interval is one of the most significant measures used to check how essential a machine learning model is. It is the measure of your confidence that the interval estimated contains the population parameter or the population mean or any of those parameters. The statistician uses the confidence interval to describe the amount of uncertainty related to the sample estimate of a population parameter.

For example, let's say that you survey a group of cat owners to see how many tins of cat food they purchase in 1 year [1]. You test your statistics such as the 99% confidence level, and you get a confidence interval of 100, 200. That means cat owners buy pet food in range of 100 to 200 tins in a year. The confidence level is 99%, i.e. results are very confident. So, your confidence interval here will be 100, 200, and your confidence level will be 99%. This is the difference between confidence interval and confidence level. Within your confidence interval, your value is going to lie, and your confidence level will show you have confidence about your estimation.

Now, let's look at the margin of error. The greatest possible distance between the point estimate and the value of the parameter is known as margin of error for a given level of confidence that it is estimated [1]. We can say that it is a deviation from the actual point estimate. The margin of error can be calculated using the following formula:

$$E = Zc * \sigma / \sqrt{n}$$

where Zc is the critical value of the confidence interval and is multiplied by standard deviation divided by root of the sample size and n is the sample size. Now let's understand how you can estimate the confidence intervals.

The level of confidence is denoted by c. c is the probability that the interval estimate contains a population parameter. Let's say that we are trying to estimate the mean. The level of confidence is the probability that the interval estimate contains a population parameter. This interval between –Zc and Zc are the area of this curve is nothing but the probability that the interval estimate contains a population parameter as shown in Figure 2.30. It contains the value that you are predicting are known as critical values, indicating lower limit and higher limit conference levels presented as Zc score. This score can be calculated by using the standard normal table.

Let's say that the level of confidence is 90%, which means that you are 90% confident that the interval contains the population means. The remaining 10% is equally distributed on the tale regions of 0.05 and 0.05. On either side of zero, you see distributor at the other leftover percentage, now these scores are calculated from the table as mentioned before as 1.645 is calculated from the standard normal table. The estimating level of confidence is demonstrated in Figure 2.31.

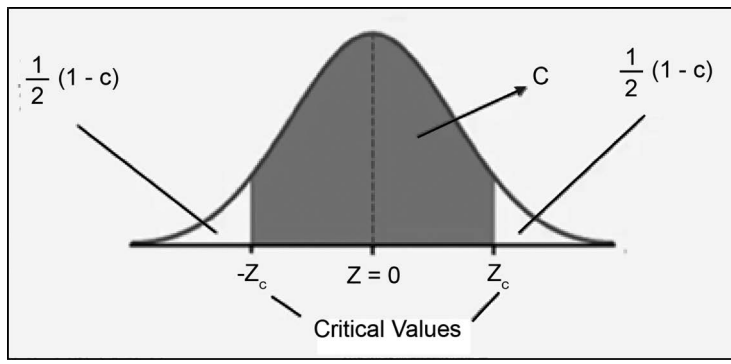


FIGURE 2.30
Estimating confidence.

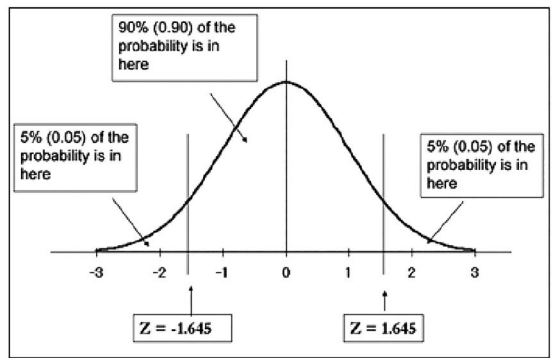


FIGURE 2.31
Estimating level of confidence.

The steps that are involved in constructing a confidence interval are as follows: First, we start by identifying a sample statistic. This is a statistic that we will use to estimate a population parameter. This can be anything like the mean of the sample. The next step is to select a confidence level. The confidence level describes the uncertainty of a sampling method. Then, we find the margin of error, which has been discussed earlier, based on the equation that we have discussed earlier. Finally, we will specify the confidence interval.

2.11 Summary

Probability and statistics are the basis to understand and execute the data science domain applications. This chapter discusses the fundamental terminology and definitions of data types and variables. It also explains the base statistics required in data science with sampling techniques. The information analysis and computation using information and gain theory are elaborated. The probability theory with types and distributions is demonstrated with examples. The Bayes theorem and its use to resolve data science applications are explained. Finally, the inferential statistics concept is introduced to support data science learnings.

Exercise

1. Define the term statistics and probability.
2. Compare and contrast statistics and probability concepts.
3. What are the different data types?
4. Elaborate on the difference between discrete data and continuous data.
5. What are the different types of variables?
6. Differentiate between numerical and categorical variables.
7. What are the different types of statistics?
8. Describe descriptive statistics and inferential statistics.
9. What are the population and sample in probability?
10. What are the different sampling techniques?
11. Define random sampling, systematic sampling, stratified sampling, and cluster sampling.
12. Define the terms information gain and entropy.
13. Demonstrate information gain and entropy using a suitable example.
14. Explain the entropy concept using a suitable example.
15. Explain information gain using a suitable example.

16. Define the probability distribution function.
17. What is a normal distribution?
18. Discuss standard deviation with a suitable example.
19. What is the central limit theorem (CLT)? Demonstrate CLT with a suitable example.
20. Define marginal probability, conditional probability, and joint probability.
21. Define the Bayes theorem.
22. What are the different methods of estimation?
23. Define point estimation with a suitable example.

References

1. https://www.edureka.co/blog/statistics-and-probability/
2. https://www.slideshare.net/SarfrazAhmad2/descriptive-statistics–83286963
3. https://www.indeed.com/career-advice/career-development/descriptive-vs-inferential-statistics
4. https://fr.slideshare.net/EdurekaIN/statistics-and-probability-tutorial-statistics-and-probability-for-data-science-edureka

3

Databases for Data Science

3.1 SQL – Tool for Data Science

Structured Query Language (SQL) is a language designed for managing data in a relational database. Since the 1970s, it is the most common method of accessing data in databases today. SQL has a variety of built-in functions that allow its users to read, write, update, and delete data [1]. It is a popular language for data analysts for a few reasons:

- It can directly access a large amount of data without copying it in other applications.
- It can be used to deal with data of almost any shape and massive size.
- Data analysis done in SQL is easy to audit and replicate as compared to an Excel sheet or CSV file. Excel is great with smaller datasets but not useful for large-sized databases.
- Joining tables, automating, and reusing code are much easier in SQL than in Excel.
- SQL is simple and easy to learn not just for engineers, developers, data analysts, data scientists, but for anyone willing to invest a few days in learning.

Many organizations store a large amount of data within SQL Server by using different application software. SQL Server has fine-grained security features along with more than 36 data types. When these are applied, the data professional can query the data, and only the allowed datasets are returned.

3.1.1 Basic Statistics with SQL

Mean Value
The average of the dataset is calculated by dividing the total sum by the number of values (count) in the dataset. This operation can be performed in SQL by using built-in operation Avg.

DOI: 10.1201/9780429443237-4

```
SELECT  Avg(ColumnName) as MEAN
FROM   TableName
```

Mode Value

The mode is the value that appears most frequently in a series of the given data. SQL does not provide any built-in function for this, so we need to write the following queries to calculate it.

```
SELECT TOP 1 ColumnName
FROM TableName
GROUP BY ColumnName
ORDER BY COUNT(*) DESC
```

Median value

In a sorted list (ascending or descending) of numbers, the median value is the middle number and can be more descriptive of that dataset than the average. For an odd amount of numeric dataset, the median value is the number that is in the middle, with the same amount of numbers below and above [2]. For an even amount of numbers in the list, the middle two numbers are added together, and this sum is divided by two to calculate the median value. SQL does not have a built-in function to calculate the median value of a column. The following SQL code can be used to calculate the median value [3].

Let us assume an Emp table with salary details of ten employees as follows (Table 3.1).

Code to calculate the median of salary column

```
SET @rindex := -1;
 SELECT
   AVG(m.sal)
FROM
   (SELECT @rindex:=@rindex + 1 AS rowindex,
Emp.salary AS sal
```

TABLE 3.1

Emp Table with Ten Employees Salary Details

salary	name
1,000	Amit
2,000	Nisha
3,000	Yogesh
4,000	Puja
9,000	Ram
7,000	Husain
8,000	Risha
5,000	Anil
10,000	Kumar
6,000	Shiv

```
    FROM Emp
    ORDER BY Emp.salary) AS m
WHERE
m.rowindex IN (FLOOR(@rindex / 2) , CEIL(@rindex / 2));
```

The above code will return median value 5,500 for the given Emp table. In the above code, the internal subquery sorts salary and creates @rindex as an incremental index. The outer subquery will fetch the middle items in the array. If the dataset contains an odd number of items, then the SELECT clause of the outer query calculates the average of those two values as the median value.

3.1.2 Data Munging with SQL

Data munging (or wrangling) is the phase of data transformation. It transforms data into various states so that it is simpler to work and understand the data. The transformation may lead to manually convert or merge or update the data manually in a certain format to generate data, which is ready for processing by the data analysis tools. In this process, we actually transform and map data from one format to another format to make it more valuable for a variety of analytics tools.

Some frequently used data munging functions are described below:

UPPER()
The UPPER() string function is used to convert the lower case to the upper case of the input text data.

```
Input:        SELECT UPPER('welcome to Data Science')
Output:       WELCOME TO DATA SCIENCE
```

LOWER()
The **LOWER()** string function is used to convert the upper case to the lower case of the input text data.

```
Input:        SELECT LOWER('WELCOME TO DATA SCIENCE')
Output:       welcome to data science
```

TRIM()
The TRIM() string function removes blanks from the leading and trailing position of the given input data.

```
Input: SELECT TRIM('       WELCOME TO DATA SCIENCE       ')
Output:       'WELCOME TO DATA SCIENCE'
```

LTRIM()
The LTRIM() string function is used to remove the leading blank spaces from a given input string.

```
Input:        SELECT LTRIM('       WELCOME TO DATA SCIENCE')

Output:              'WELCOME TO DATA SCIENCE'
```

RTRIM ()
The RTRIM() string function is used to remove the trailing blank spaces from a given input string.

```
Input:        SELECT RTRIM('WELCOME TO DATA SCIENCE        ')

Output:              'WELCOME TO DATA SCIENCE'
```

RIGHT()
The RIGHT() string function is used to return a given number of characters from the right side of the given input string.

```
Input: SELECT RIGHT('WELCOME TO DATA SCIENCE', 7)

Output: SCIENCE
```

LEFT()
The LEFT() string function is used to return a given number of characters from the left side of the given input string [4].

```
Input:              SELECT LEFT('WELCOME TO DATA SCIENCE', 7)

Output:       WELCOME
```

REPLACE()
The REPLACE() string function replaces all the occurrences of a source substring with a target substring of a given string.

```
Input:              SELECT REPLACE('WELCOME TO DATA SCIENCE',
'DATA', 'INFORMATION')

Output:       WELCOME TO INFORMATION SCIENCE
```

3.1.3 Filtering, Joins, and Aggregation

Joins
In SQL, joins are commands that are used to combine rows from two or more tables. These tables are combined based on a related column between those tables. Inner, left, right, and full are four basic types of SQL joins. Venn diagram is the easiest way to explain the difference between these four types. The result of joining two tables can be represented by the following Venn diagram (Figure 3.1):

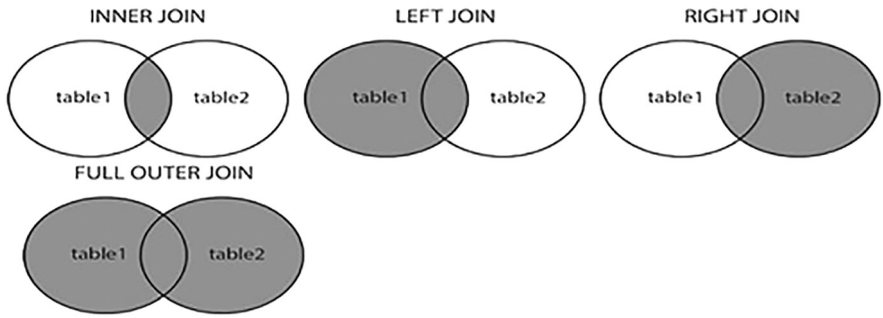


FIGURE 3.1
Venn diagram of join operations.

In our relational database, let us assume two datasets, Table 3.2 (Emp) and Table 3.3 (Dept).

Inner Join
The inner join selects all rows from both tables as long as there is a match between the columns. This join returns those records that have matching values in both tables. So, if you perform an inner join operation between the Emp table and the Dept table, all the records that have matching values in both the tables will be given as output [5].

```
SELECT *
FROM Emp inner join Dept
on Emp.ID = Dept.ID;
```

Result of the above query (Table 3.4):

TABLE 3.2
Emp Table

ID	STATE
10	AB
11	AC
12	AD

TABLE 3.3
Dept Table

ID	BRANCH
11	Computer
12	Civil
13	Mech

TABLE 3.4

Result after Inner Join

ID	STATE	ID	BRANCH
11	AC	11	Computer
12	AD	12	Civil

Left Outer Join

The left outer join (or left join) returns all the rows of the left side table and matching rows in the right side table of the join. The rows for which there is no matching row on the right side, the result will contain *NULL*. From the left table (Emp), the left join keyword returns all records, even if there are no matches in the right tables (Dept).

```
SELECT *
FROM Emp left outer join Dept
on Emp.ID = Dept.ID;
```

Result of the above query (Table 3.5):

Right Outer Join

The right outer join (or right join) returns all the rows of the right side table and matching rows for the left side table of the join. The rows for which there is no matching row on the left side, the result will contain *NULL*.

```
SELECT *
FROM Emp right outer join Dept
on Emp.ID = Dept.ID;
```

Result of the above query (Table 3.6):

TABLE 3.5

Result Table after Left Join

ID	STATE	ID	BRANCH
10	AB	NULL	NULL
11	AC	11	Computer
12	AD	12	Civil

TABLE 3.6

Result Table after Right Outer Join

ID	STATE	ID	BRANCH
11	AC	11	Computer
12	AD	12	Civil
NULL	NULL	13	Mech

Full Outer Join
The full outer join (or full join) returns all those records that have a match in either the left table (Emp) or the right table (Dept) table. The joined table will contain all records from both the tables, and if there is no matching field on either side, then fill in NULLs.

```
SELECT *
FROM Emp full outer join Dept
on Emp.ID = Dept.ID;
```

Result of the above query (Table 3.7):

Aggregation
SQL provides the following built-in functions for aggregating data.

The COUNT() function returns the number of values in the dataset ("salary" is column in table "Emp").

```
SELECT COUNT(salary)
FROM Emp
```

The AVG() function returns the average of a group of selected numeric columns values [6].

```
SELECT AVG(salary)
FROM Emp
```

The SUM() function returns the total sum of a numeric column.

```
SELECT SUM(salary)
FROM Emp
```

The MIN() function returns the minimum value in the selected column.

```
SELECT MIN(salary)
FROM Emp
```

The MAX() function returns the maximum value in the selected column.

```
SELECT MAX(salary)
FROM Emp
```

TABLE 3.7

Result Table after Full Outer Join

ID	STATE	ID	BRANCH
10	AB	NULL	NULL
11	AC	11	Computer
12	AD	12	Civil
NULL	NULL	13	Mech

Filtering:
The data generated from the reports of various application software often results in complex and large datasets. This dataset may consist of redundant records or impartial records. This useless data may confuse the user. Filtering this redundant and useless data can also make the dataset more efficient and useful. Data filtering is one of the major steps involved in data science due to various reasons, and some are listed below:

- During certain situations, we may require a specific part of the actual data for analysis.
- Sometimes, we may require reducing the actual retrieved data by removing redundant records as that may result in wrong analysis.
- Query performance can be greatly enhanced by applying it to refined data. Also, it can reduce strain on application.

Data filtering process consists of different strategies for refining and reducing datasets. To understand data filtering using SQL, we will use the following dataset throughout further queries (Table 3.8).

Syntax to filter data using WHERE

```
SELECT *
FROM tablename;
```

The above query extracts all data from the table. An asterisk (*) in the above simple query indicates that "select all the data" in the table. In the above query, when we add WHERE clause with the condition after WHERE, it filters data in the table and returns only those records that satisfy the condition given after WHERE clause [7].

WHERE clause can be used as follows:

```
SELECT *
FROM tablename
WHERE columnname = expected_value;
```

TABLE 3.8

"workers" Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

Instead of the equal sign (=) operator in the condition statement of **WHERE** clause, we can use the following operators too:

- **>** (greater than),
- **<** (less than),
- **>=** (greater than or equal to),
- **<=** (less than or equal to), and
- **!=** (not equal to).

Suppose we want to extract details of those employees who are working in "HR" department in the above workers table, then we can write the query as follows:

```
select *
from workers
where DEPTNAME='HR';
```

The result of the above query is (Table 3.9):

```
select *
from workers
where SALARY<=47000;
```

The result of the above query is (Table 3.10):

To fetch required data, sometimes, we may require to force two or more conditions. We can use AND, OR operators to achieve this. Only those records that satisfy all the conditions in the query will be retrieved when AND operator is used between two conditions. For example, to find workers

TABLE 3.9

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

TABLE 3.10

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing

in the HR department who have salary more than 47,000, we can write the query as follows (Table 3.11):

```
select *
from workers
where SALARY<=47000 AND DEPTNAME='HR';
```

If OR is used between two conditions, then all records that satisfy either condition will get retrieved along with records that satisfy both conditions. The following query will fetch the details of the workers who are working in the HR department or who have a salary less than 36,000 (Table 3.12).

```
select *
from workers
where SALARY<=36000 OR DEPTNAME='HR';
```

Sometimes, we may want to match a pattern in text data. The LIKE clause can be used to specify a pattern matching condition. Two wildcards, percent sign "%" and underscore "_", are used to specify conditions. The percent sign is used to represent any string of zero or more characters, and underscore represents a single number or character. For example, to retrieve ENAME that ends with character "y" of workers table, we can write the query as follows:

```
select ENAME
from workers
where ENAME like '%y';
```

The above query retrieves the results as follows (Table 3.13):

The following query retrieves records of salary, which has "5" at second position in workers table.

TABLE 3.11

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR

TABLE 3.12

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

TABLE 3.13

Result Table

ENAME
Jerry
Bobby

```
select SALARY
from workers
where SALARY like '_5%';
```

Will return result as follows (Table 3.14):

Sometimes, we may need to filter records based on match of multiple values in a given dataset. The SQL IN operator allows you to test if the given expression matches any value in the list of values. If the records matched with any one of the values in the list, then it is returned as result. For example,

```
select *
from workers
where DEPTNAME IN('Testing', 'Workshop');
```

The above query returns the details of those workers whose DEPTNAME is "Workshop" or "Testing" (Table 3.15).

Sometimes, we may want to exclude some values; we can use NOT keyword in query. The following query returns those workers' details whose DEPTNAME is not "Workshop" or "Testing" (Table 3.16).

```
select *
from workers
where DEPTNAME NOT IN('Testing', 'Workshop');
```

TABLE 3.14

Result Table

SALARY
35,000
45,000
45,000

TABLE 3.15

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop

TABLE 3.16

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

A subquery is a query within another query. A subquery (called a nested query or subselect) is a SELECT query embedded within the WHERE clause of another query. The data returned by the subquery (inner query) is used by the outer query in the same way literal values are used. A subquery is used to return data that will be used in the main or the outer query as a condition that the data must satisfy to be retrieved. Subqueries provide an easy way to handle the queries that depend on the results from another query. For example, in the following query, the inner query retrieves EID of workers who work in "HR" department or get salary >= 40,000. The main query uses the result of the inner query and retrieves workers' details whose EID matches with EIDs returned by the inner query (Table 3.17).

```
select *
from workers
where EID IN (select EID from workers where DEPTNAME='HR' OR
SALARY>=40000);
```

Sometimes, we may need to filter records based on match of a large range of values. You can use the keyword BETWEEN for this purpose. It allows you to specify a start value and an end value of required range. This clause is a shorthand representation for two conditions with >= and <= operators. For example, to retrieve details of those workers having salary >= 30,000 and <= 45,000, we can write the query as follows (Table 3.18).

```
select *
from workers
where SALARY BETWEEN 30000 and 45000;
```

TABLE 3.17

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

TABLE 3.18

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing

To retrieve details of workers whose salary is not in the range of >= 30,000 and <= 45,000, we can write the query using NOT BETWEEN clause as follows (Table 3.19):

```
select *
from workers
where SALARY NOT BETWEEN 30000 and 45000;
```

3.1.4 Window Functions and Ordered Data

SQL window functions calculate their result based on a set of rows rather than on a single row. In SQL window functions, the word "window" refers to a set of rows. It is similar to aggregate functions, but when we use the GROUP BY clause with aggregate functions, it groups the result set based on one or more columns. The aggregate function is performed on the rows as an entire group. SQL window functions generate a result with some attributes of an individual row together with the results of the window function [8].

Window functions can be called with the SELECT statement or the ORDER BY clause, but cannot be called in the WHERE clause. You will notice that all the examples in this article call the window function in the SELECT column list. The window function calculates on a set of rows and returns a value for each row from the given query. In the window function, the term window represents the set of rows on which the function operates. It calculates the returned values based on the values of the rows in a window.

In the window function query syntax, the window is defined using the OVER() clause. In a single query, we can also include multiple window functions.

TABLE 3.19

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

The OVER() clause has the following subclauses:

- PARTITION BY clause to define window partitions to form groups of rows on which window function will be applied.
- ORDER BY clause for logical sorting of rows within a partition.

To demonstrate SQL window function, we will use the following "workers" table (Table 3.20):

RANK () Window Function

The RANK() function returns the position of any row in the specified partition. The OVER and PARTITION BY functions are used to divide the result set into partitions according to specified criteria. Further, ORDER BY clause can be used to sort data in ascending or descending order based on some attribute.

To rank salaries within departments, we can write the query as follows:

```
SELECT
              RANK() OVER (PARTITION BY DEPTNAME ORDER BY
SALARY DESC)
                     AS DEPT_RANK,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

We can see the results below (Table 3.21):

If we exclude PARTITION BY clause from the above query, then we will get the result as follows:

```
SELECT
              RANK() OVER (ORDER BY SALARY DESC)
```

TABLE 3.20

"workers" Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR
Reyon	16	30,000	305	Testing
Bob	22	51,000	301	Workshop

TABLE 3.21

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME
1	HR	308	58,000	Bobby
2	HR	308	45,000	Niya
1	Testing	305	45,000	Alice
2	Testing	305	35,000	Jerry
3	Testing	305	30,000	Reyon
1	Workshop	301	51,000	Bob
2	Workshop	301	50,000	Tom
3	Workshop	301	30,000	John

```
                AS DEPT_RANK,
            DEPTNAME,
            DEPTID,
            SALARY,
            ENAME,
            EID
FROM workers;
```

Then, we will get the following result (Table 3.22):

The RANK() function skips the rank 5 and rank 8 in the above result because two rows share the fourth rank and two records share the seventh rank. The RANK function skips the next k–1 ranks if there is a tie between k previous ranks.

Suppose we want to find out each employee's salary ranks in relation to the top salary of their department. This can be calculated by following math expression:

```
Salary / max_salary_in_dept
```

TABLE 3.22

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME
1	HR	308	58,000	Bobby
2	Workshop	301	51,000	Bob
3	Workshop	301	50,000	Tom
4	HR	308	45,000	Niya
4	Testing	305	45,000	Alice
6	Testing	305	35,000	Jerry
7	Workshop	301	30,000	John
7	Testing	305	30,000	Reyon

TABLE 3.23

Result Table

DEPTNAME	DEPTID	SALARY	ENAME	EID	SMATRIX
HR	308	58,000	Bobby	17	1.0000
HR	308	45,000	Niya	38	0.7759
Testing	305	45,000	Alice	18	1.0000
Testing	305	35,000	Jerry	15	0.7778
Testing	305	30,000	Reyon	16	0.6667
Workshop	301	51,000	Bob	22	1.0000
Workshop	301	50,000	Tom	24	0.9804
Workshop	301	30,000	John	11	0.5882

The next query will show all employees ordered by the above metric; the employees with the lowest salary (relative to their highest departmental salary) will be listed first (Table 3.23):

```
SELECT  DEPTNAME,
             DEPTID,
             SALARY,
             ENAME,
             EID,
            SALARY / MAX(SALARY) OVER (PARTITION BY DEPTNAME
ORDER BY SALARY DESC)
                    AS SMATRIX
FROM workers
ORDER BY DEPTNAME
```

PERCENT_RANK()

In SQL Server, the **PERCENT_RANK()** function calculates the SQL percentile rank of each row. This percentile ranking number may range from zero to one.

For each row, **PERCENT_RANK()** calculates the percentile rank using the following formula:

$$(\text{Rank} - 1)/(\text{total\_number\_rows} - 1)$$

In this formula, rank represents the rank of the row. total_number_rows is the number of rows that are being evaluated. It always returns the rank of the first row as 0.

It divides the rows into partitions by using the **PARTITION BY** clause, and the **ORDER BY** clause logically sorts (in ascending or descending order) rows for each partition. The percentile rank value is calculated for each ordered group independently (Table 3.24) [7].

```
SELECT
             PERCENT_RANK() OVER (PARTITION BY DEPTNAME ORDER
BY SALARY DESC)
```

TABLE 3.24

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME	EID
0	HR	308	58,000	Bobby	17
1	HR	308	45,000	Niya	38
0	Testing	305	45,000	Alice	18
0.5	Testing	305	35,000	Jerry	15
1	Testing	305	30,000	Reyon	16
0	Workshop	301	51,000	Bob	22
0.5	Workshop	301	50,000	Tom	24
1	Workshop	301	30,000	John	11

```
                AS DEPT_RANK,
            DEPTNAME,
            DEPTID,
            SALARY,
            ENAME,
            EID
FROM workers;
SELECT
            PERCENT_RANK() OVER (ORDER BY SALARY DESC)
                AS DEPT_RANK,
            DEPTNAME,
            DEPTID,
            SALARY,
            ENAME,
            EID
FROM workers;
```

The above query gives the following result (Table 3.25):

TABLE 3.25

Result Table

DEPT__PERCENT_RANK	DEPTNAME	DEPTID	SALARY	ENAME	EID
0	HR	308	58,000	Bobby	17
0.14285714285714285	Workshop	301	51,000	Bob	22
0.2857142857142857	Workshop	301	50,000	Tom	24
0.42857142857142855	HR	308	45,000	Niya	38
0.42857142857142855	Testing	305	45,000	Alice	18
0.7142857142857143	Testing	305	35,000	Jerry	15
0.8571428571428571	Workshop	301	30,000	John	11
0.8571428571428571	Testing	305	30,000	Reyon	16

DENSE_RANK ()

The DENSE_RANK () window function calculates the rank of value in a group of rows based on the ORDER BY expression specified in the OVER clause. For each partition, rank starts from 1. Rows with the same values receive the same rank. DENSE_RANK function does not keep gaps in ranks if there is a similarity between previous one or more rows ranks. This feature makes it different from RANK() function (Table 3.26).

```
SELECT
              DENSE_RANK() OVER (ORDER BY SALARY DESC)
                     AS DEPT__DENSE_RANK,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM  workers;
```

NTILE()

The SQL NTILE() function partitions a logically ordered dataset into a number of buckets demonstrated by the expression and allocates the bucket number to each row. The buckets are numbered from 1 through expression where the expression value must result in a positive integer value for each partition [9]. For example, the following query will allocate rows to three buckets (Table 3.27).

```
SELECT
              ENAME,
              EID,
              DEPTID,
              DEPTNAME,
              SALARY,
              NTILE(3) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY)
                     AS BUCKETS
FROM  workers;
```

TABLE 3.26

Result Table

DEPT__DENSE_RANK	DEPTNAME	DEPTID	SALARY	ENAME	EID
1	HR	308	58,000	Bobby	17
2	Workshop	301	51,000	Bob	22
3	Workshop	301	50,000	Tom	24
4	HR	308	45,000	Niya	38
4	Testing	305	45,000	Alice	18
5	Testing	305	35,000	Jerry	15
6	Workshop	301	30,000	John	11
6	Testing	305	30,000	Reyon	16

TABLE 3.27

Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	PREVIOUS_PERSON_SALARY
Niya	38	308	HR	45,000	1
Bobby	17	308	HR	58,000	2
Reyon	16	305	Testing	30,000	1
Jerry	15	305	Testing	35,000	2
Alice	18	305	Testing	45,000	3
John	11	301	Workshop	30,000	1
Tom	24	301	Workshop	50,000	2
Bob	22	301	Workshop	51,000	3

If PARTITION BY clause is excluded from the above query, then it will give results as follows (Table 3.28).

```
SELECT
              ENAME,
              EID,
              DEPTID,
              DEPTNAME,
              SALARY,
              NTILE(3) OVER (ORDER BY SALARY)
                    AS BUCKETS
FROM  workers;
```

CUME_DIST()

The SQL window function CUME_DIST() returns the cumulative distribution of a value within a partition of values.

The cumulative distribution of a value calculated by the number of rows with values less than or equal to (<=) the current row's value is divided by the total number of rows.

$$N/\text{totalrows}$$

TABLE 3.28

Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	BUCKETS
John	11	301	Workshop	30,000	1
Reyon	16	305	Testing	30,000	1
Jerry	15	305	Testing	35,000	1
Niya	38	308	HR	45,000	2
Alice	18	305	Testing	45,000	2
Tom	24	301	Workshop	50,000	2
Bob	22	301	Workshop	51,000	3
Bobby	17	308	HR	58,000	3

TABLE 3.29
Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	CUME_DIST_VALUE
Niya	38	308	HR	45,000	0.5
Bobby	17	308	HR	58,000	1
Reyon	16	305	Testing	30,000	0.3333333333333333
Jerry	15	305	Testing	35,000	0.6666666666666666
Alice	18	305	Testing	45,000	1
John	11	301	Workshop	30,000	0.3333333333333333
Tom	24	301	Workshop	50,000	0.6666666666666666
Bob	22	301	Workshop	51,000	1

where N is the number of rows with the value less than or equal to the current row value and total rows is the number of rows in the group or result set. Function returns value having a range between 0 and 1 (Table 3.29).

```
SELECT
             ENAME,
             EID,
             DEPTID,
             DEPTNAME,
             SALARY,
             CUME_DIST() OVER (PARTITION BY DEPTNAME ORDER BY
SALARY)
                    AS CUME_DIST_VALUE
FROM  workers;
```

ROW_NUMBER()
The SQL window function **ROW_NUMBER()** is used to display a row number for each row within a specified partition.

```
SELECT
        ROW_NUMBER() OVER (PARTITION BY DEPTNAME ORDER BY
SALARY DESC) AS   ROW_NUM,
             DEPTNAME,
             DEPTID,
             SALARY,
             ENAME,
             EID
FROM workers;
```

The above query will give the following results (Table 3.30):

AVG()
A *window function* applies function across a set of table rows that are related to the current row. The window function does not cause rows to be clustered

TABLE 3.30

Result Table

ROW_NUM	DEPTNAME	DEPTID	SALARY	ENAME	EID
1	HR	308	58,000	Bobby	17
2	HR	308	45,000	Niya	38
1	Testing	305	45,000	Alice	18
2	Testing	305	35,000	Jerry	15
3	Testing	305	30,000	Reyon	16
1	Workshop	301	51,000	Bob	22
2	Workshop	301	50,000	Tom	24
3	Workshop	301	30,000	John	11

into a single output row; the rows maintain their separate identities. The window function is able to access more than just the current row of the query result [6]. To calculate average value of each partition, we can use window function **AVG().** To calculate average salary in each department, we can write the query as follows:

```
SELECT
              AVG(SALARY) OVER (PARTITION BY DEPTNAME)
                    AS AVG_SALARY,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

The above query will give the following results (Table 3.31):

Also, moving aggregate can be calculated by adding **ORDER BY** clause along with **PARTITION BY** in window function with **AVG().**

TABLE 3.31

Result Table

AVG_SALARY	DEPTNAME	DEPTID	SALARY	ENAME	EID
51,500.0000	HR	308	45,000	Niya	38
51,500.0000	HR	308	58,000	Bobby	17
36,666.6667	Testing	305	35,000	Jerry	15
36,666.6667	Testing	305	45,000	Alice	18
36,666.6667	Testing	305	30,000	Reyon	16
43,666.6667	Workshop	301	30,000	John	11
43,666.6667	Workshop	301	50,000	Tom	24
43,666.6667	Workshop	301	51,000	Bob	22

```
SELECT
              AVG(SALARY) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY DESC)
                    AS AVG_SALARY,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

The above query will give the result as follows (Table 3.32):

SUM()

The **SUM()** window function returns the sum of input column or the expression across input values in each partition. For example, to calculate sum of salaries of workers in each department, we can write the query as follows:

```
SELECT
              SUM(SALARY) OVER (PARTITION BY DEPTNAME)
                    AS SUM_SALARY,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

The above query will give the result as follows (Table 3.33):

If we want to calculate moving sum of salaries of each department, then we can add an ORDER BY clause in the above query (Table 3.34).

TABLE 3.32

Result Table

AVG_SALARY	DEPTNAME	DEPTID	SALARY	ENAME	EID
58,000.0000	HR	308	58,000	Bobby	17
51,500.0000	HR	308	45,000	Niya	38
45,000.0000	Testing	305	45,000	Alice	18
40,000.0000	Testing	305	35,000	Jerry	15
36,666.6667	Testing	305	30,000	Reyon	16
51,000.0000	Workshop	301	51,000	Bob	22
50,500.0000	Workshop	301	50,000	Tom	24
43,666.6667	Workshop	301	30,000	John	11

TABLE 3.33

Result Table

SUM_SALARY	DEPTNAME	DEPTID	SALARY	ENAME	EID
103,000	HR	308	45,000	Niya	38
103,000	HR	308	58,000	Bobby	17
110,000	Testing	305	35,000	Jerry	15
110,000	Testing	305	45,000	Alice	18
110,000	Testing	305	30,000	Reyon	16
131,000	Workshop	301	30,000	John	11
131,000	Workshop	301	50,000	Tom	24
131,000	Workshop	301	51,000	Bob	22

TABLE 3.34

Result Table

SUM_SALARY	DEPTNAME	DEPTID	SALARY	ENAME	EID
58,000	HR	308	58,000	Bobby	17
103,000	HR	308	45,000	Niya	38
45,000	Testing	305	45,000	Alice	18
80,000	Testing	305	35,000	Jerry	15
110,000	Testing	305	30,000	Reyon	16
51,000	Workshop	301	51,000	Bob	22
101,000	Workshop	301	50,000	Tom	24
131,000	Workshop	301	30,000	John	11

```
SELECT
              SUM(SALARY) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY DESC)
                     AS SUM_SALARY,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

COUNT()

The **COUNT()** window function counts the number of rows defined by the expression in partition. To count employees in each department, we can write the query as follows:

```
SELECT
              COUNT(ENAME) OVER (PARTITION BY DEPTNAME)
                     AS COUNT_ENAME,
              DEPTNAME,
```

```
            DEPTID,
            SALARY,
            ENAME,
            EID
FROM workers;
```

It gives the following result (Table 3.35):

MIN() and MAX()
The aggregate window functions **MIN() and MAX()** return the minimum and maximum values of an expression within a specified window. The following query will return the maximum and minimum salaries of workers in each department.

```
SELECT
            DEPTNAME,
            DEPTID,
            SALARY,
            ENAME,
            EID,
            MAX(SALARY) OVER (PARTITION BY DEPTNAME) AS
MAX_SAL,
            MIN(SALARY) OVER (PARTITION BY DEPTNAME) AS
MIN_SAL
FROM workers;
```

The above query will return the result as follows (Table 3.36):

LEAD()
SQL **LEAD()** function has a capacity that gives admittance to a column at a predefined actual counterbalance which follows the current row. For example, by utilizing the **LEAD()** function, from the current line, you can get information of the following line, or the second line that follows the current line, or the third line that follows the current line, etc.

TABLE 3.35

Result Table

COUNT_ENAME	DEPTNAME	DEPTID	SALARY	ENAME	EID
2	HR	308	45,000	Niya	38
2	HR	308	58,000	Bobby	17
3	Testing	305	35,000	Jerry	15
3	Testing	305	45,000	Alice	18
3	Testing	305	30,000	Reyon	16
3	Workshop	301	30,000	John	11
3	Workshop	301	50,000	Tom	24
3	Workshop	301	51,000	Bob	22

TABLE 3.36

Result Table

DEPTNAME	DEPTID	SALARY	ENAME	EID	MAX_SAL	MIN_SAL
HR	308	45,000	Niya	38	58,000	45,000
HR	308	58,000	Bobby	17	58,000	45,000
Testing	305	35,000	Jerry	15	45,000	30,000
Testing	305	45,000	Alice	18	45,000	30,000
Testing	305	30,000	Reyon	16	45,000	30,000
Workshop	301	30,000	John	11	51,000	30,000
Workshop	301	50,000	Tom	24	51,000	30,000
Workshop	301	51,000	Bob	22	51,000	30,000

The **LEAD()** function syntax is given below:

```
LEAD(return_value [,offset[, default ]])
OVER (
    PARTITION BY ...
       ORDER BY ...
)
```

In the above syntax, return_value specifies the return value of the following row offsetting from the current row. Offset represents the number of rows forward from the current row from which to access data. The offset must be a nonnegative integer. If the offset is not specified, then it is set default to 1. When offset goes beyond the scope of the partition, then function returns default value. If the value is not specified, then NULL is returned.

The LEAD() function applies to the partitions that are created by the PARTITION BY clause. If PARTITION BY clause is not used, then the whole result set is treated as a single partition.

The sorting of the rows in each partition is done by the ORDER BY clause to which the LEAD() function applies. The following query will extract the salary of the next person in the department, and if the next person is not available in the list, then it will return a NULL value.

```
SELECT
             ENAME,
             EID,
             DEPTID,
             DEPTNAME,
             SALARY,
             LEAD(SALARY) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY)
                    AS NEXT_PERSON_SALARY
FROM  workers;
```

The above query will give the result as follows (Table 3.37):

The LEAD() function can also be very useful for calculating the difference between the value of the current row and the value of the following row. The following query finds the difference between the salaries of person in the same department.

```
SELECT
              ENAME,
              EID,
              DEPTID,
              DEPTNAME,
              SALARY,
              LEAD(SALARY) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY)-SALARY
                        AS SALARY_DIFFERENCE
FROM workers;
```

The above query will give the result as follows (Table 3.38):

FIRST_VALUE()

The SQL window function FIRST_VALUE() returns the first value in an ordered group of a result set or window frame. The following query returns the first salary value in each department ordered by salary.

```
SELECT
        FIRST_VALUE(SALARY) OVER (PARTITION BY DEPTNAME ORDER
BY SALARY DESC) AS   FIRST_ROW,
              DEPTNAME,
              DEPTID,
              SALARY,
              ENAME,
              EID
FROM workers;
```

TABLE 3.37

Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	NEXT_PERSON_SALARY
Niya	38	308	HR	45,000	58,000
Bobby	17	308	HR	58,000	NULL
Reyon	16	305	Testing	30,000	35,000
Jerry	15	305	Testing	35,000	45,000
Alice	18	305	Testing	45,000	NULL
John	11	301	Workshop	30,000	50,000
Tom	24	301	Workshop	50,000	51,000
Bob	22	301	Workshop	51,000	NULL

TABLE 3.38

Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	SALARY_DIFFERENCE
Niya	38	308	HR	45,000	13,000
Bobby	17	308	HR	58,000	NULL
Reyon	16	305	Testing	30,000	5,000
Jerry	15	305	Testing	35,000	10,000
Alice	18	305	Testing	45,000	NULL
John	11	301	Workshop	30,000	20,000
Tom	24	301	Workshop	50,000	1,000
Bob	22	301	Workshop	51,000	NULL

The above query will give the following result (Table 3.39):

LAST_VALUE()

The SQL window function LAST_VALUE() returns the last value in an ordered group of a result set. The following query returns the last salary value in each department ordered by salary.

```
SELECT
        LAST_VALUE(SALARY) OVER (PARTITION BY DEPTNAME ORDER
BY SALARY DESC) AS   LAST_ROW,
             DEPTNAME,
             DEPTID,
             SALARY,
             ENAME,
             EID
FROM workers;
```

TABLE 3.39

Result Table

FIRST_ROW	DEPTNAME	DEPTID	SALARY	ENAME	EID
58,000	HR	308	58,000	Bobby	17
58,000	HR	308	45,000	Niya	38
45,000	Testing	305	45,000	Alice	18
45,000	Testing	305	35,000	Jerry	15
45,000	Testing	305	30,000	Reyon	16
51,000	Workshop	301	51,000	Bob	22
51,000	Workshop	301	50,000	Tom	24
51,000	Workshop	301	30,000	John	11

TABLE 3.40

Result Table

LAST_ROW	DEPTNAME	DEPTID	SALARY	ENAME	EID
58,000	HR	308	58,000	Bobby	17
45,000	HR	308	45,000	Niya	38
45,000	Testing	305	45,000	Alice	18
35,000	Testing	305	35,000	Jerry	15
30,000	Testing	305	30,000	Reyon	16
51,000	Workshop	301	51,000	Bob	22
50,000	Workshop	301	50,000	Tom	24
30,000	Workshop	301	30,000	John	11

The above query will give the following result (Table 3.40):

LAG()

We can use a SQL window function LAG() to access previous row's data based on defined offset value. It works similar to a LEAD() function. In the SQL LEAD() function, we access the values of subsequent rows, but in LAG() function, we access previous row's data. It is useful to compare the current row value from the previous row value.

```
SELECT
            ENAME,
            EID,
            DEPTID,
            DEPTNAME,
            SALARY,
            LAG(SALARY) OVER (PARTITION BY DEPTNAME ORDER BY
SALARY)
                  AS PREVIOUS_PERSON_SALARY
FROM  workers;
```

The above query finds the salary of the previous person in each department based on logically sorted salary value. As no previous row is available for the first row in each department, it returns a NULL value. The result of the above query is given below (Table 3.41):

3.1.5 Preparing Data for Analytics Tool

One of the primary steps performed for data science is the cleaning of the dataset you are working with. The valuable information or patterns you extract from your dataset are just in the same class as the data itself, so maximum of the time spent by a data scientist or analyst includes preparing datasets for use in analysis. SQL can help to speed up this step. Various SQL queries can be used to clean, update, and filter data, by eliminating redundant and

TABLE 3.41

Result Table

ENAME	EID	DEPTID	DEPTNAME	SALARY	PREVIOUS_PERSON_SALARY
Niya	38	308	HR	45,000	NULL
Bobby	17	308	HR	58,000	45,000
Reyon	16	305	Testing	30,000	NULL
Jerry	15	305	Testing	35,000	30,000
Alice	18	305	Testing	45,000	35,000
John	11	301	Workshop	30,000	NULL
Tom	24	301	Workshop	50,000	30,000
Bob	22	301	Workshop	51,000	50,000

unwanted records. This can be done with the different SQL clauses like CASE WHEN, COALESCE, NULLIF, LEAST/GREATEST, Casting, and DISTINCT.

The following example table will be used to demonstrate and understand how the above functions work for dataset cleaning (Table 3.42).

CASE

WHEN

The CASE statement goes through various conditions specified with WHEN clause and returns a value when the first condition is met. It works like nested IF-THEN-ELSE statement. Once a condition is true, it will return the value specified after THEN. Value in the ELSE clause is returned, if no conditions are true. It returns NULL when no conditions are true, and no ELSE part is specified in the query.

Suppose we fetch all data of the above sales table and want to add an extra column that labels as summary which categorizes sales into More, Less, and Avg, this table can be created using a CASE statement as follows:

```
SELECT *,
    CASE
        WHEN quantity >= 10 THEN 'More'
        WHEN quantity >= 6 THEN 'Avg'
        ELSE 'Less'
    END AS summary
```

TABLE 3.42

"sales" Table

sale_no	product_id	quantity	price	customer_name
5,001	3	4	21,000	John
5,002	11	NULL	17,000	Anna
5,003	94	10	105,000	Tom
5,004	86	8	27,000	Nora
5,005	88	18	8,000	Tom

```
FROM
    sales;
```

This query will give the following output (Table 3.43):

COALESCE

Some records of database may consist of NULL values, but while applying statistics to these datasets, you may need to replace these NULL values with some other data. This can be done effectively by the COALESCE function. The first parameter to this function is a column that may consist of NULL, and the second represents value that replaces NULL. It replaces all NULL values specified in column by the second default value given in the function.

The following example replaces NULL by –1 in the quantity column.

```
SELECT
customer_name ,product_id,
COALESCE(quantity, -1) AS quantity
FROM
    sales;
```

This query produces the following results (Table 3.44):

NULLIF

NULLIF function takes two parameters and will return NULL if the first parameter value equals the second value else returns the first parameter.

As an example, imagine that we want to replace product_id value 11 by NULL. This could be done with the following query:

TABLE 3.43

Result Table

sale_no	product id	quantity	price	customer_name	summary
5,001	3	4	21,000	John	Less
5,002	11	NULL	17,000	Anna	Less
5,003	94	10	105,000	Tom	More
5,004	86	8	27,000	Nora	Avg
5,005	88	10	8,000	Tom	More

TABLE 3.44

Result Table

customer_name	product_id	quantity
John	3	4
Anna	11	–1
Tom	94	10
Nora	86	8
Tom	88	10

```
SELECT
sale_no,
customer_name,
NULLIF(product_id, 11) AS product_id
FROM
    sales;
```

The result of the above query is (Table 3.45):

LEAST/GREATEST

The LEAST and GREATEST are frequently used functions for data cleaning. These functions return the least and greatest values from the given set of elements, respectively. These functions are useful to replace value in list, especially if it is too high or low.

For example, minimum price needs to be 10,000 in the above table. This can be done by the following query. Price 8,000 is replaced by value 10,000 in the last row, as 8,000 is less than 10,000, and it replaces it by max value among these two.

```
SELECT
sale_no,
product_id,
    quantity,
GREATEST(10000, price) as price
FROM
    sales;
```

This query will give the following output (Table 3.46):

```
SELECT
sale_no,
product_id,
    quantity,
LEAST(10000, price) as price
FROM
    sales;
```

This query will give the following output (Table 3.47):

TABLE 3.45

Result Table

sale_no	customer_name	product id
5,001	John	3
5,002	Anna	NULL
5,003	Tom	94
5,004	Nora	86
5,005	Tom	88

TABLE 3.46

Result Table

sale_no	product_id	quantity	price
5,001	3	4	21,000
5,002	11	NULL	17,000
5,003	94	10	105,000
5,004	86	8	27,000
5,005	88	10	10,000

TABLE 3.47

Result Table

sale_no	product_id	quantity	price
5,001	3	4	10,000
5,002	11	NULL	10,000
5,003	94	10	10,000
5,004	86	8	10,000
5,005	88	18	8,000

DISTINCT

The DISTINCT keyword returns only distinct values in the specified column value sets. For example, to extract all the unique names in the sales table, you could write the following query:

```
SELECT
    DISTINCT customer_name
FROM
    sales;
```

DISTINCT clause can also be applied to multiple columns to get the distinct combinations of the specified column. The above query gives the following result: It removed duplicate names from the customer_name column (Table 3.48).

TABLE 3.48

Result Table

customer_name
John
Anna
Tom
Nora

3.2 Advanced NoSQL for Data Science

NoSQL is a database design that can accommodate various data models, including key-value, document, columnar, and graph formats. NoSQL, which means "not only SQL", is an alternative to relational databases in which data is stored in tables and has a fixed data schema. NoSQL databases are very useful for working with large distributed data. The NoSQL databases are built in the early 2000s to deal with large-scale database clustering in web and cloud applications.

NoSQL has a **flexible schema**, unlike the traditional relational database model. All rows can have different structures or attributes. NoSQL databases are found to be very useful for handling really big data tasks because it follows the Basically Available, Soft State, Eventual Consistency (**BASE**) approach instead of Atomicity, Consistency, Isolation, and Durability – commonly known as **ACID properties**. Two major drawbacks of SQL are **rigidity** when adding columns and attributes to tables and **slow performance** when many tables need to be joined and when tables store a large amount of data. NoSQL databases tried to overcome these two biggest drawbacks of relational databases. NoSQL offers a more flexible, schema-free solution that can work with unstructured data.

In this section, we will study different categories of NoSQL, including document databases, key-value databases, and graph databases.

3.2.1 Why NoSQL

NoSQL supports unstructured data or semi-structured data. In many applications, an attribute usually needs to be added on the fly, for specific rows, but not every row, and may be of different types than attributes in the rows. Now let us explore some NoSQL features to understand why you should choose NoSQL databases for data science.

Features:

- It is not using the relational model to store data.
- NoSQL running well on clusters.
- It is mostly open-source.
- NoSQL is capable to handle a large amount of social media data.
- NoSQL is schema-less.

3.2.2 Document Databases for Data Science

Document-based NoSQL databases store the data in the JSON object format. Each document has key-value pairs like structures. The document-based NoSQL databases are simple for engineers as they map items as a JSON object. JSON is a very common data format truly adaptable by web developers and permits us

to change the structure whenever required. Some examples of document-based NoSQL databases are CouchDB, MongoDB, OrientDB, and BaseX.

JSON document format:

```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "contribs" : [ "Fortran", "ALGOL", "Form", "FP" ],
  "awards" : [
    {
      "award" :"Dowell Award",
      "year" : 1988,
      "by" :"Computer Society"
    },
{
      "award" :"First Prize",
      "year" : 1993,
      "by" : "National Academy of Engineering"
    }
  ]
}
```

3.2.3 Wide-Column Databases for Data Science

Similar to any relational database, this wide-column database stores the data in records, but it can also store very large numbers of dynamic columns. It groups the dynamically added columns into column families. Instead of having multiple tables like relational databases, we have multiple column families in wide-column databases. Examples of wide-column types of databases are Cassandra and HBase (Figure 3.2).

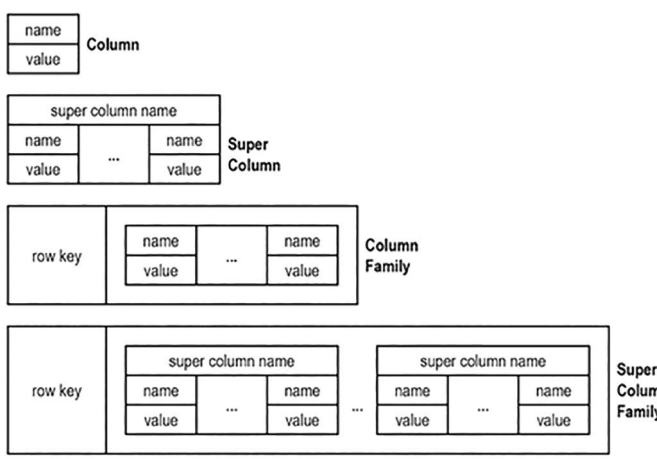


FIGURE 3.2
Pattern for wide-column database.

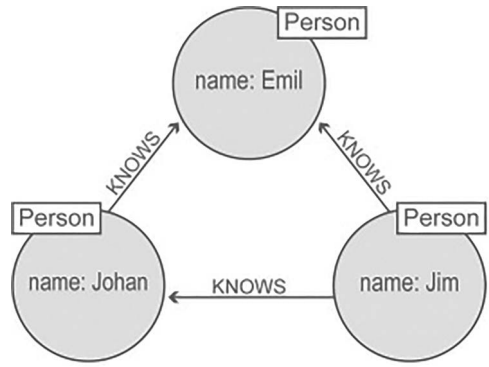


FIGURE 3.3
Simple pattern for graph database.

3.2.4 Graph Databases for Data Science

Graph database stores the data in the form of nodes and edges. The node stores information about the main entities like people, places, and products, and the edge stores the relationships between them. Graph database is very useful to find out the pattern or relationship among data like a social network and recommendation engines. Examples of graph databases are Neo4j and Amazon Neptune (Figure 3.3).

3.3 Summary

Structured Query Language (SQL) has been around for quite a long time. It is a programming language utilized in dealing with the data stored in a relational database. The data scientist can utilize SQL to read, update, search, and analyze the data in a database and create useful patterns or associations to drive a decision-making process. In the first section of this unit, we studied basic statistics with SQL to find the mean, mode, and median of a given dataset. We also looked at a few built-in functions for data munging (or data wrangling which is the phase of data transformation). We then moved on to study many built-in functions to perform data filtering, joins, and aggregation. We studied different built-in window functions to find various ranks and ordered data. Then, we studied how various SQL queries can be used to clean, update, and filter data, by eliminating redundant and unwanted records. To conclude the second section that is NoSQL (i.e., not only SQL), we also got a brief of NoSQL and why it is useful for an unstructured dataset. Also, we studied different categories of NoSQL databases including documents databases, wide-column databases, and graph databases.

Exercise

1. Write a SQL statement to display all the information of all salesmen shown in table below [10]

```
salesman_id  |    name    |   city    | commission
-------------+------------+-----------+------------
        5001 | James Hoog | New York  |       0.15
        5002 | Nail Knite | Paris     |       0.13
        5005 | Pit Alex   | London    |       0.11
        5006 | Mc Lyon    | Paris     |       0.14
        5007 | Paul Adam  | Rome      |       0.13
        5003 | Lauson Hen | San Jose  |       0.12
```

Salesmen Id	Name	City	Commission
5001	James Hooq	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	012

2. Write a SQL statement to display a string "This is SQL Exercise, Practice and Solution".
3. Write a query to display three numbers in three columns.
4. Write a query to display the sum of two numbers 10 and 15 from RDM server.
5. Write a SQL statement to display specific columns like name and commission for all the salesmen (refer table from 1 [10]).
6. Write a query to display the columns in a specific order like order date, salesman id, order number, and purchase amount for all the orders.

```
ord_no  purch_amt     ord_date   customer_id  salesman_id
------ -----------  ------------ -----------  -----------
70001     150.5      2012-10-05     3005         5002
70009     270.65     2012-09-10     3001         5005
70002      65.26     2012-10-05     3002         5001
70004     110.5      2012-08-17     3009         5003
70007     948.5      2012-09-10     3005         5002
70005    2400.6      2012-07-27     3007         5001
70008    5760        2012-09-10     3002         5001
70010    1983.43     2012-10-10     3004         5006
```

```
70003    2480.4       2012-10-10     3009         5003
70012     250.45      2012-06-27     3008         5002
70011      75.29      2012-08-17     3003         5007
70013    3045.6       2012-04-25     3002         5001
```

7. Write a query which will retrieve the value of salesman id of all salesmen, getting orders from the customers in orders table without any repeats [10].

```
ord_no    purch_amt    ord_date   customer_id  salesman_id
-------  ----------   ----------  -----------  -----------
70001        150.5    2012-10-05     3005         5002
70009        270.65   2012-09-10     3001         5005
70002         65.26   2012-10-05     3002         5001
70004        110.5    2012-08-17     3009         5003
70007        948.5    2012-09-10     3005         5002
70005       2400.6    2012-07-27     3007         5001
70008       5760      2012-09-10     3002         5001
70010       1983.43   2012-10-10     3004         5006
70003       2480.4    2012-10-10     3009         5003
70012        250.45   2012-06-27     3008         5002
70011         75.29   2012-08-17     3003         5007
70013       3045.6    2012-04-25     3002         5001
```

8. Write a SQL query to know the winner of the 1971 prize for Literature.

```
YEAR SUBJECT       WINNER                COUNTRY
CATEGORY
------------------------------------------------------------------------
1970 Physics     Hannes Alfven           Sweden
Scientist
1970 Physics     Louis Neel              France
Scientist
1970 Chemistry   Luis Federico Leloir    France
Scientist
1970 Physiology  Ulf von Euler          Sweden
Scientist
1970 Physiology  Bernard Katz           Germany
Scientist
1970 Literature  Aleksandr Solzhenitsyn Russia
Linguist
1970 Economics   Paul Samuelson         USA
Economist
1970 Physiology  Julius Axelrod         USA
Scientist
1971 Physics     Dennis Gabor           Hungary
Scientist
```

```
1971 Chemistry   Gerhard Herzberg       Germany
Scientist
1971 Peace       Willy Brandt           Germany
Chancellor
1971 Literature  Pablo Neruda           Chile
Linguist
1971 Economics   Simon Kuznets          Russia
Economist
1978 Peace       Anwar al-Sadat         Egypt
President
1978 Peace       Menachem Begin         Israel
Prime Minister
1987 Chemistry   Donald J. Cram         USA
Scientist
1987 Chemistry   Jean-Marie Lehn        France
Scientist
1987 Physiology  Susumu Tonegawa        Japan
Scientist
1994 Economics   Reinhard Selten        Germany
Economist
1994 Peace       Yitzhak Rabin          Israel
Prime Minister
1987 Physics     Johannes Georg Bednorz Germany
Scientist
1987 Literature  Joseph Brodsky         Russia
Linguist
1987 Economics   Robert Solow           USA
Economist
1994 Literature  Kenzaburo Oe           Japan
Linguist
```

9. Write a SQL query to display the name and price of all the items with a price equal to or more than Rs. 250, and the list contains the larger price first and then by name in ascending order.

```
PRO_ID PRO_NAME                   PRO_PRICE    PRO_COM
------- ------------------------- ---------- ----------
    101 Mother Board                    3200         15
    102 Key Board                        450         16
    103 ZIP drive                        250         14
    104 Speaker                          550         16
    105 Monitor                         5000         11
    106 DVD drive                        900         12
    107 CD drive                         800         12
    108 Printer                         2600         13
    109 Refill cartridge                 350         13
    110 Mouse                            250         12
```

10. Explain what NoSQL databases are.
11. Explain the purpose of NoSQL databases.
12. List the benefits of NoSQL database over traditional RDBMS database.
13. Identify various types of NoSQL databases.
14. List the differences between NoSQL and RDBMS.
15. Write a NoSQL query to display all the documents in the collection restaurants [10].

Structure of "restaurants" collection:

```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A",
"score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A",
"score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A",
"score": 10 },
     { "date": { "$date": 1322006400000 }, "grade": "A",
"score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B",
"score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

16. Write a NoSQL query to display the fields restaurant_id, name, borough, and cuisine for all the documents in the collection restaurants.
17. Write a NoSQL query to find the restaurant name, borough, longitude and latitude, and cuisine for those restaurants that contain "mon" as three letters somewhere in its name.
18. Write a NoSQL query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

19. Write a NoSQL query to find the restaurant Id, name, and grades for those restaurants that achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many survey dates.
20. Write a NoSQL query to find the restaurant Id, name, borough, and cuisine for those restaurants that belong to the borough Staten Island or Queens or Bronx or Brooklyn.
21. Write a NoSQL query to display the fields restaurant_id, name, borough, and zip code, but exclude the field_id for all the documents in the collection restaurants.

References

1. https://mode.com/sql-tutorial/introduction-to-sql/
2. https://stackoverflow.com/questions/38549/what-is-the-difference-between-inner-join-and-outer-join/38578
3. https://www.investopedia.com/terms/m/median.asp
4. Kathi Kellenberger, Scott Shaw. 2014. *Beginning T-SQL*. Springer Science and Business Media LLC, Germany.
5. https://academy.vertabelo.com/blog/sql-window-functions-examples/
6. https://www.postgresql.org/files/documentation/pdf/9.1/postgresql-9.1-US.pdf
7. Scott Shaw, Kathi Kellenberger. 2012. *Beginning T-SQL*. Springer Science and Business Media LLC, Germany.
8. https://www.edureka.co/blog/sql-joins-types
9. https://docs.yugabyte.com/latest/api/ysql/exprs/window_functions/function-syntax-semantics/
10. https://www.w3resource.com/sql-exercises/sql-retrieve-from-table.php

Part II

Data Modeling and Analytics

4

Data Science Methodology

Big data is an umbrella term for the nontraditional techniques and technologies that are required to collect, aggregate, process, and gain insights from massive datasets. In this entire process, there are various steps like data acquisition, preprocess, mining, prediction, and visualization. At each stage, sophisticated tools are used. Big data analytics[1] is widely adopted in business for financial and operational affairs to make better decisions ultimately increasing the revenue. This gave rise to the need for skilled force to do the analytics and make the best use of big data. This chapter covers the basic introduction to big analytics, data preparation, model planning, and model building.

4.1 Analytics for Data Science

Big data term consists of two words in which "big" is an adjective, followed by "data" which is a noun. So let us understand the word "data" and then the significance of the adjective "big". What is data? Data is everything that surrounds us. It can be seen in the form of images, texts, audios, videos, signals, etc. Digital transformation from the last few decades is the reason why enormous data is getting generated every moment. The big data division consists of three types: structured, semi-structured, and unstructured. Structured data has a well-defined structure, i.e., in columns and rows, So that it can be easily managed, accessible, and used by humans or computers. Semi-structured data is a type of structured data that does not adhere to the formal structure in relational databases or other types of data tables, but instead includes tags or other markers to distinguish semantic elements and maintain hierarchies of record within the data. Unstructured data is very different as it does not follow the structure nor maintains standard hierarchy. It is different all the time. But often, it may have details about data and time. Big data is discussed with five of its characteristics that are known as 5V's: volume, velocity, variety, veracity, and value.

Let us discuss it one by one. Figure 4.1 illustrates the 5V's of big data, which are considered as crucial factors.

DOI: 10.1201/9780429443237-6

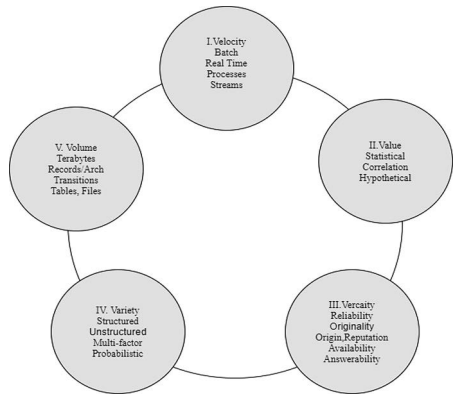


FIGURE 4.1
The 5 V's of big data.[16]

Volume

Volume is a huge amount of information. The size of the data plays a very crucial part in determining the relevance and importance of the data.[2] This indicates that whether or not a particular data should be classified as big data depends on the amount of the data.[2] Therefore, it is important to find a characteristic "volume" when dealing with big data.

Example: Global Internet traffic in 2016 was measured at 6.2 exabytes (6.2 billion GB) per month. We will also have almost 40,000 exabytes of data by the year 2020.

Velocity

The term "velocity" refers to the speed of generation of data. The data flows in from sources such as machines, networks, social media, and mobile phones. This inflow is supposed to be as fast as close to real time as possible. Velocity can give a greater competitive advantage when compared to the volume. Sometimes, the need is to get limited data in real time rather than getting a bulk of data at low speed.

Example: More than 3.5 billion searches are made on Google in a day. Facebook users are also rising year by year by 22% (approx.).

Variety

The third V of big data is the variety. Variety talks about the nature of data. This data might be structured, semi-structured, and unstructured data.[2] It also refers to heterogeneous sources. When the data comes from both inside and outside of an enterprise, it brings variety along with the volume from various resources.[2] It can be structured, semi-structured, and unstructured.[2] All types of data like photos, videos, and audios, making about 80% of the data to be completely unstructured and structured data, are just the tip of the iceberg.[3]

Veracity
The data that is collected from a variety of sources in a huge amount at very high speed makes it vulnerable to inconsistencies and uncertainty. This means the data may get messy. Thus, monitoring the quality and accuracy of the data can be a challenging task. As we know, a major part of the data is unstructured and irrelevant.[3] Big data needs to be cleaned to make it reliable enough.

For example, bulk data can generate uncertainty, while less data can convey half or incomplete information.[2]

Value
All the 4 V's have their importance. But the fifth V that stands for value is at the top of the big data pyramid. The data that comes which is so large must be processed to extract out the value or knowledge out of it. It is not just the volume that matters but also the insights that we derive from it.

4.2 Examples of Data Analytics

This section gives you a brief of how big data analytics is used in real-life problems and what positive effect it can have on any business. Big data analytics is used in many organizations to generate reports and dashboards based on huge data of past and present. Some of the examples of big data analytics[4] are listed below:

1. Fraud detection report is commonly used in banking sectors to identify transactions involving fraud, hacking, unauthorized access to the account, etc.
2. Live tracking report that transportation sectors such as Meru, Ola, Uber, and Mega typically use to track cars, customer requests, payment processing, emergency warning, and find regular needs and revenue, and so on.
3. Sales forecast and plan analysis that is often used by all sectors to assess their customers' sales, profits, and needs and also used to evaluate the future target, etc.
4. Often used for handling live data in many entertainment pages, share market, real-time Sensex data, etc., many reports focused on live data.
5. Generate different types of alerts based on various events, such as alerts created by the data center, using various big data analytics examples notifications.
6. Google analytics reports that we can get how many users visit count, where the user is from, which computer the client is using, etc.

7. Nowadays, many healthcare organizations have quickly introduced big data predictive analytics to improve our daily lives. It has been used to update many healthcare sector procedures and has also been used to enhance outcomes for whole populations.
8. Applications of big data analysis have also played a critical role in many crises. Earthquake killed and injured many people in Nepal in April 2015. In this situation, SAS based in North Carolina was established by analytics, which played a massive role in rescue and relief operations.
9. In online and physical protection, big data analytics has been used to detect malicious activities, take different measures to deter such attacks, introduce real-time surveillance to minimize fraud activity, and also trigger alerts against suspicious acts.

4.3 Data Analytics Life Cycle

The data analytics life cycle is required for problems related to big data and data science applications. Figure 4.2 represents the data analytics life cycle, which is a circular process consisting of six basic phases. The method is iterative to depict a specific project; the project returns to an earlier phase as new information is discovered. The life cycle of data analytics defines best practices in the analytical process from discovery to project completion.

You will still learn something new in a phase as you can see in the graphic to allow you to go back and refine the research performed in the preceding process. For this purpose, the graphic is displayed as a loop and circular arrows are meant to demonstrate that you should travel iteratively between

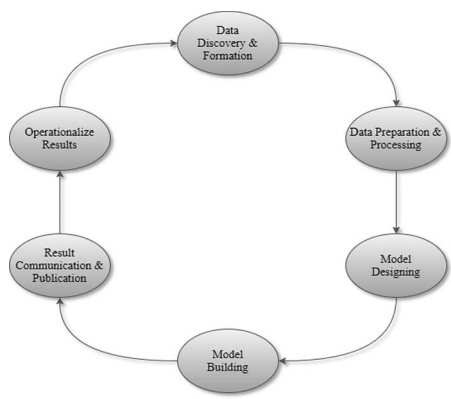


FIGURE 4.2
The data analytics life cycle.

phases before you have enough details to keep going forward. The callouts are questions you can ask yourself to determine whether you have made adequate progress to proceed to the next step of the cycle. Phases involved in the process are data creation, data preparation, system preparing, plan construction, results communication, and operationalization. Throughout this section, we will discuss in depth the process of data preparation, model planning, and model building as these are directly related to data science in big data analytics and will discuss the rest of the phases in brief.

4.3.1 Data Discovery

This is the first step in the life cycle of data analytics. The data science team has to analyze the problem during this process and establish meaning and understanding. Group will find out what data sources are needed and accessible for the project. The team also builds up initial hypotheses that can be checked with the data later. Throughout this step, the team will carry out five key activities: define the data sources, capture the aggregate data sources, review the raw data, determine the required data structures and resources, and scope the kind of data infrastructure needed for this type of problem.

4.3.2 Data Preparation

The preparation of data involves some cleaning as well as choosing some appropriate samples for training and testing. Also, any appropriate combining or aggregating of datasets or elements is done during this level. This step aims to create the dataset to be used in the process's subsequent modeling phase. This phase takes a lot of time and is the most labor-intensive as compared to other phases. Almost half of the time of the project is spent in this phase. In this phase, the team needs to create an environment that is separate from the production environment. This is done by creating an analytical sandbox. ETLT is performed in which Extraction, Transformation, and Loading (ETL) of the data and Extract, Load and Transform (ELT) of the data is executed by the teams to get data into the sandbox for analysis. The team also has to think about the data conditioning methods to turn it into a format to facilitate further analysis. This may be done by visualizing the data using charts and graphs to understand the data in a better way. This may help in understanding the trends, outliers, cycles, and various relations between the data variables. Many a time, the team underestimates the phase of data preparation and therefore ends up coming back to this phase. Sometimes, when the team is in the model building phase, they realize that the data they are working on is not up to the mark and not giving proper results. As a result, they have to go back to the data preparation phase to check exactly where things went wrong.[5]

Preparing the analytical sandbox:
Realizing the analytical sandbox is the very first subphase of data planning. The data preparation process includes the existence of an analytical sandbox (workspace) in which the team can work with the data for the duration of the project and conduct analytics. While developing the analytical sandbox, the relevant data of large amounts and variety is aggregated. This can include all kinds of data like summary-level aggregated data, raw data feed, and unstructured text data, and data from call logs or weblogs, depending upon the kind of analytics the team wants to undertake. Consider an example where the team would need to deal with financial data from a client. Instead of dealing with the production version of the organization's main database, the team will access a copy of the financial data from the computational sandbox, as this will be closely managed and required for financial reporting. This is always the case that data scientists need the full amount of data to work on but have access to restricted data. Due to these conflicting opinions on access and use of data, data science needs to clarify the objectives and convey what they are trying to achieve. The size of the analytical sandbox can vary depending on the project. There is a good rule one should keep in mind while making the sandbox. The size of the sandbox should be planned at least 5–10 times the size of the original datasets, partly because multiple copies of the data can be created to serve as specific tables or data stored for specific kinds of analysis.

Figure 4.3 shows the ETL process, which is a type of data integration process:

Performing ETL
ETL consists of process series and application series. To build a data center, it is done to combine all the data coming from various sources. It consumes quite a sizeable amount of effort to develop a data warehouse. It needs data analysts, database designers, and software developers to have the skills.[6] The data warehouse changes periodically with the source of the data. Additionally, the data warehouse needs to evolve as business shifts – to maintain its value

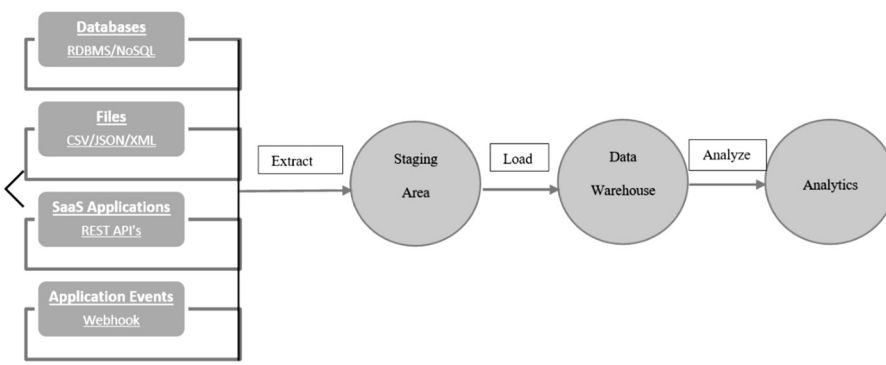


FIGURE 4.3
The ETL process.

as a tool for decision-makers, as a consequence the ETL often changes and expands.[6] ETL stands for extract, load, transform. Information is extracted from an OLTP database in the ETL process, then transformed to suit the data warehouse system, and eventually loaded into the database of the data warehouse.[6] The extraction of data can also be achieved from non-OLTP systems. The ETL consists of three main processes: extraction, transformation, and loading. Let us see all the steps one by one.

i. Extraction

This is the very first step in the cycle of ETL. Few items should be remembered when carrying out the extraction. The team is expected to be well aware of the use of ODBC/JDBC drivers to link to database sources, should understand all data structures of the data sources, and should also be aware of the handling of resources of various nature.[7] Extraction takes place in phases which are the initial extraction and the data extraction modified. Initial extraction only takes place until the data is collected in vast amounts from various resources. The gradual extraction is called data capture modification (CDC). This method extracts back in time the data that has changed about a well-defined case. Based on the update cycle and company requirements, this phase is periodic.

ii. Transformation

In ETL's second phase, the data is properly cleaned and is ready for further use. The team makes sure the data is correct, complete, coherent, and unambiguous. The process involves washing, installation, and data integration.

iii. Loading

The final step on ETL is to load the data into the multidimensional structure of the target. Through this, the data extracted and transformed through previous steps is loaded into the dimensional framework for further processing where it is accessed by the team.

The data scientists often want to hold the raw data as it is and load it into the warehouse. The rationale for this approach is that the conservation and preservation of raw data in the sandbox is of great importance before any transformation takes place. Consider, for example, an essay on credit card use for fraud detection. Outliers in this data population will also reflect high-risk transactions that may suggest a fraudulent activity on the credit card. Now, if ETL is used, it will accidentally clean or filter out the outliers required to detect fraudulent credit card activity when analyzing the data. The team will want clean data and aggregated data and may need to maintain a backup of the original data to compare or assess for patterns and insights that may have existed in the data before the cleaning process. The team may require to consider how to parallel the transfer of datasets into the sandbox, depending

on the size and number of the data sources. For this reason, the transfer of large volumes of data is often referred to as the big ETL.

An application programming interface (API) also has gained considerable popularity. Many websites use APIs to provide access to large amounts of data to support projects. Twitter API is a fine example of such an API that allows millions of tweets to be downloaded to do projects.

Tools for data preparation
To make things easier, there are several tools available that are specially used for data preparation:

- Hadoop
  Hadoop allows data scientists to explore the complexities that exist in the data, even if they cannot make sense of it. It allows data scientists to store data where it is, and that is the whole point of data discovery. A data scientist does not need to grasp the specifics when dealing with large quantities of data.
- Alpine Miner
  This tool includes a graphical user interface (GUI) to develop analytical workflows, including data manipulation and a sequence of analytical events on PostgreSQL and other big data sources, such as structured data mining techniques (e.g., first select the top 100 customers, and then run descriptive statistics and clustering).
- OpenRefine
  OpenRefine is a standalone open-source tool for data cleanup and transformation to other formats, called data wrangling. The operations are more like spreadsheets, i.e. on rows and columns. This tool is mostly used for cleaning messy data, transformations of data, parsing data from websites, etc.
- Data wranglers
  With Trifacta Wrangler, any user with a Mac or Windows machine and an Internet connection can download, install, and start using Trifacta immediately. Trifacta Wrangler empowers analysts to wrangle various data sources on their desktop in preparation for use in analytical or visualization tools such as Tableau.

The data preparation phase takes a lot of time, but it lays the foundation for further work. The quality data is now ready to get fed into the model.

4.3.3 Model Planning

This is the third phase of the data analytics cycle. Based on the type of project, the team selects a possible analytical model and the corresponding variables and other inputs. The model planning phase may seem straightforward, but it is often necessary to perform extra data exploration, data conditioning, and

transformations to prepare the data for the model building phase. Some of the activities to be considered in this phase are as follows:

- Assessing the structure of the datasets: The structure of the datasets is to be studied properly. It is one factor that dictates the tools and analytical technique for the next phase.
- To ensure that the analytical technique will meet the goals and objectives, the team is trying to achieve.
- In some cases, a single model does not suffice the requirements. Therefore, series of techniques as part of the large analytical workflow is needed.

Data exploration and Variable selection
In this substep, the main objective of exploring the data is to know the relationships among the variables. Not only is it important to understand the relationships between the input variables and the outcome variables, but it is also important to understand the relationship, if any, between the input variables. When there is a significant correlation between two or more input variables, it may be useful to perform some variable reduction activities.

Model selection
The main goal of this substep is to choose an analytical technique or a shortlist of candidate techniques based on the end of the project or the purpose of analysis, for example, exploratory or prediction. For the selection of a model, the types of input and output variables play an important role. The team has to decide whether they should use one single model or series of models depending on the type of analysis they are doing. After selecting the model, a proper analytical tool is to be determined to fit the selected model. It is often useful to revisit the analytic challenge at this stage of the project and to ensure that the analytic challenge is still relevant and that there is not any scope creep in the project.

Common tools for the model planning phase
Some tools listed below are commonly used in the model planning phase:

- R

  R is the leading analytical method in the industry and is widely used to process statistics and data. This can manipulate the data easily and show it in a variety of ways. In many respects, SAS has been overtaken, such as data performance, production, and outcome. R installs and runs on several platforms, including UNIX, Windows, and macOS. It has 11,556 packages, which allows you to browse by type of packages. R also provides tools for the automated installation of all packages as per user requirement, which can be conveniently installed with big data, too.

- Tableau
  Tableau Public is a free program that links any data source, whether it be a corporate data warehouse, Microsoft Excel, or web-based data, and generates visualizations of results, charts, dashboards, etc., with web-based real-time updates. They can also be shared with the customer, or via social media. This allows the right to view the file in different formats. The big data capabilities of Tableau make them important, and one can better analyze and visualize the data than any other business program for data visualization.
- SAS
  SAS is easy to access, is easy to handle, and can analyze data from any source. SAS has launched a wide range of customer intelligence products and various web, social media, and marketing analytics SAS modules which are commonly used to profile customers and prospects. This can also predict, monitor, and refine their social behaviors.

4.3.4 Model Building

In the model building phase, the selected analytical technique is applied to a set of training data. This process is known as "training the model". A separate set of data, known as the testing data, is then used to evaluate how well the model performs. This is sometimes known as the pilot test. Often, the fitted model is to be applied to future observations. So, it is not typically sufficient to obtain the best model that explains all of the data; one must build a model that adequately predicts the future.[8]

Creating functional models that are appropriate for a particular situation requires careful attention to ensure that the models that are being built eventually follow the goals outlined in phase 1. Questions to be considered include the following:

- Does the model appear valid and accurate on the test data?
- Does the output/behavior of the model make sense to domain experts? In other words, does it seem as though the model provides answers that make sense in this context?
- Do the parameter values of the fitted model make sense in the context of the domain?
- Is the model sufficiently accurate to meet the goal?
- Does the model avoid intolerable mistakes?
- Are more data or more inputs needed? Do any of the inputs need to be transformed or eliminated?
- Will the kind of model chosen support the runtime requirements?
- Is a different form of the model required to address the business problem? If so, go back to the model planning phase and revise the modeling approach.

When the data analytics team can decide whether the model is robust enough to solve the problem, or if the team has failed, it will progress to the next phase in the data analytics life cycle.

Common Tools used for model building phase
Many tools are available for the model building phase. Below are some of the famous tools listed that are used for statistical analysis and data mining. Some of these tools are open and some are commercial.[9]

- SAS
  It is one of those data analysis tools that are designed specifically for statistical operations.[10] SAS is a closed-source business method used for data mining by major corporations.[10] SAS uses the programming language of base SAS to implement statistical modeling.[11] It is widely used by professionals and businesses working on stable commercial applications. As a data scientist, SAS offers various mathematical databases and tools that you can use to model and coordinate your tests. While SAS is highly reliable and strongly supported by the company, it is very costly and is used mainly by larger industries.
- Apache Spark
  Apache Spark, or simply Spark, is a versatile analytics engine[12] and the data analysis platform that is most used.[10] Spark is intended mainly for batch processing and stream processing.[10] It comes with several APIs that make routine access to data for machine learning, SQL server, etc., easier for data scientists. It is an improvement over Hadoop and can run 100 times faster than MapReduce.[10] Spark has many machine learning APIs that can help data scientists predict data effectively.[10]
- BigML
  BigML offers structured software that uses cloud computing to meet market requirements.[10] BigML is another commonly used data science tool.[13] It offers a fully interactive, cloud-based GUI platform that you can use to process machine learning algorithms.[10] It allows collaborative data visualization and allows you to export visual graphs to your smartphone or IoT computers.[10] BigML offers an easy-to-use web interface using REST APIs, and you can create a free account or a paid account depending on your data needs.[10]
- MATLAB
  It is a closed-source software providing matrix functions, algorithmic execution, and simulation of statistical results. MATLAB is most widely used in various fields of science. MATLAB graphics library helps you to create powerful visualizations. Even MATLAB is used for manipulating images and signals. This makes a very versatile tool for data scientists for solving all issues, from data cleaning and analysis to more sophisticated deep learning algorithms.

- Jupyter
  Project Jupyter is an open-source framework based on IPython, which is designed to help developers create open-source software and experience immersive computing. Jupyter supports various languages, including Julia, Python, and R. It is a website for web applications used for writing live code, visualizations, and presentations.[10] Jupyter is a widely used platform that meets data science requirements. It is 100% open-source and is therefore free. Colaboratory is an online Jupyter program that runs on the cloud and stores data on Google Drive.
- Scikit
  Scikit-learn is a library based on Python, used to implement machine learning algorithms. A tool widely used for data science and analysis is fast and simple to implement and supports a range of machine learning features such as preprocessing, classification, regression, clustering, and reduction in dimensions.[14]
- TensorFlow
  An open-source and ever-evolving toolkit that is famous for its performance and high computational capabilities. TensorFlow has become a popular machine learning tool. This is commonly used for sophisticated machine learning applications such as deep learning. Developers called TensorFlow after tensors that are multidimensional arrays. The capability of TensorFlow to run both CPU and GPU gives it an advantage in the computing power of sophisticated machine learning algorithms.
- Weka
  Weka, or Waikato Environment for Knowledge Analysis, is a tool written in Java for machine learning.[10] This involves various methods of machine learning, such as classification, clustering, regression, visualization, and preparation of data.[10] It is an open-source GUI program, which enables the application of machine learning algorithms through an interactive interface.[15]

4.3.5 Communicate Results

This is the second last phase of the analytics cycle. After obtaining an acceptable model, the team has to communicate the project's findings and the business value of the model to the sponsors and the stakeholders. If the desired business outcome is not obtained, this result also must be communicated. The results can be communicated by preparing presentations for sponsors and analysts. While the communication results phase may seem to be the simplest and most straightforward, it can often be the most challenging part of the exercise. The team must not only provide the outcome of their efforts but, in some cases,

must also overcome the stakeholder's notions of what the most influential input variables are. Furthermore, unless the project sponsors and stakeholders are kept periodically updated on the project as well as the objective of the project, it is typical for someone to state, "This is not what I asked for!" So it is important to remind the audience about the business problem and the scope of the project. The team has to build a strategy to communicate the findings, by including caveats, assumptions, and any limitations of results. They also add the recommendation for future work or improvements to the existing processes.

4.3.6 Operationalization

When the stakeholders agree to implement the model in the production environment, the operationalization phase begins. Depending on the organization, the project team may be responsible for the model's implementation or may simply transfer the code and other technical documentation to a different team. During this phase, it is important to establish the approach to monitor the performance of the model after it is placed into production. Implementing the model in the test environment helps to minimize impacts on production. It is common to run a pilot program before fully implementing the model in production. Running a pilot helps minimize risk and further demonstrates the business value. Testing the model in a live setting allows the team to learn from the deployment and make necessary adjustments before launching across the enterprise. After the model is placed into production, it is often necessary to monitor the model's performance and establish a process to retrain and update the model. Any further communication of results often occurs during the operationalization phase; the executives will be interested in knowing the return on investment from their investment.

4.4 Summary

The chapter gives a detailed view of data science in big data analytics. It begins with the introduction to big data and covering its various characteristics. Further, the data analytics life cycle is discussed, which is a popular approach to execute an analytical project. The six phases are data discovery, data preparation, model planning, model building, communicate results, and operationalization. While following these steps, the data scientist team has to perform various operations to gain more and more insights and knowledge out of the data. At various phases, different objectives are achieved using specialized tools. A great deal of time is spent in phase 1 and phase 2, which are data discovery and data preparation. It is in these phases the data sources are identified and data is prepared for further processing. Decisions

for model planning and model building are taken as per the nature of the data. Subsequent phases like communicate results and operationalization are followed to complete the entire project.

Exercise

1. What is big data? Explain the characteristics of big data.
2. Describe any two phases of the data analytical life cycle.
3. Which phase of the analytical life cycle requires a lot of time to invest and why?
4. What are the different commercial and noncommercial tools available for the model building phase?
5. Write a short note on SAS and Tableau.
6. What is a sandbox and why it is important?

References

1. Fawcett Tom. 2013. *Data Science for Business*. Sebastopol: O'Reilly Media.
2. 2019. 5 V's of big data. https://www.geeksforgeeks.org/5-vs-of-big-data/ (accessed 15 November 2020).
3. Kiran Ravi. 2020. Big data characteristics: Know the 5'Vs of big data. Big Data and Hadoop. https://www.edureka.co/blog/big-data-characteristics/ (accessed 15 November 2020).
4. Venkat Ankam. 2016. *Big Data Analytics*. Birmingham: Packt.
5. Sherman Rick. 2015. Data preparation process. Science Direct. https://www.sciencedirect.com/topics/computer-science/data-preparation-process (accessed 18 November 2020).
6. 14 most used data science tools for 2019 – Essential data science ingredients. Data Flair. https://data-flair.training/blogs/data-science-tools/ (accessed 13 November 2020).
7. EI-Sappagh, Hendawi Abdeltawab M. Ahmed, Bastawissy Ali Hamed El. 2011. A proposed model for data warehouse ETL processes. *Journal of King Saud University - Computer and Information Sciences* 23(2). doi: 10.1016/j.jksuci.2011.05.005.05.005
8. Model building and Assessment. MathWorks. https://www.mathworks.com (accessed 24 November 2020).
9. Top 10 data analytics tools. Data science and analytics. Pro School. https://www.proschoolonline.com/blog/top-10-data-analytics-tools (accessed 20 November 2020).
10. EMC Education Services. 2015. *Data Science and Big Data Analytics*: Wiley Publications, US.
11. SAS documentation. http://documentation.sas.com (accessed 22 November 2020).

12. ML pipelines. Apache Spark. https://spark.apache.org/docs/latest/ml-pipeline.html (accessed 22 November 2020).
13. BigML. https://bigml.com/features (accessed 23 November 2020).
14. An introduction to Machine Learning with Sci-kit Learn. Sci-kit Learn. learn.org/stable/tutorial/basic/tutorial.html (accessed 27 November 2020).
15. Weka. https://www.cs.waikato.ac.nz (accessed 27 November 2020).
16. 2019. Compare and contrast five clustering algorithms on your own. Nursing Homeworks. https://nursinghomeworks.com/compare-and-contrast-five-clustering-algorithms-on-your-own/ (accessed 12 November 2020).

5

Data Science Methods and Machine Learning

Data analytics is a broad field where various techniques are used to extract important information from raw data and analyze the business chain supporting decision-making. Various methods are used in different stages according to the requirements and the type of datasets. Machine learning (ML) is majorly used for building analytical models to give fruitful insights. In this chapter, we will go through regression analysis and some important ML algorithms that are highly used in data analytics.

5.1 Regression Analysis

Regression analysis examines the relationship between a target (dependent variable(s)) and a predictor (independent variable(s)). It also helps us to examine the amount of impact the predictors make on the target variable. There are various regression techniques based on the number of predictors, the shape of the line of regression, and the type of target variable. The coming subtopics will give you a brief about different types of regression analysis.

5.1.1 Linear Regression

Linear regression is a supervised ML algorithm.[1] Linear regression models the linear relationship between the target variable and the predictors. When we say, the relationship between the dependent and independent variables, it means how the target variable is impacted due to a change in the magnitude of the predictors.

Linear regression can be represented as follows:

$$\text{Target variable} = \text{intercept} + (\text{constant} \times \text{predictors})$$

A linear regression model has certain important assumptions. They are as follows:

1. A linear relationship between the target variable and the predictor(s).
2. The error term should have a normal distribution.

DOI: 10.1201/9780429443237-7

3. No correlation among the residual terms. The presence of correlation among the residual terms is called autocorrelation.
4. The predictors should be independent of each other. If this is not the case, it leads to the multicollinearity problem.
5. All the residuals should possess the same variance. Such a case is called homoscedasticity. The absence of such a situation is called heteroscedasticity.

Based on the number of predictors, i.e., independent variables, linear regression is classified into two types[2]:

1. **Simple Linear Regression**[2] – The model constructed for the target is against one predictor only.
2. **Multiple Linear Regression** – The model constructed for the target is against two or more predictors.

1. Simple Linear Regression

   Simple linear regression is the easiest regression model where the linear relationship of two continuous variables is determined.[3] Among the two variables used in simple linear regression, one is the target variable and the other is the predictor.[3]

   Figure 5.1 represents the simple linear regression model, which is a statistical method that helps to summarize and study relationships between two quantitative variables.

   The main objective of the simple linear regression is to find the best-fit line with the least error. Here, the equation of the line can be written as follows:

$$f(x) = \beta_0 + \beta_1 x$$

   β_0 is the intercept of the line,

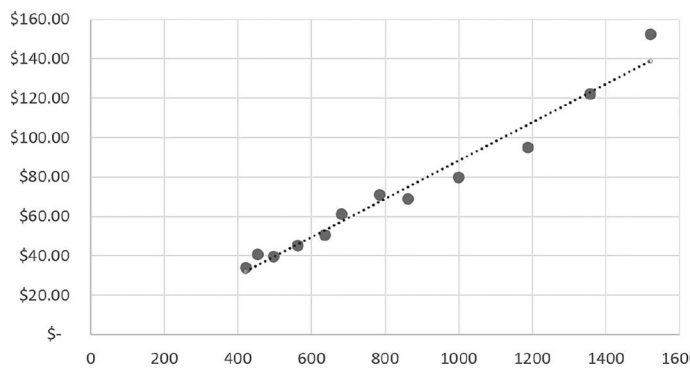


FIGURE 5.1
Linear regression model.

β_1 is the slope of the line.

The simple linear regression model defines the target variable as follows:

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

Here, "e_i" is the error term, i.e., the distance between the specific observations on the model to the regression line. Also, the intercepts and the slope (β_0 and β_1, respectively) are known as the regression coefficients.

To mathematically solve a linear regression problem, the following equations are used.

Say, our regression equation is in the form: $y_i = \beta_0 + \beta_1 x_i$

To perform regression analysis, the values of β_0 and β_1 need to be calculated.

$$\textbf{Slope}(\boldsymbol{\beta_1}) = \Sigma\left[(x_i - x)(y_i - \bar{y})\right] \Big/ \sum (x_i - x)^2$$

$$\textbf{Intercept}(\boldsymbol{\beta_0}) = \bar{y} - \beta_1 \cdot \bar{x}$$

Here, $\bar{x}$ and $\bar{y}$ are the mean of the values in the independent variable (x) and dependent variable (y), respectively.

Coefficient of Determination (R^2)

The coefficient of determination also known as the R squared is a measure that determines the difference caused by a predictor to the target value.

$$R^2 = \left\{ \frac{(1/N) \times \left(\Sigma\left[(x_i - \bar{x})^* (y_i - \bar{y})\right]\right)}{\sqrt{\sum (x_i - \bar{x})^2 / n} \times \sqrt{\sum (y_i - \bar{y})^2 / n}} \right\}^2$$

For example, if $R^2 = 0.45$, this signifies that about 45% of the difference in the target variable can be established using the predictor variable.

Let us go through a numeric example.

Problem Statement: The average rainfall (in mm) for the last 5 years in the state of Maharashtra was recorded along with the umbrellas sold in that particular year.

1. Find the equation of linear regression.
2. Suppose this year the state records a rainfall of 123.2 mm, what will be the expected number of umbrellas that will be sold?
3. Calculate the coefficient of determination (R^2) and comment on it (Table 5.1).

TABLE 5.1

Linear Regression Example

Rainfall (mm)	Umbrellas Sold
121.2	52
152.6	72
98.4	40
171	100
85.6	34

Solution:

For our convenience, we will rename the rainfall column as x_i and similarly the umbrellas sold column as y_i. We will first calculate the deviations of x_i and y_i. Here, the deviation is the difference between the rainfall recorded in a particular year and the average rainfall in 5 years (Table 5.2).

Now, we will calculate the square of both the deviations and the product of the deviations (Table 5.3).

The equation of the linear regression is $y_i = \beta_0 + \beta_1 x_i^3$

We will have to solve for the regression coefficients β_0 and β_1.[4]

TABLE 5.2

Linear Regression Example – Calculation of Deviation

Year	x_i	y_i	$(x_i - \bar{x})$	$(y_i - \bar{y})$
2015	121.2	52	–4.56	–7.6
2016	152.6	72	26.84	12.4
2017	98.4	40	–27.36	–19.6
2018	171	100	45.24	40.4
2019	85.6	34	–40.16	–25.6
Sum	628.8	298		
Mean	125.76	59.6		

TABLE 5.3

Linear Regression Example – Squaring the Deviations and Product of the Deviations

Year	x_i	y_i	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$	$(x_i - \bar{x})^2(y_i - \bar{y})^2$
2015	121.2	52	20.7936	57.76	34.656
2016	152.6	72	720.3856	153.76	332.816
2017	98.4	40	748.5696	384.16	536.256
2018	171	100	2,046.658	1,632.16	1,827.696
2019	85.6	34	1,612.826	655.36	1,028.096
Sum	628.8	298	5,149.232	2,883.2	3,759.52
Mean	125.76	59.6			

$$\beta_1 = \sum\left[(x_i - x)(y_i - \bar{y})\right] / \sum\left[(x_i - \bar{x})^2\right]^4$$

$$= \left[3759.52/5149.232\right]$$

$$= 0.730$$

$$\beta_0 = \bar{y} - \beta_1 \cdot \bar{x}$$

$$= 59.6 - (0.730 \times 125.76)$$

$$= 59.6 - 91.805$$

$$= -32.205$$

We know that $\beta_1 = 0.730$ and $\beta_0 = -32.205$. Therefore, our regression line equation is as follows:

$$y_i = 0.730 \times x_i - 32.205$$

$$y_i = 0.730 \times x_i - 32.205$$

$$= 0.730 \times 123.2 - 32.205$$

$$= 57.731$$

So, as per our regression equation, if the state of Maharashtra experiences rainfall of 123.2 mm, the expected number of umbrellas to be sold will be approximately 58 (57.731).

Coefficient of Determination (R^2)

The formula to calculate R^2 is as follows:

$$R^2 = \left\{ \frac{(1/N) \times \left(\sum\left[(x_i - \bar{x}) * (y_i - \bar{y})\right]\right)}{\sqrt{\sum (x_i - \bar{x})^2/n} \times \sqrt{\sum (y_i - \bar{y})^2/n}} \right\}^2$$

$$= \left\{ \frac{(1/5) \times \left([3759.52]\right)}{\sqrt{5149.232/5} \times \sqrt{2883.2/5}} \right\}^2$$

$$= \left\{ \frac{751.904}{32.091 \times 24.013} \right\}^2$$

$$= \left\{ \frac{751.904}{770.601} \right\}^2$$

$$= \{0.976\}^2$$

$$= 0.953$$

The coefficient of determination (R^2) is 0.953, which depicts that 95% of the difference in the number of umbrellas sold can be described by its relationship to the rainfall (in mm).

2. Multiple Linear Regression

Multiple linear regression is simply called multiple regression.[5] Multiple regression is the same as that of simple regression, just the difference is that, in multiple regression, more than one independent variable (predictor) is used to model the target variable.

In multiple linear regression, the dependent variable is of continuous type, whereas the independent variables are of either categorical type or continuous type.[6] Though a multiple linear regression model uses multiple predictors to find the best-fit line, every individual feature should be capable to find a relation with the target variable. As, in this type of regression, various predictors are used, the best-fit line is plotted in a multidimensional area with the data elements.

The linear equation of the multiple linear regression model is mathematically represented as follows:[3]

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n^3$$

where y_i is the target variable that can be predicted,

$x_1, x_2, x_3, \ldots, x_n$ are the independent variables (predictors),

$\beta_0, \beta_1, \beta_3, \ldots, \beta_n$ are the regression coefficients.

In the multiple linear regression model, the independent variables can be of categorical type. These variables cannot be directly used to build a model. Such variables are converted into dummy variables and then used to build a linear model. For example, suppose we have an independent variable that depicts whether the student has won or lost a certain competition, then its dummy variable can be represented as 1 or 0 where 1 represents the presence of the value and 0 represents the absence of the value or vice versa. Table 5.4 gives you a clear idea of the dummy variable usage.

TABLE 5.4

Dummy Variable Trap Example

Competition Results	Win	Lose
Win	1	0
Lose	0	1
Lose	0	1
Win	1	0
Lose	0	1

Dummy Variable Trap

The dummy variable trap is one of the important aspects of analysis that can give rise to a serious multicollinearity issue. If a categorical variable has too many values, say, it has n values, then one should always consider (n–1) dummy variables. The reason behind it is that if we define the dummy variables, the nth dummy variable will be of the same meaning as the exception of the other dummy variables. Defining "n" dummy variables where (n–1) dummy variables convey the same information leads to a dummy variable trap.

The following example will give a clear vision for the dummy variable trap.

Suppose, in a dataset, we have a feature "season" which has three values – rainy, summer, and winter. Without considering the dummy variable trap, we may name the categorical values which are rainy, summer, and winter as 0, 1, and 2, or any numerical value for that matter. But this will give rise to the multicollinearity problem. To avoid this, we will define just two dummy variables as follows:

Let us name the dummy variable for rainy and summer as S_1 and S_2, respectively.

- If season = rainy, $S_1 = 1$, otherwise 0.
- If season = summer, $S_2 = 1$, otherwise 0.

According to the above conventions, we will easily identify the season values using the dummy variables. It is quite evident from the above-mentioned conventions that when both S_1 and S_2 will be equal to zero, the season can be interpreted as the winter season (Table 5.5).

5.1.2 Logistic Regression

Logistic regression models are statistical models constructed to be applied to a binary-dependent variable. Logistic regression is applied only for binary target variables such as yes or no, true or false, and win or lose. Logistic regression can also be applied to ordinal data such as high-medium-low and good-better-best.

TABLE 5.5

Dummy Variable Trap Example – Solution

Seasons	Rainy (S_1)	Summer (S_2)
Summer	0	1
Winter	0	0
Rainy	1	0

Here, the response (dependent variable) is known as the logit and is calculated using the logit function.

The logit function is nothing but the natural log of the odds.

Odds $= \frac{P}{1-P}$, where P is the probability of the occurrence of the events.

$$\log(\text{odds}) = \text{logit}(P) = \ln\left\{\frac{P}{1-P}\right\}$$

Now, you can go backward and find probability using the odds.

$$P = \frac{\text{odds}}{1+\text{odds}}$$

In logistic regression, the response is not modeled directly as in the linear regression, but in the form of probability of the response belonging to a specific category. Also, as logit is an outcome of the odds and odds being the outcome to the probability, using the sigmoid function, we can conclude that

$$P = \frac{e^{\text{linear regression}}}{1+e^{\text{linear regression}}}$$

As studied in linear regression, $\mathbf{y = \beta_0 + \beta_1 x,}$ we can write the above equation as follows:

$$P = \frac{e^{\beta 0 + \beta 1 x}}{1+e^{\beta 0 + \beta 1 x}}$$

Therefore, we can write the equation of logistic regression as follows:

$$\ln\left\{\frac{P}{1-P}\right\} = \beta_0 + \beta_1 x$$

Logistic regression is of three types:

1. **Binomial Logistic Regression** – In this type of logistic regression, the dependent variable can have just two value types – say "0" or "1" representing "yes" or "no", "true" or "false", "win" or "lose", "malignant" or "benign", etc.

2. **Multinomial Logistic Regression** – Multinomial logistic regression has a dependent variable with more than two outcomes but of unordered type with no specific sequence, e.g., "Alcohol", "Soft Drinks", "Water". (We will study this type of regression in detail in the upcoming section.)
3. **Ordinal Logistic Regression** – Here, the target variable has ordinal valued data. For example, "Good", "Better", and "Best" can be represented as "0", "1", and "3".

As we have studied linear regression and logistic regression, we will go through the major differences between both of them.

1. In linear regression, the target variable is of continuous type, whereas, in logistic regression, the target variable is of discrete type.
2. The main objective of linear regression is to find the best-fit line, whereas the aim of logistic regression is to find the best-fitting S curve.
3. A linear relationship is mandatory in linear regression, whereas it is not required in logistic regression.
4. The linear regression model is assumed to be normally distributed, whereas the logistic regression model is assumed to be binomially distributed.

5.1.3 Multinomial Logistic Regression

The multinomial logistic regression model is a type of classification model, which predicts the probability of more than two categories of the target variable. Multinomial regression predicts the outcome of the nominal type; that is, the categories of the dependent variable cannot be ordered in a significant way. Figure 5.2 represents multinomial logistic regression.

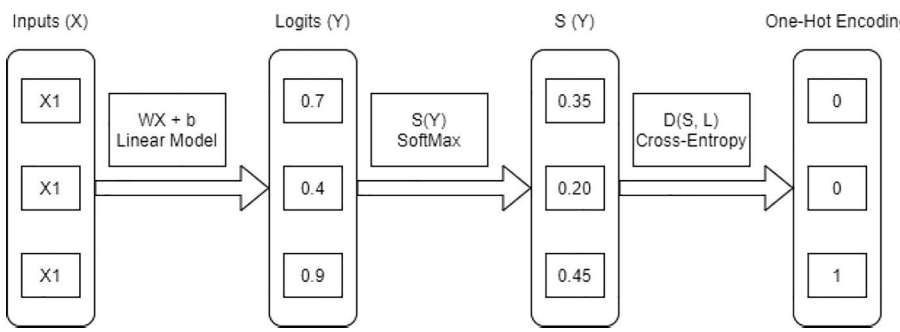


FIGURE 5.2
Multinomial logistic regression.

Multinomial logistic regression stages:

1. Inputs,
2. Linear,
3. Logits,
4. Softmax function,
5. Cross-entropy, and
6. One-hot encoding.

1. Inputs
   The inputs here are the features of the dataset. Say, the goal is to predict the outcome of the type of flower in the Iris dataset, and then, the length and width of the petal, etc., will be considered as the necessary characteristics and are given as inputs to our multinomial logistic regression model. Always remember to convert the categorical valued features to numerical. For example, "very poor", "poor, good", "better", and "best" can be converted to "0", "1", "2", and "3" and then provided to the model.
2. Linear Model
   The linear model equation used here is the same used in linear regression.
3. Logits
   The logits are the output of the linear model. Logits are also known as scores. Logits are proportional to the calculated weights. The change in calculated weights marks a change in the logits.
4. Softmax function

Here, the softmax function is used to generate the probabilities of the given logits found in the earlier stage. The softmax function returns high probability values for the high logits and fewer probability values for the other logits.

   The probabilities calculated using the softmax function are from 0 to 1, and the sum of all the probabilities is equal to 1.
5. Cross-Entropy
   Cross-entropy is used to calculate the difference between the softmax function-generated probability values and the one-hot coding matrix. For the true target category, the distance is less and vice versa.
6. One-Hot coding
   One-hot coding is nothing but a numerical representation for the various target classes of the dependent variables. The one-hot coding matrix takes the value 1 for the correct target class and 0 for all the remaining target classes.

Parameter Optimization
The expected output after training the classifier is called the calculated weights. The weights are used to calculate the loss function, which is the cost function.

Parameter optimization is carried out iteratively until the loss function associated with the calculated weights is less as much as possible.

5.1.4 Time-Series Models

A time series is a set of observations taken by a variable at different times.

Time-series analysis is a technique where various methods and models are taken up with the time-series data to extract meaningful insights and data characteristics. In time-series analysis, three types of data are considered. They are as follows:

1. **Time-Series Data** – A time-series data is a set of observations taken by a variable at different times.
2. **Cross-Sectional Data** – A cross-sectional data is the data observed for multiple parameters at the same time.
3. **Pooled Data** – A pooled data is a mixture of time-series data and cross-sectional data.

Mainly, there are three time-series models, which are as follows:

1. **Autoregressive (AR) Model** – The AR model uses the previously observed values to predict the output.
2. **Integrated (I) Model** – This model follows the differencing of the observation's technique.
3. **Moving Average (MA) Model** – In this model, the outcome variable is linearly dependent on the previous error terms.

A combination of the above-mentioned models gives us various time-series models.

Various time series are highly used in the finance sector. In this chapter, we will study the autoregressive integrated moving average (ARIMA) model.

1. ARIMA Model[7]

   ARIMA model produces the output variable using its calculations on the previous values of the variables.[7] Every model feature in the ARIMA (AR, I, MA) contributes to building a model to find the output that fits the model well.

   Three parameters are used for the ARIMA model:

   - **p** – The number of lag time orders.
   - **q** – The moving average order term.
   - **d** – The number of times the previous values have been subtracted.

The nonseasonal ARIMA is denoted with the above three parameters, ARIMA (p, d, q), whereas the seasonal ARIMA (SARIMA) reveals the seasonality. It is represented similarly to the ARIMA model but with a little modification.

SARIMA (p, d, q) $(P, D, Q)_m$

where p – Autoregressive order term.

q – Moving average order term.

d – Integrated order term.

(P, D, Q) – It is the AR, I, MA order in terms of the seasons.

m – Number of periods in every season.

We will go through the ARCH and GARCH models in the upcoming chapter.

5.2 Machine Learning

As we all know that data science is a field that also gives important insights from huge chunks of data, in the upcoming future, automation will be replacing all human tasks. To get maximum efficiency in achieving this, it is important to use smart devices. It is but obvious that artificial intelligence (AI) has to be used to reach the goal, but most importantly, we should remember that ML is at the heart of AI. A data scientist is a professional who predicts future actions based on previous patterns. Hence, data scientists must be thorough with various ML algorithms that will help them to generate machines to make better decisions with minimal human intervention.

5.2.1 Decision Trees

A decision tree is a predictive modeling technique that comes under supervised algorithms of ML.[8] A decision tree can be considered as a flowchart-type structure that uses various ways and conditions to split a dataset. The decision tree is built on two entities, the decision node also called the internal node, where the dataset is split based on conditions, and the leaf node where the result is achieved.

This algorithm can perform both regression and classification problems. The difference between the decision tree in regression and the decision tree in classification is that the decision variable is of continuous type in the former and categorical type in the latter.

Attribute Selection Measures:

When constructing the decision tree, the attribute selection measure is used to choose the best attribute for the tree nodes. By this measure, we can easily select the best attribute for the tree nodes. The two popular ways for attribute selection measure are as follows:

1. Gini Index

   The Gini index is an attribute selection measure to determine the probability of a randomly selected variable being incorrectly recognized. An attribute with a comparatively lower Gini index is preferred. A Classification and Regression Tree (CART) algorithm uses the Gini index as it splits the dataset only in the binary format.

   The formula to find the Gini index is as follows:

   $$\text{Gini index} = 1 - \sum_j P_j^2$$

2. Information Gain

   Information gain is built on the drop in entropy after the dataset is split into an attribute. The attribute with the highest information gain is considered to be split first.

   Entropy – Entropy refers to the uncertainty in a set of examples.

   $$\text{Entropy}(X) = \sum \frac{P_i N_i}{P+N} \times \text{I.G}(P_i N_i)$$

   $$\text{Information gain} = \frac{-P}{P+N}\log_2\left(\frac{P}{P+N}\right) - \frac{-N}{P+N}\log_2\left(\frac{N}{P+N}\right)$$

   $$\text{Gain} = \text{I.G} - E(X)$$

Pruning

Pruning is a technique that removes the unimportant nodes of the tree, thus decreasing its size without changing its accuracy.

One decision tree algorithm is the ID3 algorithm. ID3 is an acronym for Iterative Dichotomiser 3. ID3 is a classification algorithm that uses information gain as an attribute selection measure, and it constructs a decision tree by selecting the best attribute having maximum information gain.

Steps to build a decision tree:

1. Assign the complete dataset, say D as the root node.
2. By utilizing the attribute selection measures, select the prime attribute of the dataset.
3. Once you have the prime attribute, divide the dataset D into subsets having values of the prime attribute.
4. Create a decision node in such a way that it has the prime attribute.
5. Repeat this procedure for all the subsets, until a point where the decision tree cannot be further divided.

Advantages of the decision tree:

- A decision tree is simple to understand.
- A decision tree works well with unnormalized data.

Disadvantages of the decision tree:

- Even a minor change in the data can affect the entire structure of the tree, thus causing instability.
- For a larger dataset, a decision tree is time-consuming and can get complex.

5.2.2 Naïve Bayes

Naïve Bayes algorithm is a classification algorithm that focuses on the implementation of Bayes theorem, assuming that all the features of the same class are independent of one another. As the base of the naïve Bayes classifier is the Bayes theorem, we will first go through the Bayes theorem.

Bayes Theorem

Bayes theorem finds out the posterior probabilities. A posterior probability is a probability of an event occurring given some another event (feature) has already been observed.

Mathematical Representation of the Bayes Theorem[9]:

$$P(M|N)^9 = \frac{P(N|M) \times P(M)}{P(N)}$$

- P(N|M) is the posterior probability of M.
- P(N) is known as evidence.
- P(M) is known as the prior probability of M.

The two assumptions that are made in the naïve Bayes algorithm in accordance with the Bayes theorem are that the features are independent of each other and are equally responsible for the result.

Suppose we consider m as the target variable of a class and variable N as the various features.

$N = (n_1, n_2, n_3, n_4, \ldots n_n)$

The updated Bayes theorem including the two assumptions will be as follows:

$$P(m|n_1, n_2, n_3, \ldots n_n) = \frac{P(n_1|m) \times P(n_2|m) \ldots P(n_n|m) \times P(m)}{P(n_1)P(n_2)P(n_3)\ldots P(n_n)}$$

In the above equation, as the denominator is constant for any input variable, removing the denominator term will give us a proportionality.

$$P(m|n_1, n_2, n_3, \ldots n_n) \alpha P(m)\ \Pi_{i=1}^{n} P(n_i|m)$$

A dataset can be multivariate and so the classification. In such cases, it is mandatory to find the target variable yielding maximum probability. This can be achieved using the below-mentioned formula:

$$M = \text{argmax}_m P(m) \Pi_{i=1}^{n} P(n_i \mid m)$$

There are three types of naïve Bayes classifiers. They are as follows:

- **Bernoulli Naïve Bayes –** Bernoulli naïve Bayes is the same as the multinomial model, just the difference is that the features take Boolean values, such as yes or no and success or failure.
- **Multinomial Naïve Bayes –** Multinomial naïve Bayes is majorly used for document classification. Multinomial distribution is used by this model.[10]
- **Gaussian Naïve Bayes –** When the features consist of continuous values rather than discrete values, it is assumed that the Gaussian distribution (also known as the normal distribution) is used.

Advantages of Naïve Bayes Classifiers:

- It is simple and easy to understand.
- The testing data can be evaluated efficiently on small training data.

Disadvantages of Naïve Bayes Classifiers:

- The assumption of independent features is rare to find in real life.
- The zero-frequency problem. This problem arises when the model encounters a categorical variable in test data not earlier examined in the training dataset. In such circumstances, the model sets the probability of such a variable to zero.

5.2.3 Support Vector Machines

Support vector machine (SVM) is a supervised learning algorithm that is highly used for classification but can also be used for regression. The idea behind the SVM algorithm is to place all the data points of the dataset in multidimensional data space (of the size of the number of features in the dataset) and then explore the best hyperplane that differentiates the classes efficiently. The main motive of SVM is to find the ideal hyperplane to yield greater accuracy.

Important terms in SVM:

- **Support Vectors** – The data points that are of the closest proximity to the hyperplane.
- **Hyperplane** – It is the decision plane that divides the two classes of different objects.

- **Margin** – It can be defined as the distance between the two lines at the data points of the different groups.

SVM can handle linearly separable data easily. Linearly separable data is the data that can be easily plotted on a graph and can be differentiated using a line.

1. Figure 5.3 illustrates linearly separable data.
2. Figure 5.4 represents nonlinearly separable data.

Practically, to separate nonlinearly separable data, kernel functions are used. The kernels take the input data and convert it into linearly separable data by adding more dimensions to it. Various SVM algorithms use different kernel functions according to its need.

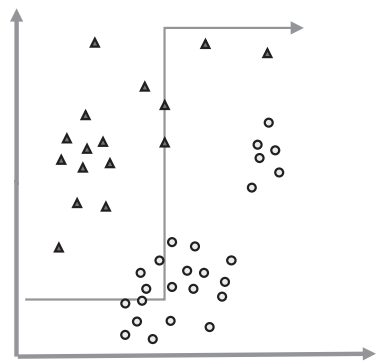


FIGURE 5.3
Linearly separable data.

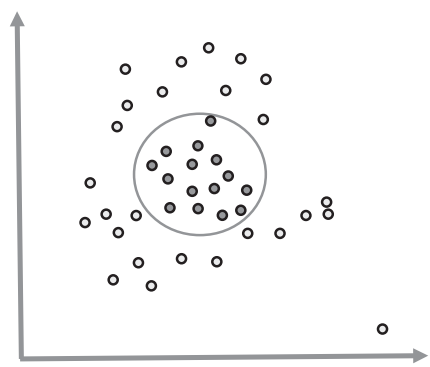


FIGURE 5.4
Nonlinearly separable data.

The kernel functions are as follows:

1. Linear kernel,
2. Polynomial kernel,
3. Gaussian kernel,
4. Radial basis function (RBF) kernel,
5. Laplace RBF kernel,
6. Hyperbolic tangent kernel,
7. Sigmoid kernel,
8. Bessel function of the first kind kernel, and
9. ANOVA RBF kernel.

Advantages of SVM:

- It can handle nonlinearly separable data well using kernel functions.
- SVM models are highly stable.

Disadvantages of SVM:

- Kernelized SVM is complex and difficult.
- SVM is time-consuming when it comes to larger datasets.

5.2.4 Nearest Neighbor learning

Nearest neighbor methods are used to differentiate a new data point on the basis of its similarity with the existing data points. These nearest neighbor methods can be used for classification as well as regression. The nearest neighbor learning model is applied as a classifier for discrete types of labels and as a regressor for continuous types of labels. There are various types of nearest neighbor algorithms such as the brute-force approach, KD tree, k-NN, and ball tree. But the one that is highly used in data science and has proven to be effectively modeled is the k-NN algorithm.

The k-NN algorithm is a simple ML algorithm that is used for regression as well as classification problems. The prediction for an unknown data instance is found by searching for the most similar k instances in the training dataset and returning the most similar data instance. The k-NN algorithm does not assume any facts on the basis of the primary data, and hence, it is also known as the non-parametric algorithm. Also, the k-NN algorithm uses the entire dataset to train the model as a classifier, and hence, it is also called the lazy learning algorithm.

k-NN Algorithm Implementation:

1. Select the value of k, i.e., the closest data points.
2. Repeat the below process for all the new data instances:

i. Calculate the Euclidean distance between the new data instance and the training data points.
ii. Sort the distance parameters in ascending order.
iii. Select the first k parameters (the k-nearest data points).
iv. Assign the class that is most frequent in the k rows.

3. End.

5.2.5 Clustering

Clustering is a form of unsupervised learning. In unsupervised learning, the features of a dataset are modeled without any label references. Clustering is the process of dividing the data points into many clusters in such a way that data points in the same clusters are more similar to other data points in the same cluster and are different from data points in other clusters.[11] It is generally a compilation of objects in terms of similarity and dissimilarity among them.

Clustering can be of two types:

1. **Hard Clustering** – In hard clustering, every data point is specifically assigned a cluster or is not assigned. For example, in the breast cancer dataset, the person will be placed in the malignant cluster or the benign cluster. The person (data points) will be in either of the clusters.
2. **Soft Clustering** – Soft clustering is when there is uncertainty in the assignment of the cluster. In this type of clustering, probabilities are assigned to data points for a particular cluster.

Types of clustering algorithms:
Various clustering algorithms are classified based on their own rules to define the similarity between the data points.

1. **Centroid Models** – These types of clustering algorithms define the similarity in accordance with the closeness of the data point to the center of the cluster. One of the most popular clustering algorithms of this type is k-means clustering. In these models, it is necessary to know the number of clusters to be defined.
2. **Connectivity Models** – Connectivity models are the models based on the premise that the data points closer to one another in the dataset are said to be similar to each other. These models can be built in two ways, one being that all the data points are placed in different clusters and then grouped according to the decrease in the distances. Another approach is that all the data points are placed in a single cluster and then segregated as the distance increases. Hierarchical clustering is an example of connectivity-based models.

3. **Distribution-Based Models** – These models assume data points to be clustered in the various distributions such as the normal distribution, Gaussian distribution, and expectation–maximization algorithm.
4. **Density-Based Models** – These models connect the high-density areas in the data space into clusters. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm comes under this category.

The topmost used clustering algorithms in data science are as follows:

1. k-Means clustering,
2. Agglomerative hierarchical clustering,
3. DBSCAN clustering, and
4. Expectation–maximization clustering.

We have covered k-means clustering, agglomerative hierarchical clustering, and DBSCAN clustering in the upcoming chapter. In this chapter, we will go through the expectation–maximization clustering.

1. Expectation–maximization clustering

   The expectation–maximization algorithm is an algorithm that is used to find the correct parameters for the model. This algorithm is highly used when the data used for building the model is not complete, which means it has various missing values. These missing values thus make it more difficult in finding the optimal parameters for the model. In the data science field, the missing values are also called latent variables. As the latent variable values are unknown in the first place, the expectation–maximization algorithm comes into the picture. This algorithm makes use of the currently provided data to find the best-fitting values for the latent variables, and once that is done, it then finds the model framework. This model framework is then used to modify the latent variable values, and this goes on until the connection with one another.

Algorithm:

1. Give incomplete data to the model.
2. Predict the values for the latent variables using the current incomplete data. This step is also known as the expectation step or simply E step.[12]
3. Update the model parameters using the incomplete data updated with the values generated for the latent variables in the previous step (E step).
4. Reiterate steps 2 and 3 until an optimal solution is obtained.

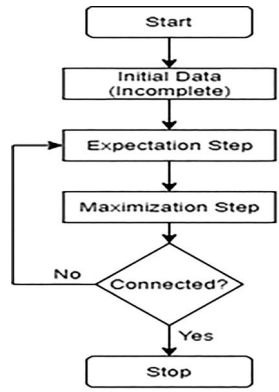


FIGURE 5.5
Expectation–maximization algorithm.

Flowchart: Figure 5.5 represents the flowchart of expectation–maximization algorithm.

5.2.6 Confusion Matrix

The confusion matrix is an approach to calculate the performance of the classifier. It is nothing but an n × n matrix, where the value of n changes as per the number of the feature classes. It examines the actual values and the predicted values predicted by a model and gives us a thorough view of the performance and the errors made by our classifier.

Let us study a binary classifier with a 2 × 2 matrix, as represented in Figure 5.6.

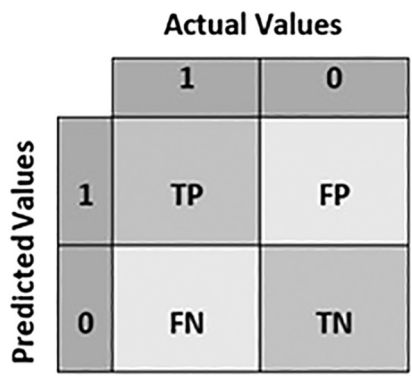


FIGURE 5.6
Confusion matrix.

Terms used in the confusion matrix are as follows:

1. **True Positive (TP) –** True positive is the case when the actual and the predicted values are the same, which means that the model has rightly predicted the positive values.
2. **True Negative (TN) –** False negative is the case when the actual and the predicted values are the same, which means that the model has rightly predicted the negative values.
3. **False Positive (FP) –** False positive is the case when the classifier incorrectly predicts an actual negative value to be a positive value.
4. **False Negative (FN) –** False negative is the case when the classifier incorrectly predicts an actual positive value to be a negative value.

Various performance measures can be deduced using the confusion matrix. They are as follows:

1. **Accuracy –** Accuracy can be defined as the measure to calculate the number of right predictions that our classifier has made using the actual and the predicted values.
   Accuracy is mathematically represented as follows:
   $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
2. **Precision –** Precision is the measure that evaluates the number of correctly predicted values that were positive. The reliability of the model can be deduced using precision.
   $$\text{Precision} = \frac{TP}{TP + FP}$$
3. **Recall/Sensitivity/TPR** – Recall is the measure used to evaluate the number of actual positive values that our classifier was able to predict correctly among the predicted positive values. The recall is also known as the sensitivity of the model.
   $$\text{Recall} = \frac{TP}{TP + FN}$$
4. **Specificity (TNR)** – Specificity is also called the true-negative rate.[13] It is the measure to calculate the number of rightly made negative predictions among the actual negative values.
   $$\text{Specificity} = \frac{TN}{TN + FP}$$
5. **F1 Score –** F1 score is a metric that uses both precision and recall. It is the weighted average of both. F1 score can be calculated considering the harmonic mean of precision and recall.

$$\text{F1 score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

$$= \frac{2\ (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

6. **ROC Curve** – ROC is an acronym for receiver operator characteristics. It is a graph plotted with the sensitivity (true-positive rate) and the 1-specificity (false-positive rate). This curve helps us distinguish between the positive and the negative values. The higher the area under the curve (AUC), the models get more efficient in distinguishing between the positive and the negative classes.

You will get the AUC equal to 1 when the model correctly identifies all the positive and negative values as represented in Figure 5.7.

However, when the AUC = 0, one can conclude that the classifier has incorrectly predicted all the values, i.e., all the negative values as positive values and vice versa.

When the AUC is between 0.5 and 1, it specifies that the classifier is successful in identifying a greater number of the true positives and true negatives than the false positives and false negatives as represented in Figure 5.8.

When the AUC is 0.5, it means that the classifier is randomly predicting the classes to be positive or negative classes as represented in Figure 5.9.

The two most important terms in the confusion matrix are the type of errors. The type of errors in the confusion matrix can be broadly classified into two types.

1. Type I error and
2. Type II error.

FIGURE 5.7
ROC curve (AUC = 1).

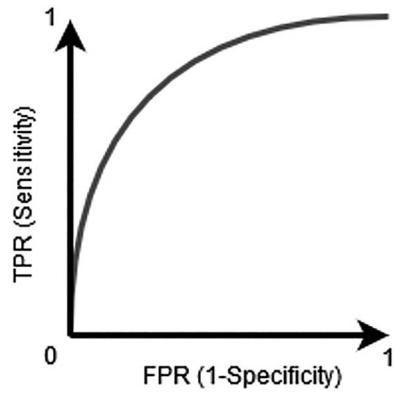


FIGURE 5.8
ROC curve (AUC between 0.5 and 1).

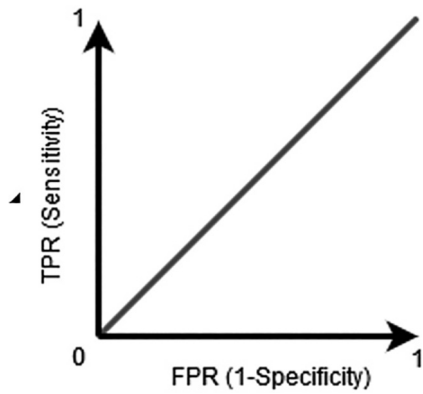


FIGURE 5.9
ROC curve (AUC = 0.5).

1. Type I Error

   Type I error occurs in the false-positive case. Type I error is the error when the model predicts the class to be positive when it is negative, i.e., false.

   For example, our model predicted a man to have died, where he is not dead (alive).

2. Type II Error

   Type II error occurs in the false-negative case. Type II error is the error when the model predicts the class to be negative when it is positive, i.e., true.

   For example, our model predicted a man to be not alive when he is alive.

5.3 Summary

Regression analysis plays a very important role in data analytics. In this chapter, we first studied the two basic types of regressions, that is, linear regression and logistic regression. We have shown how the equations are deduced for the linear regression and logistic regression. Further on, we also studied multinomial logistic regression; regression with more than one discrete outcome is generated. We have also tried to give you a brief introduction to the time-series models in data analytics. After covering the regression analysis section of the chapter, we then studied different ML algorithms that are highly used in data science such as the decision tree, naïve Bayes, SVMs, nearest neighbor learning, and clustering. To conclude the second section that is ML, we got a brief of the confusion matrix, which is widely used to calculate the performance of the classifier.

Exercise

1. Calculate the regression coefficients for the following data. Also, guide through the coefficient of determination (R^2).

X	4	5	6	7	8	9
Y	17	18	19	20	21	22

2. Differentiate between linear regression and logistic regression.
3. Define the terms used in the logistic regression.
4. Explain the importance of removing the outliers in clustering.
5. On which assumptions are the Bayes theorem updated to naïve Bayes?
6. What is the difference between precision and recall?
7. Define type I and type II errors with an example.

References

1. Kalra Saloni, Lamba Rishab, Sharma Manoj. 2020. Machine learning based analysis for relation between global temperature and concentrations of greenhouse gases. *Journal of Information and Optimization Sciences* 41(1): 73–84. https://doi.org/10.1080/02522667.2020.1715559.
2. Choudhary Yasir. 2017. Linear regression implementation in Python. Yasir's Blog. http://yasirchoudhary.blogspot.com/2017/ (accessed 20 September 2020).

3. Yan Xin, Su Xiao Gang. 2009. *Linear Regression Analysis*. Singapore: World Scientific Publishing.
4. Linear regression example. Regression. Stat Trek. https://stattrek.com/regression/regression-example.aspx (accessed 12 September 2020).
5. Dwivedi Divyansh. 2018. Machine learning for beginners. https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab (accessed 25 September 2020).
6. Frost Jim. Regression tutorial with analysis examples. Statistics by Jim. https://statisticsbyjim.com/regression/regression-tutorial-analysis-examples/ (accessed 18 September 2020).
7. Bhargavi K. 2018. USD exchange rate forecasting using ARIMA, MLP and recurrent neural networks. *International Journal of Engineering Research & Technology (IJERT) NCRTS* 27: 3.
8. Goyal Kechit. 2021. 6 types of supervised learning you must know about in 2021. Artificial Intelligence. upGrad blogs. https://www.upgrad.com/blog/types-of-supervised-learning/ (accessed 20 January 2021).
9. Ardhapure Omkar, Patil Gayatri, Udani Disha, Jetha Kamlesh. 2016. Comparative study of classification algorithm for text based categorization. *International Journal of Research in Engineering and Technology* 05: 217–220.
10. Saxena Rahul. 2017. How the Naive Bayes classifier works in machine learning. https://dataaspirant.com/naive-bayes-classifier-machine-learning/ (accessed 15 September 2020).
11. The Anh Pham, Thang Nguyen Duc, Vinh La The, Lee Young-Koo, Lee Sungyoung. 2013. Deflation-based power iteration clustering. *Applied Intelligence* 39: 367–385. https://doi.org/10.1007/s10489-012-0418-0.
12. Han Mei, Zhao Yong, Li Gaoming, Reynolds Albert C. 2011. Application of EM algorithms for seismic facices classification. *Computational Geosciences* 15: 421–429. https://doi.org/10.1007/s10596-010-9212-4.
13. Scutari Marco. 2015. Statistical models - Log-linear models. University of Oxford. https://www.bnlearn.com/about/teaching/msc-loglinearmodels.pdf (accessed 25 September 2020).

6

Data Analytics and Text Mining

6.1 Text Mining

Text data is ubiquitous and growing rapidly. Text data is also very important because it contains knowledge about users, especially preferences and opinions; text mining is a rapidly changing and constantly developing field of study and research. The volume of social media, blogs, websites, articles, and other text data is growing rapidly. There is a need for optimal analysis, extracting knowledge and insight, and categorization of meaningful text content. Following is the text mining process[1]:

1. Text Transformation
   Text transformation is a method used to monitor and control the capitalization of the text. The two main ways of document representation are bag of words and vector space.
2. Data Preprocessing
   Data preprocessing is used in the field of text mining to derive valuable information and knowledge from unstructured text data.
3. Feature selection
   Feature selection is a significant part of data mining. Feature selection can be defined as the process of reducing the input of processing or finding essential information sources.[2] The feature selection is also called variable selection.
4. Data mining
   Furthermore, the data mining process combines with the conventional process.
5. Evaluation
   Finally, the result is evaluated. The diagram of the data mining process is given in Figure 6.1.[3]

DOI: 10.1201/9780429443237-8

FIGURE 6.1
Text mining process.

Difference between data mining and text mining:

Data Mining	Text Mining
Data mining is a statistical technique for processing raw data in a structured form.	Text mining is a part of data mining that deals with words, phrases, and sentences.
It has similar systems across the world.	It has a variety of systems, and it has many dialects and languages.
Data is precise and accurate.	Data can be ambiguous, and the sentiment may be unrelated to the words.
Issues: Issues with missing values, outliers, etc.	Issues: Spelling errors. Differing values of proper nouns such as names. Varying quality of language translation.
Preexisting databases and spreadsheets are used to collect information.	The text is used to collect high-quality information.

6.1.1 Major Text Mining Areas

Figure 6.2 represents text mining areas.

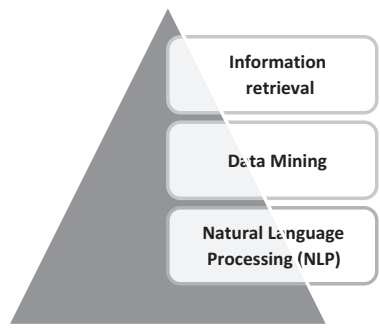


FIGURE 6.2
Text mining areas.

6.1.1.1 Information Retrieval

Information retrieval is an automatic process that extracts organized data, important words, attributes, and relationships between entities from loosely organized data and unstructured data. Typically, it helps in converting the unstructured text into a structured form and obtains important information.[4]

6.1.1.2 Data Mining

Data mining refers to finding out the hidden patterns from the extracted data. Data mining tools can forecast future patterns and behaviors by finding unknown patterns. It is useful to address business questions and deal with traditionally time-consuming problems.[3]

6.1.1.3 Natural Language Processing (NLP)

NLP is a subfield of artificial intelligence. It deals with the study of human language. The entire aim of the NLP is to read, decode, comprehend, understand, and make sense of the human languages in a useful manner. The development of NLP applications is difficult because of the complexity in the interaction between humans and computers.[5]

Case Study: Text Mining Twitter Data

In this, we are going to look at how various terms are related, i.e., how well words are connected and what kinds of words are important. We will gain the idea of what topics people are discussing and what is the trending topic, and we can also analyze different sentiments.

Following are the R packages:

Package tm: A framework for text mining applications.

Package igraph: Network analysis packages of R.

Step 1: Text Cleaning

We will call the two packages *tm* and *igraph* and load the Twitter data and the rest of the steps will perform text cleaning.

```
library(tm)
library(igraph)

twitter_data = read.table(file="results_olympics.csv", header=TRUE,sep=",",encoding="UTF-8", stringsAsFactors = FALSE)
# We may need to remove the first charactor 'b' from the string.
twitter_data$text = paste(substr(twitter_data$text,2,nchar(twitter_data$text)))

text = twitter_data$text
text_clean = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", text) # remove retweet entities
text_clean = gsub("@\\w+", "", text_clean) # remove @People
text_clean = gsub("[[:punct:]]", "", text_clean) # remove puctuation symbols
text_clean = gsub("[[:digit:]]", "", text_clean) # remove numbers
text_clean = gsub("http\\w+", "", text_clean) # remove links
```

Step 2: Creation of Corpus and Term-Document Matrix

Further, we create a *corpus*, which is a set of words in the dataset. And, then, we need to create a term-document matrix. This is to indicate which term or word is appearing in which document and how often.

```
text_corpus = Corpus(VectorSource(text_clean)) # corpus
text_corpus = tm_map(text_corpus, tolower) # convert to lower case
text_corpus = tm_map(text_corpus, removeWords, c(stopwords("english"), "olympics")) # remove stopwords
text_corpus = tm_map(text_corpus, stripWhitespace) # remove extra white spaces
text_corpus = tm_map(text_corpus, PlainTextDocument)

tdm = TermDocumentMatrix(text_corpus) # term-document matrix
m = as.matrix(tdm) # convert to a matrix
```

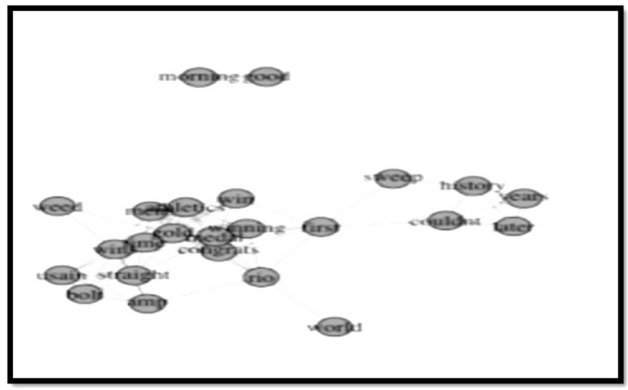
Step 3: Popular Words

There are n number of words, and some of the words are not useful. Stop words are the most commonly used words such as *a, an, the, is are, and she.* Stop words are first filtered.

```
# We choose popular words (word frequency > 90% percentile)
wf = rowSums(m)
m1 = m[wf>quantile(wf,probs=0.9), ]

# remove columns with all zeros
m1 = m1[,colSums(m1)!=0]

# change it to a Boolean matrix
m1[m1 > 1] = 1

# transform into a term-term adjacency matrix
termMatrix = m1 %*% t(m1)
```

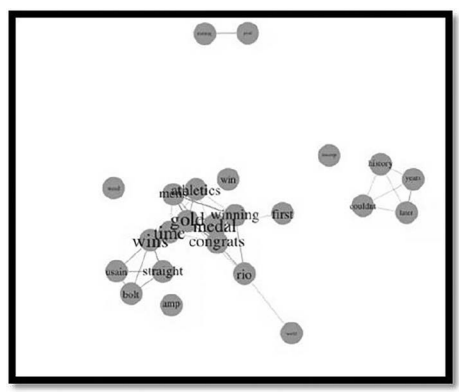
Step 4: Build and Plot Graph

```
# build a graph from the above matrix
g <- graph.adjacency(termMatrix, weighted = T, mode = "undirected")

# remove loops
g <- simplify(g)

# set labels and degrees of vertices
V(g)$label <- V(g)$name
V(g)$degree <- degree(g)

# plot a graph
set.seed(3535)
layout1 <- layout.fruchterman.reingold(g)
plot(g, layout=layout1)
```



The better graph: The revised graph makes important terms stand out.

```
V(g)$label.cex <- 1.2 * V(g)$degree / max(V(g)$degree) + 0.2
V(g)$label.color <- rgb(0.0, 0.0, 0.2, 0.8)
V(g)$frame.color <- NA
egam <- (log(E(g)$weight) + 0.3) / max(log(E(g)$weight) + 0.3)
E(g)$color <- rgb(0.5, 0.5, 0.0, egam)
E(g)$width <- egam

# plot the graph in layout1
plot(g, layout=layout1)
```



CONCLUSION

What we have done here is a social network study, but what we have just seen here are words in a tweet. This method can be extended to all sorts of social network analysis. You could have users, nodes on Twitter, Facebook, and other things. We can use any term in term-document matrix. Using this method and working on many steps further we can do any type of analysis by using any data.[6,7]

Spam Email Classification:

The NB classifier is a probabilistic learning algorithm that derives from Bayesian decision theory. The probability of a message d being in class c, P(c|d), is calculated as follows[8]:

$$P(c|d) \propto P(c) \prod_{k=1}^{m} P(t_k|c)$$

P(tk|c) – the conditional probability of feature tk occurring in a message of class c, which can be used to measure how much evidence tk contributes that c is the correct class.

P(c) – prior probability of a message occurring in class c.

In email classification, the class of a message is determined by finding maximum a posteriori class (c_{MAP}) defined by the following:

$$c_{MAP} = \arg \max_{c \in \{c_1, c_s\}} P(c|d) = \arg \max_{c \in \{c_1, c_s\}} P(c) \prod_{k=1}^{m} P(t_k|c)$$

The multiplication of probabilities is often converted to the addition of logarithms of probabilities, and, therefore, the maximization of the equation is alternatively performed by the following:

$$c_{MAP} = \arg \max_{c \in \{c_1, c_s\}} \left[\log P(c) + \sum_{k=1}^{m} \log P(t_k|c) \right]$$

All model parameters, i.e., class priors and feature probability distributions, can be estimated with relative frequencies from the training set.[9] When a given class and message feature do not occur together in the training set, the corresponding frequency-based probability estimate will be zero, which would make the right-hand side of the above equation undefined.[9] This problem can be mitigated by incorporating some corrections such as Laplace smoothing in all probability estimates.[9]

Apart from this, many other algorithms such as LogitBoost (LB), support vector machine (SVM), augmented latent semantic indexing space model (LSI), and radial basis function (RBF) networks are used.[10]

6.2 Text Analytics

Text analysis refers to the representation, processing, analyzing, and modeling of a huge amount of text data to provide valuable insights.[11] Text mining – the method of determining the relationship and interesting patterns in large text collections – is an essential component of text analysis.[11] Text analysis is the automated process of understanding and sorting unstructured text data with machine learning to mine for valuable insights. This technique helps us to automatically extract, classify, and identify text data, such as comments, queries, messages, requests, tweets, emails, product reviews, and survey responses. Table 6.1 provides some sample data sources and data formats that text analysis would have to deal with.

6.2.1 Text Analysis Subtasks

6.2.1.1 Cleaning and Parsing

Before you can do a text analysis project, you also need to do a lot of text cleaning and parsing. That is because much of the text is formed and stored so that people can understand it, and it is not always easy for a machine to process that text. Computers work well and are familiar with the data when there is a structure to a data source or, at least, some regular patterns that it can identify. Most cleaning and parsing for text analysis involves increasing the regularity (e.g., fixing typos) or adding structure (tagging certain words as important, or even splitting documents up into different sections that have special meaning – title, authors, chapters, etc.). Typically, parsing is a process that uses unstructured text and creates a structure for further analysis. Unstructured text can be a simple text file, a blog article, an extensible markup language (XML) file, or a hypertext markup language (HTML) file. Parsing deconstructs and works out on the text given and makes it in a more organized way for subsequent steps.

TABLE 6.1

Data Source

Data Source	Data Structure Type	Data Format
Email	Unstructured	TXT, MSG, IMG
Web pages	Semi-structured	HTML, IMG
Server logs	Semi-structured or quasi-structured	LOG or TXT
News articles	Unstructured	TXT, HTML
Literature	Unstructured	TXT, DOC, HTML, or PDF
Social network API firehouses	Semi-structured	XML, JSON, or RSS

6.2.1.2 Searching and Retrieval

This procedure is originated from library science. It is the identification of documents in a corpus that contains key terms such as words, phrases, topics, and entities.

6.2.1.3 Text Mining

Text mining is sometimes referred to as text data mining. Text data mining is the process of transforming unstructured text data into a structured format. It identifies meaningful patterns and new insights from the texts. Text mining uses natural language processing to transform the unstructured text to structured which is suitable for machine learning algorithms.[12]

6.2.1.4 Part-of-Speech Tagging

Part-of-speech (PoS) tagging can be known as the process of assigning the parts of speech to the selected word. Simply put, it can be said that PoS labeling is the function of labeling every word in a sentence with its appropriate part of speech. Let us see an example of PoS tagging mentioned in Table 6.2.

6.2.1.5 Stemming

Stemming is considered an important process. It strips the words and extracts the base form of that particular word. This step is done by removing the affixes of the word. For example, the stem of the word "sleeping" and "sleep's" is "sleep".

The list of various stemming algorithms is given in Figure 6.3.

TABLE 6.2

Example of PoS

Noun	Verb	Preposition	Noun
Jane	went	to	School

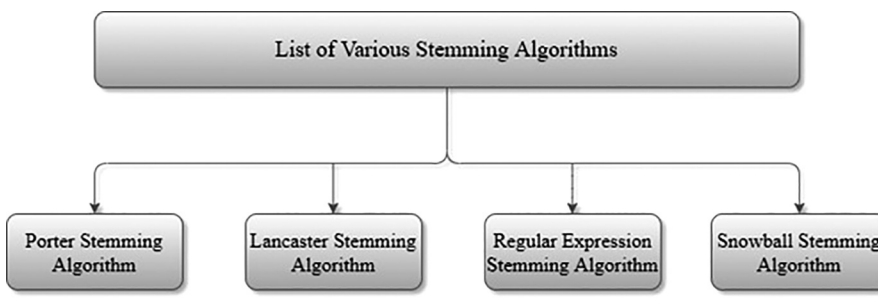


FIGURE 6.3
Stemming algorithms.

```
# import these modules
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("plays :", lemmatizer.lemmatize("plays"))
print("jumping :", lemmatizer.lemmatize("jumping"))
```

FIGURE 6.4
Example of lemmatization.

6.2.1.6 Lemmatization

The lemmatization technique is similar to stemming. It finds the root word of the given word. The output of this technique is called "lemma". In short, it gives the base form of the given word. The example of lemmatization is given in Figure 6.4.

Output:

```
plays : play
jumping : jump
```

6.2.2 Basic Text Analysis Steps

- **Collection of Raw Data**
  Raw data is referred to as primary data. It is the starting point of text analysis. The data may be comments and reviews on websites, news portals, blogs, articles, server logs, etc. This data is continuously monitored by the companies.
- **Data Representation**
  In this phase, the text data is transformed into a better-organized form using the text normalization techniques. Tokenization

is one of the text normalization techniques where the words are separated from the actual body of the text. The text is converted into a collection of tokens, where tokens are words, numbers, and punctuations.

For example, "Sam went to the park".

After tokenization: {Sam, went, to, the, park}

- **Document grouping**

  Document grouping can be achieved by clustering or classification methods. One of the best ways is topic modeling. Topic modeling offers tools to automatically organize, scan, understand, and summarize vast quantities of information. Topic models are statistical models that analyze words from a collection of documents, evaluate themes over the text, and discover how the themes are related or modified over time.

- **Determining Sentiments**

  Sentiment analysis used various algorithms to determine the opinions, which is required to obtain subjective information from the text. The sentiments can be classified as negative and positive sentiments.

- **Gaining Insights**

  This is a visual representation of the opinions. For example, the word cloud removes all the stop words and displays only important words so that data scientists can see the opinions.

6.3 Introduction to Natural Language Processing

NLP is a form of artificial intelligence that helps the machine to read, decode, comprehend, and understand the text by simulating the human ability to understand the language.[13] NLP techniques utilize a range of approaches, including linguistics, semantics, statistics, and machine learning, to extract entities, relationships, and to understand the meaning, allowing an understanding of what is being said or written.[13] Instead of just understanding single words or a set of words, NLP lets computers understand whole sentences as they are spoken or written by a human.[13] It uses many methodologies to decipher ambiguities in language, including automatic summarization, PoS tagging, disambiguation, entity extraction, and relation extraction, as well as disambiguation and natural language understanding and recognition.[13]

6.3.1 Major Components of NLP

a. Natural Language Understanding (NLU):
In this phase, the speech input is transformed into useful representations to evaluate different aspects of the language. Although the natural language is so rich in forms and structures, it is also really complex and ambiguous (Figure 6.5).

- There may be numerous forms of ambiguity, which are mentioned below such as lexical ambiguity, which is quite simple, i.e., word-level ambiguity. For example, a "present" could be a noun or a verb.
- Then, there can be syntactical ambiguity, which occurs when a sentence is parsed in different ways. For example, in the sentence "The professor saw the boy with the telescope", it is ambiguous whether the professor saw the boy carrying a telescope or he saw him through his telescope.
- Furthermore, there is a referential ambiguity. Let us discuss this with an example, "John went to meet Sam". He said, "I am tired". Now, who exactly is tired is not made clear in this sentence.

b. Natural Language Generation (NLG):
For the generation of the output text, the intermediate representation requires to be converted back to the natural language format. Thus, there are several subprocesses involved in this process.

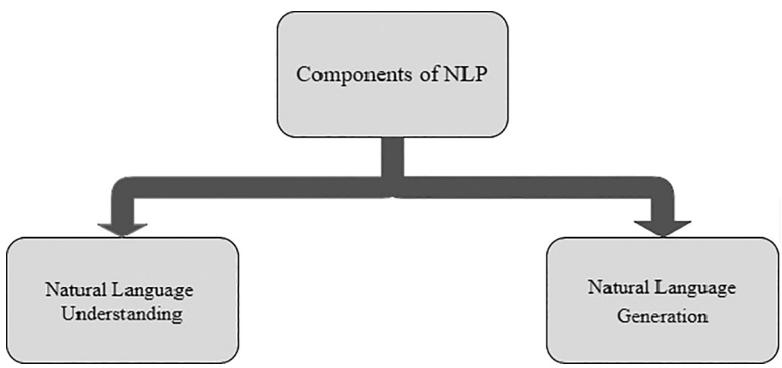


FIGURE 6.5
Major components of NLP.

They are as follows:

- The text planning phase includes extracting relevant and important content from the knowledge base.
- Sentence planning involves selecting correct words, forming meaningful sentence with language grammar, and setting the tone for the same.
- Text realization is a process of mapping the planned sentence into a structure.

6.3.2 Stages of NLP

1. Lexical Analysis
   Lexicon is the words and phrases in the language. The lexicon analysis phase deals with the recognition and identification of the structure and formation of the sentences. It divides the paragraphs into sentences, phrases, and words (Figure 6.6).
2. Syntactic Analysis
   In this phase, the grammar of the sentence is examined to get the relationships among different words in the sentence. The sentence is parsed as nouns, verbs, adjectives, and another part of a sentence.
3. Semantic Analysis
   In the semantic analysis, the actual true meaning of the sentence is extracted from the structure and the words used. It checks whether the sentence is meaningful. It maps the object with its syntactic structure to decide the correctness and accuracy of the sentence. For example, "cold fire" will be termed a wrong sentence.
4. Discourse Integration
   The meaning of discourse for NLP is nothing but the context of the sentence or a word. Generally, the meaning of a sentence majorly depends on or changes according to the context, i.e., the sentence before and after that given sentence. In disclosure integration, the meaning of a sentence gets verified with the sentence before it.
5. Pragmatic Analysis
   This phase deals with the meaning of the sentence in various situations. In the pragmatic analysis, the sentences are reinterpreted to verify the correctness of the meaning in the given context. Real-world knowledge is required.

FIGURE 6.6
Stages of NLP.

6.3.3 Statistical Processing of Natural Language

Statistical processing of natural language represents the classical model of information retrieval systems and is characterized by each document's set of keywords, known as the term index. It is based on a *bag of words*. Here, all the terms in a document are considered as its index terms. Every term is assigned a weight in the function of its importance, mostly determined by its appearance frequency within the document. This way, the word's order, structure, meaning, etc., are not considered. These models are then limited to pairing the document's words with that of the query's. Its simplicity and efficacy have become the most commonly used contemporary models in textual information retrieval systems.[14] The statistical processing model involves two stages, which are as follows[15–18]:

6.3.3.1 Document Preprocessing

This consists of preparing the document for its parameterization, eliminating any elements considered as superfluous.

6.3.3.2 Parameterization

A stage of minimal complexity once the relevant terms have been identified. This consists of quantifying the characteristics of the documents, i.e., the terms.

6.3.4 Applications of NLP

- Virtual Assistants
  The virtual assistants are used for automatic question answering. It is designed to comprehend and understand natural language and deliver the relevant response through natural language generation.
- Sentiment Analysis
  Mostly used on website and social media tracking, NLP is an excellent method for interpreting and analyzing the replies to business messages published on social media platforms. It helps to evaluate the attitude and emotional state of a person who comments on posts. This is implemented through a combination of NLP and statistics by assigning text values as positive, negative, or neutral. This will help in making necessary adjustments in business according to customer's opinions.
- Automatic Summarization
  In automatic summarization, the huge text is simplified into readable form. It summarizes text, by retrieving important and relevant information.
- Machine translation
  Machine translation is the most complex and difficult application of NLP. Machine translation is automatically translating text from one

language to another.[19] It requires a deep knowledge that humans have,[19] for example, grammar, semantics, and facts about the real world. It is required to carry out all the translations in a proper manner.[19]

- Speech Recognition
  In speech segmentation, from the given sound clip of a person or people speaking, words are separated. Speech segmentation is a subtask of speech recognition and is typically grouped with it.

6.4 Summary

Data comes from various resources, and it is difficult to digest the whole data. Text mining is the process of gaining insights and knowledge from huge data sources. It saves time and resources and helps to summarize the documents. Text mining extracts a piece of high-quality information from data. It is a five-step process – text transformation, data preprocessing, feature selection, data mining, and evaluation. Text analytics helps to extract concepts from the text and present them more simply. It has various subtasks such as parsing through the sentence, lemmatization, stemming, and PoS marking. Finally, NLP has come of age over the past 10 years, with products such as Siri, Alexa, and Google's voice search employing NLP to understand and respond to user requests. NLP helps computers to read, comprehend, and infer meaning from human languages.

Exercise

1. What is text mining?
2. Explain major areas of text mining.
3. What is stemming? List down stemming algorithms.
4. Describe the lemmatization process.
5. What are the components of NLP?
6. Write in detail stages of NLP?
7. Give any three applications of NLP.

References

1. Vallez Mari, Pedraza-Jimenez Rafael. Natural language processing in textual information retrieval and related topics – Hipertext – (UPF (rclis.org)) (accessed September 26 2020).

2. Duke University Libraries. http://guides.library.duke.edu/c.php?g=289707&p=1930855 (accessed October 5 2020).
3. Wiley Publications. 2015. Data science & big data analytics: Discovering, analyzing, visualizing and presenting data: Chapter 9 (accessed October 7 2020).
4. Maheshwari Anil. 2015. Data analytics made accessible: Chapter 11 (accessed October 12 2020).
5. Shan Jiang, ChengXiangZhai. 2014. Random walks on adjacency graphs for mining lexical relations from big text data. *Proceedings of IEEE BigData Conference* (accessed October 7 2020).
6. Standford group. https://nlp.stanford.edu/ (accessed October 10 2020).
7. Coursera. https://www.coursera.org/ (accessed October 13 2020).
8. Expert.ai. https://www.expert.ai/blog/nlp-big-data-everyone-know/ (accessed October 5 2020).
9. UKessays.com. https://www.ukessays.com/essays/computer-science/real-time-information-filtering-computer-science-essay.php (accessed October 5 2020).
10. Educba. https://www.educba.com/text-mining/ - Educba (accessed October 11 2020).
11. Text Analysis. https://monkeylearn.com/text-analysis/ (accessed October 6 2020).
12. Zhai ChengXiang, Massung Sean. Text data management and analysis: A practical introduction to information retrieval and text mining, Association for Computing Machinery (ACM). https://dl.acm.org/doi/book/10.1145/2915031 (accessed October 6 2020).
13. Min-Yuh-Day. Social media opinion mining. http://mail.im.tku.edu.tw/~myday/slides/M03_3_M04_1_Social_Media_and_Opinion_Mining_2016.pdf (accessed October 7 2020).
14. https://moam.info/introduction-to-information-retrieval_59c1c9b21723ddc052bf1780.html
15. Manning C.D., Raghavan P., Schutze H. 2008. Document and query weighting schemes, in *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press (accessed October 10 2020).
16. Hu M., Liu B. 2004. Mining and summarizing customer reviews. *Proceedings of the Tenth ACM S/GKDD International Conference on Knowledge Discovery and Data Mining* (accessed October 5 2020).
17. Pak A., Paroubek P. 2010. Twitter as a corpus for sentiment analysis and opinion mining (accessed October 11 2020).
18. Michael W. Berry, Jacob Kogan. 2010. Text mining applications and theory (accessed October 13 2020).
19. Manning Chris, HinrichSchütze. May 1999. *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, MA.

Part III

Platforms for Data Science

7

Data Science Tool: Python

7.1 Basics of Python for Data Science

Data science is a combination of different tools, algorithms, and machine learning concepts to discover hidden insights from raw data.

The notable factor to use Python in data science is the availability of different libraries for data science or data analytics. NumPy (for array-based data manipulation), pandas (for labeled data manipulation), SciPy (for scientific computing), and scikit-learn (for ML algorithms) are a few among the many libraries that are generally used in data science.

This section will cover the basics of Python for data science that every data science aspirant should know.

Python Identifiers

A Python identifier is a name that consists of a series of letters, numbers, and underscore characters.[1]

A Python identifier should follow the below naming conventions[2]:

1. Variable and function names start with a lowercase letter.[3]
2. An identifier starting with an underscore character generally indicates it to be a private identifier.[3]
3. Classes name begins with an uppercase letter.
4. The identifier cannot hold names among the reserved Python keywords (you can get the list of Python keywords using help() built-in function).

Python Variables

Python has no syntax or command to declare a variable. The name assigned to a variable is its variable. It can also be called as a Python identifier. These variables are case sensitive and should ideally begin with a letter. To improve the readability of the variable name, it is better to use lowercase-lettered words which are separated using an underscore character, for example, mixed_case, data_des, etc.

DOI: 10.1201/9780429443237-10

Python Data types

Python supports many basic data types, such as integer, floating-point, character, or string. In Python, we need not declare the data type assigned for the variable. Instead, the Python interpreter identifies the data type of the variable using the value assigned to it. The other two data types are Boolean and None. The Boolean data type takes two values: True or False. The None data type indicates a null value.

```
In[1]: # An integer value
In[2]: # A string of characters
         x = 12
x='Hello!'
type(x)
type(x)
Out[1]:intOut[2]:str
In[3]: # A float value
In[4]:# A Boolean Value
         x = 3.4
x=False
type(x)
type(x)
Out[3]: float
Out[4]:bool
In[5]: # A None value
         x = None
type(x)
Out[5]:NoneType
```

Python Data Structures

There are four data structures in Python:

1. List
2. Tuple
3. Dictionary
4. String

1. List

Python lists are ordered with a certain count. The values in the lists are enclosed in square brackets and are separated using commas. The list indices start with 0. The list can hold both homogenous (i.e., [1, 6, 4, 7]) as well as heterogeneous (i.e., [1, 'a', 'd', 4, 6]) data. Lists are mutable in Python and thus can be altered by adding new items, deleting some items, or modifying the existing items.

A list can be created in four different ways:

1. Comma-separated values – [2, 5, 3, 8, 3]
2. Empty list – []

3. Singleton list –[a]
4. Using list () function

A list can be accessed using the following index numbers:

```
In[6]:
list = [1, 2, 3, 4, 5, 6, 7,8,9];
print ("list[6]: ", list[6])# prints the item at the 6th
position
print("list[2:7]: ", list[2:7])# prints the items from 2nd to
6th position
print("list[:6]: ", list[:6])# prints the items till 6th
position
print("list[7:]: ", list[7:])# prints the item from the 8 th
position
print("list[:-1]: ", list[:-1])# prints the last item in the
list
print("list[::-1]: ", list[::-1])# reverses the list
Out[6]:
list[6]: 7
list[2:7]: [3, 4, 5, 6, 7]
list[:6]: [1, 2, 3, 4, 5, 6]
list[7:]: [8, 9]
list[:-1]: [1, 2, 3, 4, 5, 6, 7, 8]
list[::-1]: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

List Updation:
Modyfying the existing items:

```
In[7]:
list = [1, 2, 3, 4, 5, 6, 7,8,9];
print ("Value at index 4 : ")
print (list[2])
list[4] = 12;
print( "New value  at index 4 : ")
print (list[4])
Out[7]:
Value at index 4 : 3
New value  at index 4 : 12
```

Adding new items:

```
In[8]:
list = [1, 2, 3, 4, 5, 6, 7,8,9];
print(list)
list.append(11)
print(list)
```

```
Out[8]:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 11]
```

You can also append an entire list into the existing list:

```
In[9]:
list = [1, 2, 3, 4, 5, 6, 7,8,9];
print(list)
list.append([11,12,13])
print(list)
Out[9]:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9,[ 11,12,13]]
```

Deleting items in the list:

```
In[10]:
list = [1, 2, 3, 4, 5, 6, 7,8,9];
print(list)
del list[2];
print(list)
Out[10]:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 4, 5, 6, 7, 8, 9]
```

The items in the list can also be deleted using the remove() function.

2. Tuple

A tuple is a collection of ordered elements and is immutable. Any sort of modification in the tuple is prohibited and needs the creation of a new tuple. Same as lists, a tuple can hold homogenous and heterogeneous data. All the items in the tuple are enclosed in parenthesis and are separated using commas.

Tuple can be created in the following ways:

1. Comma-separated values – (2, 5, 3, 8, 3)
2. Empty tuple – ()
3. Singleton tuple – (a,)

   Note: Even when creating a single-valued tuple, a comma must be included after the value.
4. Using the tuple () function

A tuple can be accessed as follows:

```
In[11]:
tup = ('a',' b',' c', 'd', 'e',' f');
print ("tup [5]: ", tup[5])# prints the item at the 5th position
```

```
print("tup [2:4]: ", tup [2:4])# prints the items from 2nd to
3rd position
Out[11]:
tup [5]: f
tup [2:4]: (c, d)
```

Tuple updation:

```
In[12]:
tup1 = ('a', 'b', 'c', 'd', 'e', 'f')
tup2=(1,2,3,4)
tup3=tup1+tup2
print(tup3)
Out[12]:
('a', 'b', 'c', 'd', 'e', 'f', 1, 2, 3, 4)
```

Deleting tuple:
As the tuple is immutable, the entire tuple is deleted using the del() function.

```
In[13]:
print (tup1);
del tup1;
print ("After deleting tup : ");
print (tup1);
Out[13]:
('a', 'b', 'c', 'd', 'e', 'f')[4]
After deleting tup :
---------------------------------------------------------------------------
NameErrorTraceback (most recent call last)
<ipython-input-85-af0fc5861ae2> in <module>()[3]
      2del tup1;
      3 print ("After deleting tup : ");
----> 4 print (tup1);
NameError: name 'tup1' is not defined
```

3. Dictionary

A dictionary is an unordered sequence that contains key-value pairs. A dictionary has unique keys, but the values can be repeated. The keys in the dictionary are immutable, whereas the value can have any type. Keys and values are separated using a colon, and the whole dictionary is enclosed in curly brackets.[3]

Accessing values in Dictionary[3]:

```
In[14]:
dict = {
  "Name": "Pallavi",
  "Age": "23",
```

```
  "College": "PCCOE"
}
print(dict)
dict["Name"]
Out[14]:
{'Name': 'Pallavi', 'Age': '23', 'College': 'PCCOE'}
'Pallavi'
```

Updating the dictionary:

```
In[15]:
dict = {
  "Name": "Pallavi",
  "Age": "23",
  "College": "PCCOE"
}
dict["Age"]=22
dict
Out[15]:
{'Age': 22, 'College': 'PCCOE', 'Name': 'Pallavi'}
```

Deleting dictionary items:

Every element in the dictionary can be deleted individually or as a whole. Also, the entire dictionary can be deleted.

```
In[16]:
del dict['Name'];
dict
Out[16]:
{'Age': '23', 'College': 'PCCOE'}
```

Clearing all the values in dictionary:

```
In[17]:
dict.clear();
dict
Out[17]:{}
```

The entire dictionary can be deleted using **del dict.**

4. String

A string is a combination of zero or more characters enclosed in single quotes or double quotes. Python treats single quotes the same as double quotes and vice versa. Creating a string is the same as assigning the value to a variable in Python.

Example: str1=" Data Science in Python"

Str2=" Python data structures"

String manipulations:

```
In[18]:
str1 = 'This is an example of String Manipulation in Python'
print(str1)
print(str1[0:13])
print('Text length = ', len(str1))
print('Upper case:', str1.upper())
print('Lower case:', str1.lower())
Out[18]:
This is an example of String Manipulation in Python
This is an ex
n Python
Text length = 51
Upper case: THIS IS AN EXAMPLE OF STRING MANIPULATION IN
PYTHON
Lower case: this is an example of string manipulation in
python
```

String Modification:

```
In[19]:
str1 = "String"
print("Text before Modification:", str1)
print()
str1 += ' Modification!'
print("Text After Modification:", str1)
Out[19]:
Text before Modification:String
Text After Modification: String Modification!
```

7.2 Python Libraries: DataFrame Manipulation with pandas and NumPy

Python provides its large library support for various stages in data science. Python libraries contain various tools, functions, and methods to analyze data. Each library has its focus on data mining, data visualization, etc.

There are various Python libraries used for data science. Some of them are listed below:

1. Data mining
   - Scrapy
   - BeautifulSoap

2. Data processing and modeling
   - NumPy
   - pandas
   - SciPy
   - Scikit-learn
   - XGBoost
3. Data visualization
   - Matplotlib
   - Seaborn
   - Pydot
   - Plotly

In this section, we will see pandas and NumPy Python libraries in detail.

pandas
pandas is a free Python data analysis and data handling software library. pandas provides a variety of high-performance and convenient data structures along with operations for data manipulation. pandas is also known for its range of tools for reading and writing data between in-memory data structures and various file formats.

The standard form to import the pandas module is:

```
In[20]:import pandas as pd
```

Note: pd is just an alias. Any name can be used as an alias.

Load and examine the dataset:
DataFrame is a two-dimensional array where each feature (column) is simply a shared index series. A DataFrame can be viewed as a generalization of the NumPy array.

A dataset can be read and loaded to the pandas DataFrame using the below code syntax:

```
In[21]:df = pd.read_csv('iris.csv')
```

The first glance at the dataset can be made using the following functions:

- **head()** – This built-in function displays the first five rows by default. Passing any integer in the function will display that many certain rows starting from the top.
- **tail()** – This built-in function displays the last five rows by default. Passing any integer in the function will display that many certain rows from the bottom.

- **sample()** – This built-in function displays a random row from the dataset. Passing any integer in the function will display that many random rows.
- **info()** – This built-in function displays a complete summary of the dataset, including the no. of rows, columns, the data types of the features, the null values, etc.
- **describe()** – This built-in function displays comprehensive statistics that sums up the central tendency, the shape of the dataset, and dispersion.

DataFrame Indexing

.loc() – This function is used for label-based indexing. This method recovers rows that get only labeled index as input and return rows (Figure 7.1).

Syntax: pandas.DataFrame.loc[]

```
In[22]:
import pandas as pd
data = pd.read_csv('Downloads\\MovieData.csv', index_col
="director_name")
first = data.loc["Gore Verbinski"]
first.head(10)
Out[22]:
```

.iloc(): This method retrieves rows using integer-based indexing (Figure 7.2).

director_name	color	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
Gore Verbinski	Color	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
Gore Verbinski	Color	313.0	151.0	563.0	1000.0	Orlando Bloom	40000.0	423032628.0
Gore Verbinski	Color	450.0	150.0	563.0	1000.0	Ruth Wilson	40000.0	89289910.0
Gore Verbinski	Color	362.0	107.0	563.0	939.0	Ray Winstone	40000.0	123207194.0
Gore Verbinski	Color	271.0	143.0	563.0	1000.0	Orlando Bloom	40000.0	305388685.0
Gore Verbinski	Color	125.0	123.0	563.0	8000.0	Brad Pitt	24000.0	66808615.0
Gore Verbinski	Color	156.0	102.0	563.0	385.0	Hope Davis	12000.0	12469811.0

7 rows × 27 columns

FIGURE 7.1
Usage of the loc() function.

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gr
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	76050584
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	3094041
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	44813064

3 rows × 28 columns

FIGURE 7.2
Usage of the iloc() function.

```
In[23]:
import pandas as pd
data = pd.read_csv('Downloads\\MovieData.csv')
first = data.iloc[[0,1,3]]
Out[23]:
```

Elements in the data frame can be dropped using the drop() method.
For eg.:df.drop(4)
The above command will drop the row at index 4.

Data frame operations
Sort the data frame
The data frame can be sorted in two ways:

- **sort_index** – This method sorts the data frame based on the index of the certain values.
- **sort_values** – This method sorts the data frame based on the values in the specified columns.

The sort functions create a new data frame of the sorted values. To make the change in the same data frame, set the inplace attribute to True. The sort functions sort the data frame in the ascending order by default, i.e., the ascending attribute is set to True by default. Change the value to this parameter to True or False as per the requirements (Figures 7.3 and 7.4).

	total_bill	tip	sex	smoker	day	time	size
243	18.78	3.00	Female	No	Thur	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
240	27.18	2.00	Female	Yes	Sat	Dinner	2
239	29.03	5.92	Male	No	Sat	Dinner	3

FIGURE 7.3
Sorting the index in the descending manner.

	total_bill	tip	sex	smoker	day	time	size
243	18.78	3.00	Female	No	Thur	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
240	27.18	2.00	Female	Yes	Sat	Dinner	2
239	29.03	5.92	Male	No	Sat	Dinner	3

FIGURE 7.4
Sorting the index in the ascending manner.

```
In[24]:dfa.sort_index(ascending=False).head()
Out[24]:


In[25]:dfa.sort_values(by='tip', ascending=True).head()
Out[25]:
```

NumPy

NumPy is an acronym for Numerical Python. NumPy library is majorly used for the functioning of arrays. It can also be used in linear algebra and matrices. The motive of the arrays is set by the list in Python. In a field such as data science, speed and resources play a vital role. To overcome this disadvantage of the lists, the NumPy arrays are used.

The major difference which makes the NumPy arrays to be more prominent than the Python lists is the vectorized operations supported by the NumPy arrays and not by the Python lists.

ndarray is an array class used in NumPy. All the items in the NumPy arrays are enclosed in square brackets and are initialized using lists.

Creation of the NumPy arrays:

Various NumPy arrays can be created based on the ranks, using a tuple or a list.

```
In[26]:
import numpy as np[3]
arr1 = np.array(['a', 'b', 'c', 'd'])[3]
print("Rank 1 Array : \n", arr1)
print()
arr2 = np.array([[1, 2, 3], ['a', 'b', 'c']])[5]
print("Rank 2 Array: \n", arr2)
print()
arr3 = np.array((4, 5, 6))
```

```
print("\n Tuple passed Array:\n", arr3)
```

Out[26]:

```
Rank 1 Array :
['a' 'b' 'c' 'd']

Rank 2 Array:
[['1' '2' '3']
['a' 'b' 'c']]

Tuple passed Array:
[4 5 6]
```

Accessing the array using its index:
Slicing an array is easier in Python using NumPy. Slicing is accessing a range of items in the original array to form a new array.[6]

In[27]:

```
import numpy as np
arr=np.arange(10)
print(arr)
print("arr[:5]", arr[:5])#prints elements till the 4th
position
print("arr[3:]", arr[3:])#prints elements from the 3rd
position to the end
print("arr[::5]", arr[::5])#prints elements divisible by 5
print("arr[1::2]", arr[:5])#prints elements starting from the
1st position and step by 2
arr1 = np.array([[1,2,3],
                 [4,5,6],
                 [7, 8, 9]])[1]
print("Initial Array: ")[1]
print(arr1)
print()
sliced_arr = arr1[:2,:2]
print ("Array with first 2 rows and 2 columns", sliced_arr)
```

Out[27]:

```
[0 1 2 3 4 5 6 7 8 9]
arr[:5] [0 1 2 3 4]
arr[3:] [3 4 5 6 7 8 9]
arr[::5] [0 5]
arr[1::2] [0 1 2 3 4]
Initial Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Array with first 2 rows and 2 columns:
 [[1 2]
 [4 5]]
```

Basic array operations
The NumPy array also allows us to perform some basic operations such as addition, subtraction, etc. Unary and binary operations also can be performed on the arrays.

```
In[28]:
import numpy as np
arr1 = np.array([1,2,3,4])
arr2 = np.array([6,7,8,9])
print("arr1:", arr1)
print ("arr1 after adding 2 to every element:", arr1 + 2)
print("\narr2:", arr2)
print("arr1 after subtracting 1 from every element:", arr2-1)
print ("\nSum of all array elements in arr1: ", arr1.sum())
print ("\nArray sum:\n", arr1 + arr2)
Out[28]:
arr1: [1 2 3 4]
arr1 after adding 2 to every element: [3 4 5 6]
arr2: [6 7 8 9]
arr1 after subtracting 1 from every element: [5 6 7 8]

Sum of all array elements in arr1: 10
Array sum:
[ 7  9 11 13]
```

7.3 Exploration Data Analysis with Python

Statically, EDA is a technique that evaluates datasets to contextualize their primary characteristics, often with visualization. In short, we explore the dataset. Similarly, when constructing an ML model, data exploration is very important to figure out the sensibility in your dataset. The main objective of EDA is to make it competent to work with any ML algorithm.

Objectives of EDA:

1. Description of the dataset
2. Removing corrupted data
3. Handling outliers
4. Evaluation of the relationship between variables through visualization

1. Description of the dataset

A data scientist should always know the data and other vital statistics of the dataset before moving further. The most basic property that can be used too so is the describe() function in Python. Applying

```
In [5]: import numpy as np
        import pandas as pd
        df=pd.read_csv('C:\\Users\PC\Downloads\MovieData.csv')
        df.describe()
```

Out[5]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+03	5.043000e+03	
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+07	8.366816e+04	
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+07	1.384853e+05	
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02	5.000000e+00	
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+06	8.593500e+03	
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+07	3.435900e+04	
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+07	9.630900e+04	
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08	1.689764e+06	

FIGURE 7.5
Description of the dataset.

the describe() function to a DataFrame in pandas provides comprehensive statistics, summarizing various factors, such as the dispersion, the shape using count, std, mean, std, etc (Figure 7.5).

2. Removing corrupted data

   Missing values when not handled properly can reduce the excellence in the performance matrix. This can also lead to the false classification which can harm our model thus built. Missing or NULL values can be handled in various ways:

   Drop NULL or missing values:

   Although this is the simplest way to handle missing or NULL values, it is usually not used as it shrinks the dataset by deleting such observations. In Python, the dropna() function is used to delete the missing values from the dataset.

```
In[29]:df.shape
Out[29]: (5043, 28)
In[30]:df.dropna()
df.shape
Out[30]: (5043,28)
```

The above code snippet denotes that there are no missing values in the dataset.[7]

Fill missing values:

In this method, missing values are replaced with a statistic such as mean, median, mode, min, max, or even any user-defined value. The fillna() function is used in Python for filling the missing values.

Suppose, we have missing values in the above-used dataset, then the below code snippet depicts to replace the NULL/missing values with the mean of the data points in the dataset.[1]

```
In[31]: df['duration']=df['duration'].fillna(df.mean())
```

Predict missing values with an ML algorithm:
Predicting missing values is the most efficient way to handle the missing data. One can use the regression model or classification model (depending on the dataset) to predict the missing data.

3. Handling outliers
   Outliers are the data points that are far different from the crowd of data points. This can be an indication of the variation in the dataset. Outliers can be detected using boxplot, scatterplot, inter quartile range, and Z-score.
4. Evaluation of the relationship between variables through visualization
   Visualization helps us understand the various relationship between features. Some widely used techniques for data visualization in Python are the histograms and the heatmaps. The histogram helps to assess probability distribution that is quite easy to interpret. Python provides various options to build and plot histograms. The heatmap is represented where the data points are denoted using colors.

7.4 Time Series Data

Time series data is a type of structured data and is used in various domains, such as accounting, ecology, economics, neuroscience, etc. Time series can be referred to as anything and everything that is measured in the form of time. The data points in time series appear at regular intervals according to some rules. Time series may also be uneven without an even fixed unit of time.

Date and time data come in various types:

- **Timestamps –** The timestamp refers to particular instants in time (e.g., July 24, 2020, at 8:00 a.m.).[2]
- **Time Intervals and Periods –** Time intervals and periods refer to a time length between the initial and the final point. Periods generally refer to a specific case of time intervals where each time interval is of a uniform length and does not coincide (e.g., 24-hour period of days).[2]
- **Time Deltas or Durations** – It refers to an exact length of time (e.g., a duration of 12.43 seconds).[2]

This section will give you a brief overview of the time series in pandas with the correct approach to be used while working on time-series in Python or pandas.

Dates and Times in Python
Native Python datetime:
There exist data types for date and time and also calendar functionality in the Python standard library. The datetime is broadly used in time series:

```
In[32]: from datetime import datetime[8]
date_now = datetime.now()[8]
date_now
Out[32]:datetime.datetime(2020, 8, 7, 23, 4, 3, 792268)
```

The datetime object stores both the date and time up to the microsecond.

```
In[33]:now.year, now.month, now.day
Out[33]: (2012, 8, 4)
```

You can parse dates from a variety of string formats using the dateutil module[2]:

```
In[34][2]:from dateutil import parser
date = parser.parse("8th of August, 2020")
date
Out[34][2]:datetime.datetime(2020, 8, 8, 0, 0)
```

Datetime64 of NumPy:
The NumPy team has added a set of the time series data type to NumPy.[2] The data type "datetime64" encrypts dates as 64-bit integers, allowing data arrays to be displayed very efficiently.[2]

```
In[35][2]: import numpy as np
date = np.array('2017-08-04', dtype=np.datetime64)
date
Out[35]: array('2017-08-04', dtype='datetime64[D]')
```

We can even perform vectorized operation on the formatted date:

```
In[36]: date + np.arange(8)
Out[36]: array(['2017-08-04', '2017-08-05', '2017-08-06',
'2017-08-07',
       '2017-08-08', '2017-08-09', '2017-08-10',
'2017-08-11'],
dtype='datetime64[D]')
```

pandas Time Series: Indexing by Time[2]
Using the pandas time series module, you can index your data using timestamps.

```
In[37]:import pandas as pd
index = pd.DatetimeIndex(['2017-07-04', '2018-09-12', '2015-
02-11', '2015-08-04'])
data = pd.Series([0, 1, 2, 3], index=index)[2]
data
Out[37][2]: 2017-07-04 0
                2018-09-12    1
                2015-02-11    2
                2015-08-04    3
dtype: int64
```

We can also obtain the sliced data by passing a year to get data from that year.[2]

```
In[38][2]: date['2015']
Out[38][2]: 2015-02-11 2
                2015-08-04    3
dtype: int64
```

7.5 Clustering with Python

Clustering is a form of unsupervised learning. In unsupervised learning,[9,10] the features of a dataset are modeled without any label references.

Clustering is the process of dividing the data points into many clusters in such a way that data points in the same clusters are more similar to other data points in the same cluster and are different from data points in other clusters. It is generally a compilation of objects in terms of similarity and dissimilarity among them.

For example, the data points clustered together in the graph below can be classified into one single group. We can distinguish the clusters, and we can identify that there are two clusters in the below picture (Figure 7.6).

There are various clustering algorithms in ML. The following are the most common clustering techniques used in data science:

1. k-Means clustering
2. Agglomerative clustering
3. DBSCAN clustering

1. k-Means clustering algorithm

   The simplest unsupervised learning algorithm is the k-means clustering algorithm. k-Means clustering comes under the partitioning method of clustering. k-Means algorithm classifies "n" observations

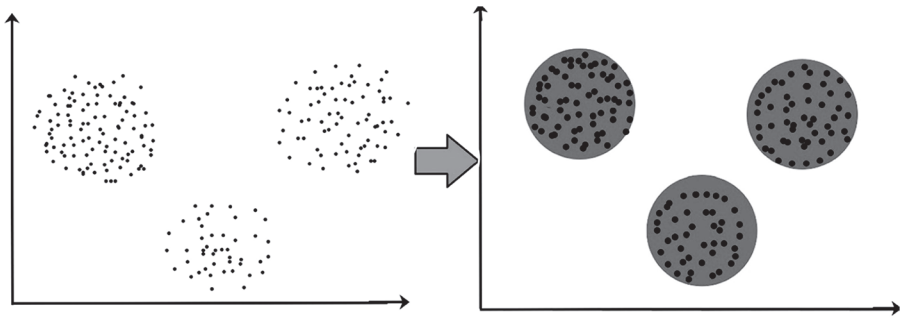


FIGURE 7.6
Clustering.

in "k" clusters where each observation lies to a cluster where the nearest mean serves as a cluster prototype.

Suppose we are given a database of "n" objects and the partitioning method constructs "k" partition of data,[11] each partition will represent a cluster and $k \leq n$.[11] It means that it will classify the data into "k" groups, which satisfies the following requirements[11]: –

- Each group contains at least one object.[11]
- Each object must belong to exactly one group.[11]

Steps:

1. k-Objects from the dataset are randomly assigned as cluster centers.
2. Based on the mean values, reassign each object to its most similar object(cluster).
3. With the updated values, recalculate the mean of the cluster.
4. Repeat Step 4 until no change is observed.

k-Means clustering is very simple to be implemented in Python using scikit-learn.

In the code snippet below, we created an artificial dataset using make_blobs. To use the k-means class in Python, we need to import it from the sklearn. cluster. Set the value of "k" in the n_cluster parameter, and then run the algorithm by calling the fit method (Figure 7.7).

```
In[39][12]: from sklearn.datasets import make_blobs
fromsklearn.cluster import KMeans
fromadspy_shared_utilities import plot_labelled_scatter
         X, y = make_blobs(random_state = 10)
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
```

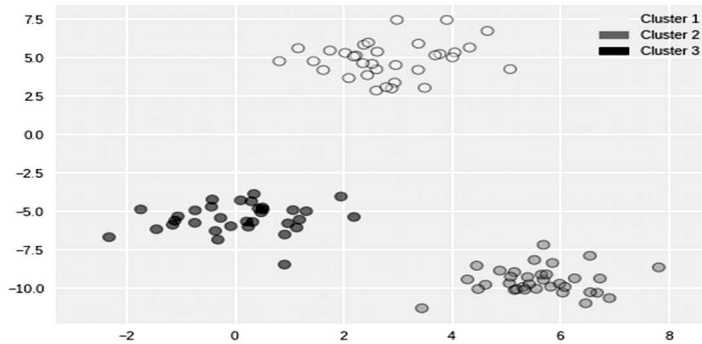


FIGURE 7.7
k-Means clustering.

```
plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1',
'Cluster 2', 'Cluster 3'])
```

Out[39]:

2. Agglomerative clustering

   Agglomerative approach is a type of a hierarchical method of clustering.[12] The clusters here are formed in a tree structure based on hierarchy. This approach is also known as the bottom-up approach.[13] In this, we start with each object forming a separate group.[13] It keeps on merging the objects or groups that are close to one another.[13] It keeps on doing so until all of the groups are merged into one or until the termination condition holds.[13]

   Linkage criteria for agglomerative clustering:

   - **Ward's Method** – This method looks for two clusters, giving the least increase in total variance among all clusters, and then merge them.
   - **Average Linkage** – Average linkage search for two clusters to merge which has the least average distance between the points.
   - **Complete Linkage** – This linkage, also called the maximum linkage, looks for two clusters having the maximum distance between points to be merged.

   Ward's method is usually compatible with most of the datasets, but if you want variety in the sizes of the clusters, the complete and average linkage can be used (Figures 7.8 and 7.9).

```
In[40][12]:from sklearn.datasets import make_blobs
fromsklearn.cluster import AgglomerativeClustering
fromadspy_shared_utilities import plot_labelled_scatter
```

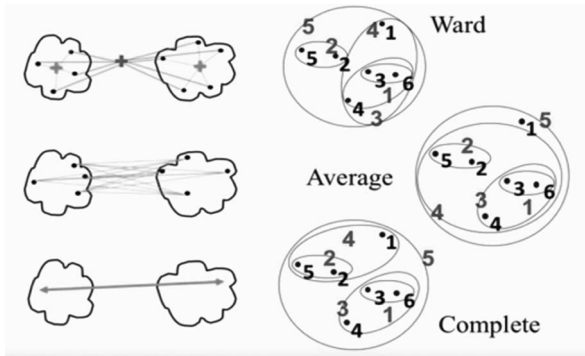


FIGURE 7.8
Linkage criteria for agglomerative clustering.

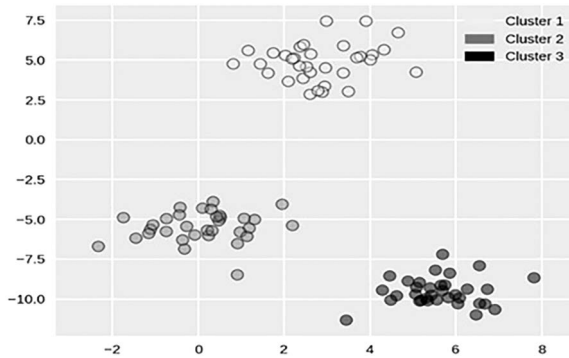


FIGURE 7.9
Agglomerative clustering.

```
X, y = make_blobs(random_state = 10)
cls = AgglomerativeClustering(n_clusters = 3)
cls_assignment = cls.fit_predict(X)
plot_labelled_scatter(X, cls_assignment, ['Cluster 1',
'Cluster 2', 'Cluster 3'])
Out[40]:
```

3. DBSCAN clustering

DBSCAN is an acronym for the density-based spatial clustering of applications with noise. In DBSCAN, one need not specify the number of clusters in advance. Also, DBSCAN is well suited for datasets with complex cluster shapes. In the DBSCAN method, clusters refer to the area that is denser in the data space, being isolated by the areas which are much less densely populated or empty.

Minimum points (MinPts) and epsilon (eps) play a very important role in the DBSCAN algorithm. Points lying in the larger dense

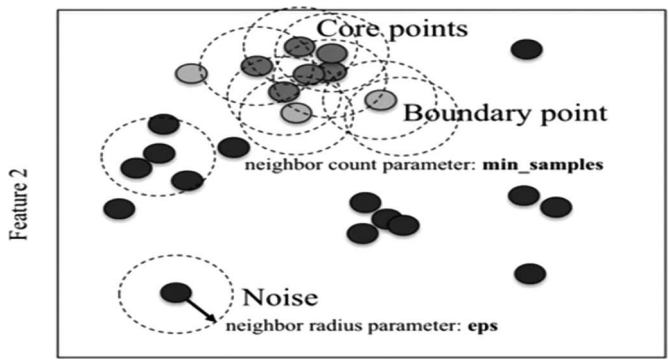


FIGURE 7.10
DBSCAN clustering.

region are referred to as core samples. The core samples which are apart by a distance of eps units are put into one cluster. After the points being distinguished as core samples, the data points not making it into any cluster are said to be as noise. Boundary points are the points that fall within the eps units' distance but cannot be termed as core points in themselves (Figure 7.10).

Same as that of the other clustering methods, DBSCAN is imported from the scikit-learn cluster library (Figure 7.11).

```
In[41]12:from sklearn.cluster import DBSCAN
fromsklearn.datasets import make_blobs
  X, y = make_blobs(random_state = 9, n_samples = 25)
dbscan = DBSCAN(eps = 2, min_samples = 2)
cls = dbscan.fit_predict(X)
```

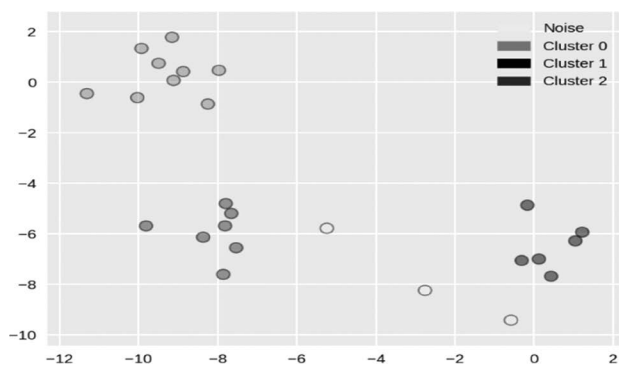


FIGURE 7.11
DBSCAN clustering implementation.

```
print("Cluster membership values:\n{}".format(cls))
plot_labelled_scatter(X, cls + 1, ['Noise', 'Cluster 0',
'Cluster 1', 'Cluster 2'])
Out[41]:
```

7.6 ARCH and GARCH

The ARCH Model

The ARCH is an acronym that stands for autoregressive conditional heteroskedasticity. It is an approach that offers a way to model a transformation in variation in a time series that is dependent on time, such as an increase or decrease in volatility.[14] It is a process that directly models variance change over time in a time sequence.[14]

Particularly, the ARCH model features the variance at a time interval as a component of error terms (e.g., zero mean) in the mean process.[14]

A lag factor must be defined to determine the number of previous error terms to be used in the model.[14]

p: The number of lags squared error terms to be used in the ARCH model.[14]

An ARCH model is built in three stages:

1. Model definition
2. Model fitting
3. Forecasting

In the below code snippet, we have created a simple dataset using the gauss() function that will generate a range of random numbers with the mean equal to 0 and the variance initially at 0.0 that increases gradually.[14] It is better to split the dataset into the training and testing dataset as we can perform model fitting on the training dataset and assess its functioning on the test set.[14] A model is defined by using the arch_model() function. In this case, we have assigned the lag parameter, i.e., p=15. The model is then used on the data by the fit() function.[14] At last, we can observe our model by using the forecast() function (Figure 7.12).

```
In[42][14]:from random import gauss
from random import seed
frommatplotlib import pyplot
from arch import arch_model
seed(1)
data = [gauss(0, i*0.01) for i in range(0,100)]
n_test = 10
train, test = data[:-n_test], data[-n_test:]
```

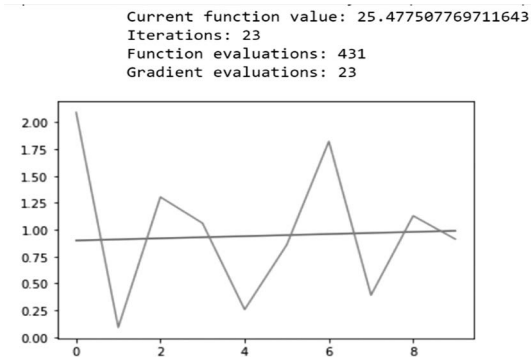


FIGURE 7.12
ARCH model implementation.

```
a_model = arch_model(train, mean='Zero', vol='ARCH', p=15)
model_fit = a_model.fit()
yhat = model_fit.forecast(horizon=n_test)
var = [i*0.01 for i in range(0,100)]
pyplot.plot(var[-n_test:])
pyplot.plot(yhat.variance.values[-1, :])
pyplot.show()
Out[42]:
```

The GARCH Model

GARCH is an acronym for generalized autoregressive conditional heteroskedasticity.[14] The GARCH model is an extended version of the ARCH model.[14] Explicitly, the model integrates lag variation terms, along with residual errors during the mean process.

In GARCH model:

- **p** – No. of lag variances.
- **q** – No. of lag residual errors.

Note: The "p" parameter used in the ARCH model is denoted as the "q" parameter in the GARCH model.

As stated earlier, the GARCH model colligates the ARCH model, so GARCH(0,q) is the same as ARCH(q).

A GARCH model can be fitted using the ARCH library itself with just vol="GARCH" instead of ARCH, and both lag parameters will be included (Figure 7.13).

```
In[43][14]:from random import gauss
from random import seed
frommatplotlib import pyplot
from arch import arch_model
```

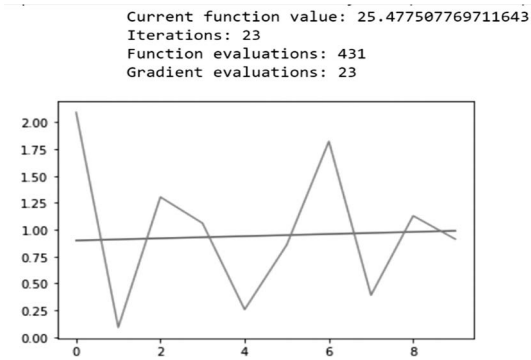


FIGURE 7.13
GARCH model implementation.

```
seed(1)
data = [gauss(0, i*0.01) for i in range(0,100)]
n_test = 10
train, test = data[:-n_test], data[-n_test:]
model = arch_model(train, mean='Zero', vol='GARCH', p=15,
q=15)
model_fit = model.fit()
yhat = model_fit.forecast(horizon=n_test)
var = [i*0.01 for i in range(0,100)]
pyplot.plot(var[-n_test:])
pyplot.plot(yhat.variance.values[-1, :])
pyplot.show()
Out[43]:
```

7.7 Dimensionality Reduction

Dimensionality reduction is an example of an unsupervised algorithm in which the structure of the dataset concludes the labels or other valuable information.[2] Dimensionality reduction usually aims to retrieve a few low-dimensional representations of data that, in a way, maintains the relevant qualities of its entire dataset.[2] It converts your original dataset that consists of, say, 250 dimensions and finds an estimated kind of dataset that uses, say, only ten dimensions.

The most popular dimensionality reduction techniques used in data science are as follows:

i. Principal component analysis (PCA)
ii. Manifold learning.

1. PCA

   PCA is a technique which extracts a new set of variables from an existing wide range of variables. Such freshly extracted variables are regarded as the principal components. The linear collection of the initial variables is known as principal components. The principal components are retrieved in such a manner that the very first principal component describes the maximum variance in the dataset. The second principal component seeks to outline the remaining variance in the dataset. This principal component is not correlated to the main, i.e., the first principal component. The next principal component aims to describe the variation that has remained unexplained by the first two principal components and so on.

   To apply the PCA, we need to first import PCA class from sklearn. decomposition. The data transformation is performed by using the transform and fir methods of the StandardScaler class. This is essential as to have zero mean and unit variance for the range of values in each feature. While creating the PCA object, we retrieve just the first two principal components to reduce the dimensionality. Then by creating a scatterplot using these two new features (two principal components), we can interpret the formation of clusters (Figure 7.14).

```
In[44]12: from sklearn.preprocessing import StandardScaler
fromsklearn.decomposition import PCA
fromadspy_shared_utilities import plot_labelled_scatter
             # each feature should be centered (zero mean) and
with unit variance
X_normalized = StandardScaler().fit(X_fruits).
transform(X_fruits)
pca = PCA(n_components = 2).fit(X_normalized)
X_pca = pca.transform(X_normalized)
```

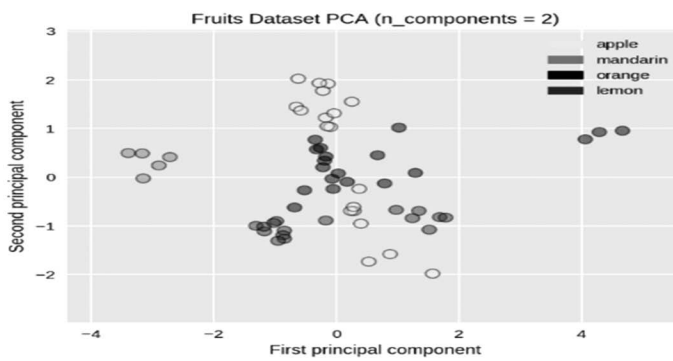


FIGURE 7.14
Principal component analysis.

```
plot_labelled_scatter(X_pca, y_fruits, ['apple', 'mandarin',
'orange', 'lemon'])
plt.xlabel('First principal component')
plt.ylabel('Second principal component')
plt.title('Fruits Dataset PCA (n_components = 2)');
Out[44]:
```

2. Manifold Learning

A class of unsupervised learning that aims to classify datasets as low-dimensional manifolds enclosed in high-dimensional spaces is called manifold learning. Manifold learning is also known for its visualizations.

One commonly used method of manifold learning is called **multidimensional scaling (MDS).** There are many types of MDS, but they all have a general aim: to visualize a high-dimensional space and project it into a lower dimensional space.[12] In this way, the high-dimensional data can be visualized using its clustering behavior (Figure 7.15).

Similar to the PCA, each feature needs to be normalized so that its features possess zero mean and unit variance. Once you have imported MDS class from sklearn.manifold and converted your input data, you can create an MDS object, defining the number of dimensions(components)—usually assigned as two for visualization (Figure 7.16).

```
In[45][12]: from adspy_shared_utilities import
plot_labelled_scatter
fromsklearn.preprocessing import StandardScaler
fromsklearn.manifold import MDS
# each feature should be centered (zero mean) and with unit
variance
```

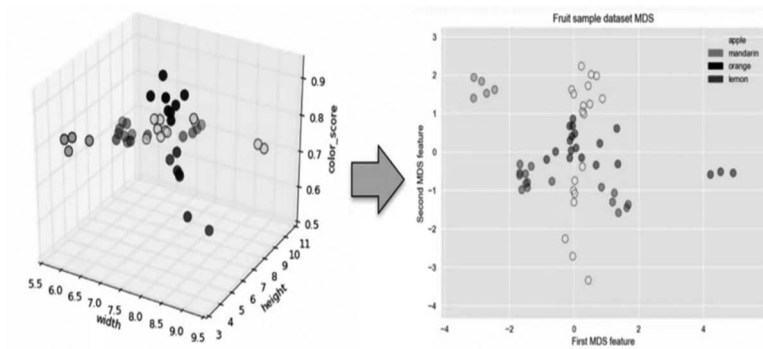


FIGURE 7.15
High-dimensional dataset to two-dimensional dataset.

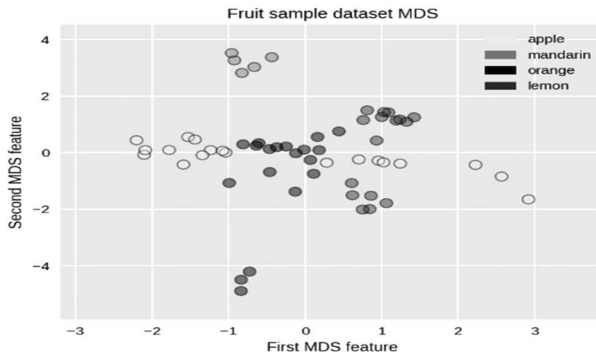


FIGURE 7.16
Multidimensional scaling.

```
X_fruits_normalized = StandardScaler().fit(X_fruits).
transform(X_fruits)
mds = MDS(n_components = 2)
X_fruits_mds = mds.fit_transform(X_fruits_normalized)
plot_labelled_scatter(X_fruits_mds, y_fruits, ['apple',
'mandarin', 'orange', 'lemon'])
plt.xlabel('First MDS feature')
plt.ylabel('Second MDS feature')
plt.title('Fruit sample dataset MDS');
```

Out[45]:

A particularly beneficial manifold learning algorithm to visualize the data is called the **t-distributed stochastic neighbor embedding (t-SNE) algorithm.**[12] t-SNE uses a two-dimensional depiction of the data points so that the distances within the points in the 2D scatterplot correspond as closely as possible to the distances between the same data points in the high-dimensional space.[12] In particular, t-SNE gives significant emphasis to preserve knowledge about distances between neighboring data points.[12] The code is very similar to that of MDS and essentially only replaces MDS with t-SNE. t -SNE seems to function best on datasets with a more excellently defined local structure (Figure 7.17).

```
In[46][12]:from sklearn.manifold import TSNE
tsne = TSNE(random_state = 0)
X_tsne = tsne.fit_transform(X_fruits_normalized)
plot_labelled_scatter(X_tsne, y_fruits, ['apple', 'mandarin',
'orange', 'lemon'])
plt.xlabel('First t-SNE feature')
plt.ylabel('Second t-SNE feature')
plt.title('Fruits dataset t-SNE');
Out[46]:
```

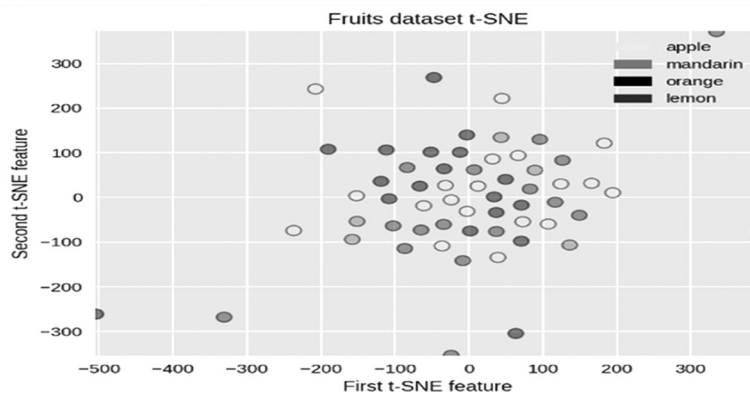


FIGURE 7.17
Implementation of t-SNE algorithm.

7.8 Python for Machine ML

ML is the primary mode by which data science expresses itself to the wider world.[2] ML is where computational and algorithmic expertise of data science get together with the statistical aspects of data science.[2]

It is better to think of ML as a means of building data models in data science.[2] Mainly, ML includes creating mathematical models to help you understand the data.[2]

Machine learning[15] can be classified into two major categories:

i. Supervised learning
ii. Unsupervised learning.

1. Supervised learning

Supervised learning involves modeling the relationship between the measured characteristics of the data and a few labels associated with the data.[2] Once this model is established, it can be used to put in labels to new, unknown data.[2]

Supervised learning is further subdivided as follows:

i. **Classification** – Classification is a supervised ML approach in which the algorithm learns from the provided data input and then classifies new observations using this learning.
ii. **Regression** – Regression analysis examines the relationship between a (dependent) target and a predictor (independent variable(s)). The labels here are continuous.

2. Unsupervised learning

   In unsupervised learning, the features of a dataset are modeled without any label references. Unsupervised learning is a computational training using information that is not marked or labeled, which enables the algorithm to operate without guidance on this information.[16]

   Unsupervised learning is classified into two categories of algorithms:

   i. **Clustering** – A clustering problem is where you want to locate the underlying data clusters such as consumer grouping by purchasing behavior.

   ii. **Dimensionality reduction** – Dimensionality reduction is the transformation of high-dimensional dataspace to low-dimensional dataspace, retrieving the important qualities of the dataset.

   Furthermore, semi-supervised learning lies somewhere between the supervised and unsupervised learning. The semi-supervised learning is also useful if incomplete labels exist.

Scikit-Learn

Among the various Python modules that provide implementations of many ML algorithms, the most popular is scikit-learn. Scikit-learn offers a lot of common algorithms, such as clustering, dimensionality reduction, classification, and regression. Scikit-learn is featured with uniform and slimmed-down application programming interfaces (APIs) with practical and detailed documentation. An advantage of this consistency is to make the transition to a new model or algorithm very straightforward once you are under the basic utilization and syntax of scikit-learn on one model form.[2]

Data Representation in Scikit-Learn

Data as a table

A fundamental table is a two-dimensional data grid[17] where the rows represent individual data elements, and the columns represent quantities of each element.[2]

For example, let's consider the Tips dataset. This dataset can be downloaded in the form of a pandas DataFrame using the seaborn library[2] (Figure 7.18):

```
In[47]²: import seaborn as sns
tips = sns.load_dataset('tips')
tips.head()
Out[47]:
```

Here, each row applies to a single observed tip, and the number of rows in the dataset refers to the total number of tips. Each column of data often refers to a related piece of information identifying the individual samples.[18]

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

FIGURE 7.18
Implementation using seaborn library.

Feature Matrix:
This above table structure demonstrates the knowledge can be seen as a two-dimensional numerical array or matrix that is commonly referred to as a features matrix.[2] The features matrix is assumed to be in the form as shape [n_ samples, n _features].[2] Features matrix is mostly used in NumPy arrays and also pandas DataFrame.[2] Scikit-learn modules accept SciPy sparse matrices.[2]

The samples (i.e., rows) always refer to the different objects represented by the dataset.[2] It can be anything that you can describe with related measurements.

Features (i.e., columns) refer to the discrete observations which can describe every sample in a quantitative way.[2] Features are usually real but can be Boolean or discrete.[2]

Target Array:
In addition to the features matrix, a target array is also generally used. The target array is generally one-dimensional, of length−n samples, and is commonly used in the NumPy array or pandas series.[18] The target array may have continuous numeric values or discrete classes/labels.[18] Some scikit-learn estimators do handle multiple target values in the form of a two-dimensional [n samples, n targets] target array.[18]

Often, one point of confusion is how the target array differs from the other column features. The distinguishing feature of the target array from the column features is that it is usually the dependent variable, i.e., the quantity that needs to be predicted from the data. For example, in the preceding data, we may wish to construct a model that can predict future tips to be given based on other measurements; in this case, the future tips column will be regarded as a feature.[2,19]

Scikit-Learn's Estimator API:
The scikit-learn API has been built on the following principles:

- Consistency
  A common interface is drawn from a bounded set of methods along with regular documentation.[2]

- Inspection
  All values of the specified parameters are shown as public attributes.[2]
- Limited hierarchy of objects
  Datasets are represented using NumPy arrays, SciPy sparse matrices, pandas DataFrame, algorithms using Python classes, and Python strings used by parameter names.[2]
- Composition
  Many ML tasks can be demonstrated as sequences of more basic algorithms.[2] Scikit-learn uses this feature as an advantage.[2]
- Sensitive defaults
  The module defined an appropriate default value whenever the models are in search of user-specified parameters.

7.9 KNN/Decision Tree/ Random Forest/SVM

1. KNN Algorithm
   The KNN algorithm is a simple ML algorithm that is used to solve regression and classification problems. The prediction for an unknown data instance is found by searching for the most similar k-instances in the training dataset and returning the most similar data instance.
   KNN Algorithm Implementation:
   1. Load the dataset (training and testing dataset)
   2. Select the value of k, i.e., the closest data points.[1]
   3. Repeat the below process for all the new data instances:
      i. Calculate the Euclidian distance between the new data instance and the training data points.[7]
      ii. Sort the distance parameters in ascending order.
      iii. Select the first k parameters (The k nearest data points).
      iv. Assign the class that is most frequent in the k rows.
   4. End

   In Python, KNeighborsClassifier class of the sklearn module is used in KNN as a classifier. After reading the dataset using the pandas data frame, the dataset is processed using the iloc function and then split into the training and testing set. The training set is then scaled using the StandardScaler class of the sklearn module. This scaled dataset is trained on a model using the KNeighborsClassifier class, and thus, the prediction is made (Figure 7.19).

```
Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor       1.00      0.95      0.97        19
 Iris-virginica       0.96      1.00      0.98        22

       accuracy                           0.98        60
      macro avg       0.99      0.98      0.98        60
   weighted avg       0.98      0.98      0.98        60

Accuracy: 0.9833333333333333
```

FIGURE 7.19
Implementation of KNN algorithm.

```
In[48]:
fromsklearn.datasets import load_breast_cancer
fromsklearn.preprocessing import StandardScaler
fromsklearn.neighbors import KNeighborsClassifier
fromsklearn.metrics import classification_report,
accuracy_score
fromsklearn.model_selection import train_test_split
importnumpy as np
importmatplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('C:/Users/PC/Downloads/iris.csv')
dataset.head()
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.40)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
classifier = KNeighborsClassifier(n_neighbors = 8)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test, y_pred)
print("Accuracy:", result2)
Out[48]:
```

2. Decision Tree

As the name suggests, a decision tree is a tree-like structure where each internal node constitutes a feature, each leaf node represents

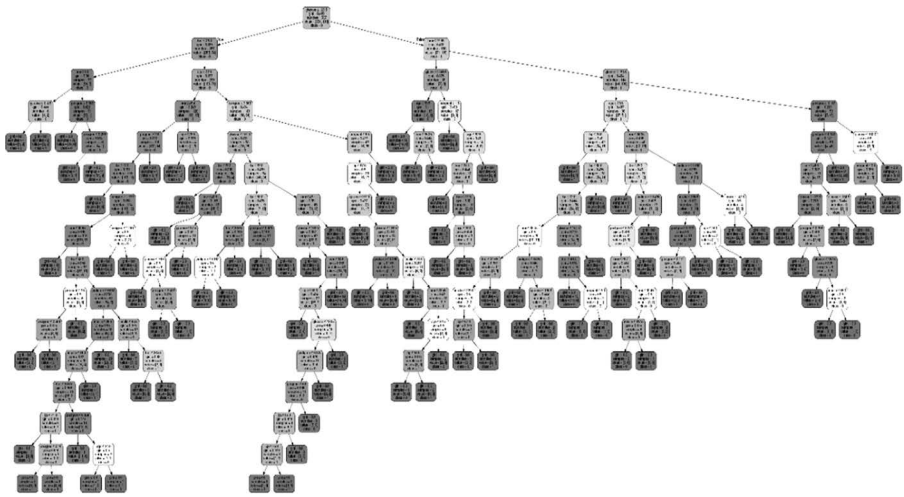


FIGURE 7.20
Implementation of decision tree algorithm.

the output (class label), and the branches represent decision rules. A decision tree is a supervised learning ML technique that can be used for regression and classification problems, but it is mostly used for classification.

A decision tree is a two-phase process:

1. Building Phase
   - Data preprocessing
   - Splitting the dataset into training and testing set.
   - Training the classifier

2. Operational Phase
   - Building predictions
   - Computing accuracy

Decision Tree as Classifier (Figure 7.20):

```
In[49]: import pandas as pd
fromsklearn.tree import DecisionTreeClassifier
fromsklearn.model_selection import train_test_split
fromsklearn.tree import export_graphviz
fromsklearn.externals.six import StringIO[20]
fromIPython.display import Image
importpydotplus
```

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin',
'bmi', 'pedigree', 'age', 'label']21
path=pima = pd.read_csv(r"C:\pima-indians-diabetes.csv",
header = None, names =col_names)21
pima.head()21
feature_cols = ['pregnant', 'insulin', 'bmi', 'age',
'glucose', 'bp', 'pedigree']
X = pima[feature_cols]
y = pima.label
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.3, random_state = 1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
dot_data = StringIO()21
export_graphviz(clf, out_file=dot_data, filled=True,
rounded=True, special_characters=True, feature_names =
feature_cols, class_names=['0', '1'])21
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())21
graph.write_png('Pima_diabetes_Tree.png')21
Image(graph.create_png())21
Out[49]:
```

3. Random Forest Algorithm

The random forest algorithm builds decision trees on datasets and then gets the prediction from each of them and eventually chooses the best prediction by voting.[21] This method is better than a single decision tree as it lowers the overfitting.

Implementation of Random Forest Algorithm:

i. Select random samples from the dataset.
ii. A decision tree is constructed for every sample and prediction is also calculated from every decision tree.
iii. Voting is carried out for every prediction.
iv. The highest voted prediction is considered as the final prediction.

Advantages:

- Overfitting is reduced as the result of various decision trees is combined.
- Random forest has less variation as compared to the decision tree.
- It results in good accuracy.

Disadvantages:

- It is time-consuming compared to a decision tree.
- Complexity is a major disadvantage.

In[50][19,22]**:**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
headernames = ['sepal-length', 'sepal-width', 'petal-length',
'petal-width', 'Class']
data = pd.read_csv("iris_new.csv", names = headernames)
data.head()
X = data.iloc[:, :-1].values
y = data.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30)
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
confusion = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Out[51][21,22]**:**

```
Confusion Matrix:
[[14  0  0]
 [ 0 13  2]
 [ 0  0 16]]
Accuracy: 0.9555555555555556
```

4. SVM

   The SVM model is simply a representation of different classes in a multidimensional hyperplane.[21] In the SVM, to minimize the error, the hyperplane is formed in an iterative manner.[21]

   Important terms in SVM:

   - **Support Vectors** – These are the data points that are of closest proximity to the hyperplane.[21]
   - **Hyperplane** – It is the decision plane that divides the two classes of different objects.
   - **Margin** – It can be defined as the distance between the two lines at the data points of the different groups.[21]

In[52][23]**:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
```

```
# reading csv file and extracting class column to y.
x = pd.read_csv("breast_cancer.csv")
a = np.array(x)
y = a[:, 3] # classes having 0 and 1
x = np.column_stack((x.malignant, x.benign))
x.shape
from sklearn import svm # "Support Vector Classifier"
clf = svm.SVC(kernel="linear")
# fitting x samples and y classes
clf.fit(x, y)
clf.predict([[120, 990]])
Out[52]:array([1.])
```

7.10 Python IDEs for Data Science

IDE is an acronym for "integrated development environment" which combines all the different facets of writing code — code editor, compiler/interpreter, and debugger — in a single application. With IDEs, it is easier to initiate programming new applications easily as we do not need to set up various utilities and learn about different tools for running a program.

The debugger tool in the IDEs helps us evaluate the variables and inspect the code to isolate the errors.

Some of the Python IDEs that are used for data science are given as follows:

1. Jupyter Notebook

   Jupyter Notebook (formerly IPython Notebooks) is an open-source, web-based, interactive computational application to create and share equations, live code, visualization, and narrative text documents. A Jupyter Notebook document is a JavaScript Object Notation document that follows a schema and contains an ordered list of input/output cells. In Jupyter Notebook, you can write and test your code snippets independently. The Markdown editor (for text explanations) and visualization make it easier to interpret the notebook. With the different magic commands along with the range of notebook extensions, one can easily increase the functionalities of Jupyter. The notebooks can be in various formats from .py files to pdf and also as slides for presentation. The Jupyter Notebook supports various other languages rather than Python – Scala, R, Julia, etc.

2. Spyder

   Spyder is an open-source, strong Python IDE specially designed for data science. Spyder can be combined with several different Python modules, such as NumPy, SciPy, pandas, IPython, etc. It provides a distinctive combination of editing, analysis, debugging,

and portraying features data exploration, thorough inspection, and much more. The static code analysis property identifies style problems, possible bugs, and other quality issues in your code without executing the code. This is performed using the PyLintAnalyzer. A Profiler pane is used to evaluate the statements in the code that need to be improved to boost the efficiency of the code.

3. PyCharm

   PyCharm is a competent Python IDE that includes features such as an advanced debugger, code completion, and code inspection. It also supports web programming. It is mainly used when multiple scripts are required to communicate with each other. PyCharm is particularly useful in ML as it can incorporate libraries such as Matplotlib, scikit-learn, NumPy, pandas, etc. PyCharm IDE also supports Structured Query Language (SQL) written code and related database languages. It allows quick code completion regardless of packages and even has a few shortcuts for quick refactoring.

   The version control system integration comes with a lot of external plug-in support.

4. Visual Studio Code

   Visual Studio Code is an open-source code editor, which can also be used as an IDE. IntelliSense syntax in Visual Studio Code makes it capable of autocompletion, class, and functions type hints. Extracting methods, adding or removing imports that are not used is possible due to the refactor feature in Visual Studio Code. The Python Debug Interactive window provides an interactive debugging experience. Downloading libraries such as Django and Flask is easier in Visual Studio to build web applications. VS is also compatible with other tools such as SQL, NET, and more.

7.11 Summary

In this chapter, we learned to use Python for data science. At first, we understood the basics of Python necessary to begin the journey of data science using Python. The different data structures, list, tuple, dictionary, and string, are covered. Python libraries contribute ease to the data scientists and their aspirants. The most important Python libraries that are widely used in data science –NumPy and pandas– are studied in detail. We learned how the DataFrames in pandas are manipulated in various ways and the NumPy arrays and their advantages. We learned how EDA helps us to recover the primary characterizes in the dataset and thus making it competent to work efficiently with any ML algorithm. Time series helps us manipulate the date and time in Python using

various methods. An introductory section on ML was understood to build in different models that can be used in data science. Various clustering models – k-means, agglomerative, DBSCAN clustering, and dimensionality reduction methods – PCA, and manifold learning pushed the importance of efficient models in data science. Different ML algorithms, such as KNN, random forest, decision tree, and SVM, are also discussed. To end the section, we also got an overview of different IDEs that can be used for python in data science.

Exercise

1. How is the negative index used in Python? Give two examples.
2. List the differences between Python lists and NumPy arrays.
3. Write a Python code to create a DataFrame:
   - Using a list of tuples.
   - Using a dictionary of lists.

   Also, sort all the DataFrames in descending order by index values.
4. Create a one-dimensional NumPy array by using the arrange function.
5. Differentiate between KNN and random forest algorithm.
6. Create a one-dimensional array (arr1) and perform the following math operations on the same array:
   - Add 6 to every element in the array.
   - Then multiply it by 2
   - Create a new array (arr2) by including the numbers divisible by 7 from the modified arr1.
7. Perform deletion and update operation on the tuple.
8. Explain in detail principal component analysis.
9. Write a short note on the types of clustering.
10. Distinguish between supervised learning and unsupervised learning.

References

1. Campesato Oswald. 2020. *Python3 for Machine Learning*. Dules: Mercury Learning and Information.
2. VanderPlas Jake. 2017. *Python Data Science Handbook*. Sebastopol: O'Reilly Media.
3. Embarak Ossama. 2018. *Data Analysis and Visualization Using Python*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-4109-7.
4. Swamynathan Manohar. 2019. *Mastering Machine Learning with Python in Six Steps*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-4947-5.

5. Nelli Fabio. 2018. *Python Data Analytics*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-3913-1.
6. 2018. Python NumPy. GeeksforGeeks. https://www.geeksforgeeks.org/python-numpy/ (accessed 31 August 2020).
7. Verdhan Vaibhav. 2020. *Supervised Learning with Python*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-6156-9.
8. Vermeulen Andreas François. 2018. *Practical Data Science*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-3054-1.
9. McKinney Wes. 2012. *Python for Data Analysis*. Sebastopol: O'Reilly Media.
10. Skiadas Christos H., Bozeman James R. 2018. *Data Analysis and Applications 1*. Hoboken: John Wiley and Sons.
11. Sawant Puja, Khandare S. N. 2016. Clusterization and visualization techniques in data mining. *International Journal of Research in Computer & Information Technology (IJRCIT)* 1: 2.
12. 2017. Unsupervised Learning. Data Science in Python. https://python-data-science.readthedocs.io/en/latest/index.html (accessed 8 August 2020).
13. Clustering in data mining. Data Flair. https://data-flair.training/blogs/clustering-in-data-mining/ (accessed 22 August 2020).
14. Brownlee Jason. 2018. How to model volatility with ARCH and GARCH for time series forecasting in Python. Time Series. Machine Learning Mastery. https://machinelearningmastery.com/develop-arch-and-garch-models-for-time-series-forecasting-in-python/
15. Müller Andreas C., Guido Sarah. 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. Sebastopol: O'Reilly Media.
16. Goel Vishab. 2018. Building a simple machine learning model on breast cancer data. Towards Data Science. https://towardsdatascience.com/building-a-simple-machine-learning-model-on-breast-cancer-data-eca4b3b99fa3 (accessed 26 August 2020).
17. England Aaron, Alaudeen Mohamed Noordeen, Chopra Rohan. 2019. *Data Science with Python*. Birmingham: Packt.
18. 2017. NumPy. Ok, Panic. https://okpanico.wordpress.com/category/linguaggi/numpy/ (accessed 18 August 2020).
19. Introduction. Scikit Learn. Tutorialspoint. https://www.tutorialspoint.com/scikit_learn/scikit_learn_quick_guide.htm (accessed 15 August 2020).
20. Navlani Avinash. 2018. Decision tree classification in Python. DataCamp. https://www.datacamp.com/community/tutorials/decision-tree-classification-python (accessed 30 August 2020).
21. Basics. Machine Learning with Python. Tutorialspoint. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_quick_guide.htm (accessed 12 August 2020).
22. Kumawat Dinesh. 2019. Introduction to logistic regression – Sigmoid function, code explanation. AnalyticSteps. https://www.analyticssteps.com/blogs/introduction-logistic-regression-sigmoid-function-code-explanation (accessed 25 August 2020).
23. Classifying data using Support Vector Machines (SVMs) in Python. tutorialspoint.dev. https://tutorialspoint.dev/language/python/classifying-data-using-support-vector-machinessvms-in-python (accessed 21 August 2020).

8

Data Science Tool: R

Data science[1,2] has developed as the utmost popular domain for the last few years. The main reason for this is that there is a growing necessity to evaluate and build data insights from the data for better decision-making. R is one of the most widely used tools by statisticians to churn such data. R is a programming language that enables statistical computation which is highly used for data mining and data analysis. R provides an extraordinary platform to evaluate, extract, and visualize the insight.

In this chapter, we will go through various techniques provided by R for data science. The R programming language is a vast topic to be covered, and this chapter will give an overview of it.

8.1 Reading and Getting Data into R

R helps users to work seamlessly with the directories with the aid of certain preset functions that use the directory path as an argument or return the existing directory path. The directory functions used in R are given below:

1. **getwd()** – This built-in function in R returns the current directory path on which the R is functioning.
2. **setwd()** – Using this function, one can change the working directory by just specifying the directory path as an argument in the function.

8.1.1 Reading Data into R

Usually, the data to be read and interpreted are indeed contained in a file but is available beyond the R framework. Therefore, in such situations, importing data into R is a necessary task. CSV, Excel, JSON (JavaScript Object Notation), text, etc. file formats are supported by R. The formats provided by R are CSV, JSON, Excel, text, XML (Extensible Markup Language), etc. Mostly, the tabular file format is used for reading data into R.

Reading data from files:

One of the most common functions that are used to read data is the read.table() function.

DOI: 10.1201/9780429443237-11

read.table() function is used to read tabular data[3] within a text file.

read.table() consists of a lot of parameters in it. Some of the most widely used are given below:

- **file** – Filename or the path.
- **header** – A logical value for the header line. True sets the first line for the header names, and False omits the header line and sets the first row as the first line.
- **col.names** – This parameter is used to set the name of the columns if the header is set to "False".
- **nrows** – The number of rows to be included in the table from the entire dataset.
- **comment.char** – This takes a character string as an input that specifies the comment.
- **stringAsFactors** – If set to True, all the strings are considered as factors.
- **colClasses** – A vector that specifies the class of all columns.

Input:

```
> tips<-read.table("tips.txt", nrows-120)
```

Another popular built-in function to read tabular data into R is the read.csv() function. This function is very similar to the read.table() function; the difference is just that the separator is specified using commas in read.csv() and spaces in read.table().

Input:

```
info <- read.csv("Tips.csv")
```

One important thing to keep in mind while reading tabular data into R is the size of the files. Small- to medium-sized files can be read without passing any parameters while using the function. But when handling larger datasets, certain points need to be considered as it affects the memory usage. The following aspects should be considered:

- Calculate an approximate memory essential to store the dataset.
- Specifying the number of rows (nrows) will make a significant change in the memory usage.
- Time consumption can be reduced by specifying the class using the colClasses argument.
- If your data file has no comments, specify the comment.char argument to " ", i.e., comment.char=" ".

Other most used R functions for reading data are as follows:

1. **readLines()** – readLines() is used to read lines from a text file.
   Input:

```
> data<-file("data_science.txt", "r")
> w<-readLines(data)
>close(data)
>w[2]
```

Output:[1]

```
 "Data science is related to data mining, machine learning and
big data."
```

2. **source()** and **dget()** – Both the functions, source() and dget(), are used to read R code files.
   Input:

```
source("C://data_science.R")
dget("C://data_science.R")
```

3. **load()** – This function loads all the saved objects on the workspace.
4. **unserialize()** – The unserialize() function helps reading single R objects in the binary form.

8.1.2 Writing Data into Files

R provides us with similar built-in functions as in reading data into R for writing data in R.

1. **write.table()** – write.table() is a function for writing tabular format data into files.
2. **writeLines()** – This function is used to write lines to the file.
   Input:

```
>writeLines("R programming", "data_science.txt")
```

3. **dump()** and **dput()** – These functions dump the text representation of R objects. These functions store the metadata of the objects. To remove the objects from the environment, we use the rm() function.
4. **save()** – This function saves R objects into .rdata or .rda formats.
5. **serialize()** – The serialize() function returns R objects converted to the binary form.

8.1.3 scan() Function

The scan() function is used for scanning and reading the data as described by the what parameter. The default value for the what parameter is numeric. If it is specified as "character()" or simply " ", then all the fields in the dataset are read as strings.

Input:

```
>write.table(data_1,file = "data_1.txt", row.names = FALSE)
> data_1 <- data.frame(a1 = c(1, 2,3), a2 = c(4, 5, 6))
>write.table(data_1,file = "data_1.txt", row.names = FALSE)
> data_1 <- scan("data_1.txt", what = "character")
> data_1
```

Output:

```
[1] "a1" "a2" "1" "4" "2" "5" "3" "6"
```

8.1.4 Built-in Data Sets

Approximately, 100 datasets are included in the R package datasets. Other datasets are included in all other packages.

The below command gives the list of all the available datasets.

Input:

```
> data()
```

The built-in datasets can be loaded directly by their name.

Input:

```
>data(iris)
```

To load a dataset from a particular package, use the below syntax:

Syntax: data(dataset_name, package="package_name")

Input:

```
data("iris3", package="datasets")
```

8.2 Ordered and Unordered Factors

Factors are nothing but R objects containing categorical values. The factors are created the same as vectors and are later converted into vectors. They are mostly used when columns have limited unique values.

For example, it is better to use True or False rather than 0 and 1 as factors are self-descriptive.

Unordered factors are those that do not have any initial order levels.

Example: the color of fruits, race, etc.

Ordered factors are the factors that have a natural ordered level.

Example: good, better, best.

Factors are created using the factor() function.

Input:

```
> Ranking=c("High", "Low", "Medium", "Medium", "High", "Low")
>factor_ranking<-factor(Ranking)
>class(factor_ranking)
```

Output:[1]

```
"factor"
```

The nlevels() function gives the number of levels in the factors, .i.e., the number of unique values.

Input:

```
nlevels(factor_ranking)
```

Output: [1]

```
3
```

The levels in the factor can be stated using the levels parameter in the factor() function. Setting levels is essential in statistic modeling. By default, the levels are assigned in the alphabetical order. To specifically give an order, the levels parameter can be used.

Input:

```
>factor_ranking<-factor(Ranking, levels=c("Low", "Medium",
"High"))
>factor_ranking
```

Output:

```
[1] High Low Medium Medium High Low
Levels: Low Medium High
```

By default, factors treat levels as unordered, i.e., factors will treat "high", "medium", and "low" as categorical values until it is specified that the levels are ordered. You can achieve this using the ordered parameter of the factor() function.

Input:

```
>factor_ranking<-factor(Ranking, levels=c("Low", "Medium",
"High"), ordered=TRUE)
>factor_ranking
```

Output:

```
[1] High Low Medium Medium High Low
Levels: Low < Medium < High
```

An alternative for the ordered parameter is the ordered() function.

Input:

```
>factor_ranking<-factor(Ranking, levels=c("Low", "Medium",
"High"))
>ordered(factor_ranking)
```

Output:

```
[1] High Low Medium Medium High Low
Levels: Low < Medium < High
```

8.3 Arrays and Matrices

8.3.1 Arrays

Arrays are the main storage structures specified by a given number of dimensions. Arrays are used for the space allocations at adjacent memory locations. Vectors are the example of unidimensional arrays with just one dimension which is their length. Matrices are the example of a two-dimensional array with a certain number of rows and columns. The elements in the arrays are all of the same data type. In R, arrays are created using the vectors.

An array can be called as data objects in R, where data can be stored multidimensionally.

8.3.1.1 Creating an Array

An array is constructed using the *"dim"* parameter values and the vectors inputted in the built-in function – array(). The *"dimnames"* parameter is used to give names to the rows, columns, and matrices.[4]

Input:

```
>[2] vector1 <- c(1,2)
> vector2 <- c(4,5)
> res <- array(c(vector1,vector2),dim =
c(2,2,1),dimnames=list(row.names,column.names,matrix.names))
>print(res)
```

Output[5]:

```
Matrix1
   COL1 COL2
ROW1  1  4
ROW2  2  5
```

8.3.1.2 Accessing Elements in an Array

Indices of various dimensions are used for accessing an array. Indices for various dimensions are differentiated using commas. Any combination of names or positions of the elements can be defined by different components.

The elements in the matrix can be accessed by specifying the row, column, and matrix number.[4]

Input:

```
>print(res[1,2,1])
```

Output:

```
4
```

8.3.1.3 Array Manipulation

Arrays can be manipulated in many ways in R. apply() is a built-in function in R and is used to perform calculations over the array elements.[6]

Syntax: apply(x, margin, fun)[3]

- **x** – Array
- **margin** – A value between 1 and 2 that states the area of manipulation.
  c(1) – Manipulation to be performed on rows.
  c(2) – Manipulation to be performed on the column.
  c(1,2) – Manipulation to be performed on rows as well as columns.
- **fun** – The type of function to be performed on the array elements.

Input:

```
> result <- apply(res, c(1), sum)
>print(result)
```

Output:

```
ROW1 ROW2
5  7
```

8.3.2 Matrices

Matrices are R objects where the elements are structured in a rectangular two-dimensional pattern.[4] All the elements in the matrix are of the same data type. A matrix can be constructed for characters or logical values, but mostly, matrices that contain numeric elements are highly used in mathematical computations.

8.3.2.1 Creating a Matrix

A matrix is created using the matrix() function in R.
Syntax: matrix(data, nrow, ncol, byrow, dimnames)

- **data** – Input vector
- **nrow** – Number of rows
- **ncol** – Number of columns
- **byrow** – Arrangement of the elements. True – elements to be arranged in rows.
- **dimnames** – Name allocated for rows and columns.

Input:

```
> Matrix_1<- matrix(c(3:6), nrow = 2, byrow = TRUE)[4]
>print(Matrix_1)[4]
```

Output:

```
[, 1] [, 2]
[1,]  3  4
[2,]  5  6
```

8.3.2.2 Matrix Transpose

The transpose of a matrix can be calculated using the t() function of R.
Input:

```
t(Matrix_1)
```

Output:

```
[, 1] [, 2]
[1,] 3 5
[2,] 4 6
```

8.3.2.3 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors are highly used by the data scientists as they are the core of the data science field. For example, eigenvalues and eigenvectors are very much useful in the principal component analysis which is a dimensionality reduction technique in machine learning and is highly used in the field of data science.

The "eigen()" built-in function calculates the eigenvalues and eigenvectors of a symmetric matrix.

Input:

```
> print(eigen(Matrix_1))
```

Output:

```
eigen() decomposition
$values
[1] 9.2169906-0.2169906
$vectors
      [,1]         [,2]
[1,] -0.5410795-0.7792516
[2,] -0.8409715 0.6267112
```

8.3.2.4 Matrix Concatenation

Concatenation of matrices can be performed using either rows or columns, resulting in a bigger matrix. It is essentially a data manipulation procedure where the matrices concerned have to be of appropriate size for the operation to be carried out. The matrices may be concatenated horizontally or vertically as per the requirements.

The rbind() and cbind() functions are used to concatenate a row and a column, respectively, to a matrix.

Input:

```
> B = matrix(c(12, 13), nrow = 1, ncol = 2)
>print(rbind(Matrix_1,B))
A = matrix(c(14, 15), nrow = 2, ncol = 1)
>print(cbind(Matrix_1,A))
```

Output:

```
[, 1] [, 2]
[1,]     3   4
[2,]     5   6
[3,]     12  13
     [,1] [,2] [,3]
[1,]  3  4  14
[2,]  5  6  15
```

8.4 Lists and Data Frames

8.4.1 Lists

The R list is a standard entity that consists of an ordered set of objects. Lists are single-dimensional, diverse data structures. A list can be of vectors, characters, matrices, etc.

In R, list indexing begins with 1 instead of 0 as in other programming languages.

8.4.1.1 Creating a List

The list() function is used to create a list in R. names(list_name) is used to specify names for the inner elements of the list.

Input:

```
> list_1 <- list("Blue", c(2,3,1), TRUE)
>names(list_1) <- c("Colour", "A_Vector", "Logical Value")
>print(list_1)
```

Output:

```
$Colour
[1] "Blue"
$A_Vector
[1] 21 32 11
$`Logical Value`
[1] TRUE
```

8.4.1.2 Concatenation of Lists

Two or more lists can be concatenated using the concatenation operator – "c()", in R.

Input:

```
>Num=c(22,23,34)
> list_2= list("Numbers" = Num)
>list_3=c(list_1,list_2)
>print(list_3)
```

Output:

```
$Colour
[1] "Blue"
$A_Vector
[1] 21 32 11
$`Logical Value`
```

```
[1] TRUE
$Numbers
[1] 22 23 34
```

8.4.2 Data Frames

8.4.2.1 Creating a Data Frame

Data frames are R data objects where the elements are stored in a tabular format.

Characteristics of the data frame:

- The column names cannot be empty.
- The row names of a data frame must be unique.
- The data type of the elements in the data frame is a numeric or character. A data frame can also be a factor type in R.
- Every column should have the same number of items.[7]

A data frame is a list with a data type(class) as data.frame. A data frame is created using the "data.frame()" function in R. It can be said as an equal-length list.[8]

Input:

```
>[4]Employee <- data.frame(Name=c("Robert","Catherine"),
Gender = c("Male","Female"), Age=c(24,36))
>print(Employee)
```

Output:

```
      Name Gender Age
1   Robert   Male  24
2   Catherine Female  36
```

Using the data.matrix(DataFrame_name) function, a data frame can be converted to a matrix.

8.4.2.2 Accessing the Data Frame

Syntax: dataframe_name $ column_name

Input:

```
Employee$Name
```

Output:

```
[1] "Robert" "Catherine"
```

A data frame can also be accessed the same as that of the matrix

8.4.2.3 Adding Rows and Columns

Rows and columns can be added to the existing data frame using rbind() and cbind(), respectively.

Input:

```
>emp<-data.frame(Work_Experience=c(5,10))
>Employee=cbind.data.frame(Employee, emp)
>print(Employee)
```

Output:

```
       Name Gender Age Work_Experience
    1   Robert   Male  24               5
    2 Catherine  Female  36              10
```

Input:

```
> Employee=rbind.data.frame(final, c("Elle", "Female", 32,13))
>print(Employee)
```

Output:

```
       Name Gender Age Work_Experience
    1   Robert   Male  24               5
    2 Catherine Female  36              10
    3      Elle Female  32              13
```

8.5 Probability Distributions

A lot of probability distribution operations are available in R, but in this section, we will go through normal distributions in detail.

Using the command – help("Distributions"), one can go through all the distributions available in R.

Four commands are common in all distributions. Each command returns a functionality.

- **d** – Height of the density function.[9]
- **p** – Cumulative density function.[9]
- **q** – Quantiles.[9]
- **r** – Random generated numbers.[9]

8.5.1 Normal Distribution

In R, four built-in functions are used for normal distributions.[9] help(Normal) returns the list of functions related to normal distributions. They are as follows:

i. dnorm(x, mean, sd)
ii. pnorm(x, mean, sd)
iii. qnorm(p, mean, sd)
iv. rnorm(n, mean, sd)

The parameters used in the above functions are given as follows[4]:

- **x** – Number vector
- **mean** – Mean of the vector
- **sd** – Standard deviation of the vector
- **p** – Probability vector
- **n** – Sample size

i. **dnorm()** – The dnorm() function calculates the height of the density function.
Syntax: dnorm(x, mean, sd)
Input:

```
> a<-seq(-20,+20,by=0.2)
> b <- dnorm(a, mean(a), sd(a))
>plot(a, b)
```

Output (Figure 8.1):

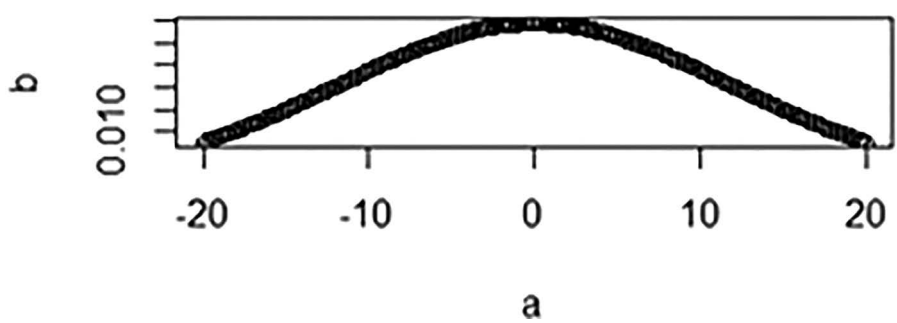


FIGURE 8.1
Normal distribution using the dnorm() function.

ii. **pnorm()** – The pnorm() function calculates the cumulative density function.

Syntax: pnorm(x, mean, sd)

Input:

```
> c <- pnorm(a, mean(a), sd(a))
>plot(a, c)
```

Output (Figure 8.2)**:**

iii. **qnorm()** – The qnorm() function can be stated as the inverse of the pnorm() function. It is used to calculate quantiles of the normal distribution.

Syntax: qnorm(p, mean, sd)

Input:

```
> a<-seq(0,1,by=0.02)
> d <- qnorm(a, mean(a), sd(a))
>plot(a, d)
```

Output (Figure 8.3)**:**

iv. **rnorm() –** rnorm() produces random numbers which are normally distributed.

Input:

```
> x<-rnorm(100)
>hist(x)
```

Output (Figure 8.4)**:**

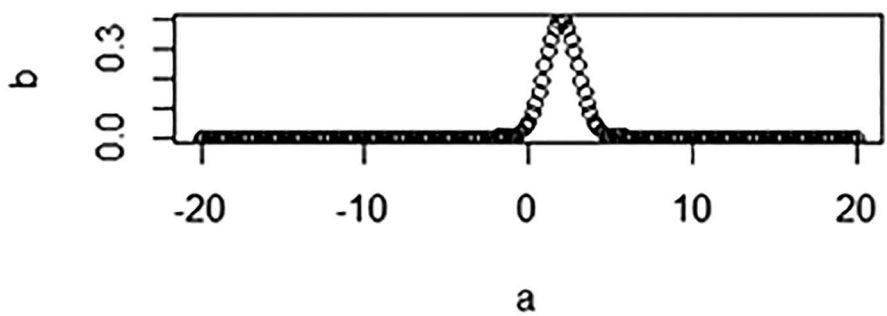


FIGURE 8.2
Normal distribution using the pnorm() function.

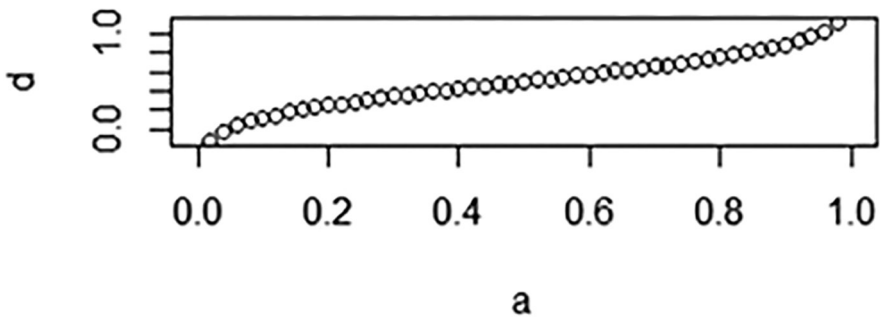


FIGURE 8.3
Normal distribution using the qnorm() function.

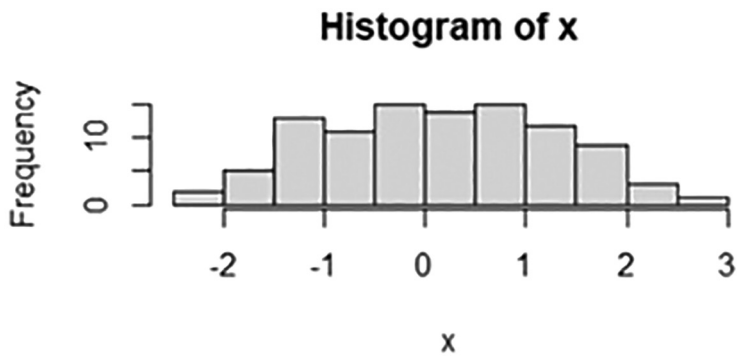


FIGURE 8.4
Normal distribution using the trnorm() function.

8.6 Statistical Models in R

The R language offers an interlinked collection of services that make the functioning of statistical models fairly easy. The performance by statistical models is limited, and you need to look for information by calling the functions available for extraction.

Before exploring statistical models and their fitting,[10] let us first learn about the basic terminologies related to the statistical models. The independent variable (explanatory variable) and the dependent variable are sometimes called a feature and target, respectively. The matrix of features is generally known as the design matrix. Also, the estimated parameters are referred to as weights or coefficients in the data science field.

8.6.1 Model Fitting

Suppose we have a target "y" that can be explained using the features x_1, x_2, etc., we will take the example of the popular iris dataset from the datasets package.

In R, before using the dataset from a certain package, the package is needed to be activated first. You can also get a thorough description of the dataset using the following commands:

Input:

```
>library(datasets)
>data(iris)
>?iris
```

The type of species in the iris dataset depends upon a collection of four features. Say, the target depends linearly on the sepal.length and the sepal. width. We can construct a linear model in R using the lm() function which is the built-in function to build a linear model.

Input:

```
>model1<-lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
>summary(model1)
```

Output:

```
Call:
lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
```

Residuals:

```
  Min   1Q Median   3Q   Max
-1.5561-0.6333-0.1120 0.5579 2.2226
```

Coefficients:

```
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.5262 0.4789 13.63 <2e-16 ***
Sepal.Width -0.2234 0.1551-1.44 0.152
---
```

Signif. codes:

```
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.8251 on 148 degrees of freedom
Multiple R-squared: 0.01382, Adjusted R-squared: 0.007159
F-statistic: 2.074 on 1 and 148 DF, p-value: 0.1519
```

As the result returned by the lm() function is a list, one can access specific elements of the model.

For example,. **Input:**

```
> print(model1$coefficients)
```

The tidy() function of the broom package converts the model to a dataframe. broom. tidy() makes it a lot easier to perform various tasks on the results of the linear model.

Input:

```
> result <- broom::tidy(model1)
>print(result)
```

8.6.2 Marginal Effects

The effect of the explanatory variables on the target variable can be known in R using the margins function under the margins package.

Input:

```
>library(margins)
>effect<- margins(model1)
>print(effect)
```

These marginal effects can also be plotted using plot() or ggplot(). You can plot the effects simply by passing the effects in the plot() function. But if you are using the ggplot() function, you first need to get the summary of these effects using the summary() function and then plot the variable storing the summary using the ggplot() function under the ggplot2 package.

8.7 Manipulating Objects

8.7.1 Viewing Objects

The objects that are created and used in the workspace can be viewed using the following two commands. Both the commands give the same result.

Syntax:

- objects()
- ls()

Input:

```
>objects()
```

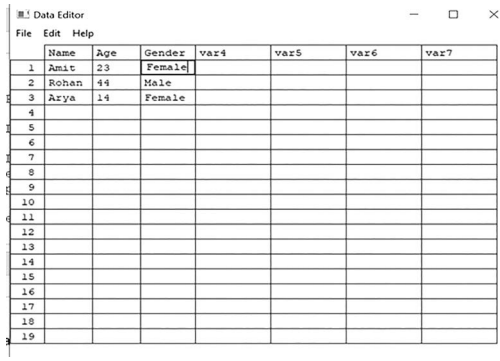


FIGURE 8.5
Text editor using the edit() function.

Output:

```
[1] "a"    "b"    "c"    "d"    "data_1"
 [6] "infert" "iris"   "iris3" "table_" "x"
 [11] "z"
```

8.7.2 Modifying Objects

The edit() function invokes a text editor, and it works with all R objects, such as the list, vector, matrix, factor, and data frame.

Input:

```
>Name = c("Amit", "Rohan", "Arya")
> Age = c(23,44,14)
> Gender = c("Female", "Male", "Female")
> data_=data.frame(Name, Age, Gender)
>edit(data_)
```

After closing this window, the modified data frame will be displayed in the console (Figure 8.5).

Output:

```
      Name Age Gender
      1  Amit  23   Male
2 Rohan  44   Male
      3   Arya  14 Female
```

8.7.3 Appending Elements

The append() function is used to append elements to the vector.

Syntax: append(a, value, after), where,

- **a** – Name of the vector.
- **value** – The element to be added.
- **after** – The position after which the element is appended.

Input:

```
> Numbers = c(1,2,3,4,5,6,7,8,9)
>append(Numbers, 15,after=4)
```

Output:

```
[1] 1 2 3 4 15 5 6 7 8 9
```

8.7.4 Deleting Objects

An element can be deleted using the minus "-" symbol. Prefixing the "-" symbol before the index deletes that element.

Input:

```
> Numbers<-Numbers[-5]
>print(Numbers)
```

Output:

```
[1] 1 2 3 4 6 7 8 9
```

remove() and rm() are used to delete objects in R. Both the function gives the same result.

Input:

```
>ls()
```

Output:

```
[1] "a" "Age" "b" "c" "d"
[6] "data_" "data_1" "Gender" "infert" "iris"
```

Input:

```
> remove("a", "Age")
>ls()
```

Output:

```
[1] "b" "c" "d" "data_" "data_1"
[6] "Gender" "infert" "iris" "iris3" "Name"
```

To delete all the objects in the workspace, the following command is used[7]:

Input:

```
>rm(list=ls())
>ls()
```

Output:

```
character(0)
```

8.8 Data Distribution

When a dataset is applied to a model, it predicts a variety of values.[12,13] These models can also be used to forecast the distribution of the data items.

The distribution includes details on the probabilities correlated with the data points. Distributions, for example, show the median values (the common values), the mean values (best estimate values), etc. Distribution can also be thought of as the border conditions of the values of a variable. We will go through the visualizing of distributions and statistics in distributions in the coming sections. Also, one can refer to the Probability Distributions section in this chapter which is covered previously.

8.8.1 Visualizing Distributions

A distribution can be instantly insightful by visualizing it.

Some of the commonly used functions for visualizing distributions are as follows:

- **dotplot()** – The built-in function dotplot() is also called as the dotchart. dotplot() is similar to the bar chart, the difference is just that the bars are substituted with dots.[4]
- **hist()** – The histogram depicts the frequency of variable values within ranges.
- **densityplot()** – This built-in function under the lattice package is used for the distribution of the numerical values.

8.8.2 Statistics in Distributions

The most useful characteristics of the distribution can be defined with common statistics.

Distributions can tell a lot about the statistical computations using the typical functions used almost everywhere in the data science field.

- **median()** – This function returns the median value of the feature.
- **mean()** – This function returns the best-estimated value that is useful to predict future values.
- **max()** – This function returns the maximum value among the values in the feature.
- **min()** – This function returns the minimum value from the feature values.

Certainty plays a very important role while predicting the target variable using the explanatory variable. As the variation among the data points increases, the uncertainty to forecast the values in the future increases. To reduce the uncertainty among the various variable values in the distribution, the following functions are used.

- **range()** – This built-in function in R returns the minimum and the maximum value among the numeric type variable.
- **var()** – The var() function returns the variance of the given explanatory variable.
- **sd()** – The sd() function returns the standard deviation of the required variable.
- **iqr()** – This function gives the interquartile range.
- **bwplot()** – The bwplot() function plots a box-whisker plot which gives us a summary of the data that includes min, max, first quartile, median, and the third quartile.

8.9 Summary

In this chapter, we got an outline of R in data science. Firstly, we went through various techniques to read and write data into R. Factors play a vital role in statistical modeling and analysis. Then, we also studied the creation and manipulation of arrays, matrices, lists, and data frames. R provides various probability distributions; in this chapter, we went through normal distributions in detail. Statistical models play a vital role in building models in data science. We covered how these models can be fitted to certain data properly. Techniques to manipulate objects are also covered in the chapter. Data distributions make it easier to get insights into the data. We studied various built-in functions provided by R that can be used for various distributions.

Exercises

1. Write an R program to create a vector containing 20 random integer values between 10 and 20.
2. Write an R program to add a new row and extract the second row from a data frame.
3. Write an R program for the concatenation of vectors.
4. Explain the different data structures in R.
5. Explain data frames and their operations in detail.
6. What are the characteristics of the data frame?
7. Write a short note on visualizing distributions.
8. What is the role of edit()?
9. List down the commands in distribution.

References

1. Peng Roger D. 2015. *R Programming for Data Science*. Victoria: Leanpub.
2. Lantz Brett. 2019. *Machine Learning with R: Expert Techniques for Predictive Modeling*. Birmingham: Packt.
3. Field Andy, Miles Jeremy, Field Zoe. 2000. *Discovering Statistics Using R*. Sage Publications.
4. R- quick guide. tutorialspoint. https://tutorialspoint.com/r/r_quick_guide.htm (accessed 25 October 2020).
5. 2019. R-arrays. Tutorial and Examples. https://www.tutorialandexample.com/r-arrays/ (accessed 26 October 2020).
6. Arrays in R. R programming Tutorial. https://www.educba.com/arrays-in-r/ (accessed 27 October 2020).
7. R Data Structures. R Tutorials. TechVidvan. https://techvidvan.com/tutorials/r-data-structures/ (accessed 30 October 2020).
8. Brouwer De Philippe. 2020. *The Basics of R. The Big R-Book: From Data Science to Learning Machines and Big Data*. Wiley Online Library. https://doi.org/10.1002/9781119632757. Ch 4.
9. Brouwer De Philippe. 2020. *Elements of Descriptive Statistics. The Big R-Book: From Data Science to Learning Machines and Big Data*. Wiley Online Library. https://doi.org/10.1002/9781119632757. Ch 8.
10. Walkowiak Simon. 2016. *Big Data Analytics with R*. Birmingham: Packt.
11. Stowell Sarah. 2014. *Using R for Statistics*. Springer Nature. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-0139-8.
12. Mailund Thomas. 2017. *Beginning Data Science in R*. California: Apress.
13. Wickham Hadley, Grolemund Garrett. 2016. *R for Data Science*. Sebastopol: O'Reilly Media.

9

Data Science Tool: MATLAB

We all know that data science is all about traversing different forms of data; analyzing important insights into the data; and making better decisions through it. Moreover, data science can also be called a process capable of telling a captivating story from the data. Once the required data is explored, to gain insights into them, various predictive analysis techniques from machine learning (ML) and deep learning are highly used as these techniques help in enhancing the performance of the system. Many such techniques are used by various professionals who have expertise in their field but are not actual programmers. In such circumstances, MATLAB comes into the picture. MATLAB is adaptive to various workflows for exploring, analyzing, and visualizing scientific data. One of the prominent uses of MATLAB is that it has built-in functions for analyzing and modeling data that bridges the gap between various domains. Hence, one can focus on their expertise and gain statistical insights using MATLAB.

9.1 Data Science Workflow with MATLAB

A classic data science project contains a process[1,2] that can be divided into three stages:

1. Data analysis
2. ML
3. Manipulating with the result.

We will go through each stage in this section:

1. Data Analysis

In this stage, the primary goal is gaining more insights into your existing data. An ideal data analysis stage comprises the following substages:

1. Aim

It is important to set your basic goals at the beginning of data analysis. It makes the data scientist's work a lot easier when they

DOI: 10.1201/9780429443237-12

have their goals set following what insights should they likely produce from the data and also, how can they use this knowledge in the future for analysis. For example, the common goals are to obtain descriptive statistics and business metrics and to build models using predictive analysis.

2. Collecting data

Once you have your goals set, you can proceed with identifying the types of data that have to be used in the project and also, the sources from where one can get the data. Based on your project, there might not be the need to obtain the new data from various experiments or conducting surveys, and one can just adopt the existing data from various data sources, such as the data files, databases, or streaming data.

3. Accessing data

When the required data for the project is obtained, one needs to access the data to perform analysis on it. Here comes the MATLAB. MATLAB provides various methods to access the data from the data sources, and it depends on its location, the type of the data, and the size of the data. It is comparatively easier to load and access smaller data files in MATLAB, and they can be loaded directly. MATLAB can efficiently extract data from databases and generate simulated data. Huge datasets can be clustered and then imported into MATLAB sample-wise or subsets.

4. Exploration and Visualization

Data Exploration

Once you are all set to explore the data, it is important to identify and fix the issues existing in data for efficient exploration in the further process. Your data source may contain missing observations, or it may have any corrupted data. Another common issue when working with datasets is the duplication of variables and their values. Apart from this, your data may have other issues that will create problems in accessing the data. If these issues are not fixed, it will intricate the analysis and generate ambiguous outcomes.

Next, look for the organization of your data; If it is a text file, look for delimited unstructured data. Look for schemas in databases. Web files mainly use the JSON (JavaScript Object Notation) format.

MATLAB provides the user with various functions, such as grouping functions that can be used to gain statistical and summary insights which further comprises information regarding the distributions, outliers, and range of the data variables and data as a whole.

Data Visualization

Data visualization[3,4] with the help of various plots such as the histograms or the scatter plots are used to find the distributions, the correlations between the variables, various models, and clusters help to find the relationship between different variables and different observations.

2. ML

   As we all know, ML consists of a set of algorithms that models a relationship between various variables in the data. ML is often an additional step and is highly used in almost all data science projects. ML algorithms use a typical model and adapt or train this model over the required datasets for future predictions. ML algorithms consist of the following steps:

   i. Selection of the model and variables
   ii. Preprocessing the data
   iii. Training the model
   iv. Testing model using new data.

3. Results

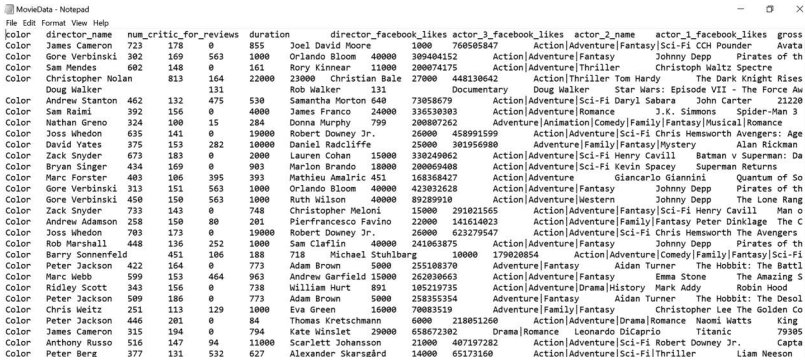
   After the completion of the above two stages – data analysis and ML – comes the results. Once you get your results from various methods and models, it is better to do a recheck on your process to avoid any mistakes. These results can now be pinned down to conclusions and produce qualitative outcomes. You can put over these results using visualizations or presentations or even in a report format. Data scientists often prefer to show up their outcomes regarding a project using the dashboards.

9.2 Importing Data

We have learned from the data science workflow that once the required source of data is acquired, we first need to import that data into MATLAB to access the data and perform our calculations on them. This section precisely covers all the important aspects related to importing the data into MATLAB.

9.2.1 How Data is Stored

We all know that data is everywhere, and it is the job of the data scientist to gain insights into any sort of data and be a part of better decision-making. Here, we will go through the different types of storing data into MATLAB.


MovieData - Notepad

File Edit Format View Help

```
Color   director_name   num_critic_for_reviews  duration        director_facebook_likes actor_3_facebook_likes  actor_2_name    actor_1_facebook_likes  gross
Color   James Cameron   723     178     0       855     Joel David Moore        1000    760505847       Action|Adventure|Fantasy|Sci-Fi CCH Pounder     Avata
Color   Gore Verbinski  302     169     563     1000    Orlando Bloom   40000   309404152       Action|Adventure|Fantasy        Johnny Depp     Pirates of th
Color   Sam Mendes      602     148     0       161     Rory Kinnear    11000   200074175       Action|Adventure|Thriller       Christoph Waltz Spectre
Color   Christopher Nolan       813     164     22000   23000   Christian Bale  27000   448130642       Action|Thriller Tom Hardy       The Dark Knight Rises
        Doug Walker             131             Rob Walker      131             Documentary     Doug Walker     Star Wars: Episode VII - The Force Aw
Color   Andrew Stanton  462     132     475     530     Samantha Morton 640     73058679        Action|Adventure|Sci-Fi Daryl Sabara    John Carter     21220
Color   Sam Raimi       392     156     0       4000    James Franco    24000   336530303       Action|Adventure|Romance        J.K. Simmons    Spider-Man 3
Color   Nathan Greno    324     100     15      284     Donna Murphy    799     200807262       Adventure|Animation|Comedy|Family|Fantasy|Musical|Romance
Color   Joss Whedon     635     141     0       19000   Robert Downey Jr.       26000   458991599       Action|Adventure|Sci-Fi Chris Hemsworth Avengers: Age
Color   David Yates     375     153     282     10000   Daniel Radcliffe        25000   301956980       Adventure|Family|Fantasy|Mystery        Alan Rickman
Color   Zack Snyder     673     183     0       2000    Lauren Cohan    15000   330249062       Action|Adventure|Sci-Fi Henry Cavill    Batman v Superman: Da
Color   Bryan Singer    434     169     0       903     Marlon Brando   18000   200069408       Action|Adventure|Sci-Fi Kevin Spacey    Superman Returns
Color   Marc Forster    403     106     395     393     Mathieu Amalric 451     168368427       Action|Adventure        Giancarlo Giannini      Quantum of So
Color   Gore Verbinski  313     151     563     1000    Orlando Bloom   40000   423032628       Action|Adventure|Fantasy        Johnny Depp     Pirates of th
Color   Gore Verbinski  450     150     563     1000    Ruth Wilson     40000   89289910        Action|Adventure|Western        Johnny Depp     The Lone Rang
Color   Zack Snyder     733     143     0       748     Christopher Meloni      15000   291021565       Action|Adventure|Fantasy|Sci-Fi Henry Cavill    Man o
Color   Andrew Adamson  258     150     80      201     Pierfrancesco Favino    22000   141614023       Action|Adventure|Family|Fantasy Peter Dinklage  The C
Color   Joss Whedon     703     173     0       19000   Robert Downey Jr.       26000   623279547       Action|Adventure|Sci-Fi Chris Hemsworth The Avengers
Color   Rob Marshall    448     136     252     1000    Sam Claflin     40000   241063875       Action|Adventure|Fantasy        Johnny Depp     Pirates of th
Color   Barry Sonnenfeld        451     106     188     718     Michael Stuhlbarg       10000   179020854       Action|Adventure|Comedy|Family|Fantasy|Sci-Fi
Color   Peter Jackson   422     164     0       773     Adam Brown      5000    255108370       Adventure|Fantasy       Aidan Turner    The Hobbit: The Battl
Color   Marc Webb       599     153     464     963     Andrew Garfield 15000   262030663       Action|Adventure|Fantasy        Emma Stone      The Amazing S
Color   Ridley Scott    343     156     0       738     William Hurt    891     105219735       Action|Adventure|Drama|History  Mark Addy      Robin Hood
Color   Peter Jackson   509     186     0       773     Adam Brown      5000    258355354       Adventure|Fantasy       Aidan Turner    The Hobbit: The Desol
Color   Chris Weitz     251     113     129     1000    Eva Green       16000   70083519        Adventure|Family|Fantasy        Christopher Lee The Golden Co
Color   Peter Jackson   446     201     0       84      Thomas Kretschmann      6000    218051260       Action|Adventure|Drama|Romance  Naomi Watts     King
Color   James Cameron   315     194     0       794     Kate Winslet    29000   658672302       Drama|Romance   Leonardo DiCaprio       Titanic         79305
Color   Anthony Russo   516     147     94      11000   Scarlett Johansson      21000   407197282       Action|Adventure|Sci-Fi Robert Downey Jr.       Capta
Color   Peter Berg      377     131     532     627     Alexander Skarsård      14000   65173160        Action|Adventure|Sci-Fi|Thriller        Liam Neeson
```

FIGURE 9.1
Dataset viewed as a text file.

The common formats in which the data that we use are stored are binary numbers, images, videos, text, audio, etc., but there are many more formats, and a data scientist can be in a situation where they have to analyze any type of data. No matter what type of data format we are using, they are stored in the numeric form in the files. Each file should be named with a distinct name, and it has an extension that specifies the data format of the file. For example, if your data consists of numbers or plaintexts, this data is stored in a simple file with the extension .txt indicating that it is a text file.

When you open a dataset[5–7] in a text file format, you will see that each row consists of many values that are parted using a tab. As shown in Figure 9.1, it is a tedious task to visualize your data when opened in a text file as the values are not properly aligned in accordance with the respective columns. Although it is difficult for the users to interpret such data, the computer reads it efficiently.

As the .txt file parts the values using a tab, a .csv file uses a comma to separate the values from each other. The full form of the abbreviation CSV is itself a comma-separated file. One can use any character, such as #, %, z, etc., to separate the values and align them in a column. Such characters used to part every column is known as the delimiter. Although there are multiple delimiters, the space, comma, and tab are used often. The data stored and organized in a tabular format is very usually used. As we know, the tabular format stores the data in the form of rows and columns. For such data, the spreadsheet software is highly used. Also, to use such type of data, a language or a tool that is compatible with such data formats needs to be used so that it can access the data and interpret it.

When you open a .jpg file or.mp3 formatted file in a notepad, you will not be able to interpret the data as it would contain a chain of nonseparated numerical values and any random characters and will be completely unidentifiable. MATLAB is very helpful in withdrawing readable data from file

formats with extensions such as .jpg, .png, .mp4, and .mp3. MATLAB can access any type of data and load it into the software and provide insightful data to the user to analyze it.

9.2.2 How MATLAB Represents Data

In the last section, we learned that MATLAB can load many types of data formats. In this section, we will go through the data organization in MATLAB, i.e., how is data organized when it is loaded from a particular file format. MATLAB organizes the data in tabular format which means in rows and columns. If the data has only one value, the size of the data is denoted as (1, 1) which specifies that the data consist of one row and one column (Figure 9.2).

If the entire data is stored in a single row, such type of data is called a row vector. The below data will be called a 1-by-5 row vector (Figure 9.3).

If the entire data is stored in a single column, such type of data is called a column vector. The below data will be called a 5-by-1 column vector (Figure 9.4).

A vector is an entity that contains observations for only one measurable variable. A data file mostly has various variables and thus can be called a data file with numerous vectors. These vectors contain numbers, text, or the combination of both. By default, MATLAB supposes every variable to be a column vector and places all its observations in a row.

FIGURE 9.2
Data matrix with one value.

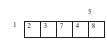


FIGURE 9.3
1-by-5 row vector.

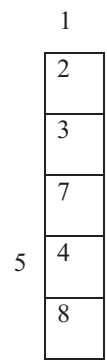


FIGURE 9.4
5-by-1 column vector.

It becomes easier to access the data elements when they are organized and stored in rows and columns using their indices. The first place is for the rows, and the second place is for the columns. For example, (2, 7) indexing will access the data elements that belong to the second row and is in the seventh column.

9.2.3 MATLAB Data Types

To import a data file to MATLAB, we use the import tool. To open the import tool, just double-click on the data file.[1] The data file will be imported and be visible on the screen. You can manually decide to take the entire dataset or to work on some of the columns (subsets). Once you import the data file into MATLAB, the MATLAB software automatically displays the data type of the variables in the cells below the column names. Most of the time, the data types are accurate, but it is better to review these automatically assigned data types. The data type specified by MATLAB tells a lot about the usage of that particular variable in the analysis. The data having the numeric data type can be used in visualizing the data or even in the required computations. Also, the data consisting of text can be used to classify the data or for assigning labels.

The data types in MATLAB can primarily be divided into four types:

1. Numbers
2. Text
3. Categories
4. Date and time

1. Numbers

   The number label is one of the most common labels used in MATLAB. This label describes using the double data type. A double data type is the data type that represents real numbers.

   Numbers can be represented in three ways in MATLAB.

   i. Single-value

   $x = 0.2$

   ii. Matrix

$$A = \begin{bmatrix} 2 & 4 & 5 \\ 8 & 7 & 3 \\ 9 & 4 & 8 \end{bmatrix}$$

   iii. Vector

$$B = \begin{bmatrix} 0.3 \\ 0.01 \\ 0.23 \end{bmatrix}$$

Note that a vector can be defined as an array of values arranged in rows or columns, and such an array of vectors is called a matrix.

2. Strings

   Strings are the text data that consists of a set of characters. Strings are quoted in inverted commas. A string can contain more than one word, i.e., a single word, and a single sentence in the inverted commas is described as strings.

   Example: name="Anna",
   Speech="I love data science".

3. Categorical data

   Categorical data seems to be similar to the strings, but it is different. Categorical data is text data, but it can take only a limited and finite number of values. Categorical data is highly used in analyzing and visualizing the variables to gain insights into them.

   Example: Suppose, we are considering the rainfall in the cities of India. In such circumstances, we might have a variable named as "states"; this variable might have repeated observations, but it will have only a finite number of unique values.

4. Date and time

   The date and time can be represented using the DateTime data type. With this data type, it is easy to perform computations on variables containing date and time. This data type represents a particular point in time. It represents the date in the format "DD-Month-YYYY" and time in the format "HH: MM: SS"

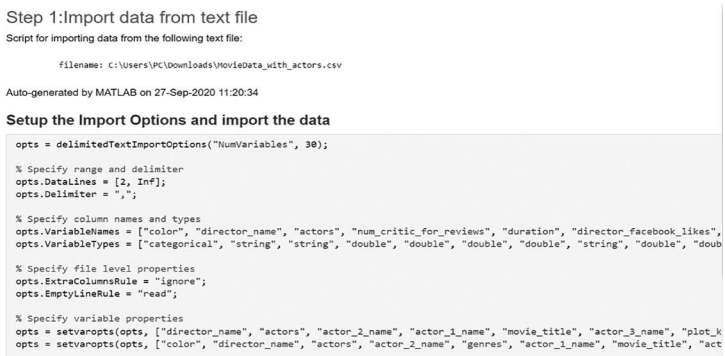
Example

Input: D.O.B=datetime (1998,6,22,5,30,0)
Output: D.O.B=22- Jun-1998 05:30:00

9.2.4 Automating the Import Process

In the previous sections, we learned about the import tool in MATLAB. The import tool gives us the information about the data in the file and also helps us to personalize the data, such as changing the variable name and changing the data type of the variable based on the needs of the required analysis. A data scientist may need to import multiple files while performing analysis. In such cases, importing all the files using the import tool is time-consuming. MATLAB provides us with an option to automate the import process. Automating the import process will not only save our time but also help us to keep this process in the documentation, so it is possible to reproduce the results in the future.

Once you are done with all the customizations of the data, you can see various options that generate the MATLAB code when you click on the import selection button. The "Generate a live script" option will generate a live script that will contain the import code for that specific data including all


Step 1:Import data from text file

Script for importing data from the following text file:

filename: C:\Users\PC\Downloads\MovieData_with_actors.csv

Auto-generated by MATLAB on 27-Sep-2020 11:20:34

Setup the Import Options and import the data

```
opts = delimitedTextImportOptions("NumVariables", 30);

% Specify range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["color", "director_name", "actors", "num_critic_for_reviews", "duration", "director_facebook_likes",
opts.VariableTypes = ["categorical", "string", "string", "double", "double", "double", "double", "string", "double", "doub

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["director_name", "actors", "actor_2_name", "actor_1_name", "movie_title", "actor_3_name", "plot_k
opts = setvaropts(opts, ["color", "director_name", "actors", "actor_2_name", "genres", "actor_1_name", "movie_title", "act
```

FIGURE 9.5
Live script.

the modifications that were carried out after loading the data into MATLAB. The live script shown below will help us to reuse the code to import the data rather than manually importing the data with the import tool (Figure 9.5).

A data scientist might come across a situation where it is needed to import multiple files with the same structure and format at several stages in the analysis. In such cases, generating an import function rather than generating a live script will be beneficial. This import function will import the original file and the other files in the same format.

The steps to use an import function are as follows:

Step 1: Click on the "Generate Function" option under the "Import Selection" tab.

Step 2: Save the generated function to the same location where all the code and data files are stored.

Step 3: Write the below code in the script. This code specifies the file name and assigns the output after calling the function.

```
Input: filename = "MovieData_with_actors.csv";

Movie = importfile(filename)
```

Step 4: Once you have completed Step 3, you can use the same code with a new filename to import another data file.

9.3 Visualizing and Filtering Data

One efficient way that the data scientists use to explore their data is visualization. One can detect outliers using visualization and then filter the data as required. Filtering the data makes it easy to perform further analysis. MATLAB

provides us with many collaborative tools and functions to visualize, choose, and update the data. In MATLAB, you can insert visualizations on the website by making them public and generating a URL. In this section, you will learn to plot data for the selected data from the table and access the table variables.

9.3.1 Plotting Data Contained in Tables

Once you import your data into MATLAB, you need to explore the data to get familiar with the data and decide the actions to be performed on it for further analysis. To get insights into the data, you can just look at the data, but this is not feasible when the data is large with thousands of observations with more variables. In such cases, visualizations have proven to be giving informative insights into the data.

Let's take an example. Here, we are using "MovieData" which contains various movies and their related information such as the rating, the title year, the director, actors, and much more. Say, we want to know about the release year of the movies over some time. This can be easily accomplished using the histogram plot of the MATLAB.

The steps required to plot a histogram are as follows (Figure 9.6):

Step 1: Import your data into MATLAB.

Step 2: Select the variable column to be plotted – title-year.

Step 3: Select the type of plot to be plotted – histogram – by clicking on the plots tab.

Step 4: Explore the histogram using the buttons given at the top right of the plot.

MATLAB provides us with an incredible feature of the autogeneration of command while surfing through the workspace. In the above example, when you select the histogram plot from the plots tab, the command to build a histogram is automatically generated in the command window.

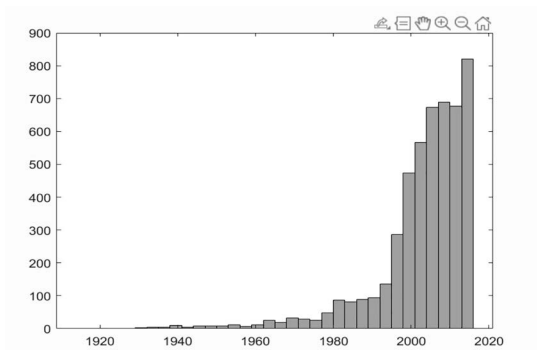


FIGURE 9.6
Histogram plot.

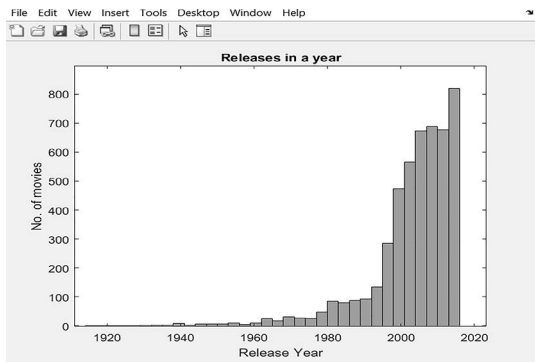


FIGURE 9.7
Histogram plot with labels.

When you plot any graph, you can also give labels to the plot as per your convenience; you can insert the Y-label, X-label, and Title to your plot in the new window under the insert tab (Figure 9.7).

In most of the data sources, it is essential to filter the data as without filtering, the plot becomes congested with lots of observations that might be unnecessary for the analysis. We will learn about filtering the data in the upcoming topics.

9.3.2 Selecting Data from Tables

Data scientists require a lot of data to generate accurate results, but when it comes to working on a single observation or more than one observation among the entire data, all the other observations are completely unessential and may cause a hindrance to produce your specific results. In such cases, filtering your data is very important. The very first filter that you can give to your data is while importing the data into MATLAB. You can select subsets of data and import only that subsets into MATLAB, automatically ignoring the remaining dataset. Once you import the data, you can filter the observation under variables just by clicking on the variable name. By clicking on the variable name, various alternatives to sort and filter that variable will be shown. A checkbox containing all the unique observations under that particular variable will be displayed. Rather than clearing the unwanted observations, you can opt to clear all the observations first and then select the necessary observations for analysis.

Let's take an example of the previously used "MovieData" data. Say, I want to perform my analysis on the observations of the movies that are released in the year 2019. So, in such circumstances, by clicking on the "title_year" variable, I will clear all and select "2019" as the only needed observation.

When you apply such changes to your data, the changes will be made on the original table by default rather than creating a separate filter table. While exploring small data, it is advised to create a separate filter table to preserve the

original table for further analysis. You can rename the filter table and rerun the code. Once you run the code again, two tables will be visible in the workspace, the original table (MovieData) and the filter table (Release_2019). Creating a different filter table occupies an additional memory; therefore, when you are exploring larger datasets, do not create a different filter table unnecessarily.

9.3.3 Accessing and Creating Table Variables

In this lesson, you will learn to use a single variable and also to add various variables to create a new variable. In MATLAB, you can also export and save your outcomes outside the workspace so that it can be used anywhere without MATLAB.

In our example of the "MovieData" data, for a particular movie, the name of the actors is reported differently. For convenience and ease in further analysis, you can add all the actor names to one variable named "actors". We have learned in the above lessons that to access the variable, the table name along with the dot operator is used.

The command to add these variables to a new variable, named "actors", is given below:

Input: `Movie.actors= Movie.actor1 _ name + ', ' + Movie.actor2 _ name + ', ' + movie.actor3 _ name;`

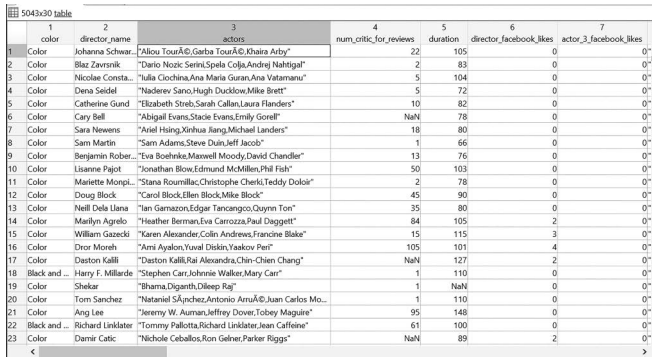
Output: You can see in the below image that after running the above command, a new variable named 'actors' is added as the last variable to the table (Figure 9.8).

You can shift any variable from one place to another just by clicking at the top-left corner of the variable name cell and dragging to the desired place. Say, I wanted to shift the newly created variable "actors" beside the "director_name". When you drag the column beside the director name, the command for such service will be generated in the command window automatically (Figure 9.9).

5043x30 table

	21 country	22 content_rating	23 budget	24 title_year	25 actor_2_fa	26 imdb_score	27 aspect_ratio	28 movie_facebook_likes	29 total_likes	30 actors
1	UK	NaN	400000	2015	0	7.7000	NaN	34	0	"Aliou TourÃ©,Garba TourÃ©,Khaira Arby"
2	Slovenia	NaN	500000	2015	0	6.4000	2.3500	18	0	"Dario Nozic Serini,Spela Colja,Andrej Nahtigal"
3	Romania	NaN	168000	2015	0	6.5000	NaN	34	0	"Iulia Ciochina,Ana Maria Guran,Ana Vatamanu"
4	USA	NaN	150000	2015	0	7	NaN	215	0	"Naderev Sano,Hugh Ducklow,Mike Brett"
5	USA	NaN	500000	2014	0	6.8000	NaN	44	0	"Elizabeth Streb,Sarah Callan,Laura Flanders"
6	USA	NaN	180000	2014	0	8.7000	NaN	88	0	"Abigail Evans,Stacie Evans,Emily Gorell"
7	USA	NaN	150000	2014	0	7.1000	NaN	116	0	"Ariel Hsing,Xinhua Jiang,Michael Landers"
8	USA	NaN	250000	2013	0	7.8000	NaN	53	0	"Sam Adams,Steve Duin,Jeff Jacob"
9	USA	NaN	1400	2013	0	6.3000	NaN	16	0	"Eva Boehnke,Maxwell Moody,David Chandler"
10	Canada	NaN	100000	2012	0	7.7000	16	0	0	"Jonathan Blow,Edmund McMillen,Phil Fish"
11	France	NaN	900000	2011	0	6.7000	16	24	0	"Stana Roumillac,Christophe Cherki,Teddy Doloir"
12	Germany	NaN	NaN	2005	0	7.5000	NaN	70	0	"Carol Block,Ellen Block,Mike Block"
13	Philippines	NaN	7000	2005	0	6.3000	NaN	74	0	"Ian Gamazon,Edgar Tancangco,Quynn Ton"
14	USA	NaN	NaN	2005	0	7.5000	1.8500	588	2	"Heather Berman,Eva Carrozza,Paul Daggett"
15	USA	NaN	750000	2002	0	7	NaN	32	3	"Karen Alexander,Colin Andrews,Francine Blake"
16	Israel	-13	NaN	2012	0	7.6000	1.8500	0	4	"Ami Ayalon,Yuval Diskin,Yaakov Peri"
17	USA	NaN	500000	2005	0	5.8000	1.3300	0	4	"Daston Kalili,Rai Alexandra,Chin-Chien Chang"
18	USA	NaN	100000	1920	2	4.8000	1.3300	0	4	"Stephen Carr,Johnnie Walker,Mary Carr"
19	India	NaN	NaN	2013	0	7.1000	NaN	2	5	"Bhama,Diganth,Dileep Raj"
20	USA	NaN	1200000	2013	2	7.2000	NaN	75	5	"Nataniel SÃ¡nchez,Antonio ArruÃ©,Juan Carlos Mo...
21	USA	NaN	35000000	1999	2	6.8000	2.3500	633	5	"Jeremy W. Auman,Jeffrey Dover,Tobey Maguire"
22	USA	NaN	23000	1991	0	7.1000	1.3700	2000	5	"Tommy Pallotta,Richard Linklater,Jean Caffeine"

FIGURE 9.8
Combining columns.



5043x30 table

	1 color	2 director_name	3 actors	4 num_critic_for_reviews	5 duration	6 director_facebook_likes	7 actor_3_facebook_likes
1	Color	Johanna Schwar...	"Aliou TourÃ©,Garba TourÃ©,Khaira Arby"	22	105	0	0"
2	Color	Blaz Zavrsnik	"Dario Nozic Serini,Spela Colja,Andrej Nahtigal"	2	83	0	0"
3	Color	Nicolae Consta...	"Iulia Ciochina,Ana Maria Guran,Ana Vatamanu"	5	104	0	0"
4	Color	Dena Seidel	"Naderev Sano,Hugh Ducklow,Mike Brett"	5	72	0	0"
5	Color	Catherine Gund	"Elizabeth Streb,Sarah Callan,Laura Flanders"	10	82	0	0"
6	Color	Cary Bell	"Abigail Evans,Stacie Evans,Emily Gorell"	NaN	78	0	0"
7	Color	Sara Newens	"Ariel Hsing,Xinhua Jiang,Michael Landers"	18	80	0	0"
8	Color	Sam Martin	"Sam Adams,Steve Duin,Jeff Jacob"	1	66	0	0"
9	Color	Benjamin Rober...	"Eva Boehnke,Maxwell Moody,David Chandler"	13	76	0	0"
10	Color	Lisanne Pajot	"Jonathan Blow,Edmund McMillen,Phil Fish"	50	103	0	0"
11	Color	Mariette Monpi...	"Stana Roumillac,Christophe Cherki,Teddy Doloir"	2	78	0	0"
12	Color	Doug Block	"Carol Block,Ellen Block,Mike Block"	45	90	0	0"
13	Color	Neill Dela Llana	"Ian Gamazon,Edgar Tancangco,Quynn Ton"	35	80	0	0"
14	Color	Marilyn Agrelo	"Heather Berman,Eva Carrozza,Paul Daggett"	84	105	2	0"
15	Color	William Gazecki	"Karen Alexander,Colin Andrews,Francine Blake"	15	115	3	0"
16	Color	Dror Moreh	"Arni Ayalon,Yuval Diskin,Yaakov Peri"	105	101	4	0"
17	Color	Daston Kalili	"Daston Kalili,Rai Alexandra,Chin-Chien Chang"	NaN	127	2	0"
18	Black and ...	Harry F. Millarde	"Stephen Carr,Johnnie Walker,Mary Carr"	1	110	0	0"
19	Color	Shekar	"Bhama,Diganth,Dileep Raj"	1	NaN	0	0"
20	Color	Tom Sanchez	"Nataniel SÃ¡nchez,Antonio AmuÃ©,Juan Carlos Mo...	1	110	0	0"
21	Color	Ang Lee	"Jeremy W. Auman,Jeffrey Dover,Tobey Maguire"	95	148	0	0"
22	Black and ...	Richard Linklater	"Tommy Pallotta,Richard Linklater,Jean Caffeine"	61	100	0	0"
23	Color	Damir Catic	"Nichole Ceballos,Ron Gelner,Parker Riggs"	NaN	89	2	0"

FIGURE 9.9
Rearranging the columns.

Input: `Movie = movevars(Movie, 'actors', 'Before', 'num _ critic _ for _ reviews');`

You can save this updated table to a file so that it can be used outside MATLAB. This is performed using the writetable() function. The syntax to save a file using the writetable() function is as follows:

Syntax: writetable(original_name, "new_file_name.extension").

9.4 Performing Calculations

Data science uses statistics to a greater extent. In several phases of a data science project, you will need to perform calculations to proceed further in the analysis. In this lesson, you will learn to perform some basic mathematical calculations; find the relation between variables; computations on categorical data; and much more. Let's get started.

9.4.1 Basic Mathematical Operations

In your journey toward becoming a data scientist, you will need to do some serious numerical computations in which some of them cannot be performed manually. Various mathematical expressions use different arithmetic operators in a single expression. The various arithmetic operators that are commonly used in MATLAB are given below[8]:

- Addition +
- Subtraction –
- Multiplication *

- Division **/**
- Exponentiation **^**
- Parentheses **()**

When an arithmetic expression is carried out in MATLAB, the result is, by default, stored in a variable named "ans"; if you want to explicitly assign a variable name, you can do so using the assignment operator.[8]

For example, the result of the expression "3 * 4", which is 7, is stored in the ans variable – ans=7. Say, we want to store the result in a variable named "result", we will do so using the "=" operator (result=3*4).

A programmer or a scientist should always assign an explanatory name to the variable. When assigning a name to a variable in MATLAB, the following rules are mandatory:

- Variable names should begin with a letter.
- Variable names can be a combination of just numbers, letters, and underscores.
- Variable names are case sensitive.

The order of operations in MATLAB are as follows:

1. Parentheses
2. Exponentiation
3. Division and Multiplication
4. Addition and Subtraction.

Let's calculate the distance between two coordinates in MATLAB. We know that the distance between two points can be calculated using the following formula:

$$D = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

Firstly, we have to assign values to the variables.

When the values are assigned, build the equation. While building the equation, grouping and the order of the equations play a very vital role.

$$D = \sqrt{\left(y_2 - y_1^2 + x_2 - x_1^2\right)}$$

Although the above equation seems to be correct, it is incorrect. This equation is a classic example to understand the importance of grouping the terms in the equation. Owing to the order of the operators, MATLAB will calculate y_1^2 rather than first subtracting y_2 and y_1 and then squaring them.

The correct equation for the distance formula will be the following:

$$D = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

Therefore, always remind yourself to group the terms in accordance with the order of operations when performing critical computations in MATLAB.

9.4.2 Using Vectors

In data science, you will encounter many situations where you will need to perform computations on a range of values. A single list of many data elements is called the vector. Let's take an example of the equation of a line:

$$y = mx + c$$

To plot the line for the y values against the x values, you need to calculate enough values for both the x and the y values to get an efficient line of the equation. In MATLAB, vectors are created using a set of values enclosed using the square brackets and parted using the commas.

For example, a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

A much efficient way to enter above such vectors that are equally spaced vectors is using the desired range. In MATLAB, ":" specifies the range. The above-mentioned vector "a" can be written as "a = 1:10"; this command will yield the same result as before with a default spacing of 1. If the user intends to change the spacing, the syntax for specifying the spacing is given as the following:

Syntax: `vector _ name = lower _ limit : spacing : upper _ limit`

For example: Input: `a = 1: 2: 10`

```
Output: a = [1,4,7,10]
```

Element-wise operations

In MATLAB, element-wise operations play an important role. There are three element-wise operations in MATLAB:

- **.*** – Multiplication element-wise operator.
- **./** – Division element-wise operator.
- **.^** – Exponentiation element-wise operator.

In MATLAB, addition and subtraction do not have any element-wise operations as they can be performed using the common way in numbers, scalars, vectors, and matrices.

Consider two vectors, a = [2, 3, 4, 5, 6] and b = [4, 5, 3, 6, 8].

Now, a + b = [6, 8, 7, 11, 14]; a–b = [–2, –2,1, –1, –2]; and a.* b = [8, 15, 12, 30, 48]

Whenever performing computations on a vector, remember the following points:

- Element-wise operations should be used for multiplication, division, and exponentiation.
- The vectors used for calculations must be of the same size.
- The vectors must either all be row vectors or column vectors.

If any of the above rules are not satisfied, MATLAB will pop up a matrix dimension error with the required command.

To solve the equation of a line, we will have to rewrite the equation as the following:

$$y = m.*x + c$$

This equation uses the element-wise operator for multiplication as x is the vector.

9.4.3 Using Functions

MATLAB provides many built-in functions, and many of them are used in various common ways. Some of the built-in functions are the plot() function to plot a graph, sqrt() function to find the sqrt of a number, scatter() function to plot a scatter plot, and bounds() function to find the lower and upper bounds of a series. The built-in functions used for mathematical calculations can take the input as the vector or a matrix.

In MATLAB, a function is created using the function keyword. The syntax for creating a function in MATLAB is given as follows:

Syntax: function [output1, output2, …outputN]=function_name(input1, input2,…inputN)

Here, input1, input2, …, inputN specifies the input arguments that give the output1, output2,…outputN.

One important thing to remember while creating functions in MATLAB is that in MATLAB, functions are stored in files and the file name should end with an extension '.m' and the file name should be the same as the function name.

As a programmer or as a scientist, it is a good practice to use command lines as they are descriptive and can give a brief description of the function. To include a command line in MATLAB, the "%" symbol is used. Ideally, the command line description for a function is given right after defining a function.

For example, let us create a function that gives us the product of five numbers.

Input:

```
function product = product_num(2, 3, 4, 5)
    % This function gives the product of the four numbers
given as the inputs
max = 2 * 3 * 4 * 5;
end
```

Output:

```
ans = 120
```

One can also get the description of the function using the help keyword. The help keyword displays the command lines provided just after the definition of the function.

Syntax: help function_name

Input: help product_num

Output: This function gives the product of the four numbers given as the inputs

Anonymous functions

The previously mentioned syntax for defining a function creates a file to store that function. A user can also create a simple function without creating a file for it. Such functions are called as the anonymous functions. Syntax for these functions is as follows:

Syntax: function_name=@ (argument_list) expression

For example, **Input**:

```
addition = @ (a, b) a +b;
result = addition (9,10)
```

Output:

```
result = 19
```

9.4.4 Calculating Summary Statistics

Once we import the required data into MATLAB or any software for that matter, it is important to inspect the data to get more information about all the columns in the dataset, its probable values, and much more. Exploring data in a small dataset is convenient to observe the data instantly. But, in the real world, the data is larger and larger and can contain observations in millions. Observing such data directly is impossible. We can get descriptive statistics of such data using the summary() function in MATLAB. This function accepts the tabular data and returns the variable name, size, and the data type of the variable. The summary() function gives an overview of all the variables included in the data. The type of information about the variable varies along with the data type of the variable. For numeric (double) data

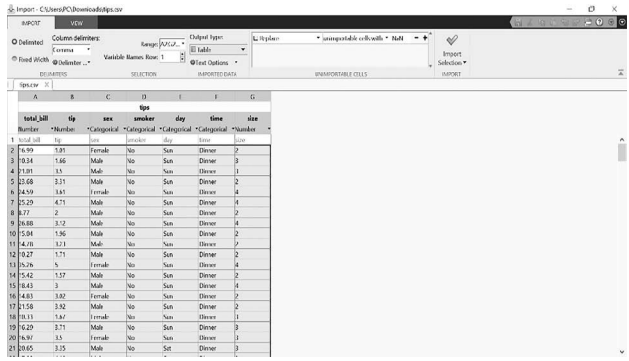


FIGURE 9.10
Tips.csv dataset.

type, it returns the minimum value, the median, and the maximum value. For categorical or logical variables, the summary() functions return the frequency count of all the unique values of the variables. The variable with the string data type has no additional information returned by the summary() function. The user might need different statistical measures depending on the variable. You can use the mode() function to find the most common value in the variable.[9] The average value can be found out using the mean() function. There are various such functions in MATLAB that can help to get descriptive statistics of the entire data or a required variable in particular.

When you import the "tips.csv" dataset into MATLAB, it will look like this (Figure 9.10)-

When we run the summary() function by passing the "tips.csv" file to the function, it will give a descriptive statistics summary as below.

Input:

```
summary(tips)
```

Output:

```
Variables:
total_bill: 244×1 doubletip: 244×1 double
        Values:   Values:
            Min           3.07 Min           1
            Median     17.795
Median   2.9
            Max          50.81
Max 10

sex: 244×1 categoricalsmoker: 244×1 categorical
        Values:Values:
            Female        87          No        151
            Male         157   Yes         93
```

```
day: 244×1 categorical  time: 244×1 categorical
        Values:Values:
            Fri         19   Dinner      176
            Sat         87   Lunch        68
            Sun         76
Thur        62
```

9.4.5 Correlations between Variables

In the data you are working with, there exist many such variables in the data that have a relationship between two different variables. Such kind of relationship can be observed while visualizing the data, but if it is needed to narrate and compare the number of relationships between variables, it is essential to know a quantitative measure that can tell us more about the correlation among the variables. MATLAB provides us with the corr() function that is used to find the relationship between two variables. The corr() function returns a number that is referred to as the correlation coefficient and is between -1 and 1. There are two types of correlation:

- **Positive Correlation** – As the name suggests, a positive correlation exists when the correlation coefficient is greater than 0. This type of correlation indicates that as the value of one variable increases, the corresponding value of another variable also increases.
- **Negative Correlation** – As the name suggests, a negative correlation exists when the correlation coefficient is less than 0. This type of correlation indicates that as the value of one variable increases, the corresponding value of another variable decreases.

A strong linear relationship between the variables exists when the correlation coefficient is closer to 1 or -1. The linear relationship between variables is said to be weak when the correlation coefficient is close to zero. Also, sometimes, when the correlation coefficient is close to zero, there is no relation among the variable at all. Note that a strong nonlinear relation can exist between two variables, even if its correlation coefficient is close to zero.

Input:`>>corr(tips.total _ bill, tips.tip)`
Output:`ans =0.6698`

9.4.6 Accessing Subsets of Data

In this section, we will learn about accessing and extracting elements from vectors, matrices, and tables in MATLAB. We will start with the vector. The vector elements are indexed consecutively. It is important to remember that

the indexing of a vector begins with 1 in MATLAB unlike in most of the programming languages, where the indexing begins with 0.

For example, let's consider a vector "Age" that will have five values in it.

Input: `age = [ 22, 23, 24, 25]`

Now, you can say that the value at index 2 is 23.

Input: `age _ 3 = age(2)`

This command will return 23 and store it in the "age_3" variable.

If you want to access the data elements within a range, you can use the colon ":".

Input: `age _ range = age(1:2)`

Output: `age _ range = 22 23`

If you want to update a value, you can make use of the below syntax.

Syntax: vector_name (index)=value;

Input:

```
>> age(2)=50;
>>age
```

Output:

```
age =22 50 24 25
```

When it comes to matrices or tables, we know that both of them are made out of rows and columns. To access or modify the values in a matrix, you need to specify the rows and columns indices. Consider a matrix below:

$$\text{Matrix\_A} = \begin{bmatrix} 2 & 4 \\ 5 & 1 \\ 3 & 9 \end{bmatrix},$$

If you want to access a data element which belongs to the second row and in the second column, the following command will do so:

Input: `matrix _ 22 = Matrix _ A (2, 2)`

Output: `matrix _ 22 =`

```
1
```

Let's take another example of a matrix where you need to extract a smaller subset matrix from the bigger matrix.

$$\text{Matrix\_B} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

If you want to extract the last two rows and all the columns from the matrix, run the following command:

Input: `matrix _ C = Matrix _ B([2,3], 1:3)`

Output: `matrix _ C =`

$$\begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

When one needs to access an entire row or entire column, the ":" operator can be used.

Input:

```
matrix _ row = Matrix _ B (2, :)
Matrix _ column = Matrix _ B (:, 2)
```

Output:

`matrix_row =`

$$\begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

`Matrix_column =`

$$\begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$$

9.4.7 Performing Calculations by Category

In this lesson, we will learn to modify the categorical data and group them for further exploration. Consider the previously used "MovieData" data. Suppose, we want to find out the total Facebook likes for the actors in a particular movie to analyze the popularity impact on the movie. To do so, we will have to combine three variables – actor_1_fb, actor_2_fb, and actor_3_fb. These variables might have NaN (Not a Number) values as well. When you add these three variables directly, if anyone or more than one of the variable observations has a NaN value, it will generate the result as NaN. This will cause the loss of Facebook likes for the actors and will pay a hindrance to our further analysis. Handling missing data is different for different datasets. In the "MovieData" dataset, it is reliable to consider the NaN value as "0", considering that there were no likes for that particular actor in the movie.

Now, the steps to add all these variables into one variable, say "Total_facebook_likes", are given below:

Step 1: Fill the NaN values of all three variables to 0.

Input:

```
>>movie.actor_1_fb(ismissing(movie.actor_1_fb))=0;
>>movie.actor_2_fb(ismissing(movie.actor_2_fb))=0;
>>movie.actor_3_fb(ismissing(movie.actor_2_fb))=0;
```

Step 2: Add the three variables to a new variable

```
>>movie.Total_facebook_likes= movie.actor_1_fb + movie.
actor_2_fb+ movie.actor_3.fb;
```

In another such scenario, say, you want to know the count of the movies in a particular group or category. This can be carried out using the groupsummary() function (Figure 9.11).

The default statistics that the groupsummary() function carries is the GroupCount.

The syntax to get the GroupCount is given below (Figure 9.12):

Syntax: variable_name=groupsummary(table_name, existing_variable_name)

Input:
```
country _ wise = groupsummary(Movie _ data, "country");
```

Output:

You can also calculate other statistics such as the average, maximum, minimum, etc. This can be achieved by specifying that particular method (Figure 9.13).

5043x30 table

	27 imdb_score	28 aspect_ratio	29 movie_facebook_likes	30 total_likes_actors	31
1	7.2000	1.8500	0	655000	
2	5.2000	2.3500	0	301748	
3	3.9000	NaN	0	283000	
4	5.6000	1.8500	706	261875	
5	9.1000	NaN	0	260808	
6	6.4000	2.3500	0	166898	
7	6	1.8500	0	139000	
8	5.7000	2.3500	0	137622	
9	6	2.3500	0	120000	
10	5.4000	2.3500	2000	105000	
11	6.6000	2.3500	0	100759	
12	5.8000	2.3500	773	98935	
13	5.6000	2.3500	0	89702	
14	7	1.8500	20000	88602	
15	6.7000	2.3500	371	88492	
16	4.9000	1.8500	431	88040	
17	8.8000	2.3500	175000	79000	
18	6.4000	NaN	0	77037	
19	8	2.3500	82000	76000	
20	6.4000	1.8500	30000	76000	

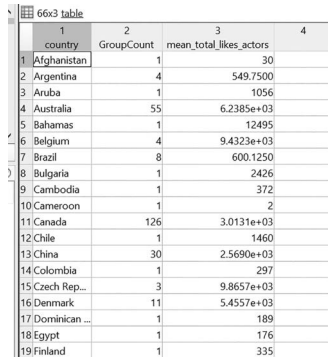
FIGURE 9.11
Adding columns and storing the results.

Movie_data | country_wise

66x2 table

	1 country	2 GroupCount	3
1	Afghanistan	1	
2	Argentina	4	
3	Aruba	1	
4	Australia	55	
5	Bahamas	1	
6	Belgium	4	
7	Brazil	8	
8	Bulgaria	1	
9	Cambodia	1	
10	Cameroon	1	
11	Canada	126	
12	Chile	1	
13	China	30	
14	Colombia	1	
15	Czech Rep...	3	
16	Denmark	11	
17	Dominican ...	1	
18	Egypt	1	
19	Finland	1	

FIGURE 9.12
Implementation of the group summary function.



66x3 table

	1 country	2 GroupCount	3 mean_total_likes_actors	4
1	Afghanistan	1	30	
2	Argentina	4	549.7500	
3	Aruba	1	1056	
4	Australia	55	6.2385e+03	
5	Bahamas	1	12495	
6	Belgium	4	9.4323e+03	
7	Brazil	8	600.1250	
8	Bulgaria	1	2426	
9	Cambodia	1	372	
10	Cameroon	1	2	
11	Canada	126	3.0131e+03	
12	Chile	1	1460	
13	China	30	2.5690e+03	
14	Colombia	1	297	
15	Czech Rep...	3	9.8657e+03	
16	Denmark	11	5.4557e+03	
17	Dominican ...	1	189	
18	Egypt	1	176	
19	Finland	1	335	

FIGURE 9.13
Calculating the mean.

Input:
```
country_wise_max = groupsummary(Movie_data, "country", "mean", "total_likes_actors");
```
Output:

9.5 Summary

In this chapter, we got an overview of MATLAB in data science. We first learned about how a data science project is divided into stages and how MATLAB is used in different stages. Whenever we have to work on any data, we first need to import that data into the workspace to perform analysis and gain insights into them. We learned how easy it is to import the data and know the automatically generated data types in MATLAB using the import tool. While exploring data, visualizations play a very important role. We went through the plots used in MATLAB to visualize data. Also, we studied to use different variables from the table and create new variables. Performing computations is at the heart of data science. To end with the chapter, we learned about various techniques that MATLAB provides to perform calculations on vectors, functions, and categorical data.

Exercise

1. Why is it essential to automate the import process?
2. How will you find the top three recorded observations of a certain variable?
3. At what location is the newly created variable added to the table? Write the command to shift the location of a variable within the table.
4. Explain the writetable() function.

5. State the importance of the summary statistics.
6. Where are the functions in stored in MATLAB?
7. Explain the element-wise operators in MATLAB?
8. Calculate the summary statistics of the movie data set.
9. Explain MATLAB data types.
10. What is data visualization and data exploration?

References

1. Martinez Wendy L., Martinez Angel R., Solka Jeffrey L. 2011. *Exploratory Data Analysis with MATLAB*. Boca Raton: CRC Press.
2. 2014. Matlab Tutorial. Tutorials point. tutorialspoint.com/matlab/index.htm (accessed 9 Sepember 2020).
3. Sharma Kaushal, Singh Kshitij. 2018. *Machine Learning with MATLAB*. India: MATLABSoluions.
4. Nylen Erik Lee, Wallisch Pascal. 2017. *Neural Data Science: A Primer with MATLAB® and Python™*. United States: Academic Press.
5. Mikhailov Eugeniy E. 2017. *Programming with MATLAB for Scientists*. Boca Raton: CRC Press.
6. 2020. MATLAB data analysis. MathWorks. https://www.mathworks.com/help/pdf_doc/matlab/data_analysis.pdf (accessed 12 September 2020).
7. Cho MoonJung, Martinez Wendy L. 2015. *Statistics in MATLAB: A Primer*. Boca Raton: CRC Press.
8. Pratap Rudra. 2010. *Getting Started with MATLAB*. New York: Oxford University Press.
9. Webb Cerian Ruth, Domijan Mirela. 2019. *Introduction to MATLAB® for Biologists*. New York: Springer Science and Business Media LLC.

10

GNU Octave as a Data Science Tool

GNU Octave is a free-source application with a high-level programming language. GNU Octave is mainly used for numerical calculations. It is primarily designed for matrix calculations such as decoding simultaneous equations, calculating eigenvalues and eigenvectors, etc. As GNU Octave possesses its programming language, it can present the data in many different ways. In the data science domain, GNU Octave is used to create and test numerical algorithms.

10.1 Vectors and Matrices

Vectors

Numeric calculations normally consist of operating with number sequences. Such sequences of numbers are called arrays, but in GNU Octave, they are treated as a kind of vectors. A vector can be stated as a 1×1 matrix that encloses a number list.

GNU Octave provides us with various ways to define a vector. Mostly, they are defined using the square brackets "[ ]".

As you can see in Figure 10.1, a row vector is defined by separating the numbers enclosed in the square brackets by spaces or commas. Similarly, a

```
octave:2>  v = [ 2 4 8]
v =

   2   4   8

octave:3> a = [ 3,6,9]
a =

   3   6   9

octave:4> b = [5;10;15]
b =

    5
   10
   15
```

FIGURE 10.1
Defining vectors.

DOI: 10.1201/9780429443237-13

```
octave:2> m = [2:9]
m =

   2   3   4   5   6   7   8   9

octave:3> n = [2:3:10]
n =

   2   5   8

octave:4> o = [10:-2:1]
o =

   10    8    6    4    2
```

FIGURE 10.2
Defining vectors within a range.

column vector is defined by separating the numbers enclosed in the square brackets by semicolons.

You can also create a vector of counting numbers using a colon ":". To create such a vector, define a range separated by a colon and enclosed in square brackets as shown in Figure 10.2.

You can also specify the increment factor within the range as shown in the above figure.

Note that if a negative increment factor has to be given, the range should be given in a descending manner.

GNU Octave also has some in-built functions to create vectors. These functions can be used to create a null matrix, an evenly spaced vector, and much more. They are as follows:

1. **linspace (a1, a2, N)** – This function creates a vector from a1 to a2 by evenly separating N elements in the vector. As shown in Figure 10.3, a vector is created starting from 1 to 5, and all the elements are evenly placed to create a vector with six elements.

```
octave:2> linspace(1,5,6)
ans =

   1.0000   1.8000   2.6000   3.4000   4.2000   5.0000
```

FIGURE 10.3
linspace () example.

2. **zeros (a1, a2, N)** – This function creates a vector from 10^{a1} to 10^{a2} by logarithmically separating N elements in the vector. As shown in Figure 10.4, a vector is created starting from 10^1 to 10^5, and all the elements are logarithmically placed to create a vector with six elements.

You can display the vector by just entering the vector name and extract the elements of the vector by enclosing the element_index in the round brackets (). To extract a range of elements, use the colon notation as shown in Figure 10.5.

Matrices

Matrices are the rectangular array elements and are represented to be in size m×n, where m and n represent the number of rows and columns, respectively.

$$A = \begin{bmatrix} 2 & 4 & 3 \\ 8 & 6 & 5 \\ 12 & 24 & 6 \end{bmatrix}$$

There are several ways to create a matrix in GNU Octave.

You can create a matrix using the same syntax as that of a vector just defining the row in different rows as shown in Figure 10.6.

```
octave:3> logspace(1,5,6)
ans =

   1.0000e+01   6.3096e+01   3.9811e+02   2.5119e+03   1.5849e+04   1.0000e+05
```

FIGURE 10.4
logspace () example.

```
octave:2> a = [1:6]
a =

   1   2   3   4   5   6

octave:3> a
a =

   1   2   3   4   5   6

octave:4> a(4)
ans = 4
octave:5> a(2:6)
ans =

   2   3   4   5   6
```

FIGURE 10.5
Extracting elements from a vector.

```
octave:5> a=[1 2 3
   4 5 6]
a =

   1   2   3
   4   5   6

octave:6> b = [-2,3,0;8,9,-6;6,2,8]
b =

  -2   3   0
   8   9  -6
   6   2   8

octave:7> c = [2:4;5:2:10;-8,2,-1]
c =

   2   3   4
   5   7   9
  -8   2  -1
```

FIGURE 10.6
Matrix creation.

```
octave:10> a
a =

   1   2   3
   4   5   6

octave:11> b
b =

  -2   3   0
   8   9  -6
   6   2   8

octave:12> a*b
ans =

   32   27   12
   68   69   18
```

FIGURE 10.7
Matrix multiplication.

Another way to create a matrix is by using the semicolon notation.[1] Enclose the entire matrix in the square brackets and separate the rows using semicolons.

GNU Octave provides us with a way to create a matrix using the range of values, i.e., using the colon notation. As shown in Figure 10.6, various ways of the colon notation can be used at different rows.[2]

Matrix Multiplication

For matrices and vector multiplication, the * symbol is used to represent the multiplication.

In matrix multiplication, it is important to remember that the matrices can be multiplied only if they are compatible with each other. Compatible matrices are the matrices in which the number of columns of the first matrix matches with the number of rows of the second matrix. As shown in Figure 10.7, matrix a and matrix b are compatible with each other and thus can be multiplied.

```
octave:13> a
a =

   1   2   3
   4   5   6

octave:14> c
c =

   7   8   9
   2   4   6

octave:15> a*c
error: operator *: nonconformant arguments (op1 is 2x3, op2 is 2x3)
```

FIGURE 10.8
Matrix multiplication error.

```
octave:16> a
a =

   1   2   3
   4   5   6

octave:17> a'
ans =

   1   4
   2   5
   3   6
```

FIGURE 10.9
Transpose of a matrix.

However, the matrices defined in Figure 10.8 cannot be multiplied as the number of columns of the matrix (i.e., 3) is not equal to the number of rows in matrix c (i.e., 2). The Octave prompt will throw an error when such matrices are requested to be multiplied.[3]

Transpose of a Matrix

A transposed matrix is a matrix whose original rows and columns are interchanged with one another.[4] Usually, the transposed matrix is represented as S^T. In GNU Octave, apostrophe (') is used to transpose a matrix (Figure 10.9).

GNU Octave provides various in-built functions that can be used for matrix manipulation. We will discuss some of them in this chapter.

1. **zeros (M, N)** – This function creates a matrix of size M×N which consists of every element as a zero.
2. **ones (M, N)** – This function creates a matrix of size M×N which consists of every element as one.
3. **eye(N)** – This function creates an identity matrix of size N×N.[5] An identity matrix is a matrix whose diagonal elements are 1, and all other elements are 0 (Figure 10.10).[5]

```
octave:3> m= zeros(2,3)
m =

   0   0   0
   0   0   0

octave:4> n = ones(3,3)
n =

   1   1   1
   1   1   1
   1   1   1

octave:5> I = eye(5)
I =

Diagonal Matrix

   1   0   0   0   0
   0   1   0   0   0
   0   0   1   0   0
   0   0   0   1   0
   0   0   0   0   1
```

FIGURE 10.10
Matrix creation functions.

Some other functions that can be used for matrix manipulation are as follows:

i. **rand()** – This function creates a matrix that consists of random numbers.
ii. **diag()** – This function creates a diagonal matrix. It is also used to extract diagonal elements of a matrix.
iii. **inv()** – This function creates an inverse of that particular matrix.
iv. **eig()** – This function calculates the eigenvectors and eigenvalues of a matrix.
v. **det()** – This function is used to calculate the determinant of a matrix.

10.2 Arithmetic Operations

GNU Octave provides various functions that are used to perform various arithmetic operations on vectors or matrices. Many functions ease the user's job to perform arithmetic operations. However, we will discuss some of them in this chapter.[6]

1. **exp(x)** – This built-in function creates a vector/matrix which consists of the exponential of every element in the vector/matrix x, i.e., it calculates e^x for all the elements in x (Figure 10.11).

```
octave:4> exp(a)
ans =

   2.7183e+00   7.3891e+00   2.0086e+01
   5.4598e+01   1.4841e+02   4.0343e+02
   1.0966e+03   2.9810e+03   8.1031e+03
```

FIGURE 10.11
exp(x) example.

2. **pow2(x) –** This function creates a vector/matrix that consists 2^x of the elements in the vector/matrix x. For example, in Figure 10.12, the result is the interpretation of [2^1, 2^2, 2^3, 2^4, 2^5, 2^6].
3. **pow2(f, x)** – This function is similar to the pow2() function. Here, f is multiplied to all the elements in the pow2(x) function. Mathematically, it can be represented as f*(2^x) (Figure 10.13).
4. **sqrt(x)** – As the name suggests, this function calculates the square root of all the elements in the vector/matrix x (Figure 10.14).[7]

```
octave:2> a=[1:6]
a =

   1   2   3   4   5   6

octave:3> pow2(a)
ans =

    2    4    8   16   32   64
```

FIGURE 10.12
pow2(x) example.

```
octave:0> pow2(3,a)

ans =

     6    12    24    48    96   192
```

FIGURE 10.13
pow2(f,x) example.

```
octave:0> sqrt(a)
ans =

   1.0000   1.4142   1.7321   2.0000   2.2361   2.4495
```

FIGURE 10.14
sqrt(x) example.

```
octave:7> a = [-5:-1]
a =

  -5  -4  -3  -2  -1

octave:8> abs(a)
ans =

   5   4   3   2   1
```

FIGURE 10.15
abs(x) example.

```
octave:14> sin(a)
ans =

   0.9589   0.7568  -0.1411  -0.9093  -0.8415

octave:15> sind(a)
ans =

  -0.087156  -0.069756  -0.052336  -0.034899  -0.017452

octave:16> rad2deg(a)
ans =

  -286.479  -229.183  -171.887  -114.592   -57.296

octave:17> deg2rad(a)
ans =

  -0.087266  -0.069813  -0.052360  -0.034907  -0.017453
```

FIGURE 10.16
Example of trigonometric functions.

5. **abs(x) –** This function returns the absolute values of all the elements in the vector/matrix x. As shown in Figure 10.15, all the negative values are replaced as the absolute values in the ans vector.[8]
6. **sin(x)** – This function returns the sine value of all the elements in the vector/matrix x in radians.[9]
7. **sind() –** This function returns the sine value of all the elements in the vector/matrix x in degrees (Figure 10.16).[9]
   Similarly, you can also calculate all trigonometric values such as cosine, sec, tan, etc. in GNU Octave.
8. **rad2deg()** – This function converts the given value in radians to degree format.
9. **deg2rad()** – This function converts the given value in the degree format to radians format.

Note that GNU provides a lot of such in-built functions that make it easy to perform arithmetic operations in GNU Octave.

10.3 Set Operations

GNU Octave has many functions to manage huge datasets. A set is nothing but a collection of distinctive elements of a vector or a matrix.[1] A vector or a matrix can be converted to a set just by removing the duplicate values using the unique function.[1] Moreover, it is not mandatory to use the unique function as all the set functions in GNU Octave removes the duplicate values before operating.[1]

1. **unique(x)** – This function removes all the duplicate values of the vector/matrix x.

2. **union(M, N)** – This function merges both the set M and N. Note that duplicate values are not included in the union of the set.
3. **intersect(M, N)** – This function extracts the common elements from both the set M and N. Note that duplicate values are not included in the intersection of the set (Figure 10.17).
4. **setdiff(M, N) –** This function returns a set containing elements that are present in set M and does not exist in set N.
5. **setxor(M, N) –** This function returns a set containing all the elements from set M and set N except the common elements.
6. **ismember(M, N)** – This function returns a logical matrix containing 1 and/or 0. M is a set and N can be a number or a set of values. If the number is present in matrix M, it will place 1 or otherwise 0. The resultant matrix is of the same size as that of set M (Figure 10.18).

```
octave:6> a = [ 2,3,4,2,3,5,6,4,6]
a =

   2   3   4   2   3   5   6   4   6

octave:7> unique(a)
ans =

   2   3   4   5   6

octave:8> b=[6,7,8,9,10]
b =

    6    7    8    9   10

octave:9> union(a,b)
ans =

    2    3    4    5    6    7    8    9   10

octave:10> intersect(a,b)
ans = 6
```

FIGURE 10.17
Example of the union and intersection of the set.

```
octave:8> setdiff(a,b)
ans =

   2   3   4   5

octave:9> setxor(a,b)
ans =

    2    3    4    5    7    8    9   10

octave:10> ismember(a,[3:4])
ans =

  0  1  1  0  0
```

FIGURE 10.18
Example of setdiff(), setxor(), and ismember().

10.4 Plotting Data

The most basic plot in GNU Octave is created using the plot() function provided by GNU Octave. The plot() function takes the two inputs, i.e., the cartesian values x and y, and plots the graph. You can add specifications to the graph by adding the title to the plot, the axis label, style of the plot, and much more. One such graph is shown below where a sin(x) plot is plotted against x.

Octave:

```
1>x = 0:0.1:2*pi;
y = sin(x);
plot(x, y, "*r", "markersize", 2);
xlabel("x");
ylabel("y = sin(x)");
title("Basic plot example");
```

In this plot, we have used the "*" marker, but GNU Octave provides many marker options. (We will discuss it further). The marker size is the size of the marker.[10,11] You can modify your plot according to the requirements of the user (Figure 10.19).

The format arguments that can be used within the plot() function are given as below:

Linestyle:

1. "-"– Solid lines
2. "."– Dotted lines
3. "--" – Dashed lines
4. "-." – Dash-dotted lines

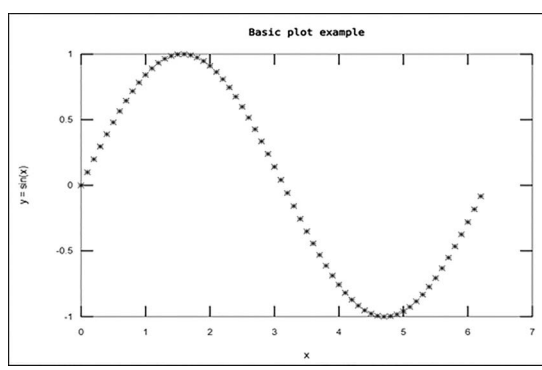


FIGURE 10.19
Basic plot example.

Marker[12]:

1. "+" – Crosshair
2. "*" – Star
3. "x" – Cross
4. "d" – Diamond
5. "v"– Downward-facing triangle
6. "<" – Left-facing triangle
7. "h" –Hexagram
8. "o" – Circle
9. "." – Point
10. "s" – Square
11. "^" – Upward-facing triangle
12. ">" –Right-facing triangle
13. "p" – Pentagram

Color[13]:

1. "k" – Black
2. "b" – Blue
3. "y" – Yellow
4. "c" – Cyan
5. "r" – Red
6. "g" – Green
7. "m" – Magenta
8. "w" – White

"label_plot": Note that label_plot is the name given to the plot legend. Also, the semicolons used in the syntax of the plot legend are mandatory.

Pie Charts

Pie charts are often used to show the data proportions in the data, i.e, the contribution of a particular category in a set of data. In GNU Octave, a pie chart is created using the pie function. The syntax of the pie() function is as follows:

Syntax: pie(x, explode, labels)

Attributes:

- *x – x is the vector based on which the pie chart is plotted. Here, the size of the slice is calculated based on the percentage it contributes to the total data.*
- *explode – explode is a vector of the same size as that of the input vector x, and it contains elements as 0(zero) or any nonzero element.[7] A nonzero element represents that slice of the pie to be exploded.*
- *labels – The label cell array represents the label to each slice.*

Octave:2>

```
pie ([17, 52, 41,28,2], [0,3,0,0,0], {"Action", "Sci- Fi",
"Drama", "Romance", "Comedy"});
title ("Pie Chart for Favorite Movies")
```

In the above example, a pie chart for favorite movies is created using the pie()function. You can see that the "Sci-Fi" slice of the pie is exploded as specified in the explode vector (Figure 10.20).

One important factor to remember in a pie chart is the missing slice.[14]

When the sum of all the elements in a vector is less than 1(sum < 1) and such vector is given as an input vector to the pie() function, in such cases, the Octave would not calculate the percentage proportion for each slice; it will use the vector elements itself to proportionate the data, leaving a missing slice for the part of the data that is undefined.[7,15]

Octave:3>

```
pie ([0.17,0.61,0.2], {"Coffee", "Tea", "Milk"});
title ("Pie Chart for Beverage Preferences")
```

In the above example, you can see that the input vector contains three elements – 0.17, 0.61, and 0.2. The sum of these elements is 0.98 (0.98 < 1). So here, Octave does not calculate the sum and the respective percentages and considers the elements to be taken in percentages and keeps a missing slice at the last (Figure 10.21).

Scatter Plot

In GNU Octave, a scatter plot is created using the scatter() function. The syntax of the scatter() function is given as the following:

Syntax: scatter (x, y, s, c, style, "filled")[1]

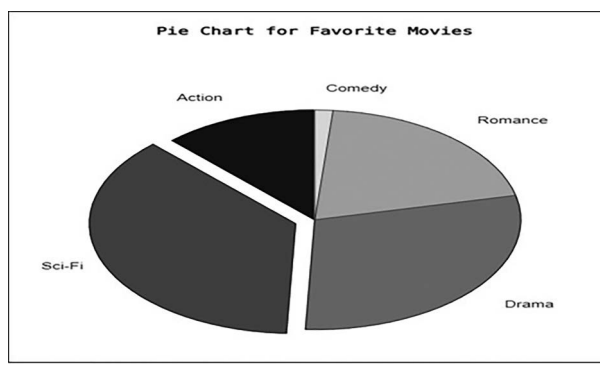


FIGURE 10.20
Pie chart example.

Attributes:

- *x, y – x and y are the vectors that are used to plot the markers.*
- *s – s is the marker size. It is computed as sqrt(s). The default value of s is 36 sq. points.*[12]
- *c – c represents the color of the marker. The notations for the different colors are mentioned above in the format arguments.*
- *style – style represents the shape of the marker. The notation for various styles of the marker is covered in the format arguments.*
- *"filled" – If this attribute is specified, the marker will be filled or else it will be a hollow marker* (Figure 10.22).

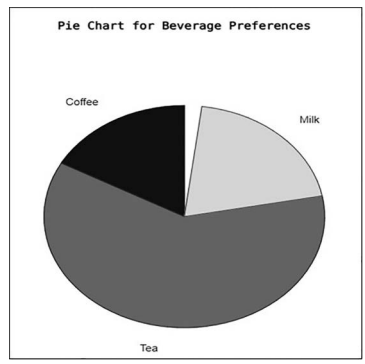


FIGURE 10.21
Example of the pie chart missing slice.

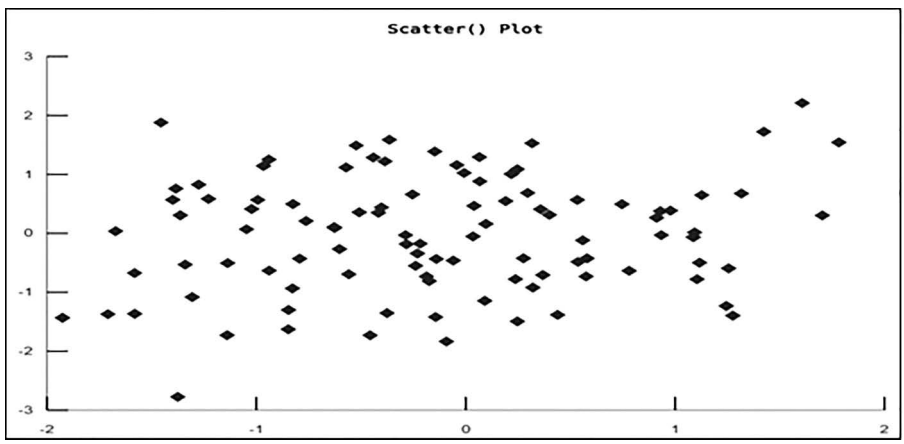


FIGURE 10.22
Scatter plot.

Octave:

```
4>1x = randn (100, 1);
y = randn (100, 1);
scatter (x, y, "m", "d", "filled");
title ("Scatter() Plot");
```

Histogram

Now, let's see how to build a histogram in GNU Octave.

In GNU Octave, the histogram is created using the hist() function.

The hist() function mainly needs one input which is the Y-values vector. Other property parameters can be included or excluded as per the requirements of the user. If the number of bins to be considered is not specified by the user, by default, Octave generates ten bins (Figure 10.23).

Let's create a histogram.

Octave:

```
5>1A = hist (randn (10000, 1), 30);
xlabel ("Value");
ylabel ("Count");
title ("Histogram");
```

Now, as you can see in the above source code, we have generated a vector of 10,000 random numbers and provided it to the hist() function to create a histogram along with 30 bins.

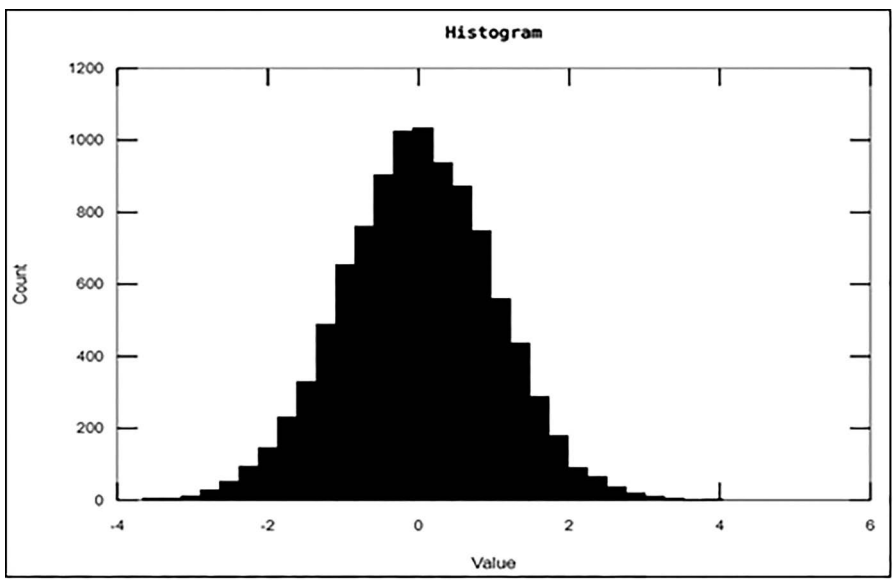


FIGURE 10.23
Histogram example.

```
octave:31> A
A =

 Columns 1 through 11:

      1      4      5      9     21     44     93    163    220    328    464

 Columns 12 through 22:

    656    814    917    908   1022   1000    868    696    581    406    296

 Columns 23 through 30:

    206    142     69     32     14     14      2      5
```

FIGURE 10.24
Storing the number of elements in the bins.

If you observe the source code, the hist() function is assigned to array A. This is carried out to store the number of elements in every bin. As we had given the input to create 30 bins, in the above figure, you can see that the array consists of 30 elements, and each element represents the number of items stored in that particular bins. The addition of these elements will sum up to 10,000 as we had generated 10,000 random numbers to create the histogram (Figure 10.24).

10.5 Summary

GNU Octave is profoundly used for numerical computations as numerical computations and vector/matrices go hand in hand. We first studied various techniques to create a vector and extract the elements of the vector.[1] We also looked at a few built-in functions to create vectors. We then moved on to study many ways to create matrices and built-in functions to manipulate it. GNU Octave provides multiple built-in functions for arithmetic operations. These arithmetic operations are highly beneficial as they ease out the process and cut down the lines of code to just a single-line function. GNU Octave also provides functions to perform set operations such as the union and intersection of sets. To wrap-up the chapter, we went through some plotting techniques provided by GNU Octave and its different formatting arguments.

Exercise

1. Explain the various ways of creating vectors and matrices.
2. What functions are used to perform operations on the matrix?
3. Define the compatible matrix with an example.

4. Compute the trigonometric functions in radians and degrees using respective functions and also by changing from degrees to radians and vice versa.
5. Create two sets, and explicitly remove the duplicate values.
6. Create a scatter plot that contains the following specifications of the marker:

   i. Style – Downward-facing triangle
   ii. Color – Green
   iii. Marker type – Filled

References

1. Eaton W. John, Bateman David, Hauberg Søren, Wehbring Rik. 2018. *GNU Octave: Free your Numbers*. United States: Free Software Foundation.
2. Long P.J.G. 2005. Introduction to Octave. http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave/octavetut.pdf (accessed 21 Decemer 2020).
3. NgAndrew. 2012. Octave tutorial.
4. Palamides Alex, Veloni Anastasia. 2011. *Signals and Systems Laboratory with MATLAB*. Boca Raton: CRC Press.
5. Kim Je Heon James. *Linear Algebra*. New York: Columbia University.
6. Lachniet Jason. 2018. Introduction to GNU Octave. United States.
7. Ongsakul Weerakorn, Dieu Ngoc Vo. 2013. *Tips for Programming in MALAB. Artificial Intelligence in Power System Optimization*. Boca Raton: CRC Press.
8. Higham J. Nicholas. 1998. Fast solution of Vandermonde-like systems involving orthogonal polynomials. *IMA Journal of Numerical Analysis*: 473–486. https://doi.org/10.1093/imanum/8.4.473.
9. Abowd, John M., Schmutte M. Ian. 2019. An economic analysis of privacy protection and statistical accuracy as social choices. *American Economic Review* 109(1): 171–202. Doi: 10.1257/aer.20170627.
10. Hansen Jesper Schmidt. 2011. *GNU Octave: Beginner's Guide*. Birmingham: Packt.
11. Rogel-Salazar Jesús. 2015. *Essential MATLAB and Octave*. Boca Raton: CRC Press.
12. Eaton W. John. 2018. Two-dimensional plots. GNU Octave. https://octave.org/doc/v4.2.2/Two_002dDimensional-Plots.html
13. Doing algebra with maxima. https://sysplay.in/blog/tag/maths/page/2/ (accessed 25 December 2020).
14. Nagar Sandeep. 2016. Introduction to Octave: For scientists and engineers.
15. Pratap Rudra. 2010. *Getting Started with MATLAB*. New York: Oxford University Press.

11

Data Visualization Using Tableau

11.1 Introduction to Data Visualization

Data visualization is a process used to represent the data in the graphical format. It is because of data visualization that various decision makers can gain insights as well as patterns and analyze it from the data represented visually. Data visualization makes us think differently about the information to develop value out of it.

Performing data visualization is not an easy task. One needs to be very careful when using different data visualization techniques and plotting the data. Various decision makers and stakeholders consider three major factors while performing data visualization:

1. **Clarity** – Clarity is achieved when the data to be visualized is appropriate and thorough. The absence of clarity in data yields false patterns which may lead to wrong implications on the new appropriate data.
2. **Accuracy** – Once you have achieved clarity, it is important to use suitable pictorial representation.
3. **Efficiency –** Efficiency plays a very important role in data visualization. One should ensure using proper data visualization techniques that showcase all the necessary data points.

Advantages of Data Visualization:

1. It helps to recognize relationships among various variables in the data.
2. It makes it easier to perform mathematical calculations.
3. It is beneficial to analyze and explore the existing patterns, thus giving out the invisible patterns.
4. It helps to recognize the areas that need to be improved.

DOI: 10.1201/9780429443237-14

There are many data visualization tools in the market which make the task of visual representation of the huge datasets significantly easier. These tools are used to display data in dashboards, reports, or simply anywhere you want to interpret data visually. Some of such data visualization tools are Whatagraph, Power BI, Tableau, etc. In this chapter, we will be looking at the Tableau software as the data visualization tool.

11.2 Introduction to Tableau

Tableau is a strong and rapidly expanding data visualization tool. It is highly used in the business intelligence domain. Tableau simplifies raw data into a format that is simple and easily intelligible. Tableau makes data analysis easy and rapid. Data visualizations created in the Tableau are along the lines of worksheets and dashboards.

The user does not need to be familiar with any technical knowledge or programming skills to use the Tableau software; thus, this data visualization tool has attracted many people from various domains.[1]

Important features in Tableau:

1. **Data Blending –** Data blending plays a vital role in data analysis and visualization and hence is the supreme feature in Tableau. Data blending comes in a role where we merge the relevant data from various sources to analyze it in one view and portray it visually using graphs.
2. **Real-Time Analysis** – When the speed is high and the dynamic real-time analysis of the data becomes complex, real-time analysis helps the user to rapidly comprehend and analyze the real-time data. Thus, Tableau helps in extracting important information from the rapidly moving real-time data with collective analysis.
3. **A Collaboration of Data –** Data analysis is not a separating task; it needs a collaboration medium for the team members, and this is the reason that Tableaus provide us with the collaboration of data. Team members can share various forms of data, take follow-ups from one another, and share easy visualizations so that others get values from the data. It is critical to make sure that everybody can comprehend the data and conclude vital decisions.[2]

Analytics Cycle

A person who goes well along with data and tries to comprehend the data moves within the analytics cycle. The analytics cycle can be explained as below:

Because of the Tableau, you can move candidly within the steps and iterate along the analytics cycle and get answers to the questions or even frame new questions.

1. **Data Discovery** – Exploring the data becomes easier with Tableau and understand the data that can be plotted visually.
2. **Data Preparation** – As mentioned above, data from various sources can be merged in Tableau. Tableau provides various functionalities that make complex and messy data come together.
3. **Data Analysis** – Data visualization is easier with Tableau. With Tableau, one can get familiar with the trends, recognize the outliers. Tableau provides us with various functionalities that allow us to get deep into complex patterns and correlations of data.
4. **Data Storytelling** – With Tableau, we can build dashboards and stories along with visualizations so that the audience understands the story behind your visualizations better (Figure 11.1).

Before we begin with Tableau, every aspiring student should know about the advantages and disadvantages of the Tableau.

Advantages of Tableau:

1. Tableau creates an automatic dashboard. Also, when the dashboard is to be opened on the mobile phone, it automatically identifies the handset and adjusts the dashboard accordingly.
2. Tableau can handle millions and trillions of data without slowing down the dashboard performance.
3. Complicated computations can be performed by incorporating Tableau with Python and R.

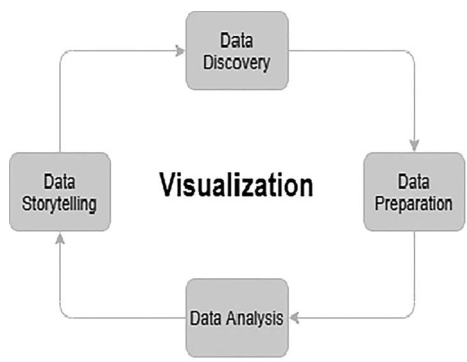


FIGURE 11.1
Analytics cycle.

Disadvantages of Tableau:

1. Tableau does not provide total security. One can only get row-level security.
2. Tableau does not automatically update the Tableau reports. One has to always manually do it at the back end.
3. Similar formatting for numerous fields cannot be performed at the same time.

11.3 Dimensions and Measures, Descriptive Statistics

Dimensions and Measures

Once you have imported the data file into Tableau, the connections and the fields in the data are displayed in the data pane on the left of the screen. Different fields are dragged on the canvas area as per the requirements. The placements of the fields onto various shelves, such as the Rows shelf and Columns shelf, will result in diverse results.[3]

The fields of the data source are placed under dimensions and measures by Tableau itself when displayed in the data pane. When using Tableau, it is essential to know what dimensions and measures exactly are:

1. **Dimensions –** Dimensions are the fields that have qualitative information (values). Dimensions are mostly used to classify and divide the details of the data. Dimensions unfold the details in the data and control the intensity of details to be displayed in the visualization.
2. **Measures –** Measures are referred to as the fields that have quantitative values that can be used to perform numeric calculations such as sum, average, count, minimum, maximum, standard deviation, etc. By default, when a measure is drawn to the canvas area, the aggregation method is applied to it by Tableau.

In the Tableau,[3,4] the fields are represented using two colors–blue and green, when they are dragged to the respective shelves. The green fields (dimensions and measures) represent the continuous nature of the values. A continuous field defines the axis in the view. The blue fields (dimensions and measures) represent the discrete nature of the value that is all the unique values that are different from one another. The discrete field defines headers in the view.

Let's take an example of the Superstore data from the Tableau repository. Suppose, I want to look at the profit earned by each category. As shown below, you can see that the dimension field named Category is added to the Columns shelf, and in the bar chart below, it has added a header in the view. Similarly, the measured field named Profit is added to the Rows shelf and this field adds an axis in the view (Figure 11.2).

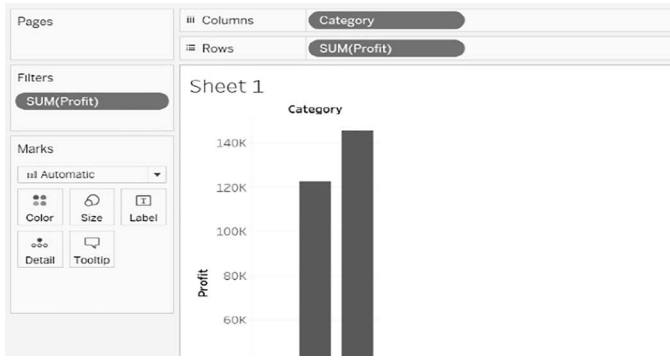


FIGURE 11.2
Dimensions and measures example.

Also, one can notice that the category field is a discrete (blue) field and has unique values – furniture, office supplies, and technology, whereas the profit field which is a measure is continuous (green), and it has various values within the profit range.

One common misunderstanding that everyone makes is that they consider dimensions and measures to be the same as discrete and continuous fields. Most of the time, we come across discrete dimensions and continuous measures. Moreover, you can convert a continuous measure to a discrete measure and a few discrete dimensions to continuous dimensions.

Descriptive Statistics

Statistical analysis plays a pivotal role in data science. Descriptive statistics are on high demand as it assists review, analyze, and gain insights into the data. Descriptive statistics is a method that summarizes and describes the data to gain insights into them. Descriptive statistics are the numbers that describe the data, and this description is not used to conclude anything.

As we know, there are two types of descriptive statistics; the first is the measures of central tendency such as the mean, median, and mode, and the second is the measure of dispersion such as the standard deviation and variance.[5] For a data scientist, it is mandatory to know these techniques. Tableau provides us with easy statistical functions that help to perform these techniques much easier.[6] Let's take a look at these descriptive statistics methods in Tableau using the sample superstore dataset from the Tableau repository.

1. Mean

   The mean, also known as the average of the data, is calculated by adding all the values and dividing by the number of values.[7] In the Tableau, it is easy to calculate the mean using the **average** function.

   Suppose, we want to know the average sales in each category. To calculate such an average, we will need to execute the following steps:

Step 1: Drag the dimension Category and the measure Sales to the canvas area.[8]

Step 2: To get the average, right-click on the Sales field and select the Measure (Average) method.[8]

Step 3: To get the average to be displayed on the respective bars, drag the Sales field from the canvas area itself to the Label property under the Marks shelf (Figure 11.3).[8]

2. Median

The median is the middle value in the data.[8] It is the value that divides the dataset into two parts. The median is also known as the[9] 0^{th} percentile of the dataset.

Say, we want to know the median profit in each category. Execute the following steps to get the median:

Step 1: Drag the dimension Category and the measure Profit to the canvas area. (As shown in the picture below, we have used the Packed Bubbles chart.)

Step 2: To get the median, right-click on the Profit field and select the Measure (Median) method.[8]

Step 3: To get the packed bubbles in different colors and in different sizes, drag the Category field to the **Color** property and the MEDIAN(Profit) to the **Size** property under the Marks shelf, respectively.

Step 4: To get the median to be displayed on the respective bubbles, drag the Category and the Profit field from the canvas area itself to the Label property under the Marks shelf (Figure 11.4).

Similarly, we can calculate the below descriptive statistics methods in Tableau (Figure 11.5).

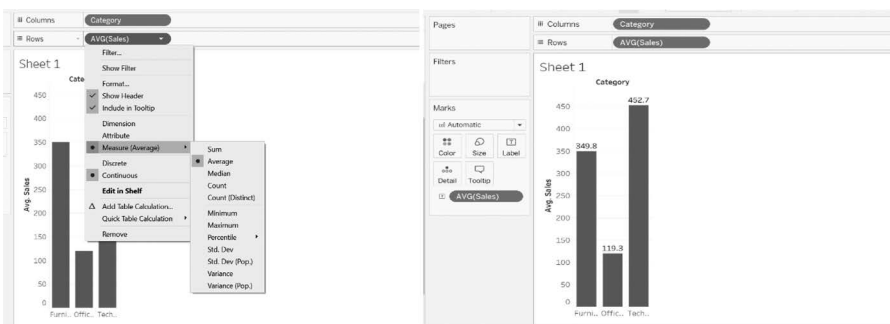


FIGURE 11.3
Average function example.

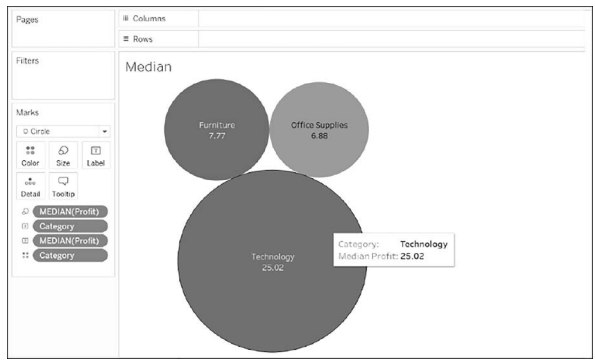


FIGURE 11.4
Median function example.

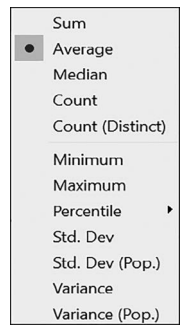


FIGURE 11.5
Descriptive statistics functions provided by Tableau.

Worksheet Summary Card

Tableau also provides us a feature known as the Summary Card. The Summary Card gives us a summary of the visuals we have plotted in the canvas area. The Summary Card provides all the descriptive statistics methods we covered above.

Let's take the example that we did while calculating the median using the packed bubbles. To get the Summary Card for this example, the following steps must be carried out.

Step 1: In the tab window, right-click the Worksheet tab and select Show Summary. Once you click on the Show Summary, the Summary Card will be activated on the right of the screen.

Step 2: You can place this Summary Card anywhere by just dragging it on the screen.

Step 3: Right-click on the Summary tab in the Summary Card and select or deselect the metrics as per the requirements (Figure 11.6).[10]

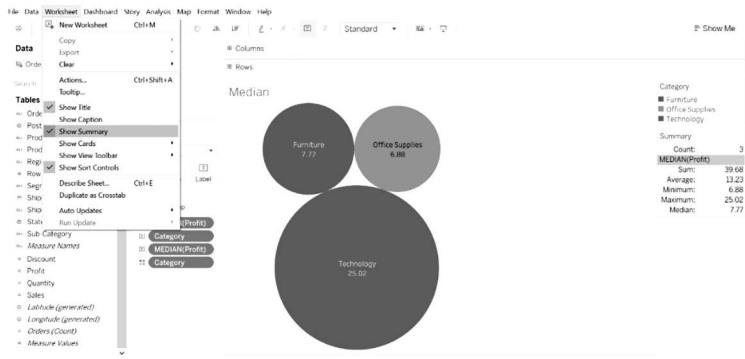


FIGURE 11.6
Worksheet Summary Card example.

11.4 Basic Charts

Tableau can provide interactive charts. Different types of charts are created using various measures and dimensions. The Show Me tab suggests various chart types based on the measures and dimensions selected by the user.[10]

A bar chat appears on the canvas area by default when a dimension is selected followed by a measure. Similarly, a textual representation of data is displayed on the canvas area when a measure is selected followed by a dimension.

The following are the various types of basic charts in Tableau.

1. Bar chart
2. Area chart
3. Line chart
4. Bullet chart
5. Scatter plot
6. Box-whisker plot
7. Pie chart
8. Bubble chart
9. Histogram
10. Highlight table
11. Gantt chart
12. Heat map

In this chapter, we will go through a few types of charts that are mentioned above:

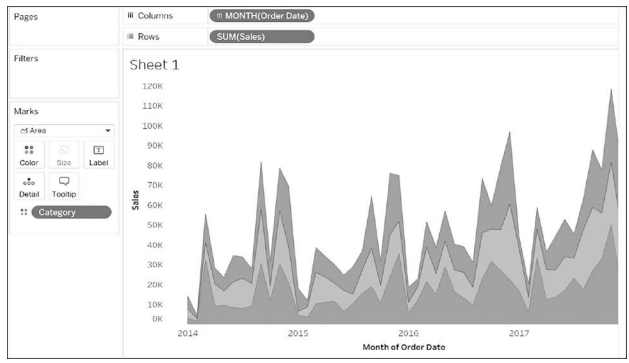


FIGURE 11.7
Area plot.

1. Area Chart

   Area charts are mostly used to analyze numeric data over different time periods. The area chart is just an extension to the line chart where the area between the lines and the axis is filled with color.

   Suppose, we want to analyze the Sales per category over the Order Date. We will create an area chart to analyze such a scenario.

   Execute the following steps to create an area chart:

   Step 1: Drag the dimension Order Date to the Columns shelf and the measure Sales to the Rows shelf.

   Step 2: Change the chart type from Automatic to Area under the Marks pane.

   Step 3: Change the Order Date type to Month.[8]

   Step 4: Drag the dimension Category to the Colors property under the Marks pane (Figure 11.7).[8]

2. Scatter plot

   Scatter plots are used to find relationships between two or more variables. Scatter plots are majorly used to find out trends in the data. A scatter plot displays discrete data points on the graph.

   Say, we have to find the relationship between the sales and the profit based on the subcategory. To create a scatter plot, perform the following steps:

   Step 1: Drag the measures Sales and Profit to the Columns and Rows shelves, respectively.[8]

   Step 2: Drag the dimension Sub-Category to the Colors property under the Marks pane (Figure 11.8).[8]

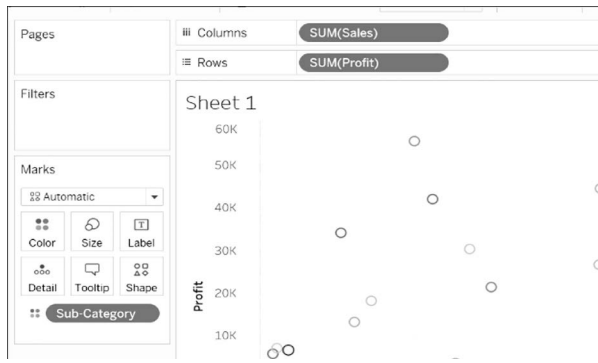


FIGURE 11.8
Scatter plot.

3. Box-whisker Plot

Box-whisker plots are also known as the boxplots. Box-whisker plots are normally used to showcase the data distributions. The box in the plot represents the first, second, and third quartiles.[11] The whiskers represent the maximum and the minimum data points of the data.[11]

Let's learn to plot a box-whisker plot in Tableau.

Step 1: Drag the dimension Segment and measure Discount to the Columns and Rows shelves, respectively.[8]

Step 2: Drag the dimension Region to the right of the Segment in the Columns shelf.[8]

Step 3: Select the box-whisker plot under the Show me tab.[8]

Step 4: Replace the dimension Region in the Marks shelf to the right of the dimension Segment under the Columns shelf.[8]

Step 5: Deselect Aggregate Measures under the Analysis tab.

Step 6: Edit the colors by right-clicking the vertical axis and selecting Edit Reference Line (Figure 11.9).[10]

4. Heat Map

Heat maps are majorly used to analyze the data with various categories using colors. The heat map can compare hundreds and thousands of categories and make it simple to interpret.

Suppose, using the Sample Superstore dataset, we want to analyze the trends in the data of the Sales in accordance with the Order Date by plotting a line graph. Such a line graph will create an overlapping of lines which will cause trouble to analyze the graph. In such cases, a heat map is a better alternative to find the trends in data.

Perform the following steps to create a heat map in the Tableau.

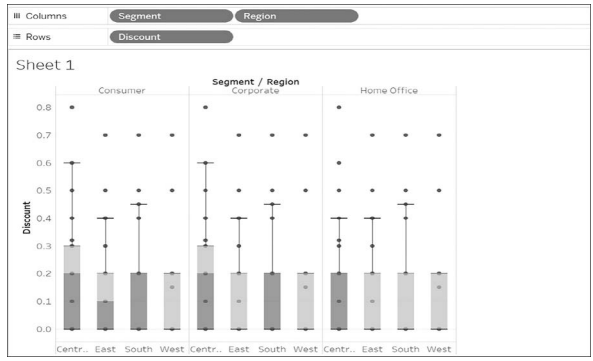


FIGURE 11.9
Box-whisker plot.

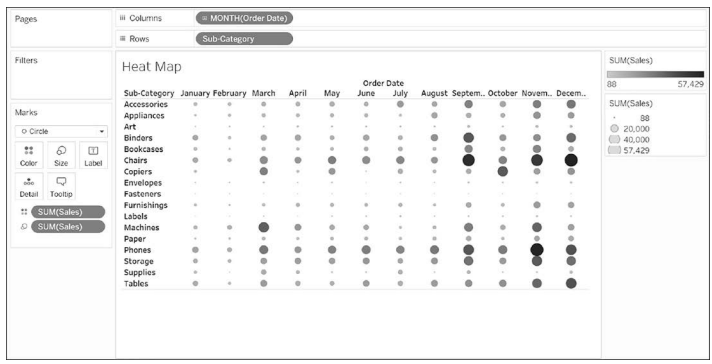


FIGURE 11.10
Heat map.

Step 1: Drag the dimension Order Date and dimension Sub-Category to the Columns and Rows shelves, respectively.

Step 2: Change the Order Date type to Month.[8]

Step 3: Change the Mark type to Circle under the Marks tab.[8]

Step 4: To get the encoding, drag the measure Sales to the Colors and Size property under the Marks tab (Figure 11.10).[8]

11.5 Dashboard Design & Principles

A dashboard is an interactive representation of united worksheets and all the essential information. Dashboards are mainly used to compare various data at one go.

Let us create a dashboard from the Sample Superstore dataset to analyze the profit performance by the country.

Step 1: Go to the new dashboard worksheet by navigating the Sample Superstore sheet.

Step 2: Drag the sheets from the left of the dashboard section to the canvas area. You can also add them by double-clicking the sheets.

Step 3: Check the Show Dashboard Title and add the dashboard title.

Step 4: Fit the view with respect to the available space.

Step 5: Place the Profit legend with respect to the sheet.

Step 6: Take your time to interact with the dashboard (Figure 11.11).

Dashboard design principles:

Dashboard design principles play a vital role in the process of creating a dashboard. Using the below principles, one will easily be able to make interactive, data-driven dashboards.

1. Be the audience
   You should always think from the side of your audience, the group of people who are going to view your dashboard on a laptop or a projector screen.
2. Relevant dashboard
   Create dashboards that can fit into every type of screen, be it smartphones, laptops, tablets, etc.
3. Use minimal types of color
   Be consistent with the colors, and use the gradients of the relevant colors.

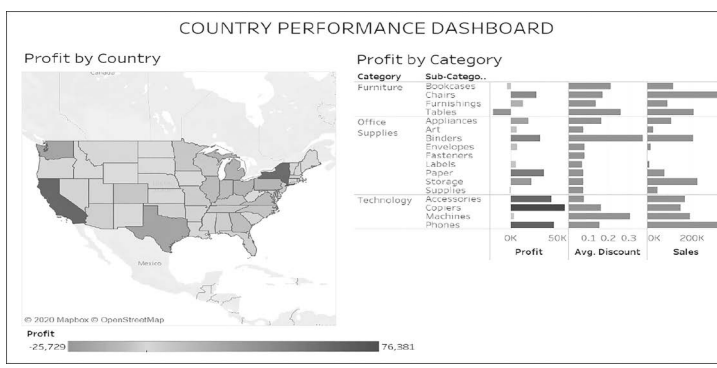


FIGURE 11.11
Country performance dashboard.

4. Layout design
   The most significant trend should be easily visible to the user. Place the comparative views beside each other.

11.6 Special Chart Types

All Tableau users know the basic charts and graphs that can be easily plotted using the **Show me** tab which contains various charts, such as line chart, pie chart, heat maps, treemaps, etc.

To make the visualization more interactive on the dashboard, Tableau also provides some advanced graphs that need a bit more computations than the graphs shown under the **Show me** tab.

The following are the advanced graphs that we are going to cover in this chapter:

1. Motion chart
2. Bump chart
3. Waterfall chart

Let's see each of these charts in detail.

1. Motion Chart
   A motion chart is nothing but a chart that shows the transformation in data dynamically.
   Suppose, we want to dynamically see the trends of profit and sales over the years. A motion chart is created in the Tableau by performing the following steps:

   Step 1: Drag the dimension Order Date to the Columns shelf and measures Profit and Sales to the Rows shelf.

   Step 2: Change the format of the Order date to Month-Year by right-clicking on it and then clicking on Custom under the More tab as shown below (Figure 11.12).

   Step 3: Drag the dimension Order Date to the Pages shelf and change the format accordingly.

   Step 4: To recognize the current trend in the run-time motion chart, change the mark type under the Marks shelf from Automatic to All.[10]

   Step 5: Set the show to Trails under the Show history visible on the right side of the window.

   Now, play the motion chart. You can control the motion chart by the tab that occurs on the right side of the window (Figure 11.13).

FIGURE 11.12
Changing the format for the motion chart.

FIGURE 11.13
Motion chart example.

2. Bump Chart

Say, you want to look at the sales of different categories in a year. You can do this using a basic line chart, but a bump chart will give you more detailed results of the same problem. Bump charts are mostly used to get an idea of the change in the approval of products over years.

Let's learn to plot a bump chart:

Step 1: Bump charts are created based on ranks. So, first, create a calculated field as shown below[8] (Figure 11.14):

Step 2: Drag Order Date to the Columns shelf and change its format to Month. Drag Category to the Colors property under the Marks pane. Also, drag the newly created calculated field Rank to the Rows shelf.[8]

Step 3: To allocate the rank by the categories, right-click the Rank → Compute Using → Category.

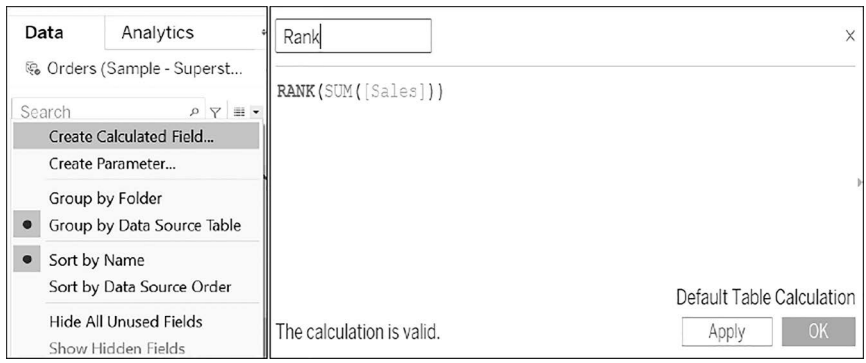


FIGURE 11.14
Creating a calculated field.

Step 4: Drag the calculated field Rank to the Rows shelf again and repeat Step 3.

Step 5: Right-click on the second Rank in the canvas area, and select Dual Axis.[10]

Step 6: Alter the Mark type to Circle of either of the Rank under the Marks pane.

Step 7: Ranks can be reversed by the following navigation – Click on the axis → Edit Axis → Reversed Scale. The bump chart is ready (Figure 11.15).

3. Waterfall Chart

A waterfall chart is exactly what it means; it depicts the flow between certain statistics.

FIGURE 11.15
Bump chart.

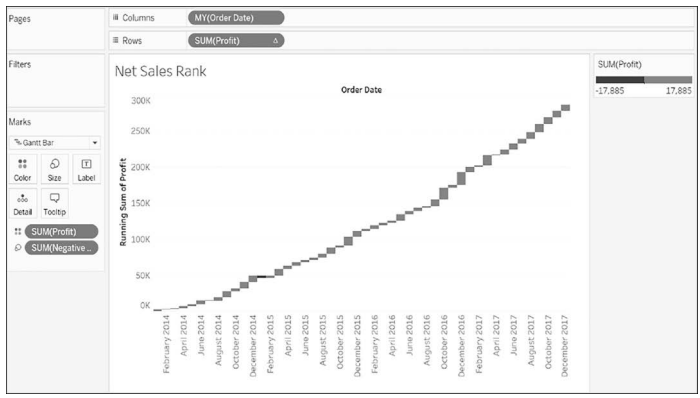


FIGURE 11.16
Waterfall chart.

A line chart is replaced by a waterfall chart as it is more interactive. A waterfall chart is used to analyze a measure – its rise and fall over the years.

Here, we will see the flow of sales over the years. To achieve this, execute the following steps:

Step 1: Drag the Order Date and Profit to Columns and Rows shelves, respectively. Change the format of the Order Date to Month-Year and for the Profit, convert the Quick Table Calculation to Running Total.[8]

Step 2: Under the Marks pane, change the Mark type to Gantt Bar. You can also alter the size accordingly.[8]

Step 3: Create a calculated field named Negative Profit and drag it to Size under the Marks pane. This field will fill the Gantt chart.

Step 4: Drag Profit to the type Color under the Marks pane (Figure 11.16).

11.7 Integrate Tableau with Google Sheets

Google Sheets is a spreadsheet utility bestowed by Google. Google Sheets is a web-based application and free of cost. It is now possible to integrate Google Sheets in Tableau using the Google Sheets connector. Tableau is strongly supported by Application Programming Interface (API) which helps us to get a seamless connection and gather productive insights.

FIGURE 11.17
Integrating Google Sheets in Tableau.

To integrate Google Sheets in Tableau, execute the following steps:

Step 1: Open the Tableau application on your device, go to the connect column, and select Google Sheets under the To a Server tab.

Step 2: After Step 1, you will be directed to a page asking to sign into your Google account.

Step 3: Grant Tableau permission to access the Google Sheets.

Step 4: Select the desired Google Sheet to integrate with Tableau (Figure 11.17).

Although one can now connect Google Sheets in Tableau, it does have certain limitations. They are as follows:

i. Tableau is not supported by dynamic updates. Changes made in the Google Sheet would not be reflected in Tableau immediately.
ii. Composite transformations are not possible in Tableau; it affects the performance.
iii. Using joins to connect various data sources, including Google Sheets, is a tedious task.
iv. Team-drive accounts cannot be used to integrate Google Sheets in Tableau.

11.8 Summary

This chapter covers the concept of data visualization using Tableau. We began with a short introduction to data visualization and Tableau to provide insights to the reader into what they can expect in the chapter. Working with

data is important in Tableau, and that is the reason to learn about the dimensions and measures used in Tableau. We also covered the different descriptive statistics methods that are provided by Tableau.

Tableau is all about building charts and gaining insights and trends from them. Further in the chapter, we learned about some basic charts such as the histogram, box-whisker plots, etc.; in the further topics, we also went through a few advanced charts, which can be used as an alternative to the basic charts, that provide an interactive design.

In the Tableau, once all the worksheets are completed, it is time to place them together in the dashboard as it becomes easy for the user to compare various visualizations. Although it may sound simple to create a dashboard, there are certain dashboard principles that one should consider for a data-driven dashboard. To wrap-up this chapter, we also learned to integrate Google Sheets with Tableau and its certain limitations in Tableau.

Exercise:
Use the Sample Superstore dataset (Questions 1–5)

1. What is the maximum and minimum amount of profit earned by the category Technologies?
2. Name the Sub-Category that had the highest number of discounts in the year 2017.
3. Name the month and the year of the Order Date which faced the highest loss for the category Furniture.
4. Plot a pie chart demonstrating the profits gained based on Ship Mode.
5. Write the steps to get the visualization as shown below and comment on it (Figure 11.18).

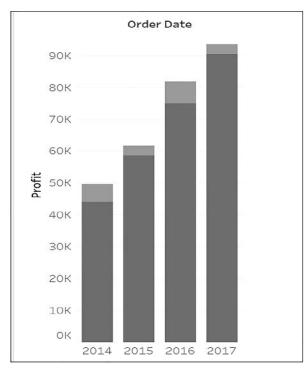


FIGURE 11.18
Exercise question.

6. Using any suitable dataset, plot a line graph and a heat map and comment on it.

   Use the Coffee Chain dataset (Questions 7–10)

7. Create a motion chart to analyze the trends in the budget profit based on the month of the order date.
8. Which state in the West Market had the highest sales in the year 2013?
9. Which product gained the minimum profit?
10. Create a dashboard for the Coffee Chain dataset, and list down the various trends in your visualization.

References

1 2018. Tableau desktop II: Intermediate. Tableau Software Inc.
2 Jones Ben. 2014. *Communicating Data with Tableau*. Sebastopol: O'Reilly Media.
3 Milligan N. Joshua. 2019. *Learning Tableau 2019*. Birmingham: Packt Publishing.
4 Loth Alexander. 2019. *Visual Analytics with Tableau*. United States: Wiley.
5 Conner Brian and Johnson Emily. 2017. Descriptive statistics. https://www.myamericannurse.com/research-101-descriptive-statistics/ (accessed December 3 2020).
6 Stirrup Jen. 2016. *Tableau Dashboard Cookbook*. United Kingdom: Packt Publishing.
7 Donnelly A. Robert. 2007. *The Complete Idiot's Guide to Statistics*. New York: Alpha Books.
8 Acharya Seema and Chellappan Subhashini. 2017. *Pro Tableau*. New York: Apress.
9 Quinn, J. Jeffrey, Ilik, Ibrahim, Qu, Kun, et al. 2014. Revealing long noncoding RNA architecture and functions using domain-specific chromatin isolation by RNA purification. *Nature Biotechnology* 32: 933–940. https://doi.org/10.1038/nbt.2943.
10 Murray G. Daniel. 2013. *Tableau Your Data!* Indianapolis: John Wiley & Sons.
11 2017. Tableau desktop I: Fundamentals. Tableau Software Inc.

Index

Note: **Bold** page numbers refer to tables and *italic* page numbers refer to figures.