



Data Science — for — **Business Professionals**

A Practical Guide for Beginners



PROBYTO DATA SCIENCE AND CONSULTING PVT. LTD.





Data Science — for — **Business Professionals**

A Practical Guide for Beginners



PROBYTO DATA SCIENCE AND CONSULTING PVT. LTD.



Data Science for Business Professionals

A Practical Guide for Beginners

by

**Probyto Data Science and
Consulting Pvt. Ltd.**



FIRST EDITION 2020

Copyright © BPB Publications, India

ISBN: 978-93-89423-280

All Rights Reserved. No part of this publication may be reproduced or distributed in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's & publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners.

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj

New Delhi-110002

Ph: 23254990/23254991

MICRO MEDIA

Shop No. 5, Mahendra Chambers,

150 DN Rd. Next to Capital Cinema,

V.T. (C.S.T.) Station, MUMBAI-400 001

Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,

Hyderabad-500195

Ph: 24756967/24756400

BPB BOOK CENTRE

376 Old Lajpat Rai Market,

Delhi-110006

Ph: 23861747

Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002
and Printed by him at Repro India Ltd, Mumbai

Dedicated to

*Students & Data Science Enthusiast
The Probyto Team Members, who are at
forefront of sharing their knowledge*

About the Author

Probyto Data Science and Consulting Private Limited (referred as Probyto) is leading solution provider in Artificial Intelligence (AI) domain for business from different sizes and industries. Probyto develops AI Solutions for businesses and delivers them through fully Managed AI platform. AI Platform enables businesses of any size to subscribe AI solutions and get started within 7 days. The vision of Probyto is to become “AI Success Partner” to our clients by “Accelerating the AI Journey” in quick, secure, scalable and affordable manner.

Probyto create AI equity by feeding in the value cycle, good talent, quality output and client value, hence creating AI equity for societies and companies. With Probyto AI resources bouquet, the innovation process is streamlined to innovate at scale.

- Founded in 2015 in India; Expanded our services to Ireland, Singapore and US
- Striving to deliver best value through our rich experience across geographies and industries using finest of Data Science and Technology
- Our approach is to become your AI Success Partner and help you climb the AI success ladder

Probyto activities and contribution in the field of AI are driven by three key goals.

- AI Democratization – Be it small shop or a large business the benefit of AI/ML should reach everyone
- Affordable AI - The cost of AI development and operations should allow higher adoption rate
- Good for Society - Whatever AI solutions Probyto develops, it needs to keep overall good of society at its core

To fulfil Probyto’s goals, the book has been written by collective experience of many of Probyto past client projects, academic collaborations and team

members for last 5 years. The collective work is represented by different experts in data driven decision making and portion they deal with in creating value for the clients. The team has experienced professionals and freshers who have gained from the approach as mentioned in the book as well.

Visit Probyto to know about our team and our offerings for academia & Industry: <https://probyto.com>

Acknowledgement

The book is not just a collection of topics in the Data Science domain but a journal of what Probyto team has learned in practical application of Data Science over past 5+ years implementing solutions and nurturing fresh talent.

This book has been possible by the support and work of a multidisciplinary team comprising of researchers, cloud architects, developers and business consultants at Probyto. Special mention goes to the team members who led the efforts for writing the book manuscript, Parvej Reja Saleh, Namachivayam Dharmalingam, Srivathshan KS, Devjit Dey, Jayeesha Ghosh and Md Rakibul Ashiquee. A special thank goes to Abhishek Singh from Probyto for facilitation of the whole effort with BPB.

The book learnings have been gathering by our numerous interactions with academic institutions, our interns, researchers and most important the clients. The feedback from clients help us build the right skillset in team and influence the freshers to look at data science as a tool to solve business problems rather than mastery of tools itself.

“I would like to express my deep gratitude to all the Probyto Team members for their valuable contribution in this book. I would like to thank Abhishek Singh, for his advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to the co-author Namachivayam for his contribution and constant support. Finally, I wish to thank SM Saleh (Father), Roushanara Saleh (Mother), BS Hasina (Aunt) and Late Saheda Akhtar (Aunt) for their constant support and encouragement.”

- Parvej Reja Saleh

“I personally thank Probyto and our team members who supports to share the knowledge to successfully write this book. Hope this book will make a good starting point of Data Science journey for the students”

- Namachivayam Dharmalingam

A big thanks for the team at BPB, for making this book possible for our freshers in India and Abroad. This book will open opportunities for students to see the Data Science domain from professional perspective and give them path to learn the valuable skills.

Preface

Data Science has emerged as a standalone industry itself serving needs of multiple other industries and sectors by providing valuable factual insights and automation of data driven tasks. Further, due to multiple reasons of which talent being most significant one, the adoption rate of Data Science is slower. It has been proven that data driven decision tools can reduce cost for companies' operations and at the same time create new markets.

Nowadays, the data science training programs are growing with high rate due to steep increase in demand of skilled candidates for open roles in Data Science domain. Data Science trainings offered by various platforms are designed to cover three crucial parts of skilling the freshers; theoretical concepts in Machine Learning, technology and programming skills, and the skills to create data-based solutions for business problems. For a fresher or enthusiast, accessing so many different aspects of data science is a challenge due to;

1. Too much and too varied content provided by platforms
2. Difficulty in stitching together technology, business and cloud skills to build a solution
3. Lack of innovative and real-world examples of application implementation

The core data science community has started to emphasise the need to re-structure the way we train the freshers by providing them a view of actual implementation of the end-to-end solution in a business set-up. It is important to equip the Data Scientist with the facts that “most accurate and optimised solution might not be the right solution in a dynamic business & technology environment”. Business value delivery is core to Data Science in any enterprise or in society.

This book tries to set first step in combining the complex parts of Data Science skills and their application in creating a real business solution. This include having enough knowledge of business processes, mathematics, technology and other technological innovation in cloud computing.

The book is divided into eight sections covering all aspects of creating value from data science in business set-up.

1. **Data Science Overview:** Explain everything a programmer needs to know about data science, from what data science is all about? Why data science is important? And how does data science work in real implementations.
2. **Mathematics and Statistics:** Introduce basics of linear algebra and its importance in solving data problems. Introduce basic mathematics to understand machine learning including optimization and calculus. Explain importance of statistics in data science. Cover key concepts of statistics required to solve data science problems.
3. **Machine Learning:** Introduces the basics of machine learning including exploratory data analysis, data preparation steps and algorithms for model training
4. **Data Engineering:** Introduces the concept of data pipelines and their significance. Also discuss how to build simple data pipelines. It also touched upon big data systems and databases.
5. **Cloud Computing:** Introduces the key enabling concept behind cloud computing – Hypervisors. It further will show with example how to work with cloud to put application on cloud for end-user use.
6. **Business Intelligence:** The business intelligence concepts are introduced what it is and how to make use of tools. Further, an example is built on Power BI to show how to frame business questions and answer them using visualisation tools.
7. **Industry Use Cases:** Two uses cases have been discussed at length to show how starting from problem we build a solution and put to use by end-user as an AI application.
8. **Self-Assessment:** The self-assessment is collection of typical questions and gaps that industry is looking for in people to hire them for entry level roles.

Downloading the code bundle and coloured images:

Please follow the link to download the *Code Bundle* and the *Coloured Images* of the book:

<https://rebrand.ly/bac0131>

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors if any, occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at:

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

1. Data Science Overview

[Structure](#)

[Objectives](#)

[Evolution of data analytics](#)

[Define data science](#)

[Domain knowledge](#)

[Mathematical and scientific techniques](#)

[Tools and technology](#)

[Data science analysis types](#)

[Data science job roles](#)

[ML model development process](#)

[Data visualizations](#)

[Result communication](#)

[Responsible and ethical AI](#)

[Career in data science](#)

[Conclusion](#)

2. Mathematics Essentials

[Structure](#)

[Objectives](#)

[Introduction to linear algebra](#)

[Scalar, vectors, matrices, and tensors](#)

[Scalar](#)

[Vectors](#)

[Matrices](#)

[Tensors](#)

[The determinant](#)

[Eigenvalues and Eigenvectors](#)

[Eigenvalue decomposition and Singular Value Decomposition \(SVD\)](#)

[Singular value decomposition](#)

[Principal component analysis](#)

[Multivariate calculus](#)

Differential Calculus

Sum rule

Power rule

Special cases

Trigonometric functions

Product rule

Chain rule

Quotient rule

Multiple variables

Partial differentiation

Total derivative

Integral calculus

Slices

Definite vs.indefinite integrals

The Gradient

The Jacobian

The Hessian

The Lagrange multipliers

Laplace interpolation

Optimization

The Gradient Descent algorithm

Conclusion

3. Statistics Essentials

Structure

Objectives

Introduction to probability and statistics

Descriptive statistics

The measure of central tendency.

Mean

Median

Mode

Measures of variability.

Range

Variance

Covariance

Standard Deviation

Measure of asymmetry

Modality

Skewness

Populations and samples

Central Limit Theorem

Sampling distribution

Conditional probability

Random variables

Inferential statistics

Probability distributions

What is a probability distribution?

Normal distribution

Binomial distribution

Poisson distribution

Geometric distribution

Exponential distribution

Conclusion

4. Exploratory Data Analysis

Structure

Objectives

What is EDA?

Need for the EDA

Understanding data

Categorical variables

Numeric variables

Binning (numeric to categorical)

Encoding

Methods of EDA

Key concepts of EDA

Conclusion

5. Data Preprocessing

Structure

Objectives

Introduction to data preprocessing

Methods in data preprocessing

[Transformation into vectors](#)
[Normalization](#)
[Dealing with the missing values](#)
[Conclusion](#)

6. Feature Engineering

[Structure](#)
[Objectives](#)
[Introduction to feature engineering](#)
[Importance of feature variable](#)
[Feature engineering in machine learning](#)
[Feature engineering techniques](#)
[Imputation](#)
[Handling outliers](#)
[Binning](#)
[Log Transform](#)
[One-hot encoding](#)
[Grouping operations](#)
[Categorical column grouping](#)
[Numerical column grouping](#)
[Feature split](#)
[Scaling](#)
[Extracting date](#)
[Applying feature engineering](#)
[Conclusion](#)

7. Machine Learning Algorithms

[Structure](#)
[Objectives](#)
[Introduction to machine learning](#)
[Brief history of machine learning](#)
[Classification of machine learning algorithms](#)
[Top 10 algorithms of machine learning explained](#)
[Building a machine learning model](#)
[Conclusion](#)

8. Productionizing Machine Learning Models

[Structure](#)

[Objectives](#)

[Types of ML production system](#)

[Batch prediction](#)

[Batch learning](#)

[REST APIs](#)

[Online learning](#)

[Introduction to REST APIs](#)

[Application Programming Interface \(APIs\)](#)

[Hyper Text Transfer Protocol \(HTTP\)](#)

[Client-server architecture](#)

[Resource](#)

[Flask framework](#)

[Simple flask application](#)

[Salary prediction model](#)

[ML model user interface](#)

[HTML template](#)

[Conclusion](#)

9. Data Flows in Enterprises

[Structure](#)

[Objectives](#)

[Introducing data pipeline](#)

[Designing data pipeline](#)

[ETL vs. ELT](#)

[Scheduling jobs](#)

[Messaging queue](#)

[Passing arguments to data pipeline](#)

[Conclusion](#)

10. Introduction to Databases

[Structure](#)

[Objectives](#)

[Modern databases and terminology](#)

[Relational database or SQL database](#)

[Install PostgreSQL and pgAdmin](#)

[Set-up a database and table](#)

[Connect Python to Postgres](#)

[Modify data pipeline to store in Postgres](#)

[Document-oriented database or No-SQL](#)

[Install MongoDB and compass client](#)

[Create a database and collection](#)

[Connect Python to MongoDB](#)

[Modify data pipeline to store in MongoDB](#)

[Graph databases](#)

[Install and start Neo4j](#)

[Add nodes and relations](#)

[Filesystem as storage](#)

[What is Filesystem?](#)

[Filesystem as data store](#)

[Hierarchy to store CSV](#)

[Conclusion](#)

11. Introduction to Big Data

[Structure](#)

[Objectives](#)

[Introducing Big Data](#)

[Definition of Big Data](#)

[Introducing Hadoop](#)

[Hadoop Distributed File System \(HDFS\)](#)

[MapReduce](#)

[YARN](#)

[Hadoop common](#)

[Setting-up a Hadoop Cluster](#)

[Installing a Hadoop Cluster](#)

[Starting Hadoop cluster in Docker](#)

[Word-count MapReduce Program](#)

[Map program](#)

[Reducer program](#)

[MapReduce JAR](#)

[Running Word Count in HDFS Cluster](#)

[Conclusion](#)

12. DevOps for Data Science

Structure

Objectives

Introduction to DevOps

Agile methodology, CI/CD, and DevOps

DevOps for data science

Source Code Management

Quality Assurance

Model objects and security

Production deployment

Communication and collaboration

Conclusion

13. Introduction to Cloud Computing

Structure

Objectives

Introducing cloud computing

Operating system model

What is virtualization?

What is cloud computing?

Types of cloud services

Infrastructure as a Service (IaaS)

Platform as a Service (PaaS)

Software as a Service (SaaS)

Types of cloud infrastructure

Public cloud

Private cloud

Hybrid cloud

Data science and cloud computing

Data

Compute

Integration

Deployment

Market growth of cloud

Conclusion

14. Deploy Model to Cloud

Structure

[Objectives](#)

[Register for GCP free account](#)

[GCP console](#)

[Create VM and its properties](#)

[Connecting and uploading code to VM](#)

[Executing Python model on cloud](#)

[Access the model via browser](#)

[Scaling the resources in Cloud](#)

[Conclusion](#)

15. Introduction to Business Intelligence

[Structure](#)

[Objectives](#)

[What is business intelligence?](#)

[Business intelligence analysis](#)

[Business intelligence process](#)

[Step 1: Data awareness](#)

[Data types](#)

[Data sources](#)

[Step 2: Store data](#)

[Data models](#)

[Data storage](#)

[Step 3: Business needs](#)

[Key Performance Indicators \(KPI\)](#)

[Data Visuals](#)

[Step 4: a Visualization tool](#)

[Time to insight](#)

[Ease of use](#)

[Step 5: Enable platform](#)

[Data access](#)

[Business users](#)

[Business intelligence trends](#)

[Gartner 2019 Magic Quadrant](#)

[Conclusion](#)

16. Data Visualization Tools

[Structure](#)

[Objectives](#)

[Introduction to data visualization](#)

[*Data visualization types*](#)

[Data visualization tools](#)

[*Visualization tool features*](#)

[Introduction to Microsoft Power BI](#)

[*Use case Microsoft Power BI*](#)

[*Microsoft Power BI console*](#)

[*Load the data*](#)

[*Create data visuals*](#)

[*Publish the visuals*](#)

[Conclusion](#)

17. Industry Use Case 1 - Form Assist

[Structure](#)

[Objective](#)

[Abstract](#)

[Introduction](#)

[Related Work](#)

[Proposed work](#)

[*Work architecture*](#)

[*NIST dataset*](#)

[*Activation function – ReLU*](#)

[*Dropout*](#)

[Data augmentation](#)

[Optimization](#)

[Feature extraction](#)

[Image thresholding](#)

[Classifier](#)

[Results](#)

[Conclusion](#)

[Acknowledgment](#)

[References](#)

18. Industry Use Case 2 - People Reporter

[Structure](#)

[Objective](#)

[Abstract](#)
[Introduction](#)
[Event detection](#)
[Work architecture](#)
[Results](#)
[Nipah virus outbreak in Kerala](#)
[CSK enters the final of IPL 2018:](#)
[OnePlus 6 launched in India](#)
[Conclusion](#)
[Acknowledgment](#)
[References](#)

19. Data Science Learning Resources

[Structure](#)
[Objective](#)
[Books](#)
[Online courses](#)
[Competitions](#)
[Blogs and magazines](#)
[University courses](#)
[Conferences and events](#)
[Meet-ups and interest groups](#)
[YouTube channels and Podcasts](#)
[Analytic reports and white paper](#)
[Talk to people](#)
[Conclusion](#)

20. Do It Your Self Challenges

[Structure](#)
[Objectives](#)
[DIY challenge 1 – Analyzing the pathological slide for blood analysis](#)
[Challenge overview](#)
[Challenge statement](#)
[Target users](#)
[Resources](#)
[IP source](#)
[DIY challenge 2 – IoT based weather monitoring system](#)

[Challenge overview](#)

[Challenge statement](#)

[Target Users](#)

[Resources](#)

[IP source](#)

[DIY challenge 3 – Facial image-based BMI calculator](#)

[Challenge overview](#)

[Challenge statement](#)

[Target users](#)

[Resources](#)

[IP source](#)

[DIY challenge 4 – Chatbot assistant for Tourism in North East](#)

[Challenge overview](#)

[Challenge statement](#)

[Target users](#)

[Resources](#)

[IP source](#)

[DIY challenge 5 – Assaying and grading of fruits for e-procurement](#)

[Challenge overview](#)

[Challenge statement](#)

[Target users](#)

[Resources](#)

[IP source](#)

[Conclusion](#)

[21. Qs for DS Assessment](#)

[Structure](#)

[Objectives](#)

[Data Science Overview](#)

[Mathematics Essentials](#)

[Statistics Essentials](#)

[Exploratory Data Analysis](#)

[Data Preprocessing](#)

[Feature Engineering](#)

[Machine Learning Algorithms](#)

[Productionizing Machine Learning Models](#)

[Data Flows in Enterprises](#)

[Introduction to Databases](#)
[Introduction to Big Data](#)
[DevOps for Data Science](#)
[Introduction to Cloud Computing](#)
[Deploy Model to Cloud](#)
[Introduction to Business Intelligence](#)
[Data Visualization Tools](#)
[Conclusion](#)

CHAPTER 1

Data Science Overview

In ancient times land was the most important asset in the world. In the modern era, machines and factories became more important than land. In the twenty-first century, however, data will eclipse both land and machinery as the most important asset.

- Yuval Noah Harari

Data is undoubtedly the most valuable asset of our society. It captures our entire understanding of mankind, space, nature, and learnings from all our existence. In modern days, we have developed tools and methods to understand that data for path-breaking discoveries in medicine, space, machine, and so on. Now, in the 21st century, we are ready to use the technology and data to catapult mankind into the data age. It is not an exaggeration in any sense, just compare how our life is different from our grandparents' life, and you would see how technology and data have changed the way we live and work.

This book titled *Introduction to Data Science for Business - A Practical Guide for Freshers* is an attempt to cover the practical scope of data science teams in real industry set-up. The book will touch upon major areas of work, how you lead there, their significance, and some examples to start hands-on training. The book will take you to a journey and significance of the multidisciplinary nature of data science.

In this chapter, we will have a preview of the books and their content to provide an overview of data science. The chapter will build the use case of what a newcomer to data science or fresher need to be aware of before starting the data science journey.

Structure

- Evolution of data analytics
- Define data science
- Domain knowledge
- Mathematical and scientific techniques
- Tools and technology
- Data science analysis types
- Data science job roles
- ML model development process
- Data visualizations
- Result communication
- Responsible and ethical AI
- Career in data science
- Summary

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of data science and the importance of domain knowledge.
- Identify the mathematical techniques and technology required to build any data science application.
- Recognize the various opportunities around the data science application development process.
- Understand the importance of data visualization and how it can be used in result communication.

Evolution of data analytics

Data has been collected and analyzed for very old times. The information capture and dissemination have been part of managing kingdoms, having records of lands and army. The modern use of statistics started in the 18th century with systematic ways of capturing data, and the evolution of printing the system helped in storing data.

In the context of the book, we would see the journey of data analysis in the following stages:

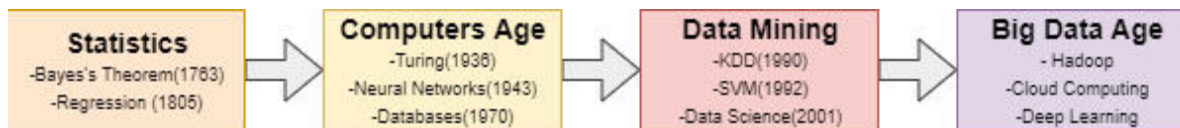


Figure 1.1: Evolution of Data Analysis

Statistics have been around for a long time sparingly across the world. The systematic development of statistics happened in the 18th century and was used in administrative purposes across the world. The great advancements in science during and post-industrial era set the foundation for the computer age. Starting with Alan Turing groundbreaking work in the Theory of computing and advancement in semiconductors, the computers started getting more powerful year on year.

During the mid-19th century, a lot of research work gone into understanding how a human brain learns and advancements in the understanding structure of the brain. Early papers with Neural Networks emerged. The methodology to learn and repeat some events was entirely different from how a distribution based statistical methods explained. Databases also started getting into exclusive use by the 1980s. Digitization also started getting well recognized in the industry and government. Same time, early experiments with networking, emails, and interconnected web were emerging.

During the late 1990s, the computers were household things in the US with Microsoft having released a power operating system MS Windows 98, macOS was in the market as well. Same time enterprising computing was on rising with giants like IBM becoming the core providers of powerful servers and the internet getting its pace. This time Data mining become prominent for creating reports, analyzing customer data, and making decisions driven by basic data analysis (MS Excel was in Windows by that time). Knowledge databases, **Support Vector Machine (SVM)**, SQL databases, and increased computing power market the early stages of Data Science before it exploded around 2008.

Hadoop, 2006, the distributed file storage and computation was a project started by visionary computer scientist, *Doug Cutting*, as he saw a huge wave of Big Data coming. By March 2009, Amazon had already started providing MapReduce hosting service, Elastic MapReduce. This started the

era of Big Data; at the same time, GPU and cloud computing made the cost of computing cheap, leading to rapid development in deep learning and cloud infrastructure.

Today, we have a better understanding of how data science creates value for companies by making them data-driven. All the ecosystem and pre-requisite to make the value of data are available now at a reasonable cost.

The data science as a discipline has grown now and had a universal appeal across the organizations and its benefits. The technology giants are shaping and directing the industry towards data-driven organizations. In [Figure 1.2](#), you can see some of the biggest companies of today are based on the latest technology and data-driven decision making:

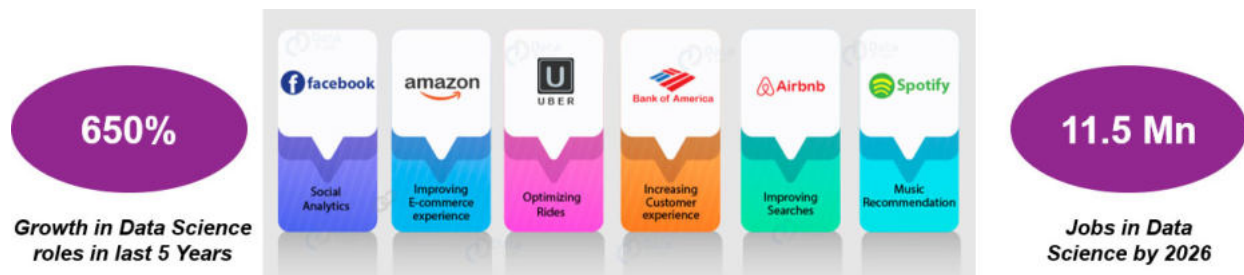


Figure 1.2: Data Science Growth

Some of the popular use cases in the industry are listed below for a reference. The reader must try to understand more about these use-cases and discover how data science adding direct value to the business:

- Fraud and risk detection
- Healthcare
- Internet search
- Targeted advertising
- Website recommendations
- Advanced image recognition
- Speech recognition
- Airline route planning
- Gaming
- Augmented reality
- Talent acquisition

- Credit scoring
- Price forecasting and many more.

The businesses are rapidly adopting new technologies and using AI/ML for competitive and comparative advantage. In the coming years, the adoption of data-driven features will be faster in governance and general public discourse.

Define data science

Data Science is the umbrella term that comprises the science and its application related to data. The early definition of data science was built on a combination of multiple disciplines, and moreover, it is expanding fields as data keeps adding value to multiple disciplines.

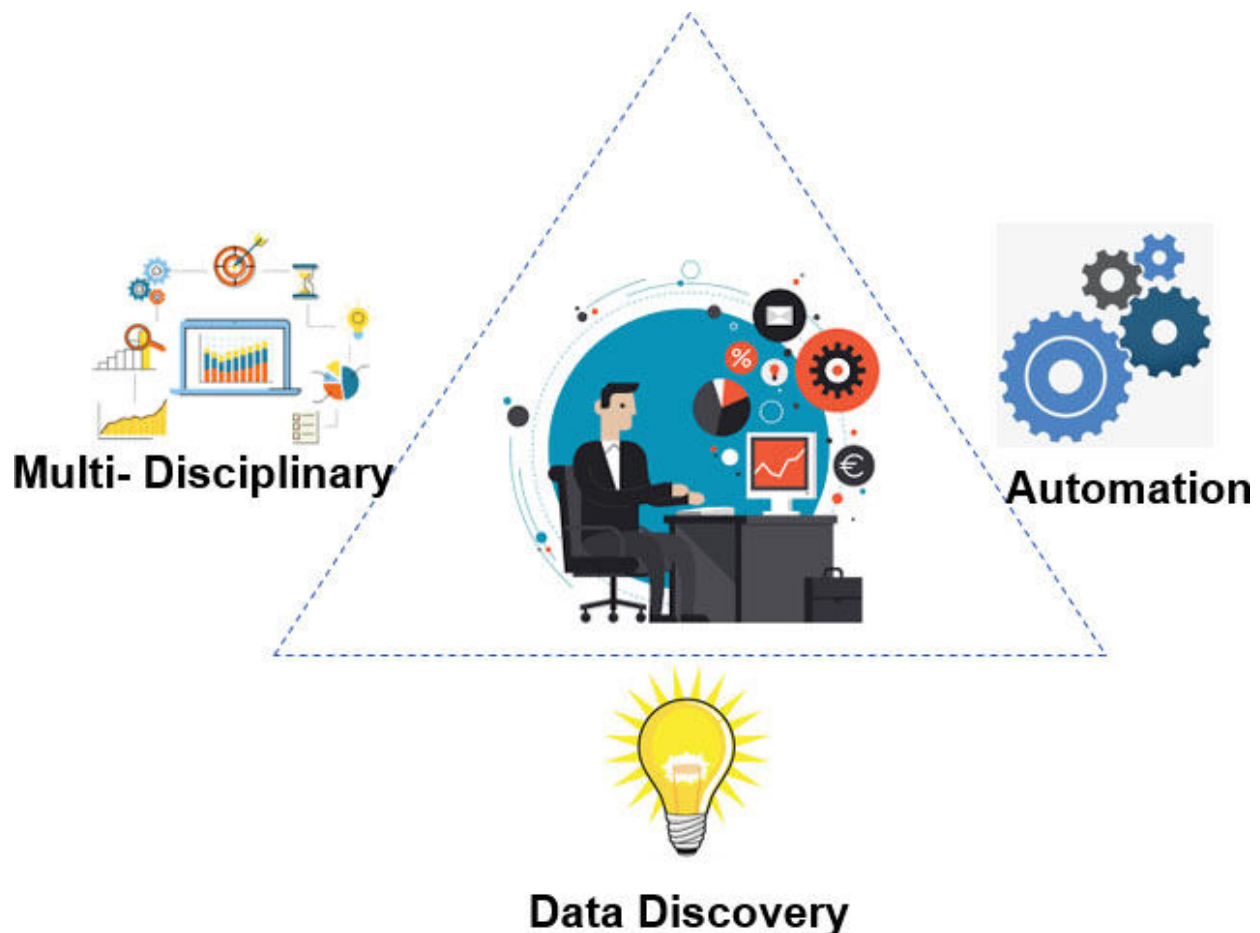


Figure 1.3: Define Data Science

The key features of these fields are as follows (as shown in [Figure 1.3](#)):

- **Multi-disciplinary:** A new discipline that combines aspects of mathematics, statistics, programming, and visualization.
- **Automation:** An automated way to analyze the enormous amount of data and extract information
- **Data discovery:** A powerful new way to make discoveries from data.

Further, data science as a discipline defines some foundational knowledge that the practitioner needs to have to be able to harness value from data. [Figure 1.4](#) is the Venn diagram which is a representation of such a practitioner:

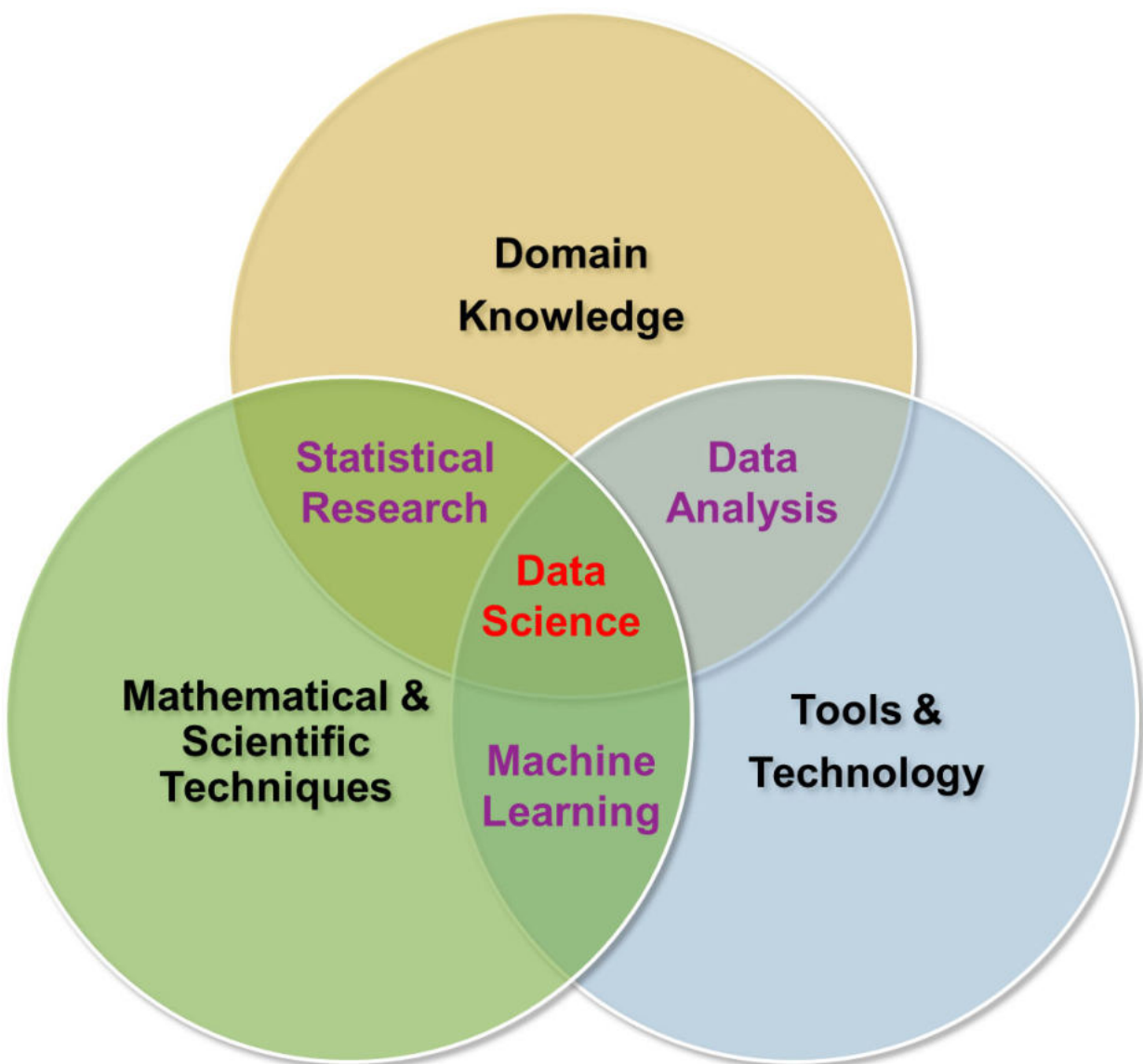


Figure 1.4: Data Science Practitioner

The core areas bring relevant expertise to help build data solutions;

- Domain knowledge
 - Engineering
 - Science
 - Business
 - Medicine
 - Economics
 - Finance
- Mathematical and scientific techniques
 - Linear algebra
 - Classic statistical tools like regression
 - Clustering and classification
 - Machine learning
- Tools and Technology
 - Programming
 - Operating systems
 - Analysis tools (R, SAS, Python)
 - Visualization tools (Tableau)

Note: The above list is just indicative of some very popular sub-areas for data scientists.

The different subareas within data science are not confined to traditional knowledge but also include the powerful method of experimentations and learning; the automated learning from machines is what we popularly call as *Machine Learning*. The tools and technology do the analysis at scale and provide the medium to deliver results to end-users.

[Domain knowledge](#)

Domain knowledge means the understanding of the domain from where the data problem/opportunity is originating. This is important as the data is a representation of a process or phenomenon captured by data. The data

scientist is responsible for discovering the relationships in data in the context of that domain.

For illustration, assume we are working on a problem statement for a bank.

The problem: The bank gets of loan applications from potential borrowers. The loan officer must make a decision to issue a loan or not.

Now there are so many domain questions to be understood before starting the analysis:

- What are the terms and conditions to issue a loan?
- How different factors affect borrowing power?
- What are the chances of full recovery of loans?
- What data about the borrower can we capture as per statutory and regulations?
- What are market conditions now? How they define the relationship with loan recovery?

And so many other aspects of the banking process and its implications on the loan approval process. Economics and bank policies need to be part of the analysis. If we do not have this background, we would just crunch the data without any context and applicability of results.

Mathematical and scientific techniques

Domain knowledge will provide the context of the data and the desired results from the data science process. Mathematical and scientific techniques provide a theoretical understanding of how to quantify the behavior the business wants to investigate the data.

For Illustration, in the previous example, we had generic domain questions. For instance,

Question: What are the chances of full recovery of a loan?

The domain expert would look at application details and, based on experience, can say it's *High*. But how HIGH? He would not be able to quantify until he has some statistical technique to define and calculate HIGH. This is the area where techniques help the domain understanding gets a quantifiable number to them, so that decision boundaries can be created.

Machine learning algorithms are the techniques of learning the relationships among datasets automatically and build functional relationships for future predictions. [Figure 1.5](#) shows how machine learning differs from the standard computer application:

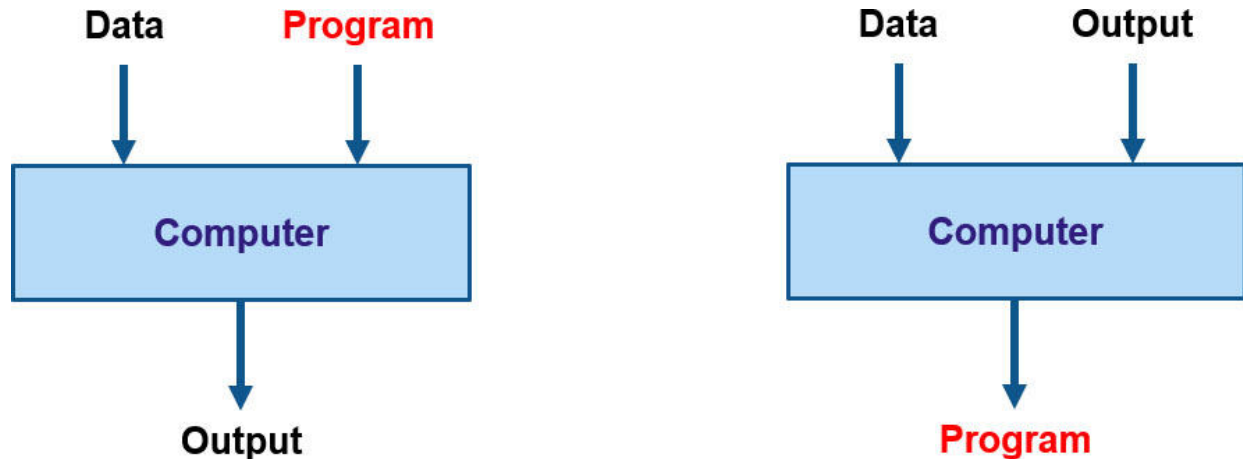


Figure 1.5: Traditional Rule-Based Vs. Machine Learning

The expert systems are driven by intuition and experience of experts. In those cases, the computer is fed with input data, and a program (collection of rules/logic) and an output is generated. Referring to our previous section loan example, the loan officer will set a bar that I will only issue a loan of less than \$25,000. Then the input becomes the loan application amount. The program becomes a rule that if more than \$25,000, then reject or accept.

Now, this sounds a good way to start deciding the loan applications by making decisions with numbers. However, what we are missing here is the lack of evidence that \$25,000 and more loan recovery are very bad. How does the loan office assume this number/logic? May be more than \$25,000 loans are very profitable for the bank as they repay.

Hence, we switch to the second approach, which does not assume anything beforehand. It takes inputs (loan amount), the output (If repaid or not), and then creates a program to identify with high accuracy which bank of the loan amount is riskier and by how much. This way, the machine learning models generate the program learning from data, and then this program can be used in the traditional system to generate output for new entities.

Machine learning can be of any of three types as below:

- **Supervised learning:** We have enough historical data of input and output pairs to learn the relationship.

- **Unsupervised learning:** We do not have an output to optimize the behavior relationship but discover the internal structure of the data.
- **Re-enforcement learning:** We keep learning from feedback from the output. This allows us to have continuous learning.

[Figure 1.6](#) depicts the type of machine learning algorithms, and each of these classes of algorithms has various types of algorithms, used for various purposes to solve different types of problems:

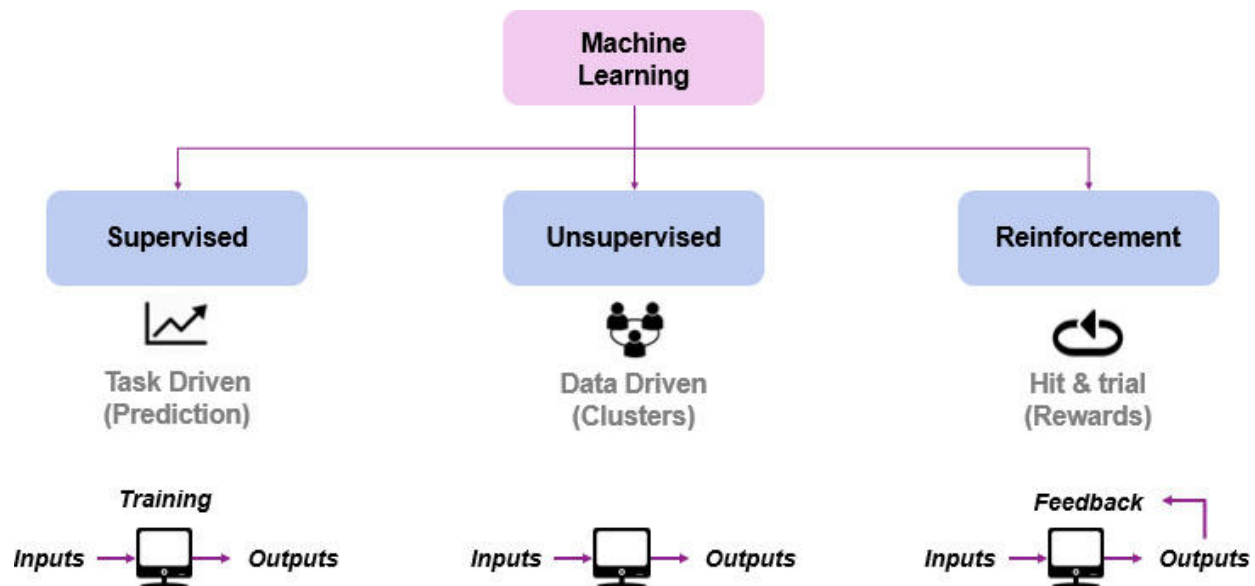


Figure 1.6: Types of Machine Learning

The *Machine Learning* section of the book will discuss aspects of machine learning in detail. Below table enumerates some popular algorithm of each type:

Algorithm category	Examples
Supervised algorithms	Naive Bayes Decision Trees Linear Regression Support Vector Machines (SVM) Neural Networks
Unsupervised algorithms	k-means clustering Association Rules Self-Organizing Maps Principle Component Analysis
Reinforcement learning	Q-Learning Temporal Difference (TD)

Table 1.1: Machine learning algorithms with category and example

All the above algorithms are now standardized for off-the-shelf use in different programming languages. R language has been the most popular among statisticians and researchers to code their algorithms, and Python is catching up fast.

To be a prudent data scientist, one has to learn the underlying concepts from mathematics as well to understand what these algorithms are actually doing. Many data science professionals ignore the solid foundation in mathematics and statistics and remain reliant on the accuracy of packages developed by others to run an algorithm. Though open-source implementations are very stable and work in most cases, however lack of knowledge of algorithms working leads to sub-optimal or wrong decision son part of models.

The *Mathematics and statistics* section of the book will have a primer to most important topics that are encountered very frequently by data science professionals in designing and implementing algorithms for business problems. The most basic concepts include:

- Linear algebra (Matrix Computations)
- Optimization
- Multivariate calculus
- Probability
- Hypothesis testing
- Distributions

And many more sub-topics. A well-read data scientist never simply trusts a package and blindly use the output of libraries. It is important to critically analyze the algorithms and has a sharp eye for any limitations or deviations from the theory of the algorithms.

Tools and technology

After acquiring the knowledge of the domain and having methods to quantify behaviors, data science requires means to implement those solutions at scale. Technology is the enabler to implement the mathematical and scientific knowledge for end-use. Internet, cloud, and programming are the

key to creating solutions for end-user. Tools are the means to accomplish a task; the knowledge to implement is the same implemented in different ways by different tools.

For example, you may want to have an addition done in C or Python; the way you do with these two programming languages may be different by the result will always be the same. Hence, it's important for data science professionals to be updated with the latest tools so that they always have the most efficient and economical way to produce results.

[Figure 1.7](#) shows the KDnuggets analytics/data science 2018 poll results for the tools the community is using to develop data solutions. Python stands out as one of the most favored tools in the community:

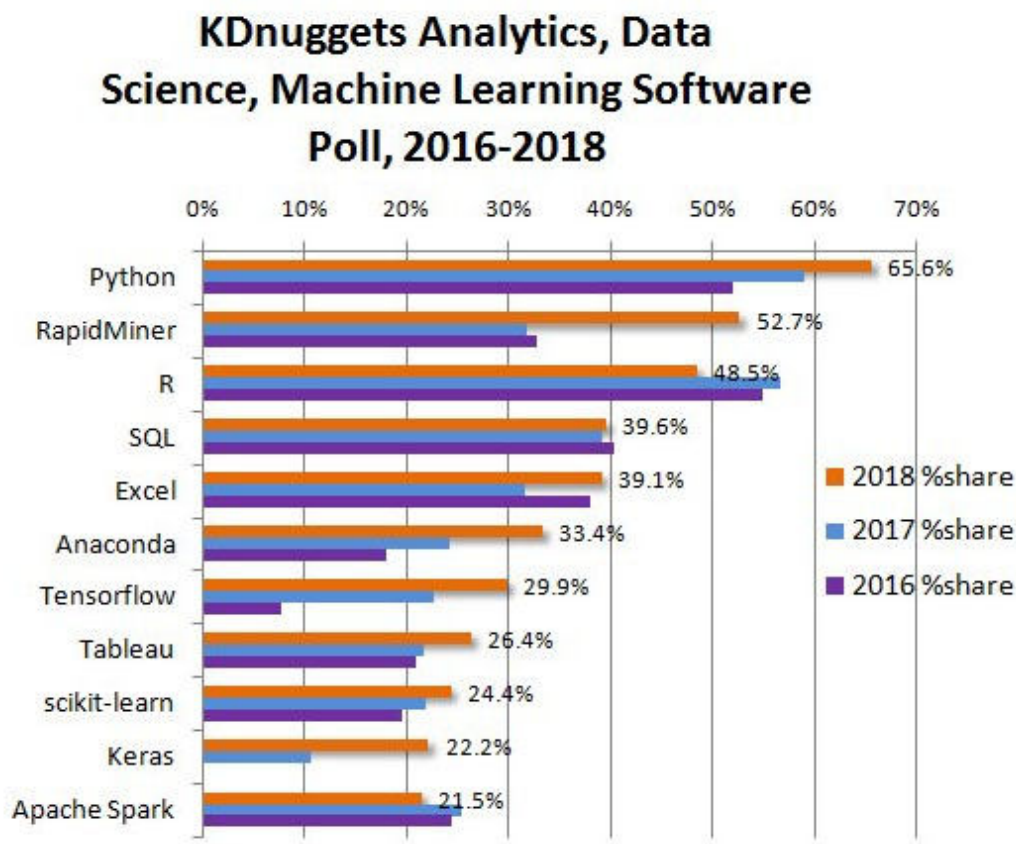


Figure 1.7: KDnuggets Analytics/Data Science 2018 poll results (Credits: <https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html>)

The open-source nature of tools has fuelled the development of programming languages and packaged resources. Proprietary tools like

Tableau and Excel are also part of the list as the enterprise uses them extensively.

Cloud computing is another set of technology which has changed the way data science is adopted, implemented, and maintained by big corporates. The whole cloud technology has made computing cheaper, on-demand, and very powerful. Now, even a small start-up can afford multi-million infrastructures by paying per hour as per need. This has changed the whole ecosystem of data-driven products. Below you see some companies which are just built on data-centric platforms and are now the biggest success stories in the corporate world:



Figure 1.8: Data Success Stories

All the above companies and solutions are hosted by public cloud vendors like Amazon Web services, Google Cloud Platform, and Azure. These platforms provide massive storage capacity and GPU computations for complex AI/ML models. Not only this, the cloud has given rise to three new types of technology service models:

- **Software as a Service (SaaS)**
- **Platform as a Service (PaaS)**
- **Infrastructure as a Service (IaaS)**

In SaaS service, a fully hosted business application is provided on subscription bases, not only that reduced cost of the company, due to consolidation nature allows SaaS provider profits. The Silicon Valley start-up ecosystem is a true reflection of how the SaaS model allowed some of the biggest companies of our time to grow from start-ups to multi-billion-dollar companies.

Data Engineering is the new role that has emerged around extensive use of cloud technologies to build data pipelines having AI/ML results to be delivered to clients and businesses. The role of a data engineer is then to make sure that the required data by algorithms and end-user is delivered on time and with integrity. The *Data engineering* section will talk in detail about *data engineering*/pipelines, and the *Cloud computing* section of the book will talk in detail the technicalities and its benefits for data science growth.

Data science analysis types

Data Science analysis types can be divided into 4 types and can be seen as how the new tools are impacting the data analysis in more advanced methodologies and tools. [Figure 1.9](#) shows the types of analysis and how they differ from each other. All these types are now grown up as a separate area within the organization and also looking for having their own well-defined job roles:

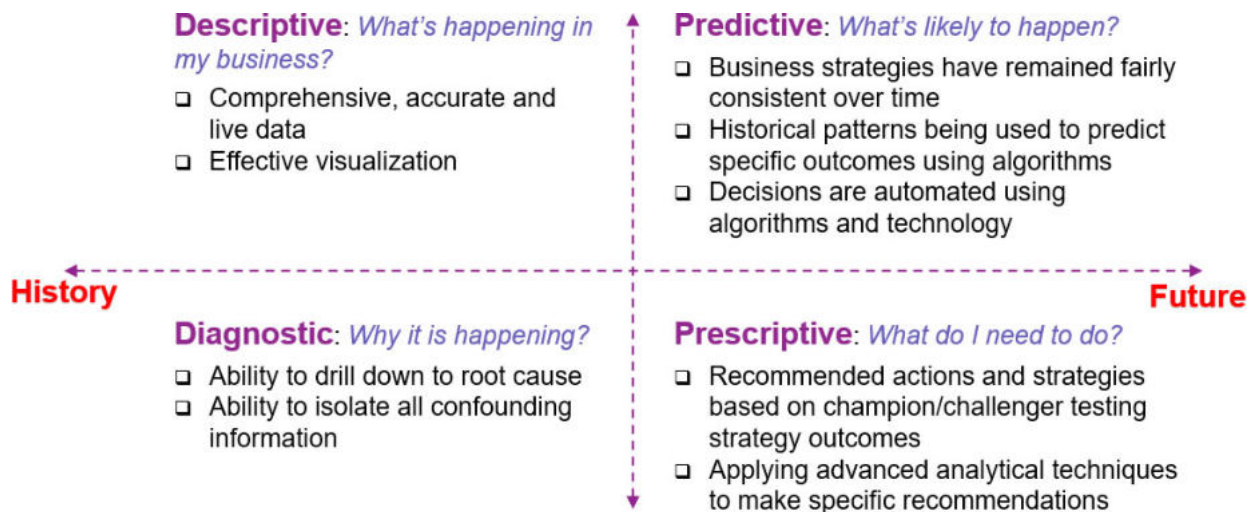


Figure 1.9: Types of data science analysis

The descriptive analysis has been historically well studied and applied in statistical methods. It reflects the empirical measurement of what is happening in the system/business under observation. The next step to that is diagnosing the happening by investigating the data by different slices and time window views to essentially answer why it happened.

For example, my business grew by 45% in last quarter is a descriptive measure, if we also add why it grew it become diagnostics as well, that is, our new product segment grew by 90%, balancing the slowdown in an old product by 45%. Now you can observe we know what happened and why it happened. Both these analyses are very business-centric and allow the business to make quick and accurate decisions.

The modern, powerful computing environment enabled predictive and prescriptive data analysis as well. The predictive analysis helps find relationships among data points and help predict one data point if another is available, while prescriptive go one step further and recommend which variable or data point to control to get the desired results.

For example, the sales will grow by 25% next quarters as the festival season is going to start next month. This is a predictive outcome of analysis, as historical data would suggest the sales go up in the festive season. Further, the prescriptive analysis outcome will be like if we give a 10% discount, the sales will go up by 35% in the upcoming festive season. Here, we can control the discount rate, and the analysis prescribes it to be 10% for optimal gains.

Data science job roles

In the previous session, you observed the types of analysis and skillsets that have emerged to define job roles that scope the work and bring synergies and in-depth analysis capabilities in professionals. If you refer to [Figure 1.4](#), which describes the three key subject areas comprising data science, you would be able to relate the job roles with the data science function. The three areas of a domain, statistical techniques, and technology give rise to three roles data analyst, data scientist, and data engineer, respectively. The roles are described in [Figure 1.10](#), with their high-level responsibilities in the role:

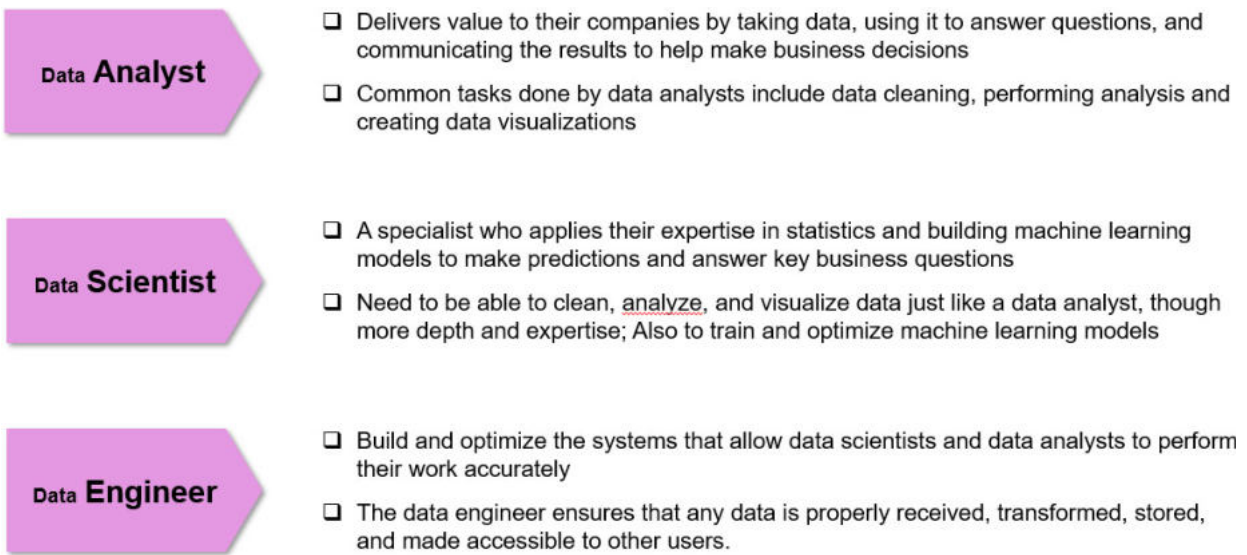


Figure 1.10: Data science job roles

The job roles are essential to understand as it's not possible to keep juggling in roles in the early stage of career. All the roles are interconnected to each other; however, as a new entrant, you need to have adequate skills in one area as a major, and you can pick another area as minor. Having the knowledge, all job roles within the data science function brings high synergy in work.

ML model development process

Data science, as perceived by most of the online courses and recent public discourse, has been around how to develop accurate models for prediction. The key area within data science is focused on the development of models, that is, artificial intelligence, machine learning, and deep learning. The areas are a complete subset of the prior to them one respectively:

- **Artificial intelligence:** Programs that have the ability to learn and reason like humans.
- **Machine learning:** Programs that have the ability to learn without being explicitly programmed.
- **Deep learning:** Programs that can learn from a vast amount of complex data using artificial neural networks.

For fresher's who are starting their journey into data science, the first they need to learn is the process of developing a machine learning model and

interpret them. [Figure 1.11](#) shows the process which has been studied and provided an indicative guideline for developing models:



Figure 1.11: Data science modeling process

The steps are arising in detail in the next section of chapters on machine learning. Here we provide a basic definition of each step:

1. **Problem definition:** Any data analysis starts with setting up an objective that we want to achieve out of model development exercise. These objectives can be in terms of hypothesis or target results in business metrics after using the models.
2. **Data collection:** Now, the data which can help solve the problem statement is gathered through different channels and sources. Best efforts are made to have accurate and timely data for the analysis.
3. **Data wrangling:** Data wrangling has many parts to it, including cleaning the data from missing and erroneous values, removing outliers, transforming the data, feature engineering, and other steps to make data ready for empirical analysis.
4. **Exploratory Data Analysis (EDA):** EDA step is a pre-model-analysis of descriptive and diagnostic nature where we use visualizations, distributions, frequency tables, and other techniques to understand the data relationships and make the choice of right algorithm for desired analysis.

5. **Machine Learning Algorithms:** Now, we train, test, and validate algorithms with appropriate dependent and independent variables, with appropriate techniques from the set of supervised, unsupervised, and reinforcement learning algorithms.
6. **Prediction and insights:** The algorithms will quantify all the relationships and allow us to make predictions and derive insights from the model outputs. The model results need to be transformed back to business language and presented on the same scale as of original data.
7. **Visualization and communication:** The results need to communicate back to business executives or end-user for making decisions. The results need to be communicated in simple terms while not undermining the assumptions of probability and modeling techniques.

The process is what has been observed in most of the cases, but it does not limit the data science professionals to explore new ways to bring value out of their data. Exploration and being curious is the key to develop good models that derive business.

Data visualizations

Data visualization is not an integral part of organizations and end-user applications. In today's applications, you would always start your application experience with a Dashboard and then go further into the application features. Data visualization tools, like Power BI and Tableau, have grown to the extent that they provide a self-service platform to build visualizations to communicate data analysis for both business intelligence purpose and exploratory data analysis from machine learning. [*Figure 1.12*](#) shows the cognitive side of visualizations and how they are beneficial to users to interpret complex data and make business decisions:



Figure 1.12: Data visualizations

In the section, Business intelligence, we talked in detail about how to build visualizations and how the tools play an important role in speed-up your analysis and communication of results to stakeholders.

Result communication

Result communication is both art and science. With years of experience, you develop the right balance between the two to become an efficient communicator of data science. You can assume this role be like that of a translator of language, here the data is to be communicated to business and vice versa. The choice of right words, right visualizations, and interpretations are important for communicating the right understanding of the data. [Figure 1.13](#) shows two examples of effective communication and a summary of analysis which could have taken weeks and multiple tools to come to the conclusions:

	Example# 1	Example# 2
Finding	<ul style="list-style-type: none"> Male shoppers contribute to higher chunk of sales over the weekend compared to females 	<ul style="list-style-type: none"> Customers are not watching the entire video to its full length. They are watching 90–95%
Insight	<ul style="list-style-type: none"> Male shoppers tend to be free from work during weekends and hence majority of their shopping is done during the same time 	<ul style="list-style-type: none"> The parts they are not watching are the title roll and the end credits
Conclusion	<ul style="list-style-type: none"> Targeting female shoppers with weekday coupons so as to balance out average daily revenue OR Targeting male shoppers with weekend coupons so as to maximize their market basket size 	<ul style="list-style-type: none"> Introduce 'Skip Intro' at the beginning of title rolls and 'Watch Next' at the beginning of end credits. Benchmark 90–95% watched content as completed and measure if customers move to the next video in the series

Figure 1.13: Result communication

It is important for data science professionals to be aware of how business communications happen and how they can be more effective to communicate their findings, and also convert their pain points into data problems. This role is usually done by experienced professionals, and junior resources need to learn from them.

Responsible and ethical AI

As data science and AI become more day to day affairs and start impacting how we operate our daily life and business, it becomes important to get aware of the negative aspects that AI will bring to the fore. The data science professionals need to be aware of the consequences of their work and the impact it can have on society. It is important to have the highest ethical practices for data science professionals. [Figure 1.14](#), from ngu.eu, shows the key areas and questions pertaining to the responsible and ethical use of AI:



Figure 1.14: Responsible and Ethical AI (Credits: ngi.eu)

The leading name in technology and consulting, along with governments, are leading the efforts to make AI use fair and ethical. It will have regulatory effects, like the **General Data Protection Regulation (GDPR)**, Data Privacy Framework, and other self-governing structures of organizations. As a professional in the domain, one needs to pay adequate attention to their work and being fair as the models developed by them may deny a loan to a needy just because you ignored a bias in historical data for class, color, or religion. It is our duty to NOT bring our biases to systems and make it a system for the future.

Career in data science

Towards the end of the overview section, Probyto, with its years of experience, would like to give some advice to budding data scientists for a successful career. The key to success in any field is asking questions and remain a learner forever. Data science is no different. Having a strong foundation in mathematics is very helpful in a long career in data science; as technology has masked a lot of hard facts behind easy to use menu driven approached, you still need to have adequate knowledge of Mathematics. [Figure 1.15](#) shows the various skills required for getting into data science career:

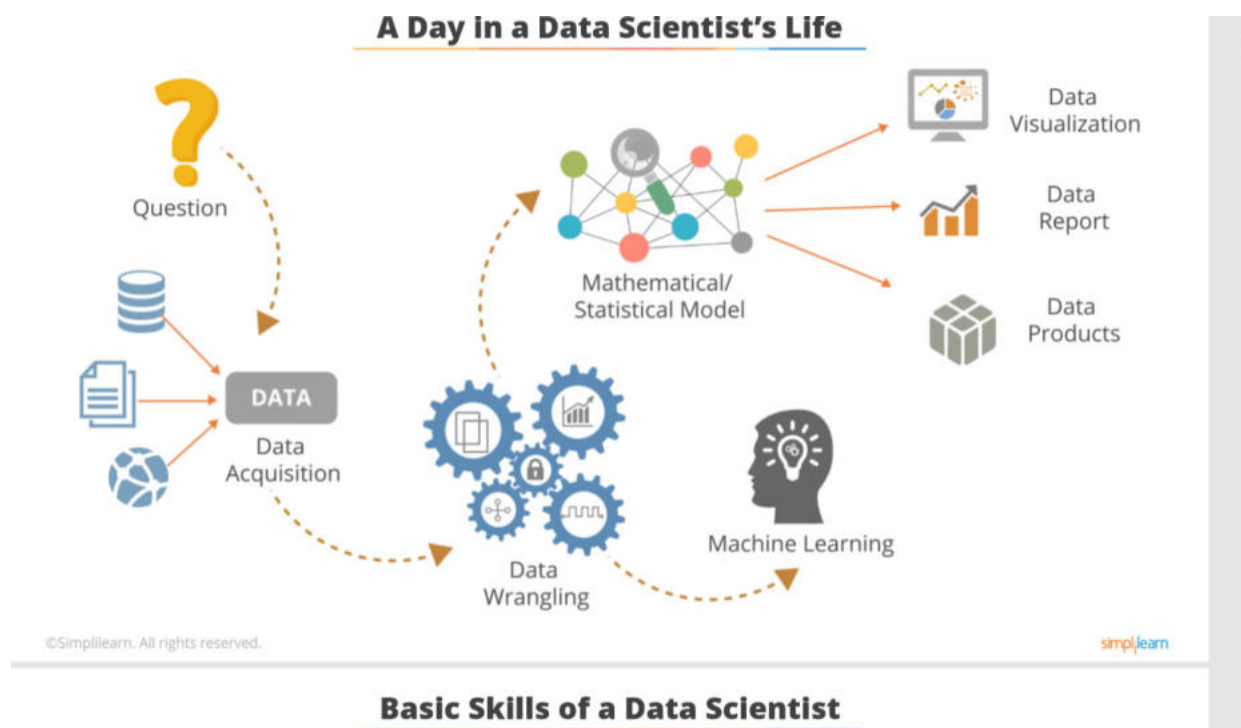


Figure 1.15: Key Skills Required for Data Science Career

Some of the skills you are required to possess for a job in data science are, not exhaustive;

1. Asking the right questions
2. Understanding data structures
3. Data exploration and interpretation
4. Applying models
5. Visualization of data

In [Chapter 19: Data Science Learning Resources](#), we also provide you a small list of some popular resources that help you be updated with data science all the latest happenings. A career in data science is very dynamic by ever-growing technology, research in algorithms, and new problems to solve. While its exciting career but also very challenging and require you to be up to date always.

Conclusion

The chapter sets the primer for the data science beginners by introducing multiple ideas, origin, process, and definitions in the data science domain. We talked about how the landscape of data analysis has been changing and evolved from past to today, what triggered those changes, and the development of data science. Then we defined data science as a Venn diagram having the right balance of domain knowledge, mathematics, and technology, to enable bringing value from data. The different types of analysis are then discussed to allow the reader to differentiate between multiple types of approaches to data as per need. We have then talked about different role types in data science and how they differ from each other. The ML model building process is also introduced for readers to have a preview of the next sections on machine learning. Data visualization is introduced as key to allowing the user to interpret data faster by a visual view rather than raw data. Result communication is an important part of becoming a translator between data and business problems. Responsible and Ethical AI is the need of the time with so much invasion of AI in our daily lives, and technology companies and government actively working to contain that. The chapter ended with some suggestions on the key skills and requirements to start a career in data science.

By reading this chapter, the reader has previewed the book and its content on an overview of data science. He or she will understand what a beginner needs to acquire before stepping into the data science journey.

In the next chapter, we will discuss the mathematics concepts needed in modern data science. This will be a strong foundation to apply data science in real-world problems.

CHAPTER 2

Mathematics Essentials

Mathematics is the foundation of any modern-day discipline of science. When we look into the modern data science principles, there must be some deep mathematical behind it. Understanding and learning the fundamentals of mathematics is very important for any data scientist or junior analyst. Because they have to apply those techniques in solving problems. Mathematics foundation works like the heart of the problem-solving using data science. Other items will come in the category of just by using an API or using the new algorithms.

In order to make a meaningful prediction and recommendation to the users, we have to use algorithms very carefully. To develop a better algorithm, we must have a strong understanding of the mathematical principles behind the algorithms. When we have the mathematical foundation very solid, then it creates more confidence for peers to work on it.

In this chapter, we are going to learn the very important mathematical concepts, which will be the strong foundation to apply data science in real-world problems.

Structure

- Introduction to linear algebra
- Scalars, vectors, matrices, and tensors
- Eigenvalues and eigenvectors
- Eigen decomposition and singular value decomposition
- The determinant
- Principal component analysis
- Introduction to the multivariate calculus
- Differential and integral calculus
- Partial derivatives

- The Gradient, Hessian, Jacobian, Laplacian, and Lagrangian Distribution
- The Gradient Descent algorithm
- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of linear algebra and data representations.
- Apply the basic properties of matrix and vectors in data science applications.
- Derive the mathematical concepts behind data science problems.
- Build the mathematical model of an algorithm.

Introduction to linear algebra

Linear algebra is the branch of mathematics to study lines and planes, vector spaces and mappings that are required for linear transforms. Linear algebra can be called basic mathematics to understand the data, which has the intention of finding related values using linear combinations. In other words, it is the application of the problem-solving system of linear equations to find unknown or new findings from data.

Vectors and matrices are the basic notions of data. When data represented using a column-based notion, we call that as vectors, if the array is used to represent the data that has been considered as matrix format.

Example:

1. $\mathbf{V} = \begin{bmatrix} 3 \\ -1 \\ 9 \end{bmatrix}$ where, V is the vector which has data in column-based representation (that is, single column with three data).
2. $\mathbf{M} = \begin{bmatrix} 2 & -1 & 4 \\ 5 & -7 & 3 \\ -9 & 8 & -6 \end{bmatrix}$ where M is the matrix data in array-based representation (that is, rows and columns are used to represent nine data).

Even though vectors and matrices are the languages to represent data, we have to understand some of the basic data representation with geometry space. The following section will briefly introduce those concepts.

Scalar, vectors, matrices, and tensors

Let us start exploring the fundamentals of data representation from mathematics perspective.

Scalar

A scalar is a one-dimensional representation of data. It contains only the magnitude in the form of numerical value. For example, consider the scenario to travel from one place to another place; the following are the data represented in the scalar values:

- Distance between the starting place and destination place (that is, 100 km)
- Speed of the vehicle (that is, 20 km/h)
- Traveling fair (that is, Rs 20)
- Weight of the person (that is, 60 kgs)

This example gives a brief idea about the scalar representation of data. When we represent data in higher-dimensional geometry, space will move with further additional information.

Vectors

A vector is a notion of one or more values of scalars. Already we have introduced the vector we can now explore the characteristics of vectors and how it is used to represent the data in geometry space. From the scenario of travel, we can take the example of vector data:

- Acceleration is given to the direction.
- The velocity of the vehicle traveling towards a destination.
- Displacement, i.e., traveling from one point to another point.

Characteristics of the vector:

- The vector contains magnitude and direction.
- Vector changes if either the magnitude or direction changes or both change.

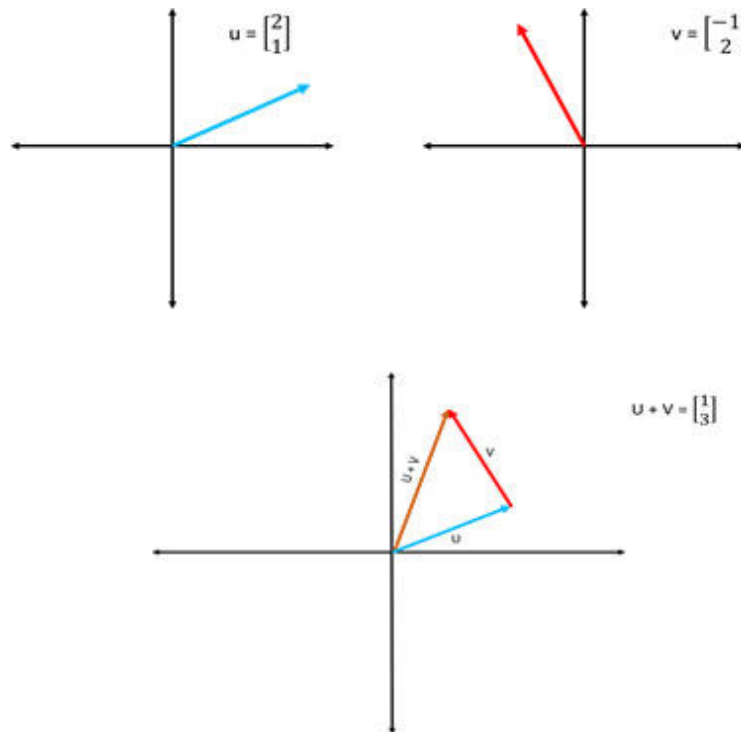


Figure 2.1: Vector Addition

Vector plays the main role in the linear algebra because both with vectors, it is in two operations. Vector addition and scaling the vector by multiplying some scalar value. To represent those operations in geometrically refer to [Figure 2.1](#) and [Figure 2.2](#):

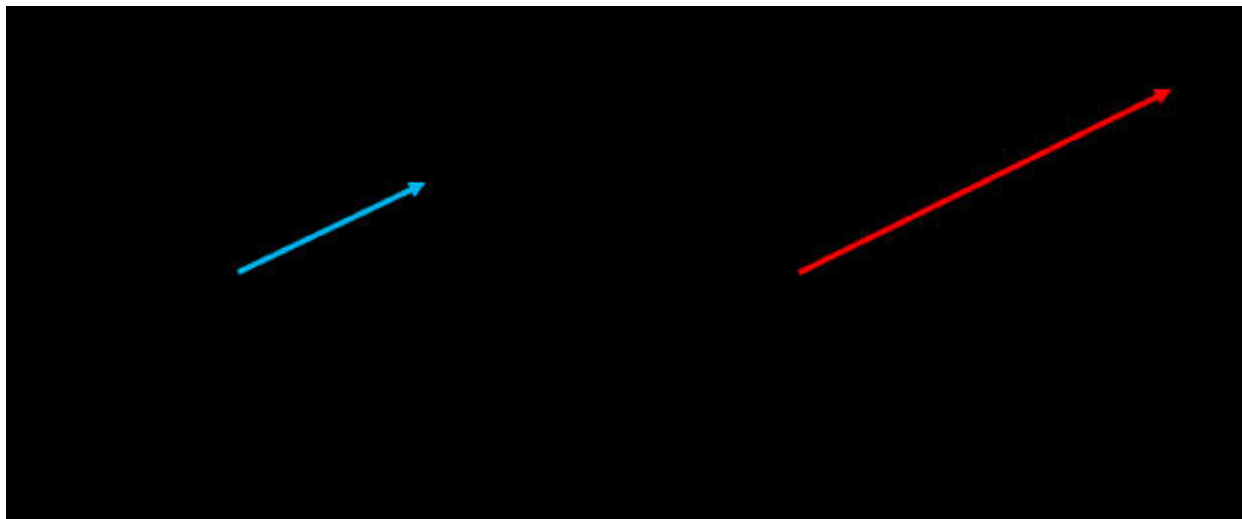


Figure 2.2: Scalar multiplication to Vector

Understanding the capabilities of vector operations provides the confidence in estimating the scalability of values and make use of them in data representation is an easier process.

Matrices

Data are represented in a rectangular arrangement is known as matrices. Where data is arranged in the form of rows and columns. For example, to represent the data with 2 rows and 3 columns, we will use the below notation:

$$A = \begin{bmatrix} 6 & -1 & 2 \\ 5 & -7 & 3 \end{bmatrix}$$

The representation of rows and columns is also known as the dimension of the matrix. In other words, we can say that is the size of the matrix. So, the dimension of the above matrix is 2 (*rows*) \times 3 (*columns*).

As matrices can hold the higher dimensional space in geometry, we can interpret that it is possible to hold data which comes in the form of multi-dimensional format.

A data in a matrix entry is simply a matrix element. Each element of data in a matrix is recognized by naming the row and column in which it appears.

For example, let us consider the above matrix A. The A₂₃ is the entry of scalar data value 3 is identified by calling the second row and third column. By using this representation, we can easily access the data from the matrix. It also provides the flexibility to pick the data directly without going into the sequential way.

In general, the matrix representation is in the following format:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & \dots & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & \dots & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & \dots & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & \dots & \dots & a_{mn} \end{bmatrix}$$

To access the element, we have to use the representation of a_{mn} , where m represents the row value, and n represents the column value. Matrices

mainly used to solve systems of equations. But first, we must learn how to represent linear systems with matrices.

A system of equations can be solved easily by representing its data in an augmented matrix. An augmented matrix in linear algebra is a matrix generated by adding columns of two different matrices, typically for the purposes of carrying out the same simple row operations on each of those matrices.

For example, consider the linear equation system:

$$x+2y+3z=5$$

$$2x+3y+z=7$$

$$x+y+z=6$$

We can represent the coefficient matrices of the system as:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 5 \\ 7 \\ 6 \end{bmatrix}$$

Then the augmented matrix is,

$$A \vee B = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & 3 & 1 & 7 \\ 1 & 1 & 1 & 6 \end{bmatrix}$$

The augment matrix will help to represent the data that can be solved using a combination of operation performed on matrices rows to get unknown values of coefficients in the linear equation representation.

Even though we can apply all the basic arithmetic operations on matrices, addition and multiplication play vital roles. Let us start the discussion on those operations.

The Matrix addition is quite easy and is performed input-wise. For example:

$$A = \begin{bmatrix} 2 & -1 & 4 \\ 5 & -7 & 3 \\ -9 & 8 & -6 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 7 & 9 \\ 3 & -6 & 3 \\ 4 & 2 & -6 \end{bmatrix} \text{ then,}$$

$$A + B = \begin{bmatrix} 2 & 6 & 13 \\ 8 & -13 & 6 \\ 5 & 10 & -12 \end{bmatrix}$$

Here, the elements from matrix A and B are mapped together, and simple addition performed to get the new value (that is, $2 + 0 = 2$). The generated new matrix follows the same rules for all the entries.

In matrix multiplication operation, we must follow the different scenarios to do the multiplication of scalar values. When we perform the operation, we must check the dimensionality of both matrices.

If we want to perform multiplication between two matrices, we have to check that the number of columns in a matrix is equal to the number of rows in the next matrix. If it is not equal, then we can't perform the multiplication.

For example, consider the matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 0 & 2 \end{bmatrix} \text{ and } B = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix} \text{ then } A.B = \begin{bmatrix} 8 & 7 \\ 20 & 15 \\ 4 & 6 \end{bmatrix}$$

Here, in matrix A , a number of columns are 2, and in matrix B number of rows is 2. Hence, we can compute the solution of AB like this.

$$A.B = \begin{bmatrix} 1*4 + 2*2 & 1*1 + 2*3 \\ 3*4 + 4*2 & 3*1 + 4*3 \\ 0*4 + 2*2 & 0*1 + 2*3 \end{bmatrix} = \begin{bmatrix} 8 & 7 \\ 20 & 15 \\ 4 & 6 \end{bmatrix}$$

If the dimension of matrices goes more than 2, then it is difficult to apply the multiplication easily for that we have to use another format to represent the data. So, handling higher dimensional data is done by tensors.

Tensors

The tensor is mathematically an N -dimensional vector, which means that an N -dimensional data can be represented by tensor. It is a multidimensional array.

Mathematically a tensor is an N -dimensional vector, which means a tensor can be used to represent N -dimensional data. A tensor is a simplification of vectors and matrices, and it can easily understand as a multidimensional array:

tensor

't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)

2	7	8	8	1	8		
2	8	4	5	9	0	4	5
2	3	5	3	6	0	2	8
7	4	7	1	3	5	2	6

tensor of dimensions [4,4,2]

Figure 2.3: Simplified tensor with minimum dimensions

We can say a vector is a 1D or 1st order tensor, and a matrix is a two-dimensional or second-order tensor. Tensors have the ability to perform all types of operations with scalars, matrices, or vectors by reformulation.

The determinant

Determinants are mathematical entities that are widely used in the analysis and find the solution of systems that have the property of linear equations. The determinant value of a matrix is a special value, and it is very important that it can be calculated from a square matrix. We will discuss the calculation of determinant of a matrix with an example.

Let $A = \begin{bmatrix} 6 & 1 \\ 4 & 3 \end{bmatrix}$ be the square matrix with size 2×2 then, determinant of matrix A is represented using the notion $|A|$. The value of the determinant is calculated using the below method:

$$|A| = 6 \cdot 3 - 4 \cdot 1 = 14$$

Now, we can generalize this calculation. Let us assume, then determinant

value, $|M|=ad-bc$. Finding the determinant value for 3×3 matrix is a similar way only. For example:

$$G = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 4 & 2 & 1 \end{bmatrix}$$

$$|G| = 1(3 \cdot 1 - 2 \cdot 5) - 2(2 \cdot 1 - 4 \cdot 5) + 3(2 \cdot 2 - 4 \cdot 3)$$

$$1(3 - 10) - 2(2 - 20) + 3(4 - 12)$$

$$1(-7) - 2(-18) + 3(-8)$$

$$-7 + 36 - 24$$

$$5$$

Therefore, the determinant is $|G|=5$. In general, we can represent the calculation using the symbol:

$$\text{If, } M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \text{ then}$$

$$|M| = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$a(e \cdot i - f \cdot h) - b(d \cdot i - f \cdot g) + c(d \cdot h - e \cdot g)$$

$$aei + bfg + cdh - afh - bdi - ceg$$

From the calculation of determined of the matrix, we can tell whether that matrix can be inverted or not. It also consider the scalar property of a matrix.

Eigenvalues and Eigenvectors

Eigenvectors and eigenvalues are used to decrease noise in data. This impacts efficiency in the computation of the tasks. So, to improve efficiency in computationally more complex tasks, estimation of Eigenvalues and Eigenvectors plays a vital role. We can represent multidimensional data in a matrix. One eigenvalue and eigenvector are useful to capture significant information that is stored in a large matrix. Performing computations on a large matrix is a very time-consuming process. One of the key methodologies to enhance the efficiency in computationally demanding tasks is to reduce the dimensions of data after ensuring most of the important information is maintained.

Before mathematically explaining the eigenvalues, we first see the details about eigenvectors. When we apply the scalar multiplication on a vector almost, it will change the direction. In opposite to this, any vector which is not changing its direction even it is multiplied by a scalar value is known as eigenvector. Multiply an eigenvector by A , and the vector Ax is a number λ times to the original value. The basic representation is $Ax = \lambda x$, where λ is an eigenvalue of matrix A . This value only determines whether the x is expanded or shrunk or unchanged when it is multiplied by A . In graphical way it is shown like [Figure 2.4](#):

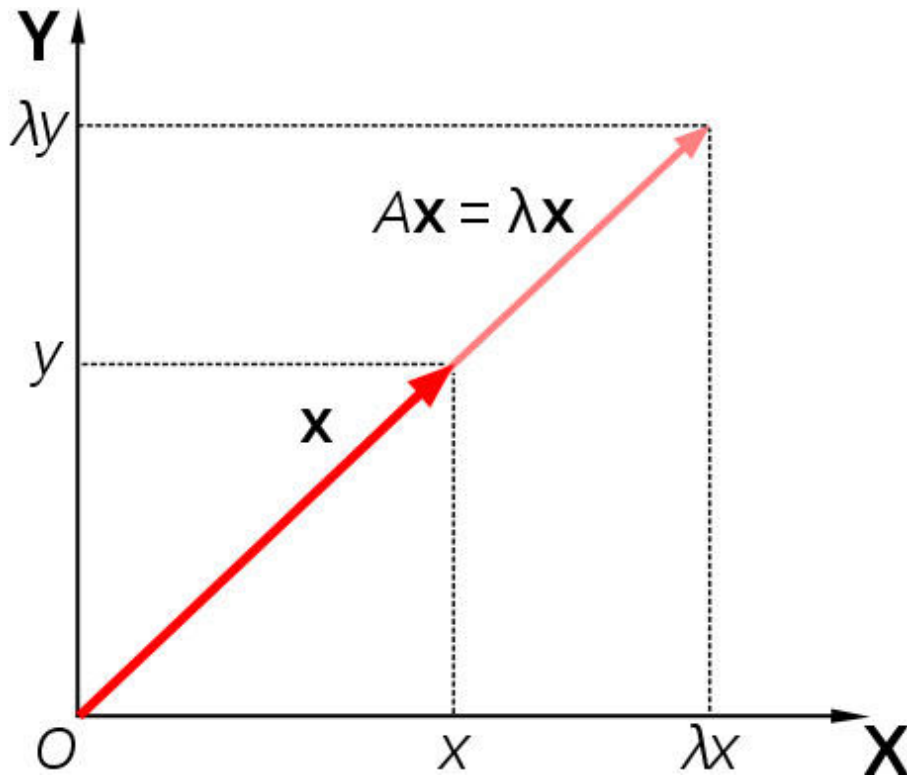


Figure 2.4: Eigenvalue λ and eigenvector A representation

Now consider the linear transformation of N-dimensional vectors defined by an n by n matrix A :

$$Ax = w,$$

$$\text{or } \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Where for each row, $w_i = A_{i1} x_1 + A_{i2} x_2 + \dots + A_{in} x_n = \sum_{j=1}^n A_{ij} x_j$

If it occurs that x and w are scalar multiples, that is if, $Ax = w = \lambda x$. Then v is the value of an eigenvector of the linear transformation of matrix A , and the scale factor λ is the eigenvalue corresponding to that eigenvector. We can rewrite the eigenvalue equation like, $(A - \lambda I) x = 0$. Where I is the n by n identity matrix, and 0 is the zero vector. The eigenvalues of A are values of λ that satisfy the equation, $|A - \lambda I| = 0$.

From this equation, we can get eigenvalue λ of matrix A by taking the roots of the polynomial. For example, consider the matrix, $M = \begin{bmatrix} 2 & 7 \\ -1 & -6 \end{bmatrix}$.

First, multiply λ to an identity matrix and then subtract the two matrices:

$$|M - \lambda I| = \left| \begin{bmatrix} 2 & 7 \\ -1 & -6 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right|$$

$$|M - \lambda I| = \begin{vmatrix} 2 - \lambda & 7 \\ -1 & -6 - \lambda \end{vmatrix} = \lambda^2 + 4\lambda - 5$$

Apply, $|M - \lambda I| = 0$. Then, $\lambda^2 + 4\lambda - 5 = 0$ by solving this equation, we can get λ values. $\lambda = -5$ and $\lambda = 1$. These values are the eigenvalues of the matrix M . To find the eigenvector substitute the values of λ in the equation $(M - \lambda I) x = 0$

Substitute $\lambda = -5$, $\begin{bmatrix} 7 & 7 \\ -1 & -1 \end{bmatrix} x = 0$. By solving this linear equation, we will get the vector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ is one of the eigenvectors for the matrix M . Using, $\lambda = 1$, we will get the vector $\begin{bmatrix} -7 \\ 1 \end{bmatrix}$ is another linearly independent eigenvector for the matrix M .

Eigenvalue decomposition and Singular Value Decomposition (SVD)

In linear algebra, eigenvalue decomposition or sometimes spectral decomposition is the factorization of a matrix into a canonical form, whereby the matrix is represented in terms of its eigenvalues and eigenvectors. Only diagonalizable matrices can be factorized in this way.

Let A be a square $n \times n$ matrix with n linearly independent eigenvectors q_i (where $i = 1, \dots, n$). Then A can be factorized as,

$$A = Q \Lambda Q^{-1}$$

Where Q is the square $n \times n$ matrix whose i th column is the eigenvector q_i of A , and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\Lambda_{ii} = \lambda_i$. Note that only the diagonalizable matrix can be factorized in this way.

The n eigenvectors q_i are usually normalized, but they need not be. A non-normalized set of n eigenvectors, v_i can also be used as the columns of Q . That can be implied by observing that the magnitude of the eigenvectors in Q gets canceled during the decomposition by the existence of Q^{-1} .

The decomposition can be derived from the fundamental property of eigenvectors.

$$Av = \lambda v$$

$$AQ = Q\Lambda$$

$$A = Q \Lambda Q^{-1}$$

For example, consider, $\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix}$ may be decomposed into a diagonal matrix through the multiplication of a non-singular matrix $B = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

$$\text{Then, } \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

For some real diagonal matrix $\begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$ multiplying both sides of the equation on the left by B :

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

The above equation can be decomposed into two simultaneous equations:

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} = x \begin{bmatrix} a \\ c \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} b \\ d \end{bmatrix} = y \begin{bmatrix} b \\ d \end{bmatrix}$$

Letting, $\vec{a} = \begin{bmatrix} a \\ c \end{bmatrix}$, $\vec{b} = \begin{bmatrix} b \\ d \end{bmatrix}$ this gives us two vector equations:

$$A\vec{a} = x\vec{a}$$

$$A\vec{b} = y\vec{b}$$

And can be represented by a single vector equation involving two solutions as eigenvalues:

$$Au = \lambda u$$

Where λ represents the two eigenvalues x and y . From eigenvalue equation we can write:

$$(A - \lambda I) u = 0$$

By solving the equation:

$$(1-\lambda)(3-\lambda)=0$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} = 1 \begin{bmatrix} a \\ c \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} b \\ d \end{bmatrix} = 3 \begin{bmatrix} b \\ d \end{bmatrix}$$

$$a = -2c \text{ and } b = 0$$

Thus, the matrix B required for the Eigen decomposition of A is

$$B = \begin{bmatrix} -2c & 0 \\ c & d \end{bmatrix}$$

That is:

$$\begin{bmatrix} -2c & 0 \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -2c & 0 \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

Singular value decomposition

In short, we can call SVD for Singular-Value Decomposition. The reduction of a matrix into its integral parts to make subsequent matrix is called as

SVD. Simply, we can call this as a matrix decomposition method to split a matrix. It makes the matrix calculation as a simpler process.

For the case of simplicity, we will focus on the SVD for real-valued matrices and ignore the case for complex numbers.

$$A = U.Sigma.V^T$$

Where A is the real $m \times n$ matrix that we wish to decompose, U is an $m \times m$ matrix, Σ (often represented by the uppercase Greek letter Sigma) is an $m \times n$ diagonal matrix, and V^T is the transpose of an $n \times n$ matrix where T is a superscript.

The diagonal values in the Σ matrix are known as the singular values of the original matrix A . The columns of the U matrix are called the left-singular vectors of A , and the columns of V are called the right-singular vectors of A .

The SVD is calculated via iterative numerical methods. We will not go into the details of these methods. Every rectangular matrix has singular value decomposition, although the resulting matrices may contain complex numbers, and the limitations of floating-point arithmetic may cause some matrices to fail to decompose neatly.

The common problem is that the response matrix is singular or close to singular, so it has no well-defined inverse. Of the various algorithms that have been developed to deal with this problem, **singular value decomposition (SVD)** has emerged as the most popular. Any matrix can be represented with SVD as follows:

$$M = \sum_{k=1}^n \vec{u}_k w_k \vec{v}_k^T$$

Where \vec{v}_k is a set of orthonormal steering magnet vectors, \vec{u}_k is a corresponding set of orthonormal BPM vectors, and w_k are the singular values of the matrix M .

Given the SVD of a matrix, the matrix inverse is:

$$M^{-1} = \sum_{k=1}^n \vec{v}_k \frac{1}{w_k} \vec{u}_k^T$$

It follows from the orthonormality of the two vector sets. It is immediately apparent from the singular value decomposition if the response matrix is singular one or more of singular values, w_k , are zero. Physically, a zero w_k implies that there is some combination of steering magnet changes, v_k , which gives no measurable change in orbit. The orbit shift from this v_k is zero at all the BPMs. Removing the terms with zero w_k from the sum in the above equation produces a pseudo inverse for orbit correction, which generates no changes in the steering magnet strengths along the corresponding eigenvectors v_k .

Principal component analysis

Principal component analysis (PCA) is a technique that is useful for the compression and classification of data. The purpose is to reduce the dimensionality of a data set (sample) by finding a new set of variables, smaller than the original set of variables that nonetheless retains most of the sample's information. By information, we mean the variation present in the sample, given by the correlations between the original variables. The new variables, called **principal components (PCs)**, are uncorrelated and are ordered by the fraction of the total information each retains.

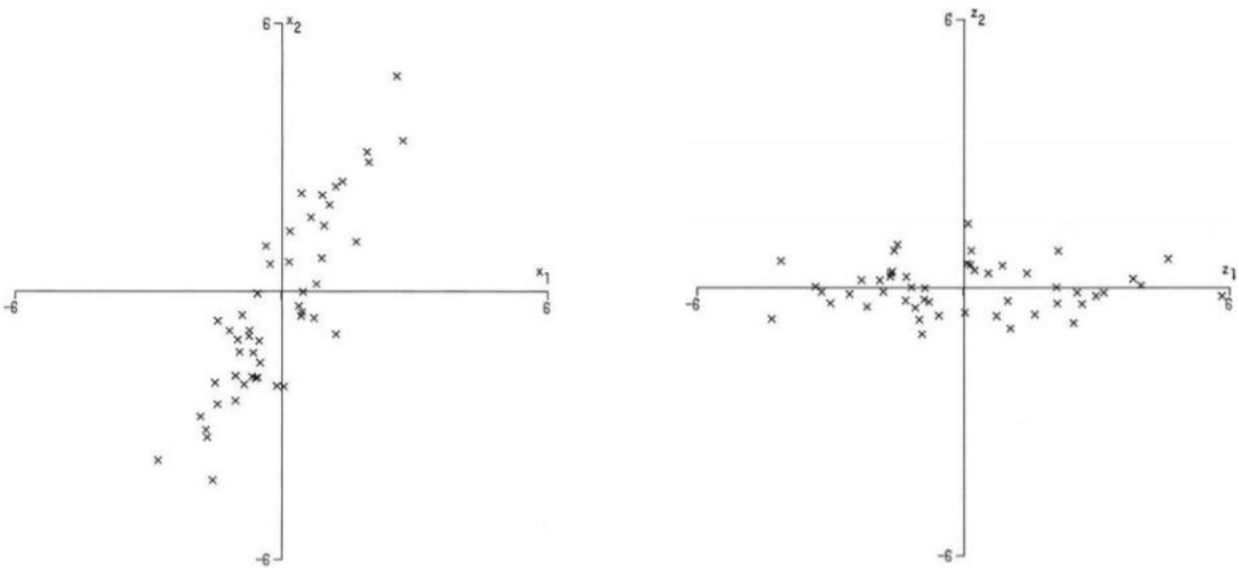


Figure 2.5: Plot of n observations with two variables and a plot of the same wrt their principal axes.

PCA is mathematically defined as an orthogonal linear transformation (meaning it rotates and scales) that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to

lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a sample space of p random variables where n observations are made. $x = (x_1, x_2, x_3, \dots, x_p)$. When we apply a linear transformation to the observations along with the first principal component z_1 we will have the below equation:

$$z_1 = a_1^T x_1 = \sum_{j=1}^p a_{j1} x_j$$

Where the vector $a_1 = (a_{11}, a_{21}, a_{31}, \dots, a_{p1})$

z_1 will be our first principal component only when $\text{var}[z_1]$ is maximum. Likewise, when the k^{th} principal component has to be calculated, the above equation can be transformed as below:

$$z_k = a_k^T x_k = \sum_{j=1}^p a_{jk} x_j$$

Where the vector $a_k = (a_{1k}, a_{2k}, a_{3k}, \dots, a_{pk})$ must be chosen such that $\text{var}[z_k]$ is the maximum subject to $a_1^T a_1 = 1$:

$$\text{var}[z_k] = a_k^T \Sigma a_k$$

Where Σ is the covariance of the variables $(x_1, x_2, x_3, \dots, x_p)$. The problem seems to be a constraint optimization problem. Applying LaGrange multipliers where λ is the LaGrange multiplier.

$$a_k^T \Sigma a_k - \lambda_k$$

Optimization involves differentiating and equating to 0 when we have to find a maximum or minimum of a function. Applying differentiation:

$$\Sigma a_k - \lambda a_k = 0 \text{ can also be written as } \Sigma a_k = \lambda a_k$$

Therefore referring to previous topics of eigenvectors we can say for Σ as a covariance matrix a_k will be the k^{th} eigenvectors and λ_k will be the k^{th} eigenvalue.

Multivariate calculus

Multivariate Calculus (also known as multivariable calculus) is the extension of calculus in one variable to calculus with functions of several variables: the differentiation and integration of functions involving multiple variables, rather than just one:

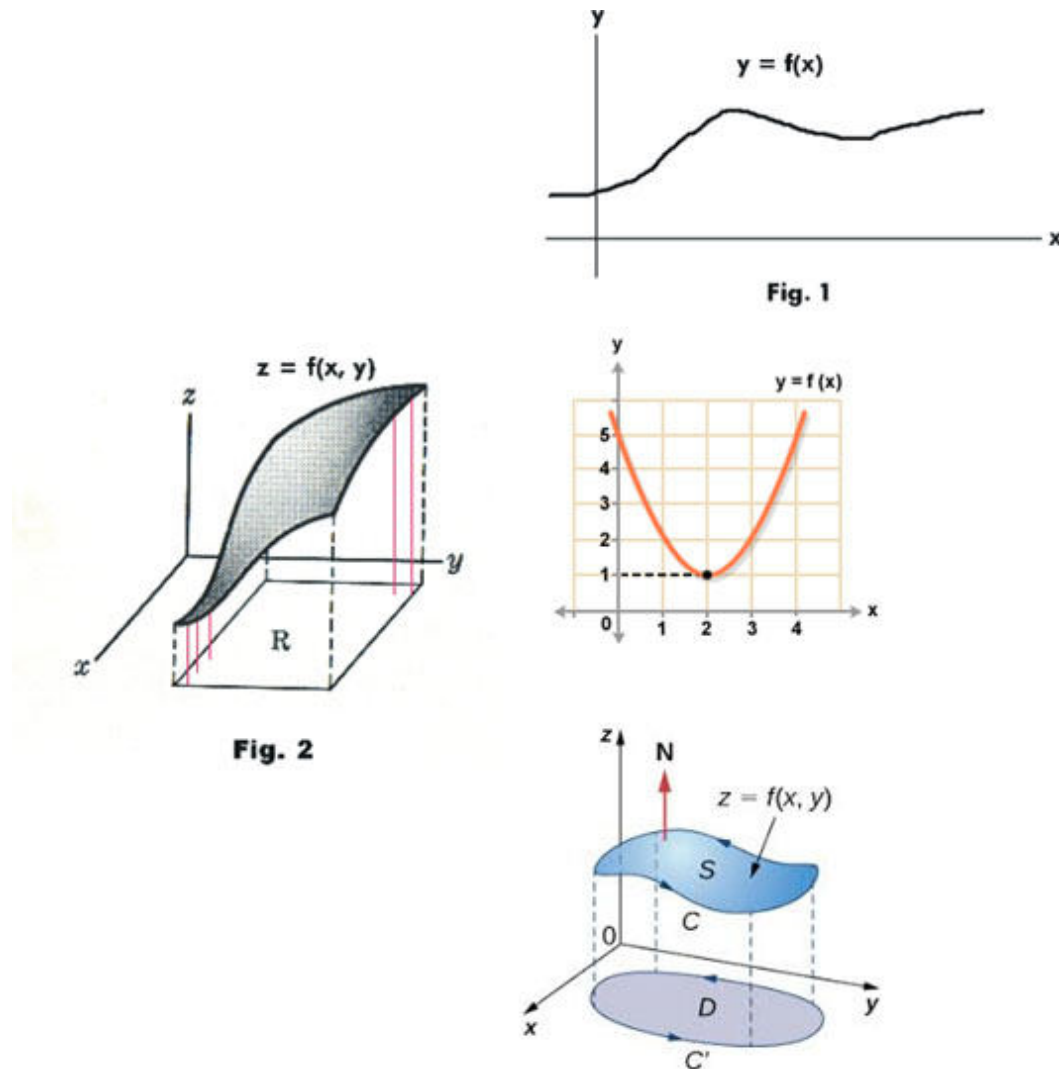


Figure 2.6: Multivariate Calculus (Credit: <https://www.toppr.com/bytes/multivariable-calculus/>)

Calculus is a set of tools for analyzing the relationship between functions and their inputs. In Multivariate Calculus, we can take a function with multiple inputs and determine the influence of each of them separately.

Why is Multivariate Calculus important in data science?

In data science, we try to find the inputs which enable a function to best match the data. The slope or descent describes the rate of change off the output with respect to an input. Determining the influence of each input on the output is also one of the critical tasks. All this requires a solid understanding of Multivariate Calculus.

What is a function?

A function is a connection between input and output. In that, the notation of $f(x)$ is a function of the variable x . This relationship can be seen as the growth over the run of how the change in one variable affects the relationship in another variable.

Differential Calculus

The gradient of a variable in a function is the rise over run against the other variables. [Figure 2.7](#) shows the gradient representation using run and rise in a plot. In the normal x, y coordinated plot, the rise over run is computed using two points plotted on the graph:

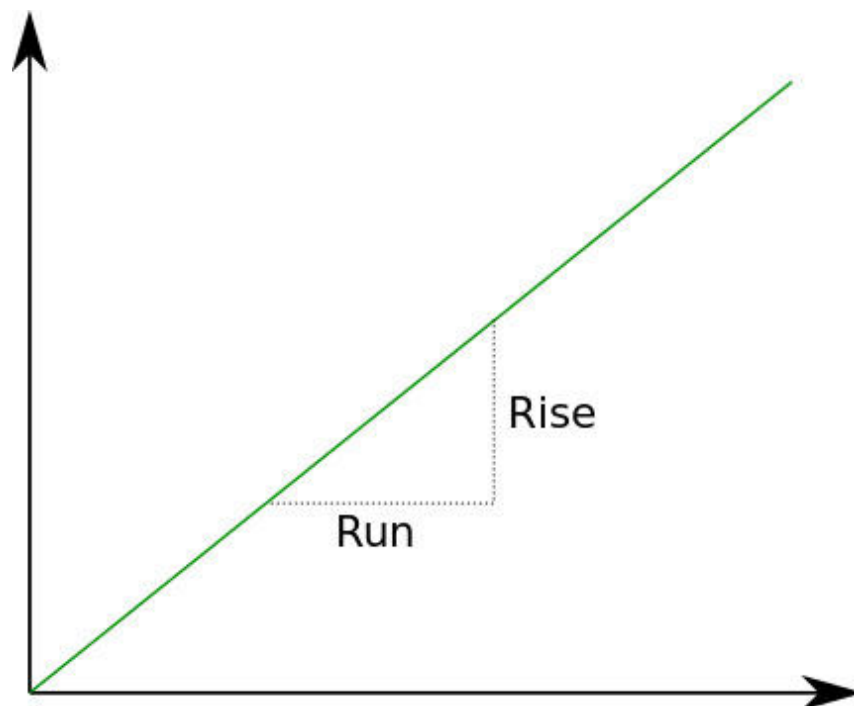


Figure 2.7: Gradient is equal to Rise over Run

The representation for the derivative or the gradient can also be shown as such:

$$\frac{df}{dx} = f'(x) = \lim_{dx \rightarrow 0} \frac{f(x + dx) - f(x)}{dx}$$

If we apply the gradient for a non-linear function, it changes based on the value of the variable change. Considering only the two data points to calculate gradient will produce inaccurate gradient value.

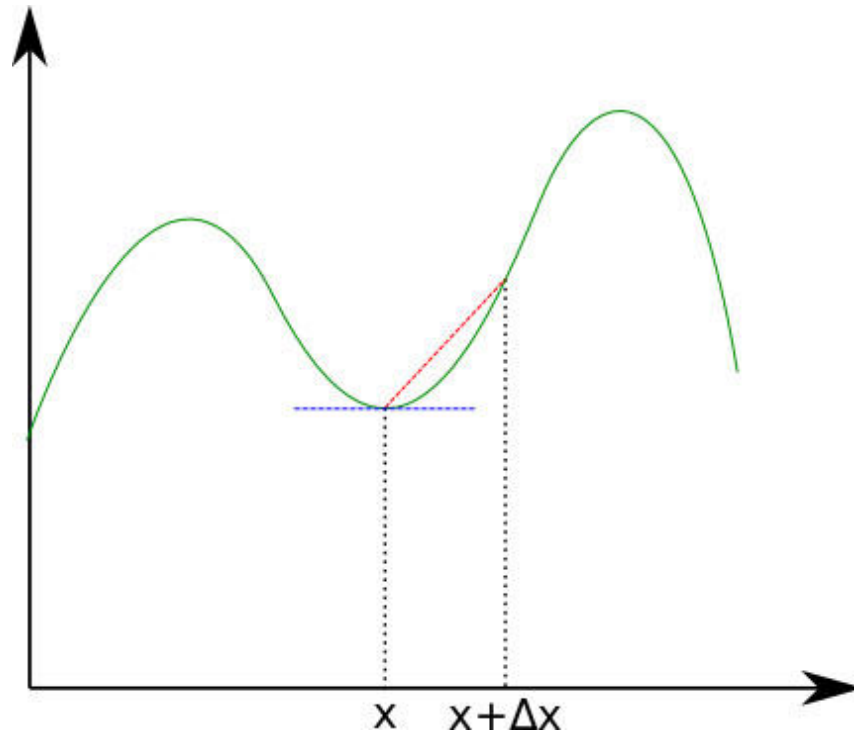


Figure 2.8: As dx tends to 0, the gradient becomes more accurate

Let us take [Figure 2.8](#) as an example. Considering two points x and $x + dx$, we can get an evaluation of the gradient at x . However, if we were to change the value of the second point, where dx tends to 0, the gradient we calculate will be more accurate. Hence, we should aim to get a dx where it is infinitely small yet not 0. Using the notation given above, let's try an example where $f(x)=3x+2$.

$$f'(x) = \lim_{dx \rightarrow 0} \frac{3(x + dx) + 2 - (3x + 2)}{dx}$$

$$\lim_{dx \rightarrow 0} \frac{3x + 3dx + 2 - (3x + 2)}{dx}$$

$$\lim_{dx \rightarrow 0} \frac{3dx}{dx} = 3$$

The above example shows how we can use the gradient calculations between the two points.

Sum rule

The derivatives of the sum of two functions are the sum of the derivative of each function on its own. This rule extends indefinitely.

$$\frac{d}{dx}(f(x) + g(x)) = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$$

Power rule

The derivative of a function with respect to a variable with an exponent is given by taking the value of the exponent and multiplying the function with it. Then deduct one from the value of the exponent.

$$\text{For } f(x) = ax^b$$

$$f'(x) = abx^{(b-1)}$$

The above equations represent how we can use the power rule in representing the function.

Special cases

The $f(x) = 1/x$ function is a special one which includes a discontinuity:

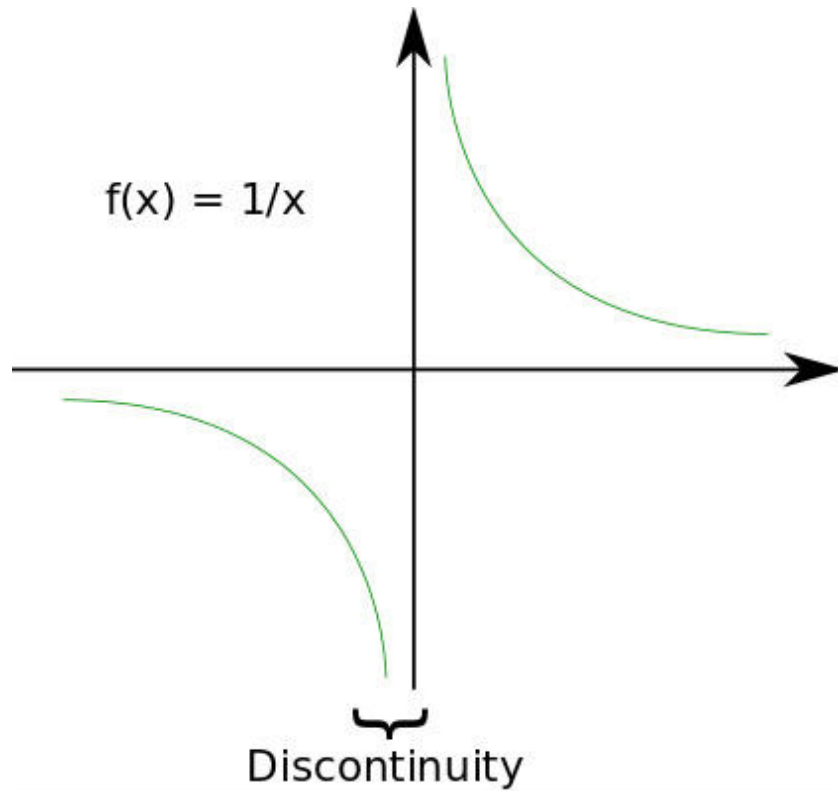


Figure 2.9: Discontinuity in the $f(x) = 1/x$ graph

[Figure 2.10](#) show that this function also has a negative gradient for all values of x , excluding $x = 0$ which is undefined:

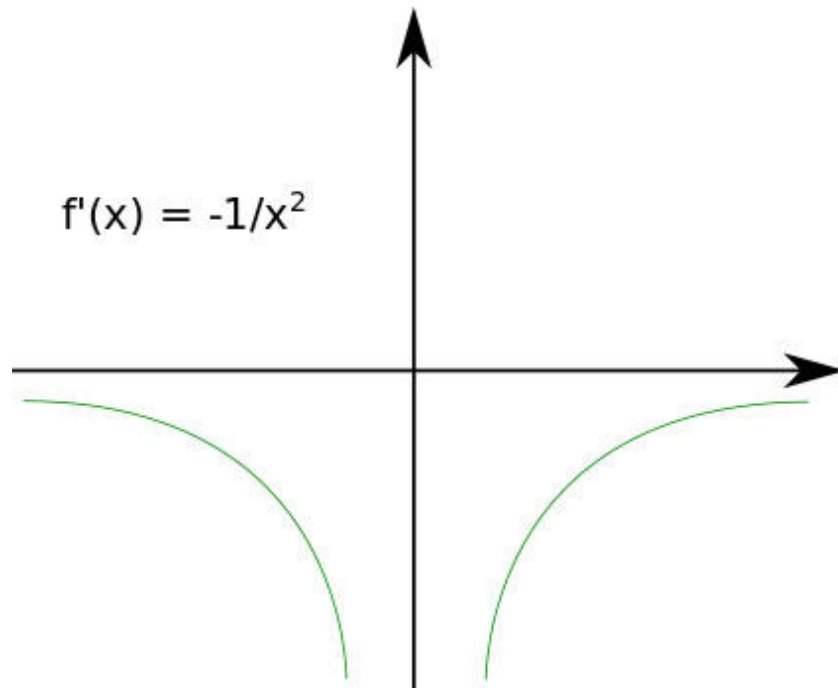


Figure 2.10: Discontinuity in the $f'(x) = -1/x^2$ graph

Another special case:

$$f(x) = e^x$$

This has the special property in which $f(x) = f'(x)$, the function is equal to its derivative. Where $f(x) = f'(x)$ holds, its values are either positive, negative, or 0. Taking the negative of the above equation gives a negative example while taking $f(x) = 0$ gives the 0 example. [Figure 2.11](#) shows the plot of the above function:

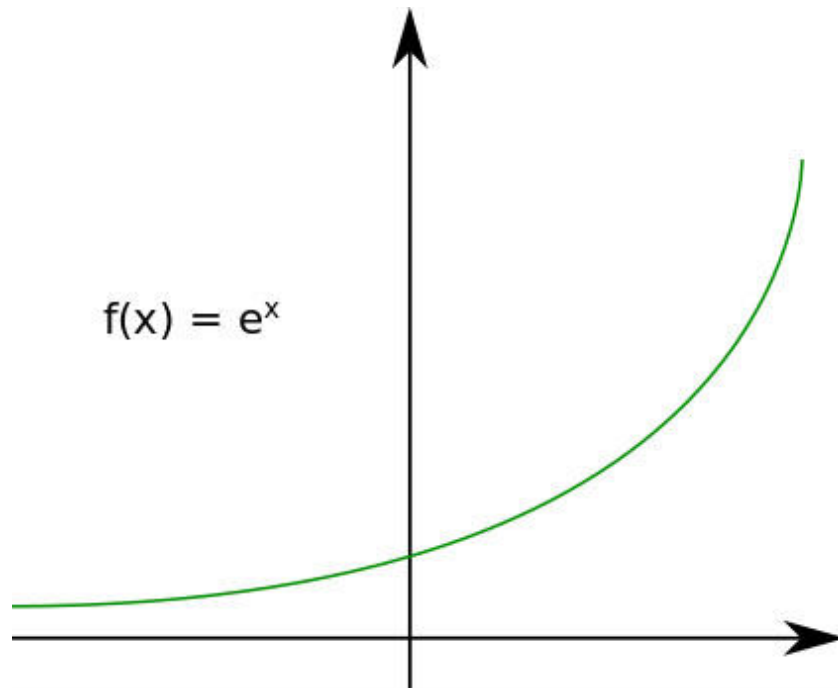


Figure 2.11: $f(x) = e^x$

These representations are the few special cases in differential calculus but not always occurs in data representation.

Trigonometric functions

The two trigonometric functions that we are focusing on will be the sine and cosine functions. [Figure 2.12](#) represents the notation $\text{sign}(x)$:

$$\text{Sine: } f(x) = \sin(x)$$

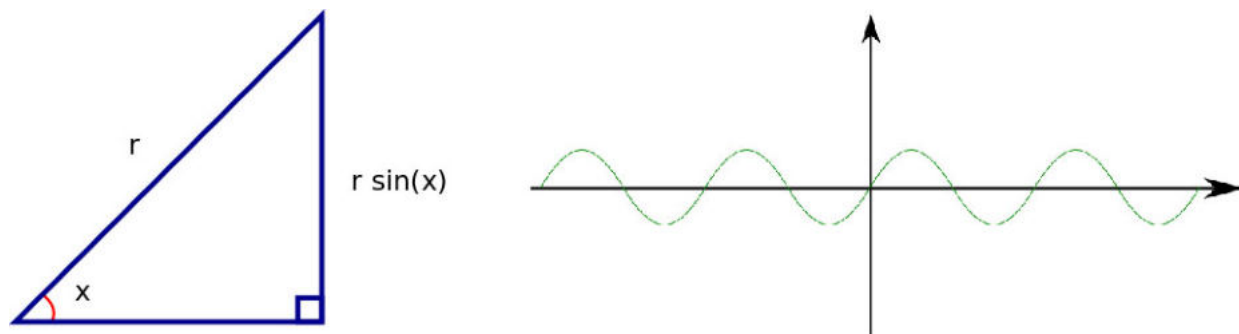


Figure 2.12: Sine triangle and sine graph

The derivative of the sine function is the cosine function which represented in [Figure 2.13](#):

$$\text{Cosine: } f(x) = \cos(x)$$

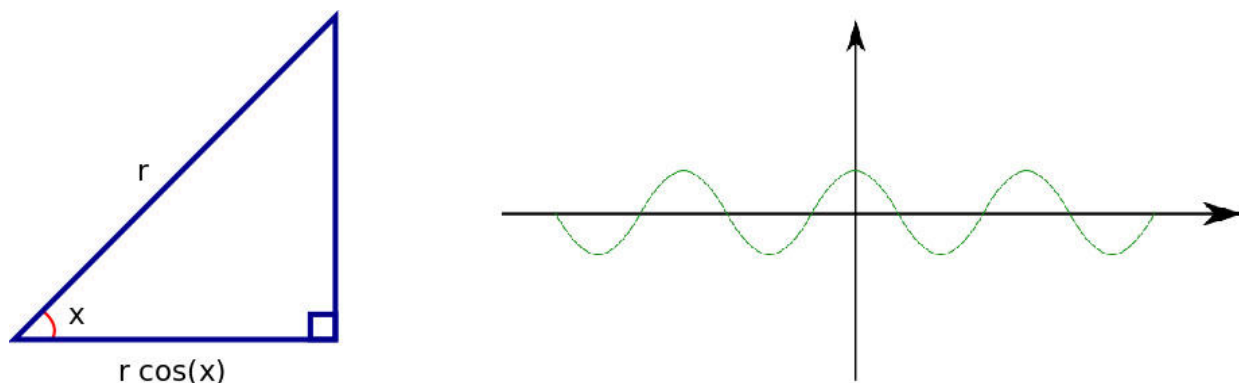


Figure 2.13: Cosine triangle and cosine graph

The derivative of the cosine function is the negative sine function. The two functions form a derivative loop that returns to the start every 4th derivation. [Figure 2.14](#) shows the sine cosine derivative loop:

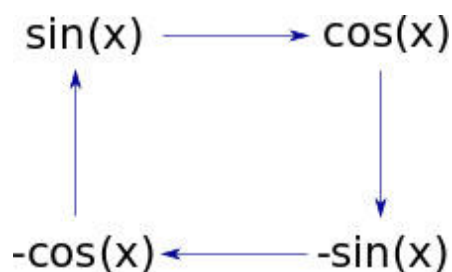


Figure 2.14: Sine Cosine derivative loop

Understanding the functions of sin and cosine will provide support when we plot the data in a graph then interpret the information from the data.

Product rule

Product rule defines identifying a derivative when two functions are in the product:

$$A(x) = f(x)g(x)$$

$$A'(x) = f'(x)g'(x)$$

$$\lim_{dx \rightarrow 0} \frac{dA(x)}{dx} = \lim_{dx \rightarrow 0} \frac{f(x)(g(x+dx) - g(x)) + g(x)(f(x+dx) - f(x))}{dx}$$

$$\lim_{dx \rightarrow 0} \frac{f(x)(g(x+dx) - g(x)) + g(x)(f(x+dx) - f(x))}{dx}$$

$$\lim_{dx \rightarrow 0} \frac{f(x)(g(x+dx) - g(x))}{dx} + \frac{g(x)(f(x+dx) - f(x))}{dx}$$

$$\lim_{dx \rightarrow 0} f(x)g'(x) + g(x)f'(x)$$

Chain rule

In general, a function has an *inside function* and an *outside function*. We can say the chain rule identifies derivatives as the *inside function* and the *outside function*. We can differentiate the outside function leaving the inside function alone and multiply all of this by the derivative of the inside function. This is represented in the below equation:

$$\text{If } h = h(p) \wedge p = p(m),$$

$$\text{Then } \frac{dh}{dm} = \frac{\frac{dh}{dp} * dp}{dm}$$

The application of chain rule is very helpful when we try to find derivatives in function has the looping property.

Quotient rule

This rule can be derived from the product rule:

$$\text{Consider a function } A(x) = \frac{f(x)}{g(x)}$$

$$\text{Then } A'(x) = \frac{g(x) * f'(x) - f(x)g'(x)}{g(x)^2}$$

This rule can be used when we want to estimate the function fractional value.

Multiple variables

In a function, there are independent variables, dependent variables, and possibly constants as well. Consider [Figure 2.15](#) shows the example of plot car's speed time to time basis. Here, time is the variable of independent in nature. Where the speed is a dependent variable. The relationship between these two variables is such that at any given speed, there can be time periods matching that particular speed, while at each time period, there is only one speed:

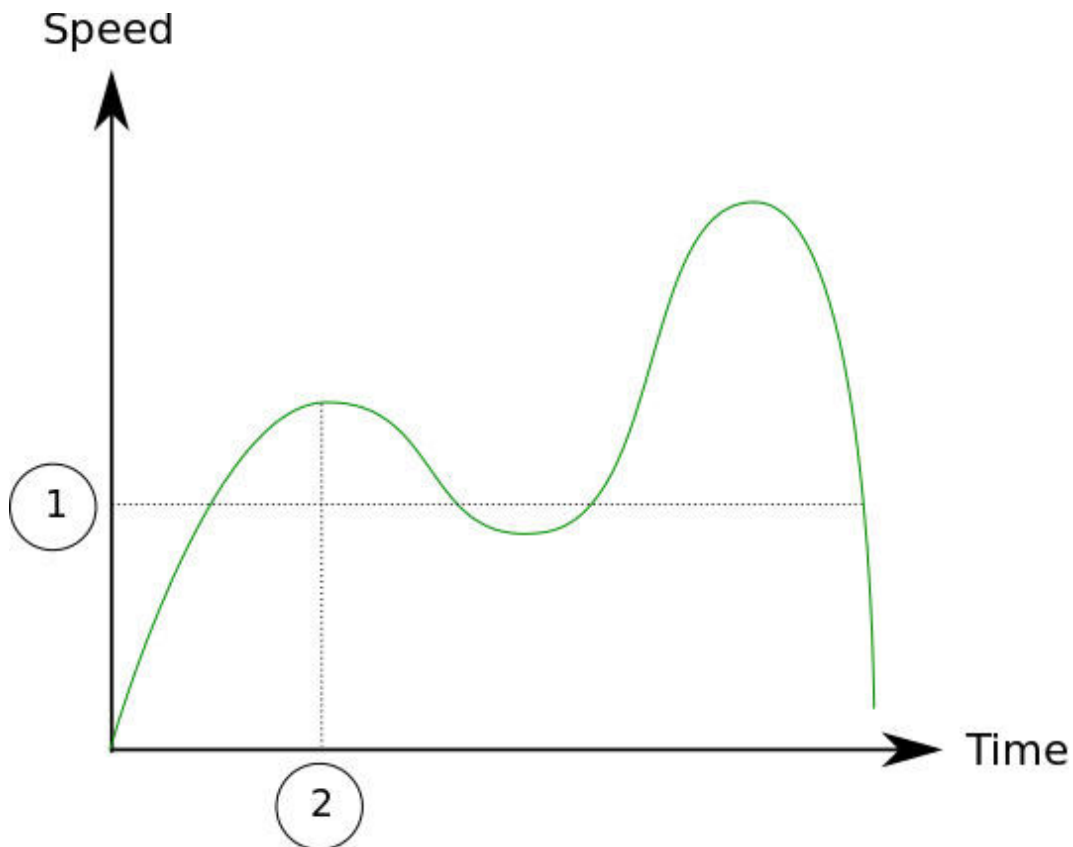


Figure 2.15: Speed vs. Time

As such, the speed of the car is dependent on the time period mentioned. The parameter's values a function that may vary in their variable types based on the context in which the function is used.

Partial differentiation

Partial differentiation is differentiating with respect to each variable in turn. For each partial differentiation, regard the other variables as constants in the differentiation context.

Total derivative

The total derivative is the derivative with respect to a specific variable where the function is dependent on the variable not only directly but also on other variables that are dependent on that specific variable.

Integral calculus

Integration is a way of adding slices to find the whole. Integration can be used to find areas, volumes, central points, and many useful things. But it is easiest to start with finding the *area under the curve of a function* like this:

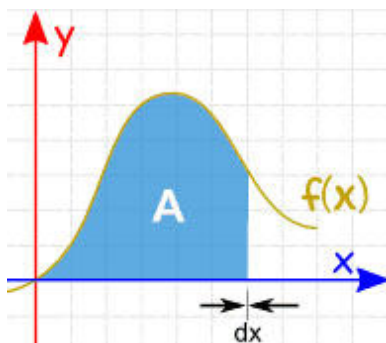


Figure 2.16: What is the area under $y = f(x)$?

Slices

We could calculate the function at a few points and add up slices of width Δx like this (but the answer won't be very accurate) as shown in [Figure 2.17](#):

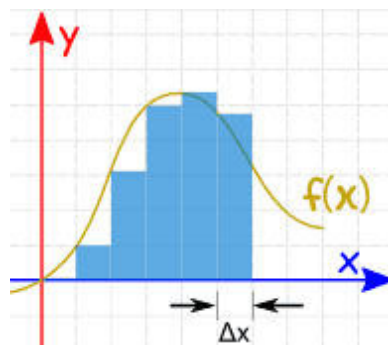


Figure 2.17: shows adding slices of width Δx :

We can make Δx a lot smaller and *add up many small slices* (the answer is getting better) as shown in [Figure 2.18](#):

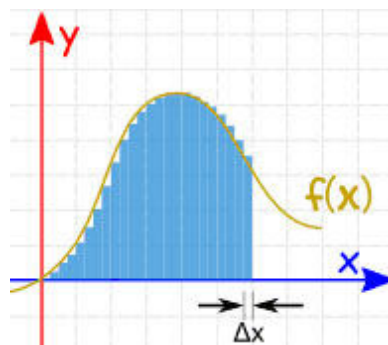


Figure 2.18: shows adding smaller slices of width Δx .

And as the slices approach zero in width, the answer approaches the *true answer*, as shown in [Figure 2.19](#).

We now write dx to mean the Δx slices are approaching zero in width.

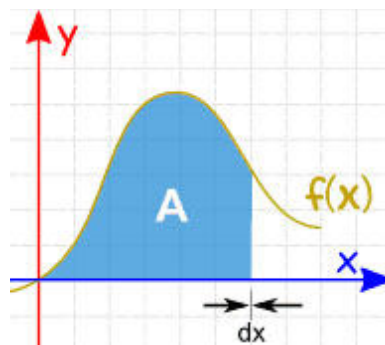


Figure 2.19: shows slices of zero width.

These are the different representations of slices in multiple variable estimations.

Definite vs.indefinite integrals

We have been doing *Indefinite Integrals* so far, which is If $f(x)$ is an anti-derivative of $f(x)$ then the most general anti-derivative of $f(x)$ is called an indefinite integral and denoted:

$$\int f(x) dx = F(x) + c$$

Where c is any constant.

[Figure 2.20](#) shows the representation of finite integral and infinite integral values. A Definite Integral has actual values to calculate between (they are put at the bottom and top of the S):

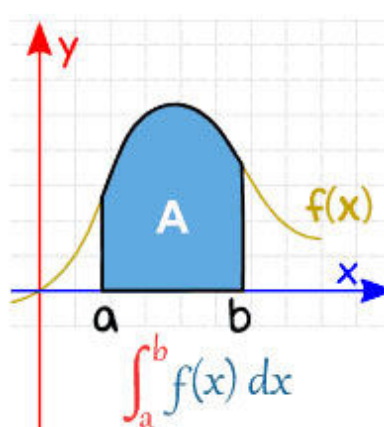
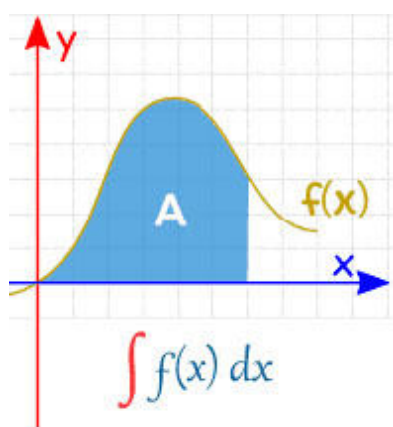


Figure 2.20: Indefinite Integral & finite Integral

The definite integral of $f(x)$ from a to b is:

$$\int_a^b f(x)dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i) \Delta x$$

Given a function $f(x)$ that is continuous on the interval $[a,b]$, we divide the interval into n subintervals of equal width (Δx) and from each interval choose a point, x_i .

The Gradient

To calculate a derivative of a function which is dependent on more than one variable or multiple variables, a Gradient takes its place. A gradient is calculated using Partial Derivatives. Also, another major difference between the Gradient and a derivative is that a Gradient of a function produces a Vector Field.

The Jacobian

The Jacobian of a set of functions is a matrix of partial derivatives of the functions. If you have just one function instead of a set of functions, the Jacobian is the gradient of the function.

The Hessian

The rate of change of the function is simply described as Hessian. It is positive definite and helps us to check if point x is a local maxima, local minima, or a saddle point. The function attains an isolated local minimum and an isolated local maximum at x , if the Hessian is positive definite negative definite at x , respectively. x is a saddle-point for function, if the Hessian has both positive and negative eigenvalues.

The Lagrange multipliers

The Lagrange multiplier technique lets you find the maximum or minimum of a multivariable function $f(x,y,...)$ when there is some constraint on the input values you are allowed to use.

Laplace interpolation

Consider a (two dimensional) data matrix with some values missing, and you want to fill the holes by interpolated values. One particularly simple but at the same time powerful method is Laplace interpolation. For each missing value, take the average over the 4 surrounding values, that is, of the entries above, below, left and right. The Laplace interpolation replaces these values by interpolated ones and writes them back to the input matrix sample2.

Optimization

In ML, the focus is on learning from data. A cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and Y.

The objective of an ML model, therefore, is to find parameters, weights, or a structure that minimizes the cost function. The cost function can be estimated by iteratively running the model to compare the estimated predictions of y by the model against the known values of y.

We will use the Mean Squared Error function to calculate the cost. At first, we will find the difference between the actual y and predicted y value ($y = mx + c$), for a given x. Then we will square this difference. Finally, we will calculate the mean of the squares for every value in X.

$$Y = mX + c$$

$$E = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2$$

$$E = \frac{1}{n} \sum_{i=0}^n (y_i - (mx_i + c))^2$$

The Gradient Descent algorithm

Gradient Descent is an iterative optimization algorithm to find the minimum of a function. Here that function is our *Loss Function*.

Let's try applying gradient descent to our approach it step-by-step:

1. Initially let $m = 0$ and $c = 0$. Let L be our learning rate. This controls how much the value of m changes with each step. L could be a small value like 0.0001 for good accuracy.
2. Calculate the partial derivative of the loss function with respect to m , and plug in the current values of x , y , m and c in it to obtain the derivative value D . D_m is the value of the partial derivative with respect to m .

$$D_m = \frac{1}{n} \sum_{i=0}^n 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n} \sum_{i=0}^n x_i(y_i - \hat{y}_i)$$

Similarly let's find the partial derivative with respect to c , D_c :

$$D_c = \frac{-2}{n} \sum_{i=0}^n (y_i - \hat{y}_i)$$

Below is a Python code showing the Gradient Descent Algorithm:

```
m = 0
c = 0
L = 0.0001 # The learning Rate
iterations = 1000 # The number of iterations to perform gradient descent
n = float(len(X)) # Number of elements in X # Performing Gradient Descent
for i in range(iterations):
    Y_pred = m*X + c # The current predicted value of Y
    D_m = (-2/n) * sum( X * (Y - Y_pred)) # Derivative w.r.t m
    D_c = (-2/n) * sum( Y - Y_pred) # Derivative w.r.t c
    m = m - L * D_m # Update m
    c = c - L * D_c # Update c
print (m, c)
```

[Figure 2.21](#) shows the red straight line as the linear regression line fitting:

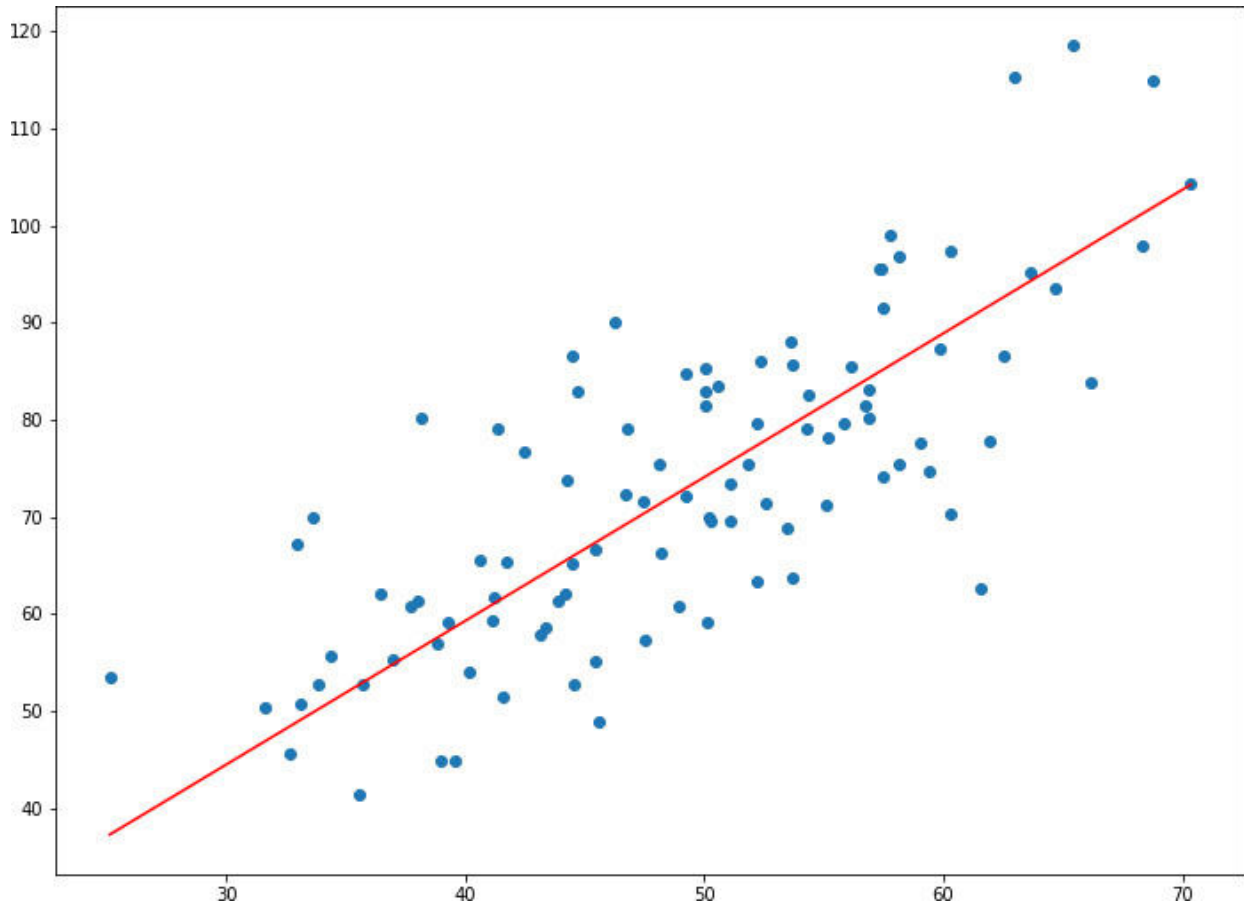


Figure 2.21: Linear regression line fitting

The equation: $Y_{pred} = mX + c$ will give the best fit linear equation for the data set as shown as the red straight line in the [Figure 2.21](#).

Conclusion

In this chapter, we have learned the essential mathematics for any data science applications. This is not limited to learn the concepts behind the algorithms. The sections are created starting from basic mathematical notation data to the advanced level mathematical concepts applied in the data science applications.

By reading this chapter, the reader will have better learning to approach any data science applications from the mathematical perspective. In the next chapter, we will start exploring statistical analysis and the techniques

involved in the data analysis. We will discuss the basic statistical concepts that a data science practitioner needs to understand.

CHAPTER 3

Statistics Essentials

If we consider data science as an art, then statistics is the key to perform the operations on it. If we see from the perspective of high-level concepts, we can say statistics is the application of mathematics to perform technical based analysis on data. Simply by using the bar chart, we infer some high-level information from the data, but if we use statistics, we can get deeper into the data and find much more information towards the objectives of the analysis. Statistics provide solid proof about our data, not just random estimation.

Using statistics, we can go further in deep and more fine-grained essential insights into how exactly the given data is structured and based on that structure how we can optimally apply other data science techniques to get even more information. In the previous chapter, we studied the essentials of mathematics. In this chapter, we're going to look at basic statistics concepts that data scientists need to know and how they can be applied most effectively.

Structure

- Introduction to probability and statistics
- Descriptive statistics
- Conditional probability
- Random variable
- Inferential statistics
- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of probability.

- Understand the importance of descriptive statistics and apply related techniques.
- Apply the conditional probability calculations in a real-world problem.
- Understand the usage of random variables and inferential statistics.

Introduction to probability and statistics

Meteor showers are rare, but the probability of them occurring can be calculated. (credit: Navicore/Flickr). Probability is the chance that something will happen and how likely it is that some event will happen.

Probability of an event happening $P(E) = \text{Number of ways it can happen } n(E) / \text{Total number of outcomes } n(T)$

We can say the probability is the measure of the chance of an event occurrence. The probability is measured with a scale of 0 to 1. Where 0 means impossibility and 1 is a certainty. In different aspects of our day to day activities, randomness and uncertainty happen. To understand those uncertainties, understanding the probability is very helpful. It enables us to create an estimation of what is going to come next. Also, we can say that probability learning works based on the informed assessment with the pattern of information collected earlier. In general, data science uses information based on the statistical properties. It must forecast or perform some analysis on the data to find some trends in it. Probability distribution has a major impact on statistical information formation. That makes that knowing probability and its application is very important to work with data science applications.

Statistics is a mathematical tool for collecting, analyzing, interpreting, and presenting the information. Statistics is used to deal with complicated issues in the actual globe so that data scientists and analysts can search for relevant trends and data modifications. In easy words, statistics can be used by making mathematical computations on it to obtain significant ideas from the information. In order to analyze raw information and create a statistical model and predict the outcome, different statistical functions, and algorithms are applied. Statistics has an impact on all areas of life, but to name a few are the stock market, insurance, weather, life sciences, retail, and education.

Descriptive statistics

In descriptive statistics, analysis of data is done so that information can be described, revealed, or summarized meaningfully so that anyone who sees it can detect specific patterns. The first stage in your statistical analysis, when looking at information, is to determine whether the dataset is a population or a sample.

[Figure 3.1](#) shows that collecting the interest items based on the study gives the population and the notation of it is capital letter N :

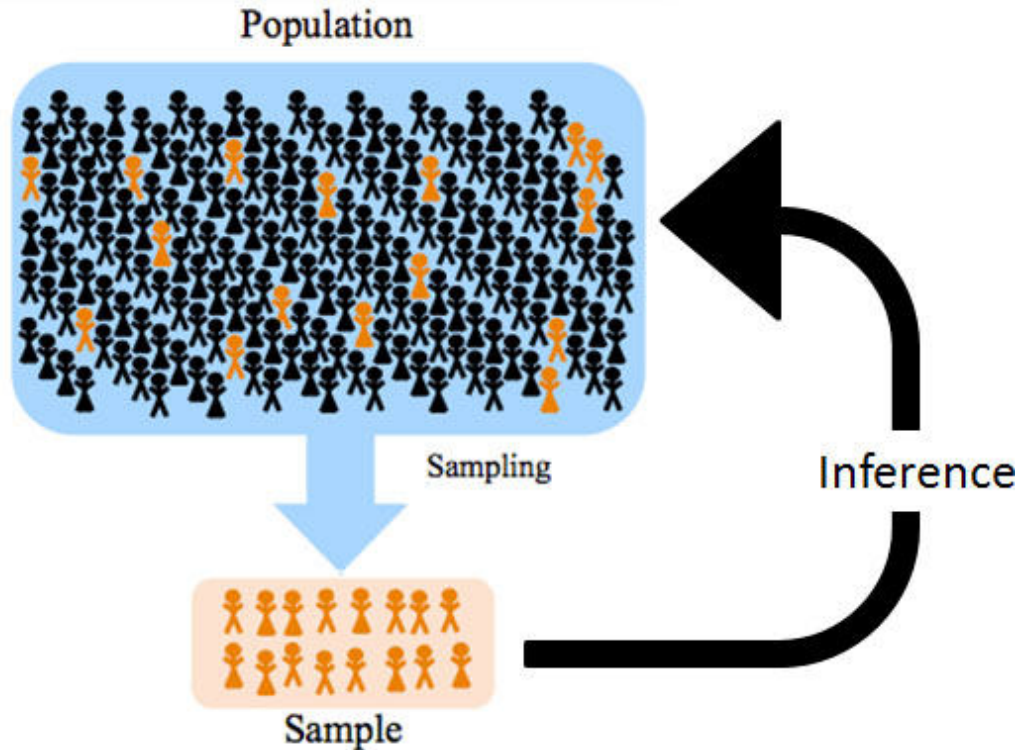


Figure 3.1: Population, Sample, and Inference

From that parameters are the calculated values in the analysis of the population. On other hand, the population subset is known as a sample. It is denoted by using the small letter n .

It is very difficult to evaluate and define the populations in real life. Studying the entire population and it influences. This process is inefficient because of time-consuming process as well; its cost is too high. Errors will easily acquire when we use the entire population as it is.

A sample is opposite to the population, meaning that a sample will consider only part of a population to evaluate, and this makes the process easier because it reduces the size of data. It directly implies the assessment time of a sample is very less expensive, and the occurrences of mistakes are very

less. If we choose a random size sample, then that must act as a representative for the entire population. This must provide the flexibility to anybody to deduct the population from the sample.

[Figure 3.2](#) shows that there are different varieties of data. In a dataset, data may be in the form of numerical or categorical values. Categorical data represented by groups or categories in which they belong. For example, ages, names, automotive brands are coming under the categorical data. Numerical data further divided into two categories discrete numbers and continuous numbers. Let us see in brief about the types of numerical data:

- **Discrete:** This type of data can only take certain values. Only a set of values to which you have access are defined. Age, number of vehicles on a road, number of fingers, for instance.
- **Continuous:** This type of data may, without limitations, take any true or fractional value between certain ranges. For example, weight, the balance of a bank account, purchase value, examination grade.

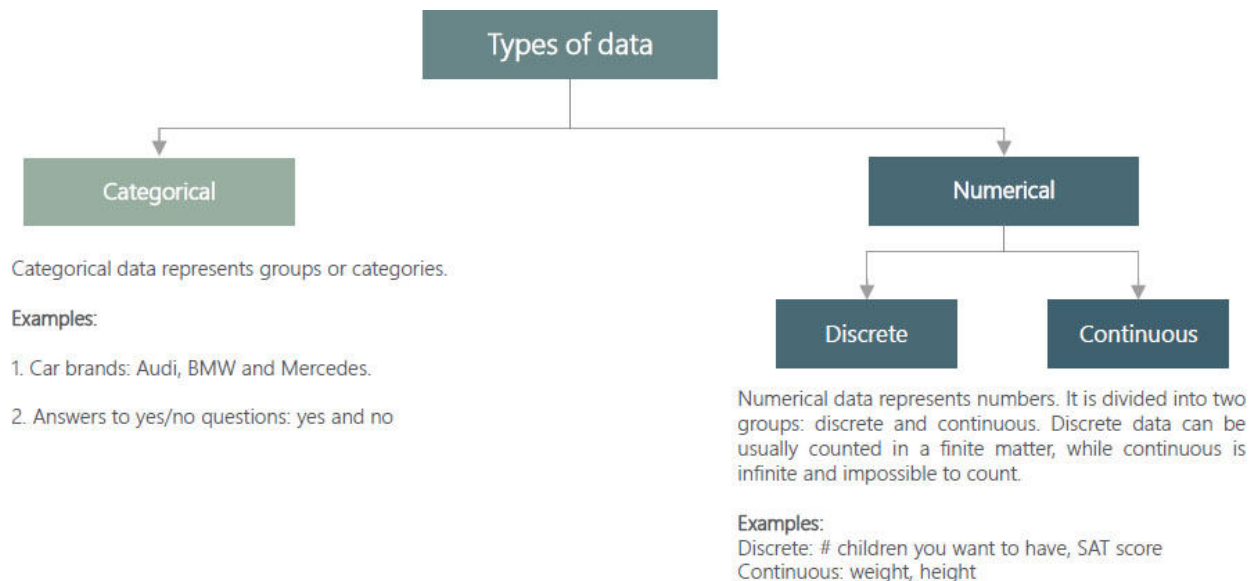


Figure 3.2: Types of Data

Measurement of data can be done in two levels: qualitative and quantitative.

- **Qualitative data:** This type of data measurement characterizes data but doesn't measure the attributes in data. It can be further divided into two groups: nominal and ordinal.

- **Nominal:** Nominal data are not numbers, and it can't be placed in any order.
Example: Gender (for example, male, female, others)
- **Ordinal:** Ordinal data consist of groups and categories in strict order
Example: Grades (for example, good, satisfy, bad)
- **Quantitative data:** It measures attributes in the data. It can be divided into two groups: interval and ratio.
 - **Interval:** It is represented by numbers, without having a true zero. In this case, the zero value is meaningless.
 - **Ratio:** It is represented by numbers and has a true zero.

Based on the context, we can take the quantitative data as an interval or ratio. For instance, consider the temperature value. When we say, 00 Celsius or 00 Fahrenheit is not having any significance, since it is not really zero.

It depends on the context in which the quantitative data is considered as an interval or ratio. Think of the temperature, for instance. There is no significance saying 0° Celsius or 0° Fahrenheit since it is not the real zero. The value of absolute zero temperature is -273.15 °C and -459.67 °F. So, the temperature must, therefore, be considered as interval data in this instance, since the zero value is irrelevant.

But if the temperature is measured in Kelvin is analyzed, the value of absolute zero is 0° Kelvin, so now the value of the temperature is ratio since it is a true zero.

[The measure of central tendency](#)

The measure of central tendency relates to the concept that there is a number that summarizes the whole set best. Mean, median, and mode are the most popular.

[Mean](#)

It is the most reliable measure of central tendency for hypothesizing a single sample population. In the case of a population value, μ symbol is used, whereas the sample means denoted by \bar{x} (Refer [Figure 3.3](#)):

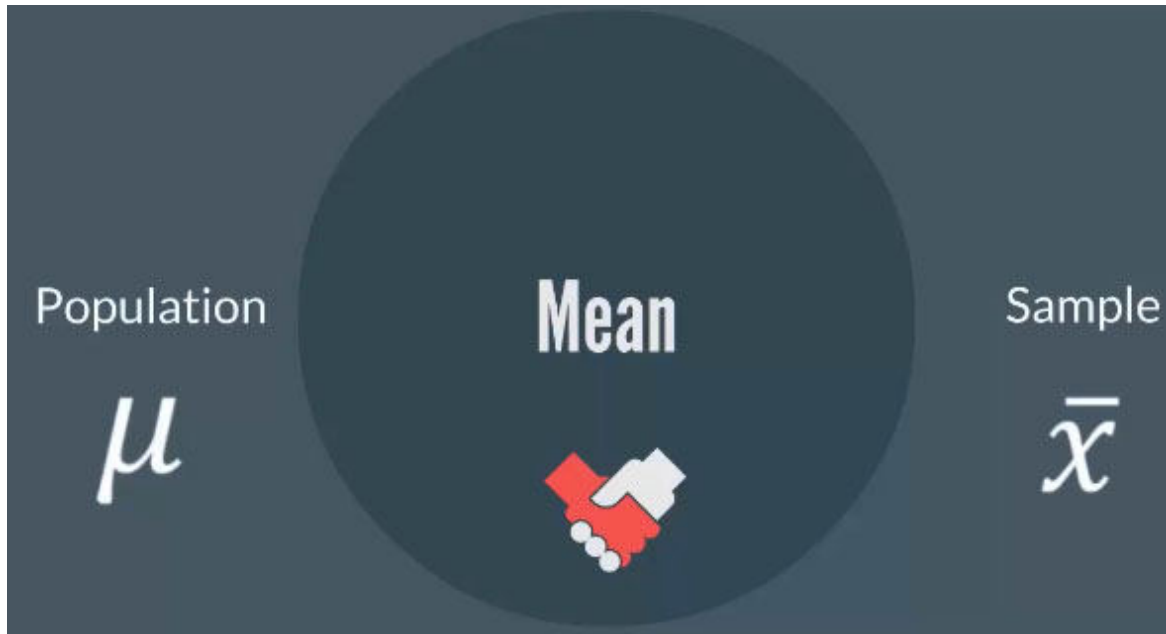


Figure 3.3: Mean value calculation

Mean is calculated by adding all the components and then divide the sum value by the number of components. This is the most commonly used measure of central tendency of data. But it can get affected by the outlier's data easily. It is not enough to come to a conclusion sometimes because of the excess amount of outliers' presence. The below equation is representing the mean of x value with n components:

$$\bar{x} = \frac{1}{n} \left(\sum_{i=1}^n x_i \right) = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Median

In an orderly ascending dataset, the median of the data set is the center value, also known as the 50th percentile. It is generally a good idea to calculate the median to prevent the error in the mean by outliers.

Below are the two equations for finding the median in the sets of data with an odd and even number of values:

$$1,3,3,6,7,8,9 \text{ Median} = 6$$

$$1,2,3,4,5,6,8,9 \text{ Median} = (4+5) \div 2 = 4.5$$

Mode

The mode gives us the most frequent value. It can be used for both numerical and categorical variables. If no single value appears more than once, then we can say that there is no mode. Rather than independently, the measures should be used together. In addition, these measures all appear at the same midline point in a normal distribution. The mean, mode, and medium are all the same!

Measures of variability

The measure of variability relates to the concept of evaluating the dispersion in our data by the mean value. Range, interquartile, variance, and standard deviation are the most common measures of variability.

Range

The range shows the difference between the largest and the smallest points in your data.

12,24,41,51,67,67,85,99

Range is $99 - 12 = 87$

Variance

The variance and standard deviation are the most challenging methods to measure the dispersion of the data from the mean value of the dataset:

$$\text{Population Variance: } \sigma^2 = \frac{\sum_{i=1}^n (X_i - X_{avg})^2}{n}$$

$$\text{Sample Variance: } s^2 = \frac{\sum_{i=1}^n (X_i - X_{avg})^2}{n-1}$$

The variance is calculated by measuring the difference between every data point and the mean and by squaring that value and adding all available data points. Lastly, the sum is divided by a total number of data points; thus the variance is calculated.

There are two main purposes for squaring the differences:

- Dispersion is a positive value because we square the subtraction to ensure that the negative values are not present and that they are not canceled.
- The effect of huge differences is amplified.

While calculating variance, squaring changes the unit of measurement from the original data. To nullify this problem, the standard deviation is computed, which is in the original unit.

Covariance

Covariance measures the changes in the two variables, x, and y, together:

$$\sigma(x, y) = \frac{1}{n} \sum_{i=1}^n (x - \mu_x)(y - \mu_y)$$

- When two variables are the same, the variance is the covariance.

$$\sigma(x, x) = \sigma^2(x)$$

- The variables are uncorrelated when the covariance value is 0
- Moreover $\sigma(x, y) = \sigma(y, x)$
- In an n-dimensional dataset, the covariance matrix, Σ , computes all possible pairs of dimensions:

$$\Sigma = \begin{bmatrix} \sigma(X_1, X_1) & \sigma(X_1, X_2) & \dots & \sigma(X_1, X_n) \\ \sigma(X_2, X_1) & \sigma(X_2, X_2) & \dots & \sigma(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(X_n, X_1) & \sigma(X_n, X_2) & \dots & \sigma(X_n, X_n) \end{bmatrix} \text{ where } X_i \text{ refers to the } i\text{-th}$$

element of all the vectors in the feature space, the covariance matrix is a **symmetric matrix**.

- From a set of vectors, if we subtract the mean value from each vector is known as Mean Centering. You can form n mean-centered vectors into a matrix, Z, where each row of the matrix will match one of the vectors. The covariance matrix is then directly proportional to the transpose of Z multiplied by Z.
- $\Sigma = Z^T Z$

Standard Deviation

Standard deviation is usually far more significant than a variance. It is the preferred variation metric, as it can be interpreted directly. Standard deviation is the square root of our variance.

$$\text{Population Standard Deviation: } \sqrt{\frac{\sum_{i=1}^n (X_i - X_{avg})^2}{n}}$$

$$\text{Sample Variance: } \sqrt{\frac{\sum_{i=1}^n (X_i - X_{avg})^2}{n-1}}$$

The best use of standard deviation is when data is in a unimodal shape (Refer [Figure 3.4](#)). In one standard deviation away from the mean, approximately 34% of data points are distributed in a normal distribution. Thus, we have 68.2% of data points arranged one standard deviation away from the mean since the normal distribution is symmetrical. Between the two standard deviations aside from the mean, approximately 95% of points are allocated, whereas, under three standard deviations, it is around 99.7%. Using Z-score, you can verify the total standard deviations below or above the mean.

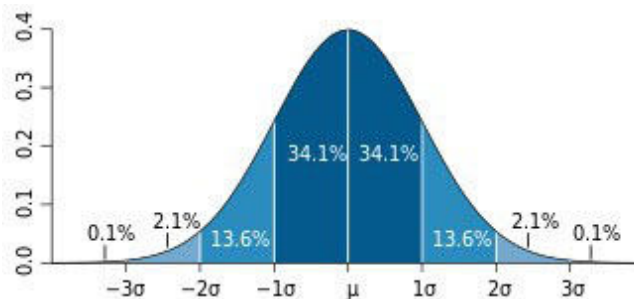


Figure 3.4: Standard Deviation calculation

This is how the measure of variability has been calculated by using standard deviation calculation. Let us move to the measure of asymmetry.

Measure of asymmetry

To measure the asymmetry in data, we can use two methods: Modality and Skewness. Let will start exploring the concepts.

Modality

The number of peaks presented by data allows for determining the Modality of a distribution. [Figure 3.5](#) shows different types of models. Generally, the majority of distribution is unimodal, and it has one value occurring frequently. There are two frequently occurring values in a bimodal. In uniform modal, the data distributed uniformly. In multi-modal data, more than two frequently occurring values will be there.

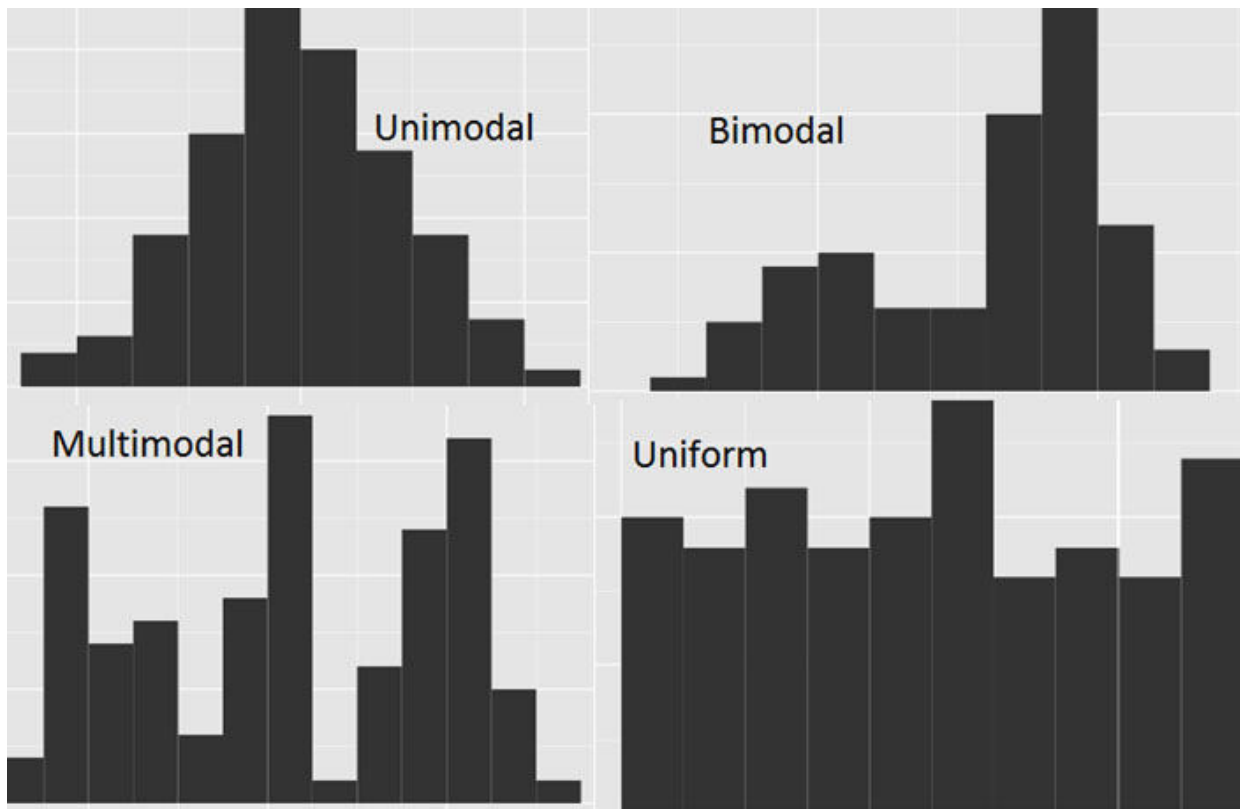


Figure 3.5: Types of Modal

Skewness

Skewness is one of the tools to measure asymmetry in data distribution. To view the where the data clustered more skewness helps to visualize it. Another important use of skewness is it can be used to capture the outliers in data. Based on the position of mean, mode, and median calculation, we find the skewness. [Figure 3.6](#) shows the different position of those values and how the asymmetry is calculated based on that. If the data has median value between the mean and mode, then it is called positive skew. In other words, we say outliers more towards the right side of the distribution. On the contrary, a median value higher than the mean value, then that is called as

left-skewed. If the mean, median, and mode are at the same point, then that distribution is called **symmetrical distribution**.

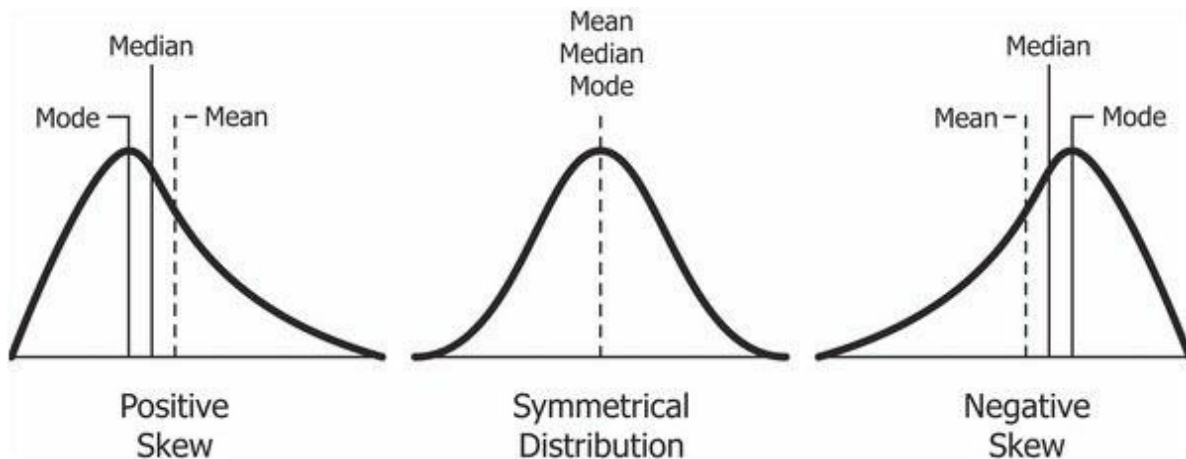


Figure 3.6: Measurement of asymmetry using Skewness

The connection between probability theory and central tendency measures are the measures of asymmetry. It helps to acquire more insights into the data we deal with.

Populations and samples

If the dataset contains entire data values, then that is called as populations. If we choose some random data from the population then that is called as samples. [Figure 3.7](#) shows that from the perspective of statistics, populations are parameters and samples are used to do statistical analysis:

- **Population:** It is the set of all possible states of a random variable. The size of the population may be either infinite or finite.
- **Sample:** It is a subset of the population. Normally, when the population is big enough to analyze the entire set, we use samples.

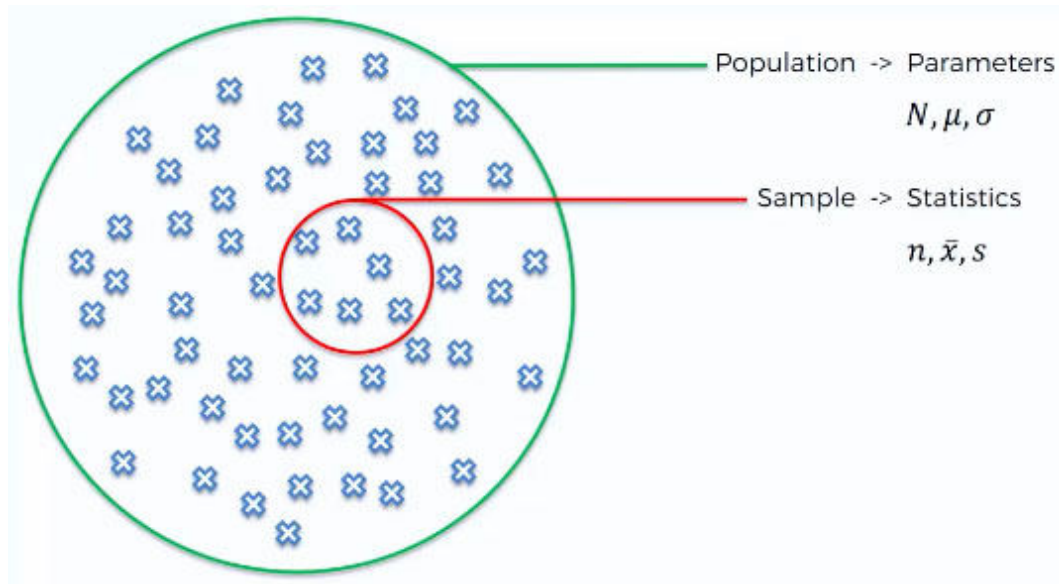


Figure 3.7: Population and Sample

Appropriate selection of populations and samples will provide more clarity on the processing of the data. That also helps to get more information about the data with less computing time and power.

Central Limit Theorem

It is a powerful and most crucial theorem of Mathematics. It states that *the sampling distribution will look like a normal distribution regardless of the population you are analyzing.*

Sampling distribution

As discussed earlier, to estimate the parameters of a population, we take a sample. But it's not the only way of extracting the exact estimates of the parameters. [Figure 3.8](#) shows the sampling distribution from a population. We can also take multiple samples from the population. For instance, we will calculate the average for every sample. So, at the end of the day, we have several mean estimate values, and we can then visualize them on a chart. This will be called the sampling distribution of the sample mean.

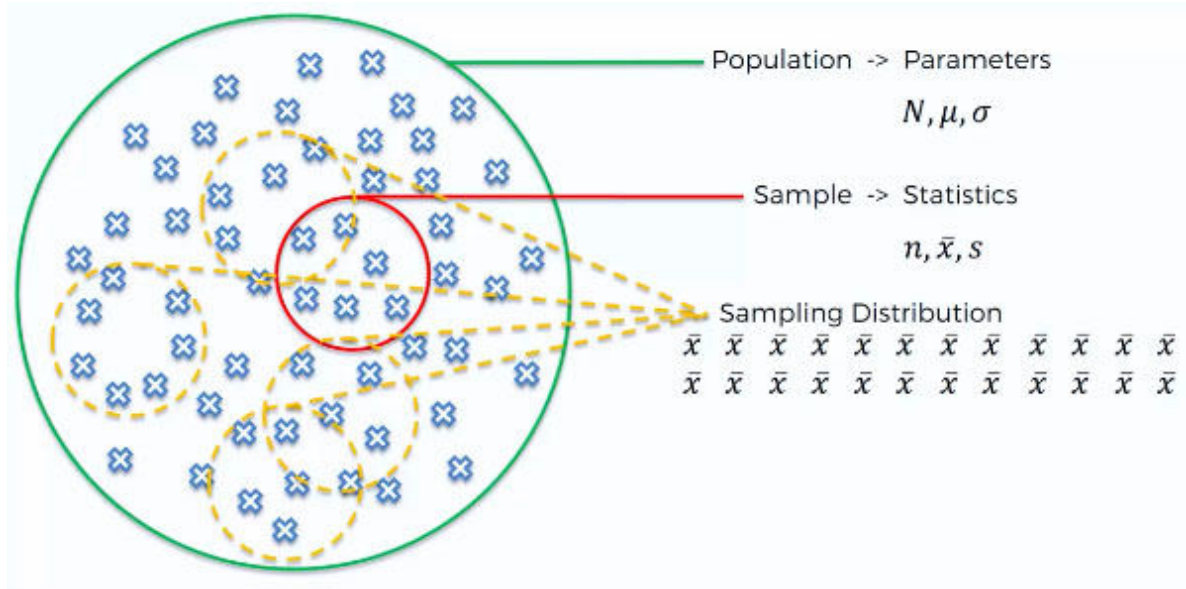


Figure 3.8: Sampling Distribution

Identifying the sample distribution on the sample means it is very important to understand the data in a better way. The visualization of sample distribution is more useful to investigate the data distribution easily.

Conditional probability

Conditional probability can be characterized as a measure of the probability of an event provided that some other event has occurred.

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

The probability of an event B if event A is equivalent to the probability of event A and event B divided by the probability of event A. Most of the data science techniques depend on Bayes' Theorem. It is a formula that describes how to update the probability of hypotheses if the evidence is provided:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Given a new set of attributes, it is possible to create a learner using the Bayes' theorem that calculates the probability of the variable belonging to some other class:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In a case, where A represents a hypothesis and B represents some observed evidence E, the equation can be written as:

$$P(H|E) = \frac{P(E|H)}{P(E)} P(H)$$

It co-relates the probability of the hypothesis before acquiring evidence $P(H)$, to the probability of the hypothesis after receiving the evidence, $P(H|E)$. Thus, $P(H)$ is called the prior probability, while $P(H|E)$ is called the posterior probability and $\frac{P(E|H)}{P(E)}$, is called the likelihood ratio. Hence, Bayes' theorem can be rephrased as *the posterior probability equals the prior probability times the likelihood ratio*.

Random variables

A random variable is a collection of probable values of a random experiment, or it can be characterized as a variable whose probable values are the product of a random experiment. There can be discrete or continuous random variables. Within a range, continuous random variables can take any value, but discrete random variables can only take certain values.

A discrete random variable can be defined as a variable that can only take into consideration a limited amount of specific values such as 0, 1, 2, 3, 4, A random variable must be discrete if it is able to take only a finite number of distinct values. For example, we can take counting the number of candidates in the examination hall, the number of students in a school, the number of patients in a doctor's chamber, the number of faulty light bulbs in a set of eight are coming under the distinct random variables.

A discrete random variable's probability distribution is a list of probabilities associated with every possible value. Let us suppose k different values are taken by a random variable X , with the probability that $X = x_i$, which can be defined as $P(X = x_i) = p_i$. The probabilities p_i must meet the following requirements:

1. $0 \leq p_i \leq 1$ for each i
2. $p_1 + p_2 + \dots + p_k = 1$

For all the discrete and continuous random variables, there must be a cumulative distribution function. For every value of x , the probability that the random variable X being less than and equal to x is given by this function. The cumulative distribution function can be determined by summing up all the probabilities for a discrete random variable.

A continuous random variable is one which takes an infinite set of possible values. They are basically measurements; for example: height, weight, the amount of salt in toothpaste, the time required to walk 1 kilometer. A continuous variable is defined over an interval of values and is expressed by the area under a curve. As the random variable may take an infinite number of values, hence the probability of occurrence of value is found to be 0.

Let us suppose that over an interval of real numbers, all values are taken by a random variable X . Then, the probability that X is in the set of outcomes A , $P(A)$, is defined to be the area above A and under a curve. The curve representing a function $p(x)$, must meet the below points:

1. There are no negative values in the curve ($p(x) \geq 0$ for all x).
2. The total area under the curve is 1.

Understanding the random variables is very important to use them effectively in the analysis of data based on inferential statistics. Let us start reading more about the inferential statistics in the next section.

Inferential statistics

In Inferential statistics, a random sample of data from a population is used for the purpose of describing the population and predicting it. Inferential statistics are basically intended to make inferences on populations based on the samples taken from data. This gives us a conclusion that descriptive statistics describe the data, and inferential statistics enables you to estimate from that data.

In this chapter, we will be analyzing the below concepts.

Probability distributions

- Normal
- Binomial

- Poisson
- Geometric
- Exponential

What is a probability distribution?

A probability distribution is a mathematical function that can be interpreted to be the probability that various possible effects happen during an experiment. Also, you can say that it is a function showing the probable values and how often they occur. It is the rule that determines the relation of the values with each other. [Figure 3.9](#) describes associations between different distributions. Most of them follow Bernoulli distribution:

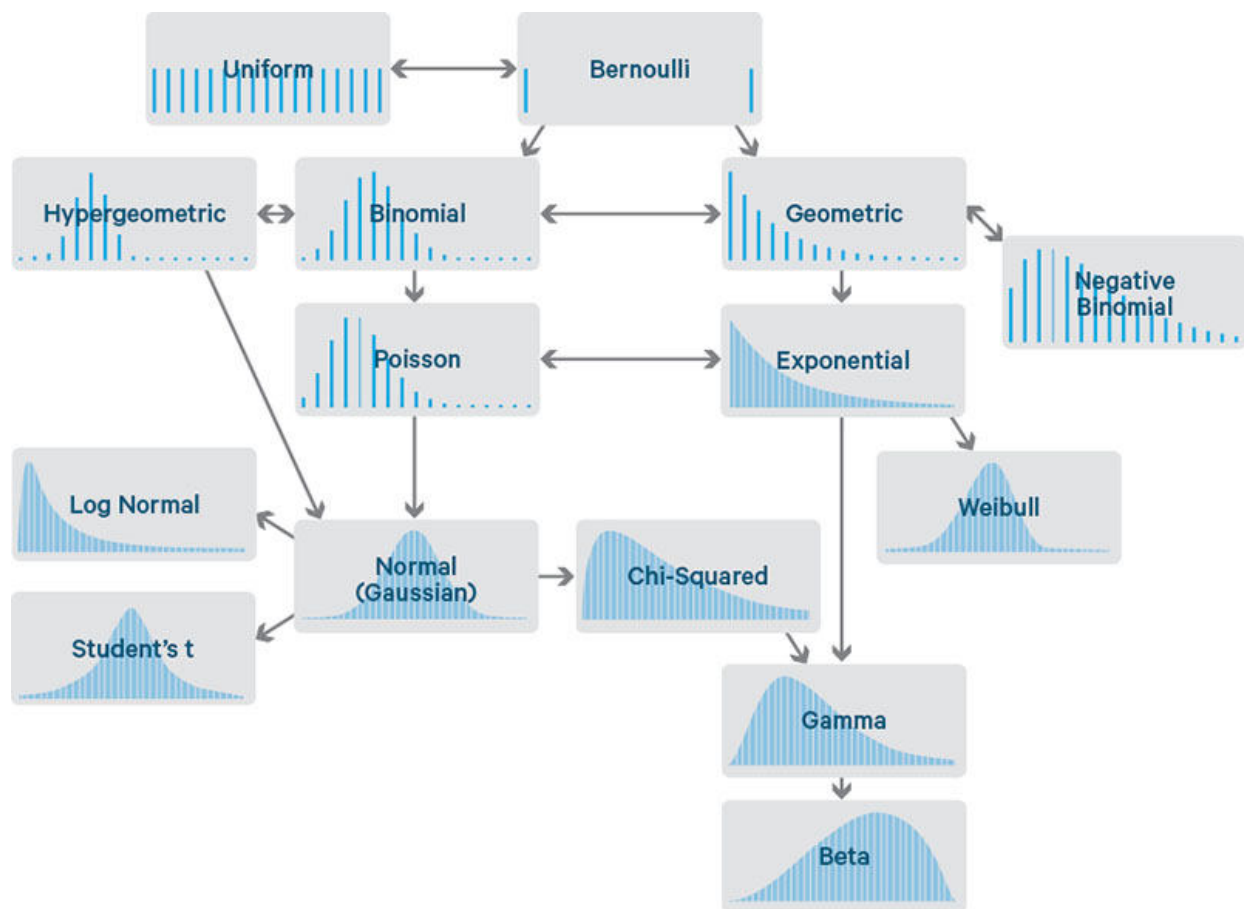


Figure 3.9: Types Probability Distribution

Let us begin with the most widely used distribution, normal distribution.

Normal distribution

The most common continuous variable probability distribution is the normal distribution, which is also known as Gaussian distribution or the Bell curve. [Figure 3.10](#) shows the normal distribution with its function. It is represented by the below notation:

$$N(\mu, \sigma^2)$$

Where N means normal, \sim represents the distribution, μ is the mean, and σ^2 is the variance.

Normal distribution does not have any skewness as it is symmetrical, and its median, mean, and mode are of the same value.

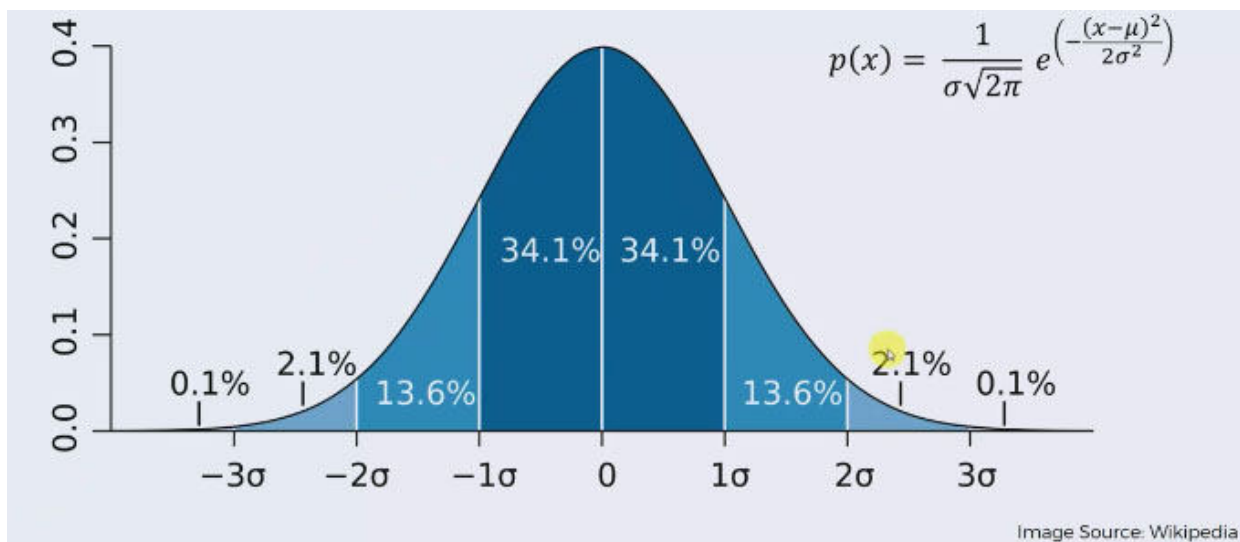


Figure 3.10: Normal Distribution

For example, using Python's `scipy.stats` module's `rvs()` method, let us generate 10000 random variables. The `loc` parameter defines the mean and `scale` defines the standard deviation of the distribution.

Let us start to generate some random numbers and see how the normal distribution is applied by code:

```
import pandas as pd
import seaborn as sns
from scipy import stats
from scipy.stats import norm, binom

# generate random numbers from N(0,1)
norm_data = norm.rvs(size=10000, loc=0, scale=1)
```


Using the Pandas library, we transform these 10000 random variables into a Series and then plot it using the histogram style:

```
pd.Series(norm_data).plot(kind="hist", bins=100)
```

From [Figure 3.11](#), we can be clearly able to visually see that a normal distribution with the highest frequency of values lying around the mean value, that is 0:

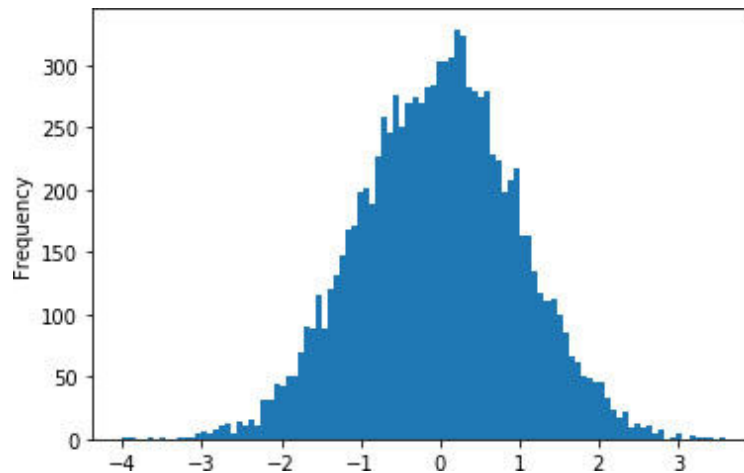


Figure 3.11: Normalized Data

As we have attained our distribution, we can able to start obtaining more insights from the data using some other functions.

Cumulative distribution function:cdf()

This function gives us the probability of a certain random observation will have a lower value than the one provided by the user. For example, imagine we select a random variable being 1.5, so what percentage of the values will be lesser than this number?

```
stats.norm.cdf(x=1.5, #Limit value to check  
loc=0, #Mean  
scale=1) #Standard Deviation  
0.9331927987311419
```

Percent point function:ppf()

It is the opposite of the Cumulative distribution function. Here, we input the probability and receive the quantile.

We want to know above which number are 93.3% of 10000 random variables!

```
stats.norm.ppf(q=0.933, #Limit value to check  
loc=0, #Mean  
scale=1) #Standard Deviation
```

1.4985130678799763

Probability density function: **pdf()**

Assuming a certain value, this function gives us the likelihood of a random variable.

```
stats.norm.pdf(x=1.2, #Limit value to check  
loc=0, # Mean  
scale=1)#Standard Deviation
```

0.19418605498321298

The Standard Normal Distribution

The Standard Normal Distribution is a case of the Normal distribution. It has a mean of 0 and a standard deviation of 1. Every Normal distribution can be 'standardized' using the following formula:

$$z = \frac{x - \mu}{\sigma}$$

Where z represents the standard normal distribution, μ represents mean value, σ , and represents the standard deviation. X is the data value.

Binomial distribution

A binomial distribution is a type of distribution that has two possible outcomes and can be simplified as the probability of a SUCCESS or FAILURE outcome in any experiment that is repeated multiple times. [Figure 3.12](#) shows the binomial distribution of data with no. of events against the frequency of occurrences. For example, flipping a coin is a type of Binomial distribution $B(n,p)$, where n is the total number of events, and p is the probability of success in each flip.

Now let us toss the coin 20 times and assume the probability of getting heads is 50% for each time. Therefore, $n = 20$ and $p = 0.5$ and our binomial

distribution is equal to $B(20,0.5)$. Now let us generate 10,000 data points assuming this distribution, meaning we will perform 10,000 experiments in which we flip a coin 20 times. Python's `stats.binom` library and `thervs()` method are used here.

```
binom_data = binom.rvs(size=10000, n=20, p=0.5)
pd.Series(binom_data).plot(kind="hist", bins = 50)
```

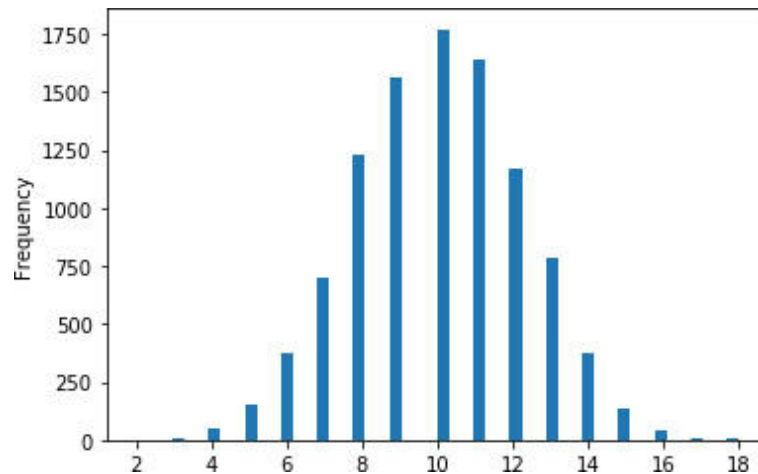


Figure 3.12: Binomial Distribution

Cumulative distribution function:cdf()

As discussed earlier, this function will provide us the probability of a random variable. For instance, what is the probability of getting 8 head in 10 flips with a coin?

```
stats.binom.cdf(k=8, n=10, p=0.8)
0.6241903616
```

Where k is the probability of $k = 8$ or less, n is the number of flips, and $p = 0.8$ is the success probability.

Probability mass function:pmf()

The binomial distribution is a discrete probability distribution; therefore, to check the proportion of observation at a certain point, we need to make use of the `pmf()` method. Let's check the probability of getting exactly 5 heads in 10 flips on our biased coin.

```
stats.binom.pmf(k=5, n=10, p=0.8)
0.026424115200000004
```

Where k is the probability of $k = 5$ heads, n is the total number of flips, and the $p = 0.5$ success probability.

Poisson distribution

The Poisson distribution is the distribution of something that occurred a number of times. It is denoted by an average rate of λ . The Poisson distribution is very useful when you want to measure events during a certain period because of repeated events. For example, consider the number of devotees coming to a temple in an hour is the data distribution. Therefore, in a certain period, we can think of such a distribution as the probability that an occurrence will happen multiple times.

When the below assumptions are valid, a distribution is called Poisson distribution:

- The results of a successful event should not be influenced by any successful event.
- The probability of success over a longer period must be equal to the probability of success over a short period.
- With a shorter interval, the probability of success approaches to zero.

Geometric distribution

The Geometric distribution is called the Negative Binomial Distribution. It analyzes the number of successes in an order of independent and identically distributed trials before a specified number of failures occur. As the number of failed attempts is the ultimate outcome of the Geometric distribution, therefore it is parameterized by the probability of that final success. For example, in the case of the Binomial distribution, you raise the question of *How many successes?* But in case of the Geometric distribution, you ask, *How many failures until a success?*

Exponential distribution

The exponential distribution is one of the continuous distributions commonly used. It models the time between events. A random variable X is said to have an *exponential distribution* with PDF:

$$f(x) = \lambda e^{-\lambda x}, x \geq 0$$

and rate $\lambda > 0$.

At any time, t , λ is the failure rate; that is, it has survived up to t .

The mean and variance of a random variable X following an exponential distribution:

$$\text{Mean} — E(X) = 1/\lambda$$

$$\text{Variance} — \text{Var}(X) = (1/\lambda)^2$$

The curve drops faster with an increasing rate, but it gets flatten with a decreasing rate. The below plot explains it better:

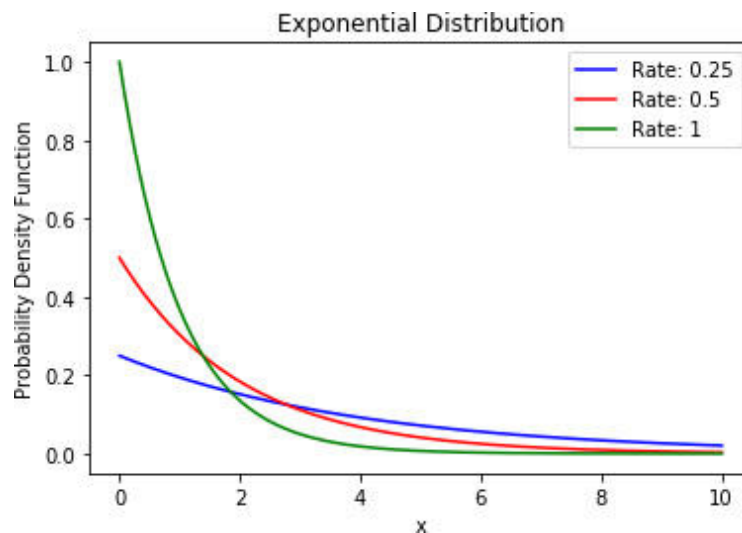


Figure 3.13: Exponential Distribution

Also, there are some formulas as given below, which ease the computation.

- The area under the density curve to the left of x can be represented by $P\{X \leq x\} = 1 - e^{-\lambda x}$
- The area under the density curve to the right of x can be represented by $P\{X > x\} = e^{-\lambda x}$
- The area under the density curve between x_1 and x_2 is $P\{x_1 < X \leq x_2\} = e^{-\lambda x_1} - e^{-\lambda x_2}$

Conclusion

Statistical analysis is the main important analysis to be made before further steps in applying any algorithms. In this chapter, we introduced the fundamentals of probability and statistics. We explored the various

descriptive statistics calculations techniques. In addition to that, we learned about the conditional probability and random variables used in the statistical analysis. We also learned important statistical distributions under the inferential statistics section. In the next chapter, we will start the machine learning concepts that are essential for any data science application.

CHAPTER 4

Exploratory Data Analysis

Analysis of the data is one of the key important steps in the machine learning process. In other words, we can say this is the first step we have to do in the entire machine learning process. By analyzing the data, we can get the clarity of the given dataset. The collected data from the various data sources may have some errors. Sometimes the data may be inconsistent with the corresponding machine learning process. In order to avoid those situations, we need to have a better understanding of the data given to the process.

Exploratory Data Analysis (EDA) is an approach to understand the dataset by making some summarization and visual representation on it. While summarizing the data, we can get some essential information that can be utilized while building our machine learning model. EDA will give better features to be used to find more useful insight from the data. This gives the different perspectives of data from visualizing the information.

So, in this chapter, we introduce the EDA process with corresponding methods used in EDA, and also we were given the key concepts to be known while processing the data in EDA process steps. Along with this, we have the example discussion to get a better understanding when they go for some real data analysis. This will enable the learners to know the basic needs to get into the analysis of data using these methods.

Structure

- Introduction to EDA
- Understanding Data
- Methods of EDA
- Key concepts of EDA
- Example Discussion
- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of EDA.
- Utilize statistic techniques to evaluate the data.
- Recognize the various approaches and steps involved to perform the EDA process.
- Implement the EDA process on a different dataset.

What is EDA?

Anyone who is exploring or working with data needs to understand the data with multiple perspectives before use that further in machine learning model building. To analyze the data, we can approach the data by visualization technique, and by applying statistical analysis, we get a better view. **Exploratory Data Analysis (EDA)** is the process of discovering hidden patterns and useful information from the data. [Figure 4.1](#) shows that the EDA process comprised of different steps such as data collection, data preprocessing, data cleaning, and data analysis. EDA will support to validate the questions on the data, which comes from the technical perspective. Results come from the EDA will provide the confidence to the machine learning model performance will be good by selecting essential features:

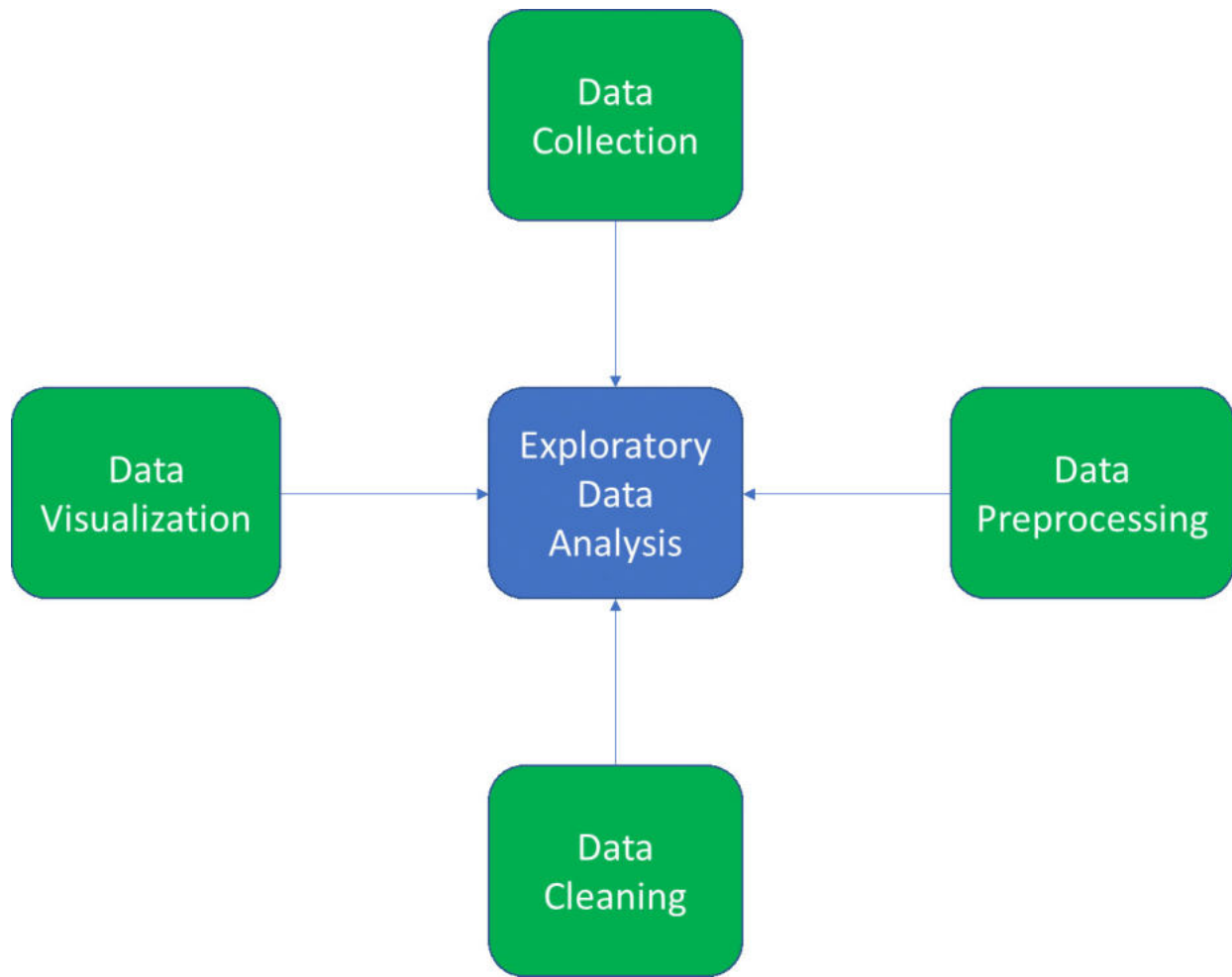


Figure 4.1: EDA Process

To be brief EDA process performs initial researches on data to uncover patterns that are hidden, to identify irregularities, and to validate the assumptions with the help of summary statistics and visual representations. Let us start to explore the EDA process from the next sections.

Need for the EDA

The main aim of the EDA process is to use statistics techniques efficiently summarize and visualizations to a better view of data, and find values about the importance of the data, its quality and to derive the new perspective and the suggestion of our analysis. EDA is always trying to give an answer to the questions on the data.

EDA is an approach for data analysis that involves a variety of techniques to:

1. Exploit understanding into a dataset
2. Discover different underlying structure into a dataset
3. Important feature extraction from the dataset
4. Identify outliers and irregularities
5. Getting the answer to the various assumptions on the dataset

EDA process is iterative in nature because we will make some thoughts and assumptions on our first look over the data, then we try to extract some useful insights from that data to build the machine learning models. Finally, we will make use of visualization techniques to preview the model results and tune them according to the applications.

Understanding data

Before getting into the learning about data exploration, let us first try to understand the types of data or levels of dimension on the data.

Data comes in various forms but can be classified into two main groups: structured data and unstructured. Structured data is data that is a form of data that has a high degree of an organization such as numerical or categorical data. Temperature, phone numbers, gender are examples of structured data. Unstructured data is data in a form that doesn't explicitly have the structure we are used to. Examples of unstructured data are photos, images, audio, language text, and many others. There is an emerging field called deep learning, which is using a specialized set of algorithms that perform well with unstructured data. In this guide, we are going to focus on structured data but provide brief information for the relevant topics in deep learning. The two common types of structured we commonly deal with are categorical variables (which have a finite set of values) or numerical values (which are continuous).

Categorical variables

Categorical variables can also be nominal or ordinal. Nominal data has no intrinsic ordering to the categories. For example, gender (male, female, other) has no specific ordering. Ordinal data has clear ordering such as three settings on a toaster (high medium and low). A frequency table (count of each category) is the common statistic for describing categorical data of each

variable, and a bar chart or a waffle chart (shown below) are two visualizations that can be used.

Numeric variables

Numeric or continuous variables can be any value within a finite or infinite interval. Examples include temperature, height, and weight. There are two types of numeric variables that are interval and ratios. Interval variables have numeric scales and the same interpretation throughout the scale but do not have an absolute zero. For example, the temperature in Fahrenheit or Celsius can meaningfully be subtracted or added (the difference between 10 degrees and 20 degrees is the same difference as 40 to 50 degrees) but cannot be multiplied. For example, a day which is twice as hot may not be twice the temperature.

The ratio scale of measurement is the most informative scale. It is an interval scale with the additional property that its zero position indicates the absence of the quantity being measured. You can think of a ratio scale as the three earlier scales rolled up in one. Like a nominal scale, it provides a name or category for each object (the numbers serve as labels). Like an ordinal scale, the objects are ordered (in terms of the ordering of the numbers). Like an interval scale, the same difference at two places on the scale has the same meaning. And in addition, the same ratio at two places on the scale also carries the same meaning.

A good example of a ratio scale is weight since it has a true zero and can be added, subtracted, multiplied, or divided.

Binning (numeric to categorical)

Binning, otherwise known as discretization, is the process of transforming numerical variables into categorical. For example, age can be categorized into 0-12 (child), 13-19 (teenager), 20-65 (adult), 65+ (senior). Binning is powerful as it can be used as a filter to reduce noise or non-linearity, and some algorithms such as decision trees require categorical data. Binning also allows data scientists to quickly evaluate outliers, invalid, or missing values for numerical values. Techniques for binning include using equal width (based on range), equal frequency in each bin, sorted rank, quantiles, or math

functions (such as log). Binning can be used based on information entropy or information gain.

Encoding

Encoding, otherwise known as a continuation, is the transformation of categorical variables into numerical (or binary) variables. A basic example of encoding is gender: -1, 0, 1 could be used to describe male, other, and female. Binary encoding is a special case of encoding where the value is set to a 0 or 1 to indicate the absence or presence of a category. One hot encoding is a special case where multiple categories are each binary encoded. Given we have k categories, this will create k extra features (thus increasing the dimensionality of the data). Another method for encoding is using a target and probability-based encoding. The category is the average value and includes a probability.

Methods of EDA

It is always better to explore each data set using multiple exploratory techniques and compare the results. Once the data set is fully understood, it is quite possible that data scientists will have to go back to data collection and cleansing phases in order to transform the data set according to the desired business outcomes. The goal of this step is to become confident that the data set is ready to be used in a machine learning algorithm.

EDA is majorly performed using the following methods:

- **Univariate visualization:** Provides summary statistics for each field in the raw data set.
- **Bivariate visualization:** It is performed to find the relationship between each variable in the dataset and the target variable of interest.
- **Multivariate visualization:** It is performed to understand the interactions between different fields in the dataset.
- **Dimensionality reduction:** It helps to understand the fields in the data that account for the most variance between observations and allow for the processing of a reduced volume of data.

Through these methods, the data scientist validates assumptions and identifies patterns that will allow for the understanding of the problem and

model selection and validates that the data has been generated in the way it was expected to. So, the value distribution of each field is checked, several missing values are defined, and the possible ways of replacing them are found.

Another side benefit of EDA is that it allows specifying or even defining the questions you are trying to get the answer to from your data. Companies that are only starting to leverage data science and AI technologies often face the situation when they realize that they have a lot of data and no ideas of what value that data can bring to their business decision making.

However, the questions always come first in data analysis. It doesn't matter how much data a company has, how many tools they have available, whether the data is historical or real-time unless business stakeholders have the questions they are trying to solve with their data. EDA can help such companies to start formalizing the right questions, since, with wrong questions, you get the wrong answers, and take the wrong decisions.

One of the important things about EDA is data profiling. Data profiling is concerned with summarizing your dataset through descriptive statistics. You want to use a variety of measurements to better understand your dataset. The goal of data profiling is to have a solid understanding of your data so you can afterward start querying and visualizing your data in various ways. However, this doesn't mean that you don't have to iterate: exactly because data profiling is concerned with summarizing your dataset, it is frequently used to assess the data quality. Depending on the result of the data profiling, you might decide to correct, discard, or handle your data differently.

Key concepts of EDA

- Two types of data analysis
 - Confirmatory Data Analysis
 - Exploratory Data Analysis
- Four objectives of EDA
 - Discover Patterns
 - Spot Anomalies
 - Frame Hypothesis

- Check Assumptions
- Two types of Exploration
 - Univariate Analysis
 - Bivariate Analysis
- Stuff has done during EDA
 - Trends
 - Distribution
 - Mean
 - Median
 - Outlier
 - Spread measurement (SD)
 - Correlations
 - Hypothesis testing
 - Visual Exploration

Let us now perform a similar approach to explore a dataset. Here, we will take a Kaggle problem (<https://www.kaggle.com>) to view insights into the data. The objective of the problem is to predict the salary of any UK job ad based on its contents (<https://www.kaggle.com/c/job-salary-prediction/overview>). We have a training data set on which we need to build a model, which includes various variables, including salary. A test data set and a validation data set are also available. We will prepare the model using Python libraries.

Initially, we will import the Python libraries, which will be used for mathematical computing, data manipulation, and data visualizations. The following code snippet shows how to import the required libraries to perform the corresponding operations.

```
import math
import numpy as np
import pandas as pd
import nltk
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

```
from IPython.display import display,HTML
from patsy import dmatrices
import warnings
warnings.filterwarnings("ignore")
%pylab inline
Populating the interactive namespace from numpy and matplotlib
Loading the dataset
```

Train dataset contains 244768 rows × 12 columns

- **Id**: A unique identifier for each job ad.
- **Title**: A freetext field supplied to us by the job advertiser as the Title of the job ad.
- **LocationRaw**: The freetext location as provided by the job advertiser.
- **LocationNormalized**: Adzuna's normalized location from within our own location tree, interpreted by us based on the raw location. This normalizer is not perfect.
- **ContractType**: `full_time` or `part_time`, interpreted by Adzuna from a description or a specific additional field we received from the advertiser.
- **ContractTime**: Permanent or contract, interpreted by Adzuna from a description or a specific additional field we received from the advertiser.
- **Company**: The name of the employer as supplied to us by the job advertiser.
- **Category**: Which of 30 standard job categories this ad fits into, inferred in a very messy way based on the source the ad came from. There is a lot of noise and error in this field.
- **SalaryRaw**: The freetext salary field we received in the job advert from the advertiser.
- **SalaryNormalised**: The annualized salary interpreted by Adzuna from the raw salary. Note that this is always a single value based on the midpoint of any range found in the raw salary. This is the value we are trying to predict.
- **SourceName**: The name of the website or advertiser from whom we received the job advert.

```
#Read Train data using Pandas library
data_train = pd.read_csv('Train_rev1.csv')

#Check the first 5 rows of the data set
data_train.head()
```

After executing this code, you will get output similar shown in [Figure 4.2](#). It displays the top 5 rows of the training dataset:

Id	Title	FullDescription	LocationRaw	LocationNormalized	ContractType	ContractTime	Company	Category	SalaryRaw	SalaryNormalized	SourceName
12612628	Engineering Systems Analyst	Engineering Systems Analyst Dorking Surrey Sal...	Dorking, Surrey, Surrey	Dorking	NaN	permanent	Gregory Martin International	Engineering Jobs	20000 - 30000/annum 20-30K	25000	cv-library.co.uk
12612830	Stress Engineer Glasgow	Stress Engineer Glasgow Salary **** to **** We...	Glasgow, Scotland, Scotland	Glasgow	NaN	permanent	Gregory Martin International	Engineering Jobs	25000 - 35000/annum 25-35K	30000	cv-library.co.uk
12612844	Modelling and simulation analyst	Mathematical Modeller / Simulation Analyst / O...	Hampshire, South East, South East	Hampshire	NaN	permanent	Gregory Martin International	Engineering Jobs	20000 - 40000/annum 20-40K	30000	cv-library.co.uk
12613049	Engineering Systems Analyst / Mathematical Mod...	Engineering Systems Analyst / Mathematical Mod...	Surrey, South East, South East	Surrey	NaN	permanent	Gregory Martin International	Engineering Jobs	25000 - 30000/annum 25K-30K negotiable	27500	cv-library.co.uk

Figure 4.2: Top 5 rows of the training dataset

Validation set: It is a sample of data used to provide an unbiased evaluation of a model fit on the training dataset.

```
#Read Validation data using Pandas library
data_val = pd.read_csv('Valid_rev1.csv')

#Check the first 5 rows of the data set
data_val.head()
```

After executing this code, you will get output similar shown in [Figure 4.3](#). It displays the top 5 rows of the validation dataset:

Id	Title	FullDescription	LocationRaw	LocationNormalized	ContractType	ContractTime	Company	Category	SourceName
0 13656201	Lead Technical Architect, C Banking	Lead Technical Architect required for a Tier *...	London	London	NaN	permanent	Scope AT Limited	IT Jobs	jobserve.com
1 14663195	RECRUITMENT CONSULTANT INDUSTRIAL / COMMERCIA...	THIS IS A LIVE VACANCY NOT A GENERIC ADVERTISE...	LEEDS, West Yorkshire	Leeds	NaN	permanent	Code Blue Recruitment	HR & Recruitment Jobs	cv-library.co.uk
2 16530664	Mechanical / Chemical / Process Engineer Cool...	Mechanical / Chemical / Process Engineer Cool...	Hampshire, South East	Hampshire	NaN	permanent	Gregory Martin International	Engineering Jobs	cv-library.co.uk
3 19047458	Trainee Mortgage Advisor West Midlands	Are you a successful, results driven person? A...	West Midlands	West Midlands	NaN	permanent	Brite Recruitment	Accounting & Finance Jobs	cv-library.co.uk
4 20881907	Mortgage Services Consultant East Midlands	Are you a successful, results driven person? D...	East Midlands	East Midlands	NaN	permanent	Brite Recruitment	Accounting & Finance Jobs	cv-library.co.uk

Figure 4.3: Top 5 rows of the validation dataset

The test data dataset contains 122463 rows \times 10 columns. It will be used to predict the SalaryRaw and SalaryNormalised. To view, the data details run the code given below. It will display the top 5 rows of the data.

```
#Read Test data using Pandas library
data_test = pd.read_csv('Test_rev1.csv')

#Check the first 5 rows of the data set
data_test.head()
```

After executing this code, you will get output similar shown in [Figure 4.4](#). It displays the top 5 rows of the test dataset:

	Id	Title	FullDescription	LocationRaw	LocationNormalized	ContractType	ContractTime	Company	Category	SourceName
0	11888454	Business Development Manager	The Company: Our client is a national training...	Tyne Wear, North East	Newcastle Upon Tyne	NaN	permanent	Asset Appointments	Teaching Jobs	cv-library.co.uk
1	11988350	Internal Account Manager	The Company: Founded in **** our client is a U...	Tyne and Wear, North East	Newcastle Upon Tyne	NaN	permanent	Asset Appointments	Consultancy Jobs	cv-library.co.uk
2	12612558	Engineering Systems Analysts	Engineering Systems Analysts Surrey ****K Loca...	Surrey, South East, South East	Surrey	NaN	permanent	Gregory Martin International	Engineering Jobs	cv-library.co.uk
3	12613014	CIS Systems Engineering Consultant	CIS Systems Engineering Consultant Bristol So...	Bristol, South West, South West	Bristol	NaN	permanent	Gregory Martin International	Engineering Jobs	cv-library.co.uk
4	22454872	CNC Miller / Programmer Fanac	CNC Miller / Programmer Fanac Fleet, Hampshire...	Fleet, Hampshire	Fleet	NaN	permanent	Gregory Martin International	Manufacturing Jobs	cv-library.co.uk

Figure 4.4: Top 5 rows of the test dataset

We can check the shape of all the three data sets using the below commands. [Figure 4.5](#) displays the shape of the training, validation, and testing datasets:

```
In [10]: data_train.shape
Out[10]: (122463, 10)

In [11]: data_val.shape
Out[11]: (40663, 10)

In [12]: data_test.shape
Out[12]: (122463, 10)
```

Figure 4.5: Checking the dimension of datasets

The training dataset can be described using the function describe(). [Figure 4.6](#) shows that descriptive statistical analysis data to understand the data furthermore:


```
In [16]: data_train.describe()
```

```
Out[16]:
```

	Id	SalaryNormalized
count	2.447680e+05	244768.000000
mean	6.970142e+07	34122.577576
std	3.129813e+06	17640.543124
min	1.261263e+07	5000.000000
25%	6.869550e+07	21500.000000
50%	6.993700e+07	30000.000000
75%	7.162606e+07	42500.000000
max	7.270524e+07	200000.000000

Figure 4.6: Statistical description of the training dataset.

This will be helpful when applying the statistical evaluation of data. Similarly, investigation of data can be done and then taken the decision based on the observation on the data.

Conclusion

This chapter provides the detailed introduction of **Exploratory Data Analysis (EDA)** with the needs of the EDA process in the data science process. Describes the various types of data that are used in real-world problems and how it can be represented in terms of mathematical values. This chapter also includes the methods of the EDA process and the important concepts necessary to understand the EDA process thoroughly. With a small EDA process example has been discussed. In the upcoming chapters, we will learn the concepts of data preprocessing, feature engineering, and machine learning model building.

CHAPTER 5

Data Preprocessing

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Whenever the data is gathered from different sources, it is collected in raw format, which is not feasible for the analysis. In [Chapter 4, Chapter 4: Exploratory Data Analysis](#), we discussed the **Exploratory Data Analysis (EDA)** that provides a better understanding of the data. For achieving good results from the applied model for machine learning, the format of the data must be in a proper manner. The collected data is said to be impure sometimes not valuable if it is missing properties corresponding to the data, values of properties, contain misleading or outliers, and identical or completely wrong data. The presence of any of these will degrade the quality of the results. To make the data useful next step is to process the data in an appropriate structure to adopt with our machine learning model. Data preprocessing is also known as a subset of data mining technique that involves transforming raw data into an understandable format. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepare raw data for further processing.

In this chapter, we will learn the fundamentals of data preprocessing and what are the methods used to preprocess the data. Also, we will discuss an example of the application of preprocessing. From this, the learner will have a better understanding of the underlying concepts of data preprocessing.

Structure

- Introduction to data preprocessing
- Methods of data preprocessing
- Application of preprocessing with example
- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of data preprocessing and the importance of machine learning model building.
- Understand the various methods involved in data preprocessing.
- Apply the data preprocessing steps in real-world datasets.

Introduction to data preprocessing

Data preprocessing is also an important step in machine learning because of the value of data and the beneficial data that can be resultant from it directly affects the capability of our model to learn; therefore, it is really important that we preprocess our data before sending it into our model. The following [Figure 5.1](#) shows the essential steps to be followed during the preprocessing of the data.

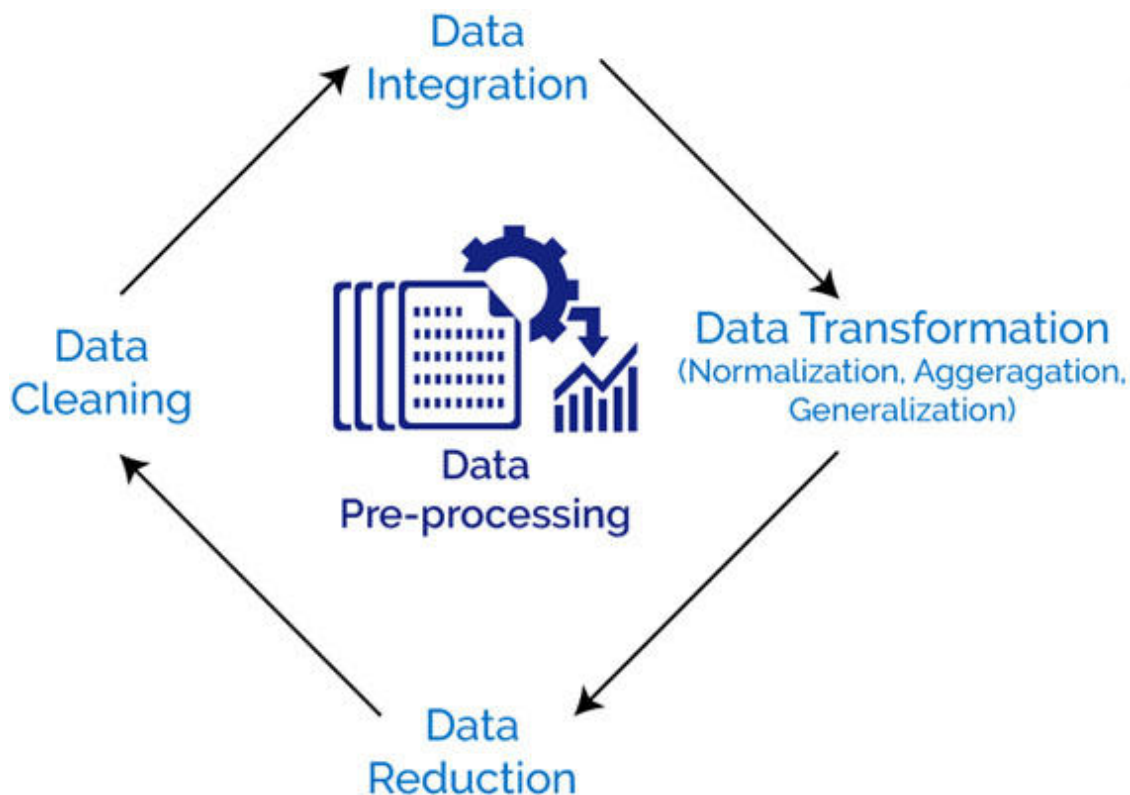


Figure 5.1: Data preprocessing lifecycle

Essential steps during data preprocessing:

- **Data cleaning:** Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
- **Data integration:** Data with different representations are put together, and conflicts within the data are resolved.
- **Data transformation:** Data is normalized, aggregated, and generalized.
- **Data reduction:** This step aims to present a reduced representation of the data in a data warehouse.

The above are the very important steps of the data preprocessing process. Let we will get into the methods that can be applied in data preprocessing.

Methods in data preprocessing

Data Preprocessing is a huge topic because the preprocessing techniques vary from data to data. Different kinds of data (images, text, sounds, videos, CSV files, and many more) have different methods for preprocessing, but there are some methods, which are common for almost any kind of data. The most important ones of these methods happen to be:

- Transformation into vectors
- Normalization
- Dealing with the missing values

Transformation into vectors

All the ML models need the input data to be in the form of vectors. If you got raw text data, you need some mechanism to convert those strings into some meaningful numerical representation, like tf-idf, word2vec, and many more. If you got images, they are processed as matrices of pixels, and if you got sounds, they need to be converted from analog waves to digital signals, if you got categorical data in a CSV file, you might want to apply label encoding or one-hot-encoding. You basically convert almost all your data into float (or in some cases, integers), so that your ML model can easily process all that. Every record (each sentence, if it's a text; soundwave, if it's a sound) represents a single row of your input, and for multiple records, you get your input matrix (generally denoted by X).

Normalization

It's highly recommended that your data is properly scaled, which means that your data should not have a huge deviation for every column (feature). If you have a column whose values are between 0–1 and you have another feature whose values are between 100–1000, then this difference of value ranges can cause large gradient updates by your optimizer, and your network/model might not converge. So, a good way will be to normalize your values, which are between 100–1000, scaling them between 0–1. Breaking into steps, the following points should be applied to get the maximum benefit out of normalization:

- **Smaller values:** Try to have all the values between 0 and 1, or -1 to 1.
- **Homogeneity:** All the columns should have values in roughly the same range.
- **Mean:** Normalize in a way that you have a mean of 0 for each column independently.
- **Standard deviation:** Normalize in a way that you have a standard deviation of 1 for each column independently.

From the normalization process, we get the properly structured features that are useful when building the machine learning model.

Dealing with the missing values

Your data might not always be the ideal one. Having missing values in a dataset is very common, and an effective way to handle missing values leads to a better model trained. One way is to replace all the missing values to be 0, provided that 0 doesn't already represent meaningful information in your data. If there are a lot of missing values in the data, and you replace them with 0, the model will eventually learn that all the 0s aren't playing any constructive role in the decision-making process of the model, and will pretty much ignore them by assigning them lower weights. If your data is consistent, especially in the case of time-series, or sequenced-based datasets, interpolation of the data becomes a meaningful option in there as well. Otherwise, mostly the missing values are replaced by mean or median values of the respective column they're a part of.

Let us continue the discussion of [chapter 4](#) example in that we already did the EDA process to understand the data and its different properties. Now we need to apply a few preprocessing techniques to get the appropriate structure of data. We will start by displaying the information about the dataset using the below commands.

```
data_train.info()  
data_val.info()  
data_test.info()
```

After running the codes, you will get results similar shown in the below figures. [Figure 5.2](#) shows the information related to the training dataset:

```
data_train.info()  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 244768 entries, 0 to 244767  
Data columns (total 12 columns):  
Id                244768 non-null int64  
Title            244767 non-null object  
FullDescription   244768 non-null object  
LocationRaw       244768 non-null object  
LocationNormalized 244768 non-null object  
ContractType      65442 non-null object  
ContractTime      180863 non-null object  
Company           212338 non-null object  
Category          244768 non-null object  
SalaryRaw         244768 non-null object  
SalaryNormalized  244768 non-null int64  
SourceName        244767 non-null object  
dtypes: int64(2), object(10)  
memory usage: 22.4+ MB
```

Figure 5.2: Training data information

Similarly, if we try to see the validation dataset information, we will get a similar output shown in [Figure 5.3](#):

```
data_val.info()  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 40663 entries, 0 to 40662  
Data columns (total 10 columns):  
Id                40663 non-null int64  
Title            40663 non-null object  
FullDescription   40663 non-null object  
LocationRaw       40663 non-null object  
LocationNormalized 40663 non-null object  
ContractType      10968 non-null object  
ContractTime      30181 non-null object  
Company           35312 non-null object  
Category          40663 non-null object  
SourceName        40663 non-null object  
dtypes: int64(1), object(9)  
memory usage: 3.1+ MB
```

Figure 5.3: Validation of data information

Understanding the test data is very important because it will give a better idea to see the particular dataset which we are going to approach. [Figure 5.4](#)

shows the resultant information related to the test dataset:

```
data_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122463 entries, 0 to 122462
Data columns (total 10 columns):
Id                122463 non-null int64
Title             122463 non-null object
FullDescription   122463 non-null object
LocationRaw       122463 non-null object
LocationNormalized 122463 non-null object
ContractType      33013 non-null object
ContractTime      90702 non-null object
Company           106202 non-null object
Category          122463 non-null object
SourceName        122463 non-null object
dtypes: int64(1), object(9)
memory usage: 9.3+ MB
```

Figure 5.4: Test data information

Based on the observations made on the above studies, now we are ready to apply the preprocessing on the data. Now let us load the libraries for the same. Then, we will start with cache the stop words and remove those words from the dataset.

```
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
stop_words = set(stopwords.words('english'))
from string import punctuation
from collections import Counter

cachedStopWords = stopwords.words("english") #cache stop words
to speed-up removing them.
wordset = set()
text = ' '.join(
    data_train['Title'].replace(r'^0-9a-zA-Z+', ' ', regex=True)
    .fillna('').str.lower()
)
data_train['Title'].replace(r'^0-9a-zA-Z+', ' ', regex=True).fillna('').str.lower().str.split().apply(wordset.update)
print(list(wordset)[1:100])
most_common_terms = Counter([w for w in text.split(' ') if w not in cachedStopWords]).most_common(50)
```


After running the above code, you will be able to see the output like the [Figure 5.5](#). Which is cleaned data from the English stop words? Only the keywords are displayed. This can be used further for preprocessing next stages:

```
cachedStopWords = stopwords.words("english") #cache stop words to speed-up removing them.
wordset = set()
text = ''
for i in range(0, len(data_train['Title'])):
    data_train['Title'].replace(r'^0-9a-zA-Z+', ' ', regex=True)
    .fillna('').str.lower()
)
data_train['Title'].replace(r'^0-9a-zA-Z+', ' ', regex=True).fillna('').str.lower().str.split().apply(wordset.update)
print(list(wordset)[1:100])
most_common_terms = Counter([w for w in text.split(' ') if w not in cachedStopWords]).most_common(50)
```

['pubaylesbury', 'ilkley', 'fuel', 'finite', 'mess', 'woolbeding', 'noninterventional', 'servicelayer', 'cutting', 'salesforcecom', 'nvqs', 'consultantc', 'substations', 'brightwellcumsotwell', 'undefended', 'northolt', 'artic', '0', 'casino', 'clear', 'jug', 'performer', 'spacejumping', 'researcherfrench', 'assistantcatering', 'sout', 'idm', 'trailing', 'vad', 'conferences', 'objectoriented', 'recognised', 'managerprofessional', 'processors', 'anaerobic', 'integratio', 'nontechnical', 'belton', 'rayleigh', 'vexpert', 'jack', 'lomond', 'capital', 'algal', '146', 'parts', 'aggregator', 'excellent', 'gi', 'rolevmware', 'purification', 'mary', 'proficient', 'gloustershire', 'onscreen', 'exrecruitment', 'biking', 'workspace', 'representative', 'compliance', 'managercredit', 'inside', 'deutschsprachiger', 'jq', 'cucm', 'brilliant', 'associate', 'hobbyist', 'buyergercery', 'rollout', 'threat', 'profilen', 'produ', 'cork', 'removal', 'managersheffield', 'sfdc', 'broughton', 'kinship', 'kyiv', 'microsoft', 'mecahic', 'yorksh', 'smd', 'snow', 'opportunitystore', 'minister', 'managersmall', 'canline', 'constrcution', 'universities', 'started', 'lime', 'pulse', 'esrc', 'tq', 'against', 'cascade', 'coshh']

Figure 5.5: Dataset after removing stop words

Next, we will do the identification of unique values from the each columns of dataset. For that we are going to assign the training dataset columns values to one variable names and applying the iteration process to get all the unique values:

```
names = data_train.columns.values
uniq_vals = {}
for name in names:
    uniq_vals[name] = data_train.loc[:,name].unique()
    print("Count of %s: %d" %(name, uniq_vals[name].shape[0]))
```

The following [Figure 5.6](#) displays the unique values for each column in the training dataset. It shows the count of the unique values:

```
names = data_train.columns.values
uniq_vals = {}
for name in names:
    uniq_vals[name] = data_train.loc[:,name].unique()
    print("Count of %s : %d" %(name, uniq_vals[name].shape[0]))
```

Count of Id : 244768
 Count of Title : 135436
 Count of FullDescription : 242138
 Count of LocationRaw : 20986
 Count of LocationNormalized : 2732
 Count of ContractType : 3
 Count of ContractTime : 3
 Count of Company : 20813
 Count of Category : 29
 Count of SalaryRaw : 97286
 Count of SalaryNormalized : 8454
 Count of SourceName : 168

Figure 5.6: Unique values for each column in training dataset

Further we are going to take the most common terms in the dataset and plot that in bar graph show in [Figure 5.6](#):

```
labels, values = zip(*most_common_terms)

indexes = np.arange(len(labels))
width = 0.7

plt.bar(indexes, values, width)
plt.xticks(indexes + width * 0.5, labels, rotation='vertical')
plt.show()
```

The following [Figure 5.7](#) shows the most common terms used in the dataset. This will be helpful to visualize the dataset further to understand better:



Figure 5.7: Most common terms

To understand the distribution of salaries based on the available trained data after preprocessing can be visualized using the below codes. [Figure 5.8](#) shows the output of the code:

```
# Distribution of salaries based on the train data
pylab.rcParams['figure.figsize'] = (20,10)
plt.hist(data_train['SalaryNormalized'], bins='auto')
plt.xlabel('Salaries')
plt.ylabel('Number of postings')
plt.title('Histogram of Salaries')
```

The following [Figure 5.8](#) displays the information on salary distribution from the training data:

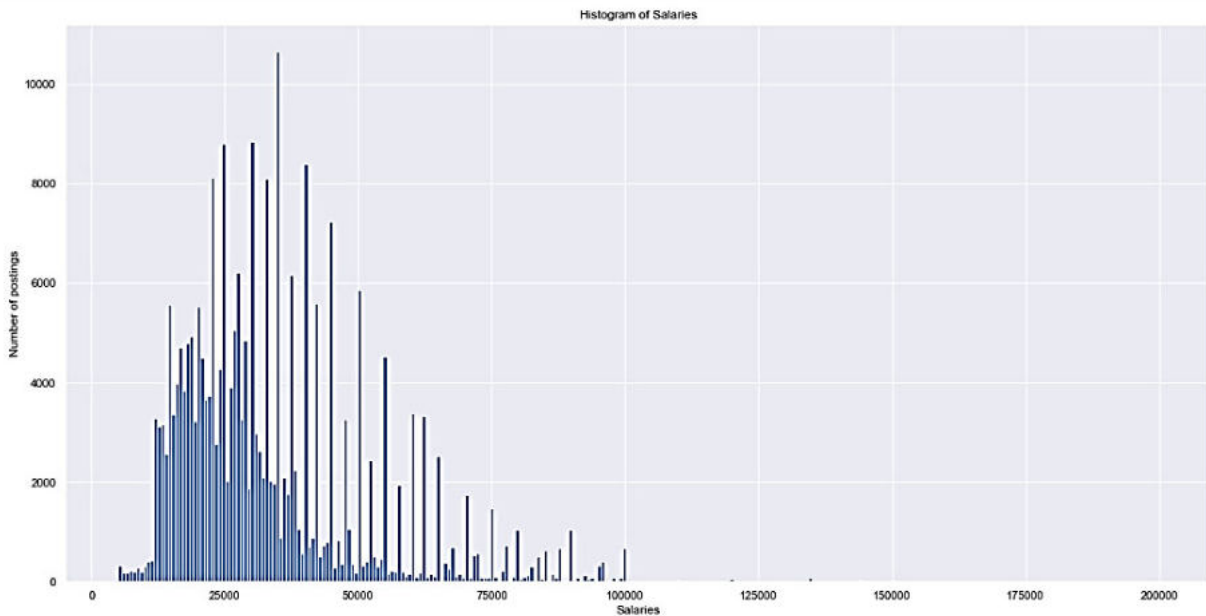


Figure 5.8: Distribution of salaries based on the train data

Once the preprocessing is completed, we can choose some of the sample data randomly to train the machine learning model. For example, we can select random 2500 rows to train the classifier by using the below codes:

```
import random
random.seed(1)
indices = list(data_train.index.values)
random_2500 = random.sample(indices, 2500)

# Subsetting the train data based on the random indices
data_train1 = data_train.loc[random_2500].reset_index()
pylab.rcParams['figure.figsize'] = (20, 10)
plt.hist(data_train1['SalaryNormalized'], bins='auto')
plt.xlabel('Salaries')
plt.ylabel('Number of postings')
plt.title('Histogram of Salaries')
```

[Figure 5.9](#) displays the dataset which randomly selected after applying the preprocessing on the training dataset:

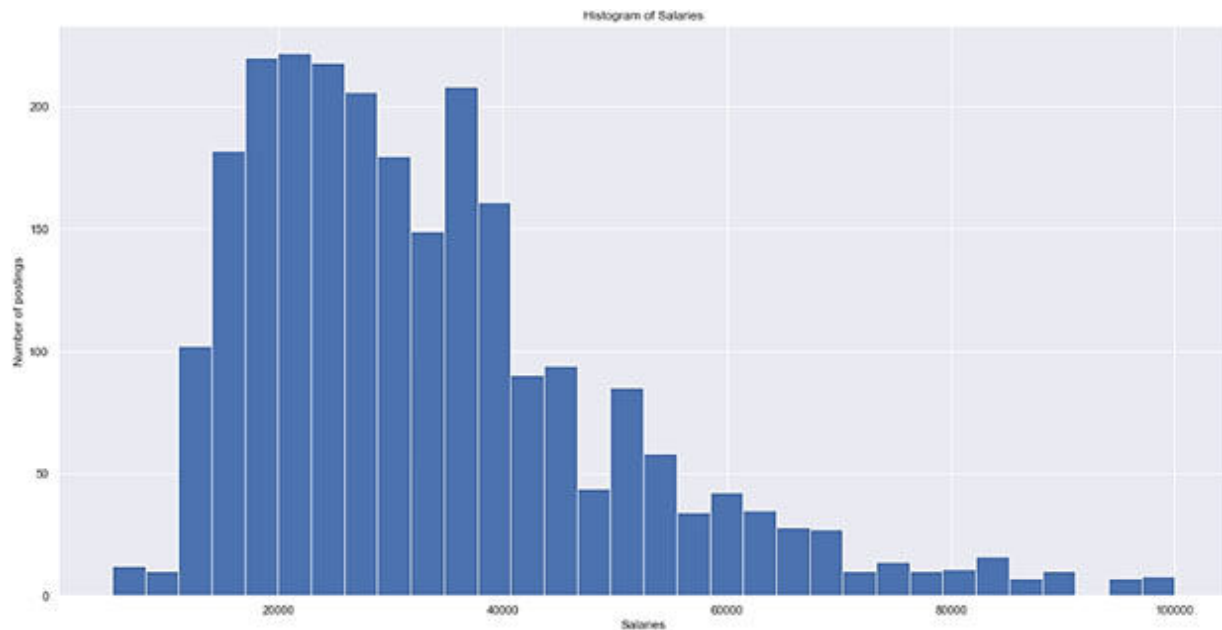


Figure 5.9: Randomly selected data to train the classifier

The above figure shows that when salary getting more the no of postings going down and also, salaries between 20000 to 40000 has the major impact in training process.

```
# To obtain the full width of a cell in a dataframe
pd.set_option('display.max_colwidth', -1)
desc = data_train1.loc[1, 'FullDescription']

# Creating a list of words from all the job descriptions in
train_df1 data
all_desc = []
for i in range(0, data_train1.shape[0]):
    desc = data_train1.loc[i, 'FullDescription']
    desc1 = desc.lower()

# Removing numbers, *** and www links from the data
desc2 = re.sub('[0-9]+\S+|\s\d+\s|\w+[0-9]+\w+|\w+
[\*]+\.\s[\*]+\s|www\.',
,→ [^\s]+'', desc1)

# Removing punctuation
for p in punctuation:
    desc2 = desc2.replace(p, '')
all_desc.append(desc2)
```

```
# Creating word tokens for all the descriptions
final_list = []
for desc in all_desc:
word_list = word_tokenize(desc)
final_list.extend(word_list)

# 3. Tagging parts of speech
pos_tagged = nltk.pos_tag(final_list)

# 4. Identifying the most common parts of speech
tag_fd = nltk.FreqDist(tag for (word, tag) in pos_tagged)
tag_fd.most_common()[:5]

Output: [('NN', 139026), ('JJ', 63231), ('IN', 57908), ('DT', 45695), ('NNS', 45681)]
```

In the next chapter, we will continue this program and show the methods of feature engineering.

Conclusion

In this chapter, we have discussed data preprocessing. It is one of the key processes in the machine learning model building. Understanding the key concepts and the techniques underlying in preprocessing will gives more confidence and clarity on the model building process. In this chapter, we have covered the basic concepts of data preprocessing with its methods. Various methods involved in preprocessing have been discussed, and finally, we demonstrated the few steps of data preprocessing and how we can approach data to get ready for the next stage.

By reading this chapter, the reader will be able to understand the basics and methods of data preprocessing. He or she will be able to work on a dataset and apply these methods. In the next chapter, we will introduce a very important topic feature engineering, which plays a vital role before going to the model building process.

CHAPTER 6

Feature Engineering

Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering.

- Andrew Ng

Feature engineering is a vital step in the machine-learning pipeline, yet this topic is rarely examined on its own. With this practical book, you will learn techniques for extracting and transforming feature—the numeric representations of raw data—into formats for machine-learning models. Each chapter guides you through a single data problem, such as how to represent text or image data. Together, these examples illustrate the main principles of feature engineering.

Machine learning fits mathematical models to data in order to derive insights or make predictions. These models take features as input. A feature is a numeric representation of an aspect of raw data. Features sit between data and models in the machine learning pipeline. Feature engineering is the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model.

Nevertheless, feature engineering is not just an ad hoc practice. There are deeper principles at work, and they are best illustrated in it. In this chapter, we introduce the feature engineering concepts to the learner. By explaining the techniques of feature engineering, the learner will have the ability to use the techniques with data. We are going to see the example application of the feature engineering process with the sample data. This will give a better perspective about the feature engineering process in real data.

Structure

- Introduction to feature engineering
- Feature engineering techniques
- Application of feature engineering with example

- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of the feature engineering process and the importance of machine learning model building.
- Understand the various techniques that can be used in the feature engineering process.
- Apply the feature engineering steps in real-world datasets.

Introduction to feature engineering

A feature is a numeric representation of raw data. A feature is also a measurable property of the object you're trying to analyze. In datasets, features appear as columns, as shown in [Figure 6.1](#):

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. t male		22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mr. female		38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, M female		26	0	0	STON/O2. 31	7.925		S
5	4	1	1	Futrelle, Mrs female		35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. W male		35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. J male			0	0	330877	8.4583		Q

Figure 6.1: Sample dataset

[Figure 6.1](#) contains a snippet of data from a sample dataset with information about the passengers on the ill-fated Titanic maiden voyage. Each feature, or column, represents a measurable piece of data that can be used for analysis: **Name**, **Age**, **Sex**, **Fare**, and so on. Features are also sometimes referred to as variables or attributes. Depending on what you're trying to analyze, the features you include in your dataset can vary widely.

There are many ways to turn raw data into numeric measurements, which is why features can end up looking like a lot of things. Naturally, features must derive from the type of data that is available. Perhaps less obvious is the fact that they are also tied to the model; some models are more appropriate for some types of features and vice versa. The right features are relevant to the task at hand and should be easy for the model to ingest. Feature engineering is the process of formulating the most appropriate features given the data, the model, and the task.

The number of features is also important. If there are not enough informative features, then the model will be unable to perform the ultimate task. If there are

too many features, or if most of them are irrelevant, then the model will be more expensive and trickier to train. Something might go awry in the training process that impacts the model's performance.

Importance of feature variable

Features are the basic building blocks of datasets. The quality of the features in your dataset has a major impact on the quality of the insights you will gain when you use that dataset for machine learning. Additionally, different business problems within the same industry do not necessarily require the same features, which is why it is important to have a strong understanding of the business goals of your data science project.

You can improve the quality of your dataset's features with processes like feature selection and feature engineering, which are notoriously difficult and tedious. If these techniques are done well, the resulting optimal dataset will contain all the essential features that might have a bearing on your specific business problem, leading to the best possible model outcomes and the most beneficial insights.

Feature engineering in machine learning

Features and models sit between raw data and the desired insights (see [Figure 6.2](#)). In a machine learning workflow, we pick not only the model but also the features. This is a double-jointed lever, and the choice of one affects the other. Good features make the subsequent modeling step easy, and the resulting model more capable of completing the desired task. Bad features may require a much more complicated model to achieve the same level of performance.

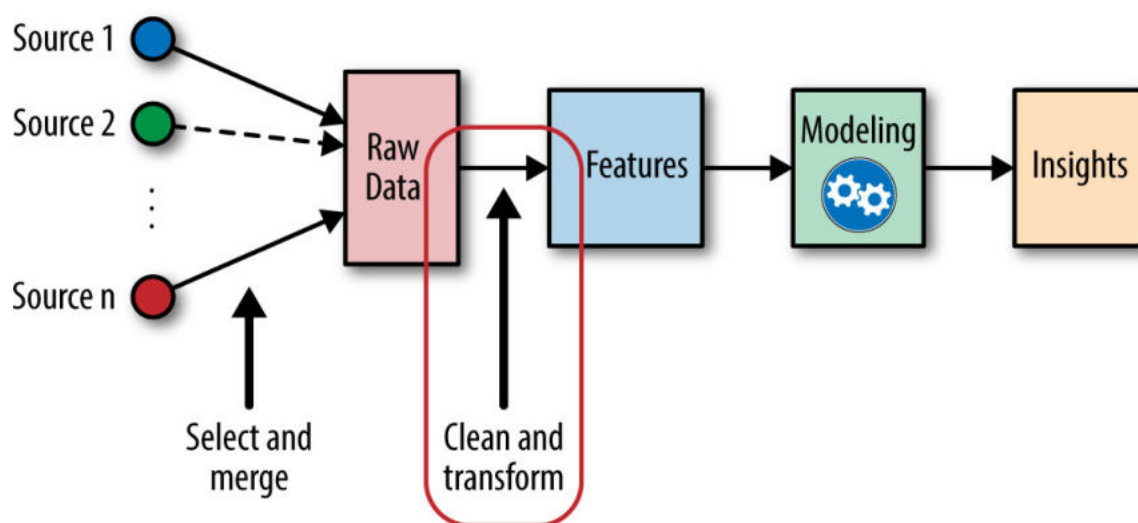


Figure 6.2: Place of feature engineering in the machine learning workflow

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. Machine learning algorithms will consume input data and produces the results. In this process, the input data contains features, which are represented in the form of a structured column. Features selected to use in algorithm development should have some specific requirements to work the algorithm efficiently. Here, the essential for feature engineering arises. Feature engineering efforts mainly have two goals:

- Input data preparation that is suitable for the algorithm requirements.
- Improving the efficiency of models built by machine learning algorithms.

These metrics are useful to show the essence of feature engineering in data science. In the next section, we have concise the main techniques involved in the feature engineering process with their brief descriptions.

Feature engineering techniques

Feature engineering has several techniques, but in this section, we covered major techniques used in the machine learning model building. The following are the important feature engineering techniques mostly used in the process of getting better features for machine learning algorithms.

- Imputation
- Handling outliers
- Binning
- Log transform
- One-hot encoding
- Grouping operations
- Feature split
- Scaling
- Extracting date

Let we will discuss each of these techniques in brief.

Imputation

Sometimes while collecting data from different sources, data will have some missing values that are one of the most common problems you can come across when we try to prepare the data for machine learning. This error may happen

because of wrong entry by humans, conversion data may miss some information, some of the privacy data will not be given for the public access, and so on. These missing values in data will affect the performance of the machine learning models. Some advanced machine learning platforms will try to drop the rows where the missing values present in data but that too will cause some performance decrease because of the size of data further reduced automatically. Few algorithms straightaway reject the dataset, which has the missing values.

In an optimal way, if the missing value goes beyond some values, we may try to drop the rows. If the data has 70% as an example missing value and try to drop the rows and columns which have missing values with higher than this value. The below pseudocode shows how to implement that in a programming way:

```
maxvalue = 0.7
#Dropping columns with missing value rate higher than maxvalue
data = data[data.columns[data.isnull().mean() < maxvalue]]
data = data.loc[data.isnull().mean(axis=1) < maxvalue]
```

An imputation is a preferable option rather than dropping because it preserves the data size. However, there is an important selection of what you impute to the missing values. I suggest beginning with considering a possible default value of missing values in the column. For example, if you have a column that only has 1 and NA, then it is likely that the NA rows correspond to 0. For another example, if you have a column that shows the *customer visit count in last month*, the missing values might be replaced with 0 if you think it is a sensible solution. Another reason for the missing values is joining tables with different sizes, and in this case, imputing 0 might be reasonable as well. Except for the case of having a default value for missing values, I think the best imputation way is to use the medians of the columns:

```
#Filling all missing values with 0
data = data.fillna(0).
#Filling missing values with medians of the columns
data = data.fillna(data.median())
```

Replacing the missing values with the maximum occurred value in a column is a good option for handling categorical columns. But if you think the values in the column are distributed uniformly, and there is not a dominant value, imputing a category like Other might be more sensible, because in such a case, your imputation is likely to converge a random selection.

```
#Max fill function for categorical columns
```

```
data['column_name'].fillna(data['column_name'].value_counts().idxmax(), inplace=True).
```

From the above approaches, we can avoid the dataset contain missing values, and we can use the imputation is one the process of basic feature engineering techniques.

Handling outliers

Before mentioning how outliers can be handled, I want to state that the best way to detect the outliers is to demonstrate the data visually. All other statistical methodologies are open to making mistakes, whereas visualizing the outliers gives a chance to make a decision with high precision. Anyway, I am planning to focus on visualization deeply in another article, and let's continue with statistical methodologies. Statistical methodologies are less precise, as I mentioned, but on the other hand, they have superiority; they are fast. Here I will list two different ways of handling outliers. These will detect them using standard deviation and percentiles. Outlier detection with *standard deviation*. If a value has a distance to the average higher than $x * \text{standard deviation}$, it can be assumed as an outlier. Then what x should be? There is no trivial solution for x , but usually, a value between 2 and 4 seems practical:

```
#Dropping the outlier rows with standard deviation
factor = 3
upper_lim = data['column'].mean () + data['column'].std () * factor
lower_lim = data['column'].mean () - data['column'].std () * factor
data = data[(data['column'] < upper_lim) & (data['column'] > lower_lim)]
```

In addition, Z-score can be used instead of the formula above. Z-score (or standard score) standardizes the distance between a value and the mean using the standard deviation. Outlier detection with percentiles, another mathematical method to detect outliers is to use percentiles. You can assume a certain percentage of the value from the top or the bottom as an outlier. The key point is here to set the percentage value once again, and this depends on the distribution of your data, as mentioned earlier. Additionally, a common mistake is using the percentiles according to the range of the data. In other words, if your data ranges from 0 to 100, your top 5% is not the values between 96 and 100. Top 5% means here the values that are out of the 95th percentile of data:

```
#Dropping the outlier rows with Percentiles
upper_lim = data['column'].quantile(.95)
```

```
lower_lim = data['column'].quantile(.05)
data = data[(data['column'] < upper_lim) & (data['column'] >
lower_lim)]
```

An outlier dilemma: Drop or cap another option for handling outliers is to cap them instead of dropping. So, you can keep your data size, and at the end of the day, it might be better for the final model performance. On the other hand, capping can affect the distribution of the data, thus it better not to exaggerate it.

```
#Capping the outlier rows with Percentiles
upper_lim = data['column'].quantile(.95)
lower_lim = data['column'].quantile(.05)
data.loc[(df[column] > upper_lim),column] = upper_lim
data.loc[(df[column] < lower_lim),column] = lower_lim
```

These are the steps that can apply while processing the data with outliers. Let us discuss the binning process now.

Binning

The main motivation of binning is to make the model more robust and prevent overfitting. However, it has a cost to the performance. Every time you bin something, you sacrifice information and make your data moreregularized.

Binning can be applied on both categorical and numerical data:

```
#Numerical Binning Example
Value Bin
0-30 -> Low
31-70 -> Mid
71-100 -> High
```

[Figure 6.3](#) illustrates the binning process applied to the numerical data:

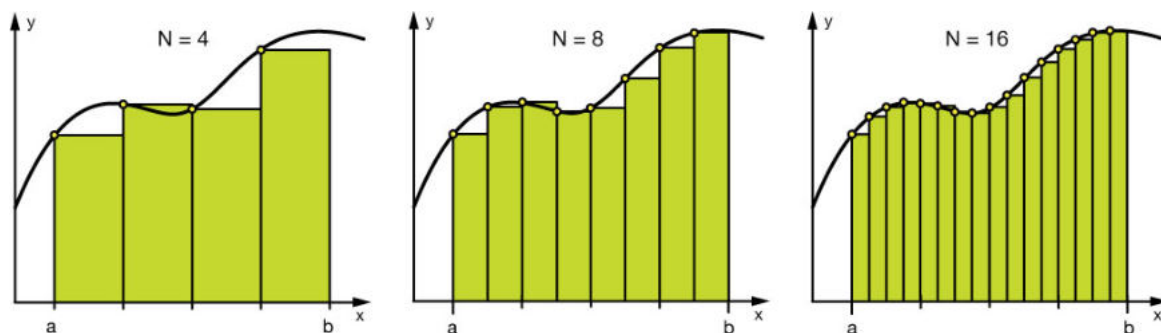


Figure 6.3: Binning illustration of numerical data

```
#Categorical Binning Example
```

```
Value Bin
```

```
Spain -> Europe
```

```
Italy -> Europe
```

```
Chile -> South America Brazil -> South America
```

The trade-off between performance and overfitting is the key point of the binning process. In my opinion, for numerical columns, except for some obvious overfitting cases, binning might be redundant for algorithms, due to its effect on model performance.

However, for categorical columns, the labels with low frequencies probably affect the robustness of statistical models negatively. Thus, assigning a general category to these less frequent values helps to keep the robustness of the model. For example, if your data size is 100,000 rows, it might be a good option to unite the labels with a count less than 100 to a new category like *Other*.

```
#Numerical Binning Example
```

```
data['bin'] = pd.cut(data['value'], bins=[0,30,70,100], labels=["Low",  
"Mid", "High"])
```

```
value bin
```

```
0      2      Low
```

```
1     45     Mid
```

```
2      7      Low
```

```
3     85     High
```

```
4     28      Low
```

```
#Categorical Binning Example
```

```

Country
0      Spain
1      Chile
2      Australia
3      Italy
4      Brazil

conditions = [ data['Country'].str.contains('Spain'),
data['Country'].str.contains('Italy'),
data['Country'].str.contains('Chile'),
data['Country'].str.contains('Brazil')]
choices = ['Europe', 'Europe', 'South America', 'South America']
data['Continent'] = np.select(conditions, choices, default='Other')

    Country      Continent
0      Spain  Europe
1      Chile  South America
2      Australia    Other
3      Italy  Europe
4      Brazil    South America

```

These are some of the examples of the binning process. We will discuss the next technique, that is, log transform.

Log Transform

Log Transform logarithm transformation (or log transform) is one of the most commonly used mathematical transformations in feature engineering.

Below are the benefits of Log Transform.

- It helps to handle skewed data, and after transformation, the distribution becomes more approximate to normal.
- In most of the cases, the magnitude order of the data changes within the range of the data. For instance, the difference between the ages of 15 and 20 is not equal to the ages 65 and 70. In terms of years, yes, they are identical, but for all other aspects, 5 years of difference in young ages mean a higher magnitude difference. This type of data comes from a multiplicative process, and log transform normalizes the magnitude differences like that.
- It also decreases the effect of the outliers due to the normalization of magnitude differences, and the model becomes more robust.

A critical note: The data you apply for log transform must have only positive values; otherwise, you receive an error. Also, you can add 1 to your data before transforming it. Thus, you ensure the output of the transformation to be positive.

```
Log(x+1)
data = pd.DataFrame({'value':[2,45, -23, 85, 28, 2, 35, -12]})
data['log+1'] = (data['value']+1).transform(np.log).
```

Note that the values are different:

```
data['log'] = (data['value']-data['value'].min()+1) .transform(np.log)
```

	value	log(x+1)	log(x-min(x)+1)
0	2	1.09861	3.25810
1	45	3.82864	4.23411
2	-23	nan	0.00000
3	85	4.45435	4.69135
4	28	3.36730	3.95124
5	2	1.09861	3.25810
6	35	3.58352	4.07754
7	-12	nan	2.48491


This is the output of data after applying the log transformation. Let us discuss the next technique that is, one-hot encoding.

One-hot encoding

One-hot encoding is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assigns 0 or 1 to them. These binary values express the relationship between grouped and encoded column.

This method changes your categorical data, which is challenging to understand for algorithms, to a numerical format, and enables you to group your categorical data without losing any information. [Figure 6.4](#) shows the one-hot encoding process applied to the User and City dataset:

User	City
1	Roma
2	Madrid
1	Madrid
3	Istanbul
2	Istanbul
1	Istanbul
1	Roma



User	Istanbul	Madrid
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

Figure 6.4: One hot encoding example in City column

After the application one-hot encoding, the dataset dimension has been changed.

Grouping operations

In most machine learning algorithms, every instance is represented by a row in the training dataset, where every column shows a different feature of the instance. Datasets such as transactions rarely fit the definition of tidy data above, because of the multiple rows of an instance. In such a case, we group the data by the instances, and then every instance is represented by only one row.

The key point of the group by operations is to decide the aggregation functions of the features. For numerical features, average and sum functions are usually convenient options, whereas for categorical features, it more complicated.

Categorical column grouping

Different ways for aggregating categorical columns:

- The first option is to select the label with the highest frequency. In other words, this is the max operation for categorical columns, but ordinary max functions generally do not return this value, you need to use a lambda function for this purpose:


```
data.groupby('id').agg(lambda x: x.value_counts().index[0])
```

- The second option is to make a pivot table. This approach resembles the encoding method in the preceding step with a difference. Instead of binary notation, it can be defined as aggregated functions for the values between grouped and encoded columns. This would be a good option if you aim to go beyond binary flag columns and merge multiple features into aggregated features, which are more informative.

```
#Pivot table Pandas Example
data.pivot_table(index='column_to_group',
columns='column_to_encode', values='aggregation_column',
aggfunc=np.sum, fill_value = 0)
```

[Figure 6.5](#) shows the output of one-hot encoding after applying the pivot table technique:

User	City	Visit Days
1	Roma	1
2	Madrid	2
1	Madrid	1
3	Istanbul	1
2	Istanbul	4
1	Istanbul	3
1	Roma	3



User	Istanbul	Madrid	Roma
1	3	1	4
2	4	2	0
3	1	0	0

Figure 6.5: Pivot table example: Sum of Visit Days grouped by Users

- The last categorical grouping option is to apply a group by function after applying one-hot encoding. This method preserves all the data -in the first option, you lose some, and in addition, you transform the encoded column from categorical to numerical in the meantime. You can check the next section for the explanation of numerical column grouping.

Numerical column grouping

Numerical columns are grouped using sum and mean functions in most of the cases. Both can be preferable according to the meaning of the feature. For example, if you want to obtain ratio columns, you can use the average of binary columns. In the same example, sum function can be used to obtain the total count either.

```
#sum_cols: List of columns to sum
#mean_cols: List of columns to average
grouped = data.groupby('column_to_group')
sums = grouped[sum_cols].sum().add_suffix('_sum')
avgs = grouped[mean_cols].mean().add_suffix('_avg')
new_df = pd.concat([sums, avgs], axis=1)
```

The above commands are used to apply the numerical column grouping method. Let us now discuss the feature split technique.

Feature split

Splitting features is a good way to make them useful in terms of machine learning. Most of the time, the dataset contains string columns that violate tidy data principles. By extracting the utilizable parts of a column into new features:

- We enable machine learning algorithms to comprehend them.
- Make it possible to bin and group them.
- Improve model performance by uncovering potential information.

A split function is a good option; however, there is no one way of splitting features. It depends on the characteristics of the column, how to split it. Let's introduce it with two examples. First, a simple split function for an ordinary name column:

```
data.name
0      Luther N. Gonzalez
1      Charles M. Young
2           Terry Lawson
3           Kristen White
4           Thomas Logsdon
#Extracting first names
data.name.str.split(" ").map(lambda x: x[0])
0      Luther
1      Charles
2           Terry
3      Kristen
4      Thomas
```

The example above handles the names longer than two words by taking only the first and last elements, and it makes the function robust for corner cases, which should be regarded when manipulating strings like that. Another case for the split function is to extract a string part between two chars. The following example shows an implementation of this case by using two split functions in a row:

```
#String extraction example
data.title.head()
0      Toy Story (1995)
1      Jumanji (1995)
2      Grumpier Old Men (1995)
3      Waiting to Exhale (1995)
4      Father of the Bride Part II (1995)
data.title.str.split("(", n=1, expand=True)[1].str.split(")", n=1, ex-
pand=True)[0]
0      1995
1      1995
2      1995
3      1995
4      1995
```

These are some of the examples to show how to use feature split in the data. Next, we will explore the scaling technique.

Scaling

In most cases, the numerical features of the dataset do not have a certain range, and they differ from each other. In real life, it is nonsense to expect age and income columns to have the same range. But from the machine learning point of view, how these two columns can be compared?

Scaling solves this problem. The continuous features become identical in terms of the range, after a scaling process. This process is not mandatory for many algorithms, but it might still be nice to apply. However, the algorithms based on distance calculations such as k-NN or k-Means need to have scaled continuous features as model input.

Basically, there are two common ways of scaling:

Normalization:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization (or min-max normalization) scale all values in a fixed range between 0 and 1. This transformation does not change the distribution of the feature and due to the decreased standard deviations, the effects of the outliers

increases. Therefore, before normalization, it is recommended to handle the outliers.

```
data = pd.DataFrame({'value':[2,45, -23, 85, 28, 2, 35, -12]})
data['normalized'] = (data['value'] - data['value'].min()) / (data['value'].max() - data['value'].min())
```

	value	normalized
0	2	0.23
1	45	0.63
2	-23	0.00
3	85	1.00
4	28	0.47
5	2	0.23
6	35	0.54
7	-12	0.10

These examples show how to apply scaling technique to normalize the data. Let us now discuss the extracting date method.

Extracting date

Though date columns usually provide valuable information about the model target, they are neglected as an input or used nonsensically for the machine learning algorithms. It might be the reason for this, that dates can be present in numerous formats, which make it hard to understand by algorithms even they are simplified to a format like 01-01-2017. Building an ordinal relationship between the values is very challenging for a machine learning algorithm if you leave the date columns without manipulation. Here, I suggest three types of preprocessing for dates:

- Extracting the parts of the date into different columns: Year, month, day, and many more.
- Extracting the time period between the current date and columns in terms of years, months, days, and more.
- Extracting some specific features from the date: Name of the weekday, Weekend or not, holiday or not, and many more.

If you transform the date column into the extracted columns like above, the information of them become disclosed and machine learning algorithms can easily understand them:

```

from datetime import date
data = pd.DataFrame({'date':
['01-01-2017',
'04-12-2008',
'23-06-1988',
'25-08-1999',
'20-02-1993',
]})

#Transform string to date
data['date'] = pd.to_datetime(data.date, format="%d-%m-%Y")

#Extracting Year
data['year'] = data['date'].dt.year

#Extracting Month
data['month'] = data['date'].dt.month

#Extracting passed years since the date
data['passed_years'] = date.today().year - data['date'].dt.year

#Extracting passed months since the date
data['passed_months'] = (date.today().year - data['date'].dt.year) *
12 + date.today().month - data['date'].dt.month

#Extracting the weekday name of the date
data['day_name'] = data['date'].dt.day_name()

```

	date	year	month	passed_years	passed_months	day_name	
0	2017-01-01	2017		1	2	26	Sunday
1	2008-12-04	2008		12	11	123	Thursday
2	1988-06-23	1988		6	31	369	Thursday
3	1999-08-25	1999		8	20	235	Wednesday
4	1993-02-20	1993		2	26	313	Saturday

With this the major techniques used in the feature engineering has been explained. Now let us discuss the application of feature engineering with the example.

[Applying feature engineering](#)

From [Chapter 5, Data Preprocessing](#), we will continue to apply the feature engineering process to the dataset, which we are already using in our examples.

We will start the application of identifying the most common parts of speech in the dataset for that we have to run the following code:

```
# Excluding stopwords from the analysis
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

list_wo_stopwords = []
for w in final_list:
    if w not in stop_words:
        list_wo_stopwords.append(w)

# 3. Tagging parts of speech
pos_tagged_wo_sw = nltk.pos_tag(list_wo_stopwords)

# 4. Identifying the most common parts of speech
tag_fd_wo_sw = nltk.FreqDist(tag for (word, tag) in
pos_tagged_wo_sw)
tag_fd_wo_sw.most_common()[:5]
```

The output would be like this shown below:

```
[('NN', 134734), ('JJ', 65288), ('NNS', 44913), ('VBG', 22828),
('VBP', 15395)]
```

We can perform the normalization operation to get the normalized values for the salary. For that we have to use the below code:

```
p_75 = np.percentile(data_train1['SalaryNormalized'], 75)
data_train1['target'] = data_train1['SalaryNormalized'].apply(lambda
x: 1 if x>=p_75 else 0)
costly_cities =
['London', 'Brighton', 'Edinburgh', 'Bristol', 'Southampton', 'Portsmouth',
', 'Exeter', 'Cardiff', 'Manchester',
'Birmingham', 'Leeds', 'Aberdeen', 'Glasgow', 'Newcastle', 'Sheffield', 'L
iverpool']
costly_cities_lower = [x.lower() for x in costly_cities]

# More robust if lower() is applied
data_train1['location_flag'] =
data_train1['LocationNormalized'].apply(lambda x: 1 if x in
costly_cities else 0)
```

The dataset has extra column job description we have to remove that to get better result:

```
# Dropping job description column from the dataset
data_train2 =
data_train1.drop(['FullDescription','index','Id','LocationRaw','Title',
'Company','LocationNormalized','SalaryRaw','SalaryNormalized',
'target'],axis=1)

data_train3 = pd.get_dummies(data_train2,drop_first=True)
X_n = np.array(data_train3)
y_n = np.array(data_train1['target'])
```

After performing the above operations our dataset will be good to supply for the next step of machine learning model. That is, machine learning model selection and fitting the data with that.

Conclusion

In this chapter, we have learned the fundamental methods that can be beneficial in the feature engineering process. This will be most important to know all those techniques when we go for real-time problems. Most of the techniques applied in different scenarios with different levels of applications. Once the dataset is prepared with an appropriate feature engineering process, then it will be easier to get better results while building our machine learning models. We have discussed the application of feature engineering using examples. By reading this chapter, the reader will be able to gain the basic understanding of feature engineering and its techniques. He or she will be able to apply these techniques in datasets. In the next chapter, we will discuss about the machine learning algorithms. We will also build a model using one of the algorithms.

CHAPTER 7

Machine Learning Algorithms

From the previous chapter, we learned the important steps that need to apply to the data before building machine learning models. **Machine learning (ML)** is a subset of **Artificial intelligence (AI)** algorithms. It makes the software applications more efficient and accurate in processing the historical data to predict the outcomes of future data. A notable point is machine learning algorithms try to get the output without explicitly programmed. The flow of machine learning is to build the algorithms with the input of data from the past and near future and use the standard statistical analysis methods to find the new outcomes whenever the new data becomes comes into the algorithm. Increasing the input data exposure to the algorithms during the training phase will make the algorithm performance better. This process of processing the input data is also known as the learning of the machine learning algorithm. This means that the algorithm will try to change and adopt the results based on the data it consumes over time. It makes the resemblance of human learning process by the experience. Machine learning algorithms always tries to fine-tune its parameters according to the learning experience gained from the dataset. Also, it considers the feedback from the previous output and improves it. So, it is important to know the concepts behind the machine learning algorithms and different types of machine learning algorithms. In this chapter, we try to cover those basic machine learning concepts and techniques in detail.

Structure

- Introduction to machine learning
- Explaining important machine learning algorithms
- Building a machine learning model with an example
- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the fundamentals of machine learning and the evolution of machine learning.
- Understand the different types of machine learning algorithms.
- Build the machine learning models for the dataset.

Introduction to machine learning

Machine learning follows the concept of making the machine to learn based on its previous experience and examples, but everything happens without being explicitly programmed. Machine learning creates a major impact on the industry because of its ability to produce output more accurately for future decision making from the data it observed. The current trend in computation power has improved from the past few years will also make the implementation of the machine learning process a better one. Machine learning is applied in various domains and industries. Let we will see some of the areas where this produces more effective results.

Application areas of machine learning are as follows:

- **Prediction problems:** Machine learning can also be useful in prediction-based systems. Let us consider the bank loan processing example to predict the potential customer who may want a home loan in the future can be classified by machine learning algorithms.
- **Image recognition:** Facial recognition application is the best example to tell the usage of machine learning. It tries to separate the people based on their face data.
- **Speech recognition:** Using the audio data detection of the language. It is also useful to translate the language into another language. Translating the human voice data to textual format to make documents.
- **Medical domain:** Machine learning is used to recognize tumorous tissues.
- **Financial industry and trading:** This company uses machine learning to investigate fraud and checks the credit card activities to find defaulters.

Let us start our voyage with a brief history of machine learning and then will discuss the different types of machine learning. Finally, we will cover the details about the machine learning model building steps with an example.

Brief history of machine learning

Machine learning is not a very new concept in the world. It has existed before six decades itself. From [Figure 7.1](#), we get the idea of machine learning starts from the 1940s itself. This is the period when the first computer system started to operate. At that time, that computer is called a numerical computing machine. Because the word *computer* was being used in the form of a human with the additional capability to perform numerical computation:

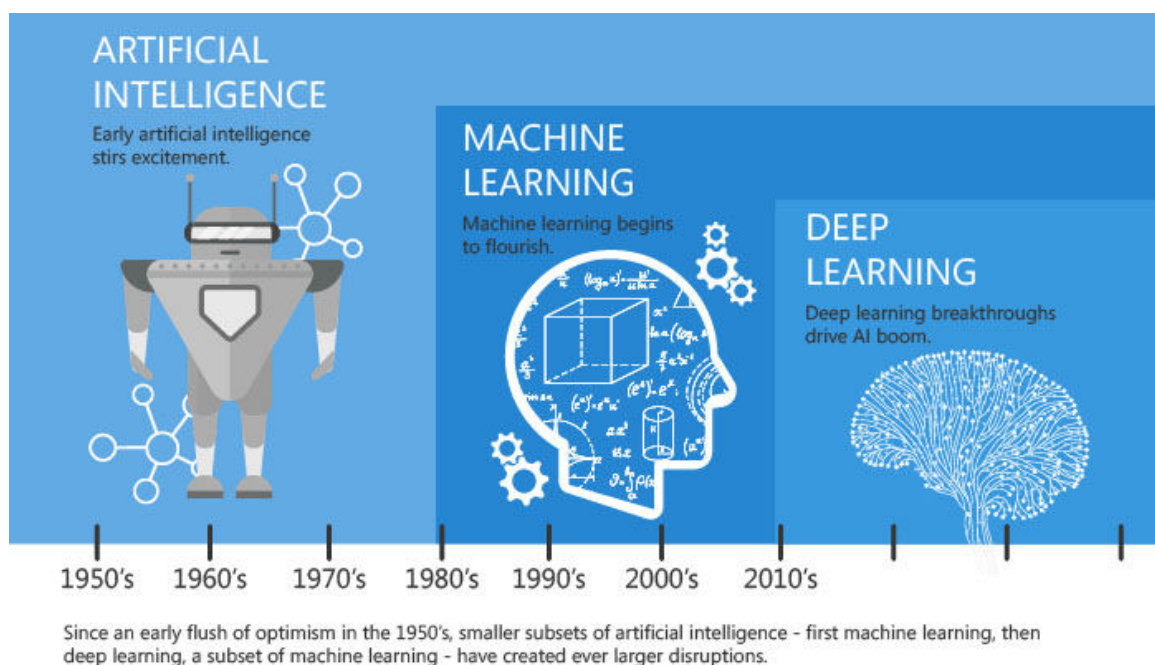


Figure 7.1: Evolution of machine learning (Credit: *Impact of Artificial Intelligence and Machine Learning on Various Industries* by Imran Uddin)

Game programming starts evolving from the 1950s by beating the world champion in checkers. This helps to improve the skills of checkers players. During that period, only the perceptron invention comes from *Frank Rosenblatt*. That was the breakthrough in the development of artificial intelligence. From there several years, we are faced with difficulties in solving certain problems using neural network concepts. In the 1990s, the statistics come into the picture of machine learning. That provides support for combining computer science with statistics to form probabilistic approaches in artificial intelligence. This changes the direction of the field towards the data-driven problem-solving methodology. It provides a base for the invention of intelligent based systems that can use large-scale data to analyze and learn. Now in trending the deep learning, a subset of machine learning used widely in the application of machine learning to solve some real-time problems.

[Classification of machine learning algorithms](#)

There are many ways to classify the machine learning algorithms. But based on the learning process, we can majorly classify them as supervised, unsupervised, and reinforcement learning algorithms. In other words, we can call this the type of machine learning technique. Let us briefly discuss the classification of machine learning algorithms.

Types of machine learning algorithms:

- **Supervised learning:** Name itself clearly says that this type of machine learning algorithm uses supervising concepts on the mapping of input data to the corresponding future prediction value. Simply we can say that input data is tagged with resultant output during the training phase of algorithms. It establishes the learning by comparing the predicted results with the actual results of input data and adjusts the model continuously until it achieves the required accuracy. Regression and classification problems are coming into this category.

[Figure 7.2](#) shows the spam prediction example. In that input, data initially marked with tags as *Spam* or *Not Spam*. This input data also called as labeled input data, which is going to use in training the supervised model. Once the model gets the training based on the labeled data, then we can test the model with new data (that is, new mails). From this process, we can get the model that can be able to predict the right output for the new data. *Decision Trees*, *Bayesian Classification*, and *Linear Regression* are the few examples of supervised machine learning algorithms.

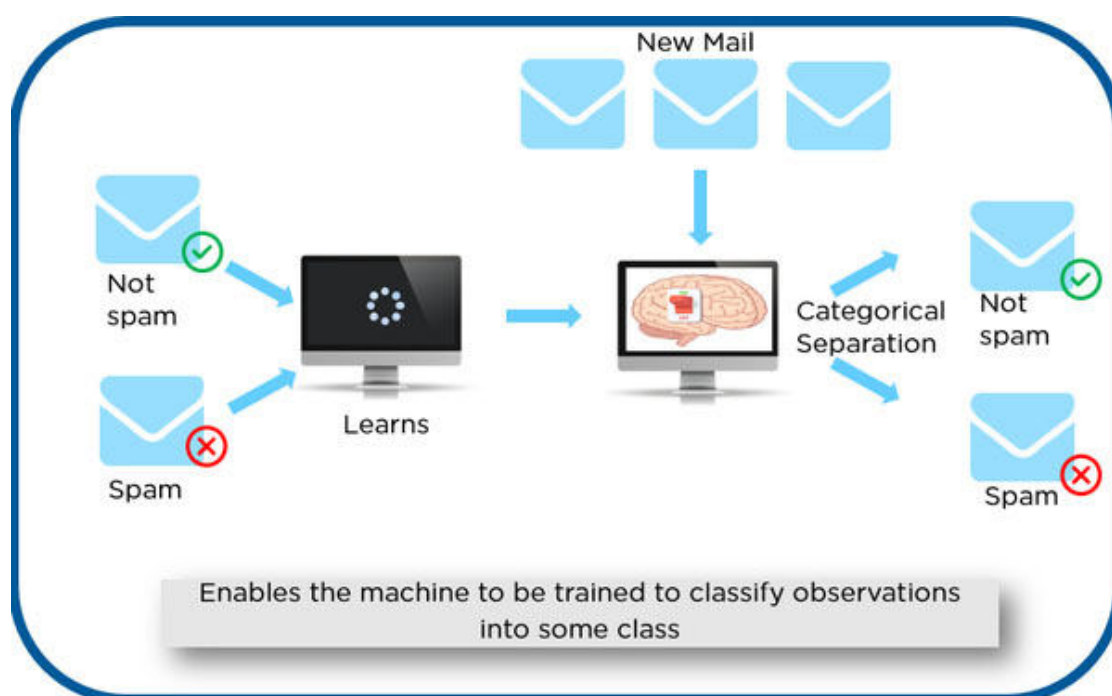


Figure 7.2: Example of Supervised Learning

- **Unsupervised learning:** In this type of machine learning algorithms Input data has no tags. Here the algorithms itself try to infer the common relationship that exists in data. Association rule learning and clustering algorithms are common examples of this type of learning method. For example, consider the classification of cartoon characters' data to identify the ducks' character. In [Figure 7.3](#) example, we have given some characters to our model, which are *Ducks* and *Not Ducks*. In our training data, we don't provide any label to the corresponding data. The unsupervised model can separate both the characters by looking at the type of data and models the underlying structure or distribution in the data in order to learn more about it. Common algorithms include independent component analysis, *K-Means*, and *Apriori* algorithms:

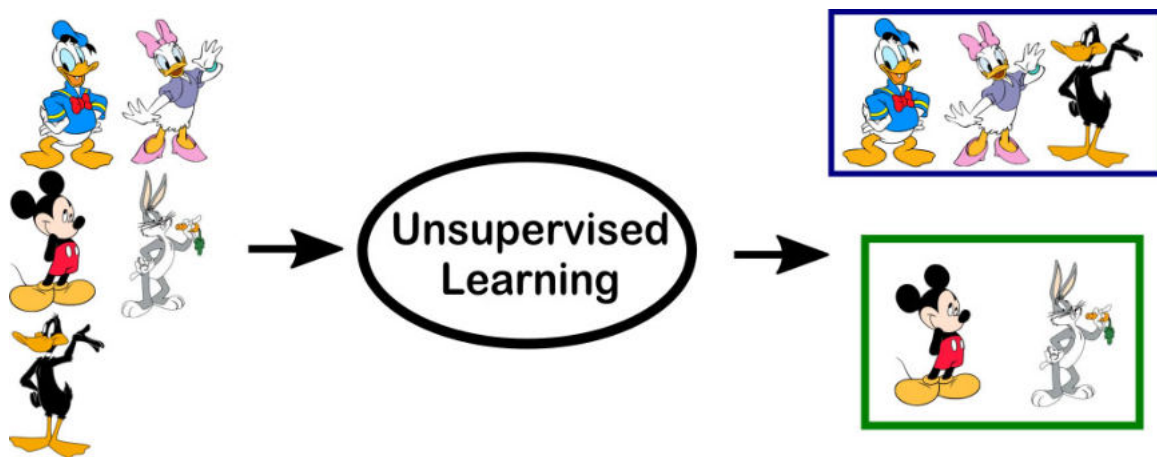


Figure 7.3: Example of Unsupervised Learning

- **Reinforcement learning:** Input data as feedback to the model, emphasizing how to act based on the environment to maximize the expected benefits. In [Figure 7.4](#) example, we can see that the agent is given 2 options, that is, a path with water or a path with fire. A reinforcement algorithm works on reward a system; that is, if the agent uses the fire path, then the rewards are subtracted, and the agent tries to learn that it should avoid the fire path. If it had chosen the water path or the safe path, then some points would have been added to the reward points; the agent then would try to learn what path is safe and what path isn't. It is basically leveraging the rewards obtained; the agent improves its environment knowledge to select the next action. The difference between supervised learning is that it does not require the correct input/output pairs and does not require a precise correction of sub-optimal behavior. Reinforcement learning is more focused on online planning and

requires a balance between exploration (in the unknown) and compliance (existing knowledge):

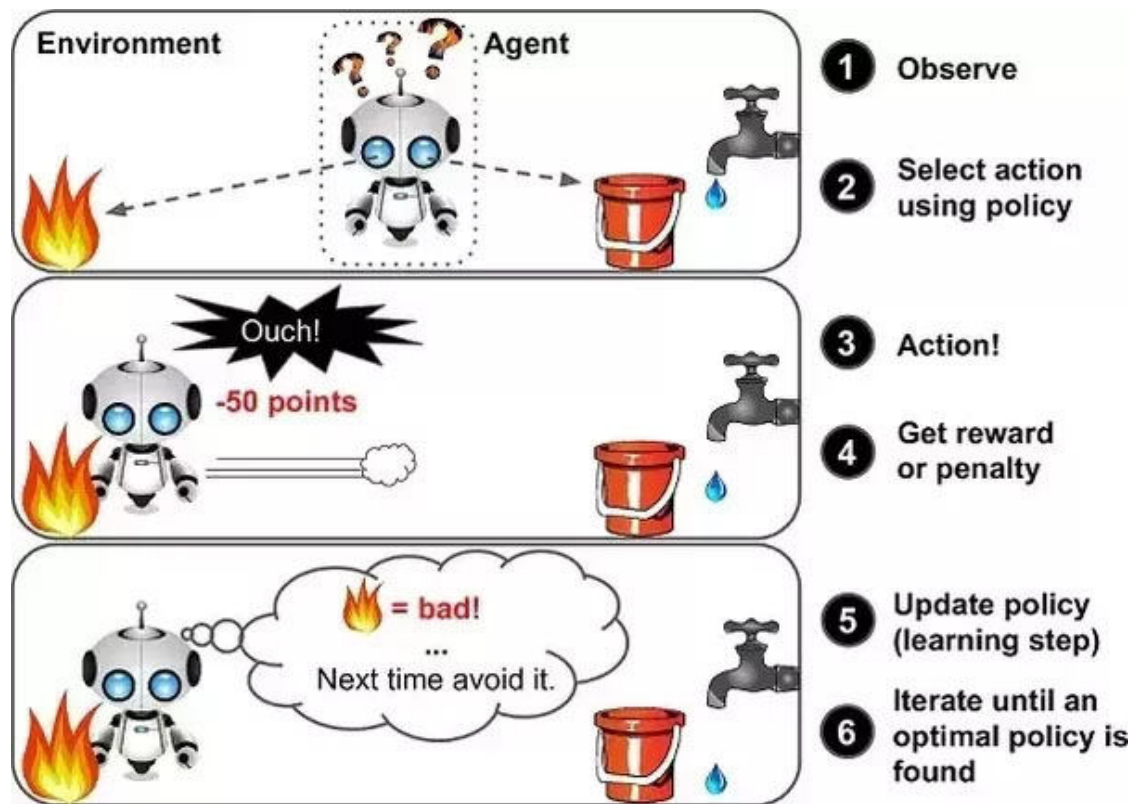


Figure 7.4: Example of Reinforcement Learning

The following [Figure 7.5](#) shows the mind map of the different types of machine learning techniques with corresponding algorithms:



Figure 7.5: Machine Learning Algorithms Mind Map

According to the function to divide, machine learning, including:

- Regression algorithm
 - Linear regression
 - Logistic regression
 - Multiple Adaptive Regression (MARS)
 - Local scatter smoothing estimate (LOESS)
- Instance-based learning algorithm
 - K — Nearest Neighbor algorithm (kNN)
 - Learning vectorization (LVQ)
 - Self-Organizing Mapping Algorithm (SOM)
 - Local Weighted Learning Algorithm (LWL)
- Regularization algorithm
 - Ridge Regression
 - LASSO (Least Absolute Shrinkage and Selection Operator)
 - Elastic Net

- Minimum Angle Regression (LARS)
- Decision tree algorithm
 - Classification and Regression Tree (CART)
 - ID3 algorithm (Iterative Dichotomiser 3)
 - C4.5 and C5.0
 - CHAID Chi-squared Automatic Interaction Detection
 - Random Forest
 - Multivariate Adaptive Regression Spline (MARS)
 - Gradient Boosting Machine (GBM)
- Bayesian algorithm
 - Naive Bayes
 - Gaussian Bayes
 - Polynomial naive Bayes
 - AODE (Averaged One-Dependence Estimators)
 - Bayesian Belief Network
- Kernel-based algorithm
 - Support vector machine (SVM)
 - Radial Basis Function (RBF)
 - Linear Discriminate Analysis (LDA)
- Clustering Algorithm
 - K — mean
 - K — medium number
 - EM algorithm
 - Hierarchical clustering
- Association rule learning
 - Apriori algorithm
 - Eclat algorithm
- Neural Networks
 - sensor

- Backpropagation algorithm (BP)
 - Hopfield network
 - Radial Basis Function Network (RBFN)
- Deep learning
 - Deep Boltzmann Machine (DBM)
 - Convolutional Neural Network (CNN)
 - Recurrent neural network (RNN, LSTM)
 - Stacked Auto-Encoder
- Dimensionality reduction algorithm
 - Principal Component Analysis (PCA)
 - Principal component regression (PCR)
 - Partial least squares regression (PLSR)
 - Salmon map
 - Multidimensional scaling analysis (MDS)
 - Projection pursuit method (PP)
 - Linear Discriminant Analysis (LDA)
 - Mixed Discriminant Analysis (MDA)
 - Quadratic Discriminant Analysis (QDA)
 - Flexible Discriminant Analysis (FDA)
- Integrated algorithm
 - Boosting
 - Bagging
 - AdaBoost
 - Stack generalization (mixed)
 - GBM algorithm
 - GBRT algorithm
 - Random forest
- Other algorithms
 - Feature selection algorithm
 - Performance evaluation algorithm

- Natural language processing
- Computer vision
- Recommended system
- Reinforcement learning
- Migration learning

Even though we have several algorithms listed above. We will briefly look into the most important and frequently used machine learning algorithms in the next section.

Top 10 algorithms of machine learning explained

The following are the very important machine learning algorithms used widely to solve real-world problems:

- **Linear regression:** For statistical technique, linear regression is used in which the value of the dependent variable is predicted through independent variables. A relationship is formed by mapping the dependent and independent variable on a line, and that line is called the regression line, which is represented by $Y = a * X + b$ where $Y = \text{Dependent variable (for example, weight)}$ $X = \text{Independent Variable (e.g., height)}$ $b = \text{Intercept}$ and $a = \text{slope}$.
- **Logistic regression:** In logistic regression, we have a lot of data whose classification is done by building an equation. This method is used to find the discrete dependent variable from the set of independent variables. Its goal is to find the best fit set of parameters. In this classifier, each feature is multiplied by a weight, and then all are added. Then the result is passed to a sigmoid function, which produces the binary output. Logistic regression generates the coefficients to predict a logit transformation of the probability.
- **Decision tree:** It belongs to a supervised learning algorithm. The decision tree can be used to classification and regression, both having a tree like structure. In a decision tree building algorithm first, the best attribute of the dataset is placed at the root, and then the training dataset is split into subsets. Splitting of data depends on the features of datasets. This process is done until the whole data is classified, and we find the leaf node at each branch. Information gain can be calculated to find which feature is giving us the highest information gain. Decision trees are built for making a training model that can be used to predict class or the value of the target variable.

- **Support Vector Machine (SVM):** The support vector machine is a binary classifier. Raw data is drawn on the n-dimensional plane. In this, a separating hyperplane is drawn to differentiate the datasets. The line drawn from the center of the line separating the two closest data-points of different categories is taken as an optimal hyperplane. This optimized separating hyperplane maximizes the margin of training data. Through this hyperplane, new data can be categorized.
- **Naive-Bayes:** It is a technique for constructing classifiers, which is based on Bayes theorem used even for highly sophisticated classification methods. It learns the probability of an object with certain features belonging to a particular group or class. In short, it is a probabilistic classifier. In this method occurrence of each feature is independent of occurrence another feature. It only needs a small amount of training data for classification, and all terms can be precomputed; thus, classifying becomes easy, quick, and efficient.
- **KNN:** This method is used for both classification and regression. It is among the simplest method of machine learning algorithms. It stores the cases, and for new data, it checks the majority of the k neighbors with which it resembles the most. KNN makes predictions using the training dataset directly.
- **K-means Clustering:** It is an unsupervised learning algorithm used to overcome the limitation of clustering. To group the datasets into clusters, the initial partition is done using Euclidean distance. Assume if we have k clusters, for each cluster, a center is defined. These centers should be far from each other, and then each point is examined thus added to the belonging nearest cluster in terms of Euclidean distance to the nearest mean until no point remains pending. A mean vector is re-calculated for each new entry. The iterative relocation is done until proper clustering is done. Thus for minimizing the objective squared error function process is repeated by generating a loop. The results of the K-means clustering algorithm are 1. The centroids of the K clusters, which are used to label newly entered data. 2. Labels for the training data.
- **Random Forest:** It is a supervised classification algorithm. Multiple numbers of decision trees taken together form a random forest algorithm, that is, the collection of many classification trees. It can be used for classification as well as regression. Each decision tree includes some rule-based systems. For the given training dataset with targets and features, the decision tree algorithm will have a set of rules. In a random forest, unlike decision trees, there is no need to calculate information gain to find the root

node. It uses the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome. Further, it calculates the vote for each predicted target. Thus, a high voted prediction is considered as the final prediction from the random forest algorithm.

- **Dimensionality Reduction Algorithms:** It is used to reduce the number of random variables by obtaining some principal variables. Feature extraction and feature selection are types of dimensionality reduction methods. It can be done by principal component analysis(PCA) is a method of extracting important variables from a large set of variables. It extracts the low dimensionality set of features from high dimensional data. It is basically used when we have more than 3-dimensional data.
- **Gradient boosting and Ada Boost Algorithms:** Gradient boosting algorithm is a regression and classification algorithm. AdaBoost only selects those features which improve the predictive power of the model. It works by choosing a base algorithm like decision trees and iteratively improving it by accounting for the incorrectly classified examples in the training set. Both algorithms are used for the boosting of the accuracy of a predictive model.

The above are the top 10 machine learning algorithms used in the industry for solving the data science problems. Let us will discuss how to build the machine learning model with an example in the next section.

[Building a machine learning model](#)

This is the process where the preprocessed features are used to build the machine learning model. Initially, the dataset must be split for training and testing. Then the data has to be fitted with different model algorithms as briefed in the previous section. Here we have the sample code that is used to implement the model for the use case, which already discussed in previous chapters. Once you run the code, you can able to get the prediction accuracy values that are used to measure the algorithm performance on the dataset. The mean squared error is also computed for each of the predicted models.

The following code defines the machine learning model using the python language. Here we start with splitting the dataset. Eventually, we will use the Bernoulli Naive-Bayes and Multinomial Naive-Bayes models:

```
from sklearn.model_selection import train_test_split
X_train_num, X_val_num, y_train_num, y_val_num =
train_test_split(X_n, y_n, → test_size=0.3, random_state=1)

# Bernoulli
```

```

from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()
clf.fit(X_train_num, y_train_num)
from sklearn import metrics
from sklearn.metrics import mean_squared_error
prediction_train = clf.predict(X_val_num)
mat_n = metrics.confusion_matrix(y_val_num, prediction_train)
mat_n
print (metrics.accuracy_score(y_val_num, prediction_train))

```

Output: 0.7533333333333333

```

# Baseline accuracy
1-(sum(y_val_num)/len(y_val_num))
# sum(prediction_train)

```

Output: 0.7493333333333334

Now let us use the CountVectorizer to count the occurrence of each word:

```

def models(l):
    # Counting the occurrence of each word in the corpus
    from sklearn.feature_extraction.text import CountVectorizer
    count_vect = CountVectorizer()
    X_train_counts = count_vect.fit_transform(l)
    count_vect.get_feature_names()
    X_matrix= X_train_counts.todense()

    y = np.array(data_train1['target'])

    # Creating the train and test split
    from sklearn.model_selection import train_test_split
    X_train_m, X_val_m, y_train_m, y_val_m =
    train_test_split(X_train_counts, y, test_size=0.3, random_state=1)

    #Multinomial
    from sklearn.naive_bayes import MultinomialNB
    clf = MultinomialNB().fit(X_train_m, y_train_m)
    labels_m = clf.predict(X_val_m)

    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import accuracy_score
    mat_m = confusion_matrix(y_val_m, labels_m)

```

```

# Bernoulli
# Changing the data to binary to input BernoulliNB
x_train_b1 = X_train_counts.todense()
X_train_counts_ber = np.where(x_train_b1 >=1,1,0)

# Creating the train and test split for bernoulli
from sklearn.model_selection import train_test_split
X_train_b, X_val_b, y_train_b, y_val_b =
train_test_split(X_train_counts_ber, y, test_size=0.3,
random_state=1)

from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB().fit(X_train_b, y_train_b)
labels_b = clf.predict(X_val_b)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
mat_b = confusion_matrix(y_val_b, labels_b)
print ('Confusion matrix:',mat_b)
print ('Accuracy using BernoulliNB:',accuracy_score(y_val_b,
labels_b))
print('Mean Squared Error:', mean_squared_error(y_val_b, labels_b))

print ('Confusion matrix:',mat_m)
print ('Accuracy using MultinomialNB:',accuracy_score(y_val_m,
labels_m))
print('Mean Squared Error:', mean_squared_error(y_val_m, labels_m))
models(all_desc)

```

[Figure 7.6](#) shows the sample output, which you will get when you run the above code. This shows that the model which builds is having the accuracy 76% when using the Bernoulli Naïve Bayes algorithm and 80% by using the Multinomial Naïve Bayes algorithm. However, the Mean Squared Error can be improved further:

```

Confusion matrix: [[524  38]
 [137  51]]
Accuracy using BernoulliNB: 0.7666666666666667
Confusion matrix: [[506  56]
 [ 93  95]]
Accuracy using MultinomialNB: 0.8013333333333333

```

Figure 7.6: Score Accuracy and Mean Squared Error of the current Model

Now let's remove the stopwords and then check the accuracy of the models and its respected mean squared error:

```
# Removing stopwords
def remove_stopwords(s):
    big_regex = re.compile(r'\b%s\b' % r'\b|\b'.join(map(re.escape, _ →
    stop_words)))
    return big_regex.sub('',s)
all_desc_wo_sw = [remove_stopwords(s) for s in all_desc]
models(all_desc_wo_sw)

Confusion matrix: [[526  36]
 [135  53]]
Accuracy using BernoulliNB: 0.772
Mean Squared Error: 0.228
Confusion matrix: [[506  56]
 [ 81 107]]
Accuracy using MultinomialNB: 0.8173333333333334
Mean Squared Error: 0.18266666666666667
```

Figure 7.7: Improved Accuracy and Mean Squared Error of the updated model

It is seen that the accuracy of both models has slightly increased now. Also, the mean squared error has decreased significantly. We can still enhance both the models by incorporating additional methods. One of them is Lemmatization. It is the process of grouping together the different forms of a word so that they can be analyzed as a single word. Let us now do the same in our existing code:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from nltk.corpus import wordnet
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

# Lemmatizing the data
```

```

all_desc_lemm = []
for i in range(0, len(all_desc_wo_sw)):
    desc = all_desc_wo_sw[i]
    desc2 = re.sub('[0-9]+\S+|\s\d+\s|\w+[0-9]+|\w+
    [\*]+\.*|\s[\*]+\s|www\.[^\s]+', '', desc)
    for p in punctuation:
        desc2 = desc2.replace(p, '')
    tagged = nltk.pos_tag(word_tokenize(desc2))
    list_lemmatized = []
    for word, tag in tagged:
        wntag = get_wordnet_pos(tag)
        if wntag is None: # not supply tag in case of None
            list_lemmatized.append(lemmatizer.lemmatize(word))
        else:
            list_lemmatized.append(lemmatizer.lemmatize(word, pos=wntag))
    k = ' '.join(list_lemmatized)
    all_desc_lemm.append(k)
models(all_desc_lemm)

```

```

Confusion matrix: [[528  34]
 [136  52]]
Accuracy using BernoulliNB: 0.7733333333333333
Mean Squared Error: 0.22666666666666666
Confusion matrix: [[509  53]
 [ 81 107]]
Accuracy using MultinomialNB: 0.8213333333333334
Mean Squared Error: 0.17866666666666667

```

Figure 7.8: Improved Accuracy and Mean Squared Error of the updated model after Lemmatization

A slight improvement in the model using the Lemmatization method can be seen. Thus we can keep on improving the model with proper methods.

Conclusion

In this chapter, we introduced the fundamental concepts of machine learning algorithm along with the history of machine learning evolution. Also, we discussed the various types of machine learning algorithms. In the next section, we briefly see the best machine learning algorithms used in the current trends. Finally, we concluded the chapter with a simple example of how to build the machine learning model. By reading this chapter, the reading will gain an understanding of various machine learning models available. He or she will also

be able to build an ML model based on a dataset. In the next chapter, we will discuss the productionizing the algorithms for implementation.

CHAPTER 8

Productionizing Machine Learning Models

The machine learning models existed for decades in our research labs and worked on by researchers to develop new features in products. The recent developments in cloud computing have lowered the price of computation and storage, as well as made internet services available across the world. Now, the end customer expects the benefits of AI/ML to be delivered to him via web-applications and smartphones. The process to deliver ML result to end customer over the internet is called productionizing the machine learning.

In previous sections, you learned about algorithms and the type of problems they solve using data. In this chapter, we will introduce the basic concepts of productizing algorithms. The chapter will also introduce REST APIs to deliver the model result in real-time environment. The interface produced for a general user will show how ML delivers value in the simplest and most impactful manner. We will productionize an ML model using the Flask framework and show the steps.

In any machine learning project, productionizing a machine learning model is a significant part. You can build an attractive website and a REST API using Flask. It's just a matter of a few lines of code.

Structure

- Type of ML production system
- Introduction to REST APIs
- Flask framework
- Jupyter application to the flask server
- Front end interface
- Key considerations for a scalable system

- Conclusion

Objectives

After studying this chapter, you should be able to:

- Understand the concepts of different types of ML production system.
- Understand the REST APIs.
- Build the Jupyter application to the flask server.
- Design the frontend interface and understand considerations for a scalable system.

Types of ML production system

ML production system refers to a large stack of processes and infrastructure to support delivering ML predictions to end-user. The production system is characterized by two key factors:

- Model training (online or offline)
- Model scoring (on-demand or batch)

The above two factors help decide on what type of production system to be built for the given use case.

The use cases can then be further divided into four quadrants based on training and scoring preferences. [*Figure 8.1*](#) shows the four quadrants:

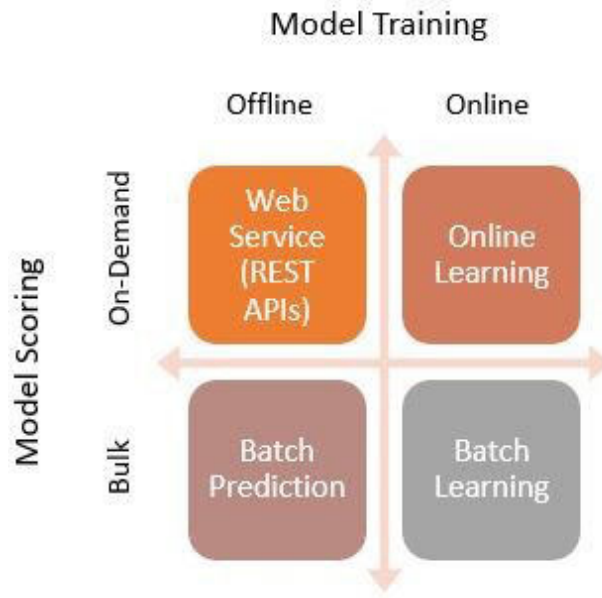


Figure 8.1: ML production systems

In the above framework, we divide the production environment into four types; all four systems cover the popular approaches applied by industry to deliver ML models to end-users. The environments are discussed below.

Batch prediction

Batch prediction is the simplest machine learning flow. This flow typically involves historic data available as a static file, for example, CSV, database dump, and many more, and model being developed in local systems. This ML flow you will find mostly in academia, research teams, and historical data analysis tasks. For example, you get a CSV file of data, train, and test your model and output a CSV file of predictions. This type of output makes very little use in industry setting accepts in historical reports.

Batch prediction puts up a sequence of automated steps and provides the output by scoring new data on a model in regular intervals and store the output in bulk as a flat file or into a database. This type of ML flow allows doing analysis at regular intervals on bulk data.

Batch learning

There are many cases where a large historical data dump requires complete retraining of the models to be applied on new datasets. In these cases, Batch

learning is required where a volume of data needs to be analyzed in real-time to create a new model. This method is also known as automated machine learning.

This method is very handy in cases like recommendation engines for large e-commerce websites, where they update all recommendation score once in every 24 hours automatically in their system.

REST APIs

This is the most popular method to deliver ML results to end-users. In this method, the model is trained offline and put behind an HTTP server to provide predictions to data points as and when required. Most of the client-side applications handle one request at a time for prediction, and hence having a REST APIs can provide system-independent on-demand scoring of models.

Assume you developed an object detection model offline, and you want to build that into a mobile application. The mobile application will be snapping one picture at a time and asking the REST API to return the predictions for objects detected. In these cases, a REST API based web service makes it easy to manage and handle requests from multiple types of systems.

Online learning

The online learning is becoming very popular as the algorithms allow training or models in a live production system with every new datapoint. This helps in keeping model up-to-date and adapt to changing behaviors very fast. Frequent changes also come with the risk of distortion by outliers. However, most of the online learning system has rollback features to make sure we do monitor and adjust online learning models in production.

You can think of a customer segmentation models deployed as online learning models, which keeps learning behavior from each new customer visiting your system. The segmentations keep adoption according to the data flowing into the system. This is a very different approach to REST API based web services, where the models are trained offline and only exposed via web services.

All types of production systems can be controlled and managed by REST APIs and through web services. Read carefully the distinction

between what the APIs functionality is in the system.

Introduction to REST APIs

REST stands for **REpresentational State Transfer**. REST is an architectural style for **APIs**. APIs stands for **Application Programming Interface**. REST architecture was introduced by *Roy Fielding* in his Ph.D. thesis *Architectural Styles and the Design of Network-based Software Architectures* in 2000 at UC Irvine. You can read the thesis [here](#).

RESTful APIs are a very vast topic that covers network architectures and application design patterns. Within the scope of this book, we will only discuss how to use the REST architecture to create APIs for our python-based ML models. To understand the REST concepts, we will briefly introduce key terms; APIs, http, client-server architecture, and resource.

Application Programming Interface (APIs)

APIs are a collection of communication protocols and subroutines used by programs to communicate with each other. APIs provide an open-source standard accepted for programs to communicate with each other and hence allowing developers to develop different parts of the application independently. APIs are developed for different types and purposes and differ in the way they work. In our scope, we will focus on Web APIs, which are delivered via Http protocol.

For example, as shown in [Figure 8.2](#), the Twitter API allows the interface between a user and twitter through the API interface:

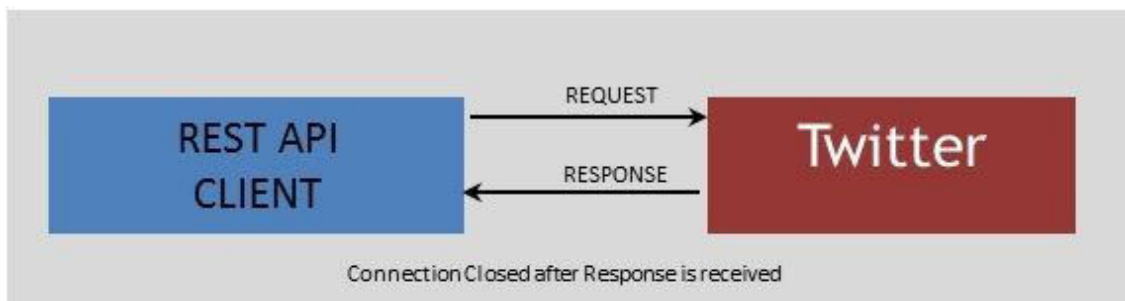


Figure 8.2: API Interfaces

An application can call the twitter API to collect information or post a new tweet. This makes the developer and twitter two completely different

software talk to each other.

Hyper Text Transfer Protocol (HTTP)

The **Hyper Text Transfer Protocol (HTTP)** is the client-server network protocol that has been in use by the World-Wide Web since 1990. It is important to note that it's the HTTP protocol that allowed the browsers and website to become so popular and accessible. Every time you access a web page, the browser sends an HTTP request messages for HTML pages, images, scripts, and styles sheets from the server as defined by the **URL**. URL stands for **Universal Resource Locator**. An example of an URL decoding for http request is shown in [Figure 8.3](#):



Figure 8.3: URL decoding for http request

An important feature of http is that it's a stateless protocol meaning that the server isn't required to store session information, and each request is independent of the other. This allows multiple requests to the same resource from clients.

HTTP request comprises three mandatory lines in any http request; method, resource path, and protocol version. An example could be:

```
GET /predict.htm HTTP/1.1
```

Here, the method type is GET (HTTP protocol allows various methods POST, PUT, DELETE, and more.), the resource path is `/predict.htm` at the server/IP, and the protocol used in HTTP version 1.1. You can read more about method types (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>).

For each request, there is a response. The response comprises of request status code, 1 or more headers, and optional body message. For example, if you are fetching a webpage from a server and the request is successful, it

will return a status code on 200 along with the request webpage content. You can read about response codes (<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>).

Client-server architecture

The Client-Server architecture is the architecture of a computer network where multiple clients (users) request and receive service from a centralized server (the host). The clients have an interface to make such requests to the server, while the server is always running and waiting for requests to deliver the resources. In an APIs driven system, the server and clients can be entirely independent and create with very different programming languages and hosted anywhere on the internet accessible by http protocol. [Figure 8.4](#) is a basic architecture of a Client-Server. You can read more (https://cio-wiki.org/wiki/Client_Server_Architecture):

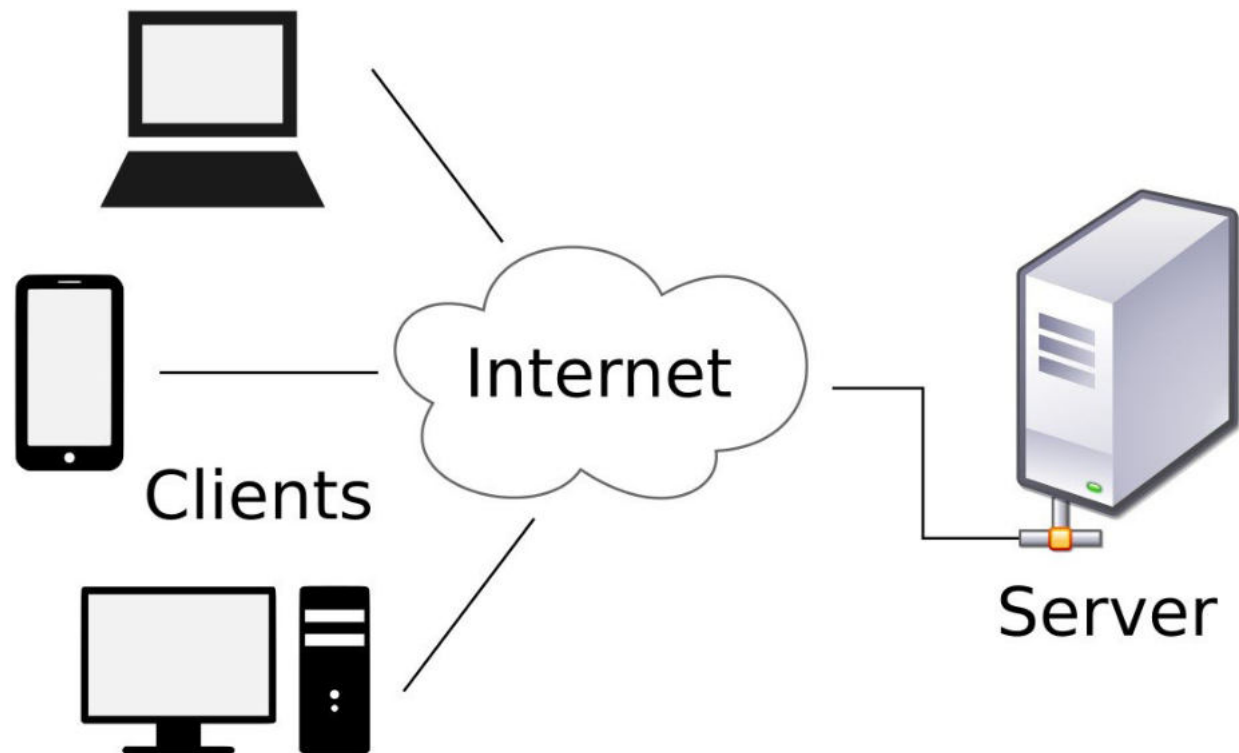


Figure 8.4: Client Server Architecture

In this model, we can develop an ML model and store it at the host machine, open a server interface for APIs, and listen to requests for prediction from multiple different clients' on-demand basis. The communication between

host and clients is controlled by TCP/UDP transport layer protocol; this is out of scope for discussion.

Resource

The resource is a fundamental concept in the RESTful API. It is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. The http methods are defined on the resource to operate on them when a request is made. **JavaScript Object Notation (JSON)**, a lightweight data-interchange format, is best suited to define the data model for RESTful resources. You can read more about JSON (<http://www.json.org/>). There are other popular representations for RESTful resources like XML, YAML, HTML, and others. You can read more about resources (<https://restful-api-design.readthedocs.io/en/latest/resources.html>). *Figure 8.5* is an example of resource access through RESTful APIs:

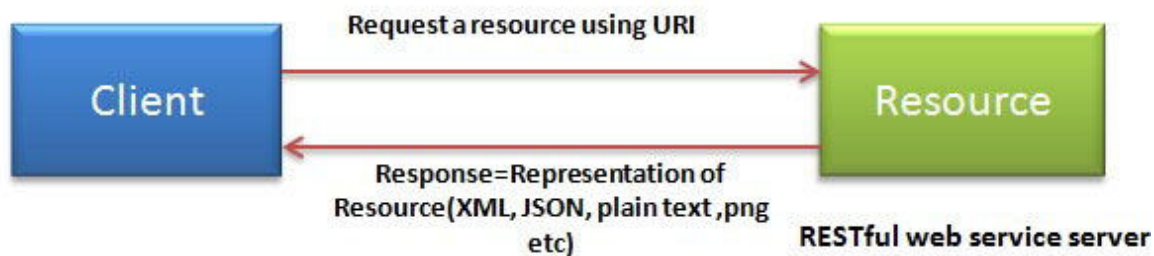


Figure 8.5: Resource Access through RESTful APIs

For example, in Twitter API, a resource can be a user, hashtag, photo, location, etc. Each resource has a unique identified by which the program operates upon them and then transfer in the appropriate format, that is, JSON.

Now, we have the basic concepts covered to define what must be the functionality of the RESTful web server/application. This application has to provide itself with information in the sort of its resources' information. Its service allows customers to access these resources via http requests. In RESTful web service, the server transfers a statement of the state of the requested resource to the client, when a RESTful API is called.

For instance, while calling a Twitter API so as to get a user's tweets, the API would return the user's name, number of followers, number of posts, and

more.

Flask framework

Flask is a python web application framework. The framework provides the tools, libraries, and infrastructure communication protocols to allow the developer to build a web application. The Flask web framework is based on python as the web application runtime environment. Flask also falls into the category of micro-frameworks, where it has little to no dependency on external libraries to server requests. The core dependency of the Flask framework is very minimal. Below is the list of those dependencies:

- **Werkzeug or Gunicorn:** It is a Web Services Gateway Interface (WSGI) utility library that helps Python web server communication with a Python application running in the server host machine
- **jinja2:** It is a template engine which allows flask to create html pages/templates and embedded them with application responses.

The Flask based Python applications are very popular in AI/ML deployment as that allows developers to directly integrate complex Python ML models into a web application. [Figure 8.6](#) shows a simple framework to productionize the ML models using Flask:

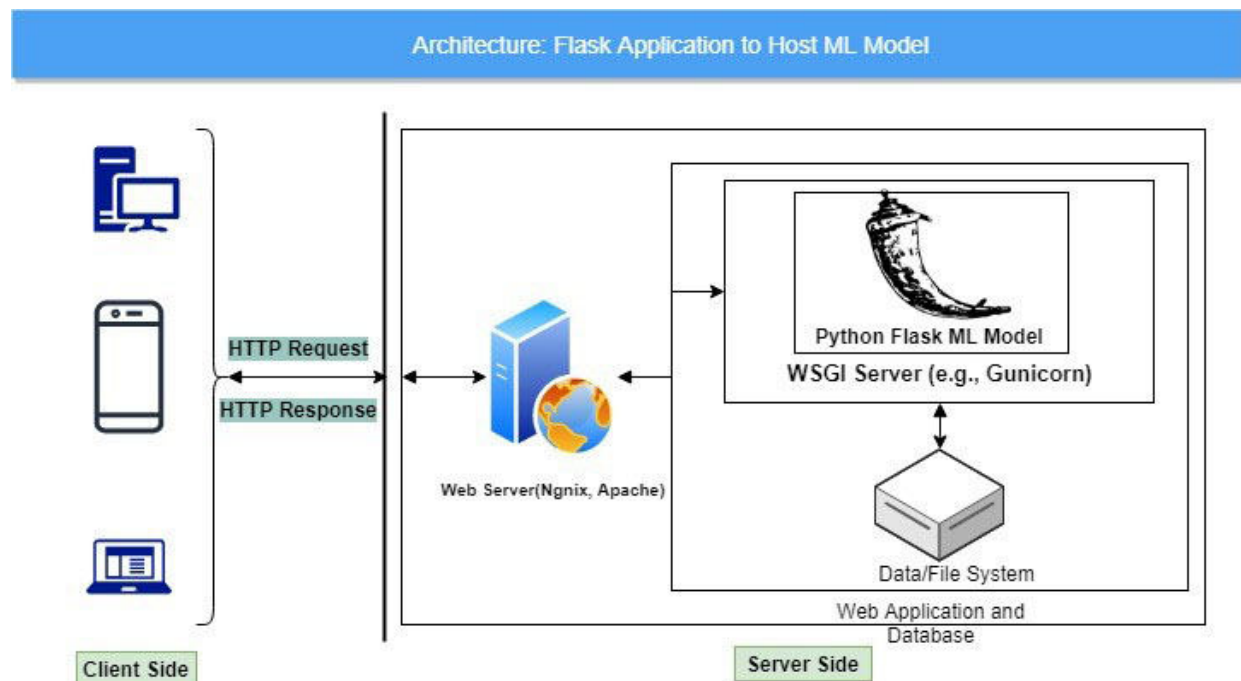


Figure 8.6: Flask Web Service Architecture

The key components and their functions are explained below:

- **Client side:** All the applications that want to access our ML model will be a unique client. It can be a mobile application or a web server or another system. The client will make an HTTP request over the RESTful APIs interface to access the resource in the host/server side.
- **Server side:** All the applications and resources required to fulfill the request by the client are on the server side of the application. In very large production environments, the architecture might require a lot of more considerations, but at basic, it requires a web server, WSGI interface, access to data/file system, and the python application running inside Flask.
- **Web Server:** The web server keeps listening to http requests over the internet for any new request coming for any client accessing the resources. Flask provides a single threaded web server to host applications, but single threaded cannot handle more than 1 request at a time. Having a scalable server like NGINX or Apache is recommended.
- **WSGI interface:** WSGI interface provides the bridge between a Python application and the web server. The flask application runs a Python code at the host machine accessible by http request.
- **Flask application:** The flask application is a micro-service itself; it encapsulates the Python application and runs behind a WSGI interface.
- **Data/File System:** The Python application can interact with databases and files system as required by the application. This layer can be communicated by the application within server-side processing using Python programming language tools and libraries.

Now let's build a machine learning model and create a Flask application to host the model.

Simple flask application

The ML models are built by data scientists usually in the **Integrated Development Environment (IDE)** for Python, or R. Jupyter is a popular IDE for Python while RStudio is very popular for R. R has a similar web

framework like Flask, named *Shiny*. R practitioners can read more about Shiny here (<https://shiny.rstudio.com/>). In this section, we will build a simple ML model in Jupyter Notebook and host that as a Flask application to respond to the HTTP request.

We will take the example of salary prediction to showcase the process for setting up a linear regression model as a web service.

Salary_prediction model

The salary prediction model tries to fit a linear regression model on the mean salary of the employee against their experience in the company. We will not focus much on the model building process but more on how a basic model is built and productionize using the Flask framework.

We will first load the data and have a look at the data structure:

```
#Import libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the datasets
data = pd.read_csv('Salary_Data.csv')

#Have a look at top 5 rows of data
data.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

The dataset has two columns, years of experience and salary. The general hypothesis we will assume here is that as the experience increases, the salary increases. The linear regression model should fit this hypothesis.

Now we create a train and test split (75% Train and 25% Test) of data. The model will be trained on Train split using sklearn linear regression. We will also store the model in a pickle object for future use. Read more about model storage (https://scikit-learn.org/stable/modules/model_persistence.html).

```
#Load the values on variables in a array
```

```

X = data[['YearsExperience']].values
Y = data[['Salary']].values

# Splitting the into the Training set and Test set
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y,
test_size = 0.25, random_state = 0)

# Fitting Simple Linear Regression to the training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)

#Store the model as a pickle object
import pickle
pickle.dump(regressor,open( "linear_reg_salary_model.p", "wb" ))

```

Now the linear regression model has been trained on training data and is available in the regression Python object. Before we declare this model ready for productizing let's look at its fit on test data:

```

# Predicting the Test set result
Y_Pred = regressor.predict(X_Test)

# Visualising the Test set results
plt.scatter(X_Test, Y_Test, color = 'red')
plt.plot(X_Train, regressor.predict(X_Train), color = 'blue')
plt.title('Salary vs Experience (Training Set)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()

```

[Figure 8.7](#): shows a reasonably good fit on the test data:



Figure 8.7: Salary vs. Experience Plot for Test Data

We can now move onto the next step of making this model available for use by end users. Now, let's revisit the productionization framework we discussed in the earlier section.

- **Batch mode:** If the users like to predict the salary of bulk of employees in their own choice or frequency. The model can be applied by loading the model from memory and using the predict method on the set of employee data. This method a CSV can be the output.
- **Web service (REST APIs):** If the results of salary are required to be available on demand for individual requests, we need to host the application on the web as REST APIs. This case is more likely to happen for individual user interfacing applications. For example, suppose this is a feature on the job portal to check the expected salary.

Now, let us show how both methods will work. For the batch model, let's have assumed a new bulk of employee is uploaded to the system, having their experience data stored in a CSV file `new_employee.csv`.

We would need to invoke our python script in our IDE, or some other method to load the new data, the stored model, and predict the expected salary and store as CSV file.

```
#Load the New Employee Data
new_data = pd.read_csv('new_employee.csv')
new_employee_exp = new_data[['YearsExperience']].values
```

```

#Load the model
import pickle
pickle.load(open( "linear_reg_salary_model.p", "rb" ))

#Make the predictions
predict_new = regressor.predict(new_employee_exp)

print(predict_new)

#Store the predictions into a csv file
np.savetxt("New Employee Salary Prediction.csv", predict_new,
delimiter=",")
[[46684.08334982]
 [57939.73594016]
 [41056.25705466]
 [64505.53328452]
 [66381.47538291]
 [54187.85174338]
 [73885.24377647]
 [75761.18587486]
 [57001.76489096]
 [80451.04112083]]

```

Now, you can see in this method we are making a bulk prediction on the offline trained model. We cannot run these predictions every time a new user comes in, asking for their salary predictions.

This is a limitation on delivering value for ML models into the real production environment for individual users.

Now we understand we cannot deliver our model as Python notebooks or python scripts as they still require a programmer to run the code and always assist whenever someone needs a prediction to be done. This is not possible to scale and available 24x7.

In method 2, below, we implement our prediction script inside a Flask app and explain its components in the comments:

```

#Load the Libraries
from flask import Flask, request, jsonify
import pickle
import json

```

```

import pandas as pd

#Start a flask app
app = Flask(__name__)

# Load the model
regressor = pickle.load(open( "linear_reg_salary_model.p", "rb"
))

@app.route('/predict', methods=['POST'])
def predict():
    #Retrieve the value of 'YearsofExperince' from the request body
    data = request.get_json()
    df = pd.DataFrame([float(data['YearsExperience'])], columns=
['content'])
    predict_new = regressor.predict(df)
    result = {'predicted_salary': predict_new.tolist()[0]}
    return json.dumps((result))

if __name__ == '__main__':
    app.run(port=3000, debug=True)

```

Store the above code in salary_flask_app.py and got to the command line to run it by using the following command. Please refer the code python sample_flask_app.py.

This will start a web server with the following details on your console. Debugger pin is randomly generated for each instance of server:

```

* Serving Flask app "salary_flask_app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a
production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 276-578-683
* Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)

```

The above out states that the web service is up and listening to <http://127.0.0.1:3000/predict> address for HTTP RESTful requests. We will

make a request by using request module in Python to test if the service returns the expected salary if we pass the YearsExperience via an HTTP web request.

```
#import the request library which allows us to make Post/Get etc
web request
import requests

#Define the address of host where the application is running
url = 'http://127.0.0.1:3000/predict'
payload = { "YearsExperience": 3.2 }
res = requests.post(url,json = payload)

print(res.json())
{'predicted_salary': [57001.7648909645]}
```

The client can send as many requests on-demand to the server to get the predictions being returned to the client. The client does not need to create a csv file or wait for the batch prediction script to run. This makes the prediction of real time for the clients.

In the above section, we showed how a model could be trained and then hosted as RESTful APIs using a flask server. Technically, we made the developers' life easy who wants to integrate the smart ML model output into an application. The APIs can deliver the required insights to the application as and when required. It is equally important for the data scientist to interact with the application as a normal user to ascertain the productization will be fruitful.

In the above example, if a user needs to check his expected salary, he needs to install python, start a Jupyter notebook, create post request using requests package, and a lot of other technical considerations. A normal user is not knowledgeable enough to use your model outputs. Therefore, we need to look at how we can create a simple interface to interact with our models. This is achieved by created website/frontend applications to accept user inputs from front-end/webpages and return the result in easy to consume over webpages. In the next section, we will build a simple webpage to deliver the expected salary to end users.

ML model user interface

The front-end engineers are responsible for creating an intuitive and impact UI to display and interact with the user. The user interface plays important role in delivering the results from a productionized ML model. In this section, we will build a small HTML page to be rendered over http and accessible over a web-browser.

The core purpose of the webpage will be to allow the user to input the experience in a number of years and get the expected salary printed on the screen.

HTML template

An HTML template will create the structure of the webpage to be delivered via a webpage. For our purpose, the webpage needs to have three essential features:

- **Input field:** An input field to allow the user to input the years of experience.
- **Output div:** An output div container to display the result returned by the Flask APIs.
- **Form:** A form will capture the input and send the input to the Flask Server for a response as a POST request.

Below, you can see a sample HTML page for our salary prediction application:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Salary Prediction ML Model</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css" integrity="sha384-
gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQU0hcWr7x9JvoRxt2MZw1T
" crossorigin="anonymous">
```



```

</head>
<body style="width:60%; margin-left:20%;">
<h1 class="text-center">Salary Prediction ML Model</h1>
<form action="/predict" method="post">
<input class="form-control" type="number" name="YearsExperience"
step="any" min = "0" placeholder="Number of years of
Experience">
<input class="btn btn-primary" type="submit" value="Predict
Salary">
</form>
    {% if salary %}
<p>The Expected Salary is: {{salary}}</p>
    {% endif %}
</body>
</html>

```

You can observe the HTML template the three essential components we described above. The form elements get data from the input field and make a POST request at /predict route. The result is returned in the variable name salary and displayed with the div below form elements.

You put this HTML file inside a folder named templates in the home directory and then defines the directory in the Flask app to point templates in this folder. The update Flask app code is provided below:

```

#Load the Libraries
from flask import Flask,request,render_template
import pickle
import pandas as pd

#Start a flask app
app = Flask(__name__,template_folder="templates")

# Load the model
regressor = pickle.load(open( "linear_reg_salary_model.p", "rb"
))

@app.route('/predict', methods=['GET','POST'])
def predict():
    if request.method == 'POST':

```

```

#Retrieve the value of 'YearsofExperince' from the request
body
data = request.form.get('YearsExperience')
df = pd.DataFrame([str(data)], columns=['content'])
print(data)
predict_new = regressor.predict(df)
return render_template('index.html',
    salary=predict_new.tolist()[0])
return render_template('index.html', salary = '')

if __name__ == '__main__':
    app.run(port=3000, debug=True)

```

You can see the app invocation mentions the relative location of templates while starting the Flask app. The /predict route now listens to both GET and POST requests, even though the form does the only POST request. The GET request makes sure the page loads even before you make your first POST request from the form.

Also, you can observe the return statement returns the full HTML page along with the predicted salary in the variable salary. You can save the Flask app as salary_flask_app_website.py in the home directory and run the app as below. Please refer to the code file Python salary_flask_app_website.py.

A webpage now can be accessed at <http://127.0.0.1:3000/predict> and can be used by end user to input his experience, and in return, it will show his expected salary. [Figure 8.8](#) is a snapshot of the webpage:

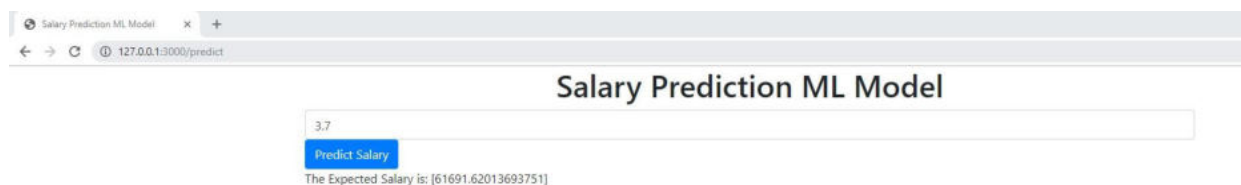


Figure 8.8: Salary Prediction ML Model Webpage

In this section, we created a basic frontend exposure of our productionized ML model. Now the user can simply access this URL and access our productionized model. This way we just exposed our ML model to a novice who can use and get benefited from the model.

You can see though we have productionized our model. But they still host on 127.0.0.1, which is our localhost/local machine. You need to keep your machine on and can be accessed only by you.

In further chapters, we will discuss how we optimize the way we productionize the models and what are options for deploying the applications on the cloud. The cloud deployment will allow you to give access to your ML models to the open world through the internet.

Conclusion

In this chapter, we have discussed the limitations of ML models being restricted to data scientists only without productionizing them. It is important to productionize the models to make them available to other applications and users. We discussed the productionizing methods, broadly classified by model training and model scoring methods. Further, we discussed RESTful APIs and https requests as a primer to our efforts to productionize the models as a web service. The salary prediction model is developed to show a simple model building and testing in Jupyter notebook settings. Then we tried to emulate a process of bulk forecasting the model developed in the previous step. Once we confirmed the bulk mode works, we introduced Flask applications and the architecture that allows our models to be exposed as RESTful APIs through Flask applications. We developed a basic Flask application that loads the models and make a prediction on a POST request. To facilitate non-technical users to access the application without making a POST request, we introduced the idea of front-end web pages. The last section then shows how a web-page can be created as a template and delivered via a Flask app to end users.

By reading this chapter, the reader has understood the productionizing methods, RESTful APIs, and HTTP requests. He or she will also gain the experience of building a basic machine learning model and developing a basic Flask application on top of it.

In the upcoming chapters, we will show how to host these applications on the cloud and make them accessible from anywhere. We will start by defining some of the technology involved in designing a data pipeline.

CHAPTER 9

Data Flows in Enterprises

An enterprise is a very complex system of technology and business processes. Applications for enterprise are designed to handle multiple sources of data operating asynchronously and controlled by different factors. Any new solution that is created needs to fit into the specific organization's technology infrastructure. In this chapter, we will define some of the technology involved in data collection and data transport from one system to another by designed a data pipeline. The data pipeline will be built for a sample application along with Python codes provided within the chapters. We will show how to host these applications on the cloud and make them accessible from anywhere. We will start by defining some of the technology involved in designing a data pipeline.

Structure

- Introducing data pipeline
- Designing data pipeline
- ETL vs. ELT
- Scheduling jobs
- Messaging queue
- Passing arguments to the data pipeline

Objectives

After studying this unit, you should be able to:

- Understand why we need data pipelines in implementation of AI/ML models
- How to design simple data pipeline for a model
- Difference between ETL and ELT, and when to use what

Introducing data pipeline

The data pipeline is the end to end process, which defines how the data flows within the enterprise system for all the data needs. The data pipeline can be subcategorized into many specific data pipelines, like data science modeling, forecasting, reporting, and many more. The data pipeline is a broader term that encompasses many processes in the transfer of data from one system to another, including security, reading, serialization, transfer, transformations, writing, and other steps.

The efficient and secure flow of the data from the warehouse to the application, application to application, and application to warehouse is a very important process to keep running the system. The key component of the data pipeline is the ETL process. **ETL** stands for **Extract Transform and Load** process, as shown in [Figure 9.1](#). In a pipeline, there may be many ETL sub-process running to deliver the data applications needed to run and store them:

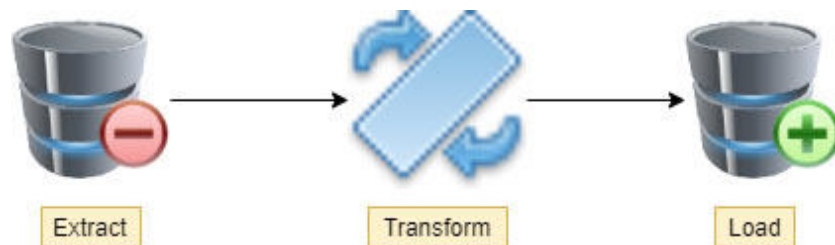


Figure 9.1: Extract Transform Load (ETL) Flow

An example of ETL can be reading the data from the manufacturing database, adding 5 days to the due date, and writing that data into the order management database. While a data pipeline is broader implications and could mean reading the data, streaming it to order management systems, applying ML models, and getting the results back to manufacturing database and storing the data in some visualization dashboard.

In this way, you can understand the data pipeline is a broader process that ensures that the system is efficiently and securely able to transfer data when required by the processes. Another important aspect of the data pipeline is that it does not start and end but is capable of taking data from any system to any system. This feature of the data pipeline makes them multi-directional in nature.

The data pipelines can be further categorized into three types, based on what kind of data flow they handle:

- **Batch processing:** Batch processing pipelines are the most popular and used to handle large volumes of data moving in/out of warehouses for various purposes of reporting, processing, and storage. The key feature of batch processing is that it is usually scheduled for a fixed time and does not require to deal with real-time data.
- **Stream processing:** Stream processing pipelines are designed to handle real-time data. The latency between the point of data origination and next process is nearly zero. This type of pipelines is used for real-time analysis of data in time-critical applications.
- **Cloud-native processing:** The cloud infrastructure provides multiple options and configurations to design the pipelines, native to cloud storage, and distributed services. They can be real-time, quasi real-time, batch, and can be configured to be hybrid. All the tools and infrastructure are provided by the cloud provider and pipeline managed by them. AWS, Azure, and Google cloud are the leading provider of managed cloud native pipelines.

To understand the different basic concepts of the pipeline, we will be showing the concepts with a live example. The reader is encouraged to follow so that he/she gets a hands-on understanding.

Business Case: Design a data pipeline to get data from *data.gov.in* website for the *Current Daily Price of Various Commodities from Various Markets (Mandi)* and store that data in a CSV file.

You have to register at *data.gov.in* to get the API key. Visit here (<https://data.gov.in/>) to create an account.

Designing data pipeline

Designing a data pipeline is a core responsibility of a data engineer. A data engineer knows where the data get generated, how that can be transferred, the memory, and bandwidth limitations of the network, security best practices, and infrastructure status. The key considerations before designing the pipeline could be to answer some of the following questions:

1. Where is the data source?

2. What type of data is to be extracted from the source?
3. What transformation to be applied to the data in the pipeline?
4. What is the destination of data?
5. Does the destination produce some data which needs to be transported or stored?
6. Is the pipeline batch processing or stream processing?
7. Who can trigger the pipeline, or will be scheduled?

And other questions to cover are configurations and infrastructure aspects of design. Overall, the whole pipeline design has to deliver the value; business is looking from the technology. We will now discuss a simple pipeline design, as explained by our example, and show its implementation in Python.

You must have the API key available with you to run the examples. [Figure 9.2](#) shows a simple data pipeline design to be written in the Python language.

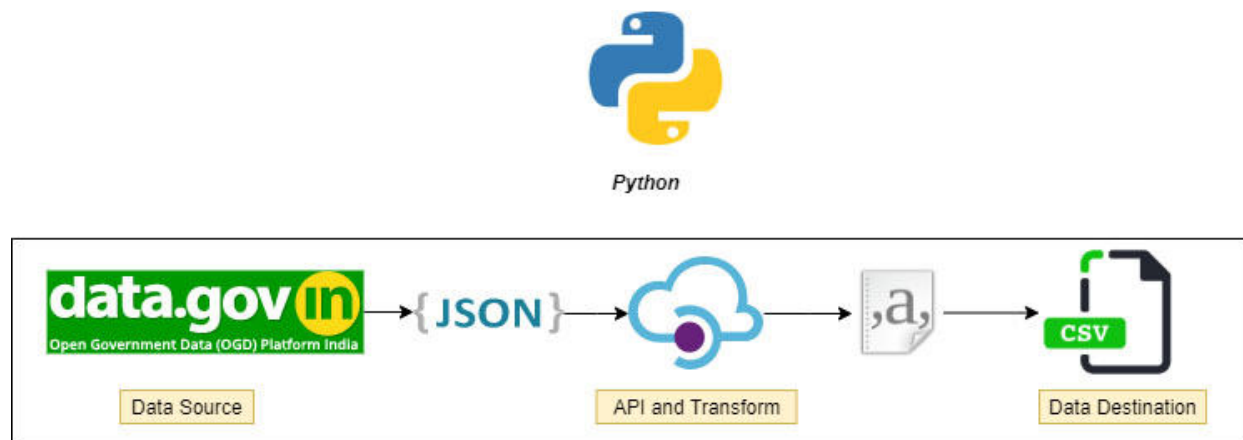


Figure 9.2: Basic Data Pipeline Design for example

The pipeline can be written in any other language as well, which provides support for various functionalities of the pipeline. Here is what this pipeline will do:

1. It will first get connected to the data.gov.in website and provide its authentication key (the API key).
2. Once the key is authorized, depending upon the parameters, the data will be transferred to the client as a JSON object.

3. The client reads the JSON file and opens up a stream to a local CSV file.
4. The client then transforms the data into CSV format and writes it into the local file.
5. Once the process is complete, the pipeline closes.

Now let's write a simple Python script to achieve this data pipeline objective:

```
##Python Script to Implement the Pipeline

#Define the API KEY (this can be found in the data.gov.in
account section for registered users)
API_KEY = <YOUR API KEY>

#Import the requests library
import requests

#Construct the GET REQUEST
response = requests.get(
    'https://api.data.gov.in/resource/9ef84268-d588-465a-
a308-a864a43d0070',
    params=[('api-key', API_KEY),
            ('format', 'json'),
            ('offset', 0),
            ('limit', 1)],
)

#Check if the request was successful - a success request returns
a status code 200
if response.status_code == 200:
    print('Success!')
else:
    print('Some Error Occurred')

#If the you see a success message then you can extract the
values from the JSON response
json_response = response.json()
Success!
```

You can see the above call to *data.gov.in*. API returns a success, which means it authenticated our API key and sends the data as JSON to our client.

The response variable has JSON in it, which is being extracted using json() function. Below we print out the output using pprint library:

```
import pprint
pprint.pprint(json_response)
{'active': '1',
 'catalog_uuid': '6141ea17-a69d-4713-b600-0a43c8fd9a6c',
 'count': 1,
 'created': 1543321994,
 'created_date': '2018-11-27T18:03:14Z',
 'desc': 'Current Daily Price of Various Commodities from
Various Markets '
 '(Mandi)',
 'field': [{'id': 'timestamp', 'name': 'timestamp', 'type':
'double'},
 {'id': 'state', 'name': 'state', 'type': 'keyword'},
 {'id': 'district', 'name': 'district', 'type': 'keyword'},
 {'id': 'market', 'name': 'market', 'type': 'keyword'},
 {'id': 'commodity', 'name': 'commodity', 'type':
'keyword'},
 {'id': 'variety', 'name': 'variety', 'type': 'keyword'},
 {'id': 'arrival_date', 'name': 'arrival_date', 'type':
'date'},
 {'id': 'min_price', 'name': 'min_price', 'type': 'double'},
 {'id': 'max_price', 'name': 'max_price', 'type': 'double'},
 {'id': 'modal_price', 'name': 'modal_price', 'type':
'double'}],
 'index_name': '9ef84268-d588-465a-a308-a864a43d0070',
 'limit': '1',
 'message': 'Resource detail',
 'offset': '0',
 'org': ['Ministry of Agriculture and Farmers Welfare',
 'Department of Agriculture, Cooperation and Farmers
Welfare',
 'Directorate of Marketing and Inspection (DMI)'],
 'org_type': 'Central',
 'records': [{'arrival_date': '04/10/2019',
 'commodity': 'Tomato',
```

```

        'district': 'Chittor',
        'market': 'Kalikiri',
        'max_price': '2000',
        'min_price': '1000',
        'modal_price': '1660',
        'state': 'Andhra Pradesh',
        'timestamp': '1570178103',
        'variety': 'Local']},
'sector': ['Agriculture', 'Agricultural Marketing'],
'source': 'data.gov.in',
'status': 'ok',
'target_bucket': {'field': '9ef84268-d588-465a-a308-
a864a43d0070',
        'index': 'daily_mandi',
        'type': '6141ea17-a69d-4713-b600-0a43c8fd9a6c'}},
'title': 'Current Daily Price of Various Commodities from
Various Markets '
        '(Mandi)',
'total': 4156,
'updated': 1570178114,
'updated_date': '2019-10-04T14:05:14Z',
'version': '2.1.0',
'visualizable': '1'}

```

You would observe that a lot of data has been returned by the API apart from the specific daily price and commodity fields. Now, it becomes important for the pipeline to read the JSON and only keep relevant data points to be written into the CSV file. The data pipeline needs to transform the data into CSV format (that is, pandas) and write in a local file.

Let's assume that we only need the following data from the JSON file to be written in the CSV file:

```

'arrival_date': '04/10/2019',
'commodity': 'Tomato',
'district': 'Chittor',
'market': 'Kalikiri',
'max_price': '2000',
'min_price': '1000',

```

```
'modal_price': '1660',  
'state': 'Andhra Pradesh',  
'timestamp': '1570178103',  
'variety': 'Local'
```

Let's write the transform script in Python to extract the above fields into a pandas data frame:

```
#Define an empty list where the extracted data will be cached  
before being written to data frame
```

```
import pandas as pd  
arrival_date = []  
commodity = []  
district = []  
market = []  
max_price = []  
min_price = []  
modal_price = []  
state = []  
timestamp = []  
variety = []
```

```
#Extract the data from json response  
records = json_response["records"]  
print("The records in records fields are of type  
",type(records))
```

```
# Run a loop over list to extract sub-list data  
for item in records:
```

```
    arrival_date.append(item["arrival_date"])  
    commodity.append(item["commodity"])  
    district.append(item["district"])  
    market.append(item["market"])  
    max_price.append(item["max_price"])  
    min_price.append(item["min_price"])  
    modal_price.append(item["modal_price"])  
    state.append(item["state"])  
    timestamp.append(item["timestamp"])  
    variety.append(item["variety"])
```

#All the data extract via API is now stored in list. let's combine all lists into one single data frame.

```
df = pd.DataFrame({'arrival_date':arrival_date,
                  'commodity':commodity,
                  'district':district,
                  'market':market,
                  'max_price':max_price,
                  'min_price':min_price,
                  'modal_price':modal_price,
                  'state':state,
                  'timestamp':timestamp,
                  'variety':variety
                  })
```

#Preview the data frame

```
df.head()
```

The records in records fields are of type <class 'list'>

You can observe that we have transformed the raw JSON file into a clean tabular structure that can be written into a CSV file as required by our pipeline objectives. You can control how many records you want from the API by setting a limit in the GET request.

We can write the data frame as a CSV file by using native pandas function: `to_csv`. We append the name of the file with a timestamp for easy recognition of the time, during which the file has been created:

#Write the file now with timestamp suffixed

```
import time
```

```
timestamp = time.strftime('%d%m%Y%H%M%S')
```

```
df.to_csv('Daily_Commodity_Prices_{}.csv'.format(timestamp),
index = False)
```

#Check if the file has been written and print pipeline success message

```
import os.path
```

```
from os import path
```

```
if
```

```
path.exists('Daily_Commodity_Prices_{}.csv'.format(timestamp)):
    print("The pipeline was executed successfully")
```

```
else:  
    print("The pipeline could not execute")  
The pipeline was executed successfully
```

The pipeline is successfully executed, and you can see the CSV file is created in the present working directory. This completes a simple data pipeline to load data from the *data.gov.in* website and store it as a CSV file.

ETL vs. ELT

We discussed ETL in the introduction section, and in the previous example, we saw a basic example of a data pipeline that does the ETL process on API data. It gets data from API source (*data.gov.in*), transforms it into required fields, and then load into a CSV file on the local disk.

ETL process does the transformation before loading the data. That step requires a clear understanding of end use of data. For instance, in our example, we wanted to monitor commodity prices and hence only extracted that information. There may be a lot of other use cases from the same data, which we do not know right now. In such cases, it's better to load the data into a persistent store and let the applications transform it as per the requirement. [Figure 9.3](#) shows the flow of an ELT data pipeline:

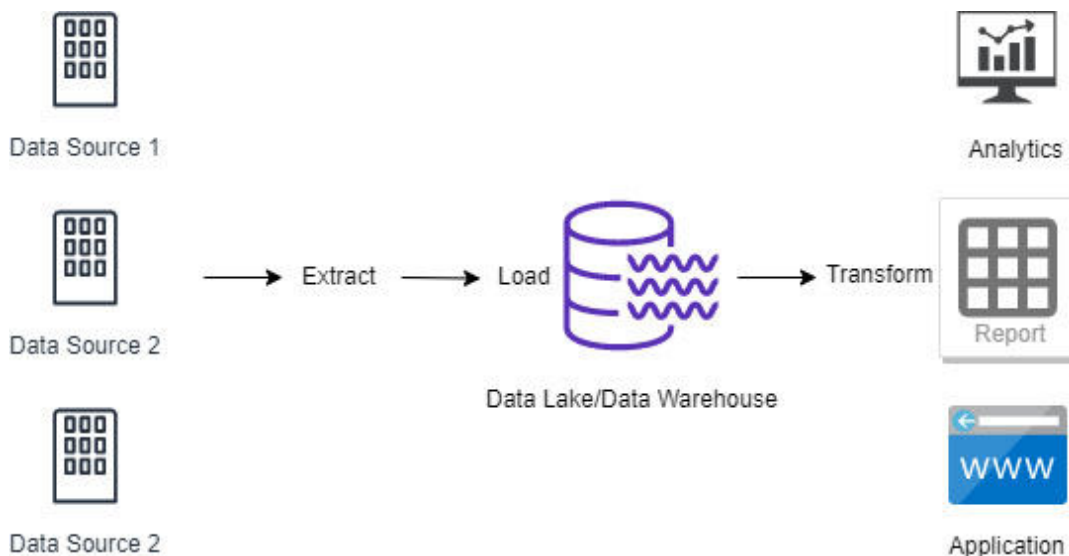


Figure 9.3: Extract Load Transform (ELT) Data Pipeline Flow

In modern data pipelines, the transform step is delayed until the end use is defined and accessed by the application. This way, the data can be stored in

raw form, and later as per requirements, it can be transformed. This new process is named the ELT process, where the order of data process is extracted, loaded, and transformed by the applications. The concept of *Data Lake* follows the ELT logic of data flow in enterprises.

We will discuss more on data storage in the next chapter, and the following chapter will talk about big data systems and how they facilitate the creation of Data Lakes.

Scheduling jobs

Scheduler, as the name suggests, is a process that can periodically execute the process, that is, jobs. This is an important concept in data pipelines as the data pipelines need to be triggered by events. In the above example, you can observe that we are able to fetch the data from data.gov.in when we run this script. But how will you be able to get the same data on an hourly basis?

The solution is certainly not manually running the script every 1 hour. This exact problem is solved by a Scheduler. A job is a set of tasks that can be executed by a Scheduler. A Scheduler is a software product that allows an enterprise to schedule and track computer batch tasks.

Let's design our example pipeline to execute the task every hour using Cron jobs. Cron is a software utility that allows us to schedule tasks on Unix-like systems. The tasks in Cron are defined in a crontab, which is a text file containing the commands to be executed.

Cron requires the frequency of running the task in the following units.

```
# _____ minute (0 - 59)
# | _____ hour (0 - 23)
# | | _____ day of the month (1 - 31)
# | | | _____ month (1 - 12)
# | | | | _____ day of the week (0 - 6) (Sunday to Saturday;
# | | | | | 7 is also Sunday on some systems)
# | | | | |
# | | | | |
# | | | | |
# * * * * * command to execute
```

We will now install the crontab library in Python and schedule the jobs to run the data pipeline script every 1 minute and check if the CSV file is automatically written at every 1-minute interval. There are 5 methods of using a crontab library; three works on Linux and two can work in windows as well. Read more details here:

```
!pip install CronTab
Requirement already satisfied: CronTab in c:\anaconda3\lib\site-
packages (0.22.6)
from crontab import CronTab

file_cron = CronTab(tabfile='filename.tab')

cron = CronTab()
job = cron.new(command='python
Code_9_1_Designing_Data_Pipeline.py')
job.minute.every(1)

cron.write()
```

The above script will run the Code_9_1_Designing_Data_Pipeline.py script at every 1-minute interval. After setting this up, the data pipeline Code_9_1_Designing_Data_Pipeline.py will be triggered by the scheduler, and results will be stored as a CSV file in the directory. Check your operating system to run cron jobs in the kernel.

[Messaging queue](#)

In many cases, the trigger of the data pipeline is not just a particular time but could be other events as well. For example, we may want to get commodity price data whenever it rains, or we may want data when a new data gets updated, and more. In all such cases, the trigger to run data pipeline can come asynchronously and must allow you to run the data pipeline job either in real-time or as per some queuing logic.

A messaging queue mechanism allows the queuing of all such triggers (messages) and runs the desired application. The working of the messaging queue is simple; there are clients that create messages, called producers, and deliver them to the queuing system. Another set of applications called consumers, connects to the queue and gets the message processed by

desired/targeted applications. [Figure 9.4](#) shows the flow of the Messaging Queue components:



Figure 9.4: Messaging Queue Components

The above example is a simple illustration of the messaging queue. Assume an example where we allow a user to fetch the data from the *data.gov.in* website on demand. The producer will create the requests and puts them in the queue, waiting for the consumer to trigger the pipeline and brings data as per user demands. This way, the system allows services to keep in touch with each other without getting blocked by responses. The consumer will fulfill the request in the order they have come and fulfill all requests. A good tutorial to learn about queues can be found from a leading messaging queue system, RabbitMQ.

Passing arguments to data pipeline

In our example, you would observe that the data pipeline does not need any argument from the user and will produce the same result every time it runs, except when the data is changed at source. In general, the data pipeline allows the user to pass custom arguments to run the pipeline to produce results that are desired.

For instance, in our example, you can see that you can pass an argument in the API call to limit the number of results to fetch from the source. Having arguments that define the number of values to fetch will allow the data pipeline to behave as per the arguments passed to the data pipeline. In our example, to incorporate such behavior, we would pass arguments via the command line.

In the following example, you can see how command line arguments can be retrieved by the `sys` library in Python. Store the below script in a Python file and name it `Code_9_3_Example_sys_args.py`:

```
#Import the library
import sys
```



```
#Print the arguments passed to the script
argument_1 = sys.argv[1]
argument_2 = sys.argv[2]

print("\n The First argument passed is ", argument_1, "\n\n The
Second argument passed is ", argument_2)
```

Now, we will call the above script with two arguments; let's say Hello and 5. The script should catch the arguments and make them available as two arguments values:

```
!Python Code_9_3_Example_sys_args.py Hello 5
The First argument passed is Hello

The Second argument passed is 5
```

We now will modify our script of the data pipeline to see if it can take the limit as command line arguments and fetch as many records as required by the user. The following script is the modified script to show the use of argv in data pipeline arguments. Save the script as Code_9_4_Command_line_data-pipeline.py and run it from console to see how the results differ for two different arguments:

```
#Import the library to catch command line arguments
import sys

#store the first argument as the limit
limit = sys.argv[1]

##Python Script to Implement the Pipeline
#Define the API KEY ( this can be found in the data.gov.in
account section for registered users)
API_KEY = <YOUR API KEY>

#Import the requests library
import requests

#Construct the GET REQUEST - here pass limit as the variable got
from command line
response = requests.get(
    'https://api.data.gov.in/resource/9ef84268-d588-465a-
a308-a864a43d0070',
    params=[('api-key', API_KEY )],
```

```

        ('format','json'),
        ('offset',0),
        ('limit',limit)],
    )

```

#Check if the request was successful - a success request returns a status code 200

```
if response.status_code == 200:
```

```
    print('Success!')
```

```
else:
```

```
    print('Some Error Occurred')
```

#If the you see a success message then you can extract the values from the JSON response

```
json_response = response.json()
```

#Display the result of data pipeline run

```
import pprint
```

```
pprint.pprint(json_response)
```

Now we will call the data pipeline script for fetching 1 record and 2 records by passing the arguments. Below is the result for 1:

```
!Python Code_9_4_Command_line_data-pipeline.py 1
```

```
Success!
```

```
{'active': '1',
 'catalog_uuid': '6141ea17-a69d-4713-b600-0a43c8fd9a6c',
 'count': 1,
 'created': 1543321994,
 'created_date': '2018-11-27T18:03:14Z',
 'desc': 'Current Daily Price of Various Commodities from
Various Markets '
 '(Mandi)',
 'field': [{ 'id': 'timestamp', 'name': 'timestamp', 'type':
'double'},
            { 'id': 'state', 'name': 'state', 'type': 'keyword'},
            { 'id': 'district', 'name': 'district', 'type': 'keyword'},
            { 'id': 'market', 'name': 'market', 'type': 'keyword'},
            { 'id': 'commodity', 'name': 'commodity', 'type':
'keyword'},
```

```
{'id': 'variety', 'name': 'variety', 'type': 'keyword'},
{'id': 'arrival_date', 'name': 'arrival_date', 'type':
'date'},
{'id': 'min_price', 'name': 'min_price', 'type': 'double'},
{'id': 'max_price', 'name': 'max_price', 'type': 'double'},
{'id': 'modal_price', 'name': 'modal_price', 'type':
'double'}],
'index_name': '9ef84268-d588-465a-a308-a864a43d0070',
'limit': '1',
'message': 'Resource detail',
'offset': '0',
'org': ['Ministry of Agriculture and Farmers Welfare',
'Department of Agriculture, Cooperation and Farmers
Welfare',
'Directorate of Marketing and Inspection (DMI)'],
'org_type': 'Central',
'records': [{'arrival_date': '06/10/2019',
'commodity': 'Tomato',
'district': 'Chittor',
'market': 'Mulakalacheruvu',
'max_price': '1900',
'min_price': '550',
'modal_price': '1500',
'state': 'Andhra Pradesh',
'timestamp': '1570383302',
'variety': 'Local'}],
'sector': ['Agriculture', 'Agricultural Marketing'],
'source': 'data.gov.in',
'status': 'ok',
'target_bucket': {'field': '9ef84268-d588-465a-a308-
a864a43d0070',
'index': 'daily_mandi',
'type': '6141ea17-a69d-4713-b600-0a43c8fd9a6c'},
'title': 'Current Daily Price of Various Commodities from
Various Markets '
'(Mandi)',
'total': 1823,
```

```
'updated': 1570383307,  
'updated_date': '2019-10-06T23:05:07Z',  
'version': '2.1.0',  
'visualizable': '1'}
```

Now we pass the argument as 2 and expect to see two records:

```
!Python Code_9_4_Command_line_data-pipeline.py 2
```

Success!

```
{'active': '1',  
 'catalog_uuid': '6141ea17-a69d-4713-b600-0a43c8fd9a6c',  
 'count': 2,  
 'created': 1543321994,  
 'created_date': '2018-11-27T18:03:14Z',  
 'desc': 'Current Daily Price of Various Commodities from  
Various Markets '  
 '(Mandi)',  
 'field': [{'id': 'timestamp', 'name': 'timestamp', 'type':  
'double'},  
 {'id': 'state', 'name': 'state', 'type': 'keyword'},  
 {'id': 'district', 'name': 'district', 'type': 'keyword'},  
 {'id': 'market', 'name': 'market', 'type': 'keyword'},  
 {'id': 'commodity', 'name': 'commodity', 'type':  
'keyword'},  
 {'id': 'variety', 'name': 'variety', 'type': 'keyword'},  
 {'id': 'arrival_date', 'name': 'arrival_date', 'type':  
'date'},  
 {'id': 'min_price', 'name': 'min_price', 'type': 'double'},  
 {'id': 'max_price', 'name': 'max_price', 'type': 'double'},  
 {'id': 'modal_price', 'name': 'modal_price', 'type':  
'double'}],  
 'index_name': '9ef84268-d588-465a-a308-a864a43d0070',  
 'limit': '2',  
 'message': 'Resource detail',  
 'offset': '0',  
 'org': ['Ministry of Agriculture and Farmers Welfare',  
 'Department of Agriculture, Cooperation and Farmers  
Welfare',  
 'Directorate of Marketing and Inspection (DMI)'],
```

```
'org_type': 'Central',
'records': [{ 'arrival_date': '06/10/2019',
  'commodity': 'Tomato',
  'district': 'Chittoor',
  'market': 'Mulakalacheruvu',
  'max_price': '1900',
  'min_price': '550',
  'modal_price': '1500',
  'state': 'Andhra Pradesh',
  'timestamp': '1570383302',
  'variety': 'Local'},
{ 'arrival_date': '06/10/2019',
  'commodity': 'Beetroot',
  'district': 'Chittoor',
  'market': 'Palamaner',
  'max_price': '3000',
  'min_price': '1000',
  'modal_price': '2000',
  'state': 'Andhra Pradesh',
  'timestamp': '1570383302',
  'variety': 'Beetroot'}],
'sector': ['Agriculture', 'Agricultural Marketing'],
'source': 'data.gov.in',
'status': 'ok',
'target_bucket': {'field': '9ef84268-d588-465a-a308-a864a43d0070',
  'index': 'daily_mandi',
  'type': '6141ea17-a69d-4713-b600-0a43c8fd9a6c'},
'title': 'Current Daily Price of Various Commodities from
Various Markets '
  '(Mandi)',
'total': 1823,
'updated': 1570383307,
'updated_date': '2019-10-06T23:05:07Z',
'version': '2.1.0',
'visualizable': '1'}
```

Now, you can allow the client to control how the data should be fetched using arguments to customize the data pipeline flow. These features make the data pipeline more robust, useful, and adaptable to the needs of the applications. These same pipelines can be built for the streaming process as well by using the message broker system like RabbitMQ, Kafka, etc. tools. There are lots of tools developed by cloud computing giants to automate the data pipeline and make the life of the developer easy. Readers are encouraged to learn more about data pipelines and read the tools at offering at GCP, AWS, and Azure cloud providers.

Conclusion

In this chapter, we have introduced the concept of the data pipeline and its importance in designing any modern-day application. The chapter then introduces the concept of the **Extract Transform Load (ETL)**. The ETL concept is an important concept within the broader domain of data pipelines. To explain concepts, we took a real example of fetching data from the *data.gov.in* website using a REST API interface and store that as a CSV file. After that example, we have talked about the significance of **Extract Load Transform (ELT)** flows for the most cases in applications, where the application does the transformation as per the need and the data pipeline extract and store data in a Data Lake. The concept of Scheduler is explained to automate the execution of a data pipeline by time frequency. Further, we have introduced the concept of message queue to handle asynchronous and on-demand run of data pipelines. The last section gives a brief understanding of the importance of passing arguments to the data pipeline to make it more adoptable to the use case. The concept is explained by showing an example of limiting the count of data fetch by getting a limit argument from the command line.

By reading this chapter, the reader will be able to gain the understanding of the use of data pipelines in the implementation of AI/ML models. He or she will also be able to figure out the difference between ETL and ELT. The reader can now design a simple data pipeline for a model.

In the next chapter, we will discuss databases and how to use them to store data. The same example will be extended to show how the data pipeline can store data into a database and then retrieve it for application needs.

CHAPTER 10

Introduction to Databases

Databases are important applications which store data for optimized data storage for faster retrieval and efficient data addition. In the previous chapter, we discussed Data in Motion, where the data was flowing from one source to another and in between getting transformed and manipulated by the data pipeline. The data finally ended up as a CSV file in local storage.

CSV storage is not an efficient way to store data as it is not stored by any logic but simply dumped by the stream of data coming in by the data pipeline. It is thus very important to store the data in an optimized way by using databases. In this chapter, we will introduce databases and use the same data.gov.in example to show how to operate with the databases. Using the example, we will show how the data pipeline can store data into the database and then retrieve it for application needs.

Structure

- Modern databases and terminology
- Relational database or SQL database
- Document oriented database or SQL database
- Graph databases
- Filesystem as storage

Objectives

After studying this unit, you should be able to:

- Understand the concept of databases
- Setup SQL and No-SQL databases
- Use Neo4j to understand graph databases

- Manage data storage and Indexing using Filesystem

Modern databases and terminology

The database is an organized way of collecting and retrieval of data through electronic means by a computer system in various ways. The database allows efficient storage of huge amounts of data having an inherent nature of search-ability. In modern times, the database technology has evolved a lot, and now it supports modern applications with TBs of data being stored and retrieved in near real-time. [Figure 10.1](#) shows the survey of the databases, ranked in terms of popularity, as provided by DB-Engine:

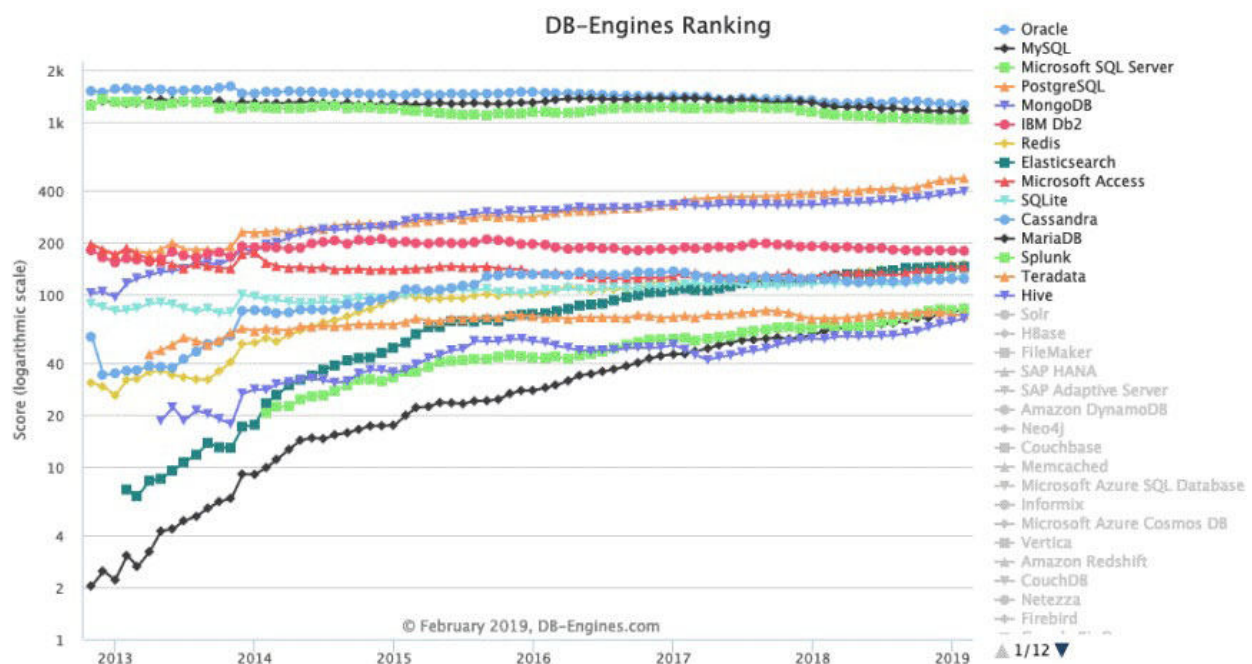


Figure 10.1: DB-Engine Rankings (Credit: DB-Engines.com)

The DB-engine survey points out the lead in SQL databases, which is Oracle, and next in the lead are MongoDB and Redis, which are the No-SQL database. There are other feature-rich databases that are suited for specific purpose and performance needs like graph databases, elastic search, and others. In this chapter, we would introduce only SQL, No-SQL, and graph databases. The domain of databases is a huge domain and provides a separate track of expertise in Information Technology. The reader, who is aiming to become database designers and DB admins, is encouraged to get into the depth of database technologies.

There is some key terminology that will help us understand this world of databases. We would provide you a plain English definition and encourage the reader to read more.

- **Query:** A query defines a single transaction on the database. It can be an update, insert, select, or delete operation.
- **Structure Query Language (SQL):** SQL defines standard rules to write a query for operating on a relational database. The databases that allow SQL types syntax to access databases are generally called SQL databases.
- **Relational databases:** In relational databases, the data is organized as a table, and it allows us to identify and access data in relation to another piece by columnar relations.
- **No-SQL:** No-SQL means a non-relational database. In this case, the data is stored without strict schema and can be stored as a document with semi-structured data.
- **Document-oriented databases:** Document oriented databases store, retrieve, and managing document-oriented information, also known as semi-structured data. These groups of databases are also called No-SQL databases.
- **ACID properties:** Atomicity, Consistency, Isolation, Durability
 - **Atomicity:** Each transaction is a unique, atomic unit of work. If one operation fails, data remains unchanged.
 - **Consistency:** All data written to the database is subject to any rules defined. When completed, a transaction must leave all data in a consistent state.
 - **Isolation:** Changes made in a transaction are not visible to other transactions until they are complete.
 - **Durability:** Changes completed by a transaction are stored and available in the database, even in the event of a system failure.
- **Database Management System (DBMS):** A DBMS provides the user all the features for definition, creation, querying, update, and administration of databases. MySQL, PostgreSQL, MongoDB, Neo4j, MS SQL are some examples of popular DBMS systems.

- **Schema:** A database schema is a logical structure of how the database is constructed and relates data points to each other via tables, indexes, etc. Static schema is defined before the program is written and hence, can only hold the data as per the schema, while dynamic schema allows the incoming data to construct schema to accommodate the incoming data.
- **JOIN operations:** The join operations allow combining different datasets by some defined logic or rules, using indexes, combined indexes. Their multiple types of joins defined with the DBMS system and vary as per the database. The generic joins are LEFT, RIGHT, INNER and OUTER join. The concept of joins is typically used in relational databases. The reader, must read details about joins, as they are very key concepts in databases.
- **Object-Relational Mapping (ORM):** ORM is the set of techniques used for translating the logical representation of objects (as in object-oriented programming) into a more atomized form that is capable of being stored in a relational database (and back again when they are retrieved). The concept of ORM is very relevant to our examples in the next sections, as the data pipeline written in Python needs to interact with the databases as the ORMs provide that bridge to allow us to operate databases using ORMs in programming languages. More details here <https://www.fullstackpython.com/object-relational-mappers-orms.html>.

In the above set of key terminologies, we tried to just scratch the surface to provide the starting point for further studies into databases. There are numerous resources for databases publicly available for study and implementations with various programming languages. With the advent of cloud computing, the databases have also adopted cloud-native forms and provide a very powerful system for applications on the cloud.

[Relational database or SQL database](#)

Relational databases are the most popular databases existing in the informational technology world. They represent data as tables and connect data in different tables by foreign keys and define each row in each table by a unique key. A very good resource to learn about important concepts in SQL can be found at w3school here (<https://www.w3schools.com/sql/>).

With relation to our scope of the book of using databases to store and retrieve the data within our data science applications, we would demonstrate how you can store and retrieve data from the PostgreSQL database and integrate that into our data pipeline in data.gov.in example.

[Install PostgreSQL and pgAdmin](#)

PostgreSQL open-source SQL based database, which is very popular and power many enterprise scale applications. In terms of syntax and terminology, it is very similar to Oracle, MS SQL, and MySQL databases. For open-source development, Postgres is most popular.

You can install Postgres from the official website. Visit this web page <https://www.postgresql.org/download/> and follow the instructions.

To have access to the database, we need some clients as well. pgAdmin is one such client to access the Postgres database. You can have any other SQL compliant client as well to manage the database. You can download and install pgAdmin from here <https://www.pgadmin.org/download/>.

Note: Read the documentation before installation as some version of pgAdmin may not be compatible with older Postgres Db engines.

Once installed, you can start the client window and see a window, similar to [Figure 10.2](#):

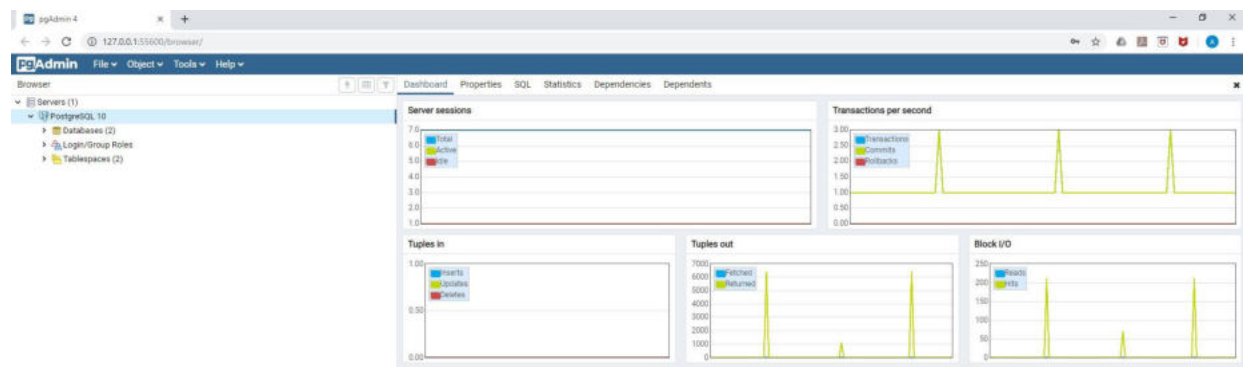


Figure 10.2: PGAdmin Window

If you have difficulty in installing the set-up, you can refer to the YouTube tutorial here <https://www.youtube.com/watch?v=e1MwsT5EJRQ>.

[Set-up a database and table](#)

In Postgres, data is stored in tables, and the tables are created within a database. So, we first need to create a database to store our daily commodity price data (refer data.gov.in example) and then create a table to store the data.

Let's create a database name data-gov-in and a table name commodity_prices with a schema same as dataframe/csv. [Figure 10.3](#) shows the window snippet of creating database in Postgres:

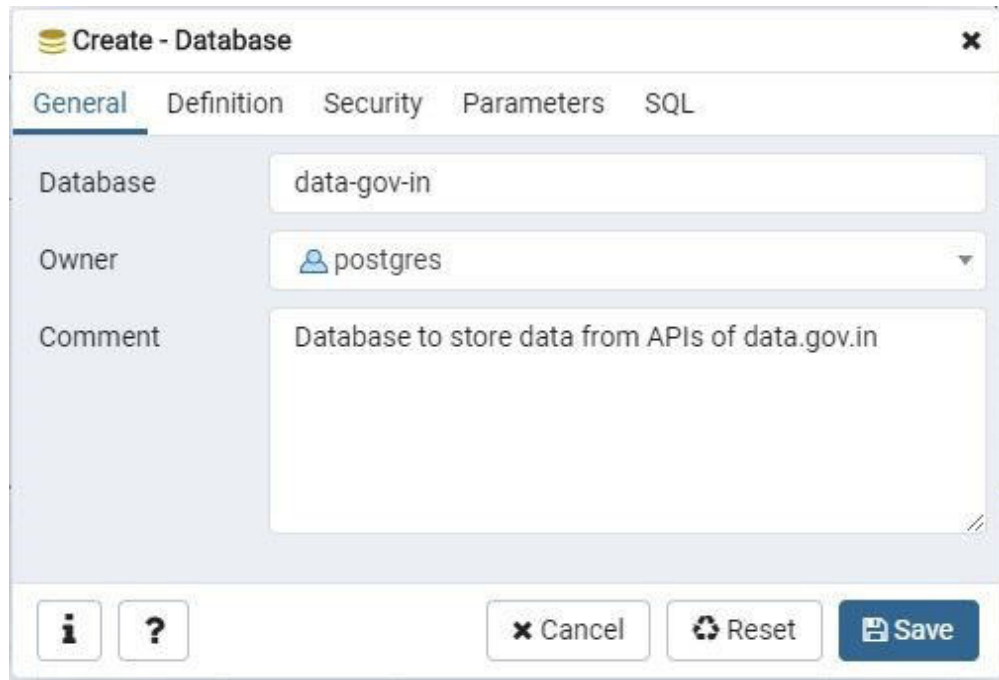


Figure 10.3: Creating database in pgAdmin

After creating the Database, create a table with following schema:

```
'field': [{'id': 'timestamp', 'name': 'timestamp', 'type':  
'double'},  
          {'id': 'state', 'name': 'state', 'type': 'keyword'},  
          {'id': 'district', 'name': 'district', 'type':  
            'keyword'},  
          {'id': 'market', 'name': 'market', 'type': 'keyword'},  
          {'id': 'commodity', 'name': 'commodity', 'type':  
            'keyword'},  
          {'id': 'variety', 'name': 'variety', 'type':  
            'keyword'}],
```

```
{'id': 'arrival_date', 'name': 'arrival_date', 'type':
'date'},
{'id': 'min_price', 'name': 'min_price', 'type':
'double'},
{'id': 'max_price', 'name': 'max_price', 'type':
'double'},
{'id': 'modal_price', 'name': 'modal_price', 'type':
'double'}],
```

[Figure 10.4](#) shows the window snippet of a table created in pgAdmin:

The screenshot shows the 'Create - Table' window in pgAdmin, specifically the 'Columns' tab. The window has tabs for General, Columns, Constraints, Advanced, Partition, Parameters, Security, and SQL. The 'Columns' tab is active, showing a table with 10 columns. The columns are: timestamp, state, district, market, commodity, variety, arrival_date, min_price, max_price, and modal_price. Each column has a data type, length, precision, and 'Not NULL?' and 'Primary key?' checkboxes. The 'Not NULL?' and 'Primary key?' checkboxes are all set to 'No'.

Name	Data type	Length	Precision	Not NULL?	Primary key?
timestamp	double precision			No	No
state	character varying	128		No	No
district	character varying	128		No	No
market	character varying	128		No	No
commodity	character varying	128		No	No
variety	character varying	128		No	No
arrival_date	character varying	128		No	No
min_price	double precision			No	No
max_price	double precision			No	No
modal_price	double precision			No	No

Figure 10.4: Creating Table in pgAdmin

Now we have the database and table ready to use in our data pipeline to store and retrieve data. As our pipeline is written in Python, first we have to find

out a suitable ORM to interact with Postgre database. We will be using SQLAlchemy to connect our Python script to database.

Connect Python to Postgres

SQLAlchemy consists of two distinct components, known as the Core and the ORM. An illustration of how it will be used is shown below:

```
#Import the SQLAlchemy Library
from sqlalchemy import create_engine
# Configuration String
#dialect+driver://username:password@host:port/database
db_string = "postgresql://postgres:postgres@localhost:5432/data-
gov-in"

db = create_engine(db_string)

# Insert a Records into table
db.execute("INSERT INTO public.commodity_prices(timestamp,
state, district, market, commodity, variety, arrival_date,
min_price, max_price, modal_price) VALUES
(1570178103, '04/10/2019', 'Andhra
Pradesh', 'Chittor', 'Tomato', 'Kalikiri', 'Local', 2000,
1000, 1660);")

# Read
result_set = db.execute("SELECT * FROM public.commodity_prices")
for r in result_set:
    print(r)

# Update
db.execute("UPDATE public.commodity_prices SET min_price= 980
WHERE commodity='Tomato'")

# Delete
db.execute("DELETE FROM public.commodity_prices WHERE
commodity='Tomato'")
(1570178103.0, '04/10/2019', 'Andhra Pradesh', 'Chittor',
'Tomato', 'Kalikiri', 'Local', 2000.0, 1000.0, 1660.0)
```

You can read more about SQLAlchemy here <https://www.sqlalchemy.org/>.

Modify data pipeline to store in Postgres

While the reader can play with database and set-up on their own, we would quickly edit our data pipeline script to write the data to database table `commodity_prices`. The following change is being done to achieve the same. The new file is named as `Code_10_1_Data_pipeline_sql.py`. `Code_10_1_Data_pipeline_sql.py` is the code to write the data frame into a database:

```
##Once the data frame df is ready - we open a database
connection and write the data frame into database
from sqlalchemy import create_engine
db_string = "postgresql://postgres:postgres@localhost:5432/data-
gov-in"

db = create_engine(db_string)

#Write data to data table
#if_exists: {'fail', 'replace', 'append'}, default 'fail'
#How to behave if the table already exists.
#fail: Raise a ValueError.
#replace: Drop the table before inserting new values.
#append: Insert new values to the existing table.

df.to_sql('commodity_prices', con=db,if_exist = 'replace')

#Retrieve data to see if got written properly
result_set = db.execute("SELECT * FROM public.commodity_prices")
for r in result_set:
    print(r)
```

Now let's run the data pipeline code to see if the database gets updated with the latest commodity prices.

```
!python Code_10_1_Data_pipeline_sql.py
Success!
```

The records in records fields are of type `<class 'list'>`:

```
(0, '1570437302', 'Andhra Pradesh', 'Chittor', 'Kalikiri',
'Tomato', 'Local', '07/10/2019', '530', '1330', '1000')
(1, '1570437302', 'Andhra Pradesh', 'Chittor',
'Mulakalacheruvu', 'Tomato', 'Local', '07/10/2019', '550',
```

'1900', '1500')
(2, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Beetroot', 'Beetroot', '07/10/2019', '2000', '4000', '3000')
(3, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Bhindi(Ladies Finger)', 'Bhindi', '07/10/2019', '500', '1000',
'750')
(4, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Bitter gourd', 'Bitter Gourd', '07/10/2019', '1250', '2250',
'1750')
(5, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Bottle gourd', 'Bottle Gourd', '07/10/2019', '250', '500',
'375')
(6, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Brinjal', 'Brinjal', '07/10/2019', '750', '1750', '1250')
(7, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Cabbage', 'Cabbage', '07/10/2019', '250', '750', '500')
(8, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Carrot', 'Carrot', '07/10/2019', '625', '1125', '875')
(9, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Cauliflower', 'Cauliflower', '07/10/2019', '600', '1200',
'900')
(10, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Cluster beans', 'Cluster Beans', '07/10/2019', '2000', '4000',
'3000')
(11, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Cucumber(Kheera)', 'Cucumbar', '07/10/2019', '1000', '2000',
'1500')
(12, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Green Chilli', 'Green Chilly', '07/10/2019', '1000', '3000',
'2000')
(13, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Potato', '(Red Nanital)', '07/10/2019', '1000', '2000', '1500')
(14, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Raddish', 'Raddish', '07/10/2019', '750', '1750', '1250')
(15, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Ridgeguard(Tori)', 'Ridgeguard(Tori)', '07/10/2019', '750',
'1250', '1000')


```
(16, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Sweet Potato', 'Sweet Potato', '07/10/2019', '600', '1000',
'800')
(17, '1570437302', 'Andhra Pradesh', 'Chittor', 'Palamaner',
'Tomato', 'Hybrid', '07/10/2019', '200', '1200', '600')
(18, '1570437302', 'Andhra Pradesh', 'Chittor', 'Piler',
'Tomato', 'Deshi', '07/10/2019', '500', '1000', '750')
(19, '1570437302', 'Andhra Pradesh', 'Chittor', 'Vayalapadu',
'Tomato', 'Local', '07/10/2019', '400', '2040', '1200')
```

Below [Figure 10.5](#) is the snapshot from the pgAdmin client GUI with above created table:

	index bigint	timestamp text	state text	district text	market text	commodity text	variety text	arrival_date text	min_price text	max_price text	modal_price text
1	0	1570437302	Andhr...	Chittor	Kalikiri	Tomato	Local	07/10/2019	530	1330	1000
2	1	1570437302	Andhr...	Chittor	Mulakala...	Tomato	Local	07/10/2019	550	1900	1500
3	2	1570437302	Andhr...	Chittor	Palaman...	Beetroot	Beetroot	07/10/2019	2000	4000	3000
4	3	1570437302	Andhr...	Chittor	Palaman...	Bhindi(Ladies...	Bhindi	07/10/2019	500	1000	750
5	4	1570437302	Andhr...	Chittor	Palaman...	Bitter gourd	Bitter G...	07/10/2019	1250	2250	1750
6	5	1570437302	Andhr...	Chittor	Palaman...	Bottle gourd	Bottle G...	07/10/2019	250	500	375
7	6	1570437302	Andhr...	Chittor	Palaman...	Brinjal	Brinjal	07/10/2019	750	1750	1250
8	7	1570437302	Andhr...	Chittor	Palaman...	Cabbage	Cabbage	07/10/2019	250	750	500
9	8	1570437302	Andhr...	Chittor	Palaman...	Carrot	Carrot	07/10/2019	625	1125	875
10	9	1570437302	Andhr...	Chittor	Palaman...	Cauliflower	Cauliflo...	07/10/2019	600	1200	900
11	10	1570437302	Andhr...	Chittor	Palaman...	Cluster beans	Cluster ...	07/10/2019	2000	4000	3000
12	11	1570437302	Andhr...	Chittor	Palaman...	Cucumbar(Kh...	Cucumb...	07/10/2019	1000	2000	1500
13	12	1570437302	Andhr...	Chittor	Palaman...	Green Chilli	Green C...	07/10/2019	1000	3000	2000
14	13	1570437302	Andhr...	Chittor	Palaman...	Potato	(Red Na...	07/10/2019	1000	2000	1500
15	14	1570437302	Andhr...	Chittor	Palaman...	Raddish	Raddish	07/10/2019	750	1750	1250
16	15	1570437302	Andhr...	Chittor	Palaman...	Ridgeguard(T...	Ridgegu...	07/10/2019	750	1250	1000
17	16	1570437302	Andhr...	Chittor	Palaman...	Sweet Potato	Sweet P...	07/10/2019	600	1000	800
18	17	1570437302	Andhr...	Chittor	Palaman...	Tomato	Hybrid	07/10/2019	200	1200	600
19	18	1570437302	Andhr...	Chittor	Piler	Tomato	Deshi	07/10/2019	500	1000	750
20	19	1570437302	Andhr...	Chittor	Vayalapa...	Tomato	Local	07/10/2019	400	2040	1200

Figure 10.5: Created Table in pgAdmin

Now let's discuss on No-SQL databases in the upcoming section.

Document-oriented database or No-SQL

No-SQL databases are getting very popular in the last two decades, and the main advantage of using the No-SQL database is the ability to store unstructured data and object-oriented programming, which makes it easy to code in OOPs scripting languages. You can read about the benefits and features here <https://www.mongodb.com/scale/advantages-of-nosql>.

The No-SQL database has a flexible schema and stores the data in document format, where each document may have semi-structure data. You can learn more about No-SQL databases and how to use them at w3schools here <https://www.w3schools.in/mongodb/introduction-to-nosql/>. To establish the use case in the data pipeline, we will show the example of data.gov.in to store the data into a MongoDB, a leading No-SQL database. In this case, we would see that we need not convert unstructured JSON file into tabular data frame, but we can directly store the whole document.

Install MongoDB and compass client

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. We would install the community edition to show the example. You can follow instructions to install the MongoDB here <https://docs.mongodb.com/v3.2/installation/>.

We will also need a client to access the database in case we want to manage the database outside the python environment as well. This is always the case as database administration is easy with GUI based clients. The client we will be using with Mongo DB is called MongoDB Compass, and you can install the same from this link, <https://docs.mongodb.com/compass/master/install/>. Once installed with default settings, you can see below a similar window, as shown in [Figure 10.6](#):

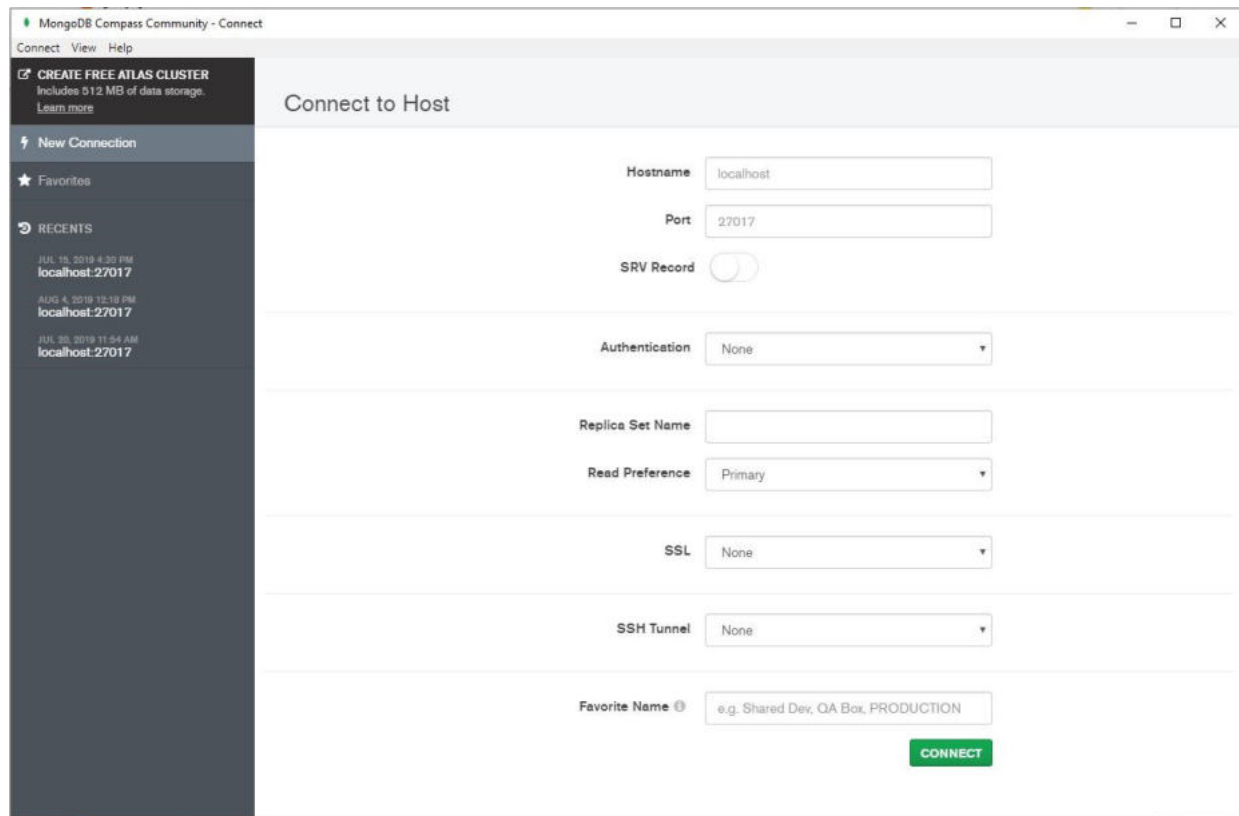


Figure 10.6: MongoDB Compass Window

You can follow instructions from this video tutorial here <https://www.youtube.com/watch?v=lu7YXYbLmdM>.

Create a database and collection

To start using the Mongo DB database, we need to create a database. The database can be created from the Mongo DB Compass GUI as well. During database creation, we have to define the collection as well. A collection is analogous to a data table in the SQL database. [Figure 10.7](#) is the snapshot of the window creating the database in Mongo DB Compass:

More Information'. At the bottom are 'CANCEL' and 'CREATE DATABASE' buttons."/>

Create Database

Database Name

data-gov-in

Collection Name

commodity_prices

☐ Capped Collection ⓘ

☐ Use Custom Collation ⓘ

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

CANCEL CREATE DATABASE

Figure 10.7: Creating Database in MongoDB Compass

We will create a database of name data-gov-in and collection as commodity_prices, similar to our SQL example.

Connect Python to MongoDB

To connect python to MongoDB, we will use a library name pymongo. Pymongo provides the ORM layer to document type interactions between Python and MongoDB. We will use the following document structure to store data.

This code describes the use of PyMongo library in python and store data in the database:

```
#Import the MongoClient from PyMongo Library
from pymongo import MongoClient
```

```
#Define the connection string to your MongoDB instance
# You remember we did not set-up any password while installing
MongoDB
```

```
client = MongoClient('mongodb://localhost:27017')
```

```
#Access the Database
```

```
db = client['data-gov-in']
```

```
#Access the Collection
```

```
posts = db.commodity_prices
```

```
#Single data to be pushed to the database
```

```
post_data = {'arrival_date': '04/10/2019',
             'commodity': 'Tomato',
             'district': 'Chittor',
             'market': 'Kalikiri',
             'max_price': '2000',
             'min_price': '1000',
             'modal_price': '1660',
             'state': 'Andhra Pradesh',
             'timestamp': '1570178103',
             'variety': 'Local'}
```

```
#Insert the first commodity price data into database
```

```
result = posts.insert_one(post_data)
```

```
#Print the inserted_id to display the unique document reference
and a success in insertion
```

```
print('First Commodity Price: {0}'.format(result.inserted_id))
```

```
First Commodity Price: 5d9b52248e19fd632e2216cd
```

Once successfully inserted, the data can be seen in the Mongo DB compass as well. We can retrieve this data back into Python by using the `find_one` method. [Figure 10.8](#) is the snapshot of the data inserted in the Mongo DB Compass:

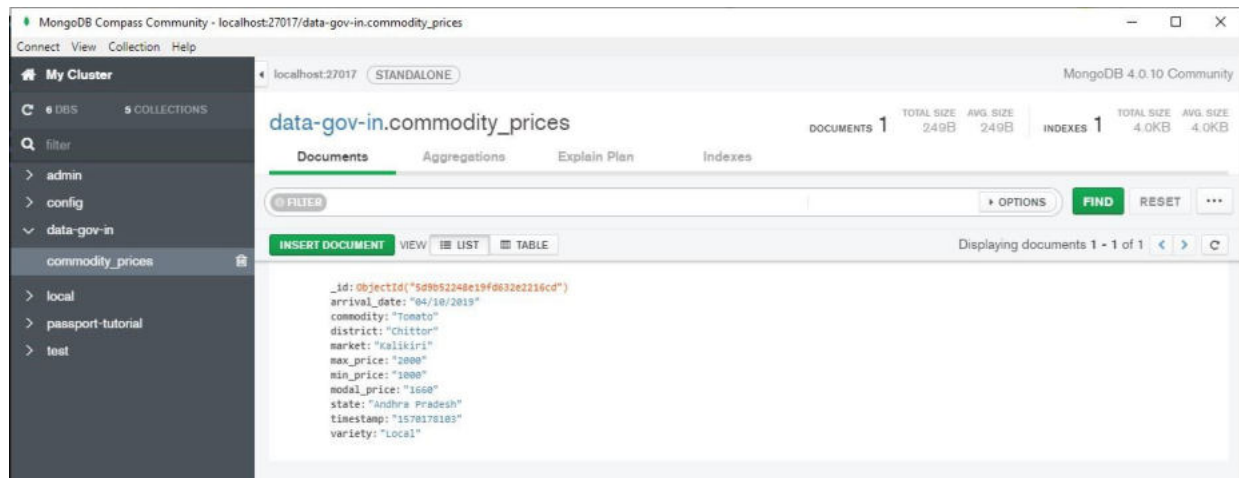


Figure 10.8: Data inserted in MongoDB Compass

The below code snippet is to retrieve a post and print it:

```
#Retrieve a post with district as Chittor
get_post = posts.find_one({'district': 'Chittor'})

#Print the retrieved document
print(get_post)
{'_id': ObjectId('5d9b52248e19fd632e2216cd'), 'arrival_date':
'04/10/2019', 'commodity': 'Tomato', 'district': 'Chittor',
'market': 'Kalikiri', 'max_price': '2000', 'min_price': '1000',
'modal_price': '1660', 'state': 'Andhra Pradesh', 'timestamp':
'1570178103', 'variety': 'Local'}
```

In the next section, we will modify the data pipeline to store in MongoDB.

Modify data pipeline to store in MongoDB

As you can observe, the Mongo DB can store the whole document in JSON format into the database, we need not worry about transformations. If we store the whole incoming stream of JSON to MongoDB, we can retrieve the data later with ease from the database. Though we can store exactly the same data as Postgres, however, we make use of MongoDB properties to store the whole JSON in this case.

We will modify the code with following to allow writing the data into MongoDB rather than CSV and store the file as `Code_10_2_Data_pipeline_nosql.py`:

```
#Import the MongoClient from PyMongo Library
from pymongo import MongoClient
#Define the connection string to your MongoDB instance
# You remember we did not set-up any password while installing
MongoDB

client = MongoClient('mongodb://localhost:27017')

#Access the Database
db = client['data-gov-in']

#Access the Collection
posts = db.commodity_prices

#Insert the first commodity price data into database
result = posts.insert_one(response)
!python Code_10_2_Data_pipeline_nosql.py
Success!
```

You can see that the database returned a success message, which means our data pipeline has successfully written the JSON data into the database. [Figure 10.9](#) shows how the new data looks like in MongoDB compass:

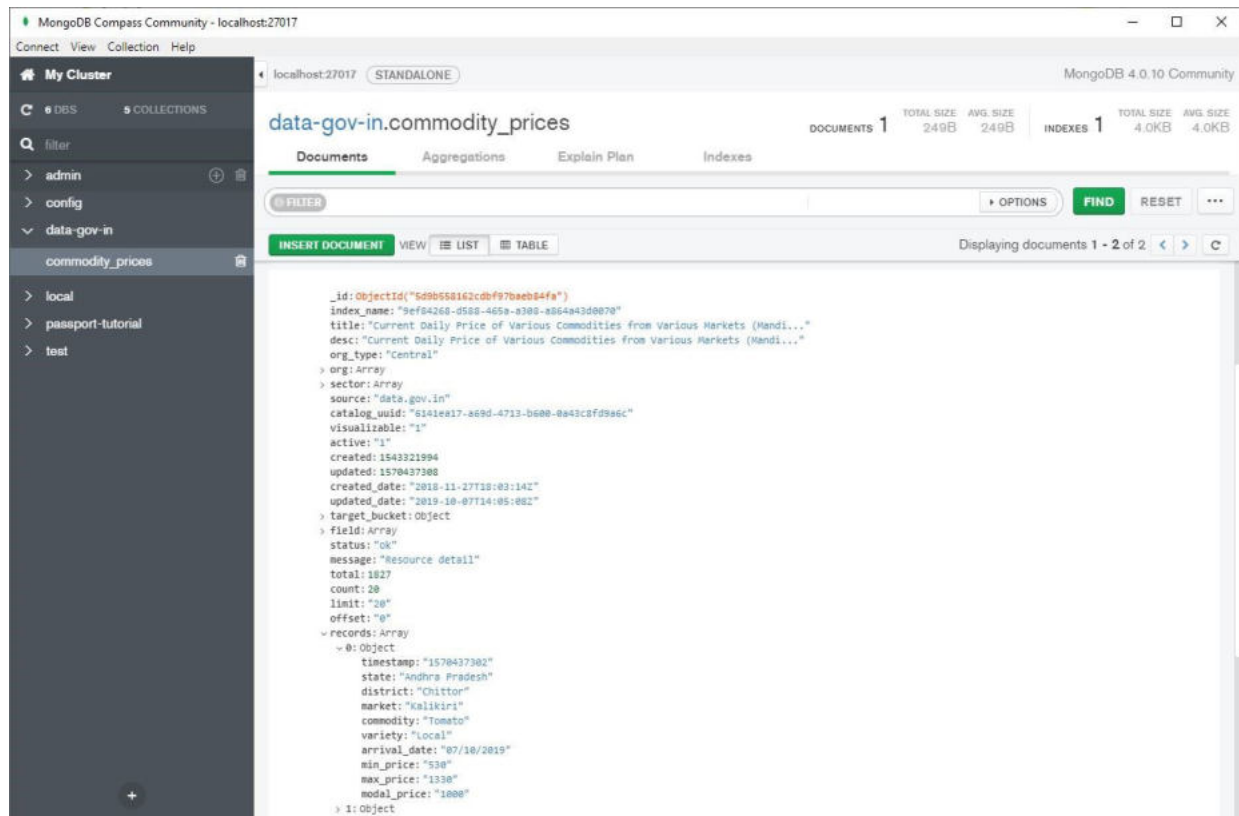


Figure 10.9: Modified data in MongoDB

This concludes the discussion on No-SQL databases. In the next section, we will discuss on the Graph databases.

Graph databases

Graph database is based on nodes and their relationships. The relations are defined by the edges, and the nodes define the entities having those relationships. The ability to directly state the relationship between two objects within a graph database provides a big advantage over relational database systems for datasets having a lot of relationships within them.

[Figure 10.10](#) is an example of a graph database:

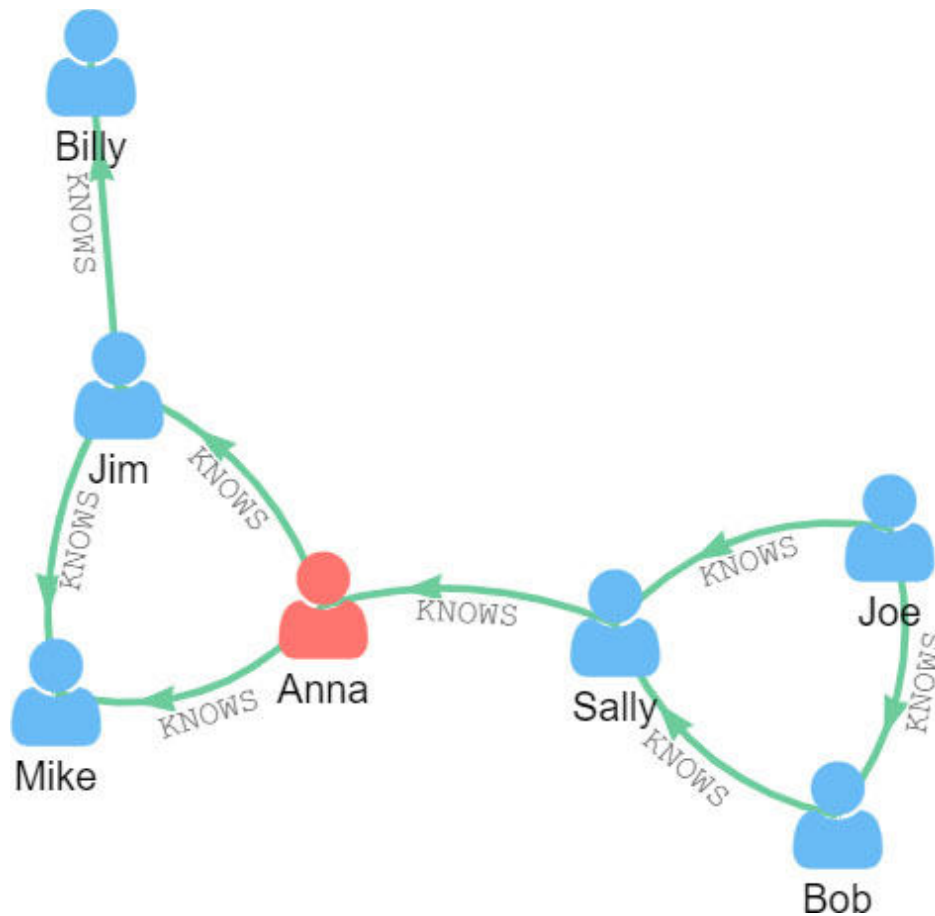


Figure 10.10: An example of Graph Database

A good use case for graph databases is social media analysis, where the relationship of entities has significant information other than the node properties themselves.

In the current example of commodity prices, let's assume that the commodities have two relationships defined as follows:

- SELLS_AT: The commodity {name,min_price, modal_price, max_price} sells at market {state, district, mandi}
- IS_OF_VARIETY: The commodity {name, min_price, modal_price, max_price} is of variety {variety}

If we create this relationship structure in our graph database, we would be able to query the relationships between the commodities faster. The graph will create three types of the node (commodity, market, and variety) and two nodes SELLS_AT and IS_OF_VARIETY.

Note: The commodity price data is not a very tightly connected row relationship. However, you can see with this example of how to work with graph databases.

[Install and start Neo4j](#)

Neo4j is a leading graph database technology. A lot of use cases in enterprises are built on top of the Neo4j Platform. Graph databases also require a strong knowledge of graph theory, as the property-based graphs also require that knowledge. The graphs we are discussing here are having node properties and not just connected entities. The reader needs to differentiate this from the graph of traveling salesman/Network Analysis kind of graph problems.

Install Neo4j by following instructions from here <https://neo4j.com/docs/operations-manual/current/installation/>. After installation, you can start the server and provide the default credentials neo4j/neo4j. You would see a window, as shown in [Figure 10.11](#):

Figure 10.11: Neo4j Graph Database

You can read more about Neo4j from <https://neo4j.com/developer/>.

[Add nodes and relations](#)

```
##Python Script to Implement the Pipeline
#Define the API KEY ( this can be found in the data.gov.in
account section for registered users)

API_KEY = <YOUR API KEY>

#Import the requests library
import requests

#Construct the GET REQUEST
response = requests.get(
    'https://api.data.gov.in/resource/9ef84268-d588-465a-
a308-a864a43d0070',
    params=[('api-key', API_KEY )],
```

```

        ('format','json'),
        ('offset',0),
        ('limit',100)],
    )

#Check if the request was successful - a success request returns
a status code 200
if response.status_code == 200:
    print('Success!')
else:
    print('Some Error Occurred')

#If the you see a success message then you can extract the
values from the JSON response
json_response = response.json()

#Define a empty list where the extracted data will be cached
before being written to data frame
import pandas as pd
arrival_date = []
commodity = []
district = []
market = []
max_price = []
min_price = []
modal_price = []
state = []
timestamp = []
variety = []

#Extract the data from json response
records = json_response["records"]
print("The records in records fields are of type
",type(records))

# Run a loop over list to extract sub-list data
for item in records:
    arrival_date.append(item["arrival_date"])
    commodity.append(item["commodity"])
    district.append(item["district"])

```

```

market.append(item["market"])
max_price.append(item["max_price"])
min_price.append(item["min_price"])
modal_price.append(item["modal_price"])
state.append(item["state"])
timestamp.append(item["timestamp"])
variety.append(item["variety"])

```

#All the data extracted via API is now stored in list. let's combine all lists into one single data frame

```

df = pd.DataFrame({'arrival_date':arrival_date,
                  'commodity':commodity,
                  'district':district,
                  'market':market,
                  'max_price':max_price,
                  'min_price':min_price,
                  'modal_price':modal_price,
                  'state':state,
                  'timestamp':timestamp,
                  'variety':variety
                  })

```

Success!

The records in records fields are of type <class 'list'>:

```

#Writing to neo4j graph
from py2neo import Graph

#Set-up the connection
graph = Graph("bolt://neo4j:admin@localhost:7687")

#Define the nodes, their properties and relationships
for index, row in df.iterrows():
    graph.run('''
        MATCH (a:commodity
        {commodity:$commodity,min_price:$min_price,modal_price:$modal_price,max_price:$max_price}), (b:market
        {state:$state,district:$district,market:$market}),
        (c:variety {variety:$variety})
        MERGE (a)-[:SELLS_AT]->(b)
    ''')

```

```

MERGE (b)-[:IS_OF_VARIETY]->(c)
'', parameters = {'commodity': row['commodity'],
'min_price': row['min_price'], 'modal_price':
row['modal_price'], 'max_price': row['max_price'], 'state':
row['state'], 'district': row['district'], 'market':
row['market'], 'variety': row['variety']})

```

The result of the graph created can be seen in the neo4j browser as well. The browser starts by default at <http://localhost:7474/browser/> and in there, run the following command to get the all node graph for an interactive view:

```

#The below query returns all the matches in the graph
MATCH(n) RETURN n

```

The graph will be similar as shown in [Figure 10.12](#). Now, let us show a simple example to query Tomato prices across all markets using py2neo library. The collect command will group the markets as per the max_price on Tomato:

Figure 10.12: An example of Neo4j Graph Database

This code shows the use of py2neo library and provides an example to query Tomato prices across all the markets:

```

#Reading from neo4j graph
from py2neo import Graph

#Set-up the connection
graph = Graph("bolt://neo4j:admin@localhost:7687")

#Run the query to bring "Tomato" prices across markets
graph.run('''
    MATCH (b:market)<-[:SELLS_AT]-(a:commodity)
    WHERE a.commodity = 'Tomato'
    RETURN a.max_price, collect(b.market)
    ''').to_table()

```

The below table shows the maximum prices of Tomato across the markets:

a.max_price	collect(b.market)
1330	['Kalikiri', 'Kalikiri']

1900	['Mulakalacheruvu', 'Mulakalacheruvu']
4000	['Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)', 'Chandigarh(Grain/Fruit)']
1800	['Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara', 'Ramanagara']
3500	['Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol', 'Hindol']
1500	['Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut', 'Baraut']

Graph databases are powerful for use cases related to connecting entities, like fraud detection, profile matching, social media analysis, recommendation engine, and many other use cases. It is important to choose the right use case before using graph databases.

Filesystem as storage

The choice of data storage for any applications require many factors to be taken care of, including the following, and many other aspects of cost and performance:

- **Type of retrieval:** How you would be retrieving the data? By querying or loading line-by-line?
- **Frequency of access:** How frequently you want to read/write the data?
- **Type of access:** Single user or multi-user

The above three are listed to show how they differ in the File system vs. the databases we discussed in previous sections.

What is Filesystem?

A filesystem manages and controls how the data is stored and retrieved from the hardware and logical storage layer. In the absence of a file system, the

disk will be just storing data without a definite way to retrieve and update it. The rules and structures which control the storage and later retrieval of files is called a file system.

The different operating system has a different way of managing files and hence different file systems. The key difference between file systems arises from different structures and logic, properties of speed, flexibility, security, and size. FAT, Apple File System, and NTFS are few popular file systems.

Filesystem as data store

The data that we are retrieving from data.gov.in is for a specific purpose and needs to be retrieved in the future when required. Filesystem at the logical layer is divided into directories and subdirectories, and the names of files. The file system defines naming conventions, storage locations along with the information such as the size of the file, as well as its attributes, location, and hierarchy in the directory. This set of metadata makes it possible to retrieve the data when required by the applications.

For instance, go back to our basic example where the data from data.gov.in was stored in a CSV file. The file was stored on the windows file system and had the attributes, as shown in [*Figure 10.13*](#):

Figure 10.13: Snippet of the attributes of the file stored

Though the file is stored and can be retrieved by applications by referring to the location, still to find a record, you need to load the entire file into memory. You cannot access the file items in any other sorted or index way.

Hierarchy to store CSV

In the case of file systems, a basic level of the hierarchy can be created to facilitate easy search and smaller chunk of files to be loaded into a disk for operations. Assume that our applications are designed in a way that commodity prices are retrieved mostly by locations. In that case, if we want to see the price of a commodity, let's say Tomato at Delhi, we should only load files for Delhi and not for all the states in India.

We can store our CSV files in the following filesystem hierarchy to facilitate faster and less costly retrieval of data:

Figure 10.14: An example of hierarchy to store CSV

In [Figure 10.14](#), you can see, if we break our CSV files into smaller chunks and store as per the hierarchy, the applications can easily access the relevant data faster. You can learn more from the given link, <https://habiletechnologies.com/blog/better-saving-files-database-file-system/>.

Conclusion

This chapter introduces the concept of databases and how they are used to store data. We discussed the key terminologies in modern databases and their significance for designing a database solution. Then we discussed setting up a SQL database, Postgre, how to use the databases, and integrated our data pipeline to store data into the database. Next, we discussed No-SQL database and how-to setup MongoDB. Further, we also introduced the idea of graph databases and showed an example of the same with neo4j. The file system is introduced in the last section to show how we can manage data storage and indexing with the file system as well. By reading this chapter, the reader has gained the understanding of the databases. He or she will now be able to setup SQL and No-SQL Databases. The reader has also acquired knowledge in using Neo4j for graph databases.

In the next chapter, we will discuss the case of what if the data outgrow our database or file system, how to manage and analyze that using Big Data.

CHAPTER 11

Introduction to Big Data

We have been talking about data all through the chapters and how technology helps us bring value from that data. The term Big Data is coined for the datasets, which are too huge to handle by our traditional set of tools and technologies. The way we store, manage, process, and control big data is very different from the way we will do the same from small datasets. A single machine processing and computation power is not enough to store process such data. While vertical scaling by increasing RAM and processors has a limit. Horizontal scaling by adding more machines is infinitely scalable. Hence, the case of Big Data lies in the distributed nature of storage and computations. In this chapter, we will discuss the Hadoop distributed file system and computation framework. We will also explain the key MapReduce function by discussing a word count example. The chapter will help you understand the basics of Big Data and Hadoop. You will also be able to manage HDFS and MapReduce and use YARN and Hadoop common utilities by following the wordcount example.

Structure

- Introducing Big Data
- Introducing Hadoop
- Settingup a Hadoop Cluster
- Word-count MapReduce program

Objectives

After studying this unit, you should be able to:

- Understand the basics of Big Data and Hadoop
- Install Hadoop Cluster
- Run an example with Hadoop

Introducing Big Data

To understand the enormity of data in real applications, consider the example of daily commodity prices we discussed in previous chapters. The *data.gov.* in website release data on the current price of the commodity in various markets. In total, it gets 3649 records per day from various **mandis**. This is about 338 Kb of CSV file. Assume that we have a machine of 4GB RAM to process this data. If we have to analyze 100 years of data, then how much memory the machine would require to load and manipulate the data?

Below is the calculation to find out the minimum RAM required to process the data:

```
# Number of Days in Year = 365
# Size of Each day File = 338 Kb
# Years worth of Data = 100
# 1 Gb contain 1048576 Kbs

Total_RAM_GB = (365*338*100)/1048576
print('\nMinimum RAM required to process the data ',
Total_RAM_GB)
Minimum RAM required to process the data: 11.765480041503906 GB
```

You can see it require at-least three times the RAM to load and then manipulate the data. As we started accumulating more data, the processing requires more RAM to process in a single machine. Hence, we require distributed computing to deal with big data.

The above example was to show, Big Data from computation requirements. Big data is impossible to store in one physical disk as well. Consider data generated by web traffic; it can generate TBs of data in a couple of hours. With disk size limited physical storage, it is important to store data in a distributed manner.

The Big Data area itself has become a vast discipline with the advent of large requirements for computations and storage facilitated by cloud computations. Distributed storage and computation are at the core of Big Data. In this chapter, we will discuss the basic concepts and some terminologies on Big Data. Further, we would show examples of how our data.gov.in pipeline can write into **Hadoop Distributed File System (HDFS)** and retrieve it using Apache Hive.

Definition of Big Data

Big Data definition has been evolving over time as we get to know more about nature and technology around the data. The core definition of Big Data still can be summarized into three aspects to define data as being big data or not:

- **Volume:** How much?
- **Variety:** How many types?
- **Velocity:** How fast?

The above three aspects are popularly called 3Vs (Volume, Variety, and Velocity) of big data, and they are explained in [Figure 11.1](#):

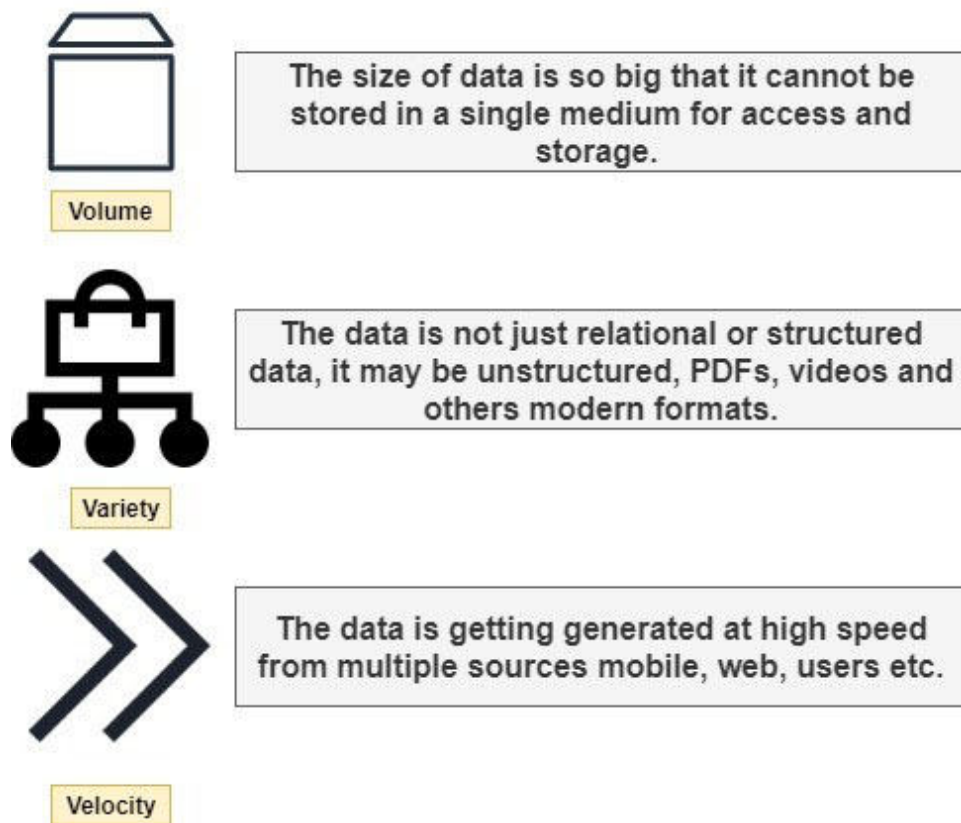


Figure 11.1: The 3Vs of Big Data

The definition has been expanded to include many other factors as well, such as veracity, value, and others. At the core of Big Data remains the power of distributed storage and processing.

Consider a few examples of big data and how they possess the above properties to require big data processing framework:

- **Volume:** The government is trying to digitize all the paper forms of land records. They are scanning the documents and storing them for future retrieval. There are millions of pages to be scanned and stored. This quality for Big Data distributed storage.
- **Variety:** The data in telecom comes from multiple sources. It can be coming from telecom towers, which is a stream of bytes, also from online recharges, OTT providers also send data to telecom provider, and it gets data from ERPs, consumer complaints and so many varied sources in a variety of formats. The data need storage that can store any type of data, and while retrieval allows flexibility.
- **Twitter:** Twitter processes millions of tweets each day coming in from so many users and bots at asynchronous times. Though the tweets have the same kind of data, but the velocity is so huge to handle in a monolithic system. It requires distributed storage and processing system to derive insights from the twitter data.

These are few examples that qualify for a Big Data infrastructure and distribute processing. The business requirements allow the IT manager to choose the right set of tools and configurations for the management of Big Data.

[Introducing Hadoop](#)

Hadoop started as Apache Nutch project in 2002 by *Doug Cutting and Mike Cafarella*. Later in 2003, Google released a white paper outlining **Google Filesystem (GFS)** and in the next paper about MapReduce. While GFS was trying to solve big data distributed storage, MapReduce started the evolution of distributed computations. Later, Doug took the project to Yahoo!, where after development and testing, publicized it as an open-source project in 2009. After 2009, the project has seen tremendous growth across industry sectors and cloud technologies.

At its core, Hadoop has two major layers, namely:

- Processing/Computation layer (MapReduce)
- Storage layer (Hadoop Distributed File System)

There are then resource manager and other common libraries assisting the process of distributed storage and computations. Each of the components described in [Figure 11.2](#) is part the Hadoop core package:

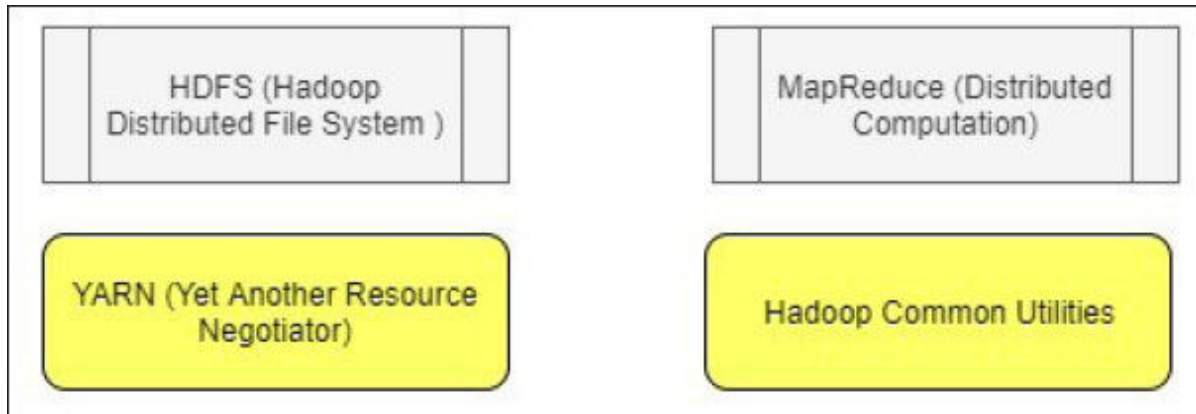


Figure 11.2: Hadoop Architecture

There are additional applications that enhance the utility and monitoring of Hadoop.

Hadoop Distributed File System (HDFS)

HDFS got conceptualized on the ideas of GFS. In this type of file system, the data is converted into blocks of memory and stored in nodes on multiple systems. The logic to break a file and then combine them for retrieval is a part of HDFS architecture. With this capability, you can store data far bigger in size for one node/system to be stored in multiple systems. It is highly fault-tolerant (a property that allows a system to function appropriately in the event that some of its components malfunction) and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets:

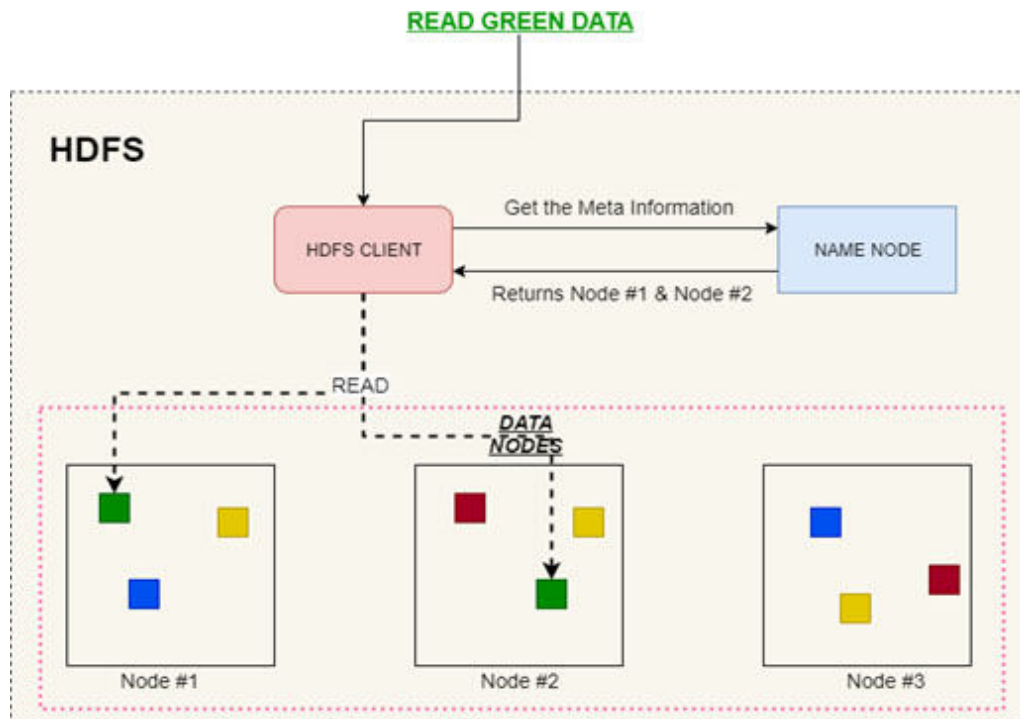


Figure 11.3: HDFS architecture

HDFS architecture has two types of nodes:

- **NameNode:** It is the master node in the HDFS architecture. The node which maintains the address table of each memory block.
- **DataNode:** These are the slave nodes in the HDFS architecture. The node which actually stores the memory block.

To achieve fault tolerance, the data memory blocks are replicated so that even if one memory block goes unavailable, we can still access the data from another node. The configuration of HDFS is managed at the metadata store in NameNode. You can read details of HDFS here, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

MapReduce

MapReduce is within the Hadoop framework, which is used to process data stored in HDFS. It is a programming model, being a key part and functional component of the Hadoop framework. MapReduce makes it possible to process petabytes of data in parallel on data nodes, by dividing them into smaller chunks. Eventually, it adds up all data from several servers to give the application a consolidated result:

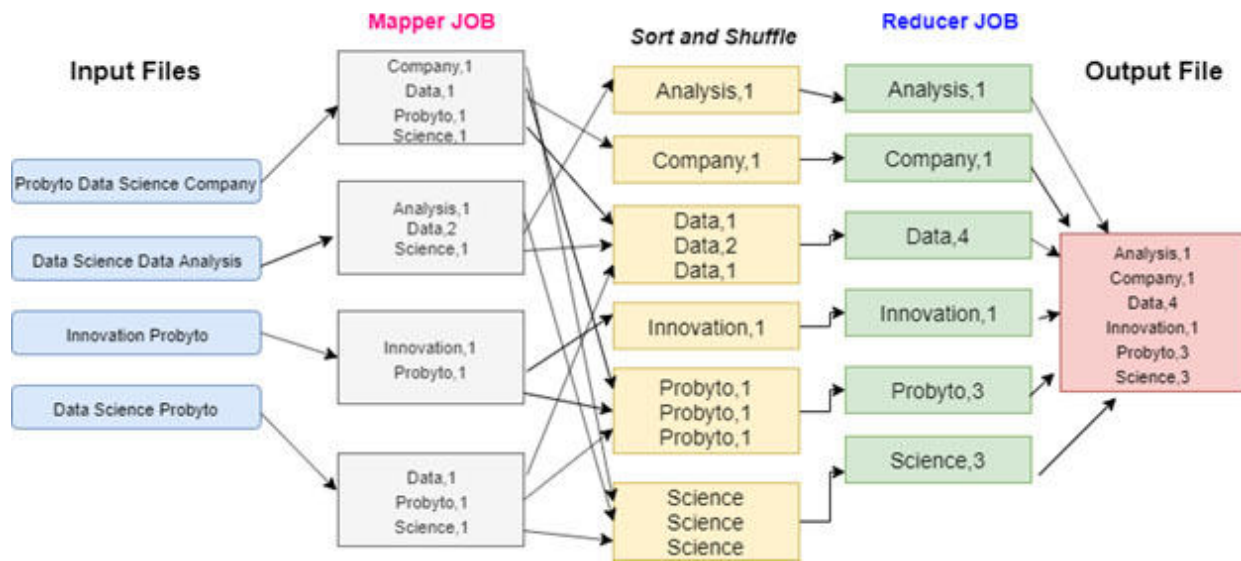


Figure 11.4: Example of a MapReduce program

[Figure 11.4](#) shows how a MapReduce program will work for a word count problem. The MapReduce program will divide the task into the small parallel process, then the process (or task) is pushed to each node to do the processing locally, once the processing is done, the Reducer program will combine the results of each node and give the result back to application or HDFS.

YARN

There is a resource management and a job scheduling technology within the Hadoop distributed processing framework known as Apache Hadoop YARN. Apache Hadoop's YARN allocates the system resources to the various Hadoop cluster applications and schedules tasks to be performed on various cluster nodes:

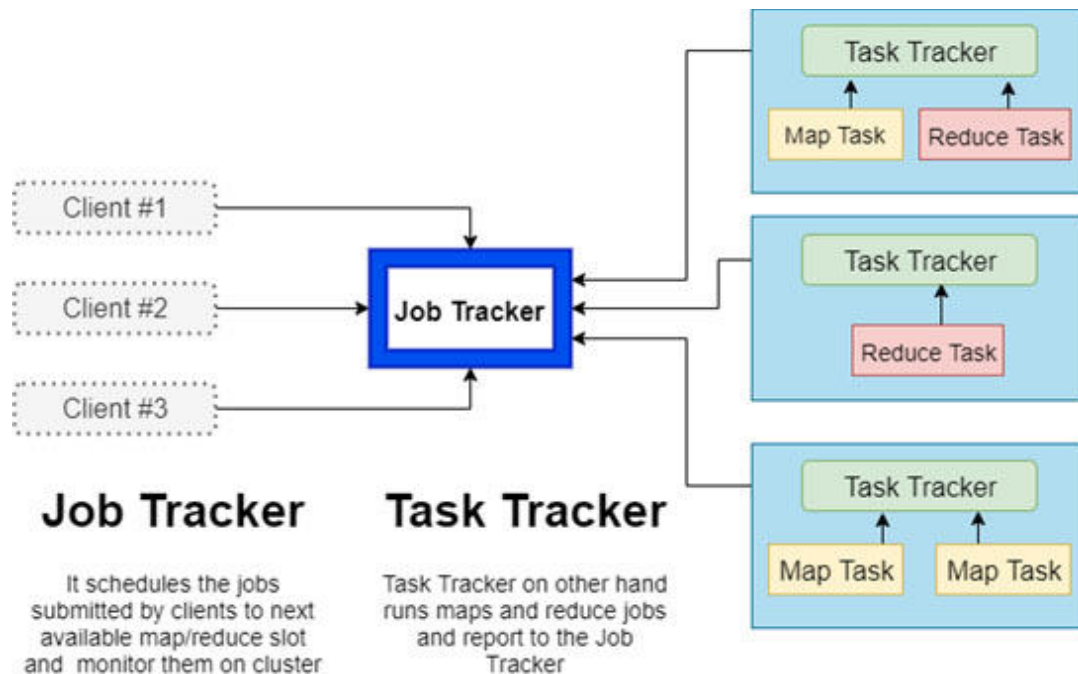


Figure 11.5: YARN Architecture

YARN can allocate resources to applications dynamically as required, which is capable of improving the use of resources and application performance in contrast with the more static MapReduce allocation approach. It also supports multiple scheduling techniques for submitting processing jobs, on the basis of a queue format. There exists a generic FIFO scheduler that operates on a first-in-first-out basis for the various applications.

Hadoop common

The backbone of the Hadoop framework is the Hadoop common package that offers the key services as well as the basic processes. The essential **Java Archive (JAR)** files and scripts required to start Hadoop is incorporated in the Hadoop Common. A contribution section is also available, provided by the Hadoop common package that includes different projects from the Hadoop Community, bringing forth the availability of source code and documentation as well.

Setting-up a Hadoop Cluster

The native filesystem in our windows or mac machine is not designed to handle distributed filesystem. The Hadoop environment needs to be installed

with its own filesystem and the process to perform operations on them. As discussed in previous sections to work with Hadoop File System, we need to set-up the four critical components of Hadoop; filesystem, resources manager and mapper/reducers functions, and Hadoop common libraries.

Installing a Hadoop Cluster

There are multiple ways to start working in the Hadoop system; depending upon your need, and you can choose the right set-up for your use case. The most popular method for starting to learn Hadoop is by installing a virtual machine and install Cloudera sandbox. The Sandbox can be downloaded from here (https://www.cloudera.com/downloads/quickstart_vms/5-13.html). The minimum requirements to start a single-node Hadoop cluster are as follows (source: **cloudera.com**):

- 64-bit VMs running on a 64-bit OS of host and VMware
- VMware with workstation 8.x or Player 4.x or Fusion 4.x
- Minimum RAM required by VM is 4 GB to install, however, to run the decent example you need minimum 10GB

As you can see, the Hadoop system is very heavy and requires a lot of infrastructures to run it. It is recommended that you check the system before trying to run and test the Hadoop system.

In real production, the Hadoop system runs on massive IT infrastructure supported by hardware and big data engineers. The minimum requirement for the production level is 64 GB RAM/16 cores/256 GB HDD per server. The configurations are just indicative to show the enormity of hardware infrastructure required to churn PB of data each day in enterprise systems.

To run through a simple example of word-count and explain the MapReduce process, we will use a Docker installation of Hadoop. Docker installation of Hadoop is not recommended for any heavy data processing in the Hadoop system. You have already been introduced to Docker concepts in the previous chapter. We will directly install the cluster and discuss the HDFS and MapReduce with word-count example.

Starting Hadoop cluster in Docker

Hadoop Cluster in Docker is an easier way to test basic examples in Hadoop and learning purposes. In this section, we will show you how you can start a Hadoop cluster using **docker-compose**. To run through the example, you need to have some prerequisites:

1. Install Docker Engine in your system (<https://docs.docker.com/install/>).

2. Check if Docker Compose, Docker Engine, is installed and working. Run following commands in your command prompt/bash:

```
$ docker --version
$ docker-compose --version
$ docker-machine --version
```

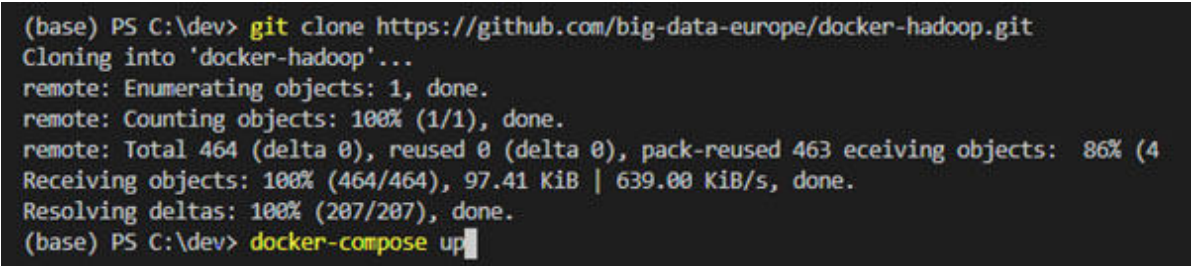
3. Make sure the Docker can use at least 4 CPUs and 8 GB RAM.

Note: The example in this chapter is developed on a PC with the following configurations 16 GB RAM, 1 TB HDD, 8 CPUs, and 2.6GHz processor.

For creating the cluster, we will be using the Docker compose file provided by the big-data-europe public repository on GitHub.

Link to Repository: <https://github.com/big-data-europe/docker-hadoop>

The below [Figure 11.6](#) shows commands you need to run on your terminal to get the required files copied to your system. This requires that you have already installed Git in your system (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>):



```
(base) PS C:\dev> git clone https://github.com/big-data-europe/docker-hadoop.git
Cloning into 'docker-hadoop'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 464 (delta 0), reused 0 (delta 0), pack-reused 463 (delta 0)
Receiving objects: 100% (464/464), 97.41 KiB | 639.00 KiB/s, done.
Resolving deltas: 100% (207/207), done.
(base) PS C:\dev> docker-compose up
```

Figure 11.6: Snippet of the terminal while cloning using Git

Once you run the **docker-compose** command, the Docker system will start searching for images, and if they do not exist will try to fetch them. The fetching process is shown in [Figure 11.7](#) below. It can take several minutes

while the system tries to fetch the images required to build the Hadoop cluster:

```
(base) PS C:\dev\bp2019\hadoop-example\docker-hadoop> docker-compose up
Creating network "docker-hadoop_default" with the default driver
Creating volume "docker-hadoop_hadoop_namenode" with default driver
Creating volume "docker-hadoop_hadoop_datanode" with default driver
Creating volume "docker-hadoop_hadoop_historyserver" with default driver
Pulling namenode (bde2020/hadoop-namenode:2.0.0-hadoop3.1.3-java8)...
2.0.0-hadoop3.1.3-java8: Pulling from bde2020/hadoop-namenode
9a0b0ce99936: Pulling fs layer
9fc38e922647: Downloading [>] 2.16MB/159.4MB
ff3988a5db71: Download complete
97f386466085: Download complete
e061bf44eff1: Downloading [>] 1.054MB/340.5MB
3aa1be443cb6: Waiting
1aff33764bc9: Waiting
a2c563ea782e: Waiting
1ec676fa35da: Waiting
d1fee30add0d: Waiting
70f0c92f3e46: Waiting
9a0b0ce99936: Downloading [=====>] 27.14MB/45.38MB
9fc38e922647: Downloading [=====>] 28.55MB/159.4MB
9a0b0ce99936: Downloading [=====>] 21.17MB/45.38MB
9fc38e922647: Downloading [=====>] 21.01MB/159.4MB
e061bf44eff1: Downloading [====>] 20.96MB/340.5MB
```

Figure 11.7: Snippet of terminal showing fetching process

After several minutes of processing, the Hadoop cluster will start, and you will see a different system of Hadoop starting up in different nodes. You will observe logs of nodemanager, namenode, history server, and datanode.

You can also view the number of servers started by the Docker system by using docker ps command, as shown in below [Figure 11.8](#):

```
(base) PS C:\dev> docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED            STATUS              PORTS              NAMES
a07006308da4       bde2020/hadoop-namenode:2.0.0-hadoop3.1.3-javai /entrypoint.sh /run... 13 minutes ago    Up 13 minutes (healthy) 0.0.0.0:3020->3020/tcp namenode
d25e6018e13c       bde2020/hadoop-historyserver:2.0.0-hadoop3.1.3-javai /entrypoint.sh /run... 13 minutes ago    Up 13 minutes (healthy) 8188/tcp            historyserver
41aa166a205        bde2020/hadoop-nodemanager:2.0.0-hadoop3.1.3-javai /entrypoint.sh /run... 13 minutes ago    Up 13 minutes (healthy) 8042/tcp            nodemanager
80a55a1ed129       bde2020/hadoop-datanode:2.0.0-hadoop3.1.3-javai /entrypoint.sh /run... 13 minutes ago    Up 13 minutes (healthy) 9864/tcp            datanode
470007c1ade        redis                                     "docker-entrypoint.s..." 2 weeks ago       Up 13 minutes       6379/tcp            docker_files_redis_1
```

Figure 11.8: Snippet of terminal showing output of docker ps

The Hadoop system also comes with UI written to access the status and health of cluster using URLs. These URLs show the basic statistics of the node and the processes it is running along with a lot of other information.

If you are running the cluster in local machine, by default all services will expose the URLs at localhost, if running on VMs or cloud you can use

Docker network inspect to get the cluster IP to get access to following URLs:

- **Namenode:** **http://<docker-hadoop_IP_address>:9870/dfshealth.html#tab-overview**
- **History server:** **http://<docker-hadoop_IP_address>:8188/applicationhistory**
- **Datanode:** **http://<docker-hadoop_IP_address>:9864/**
- **Nodemanager:** **http://<docker-hadoop_IP_address>:8042/node**
- **Resource manager:** **http://<docker-hadoop_IP_address>:8088/**

The main URLs would be the Namenode one as that contains information about all other nodes and their health. The URL will show a window similar to below [Figure 11.9](#):

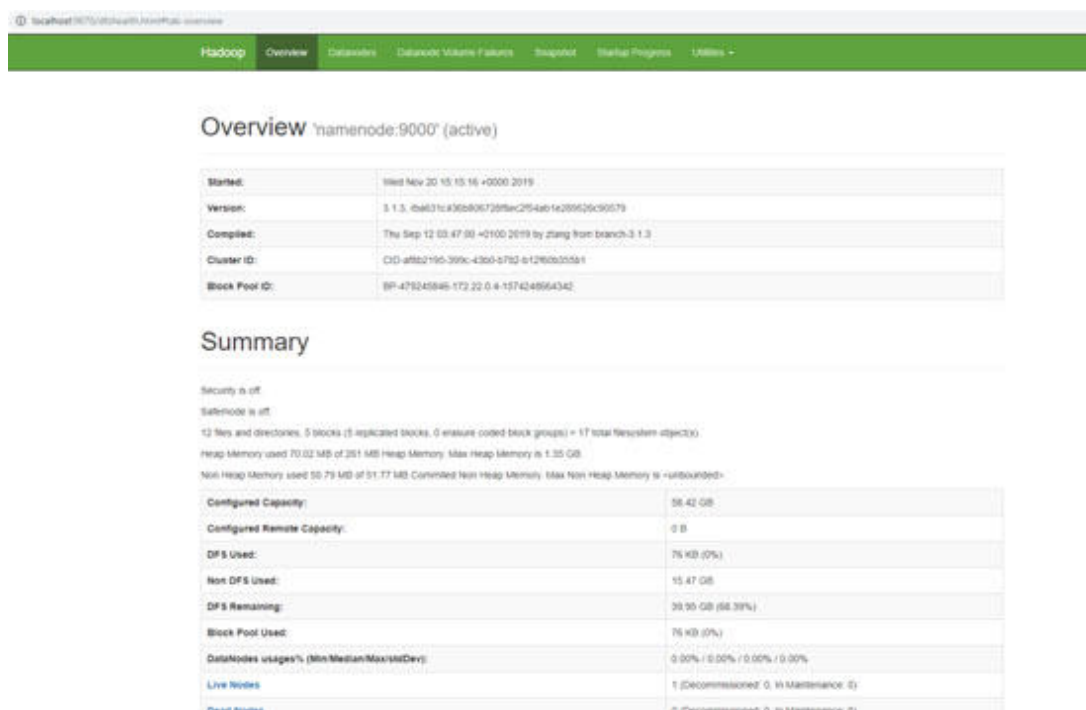


Figure 11.9: Hadoop Overview window

Now we can confirm that the Hadoop cluster is up and running. We can now push files to the HDFS and run the computations in distributed forms by using MapReduce functions. Next Section, we show you what has to be written in Map and Reduce function for a word-count sample problem. The

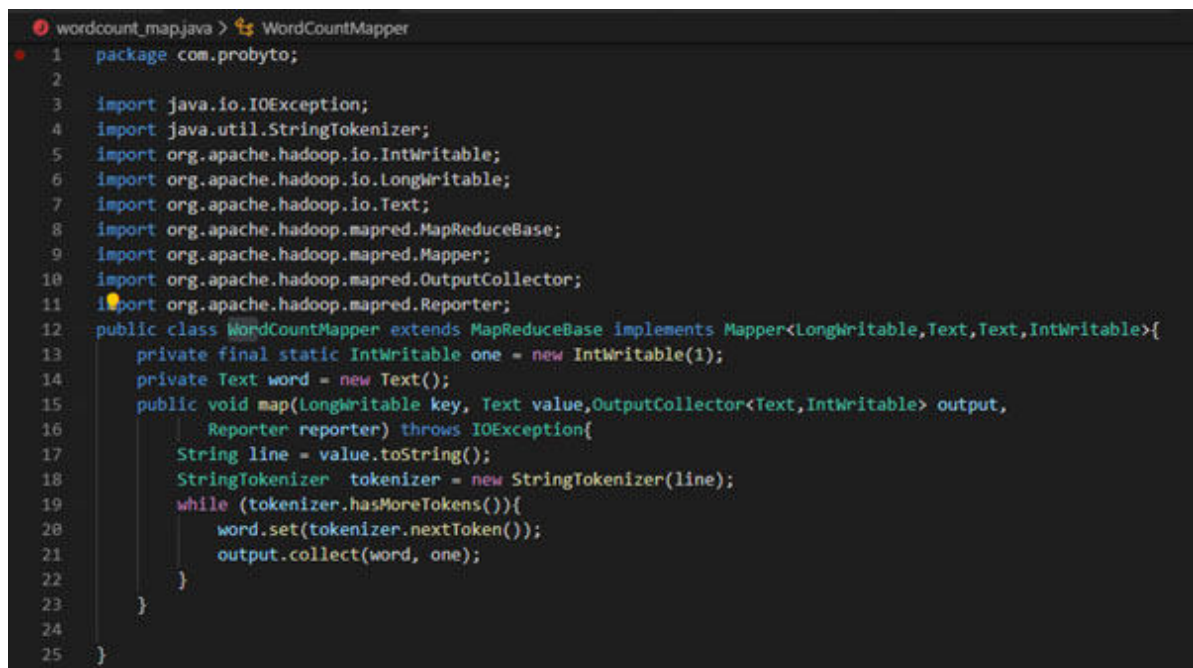
basic engine for Hadoop in Java so we have to write our MapReduce for word-count in Java.

Word-count MapReduce Program

Word count is a very important problem statement discussed in many tutorials and all 101 courses on Hadoop. The problem statement explains all the concepts of distributed storage and distributed computation on a Hadoop system. The word count problem ingests numerous files and outputs the working frequency. The data could be stored in thousands of different files across multiple machines; still, the MapReduce program will be able to successfully compute the frequency. As this is an aggregation problem, also shown in [Figure 11.4](#), it is a perfect fit to show MapReduce working.

Map program

The Map program will take all the inputs and tokenize them into words. After tokenizing, it will map each word to its count in that input file. In this way, all the input files will be tokenized and will have the word count in each file. A sample Java program for the **Mapper** class is shown below in [Figure 11.10](#):

A screenshot of a Java IDE showing the code for the WordCountMapper class. The code is written in Java and implements the Mapper interface. It includes imports for various Hadoop and Java classes, and a map method that tokenizes input text and outputs word counts.

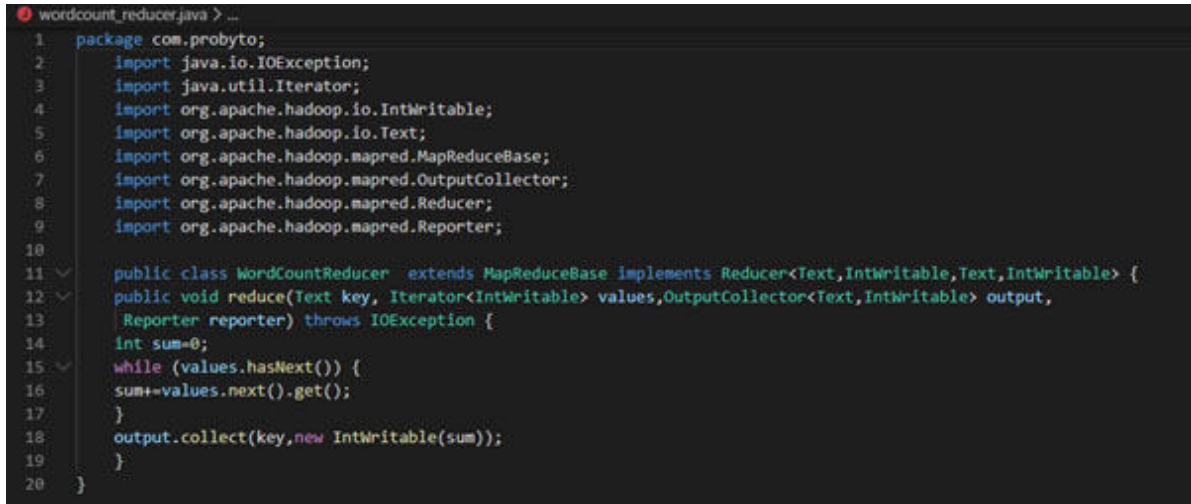
```
wordcount_mapjava > WordCountMapper
1 package com.probyto;
2
3 import java.io.IOException;
4 import java.util.StringTokenizer;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.LongWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapred.MapReduceBase;
9 import org.apache.hadoop.mapred.Mapper;
10 import org.apache.hadoop.mapred.OutputCollector;
11 import org.apache.hadoop.mapred.Reporter;
12 public class WordCountMapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>{
13     private final static IntWritable one = new IntWritable(1);
14     private Text word = new Text();
15     public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
16         Reporter reporter) throws IOException{
17         String line = value.toString();
18         StringTokenizer tokenizer = new StringTokenizer(line);
19         while (tokenizer.hasMoreTokens()){
20             word.set(tokenizer.nextToken());
21             output.collect(word, one);
22         }
23     }
24 }
25 }
```

Figure 11.10: Sample Java program for Mapper class

You can see in the above code the mapper function tokenizes the text input at line 18 and collect all words by their count in line 21.

Reducer program

The reducer program will combine the result of the mapper program for each map task running across the cluster. The mapper function for wordcount example is shown in below [Figure 11.11](#):



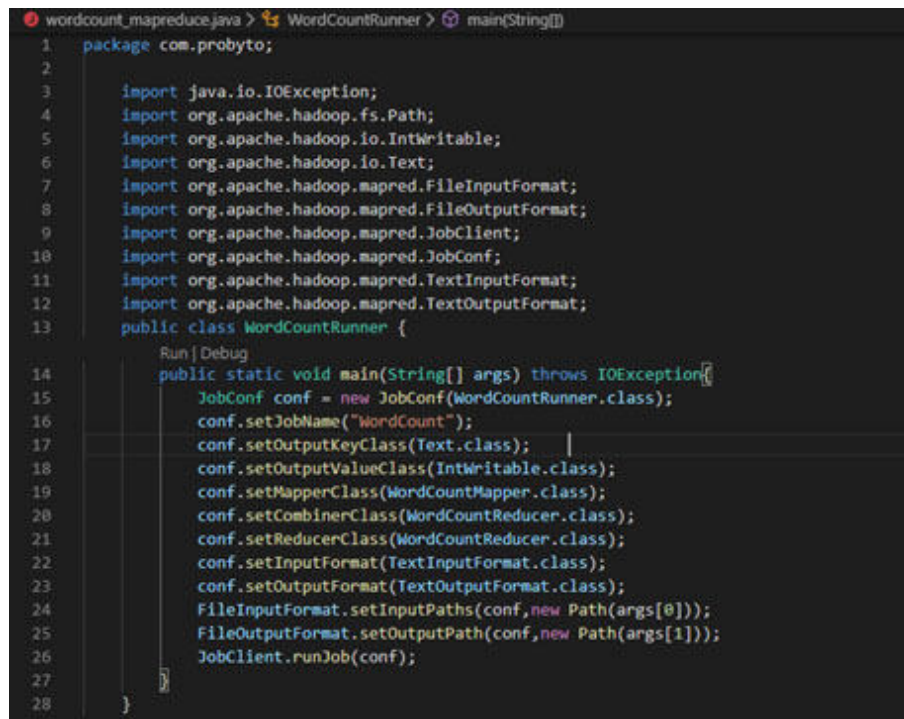
```
wordcount_reducer.java > ...
1 package com.probyto;
2 import java.io.IOException;
3 import java.util.Iterator;
4 import org.apache.hadoop.io.IntWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapred.MapReduceBase;
7 import org.apache.hadoop.mapred.OutputCollector;
8 import org.apache.hadoop.mapred.Reducer;
9 import org.apache.hadoop.mapred.Reporter;
10
11 public class WordCountReducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
12     public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
13         Reporter reporter) throws IOException {
14         int sum=0;
15         while (values.hasNext()) {
16             sum+=values.next().get();
17         }
18         output.collect(key,new IntWritable(sum));
19     }
20 }
```

Figure 11.11: Mapper function for word count example

You can see the above function sum up all tokens and create output as key-value pair in line 18.

MapReduce JAR

Now we need to combine the **Map** and **Reduce** function by writing a runner code which will trigger the mapper and reducer functions, and then allow us to get the final output. The Java code, as shown below in [Figure 11.12](#), need to be converted into an executable JAR file before pushing to the cluster for using for our word-count problem:



```

1 package com.probyto;
2
3 import java.io.IOException;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapred.FileInputFormat;
8 import org.apache.hadoop.mapred.FileOutputFormat;
9 import org.apache.hadoop.mapred.JobClient;
10 import org.apache.hadoop.mapred.JobConf;
11 import org.apache.hadoop.mapred.TextInputFormat;
12 import org.apache.hadoop.mapred.TextOutputFormat;
13 public class WordCountRunner {
14     public static void main(String[] args) throws IOException {
15         JobConf conf = new JobConf(WordCountRunner.class);
16         conf.setJobName("WordCount");
17         conf.setOutputKeyClass(Text.class);
18         conf.setOutputValueClass(IntWritable.class);
19         conf.setMapperClass(WordCountMapper.class);
20         conf.setCombinerClass(WordCountReducer.class);
21         conf.setReducerClass(WordCountReducer.class);
22         conf.setInputFormat(TextInputFormat.class);
23         conf.setOutputFormat(TextOutputFormat.class);
24         FileInputFormat.setInputPaths(conf, new Path(args[0]));
25         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
26         JobClient.runJob(conf);
27     }
28 }

```

Figure 11.12: Java code for word count problem

You can see in the above code, the job is defined at line 16. The mapper and reducer classes are called here and used to manage MapReduce operations. We need to convert these files into an executable JAR file (**Example:** <https://www.webucator.com/how-to/how-create-jar-file-java.cfm>)

For easy usage, we will provide the JAR file for wordcount problem that can be used to replicate the tutorial for data processing in Hadoop.

[Running Word Count in HDFS Cluster](#)

Now we have set-up the Hadoop cluster and checked its health as well. The MapReduce program is also written to find word count after processing multiple input files. Now we will test the word count problem with the illustrative image, as shown in [Figure 11.13](#).

We will be using 4 input files with the following content in each of them, as shown in [Table 11.1](#):

Input/f1.txt	Probyto Data Science Company
Input/f2.txt	Data Science Data Analysis
Input/f3.txt	Innovation Probyto

Table 11.1

The steps to follow would be as follows;

1. We will first enter into the namenode.
2. Create a new directory named input and write the 4 files in it.
3. Push the directory into the HDFS cluster.
4. Copy the word count JAR file into the name node.
5. Run the JAR file program inside namenode.
6. Display the output and verify it is the same as shown in [Figure 11.12](#).

Step 1: Enter into the namenode by running exec bash command, as shown in [Figure 11.13](#):

```
(base) PS C:\dev> docker exec -it namenode bash
root@a87096398da4:/# ls
KEYS bin boot dev entrypoint.sh etc hadoop hadoop-data home lib lib64 media mnt opt proc root run run.sh sbin srv sys tmp usr var
root@a87096398da4:/#
```

Figure 11.13: Snippet of terminal showing entering into namenode

Step 2: Create a new directory with mkdir and write 4 input files, as shown in [Figure 11.14](#):

```
root@a87096398da4:/# mkdir input
root@a87096398da4:/# echo "Probyto Data Science Company" > input/f1.txt
root@a87096398da4:/# echo "Data Science Data Analysis" > input/f2.txt
root@a87096398da4:/# echo "Innovation Probyto" > input/f3.txt
root@a87096398da4:/# echo "Data Science Probyto" > input/f4.txt
root@a87096398da4:/# ls
KEYS bin boot dev entrypoint.sh etc hadoop hadoop-data home input lib lib64 media mnt opt proc root run run.sh sbin srv sys tmp usr var
root@a87096398da4:/# cd input/
root@a87096398da4:/input# ls
f1.txt f2.txt f3.txt f4.txt
root@a87096398da4:/input#
```

Figure 11.14: Snippet of terminal showing creating directory and writing 4 files

Step 3: Push all the files into the HDFS cluster by using hdfs put command, as shown in [Figure 11.15](#):

```
root@a87096398da4:/input# cd ..
root@a87096398da4:/# hadoop fs -mkdir -p input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@a87096398da4:/# hdfs dfs -put ./input/* input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2019-11-20 16:53:09,310 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-11-20 16:53:10,050 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-11-20 16:53:10,499 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-11-20 16:53:10,948 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@a87096398da4:/#
```

Figure 11.15: Snippet of terminal showing files getting pushed to hdfs

Step 4: Copy the JAR file into the namenode as shown in [Figure 11.16](#):

```
(base) PS C:\dev\big2019\hadoop-example\docker-hadoop> docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS              PORTS                               NAMES
a87096398da4        bde020/hadoop-namenode:2.8.0-hadoop3.1.3-javall   "/entrypoint.sh /run..." 2 hours ago         Up 2 hours (healthy) 8.8.8.8:8020->9870/tcp              namenode
a87096398da4        bde020/hadoop-historyserver:2.8.0-hadoop3.1.3-javall   "/entrypoint.sh /run..." 2 hours ago         Up 2 hours (healthy) 8188/tcp                          historyserver
8845c5a1d129        bde020/hadoop-datanode:2.8.0-hadoop3.1.3-javall   "/entrypoint.sh /run..." 2 hours ago         Up 2 hours (healthy) 9864/tcp                          datanode
47d8687e1dde        redis                                          "docker-entrypoint.t..." 2 weeks ago         Up 2 hours          6379/tcp                          docker_files_redis_1

(base) PS C:\dev\big2019\hadoop-example\docker-hadoop> docker cp .\hadoop-wordcount.jar a87096398da4:hadoop-wordcount.jar
(base) PS C:\dev\big2019\hadoop-example\docker-hadoop>
```

Figure 11.16: Snippet of terminal showing JAR file getting copied to namenode

Step 5: The JAR file then can be executed inside the namenode with following commands, as shown in [Figure 11.17](#):

```
(base) PS C:\dev> docker exec -i namenode bash
root@a87096398da4:/# ls
KEYS  bin  boot  dev  entrypoint.sh  etc  hadoop  hadoop-data  hadoop-wordcount.jar  howe  input  lib  lib64  media  mnt  opt  proc  root  run  run.sh /sbin  srv  sys  tap  usr  var
root@a87096398da4:/# hadoop jar hadoop-wordcount.jar org.apache.hadoop.examples.wordCount input output
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2019-11-20 17:00:17,590 INFO client.RetryProxy: Connecting to ResourceManager at resourcemanager:8032
2019-11-20 17:00:17,792 INFO client.AGSPProxy: Connecting to Application History server at historyserver/172.22.0.1:10200
```

Figure 11.17: Snippet of terminal showing JAR file getting executed

Step 6: Finally, you can see the output of word count, as shown below in [Figure 11.18](#):

```
root@a87096398da4:/# hdfs dfs -cat output/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2019-11-20 17:24:43,120 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
Analysis 1
Company 1
Data 4
Innovation 1
Probyto 3
Science 3
root@a87096398da4:/#
```

Figure 11.18: Snippet of terminal showing the output of word count

In the above set of 6 steps, we showed how the wordcount example would work for 4 input files. The HDFS system allows us to do distributed computation on millions of records without any problem, though there will be latency as the data size will be huge.

There are better solutions than MapReduce now, such as Apache Spark, which is way faster and more compatible with other programming languages like Python and R for data science related programs. There are mature libraries in the Big Data ecosystem to run multiple ML algorithms in the HDFS system.

Conclusion

In this chapter, we have introduced Big Data and explained its definition. Thereafter, we have introduced Hadoop and discussed its two layers, HDFS and MapReduce. We have discussed the architecture of HDFS and also

explained the architecture of MapReduce with an example. We gave a high-level idea of YARN and Hadoop common utilities. Then we have shown setting up a Hadoop cluster and explained MapReduce solving a word count problem.

By reading this chapter, the reader has gained a basic understanding of Big Data and Hadoop. He or she will be able to excel in his or her skills to manage HDFS and MapReduce.

In the next chapter, we will introduce DevOps and discuss its functionalities. We will also discuss the source code management platforms.

CHAPTER 12

DevOps for Data Science

DevOps is a term created by coming two important IT terms; development and operations. Development is the part of the IT process where the application is developed, while operations are the process that brings the application to the operational state for end users.

In previous chapters, the focus was on developing applications and putting them to work in your local environment for the desired business purpose. Cloud brings applications to real-life by having the available application 24X7 across the globe. However, before we start the cloud computing world, we need to understand the core bridge function of DevOps, which facilitate the movement of application from developers to IT operations. This chapter will introduce some core concepts, tools, and examples to give readers a starting point into DevOps for data science.

Structure

- Introduction to DevOps
- Agile methodology, CI/CD, and DevOps
- DevOps for data science
- Source code management
- Quality assurance
- Model objects and security
- Production deployment
- Communication and collaboration

Objectives

After studying this unit, you should be able to:

- Understand the basics of software development

- Manage source code
- Discuss containers and VMs

Introduction to DevOps

DevOps is not a concrete method but rather a cultural movement. It is a series of activities that automates software development and IT department processes so that they can develop, test, and release software more confidently. The theory of DevOps is based on the development of a culture of teamwork between groups. As shown in [Figure 12.1](#), DevOps sits at the junction of three critical functions; development, **quality assurance (QA)**, and IT operations. QA is many time parts of the development process as the development needs to be of set quality before being pushed to operations:

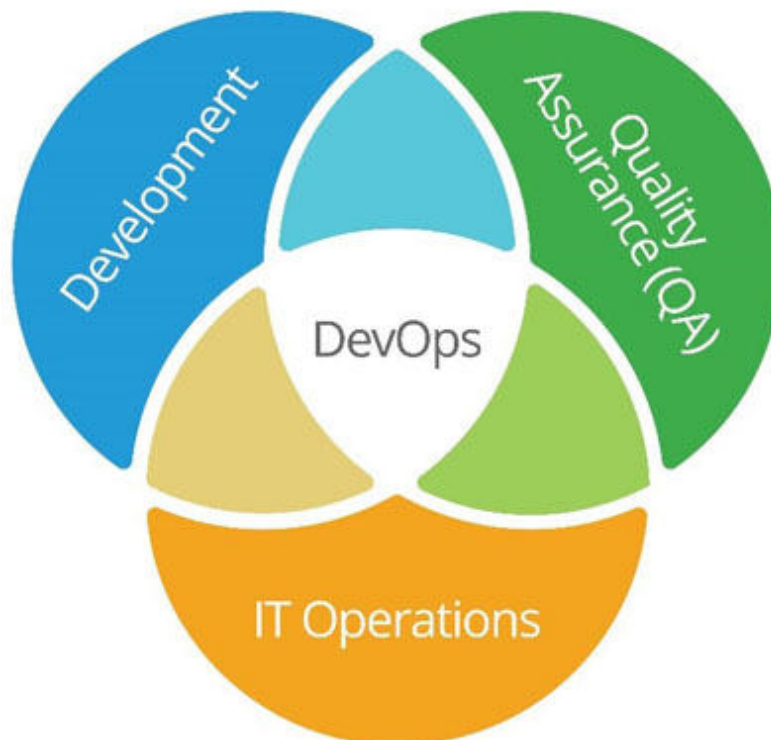


Figure 12.1: The three functions of DevOps: Development, QA, and IT Operations

This is to note that, in the case of ML models, we call out *Productization* as the process to bring models to operational use. DevOps take that as well to integrate with a larger application and deploy to cloud for public use.

Consider the following scenario to understand the genesis of DevOps.

The development team is developing an application as per the business requirements. The team just finished building the application and asked the QA team to test the code. Now, at this handover lot of problems will happen as QA team will identify bugs, which may be bug of development or wrongly stated objectives, along with actual bugs. After a round of exchanges, the QA team and development team finalizes the code and send it to the operations team for production deployment. This is the second spot where the handover of application may create issues, where the deployment team would not be able to deploy the code in production blaming the QA and Dev, while they claim it all worked fine in their environment. Even if deployed, the application caused load issues in the virtualized environment. And now, you will everyone worried and running around to run the application.

The solution to the above issues lies in better coordination among teams and timely input to the teams. The whole mindset shift in the software development process and coordination mechanism resulted in DevOps. DevOps brings all the processes together to make them work in tandem for efficient and faster delivery of IT products:

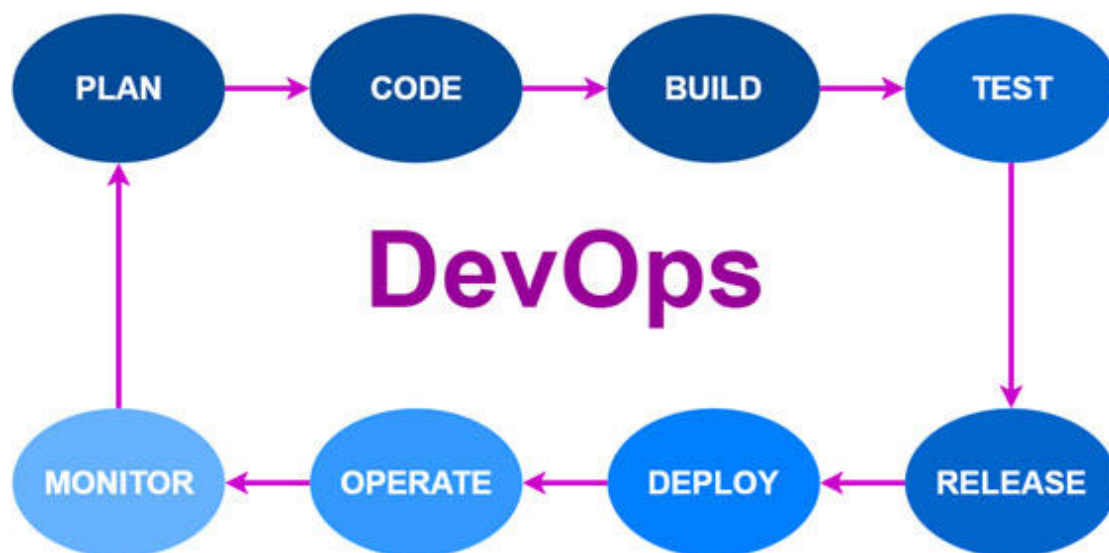


Figure 12.2: DevOps cycle

A new mindset or approach to software delivery needed to implement to increase time to deploy and reduce production issues. A new approach to the distribution of software needed to increase deployment time and reduce production problems. The new process connects all key steps in the product

and creates a feedback system to help deliver value to the whole process. The key benefits include:

- Shorter development cycles, faster innovation.
- Resilient deployment models (Reduce implementation failure, reflections, and recovery time).
- Less operational expenditures in break/fix and more time allocated to innovation.
- Better communication and cooperation.
- Reduce costs and IT staff.

In the 2015 State of DevOps Report, Puppet Labs mentioned, *High-performing IT organizations experience 60x fewer failures and recover from failure 168x faster than their lower-performing peers. They also deploy 30x more frequently, with 200x shorter lead times.* The value of this opportunity means that businesses are starting to move towards this approach for profit. A good resource to get into details can be accessed here, <https://devops.com>.

Agile methodology, CI/CD, and DevOps

The three terms, Agile methodology, CI/CD (**Continuous Integration/Continuous Delivery**), and DevOps are three common terms that are used software development. All these areas have some level of overlap and own sphere of significance in every software development cycle. [Figure 12.3](#) gives a summary of Agile, CI/CD and DevOps:

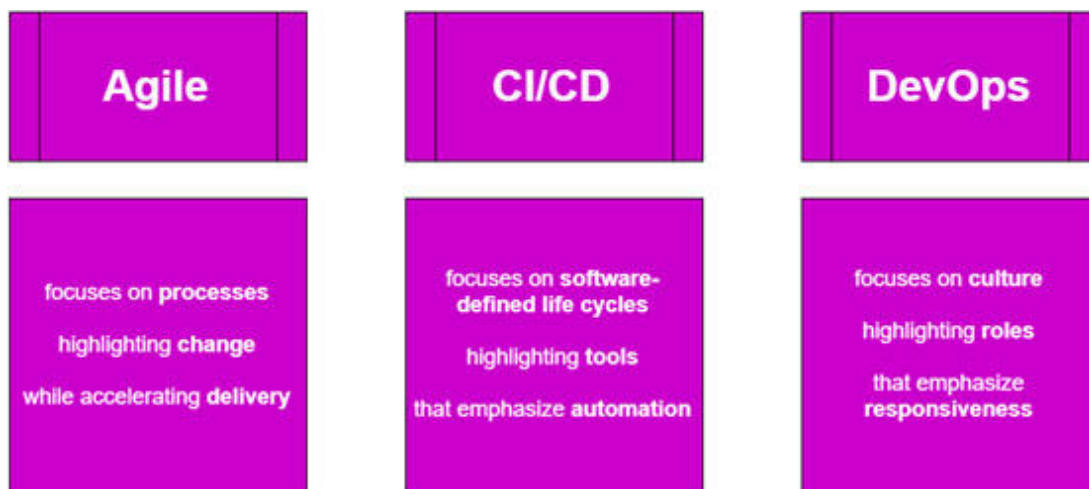


Figure 12.3: Three terms in Software Development

As you can see, the core mandate of the processes, they are tools that bring best practices to the **software development process**. The **Software Development Life Cycle (SDLC)** domain discusses these concepts in detail and provides the best practices:

- Agile focuses on processes highlighting change while accelerating delivery.
- CI/CD focuses on software-defined life cycles highlighting tools that emphasize automation.
- DevOps focuses on culture, highlighting roles that emphasize responsiveness.

In the following sections, we will discuss specific tools and subprocesses to enable data science models to be part of bigger applications and operationalized by IT operations.

DevOps for data science

Data scientist differs from developers in the scope of their work in bigger application development, though they need to care as much about their deliverable as a developer will do. While developer role is mature in software development industry, data scientist role is still evolving from a software development for data science products or features. The deliverable of data scientist including model structure, data processing, prediction variables and other, keep changing making it tough to manage in application development process. The data science models/features have a huge influence on resources and environments running them. The quality assessment and IT operations teams need to make sure that the data science features work as expected and can be deployed in the enterprise infrastructure. Cybersecurity is also a concern for data science applications as they expose data pipelines in the application. Hence, the DevOps functions also need to take care of data science DevOps needs as well. The data science DevOps is also called **Artificial Intelligence DevOps (AIOps)**. Generally, DevOps service will include:

- Source Code Management
- QA

- Model objects and security
- Production deployment
- Collaboration and communication

The areas will be discussed in brief in following sections for an overview of the respective functions. Readers are encouraged to focus on the problems the tools solve, but not the tools itself in next sections. There are many tools and frameworks available to follow best practices in DevOps for data science.

Source Code Management

Data scientist write codes which do the complex task of model training, EDA, data pipelines, and other AI/ML specific code. As the model gets mature, the data science codebase also evolves and that too in a non-linear form. The situation becomes complex when multiple people are working on same data science model. It is important to manage the data science code as well like an application code. A Source Code Management is a tool to manage the source codes, used by programmers. In Agile (<https://www.agilealliance.org/agile101/>) style code development, the product continues to develop over an indefinite time. Some form of version control is very beneficial for such applications. [Figure 12.4](#) shows a snippet of a Bitbucket repository:

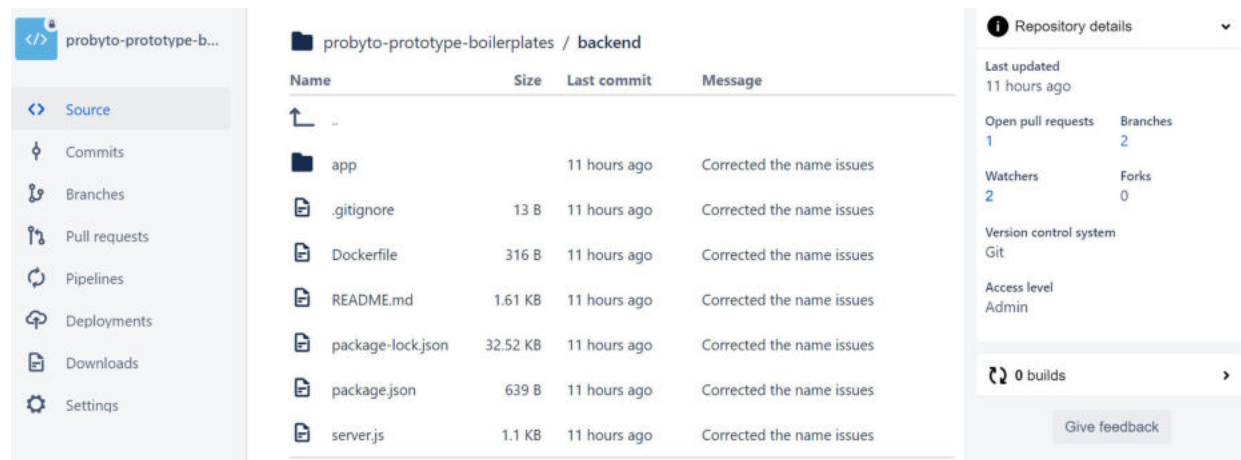


Figure 12.4: Snippet of a Bitbucket Repository

Git is a very popular version management system (<https://git-scm.com/>); SVN is another popular version management

system(<https://subversion.apache.org/>). Git works on all forms of platforms like GitHub, GitLab (<https://gitlab.com/>), and BitBucket. If you are storing binary files, consider looking into Git LFS. Readers can take a tutorial at Atlassian to master their skills in using Git version management System (link: <https://www.atlassian.com/git/tutorials>). A very useful feature among lot others is able to compare different version of codes to make sure all the changes in the script are tracked and approved before deploying to production. [Figure 12.5](#) shows a snippet of two different versions of codes getting merged in Bitbucket:

```

24 28 exports.signup = async(req, res) => {
25 29     // Save User to Database
26 30     console.log("Processing func -> SignUp");
27
28     - const { wellFormed, validDomain, validMailbox } = await emailValidator
31 + const name = req.body.name
32 + const username = req.body.username
33 + const email= req.body.email
34 + const contactNumber= req.body.contactNumber
35 + const password = req.body.password
36 + const roles = req.body.roles
37 +
38 + const { wellFormed, validDomain, validMailbox } = await emailValidator
29 39     if(wellFormed==true && validDomain==true) {
30 40
31     - User.create({
32     -     name: req.body.name,
33     -     username: req.body.username,
34     -     email: req.body.email,
35     -     contact_no: req.body.contact_no,
36     -     password: bcrypt.hashSync(req.body.password, 8)
37     - })
41 + userDao.createUser(name,username,email,contactNumber,password)
38 42     .then(user => {
39     - Role.findAll({
40     -     where: {

```

Figure 12.5: Merging Two different Version of code in Bitbucket

A data science specific problem with version control is to manage the code of Jupyter or Zeppelin notebooks. The notebook generates HTML pages and can be version managed by the Git system. For production, the code written

in Python/R or other languages is treated the same as other application development code.

Quality Assurance

Quality Assurance (QA) is a holistic term used for managing the desired quality of the application and makes sure it delivers the desired outcomes. A substantial part of QA is testing. For a data scientist testing can be broadly of two types:

- **Unit testing:** This is a testing mechanism adopted from the software development world where the test script tests that the script works as per the desired input/output pair. It does not check for logical flaws, but make sure the script works for a pre-defined case.
- **Model testing:** This testing is to make sure that the quality of data and prediction mechanism is working as expected by the offline analysis done by the data scientist. If the results deviate materially from expectations, the model cannot be applied/deployed in production.

Model testing plays a very important role as there are multiple versions of algorithms, different libraries, floating-point issues, convergence speed, and other environment-related conditions that differ between a data scientist's development environment and production environment.

It is important to test that the model performance and/or other metrics provide the same result as they do in a development environment for a given test data. [Figure 12.6](#) shows a basic test of performance between development and production environment; they both must be same to be able to get deployed:

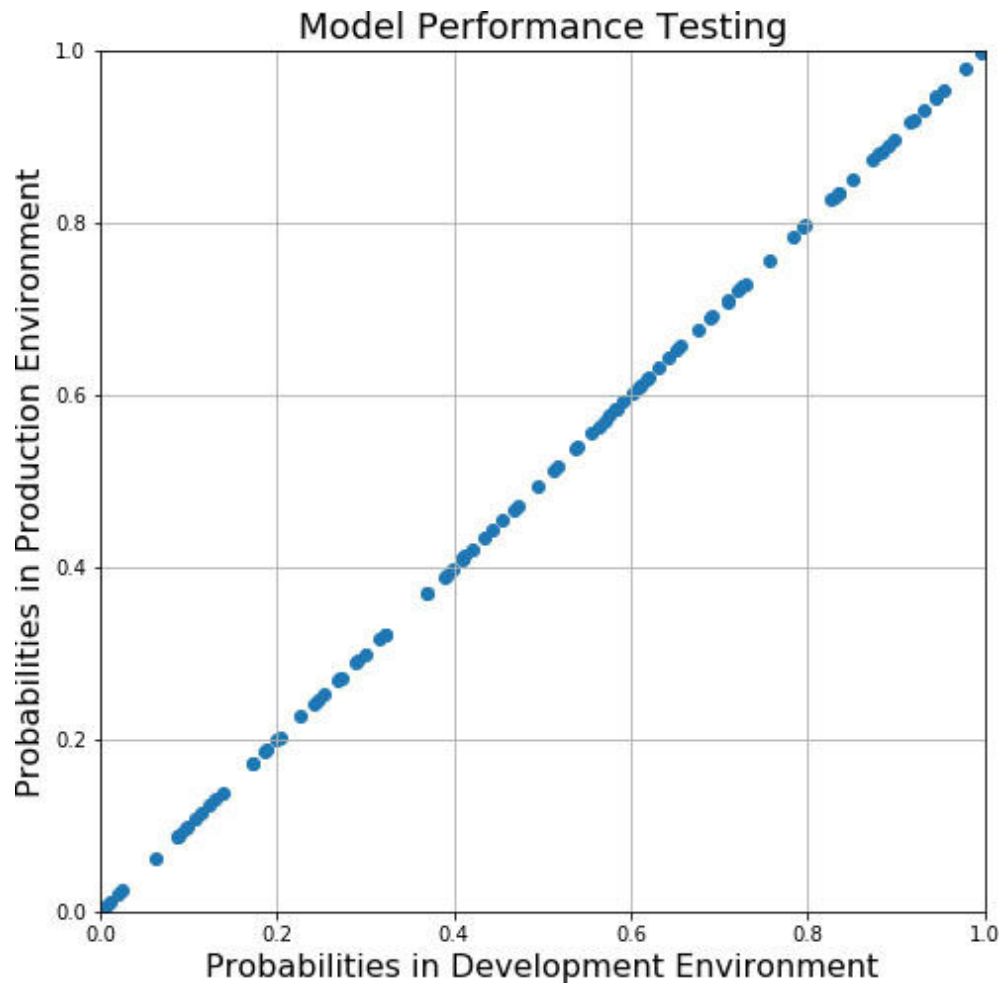


Figure 12.6: *Model Performance Testing in both Development and Production Environment*

QA has many scenarios that ask for automated testing rather than a manual testing set-up. Automated testing means that the test cases are written, and the system keeps testing automatically when the changes are encountered in the application, or new build is required.

There are multiple tools and methods of writing tests for your data science applications. It is an important part of data scientist code to provide test cases and the ability to test their model automatically. One such popular tool is Jenkins (<https://jenkins.io/>), it allows you to schedule tests, assign specific branches from a version control repository to test, and in case something breaks emails you. All the automated test process keeps running in behind without manual attention required. [Figure 12.7](#) is a snippet of the Jenkins home page:

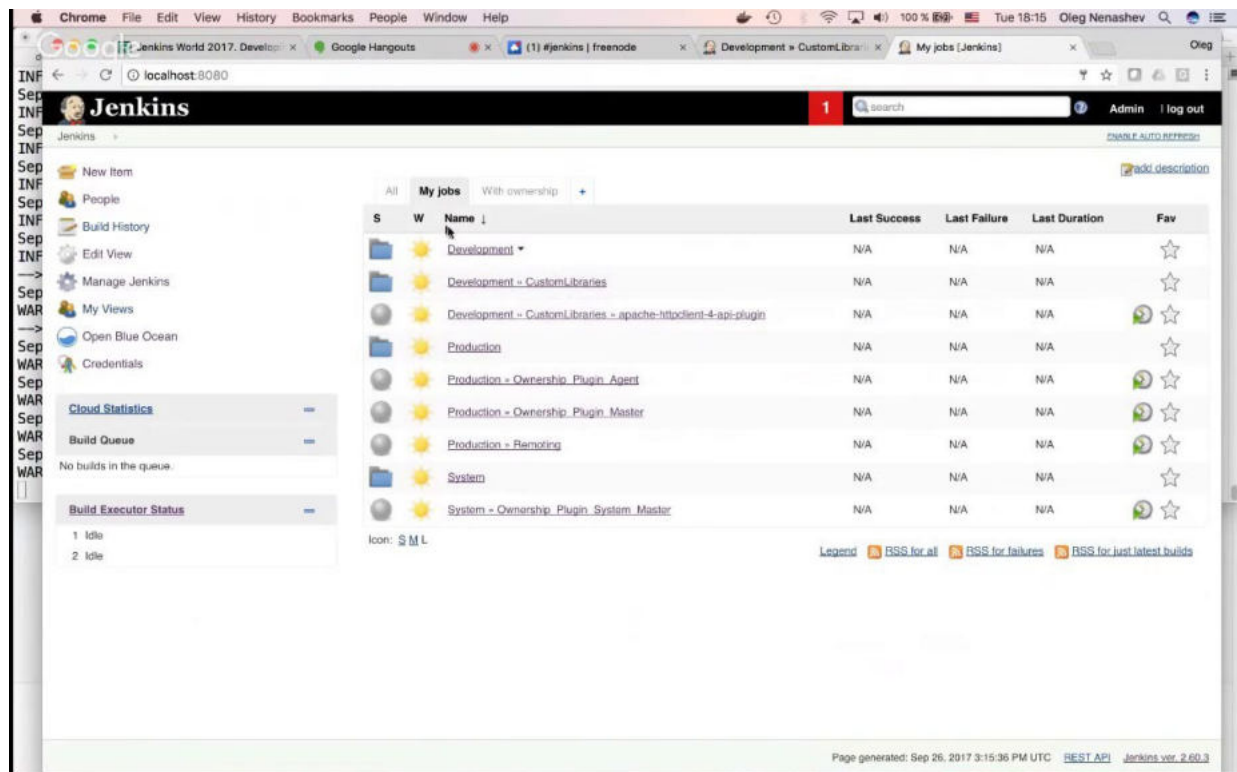


Figure 12.7: Jenkins Home Page

Unit tests are very common. JUnit (https://www.tutorialspoint.com/junit/junit_test_framework.htm) is provided for Java users. For Python developers, unittest (<https://docs.python.org/3/library/unittest.html>) is available. Nevertheless, developers can forget to execute the unit test appropriately before codes are put into production. Although crontab (<http://www.adminschoice.com/crontab-quick-reference>) can be used to run an automated test, specialized applications, such as Travis CI, Circle CI (<https://circleci.com/>) or Jenkins (<https://jenkins.io/>) are safer to be used.

Model objects and security

The trained models are usually delivered at model objects, which are a type of binaries that can be loaded in the production system for scoring/prediction on new data. There are multiple methods of creating model objects and delivering them to operations teams for deployment. In the Python world, the pickle package allows us to store the model from the current environment and then can be later loaded into another environment.

Sometimes, we also deliver the prediction model in a running API service, which has the running exe but not exposing the code.

The model objects contain sensitive information and cannot be trusted for cross-platform usage if the source of the object is not trusted. For instance, you cannot download a model object from the internet and run that, it could be malicious and cause system breakdown. At the same time, if someone gets access to your model object, he can open the model and infer training data.

Although Security is very important, it is often overlooked in the data science world. There are many cases where the data used for model training and analysis contain sensitive information like credit card or insurance records, healthcare data, requiring regulatory measures, such as GDPR, and HIPAA. When deploying at client servers, the model structure and variables require a level of authentication. Therefore, very often before they are stored in the client database, a model is deployed as an encrypted executable.

You need to be aware of encryption and decryption methods before using them for a production build. [Figure 12.8](#) below show an encrypted file looks like; even if someone gets access to model files, he would not be able to understand the model object or its content:

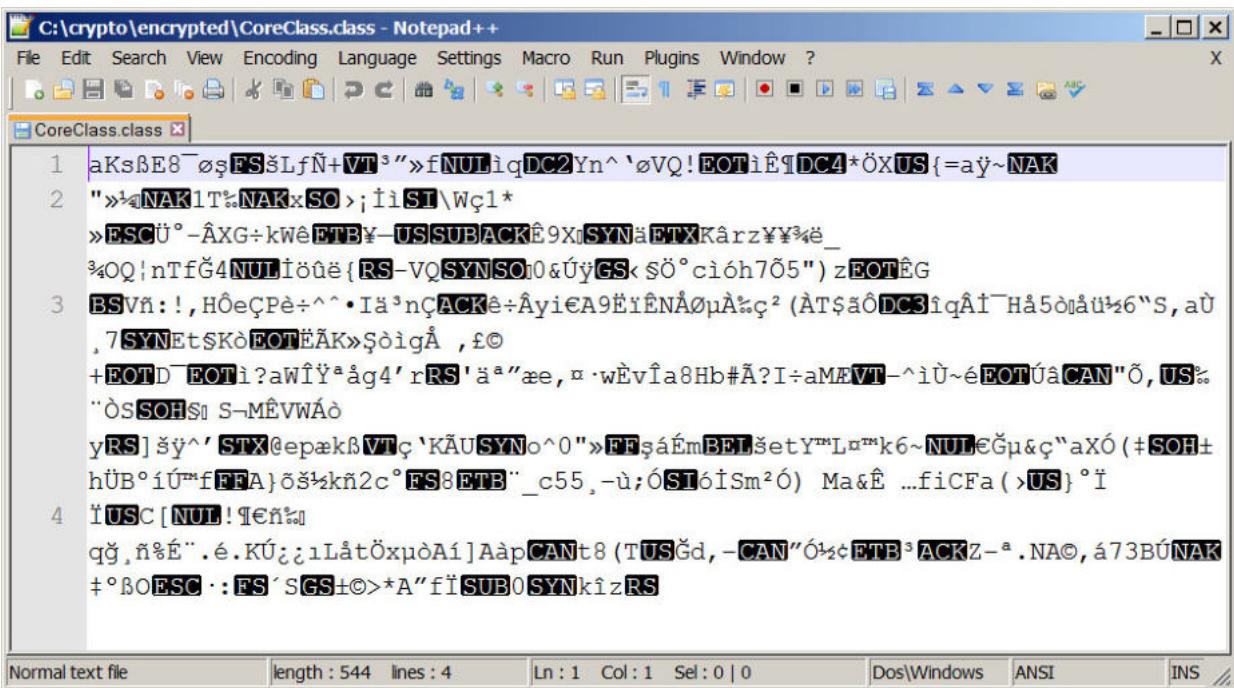


Figure 12.8: Encrypted JAR file

Human is always the weakest link in data security. It is important that data scientists are also made aware of the security protocols around the usage of data. Production standards for delivering output in an encrypted format. It is not practically possible to achieve perfect security; however, a data scientist needs to make sure they are not the weakest link to the application and data security for the company.

Production deployment

Production deployment of models is specific task that the IT operations team does in enterprise IT infrastructure. There are two keys considerations that the IT team takes sin for production deployment of models:

- Provisioning hardware/servers (physical or virtual)
- Setting up the environment

Hardware or servers in logical terminology are the machines that will be running the model code, whilethe environment is the application set-up required to run the code like Python3.6 and its dependencies. The first step to deploy is to get the computational resources ready, and that can be done by either provisioning virtual machines or actual hardware. The actual hardware will then be configured to just run that application and its dependencies. In modern applications, the cloud is preferred place for deployment on **Virtual Machines (VMs)**. You will learn more about cloud computing in upcoming chapters. [Figure 12.9](#) shows the conceptual model VM have, which in turn allows multiple applications running on the same infrastructure:

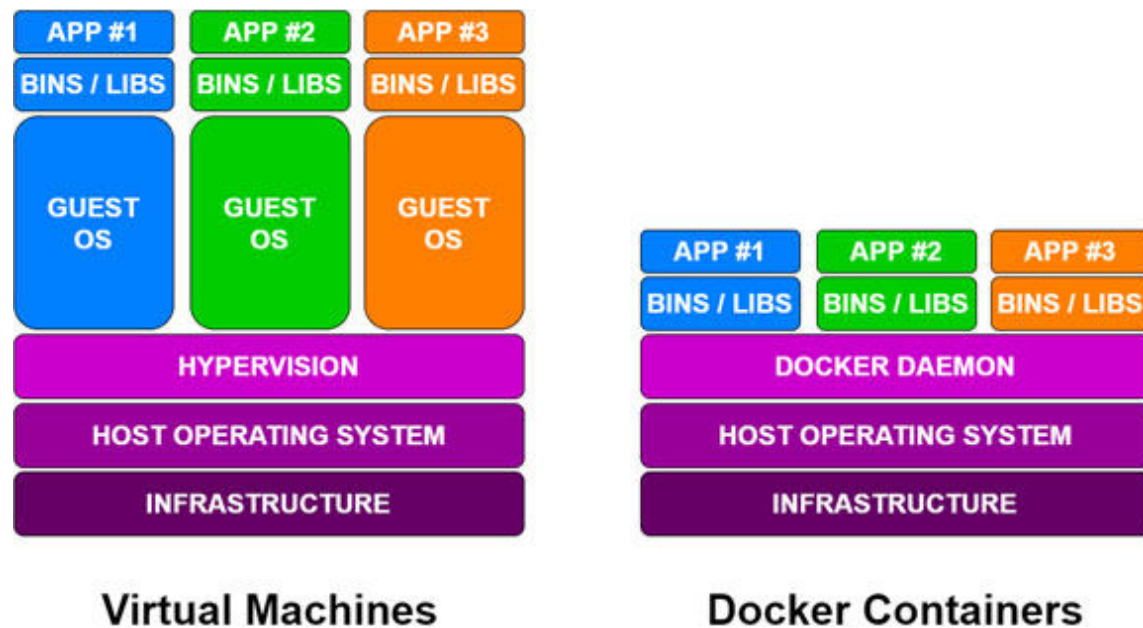


Figure 12.9: VMs vs. Docker Containers

The other way to run an application is by containerizing the applications and use Docker to run those containers. A container contains the operating system, Docker daemon, the dependencies, and the applications packaged into one unit. You just need to deploy this container onto a system running any OS, Linux, Windows, or CentOS or other.

Before the model goes to production, it needs to be tested for the environment in production, how the model will behave in that environment or efficiency of the code. This can be done by creating a VM with Virtual Box (<https://www.virtualbox.org/>) or using docker popular containerization technologies include Docker (<https://www.docker.com/>) or Kubernetes (<https://kubernetes.io/>).

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. You can deploy multiple applications and manage them using Kubernetes cluster manager. [Figure 12.10](#) shows an example of Kubernetes Cluster:

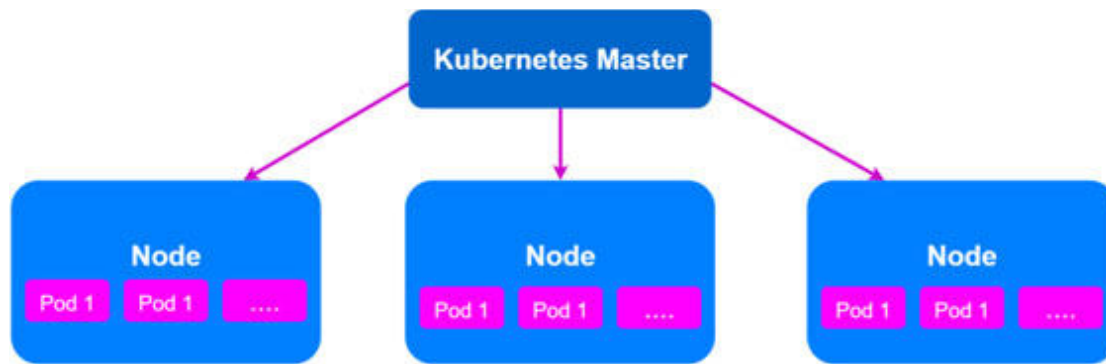


Figure 12.10: Kubernetes Cluster

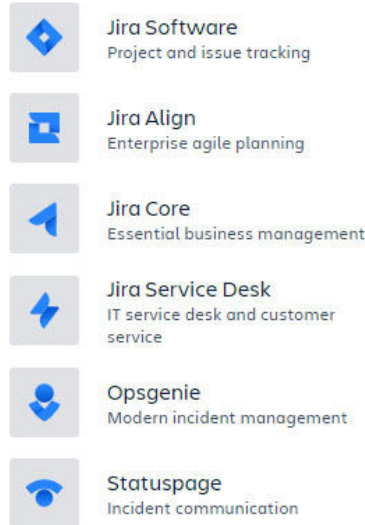
Containerization helps with scalability. This is particularly true when model-based software gets used by multiple users for training or prediction.

Communication and collaboration

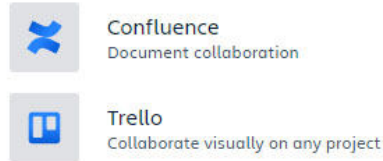
As a data scientist, it is relevant to be up-to-date with needs and expectations when dealing with DevOps or IT, which include programming languages, package versions, and many more. Indeed it is extremely difficult for DevOps and Data Scientists to tackle challenges. Data scientists are not specialists in DevOps; similarly, DevOps professionals have less knowledge in data science. Therefore, communication is important for profitable business results.

A set of right tools and mindset among team members is important to realize the power of DevOps. The tools available are aplenty and depend on the maturity stage of your team and product which tools to pick. [Figure 12.11](#) shows one of the popular toolsets provided by Atlassian for software development, equally useful to manage complex data science workstreams as well:

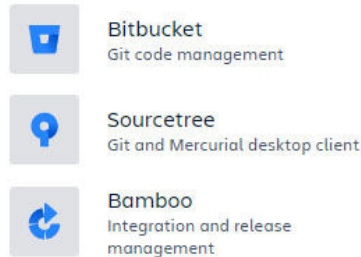
PLAN, TRACK, & SUPPORT



COLLABORATE



CODE, BUILD, & SHIP



SECURITY & IDENTITY

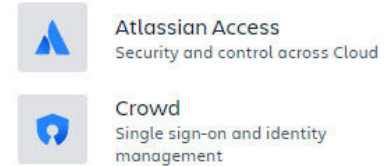

[View all products](#)

Figure 12.11: Atlassian Toolkit

The toolset includes project management, issue tracking, service desk, documentation, and all other relevant tools to allow teams to ship products faster.

Conclusion

In this chapter, we have introduced DevOps, explaining its functions and benefits. The three terms, Agile methodology, CI/CD, and DevOps, are then discussed. The functionalities of DevOps, such as integration, testing, packaging, and deployment, are discussed extensively. The source code management is then explained using examples, such as Git, Bitbucket, and many more. Further, we have talked about unittest and available platforms for performing the same. Security in production and scalability using containerization is then discussed. Thus, we have learned how to bring data science applications into operation. In the next chapter, we will introduce Cloud Computing and discuss its functionalities.

CHAPTER 13

Introduction to Cloud Computing

Storage and analysis of the huge amount of data is now the priority of organizations, big or small. Now, this has caused a big challenge for companies, as they have limited storage and computation power, and on top of it, it has limited staff to manage infrastructures. Cloud computing comes into this space to solve these issues for organizations and allow them to make the best of data.

In previous chapters, we learned about machine learning, data pipeline, and DevOps to bring data science applications into operations. In this chapter, we will introduce cloud computing and its role in enabling all that we have discussed before. Beyond facilitating the storage and computation challenges, as the cloud is paying as go, they have opened opportunities for all size organizations.

Structure

- Operating system model
- Types of cloud services
- Types of cloud infrastructure
- Data science and cloud computing
- Market growth of cloud

Objectives

After studying this unit, you should be able to:

- Understand the concepts of cloud computing, OS model, and virtualization
- Deal with the types of cloud computing and cloud infrastructure
- Understand how cloud computing and data science growth is coupled

Introducing cloud computing

Every computing device has some basic set-up over which the high-level programming works, including what all we have discussed using Python in previous chapters. To have a comprehensive understanding of cloud computing, we need to build an understanding of how computation happens in a machine or servers. If you're on a bus or train, you're buying a ticket and hanging on to your seat until you get to your destination. Also, other passengers will travel with you on the same bus, and you will hardly be disturbed where they are going. You get off the bus when you arrive, and thank the driver. Like this bus, cloud computing delivers data and information to different users and enables the technology to be used at reduced prices.

Operating system model

The simplistic way to look at the whole OS model is presented in [Figure 13.1](#). It manages computer hardware, software resources, and provides common services for software programs.

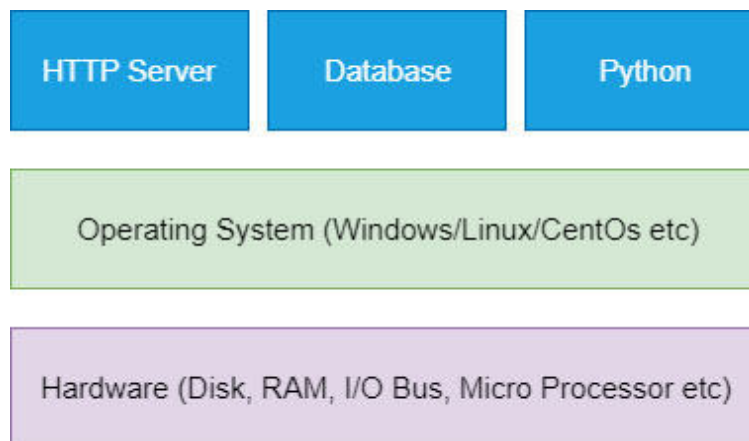


Figure 13.1: Operating System Model

It is important to understand the role of an operating system in the OS model. OS provides all the necessary functionalities for the application layer to interact with the hardware. The application layer includes your https servers, Microsoft Word, databases, and scripting languages like Python, R, Java, and many more.

When we talk about cloud, we talk about de-coupling these important layers and hence allow scalability and distribution of resources at a logical layer.

You must have observed that when you buy a PC/laptop, you do not need to buy and configure any hardware and also don't need any OS to install onto it. You get a machine that starts all your applications out-of-the-box.

Cloud computing intends to do the same over the internet. You pay for cloud access, and you can immediately start working on your machine, hosted and maintained by your cloud service provider.

In traditional system designs, the hardware of a computer and the OS are tied to each other, and hence to horizontally scale; you need to invest in hardware and also OSes. You must have heard that your system either can run Windows or Linux at a time. This was a huge limitation that did not allow dynamic OS provision on large shared hardware. Virtualization removed this constraint and made the cloud possible.

What is virtualization?

The separation of hardware layer from the OS layer will require an intermediate layer, which creates a virtual layer for OS to run on top of it. In the early days, hardware emulators were used to running OS inside an OS; also, in later years, we had VM Ware set-ups that can run a virtual environment inside Windows OS to host other operating systems. However, to enable the full potential and performance of multiple OS on the same hardware required a virtualization layer to be present between multiple OS and hardware. Virtualization does the same thing by creating that virtual layer between two parts of the OS model, such as an operating system, a server, storage, or network systems.

Virtualization allows multiple operating systems and applications on top of those instances to run concurrently on a single computer and by enabling a virtualization layer (or a Hypervisor) on the computer hardware. [*Figure 13.2*](#) shows such a set-up where three different OS are running different applications on the same hardware:

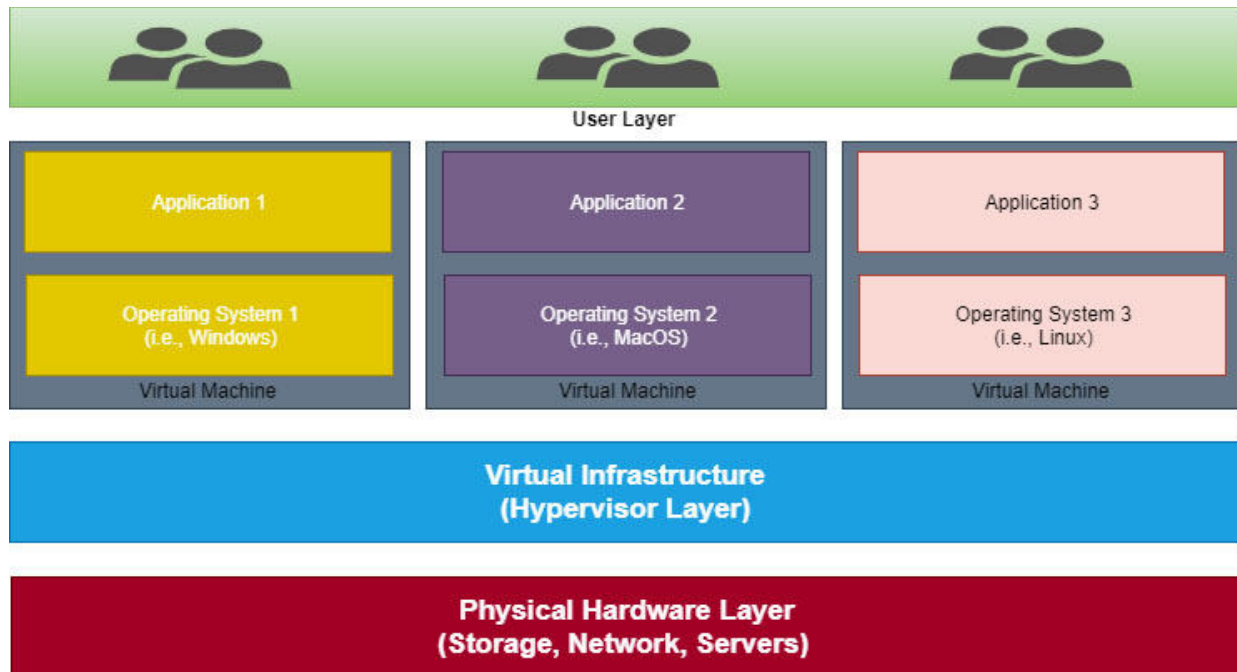


Figure 13.2: Virtualization Layers

Some of the key terms from a virtualized OS model are discussed below:

- **Physical hardware layer:** This is the physical machine having RAM, storage, I/O bus, network cards, microprocessors, routers, and other peripheral devices.
- **Virtual infrastructure layer:** This is also called the hypervisor layer. It can be thought of as an OS itself, which is customized to be a virtual layer between the hardware and OS running inside it. A hypervisor is a program that creates VM, which behaves exactly the same as a complete machine for the incoming OS. The hypervisor manages the actual physical resource share, such as memory, I/O, Network, and processing. It ensures that a completely segregated and secure environment is proving to all host OSes.
- **Virtual machine layer:** This is the layer where numerous virtual machines are created. These machines are like a fresh PC/laptop, ready to be configured with OS of choice. In terms of cloud, these are VM instances or servers running in a virtualized environment. It hosts the OS the applications they are configured to run.
- **User layer:** The user interacts with the applications hosted in virtual servers like as if it controls both OS and underlying hardware. The end-

user will never be able to differentiate between a virtual or physical server as this does not impact the application.

The virtualization layer allows the centralization of hardware and dynamically provision resources as per the demand of the user. This technology allows building Virtual Data Centers with just a bunch of high-power computing hardware. For example, a powerful server with 100 GB of RAM is good to run 25 virtual windows machines each with 4 GB RAM. So rather than buying 25 PC/Laptop, you can work with 1 server and minimal client devices.

What is cloud computing?

Cloud computing can be thought of as the virtualization done at large scale with very powerful hardware managed professionally in big physical data centers. Cloud computing is more of a popular term used to describe a service where the end-user does not need to manage the underlying hardware, where the hardware is hosted, the networking access, or the software. All the computing needs are provided over an internet browser in a matter of clicks. A generic framework of cloud computing is shown in [*Figure 13.3*](#):

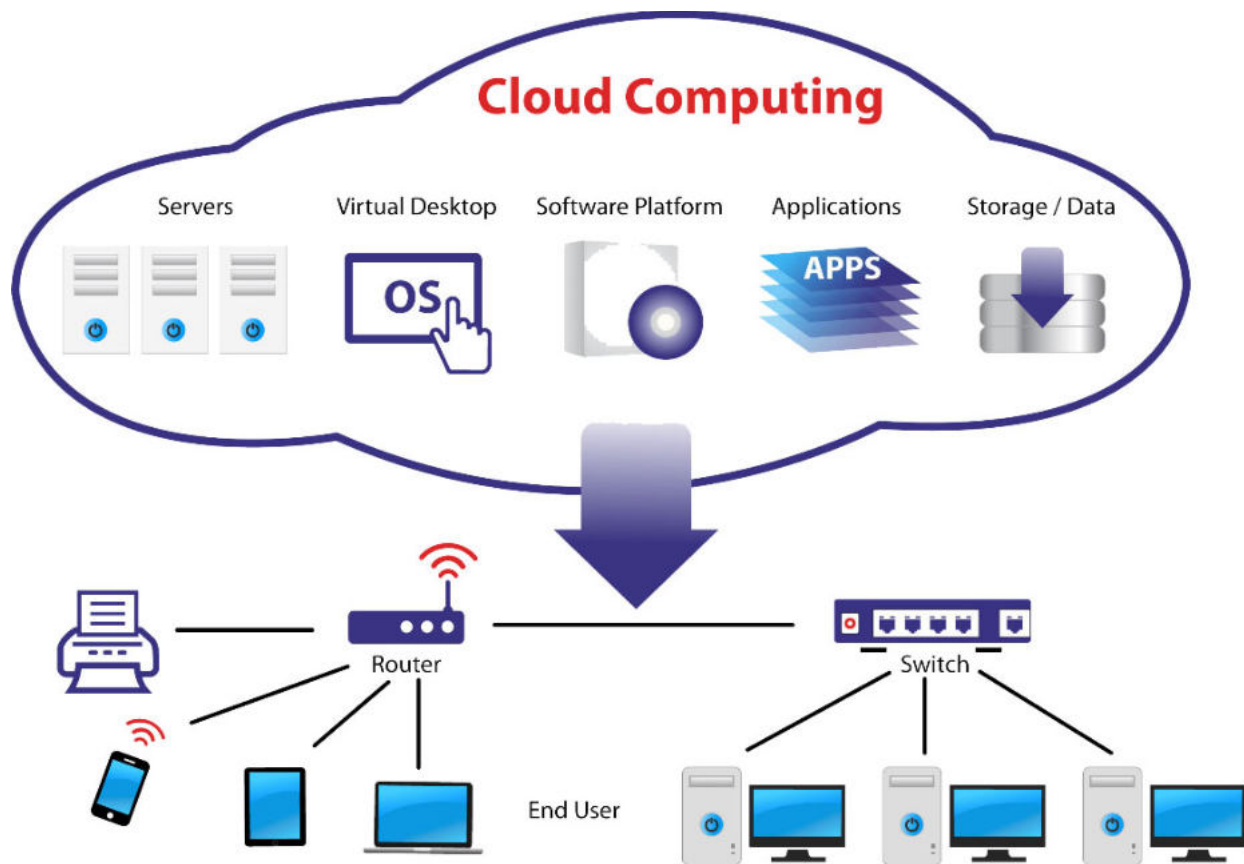


Figure 13.3: The Cloud Computing World

By decoupling hardware and software, theoretically, the virtual platform can scale to infinity if the data center has the required hardware. The resources and access can be dynamically provisioned, and virtual resources can be added, deleted, and modified on the fly. All the tools and access are provided through web browsers, and the interface is exactly the same as if you were using those computing resources locally.

While virtualization is the program that makes multi-tenant hardware, the concept of infrastructure convergence and shared service allows cloud computing, as we see now in Google, AWS, and Azure. It allows enterprises to get their applications deployed faster, with easier manageability and less maintenance and enables IT to more rapidly adjust IT resources to match unpredictable business demand.

The cloud impact is huge on modern applications and IT infrastructure; it has brought huge savings in IT infrastructure across private and government IT infrastructure. Cloud applications have grown exponentially across the

globe due to cloud computing and better internet connectivity. Cloud adoption is happening at a fast pace across the full spectrum of businesses.

Types of cloud services

Cloud computing has stacked hardware and its manageability around application into a managed environment. This opens new types of applications that are delivered through the cloud infrastructure. There are three types of services that have been built upon the cloud computing industry:

- **Infrastructure as a Service (IaaS):** On-demand infrastructure and pay only for the time you use the infrastructure.
- **Platform as a Service (PaaS):** On-demand platform access, which covers infrastructure and platform to build your own applications.
- **Software as a Service (SaaS):** On-demand access to software hosted on the cloud and pay per use.

Modern cloud services in data science and analytics domain can be stacked, as shown in [Figure 13.4](#):

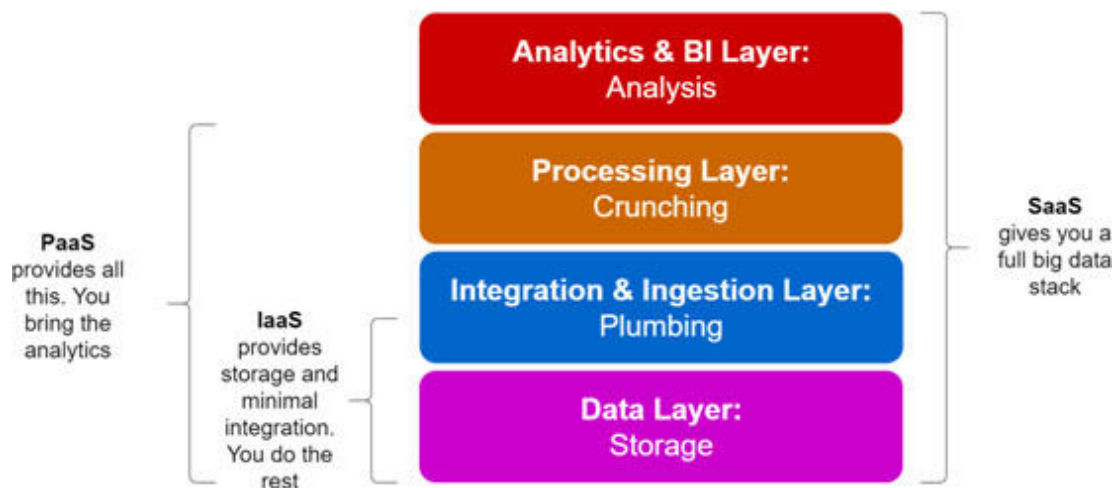


Figure 13.4: Cloud services models

Depending upon the business model of analytics delivery, the services are classified by each model as follows:

- **Data layer:** Who will provide the infrastructure, database, security, access management, and update jobs for the data to be used in

applications.

- **Integration and ingestion layer:** Who will create, schedule, execute/compute, and manage data integration and ingestion from APIs, file uploads, and ERP system.
- **Processing layer:** Who will bring value to add on the data by parsing, merging, cleaning, metadata merging, AI/ML inputs, and updates to the data.
- **Analytics and BI layer:** Who will create data visuals, dashboards, bring insights, communicate insights, publish and update the insights.

Infrastructure as a Service (IaaS)

Table 13.1 provides a detail description of IaaS:

#	Meta	Description
1	Define	Infrastructure as a Service (IaaS) is nothing but making the computing resources self-service based fully automated and highly scalable by the client itself. In this type of service, the client can access and manage the required networks, storage directly.
2	Delivery mode	Here the infrastructure will be maintained somewhere around the globe. In IaaS, virtualization technology is the core concept to deliver the resources. Such as operating systems, network, and storage. Including servers also can be provided to the client in IaaS. The client can take control over the infrastructure through dashboards and can make the changes in using API calls.
3	Characteristic	<ul style="list-style-type: none">• Entire infrastructure resources are made available as a Service.• Pay per usage.• Resources can conveniently be resized based on their needs.• Single hardware infrastructure can be utilized by multiple users.• Cloud providers can take care of managing the entire infrastructure cost depends on usage services are highly scalable multiple users on a single piece of hardware organization retains complete control of the infrastructure.
4	Advantages	<ul style="list-style-type: none">• Flexibility is the most important in this cloud computing model.• The client needs not to worry about maintaining the infrastructure. Simply they want only to use the services.

		<ul style="list-style-type: none"> • Pay per usage adds cost management very easy • Automated deployment of networks, storage, processing power, and servers. • The client has full control of the infrastructure as per their usage. • No need to bring the hardware to on-premise. • The client can have complete control of the entire infrastructure. • Easy to automate the deployment of storage, networking, servers, and processing power. • Hardware purchases can be based on consumption. • The most flexible cloud computing model.
5	Limitations	<ul style="list-style-type: none"> • The security layer between host and VMs is not in client control. • Cloud set-up may go incompatible because of the usage of legacy systems. • Internal resources and training are required to re-skill IT staff. • Multi-tenant security.
6	Best use case	In order to save time and money for buying and creating hardware and software, startups or small companies may prefer IaaS. Whereas large firms that want to scale certain of their systems shortly choose the IaaS model.
7	Example	DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

Table 13.1: IaaS

Platform as a Service (PaaS)

Table 13.2 provides a detail description of PaaS:

#	Meta	Description
1	Define	Platform as a Service (PaaS), supports to provide cloud components to specific software and delivers a full framework for developers that they can develop and build based on the user to create customized applications.
2	Delivery mode	PaaS based services provide a standard platform for software development delivered via the web console, giving developers the full concentration to building the software having concern about operating systems, software updates, storage, or infrastructure.
3	Characteristic	<ul style="list-style-type: none"> • Resources can be easily scaled up or down as it is built on virtualization technology.

		<ul style="list-style-type: none"> • It offers a range of services to help create, review, and deploy apps. • It is accessible to numerous users through the same development application. • Integrates web services and databases.
4	Advantages	<ul style="list-style-type: none"> • Simple, scalable, cost-effective development and deployment of apps. • Without worrying about maintaining the software, developers can customize apps. • The amount of coding needed will be reduced significantly. • Easy automation and migration.
5	Limitations	<ul style="list-style-type: none"> • Data security – your data is stored in remote VMs. • Vendor lock-in – once all your development happens in the cloud you might get locked-in to the vendor. • Runtime issues. • Operational limitation.
6	Best Use Case	PaaS can be very helpful in designing custom applications with simplified workflows operating on the same project with several developers while improving collaboration, deployment resources standardization, and development speed.
7	Example	AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift

Table 13.2: PaaS

Software as a Service (SaaS)

Table 13.3 provides a detail description of SaaS:

#	Meta	Description
1	Define	SaaS uses the internet to provide its customers with business applications operated by a third-party vendor.
2	Delivery mode	SaaS is distributed via the internet, removing the need for IT professionals to download and install software on every device.
3	Characteristic	<ul style="list-style-type: none"> • Managed from a centralized location. • It is hosted on a remote server/VMs. • Accessible over the internet. • Users are not responsible for hardware or software updates.

4	Advantages	<ul style="list-style-type: none"> • Out-of-the-box applications. • The deployment, maintenance, and upgrading software are no difficult process. It offers skilled staff plenty of time to spend their valuable time on urgent problems and concerns of the organization.
5	Limitations	<ul style="list-style-type: none"> • Interoperability. • Application lock-in. • Data security can be an issue. • Customization is not possible. • Feature limitations. • Performance and downtime.
6	Best Use Case	<ul style="list-style-type: none"> • Small companies or startups that need to quickly start e-commerce and have no time to deal with server issues or technology problems. • Short-term projects that need fast, easy, and affordable collaboration. • Apps that are not too often necessary, such as tax software.
7	Example	Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting

Table 13.3: SaaS

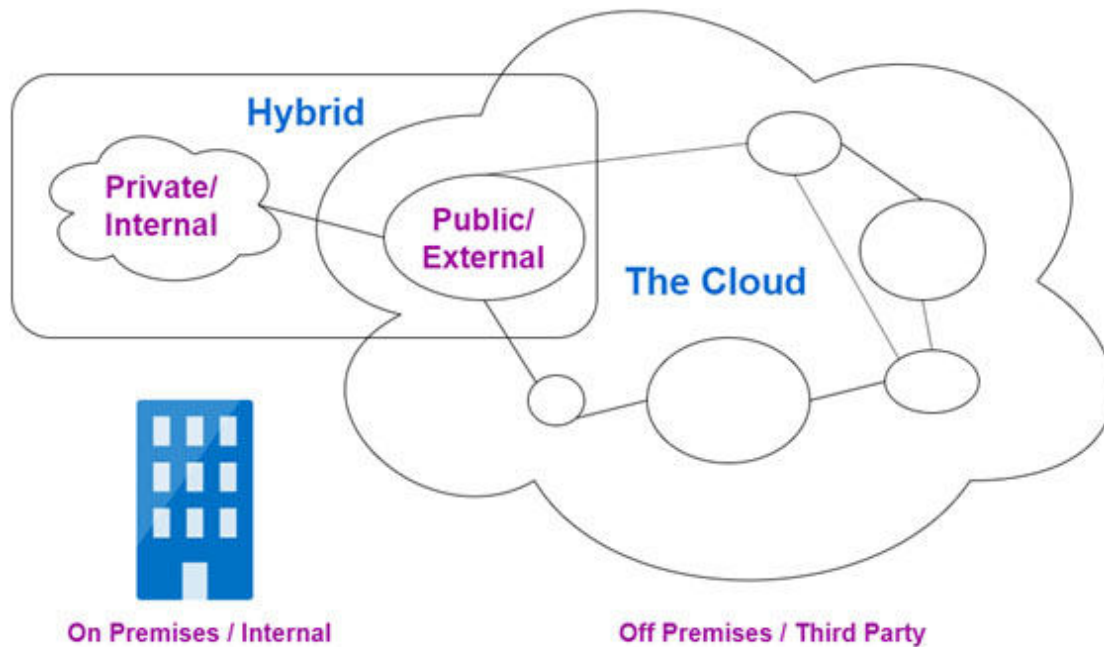
Types of cloud infrastructure

All the cloud services mentioned in the previous section are delivered through a cloud infrastructure. The type of cloud infrastructure also allows large companies to decide how much of cloud services they want to have from private vendors and how many they want to build in-house. As the technology, specifically virtualization and shared resources, is not proprietary to one vendor, it is possible to have multiple types of cloud infrastructure.

The three-key models for cloud infrastructure are:

- **Public cloud:** Independent cloud managing vendors providing service to anyone in public
- **Private cloud:** Internal cloud to an organization exclusively meant for the private use
- **Hybrid cloud:** Combination of private and public cloud, where few outside world facing applications are hosted on public cloud, while others are kept private.

[Figure 13.5](#) illustrates these three cloud, infrastructure models:



Cloud Computing Types

Figure 13.5: Cloud computing types

Public cloud

[Table 13.4](#) provides a detail description of the public cloud:

#	Meta	Description
1	Definition	The most common way to implement cloud computing is public clouds. The cloud resources are owned and operated by and delivered via the Internet by a third-party cloud service provider. With a public cloud, the cloud provider manages and administers all hardware, applications, and other supporting infrastructure.
2	Advantages	<ul style="list-style-type: none"> • No initial investments required for IT infrastructure deployment and maintenance. • High scalability and flexibility to meet unforeseen demands for the workload. • Since cloud vendor is responsible for the infrastructure management, reduced complexity, and requirements on IT expertise. • Flexible pricing choices based on various SLA offerings.

		<ul style="list-style-type: none"> • Cost flexibility enables companies to follow a lean approach for development and to concentrate resources on technology initiatives.
3	Limitations	<ul style="list-style-type: none"> • The total ownership cost (TCO) for large-scale use, particularly for medium and large companies, will increase exponentially. • Not the most viable solution for sensitive IT workloads, which are important for security and availability. • Low infrastructure visibility and control, which may not be sufficient to comply with regulatory compliance.
4	Best Use Case	<ul style="list-style-type: none"> • Predictable computing requirements for a certain number of users, such as communication services. • Software and services needed for IT and business activities. • Further requirements of resources to meet different peak requests. • Software development and test environments.
5	Example	Amazon Web Services, Google Cloud Platform, Microsoft Azure, and Alibaba

Table 13.4: Public Cloud

Private cloud

[Table 13.5](#) provides a detail description of a private cloud:

#	Meta	Description
1	Definition	The private cloud refers to a single organization's cloud solution that uses data center resources on site or operated by a third-party provider on site. The computing assets are isolated and shared with other customers through secure private networks.
2	Advantages	<ul style="list-style-type: none"> • Environments committed and safe that other organizations cannot access. • Compliance with strict regulations as organizations will conduct protocols, configurations, and safety assessments based on unique requirements for the workload. • High scalability and flexibility to meet volatile safety and performance requirements. • Efficiency and high SLA efficiency. <p>Flexible infrastructure transformation based on the organization's continuously changing business and IT needs.</p>
3	Limitations	<ul style="list-style-type: none"> • Compared to public cloud alternatives for short-term purposes, the costly solution is a high cost of ownership.

		<ul style="list-style-type: none"> • With the high-security measures in place, mobile users can have limited access to the private cloud. <p>If the cloud data center is limited to on-site computing resources, the network cannot offer high scalability to meet unpredictable requirements.</p>
4	Best use case	<ul style="list-style-type: none"> • Heavily regulated sectors and agencies of government. • Technological organizations need strong control and security over the underlying infrastructure and IT workloads. • Big companies that require the efficient and cost-effective operation of advanced data center technologies. Organizations that can invest in high-performance technology and accessibility.
5	Example	Private Data Centers for companies running out of dedicated locations

Table 13.5: Private Cloud

Hybrid cloud

[Table 13.6](#) provides a detail description of the hybrid cloud:

#	Meta	Description
1	Definition	Hybrid cloud refers to the mix of public and private solutions that are part of a cloud infrastructure environment. Usually, services are structured as an integrated network environment. Applications and data workloads are capable of sharing assets between public and private cloud deployment in compliance with operational and technological security, reliability, scalability, cost, and productivity policies.
2	Advantages	<ul style="list-style-type: none"> • Sustainable policy implementation, with security, quality, and cost demands as the basis, to disperse workloads across public and private networks. • Without exposing critical IT workloads to the inherent security risk, scalability in public cloud systems is accomplished. • High reliability as the services are allocated over various data centers through public and private data centers. • Enhanced security as critical IT workloads run on private cloud dedicated resources while ongoing workloads are spread throughout low-cost public cloud infrastructure to compromise cost expenditure.
3	Limitations	<ul style="list-style-type: none"> • It could be costly. • Solid compatibility and integration between the different locations and categories of cloud infrastructure are

		<p>required. This is a constraint on public cloud deployment for which companies do not have direct control over the infrastructure.</p> <ul style="list-style-type: none"> • A further layer of infrastructure complexity is added as companies handle and run a modern mix of private and public cloud architecture.
4	Best Use Case	<ul style="list-style-type: none"> • Various hierarchical organizations with different IT safety, regulatory, and quality criteria. • Optimize infrastructure development without jeopardizing public or private cloud technology value offerings. • Enhance the safety of current cloud solutions such as SaaS that have to be delivered through secure private networks. • Cloud resources were deliberately addressed to move and swap between the best cloud service distribution system available on the market.
5	Example	Website of travel agency hosted on AWS but booking system on private cloud

Table 13.6: Hybrid cloud

Data science and cloud computing

Cloud computing and its affordability have the largest impact on how data science growth has been in the last decade. Cloud computing has impacted all parts of a data science process and enabling organizations and society to make the best use of the data. Data Science community appreciates the fact that cloud is the medium that has taken the benefit of ML and AI (subbranches of data science) into everyone's reach as a SaaS application on mobile or web.

The data science enabled or building blocks can be divided into four (04) blocks; each block is now enabled by cloud technologies. This can be seen as follows, as shown in [Figure 13.6](#):

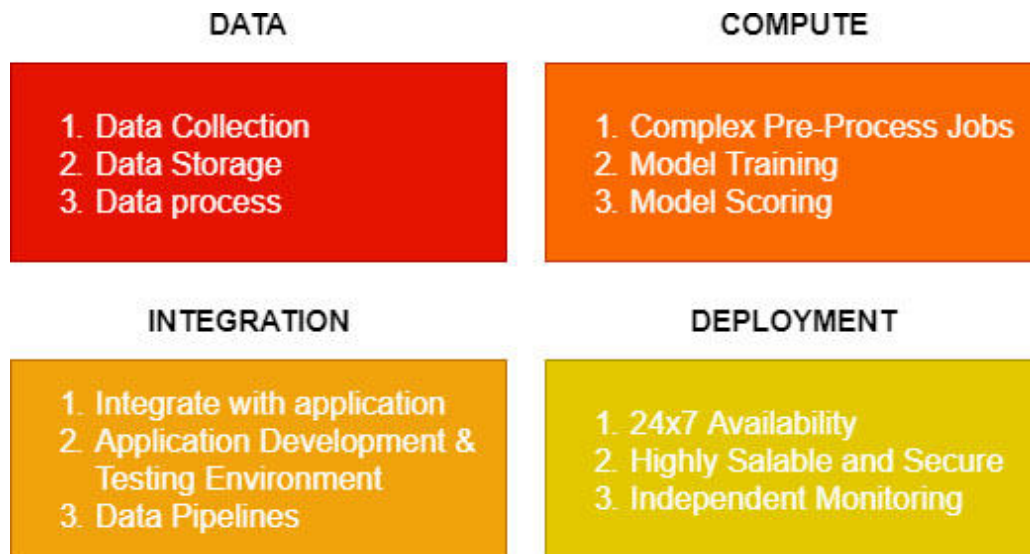


Figure 13.6: Cloud and Data Science

The four quadrants bring out the importance of cloud in the data science fields. They are discussed in brief below for the reader to understand the use of the cloud in each building block.

Data

The data is growing at a massive size, and the process of collecting it is also very diverse. There could be big data use case where the data is in Terabytes, which make it impossible to work in local machine or one machine infrastructure. Cloud computing allows a large amount of distributed storage resources to store big data.

In terms of data collection and process, the data input sources are varied; some come from APIs, some come from databases, some just file sand so one. The cloud provides the flexibility and easy provision of resources to handle such variety.

Compute

There is a limitation of a single machine for computation, and there is a high cost for machines with high computation resources. The price from a 16 GB RAM machine to 32 GB RAM machine to 64 GB RAM machine grows exponentially. For complex models, like deep learning and XGboost, it is either impossible or takes hours of CPU time to train the model. The cloud

provides powerful machines at the click of the button to run such high compute model training jobs.

It also saves money as the local hardware utilization is very low; cloud optimizes hardware utilization by using the shared resources. For example, you might need a 64 GB RAM machine for 2 hours, and rest time, it can be allocated to other users.

Integration

A data science model itself is not useful for an end-user who has no knowledge of the internal working of models. For an end-user, an AI/ML model is delivering using insights through easy-to-use the application through his/her browser or mobile phone. To enable such ease of use, the data science model needs to be integrated with applications.

Cloud provides the full end to end platform to enable the integration of applications with data science models. The data can be collected, stored, and streamed to the data science model in an integrated cloud environment.

Deployment

The AI/ML models, once integrated with the applications, need to be made available to end-user in a highly scalable and secure environment 24×7. This can be done by provisioning and managing IT infrastructure to operate 24×7. However, only large corporations can afford that kind of IT costs. The public cloud helps even small companies to deploy the applications and pay per use.

The pay per use model has been very successful as well, as it takes away the capital expenditure in IT resources away, and rather become an operating expense. This has been one of the biggest reasons for the innovation and growth of SaaS products. Even a small startup can develop and deploy an enterprise-ready application.

Market growth of cloud

Gartner anticipates global public cloud revenue to rise in 2019 by 17.5% and expects that the cloud services industry should evolve rapidly by 2022. According to Gartner, Inc., the worldwide demand for public cloud

computing services is expected to grow to \$214.3 billion by 17.5% in 2019 from \$182.4 billion in 2018. In 2019, it had been expected to grow by 27.5% to \$38.9 billion compared to \$30.5 billion in 2018 and will be a growing market for the cloud infrastructure or IaaS, as shown in [Table 13.7](#). Cloud application infrastructure services or PaaS will achieve the second-highest growth rate of 21.8 percent:

	2018	2019	2020	2021	2022
Cloud Business Process Services (BPaaS)	45.8	49.3	53.1	57.0	61.1
Cloud Application Infrastructure Services (PaaS)	15.6	19.0	23.0	27.5	31.8
Cloud Application Services (SaaS)	80.0	94.8	110.5	126.7	143.7
Cloud Management and Security	10.5	12.2	14.1	16.0	17.9
Cloud System Infrastructure Services (IaaS)	30.5	38.9	49.1	61.9	76.6
Total Market	182.4	214.3	249.8	289.1	331.2

Table 13.7: Worldwide Public Cloud Service Revenue Forecast (Billions of U.S. Dollars)

[Figure 13.7](#) shows the survey result from RightScale, on how the public cloud adoption was for major cloud vendors in 2019. Amazon Web Services leads the cloud space with the highest adoption rate and the plethora of service. All cloud vendors also have dedicated AI/ML services as well, provide complex deep learning pre-trained models for a complex task like object detection, sentiment analysis, and many more:

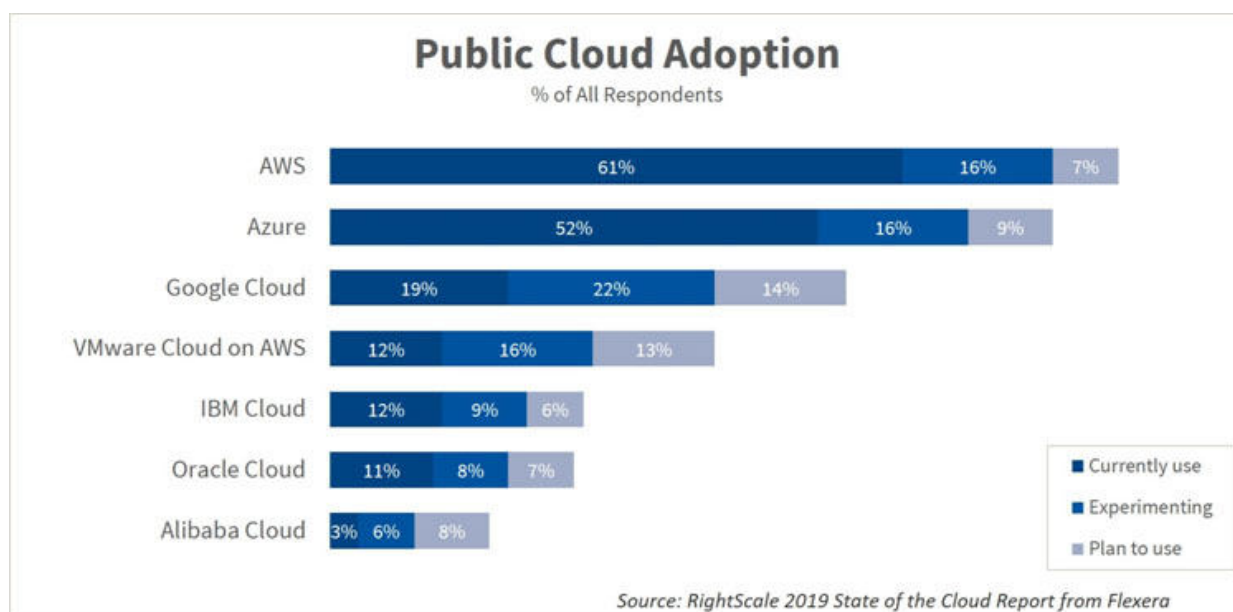


Figure 13.7: RightScale report of the Cloud Share (Credit: RightScale 2019 State of the Cloud Report from Flexera)

It is now important for data science professionals to have the capability to work on the cloud for their routine work as most of the organization has already adopted cloud for their storage and computation needs. Having a good understanding and skills in cloud computing will be an added asset to a data scientist.

Conclusion

In this chapter, we have introduced the concept of cloud computing and what are the key enablers to cloud growth. The chapter discussed the OS model to explain the user, how a machine is logically designed, and divided into major working blocks of hardware, operating system, and applications. The next section then discusses the concept of hypervisor, which can create a layer of virtual hardware, allowing multiple host OS and applications to share the same hardware. This whole concept of virtualization allows the infrastructure to be separated from applications. The cloud then can be created and provide access to shared resources as IaaS, PaaS, or SaaS model. After discussing the service models of cloud, the chapter also discusses the type of cloud infrastructure. That is public, private, and hybrid. The next section then highlights how cloud computing and data science growth are coupled across four building blocks of data science, that is, data, compute, integration, and deployment. In the end, we have also highlighted the market trends and growth prospects of the public cloud. By reading this chapter, the reader has gained an understanding of the basics of cloud computing, OS model, and virtualization. He or she has understood how to deal with the cloud infrastructure. In the next chapter, we will show a hands-on demo for the deployment of a data science application on Google Cloud Platform.

CHAPTER 14

Deploy Model to Cloud

Google Cloud Platform (GCP) enables developers to build, test, and deploy applications on Google's highly scalable and reliable infrastructure. It provides powerful tools and services. Many companies are moving to the cloud to help create software applications and manage data, and GCP is one of the many platforms available for cloud computing.

In previous chapter, we have learned about cloud computing, OS model, virtualization, and cloud computing services and infrastructures. In this chapter, we will introduce GCP and its console. We will start by a free GCP account and explore its features and functionalities.

Structure

- Register for GCP free account
- GCP console
- Create VM and its properties
- Connecting and uploading code to VM
- Executing Python model on cloud
- Access the model via browser
- Scaling the resources in cloud
- Conclusion

Objectives

After studying this unit, you should be able to:

- Understand the GCP platform
- Use the GCP console and its features
- Deploy and execute machine learning model in cloud

[Register for GCP free account](https://cloud.google.com/free/#always-free)

Google's infrastructure provides four options for creating applications in the cloud:

- Google Compute Engine
- Google App Engine
- Google Kubernetes Engine
- Google Cloud Functions

To start using the GCP, you need to have a GCP account. You will need to sign up for it at <https://cloud.google.com/free/#always-free>. [Figure 14.1](#) is the home page of the GCP:

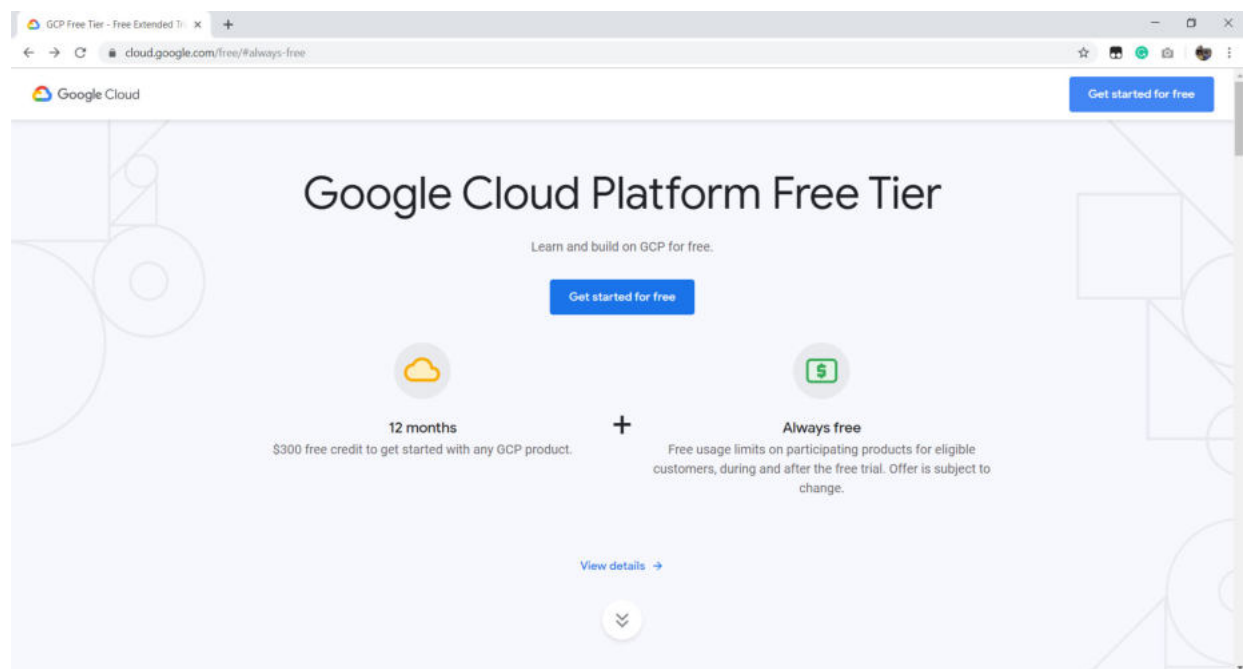


Figure 14.1: Google Cloud Platform (GCP) free tier home page

Click the **Get started for free** button and provide your requested information. Upon signup, you will be benefited with the below:

- **A 12-month** free trial with \$300 credit to use with any GCP services.
- **Always free**, which provides limited access to many common GCP resources, free of charge.

Now you are all set to create your first GCP Project. On the top right corner, click the Console button. [Figure 14.2](#) shows the Console to access, which is located in the top right of the window.

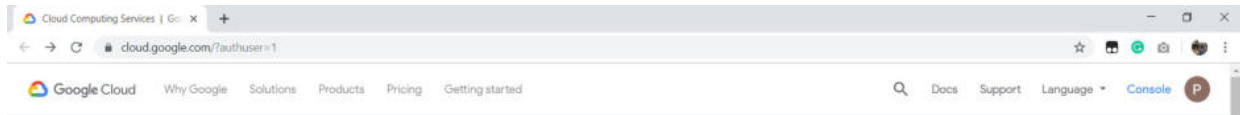


Figure 14.2: Accessing Console from GCP Console

You will then be directed to the GCP Console. Initially, we will need to create a new project.

[GCP console](#)

On the top left, near the GCP button, click the three-dot icon. [Figure 14.3](#) is the pop-up window in the GCP console to choose a project:

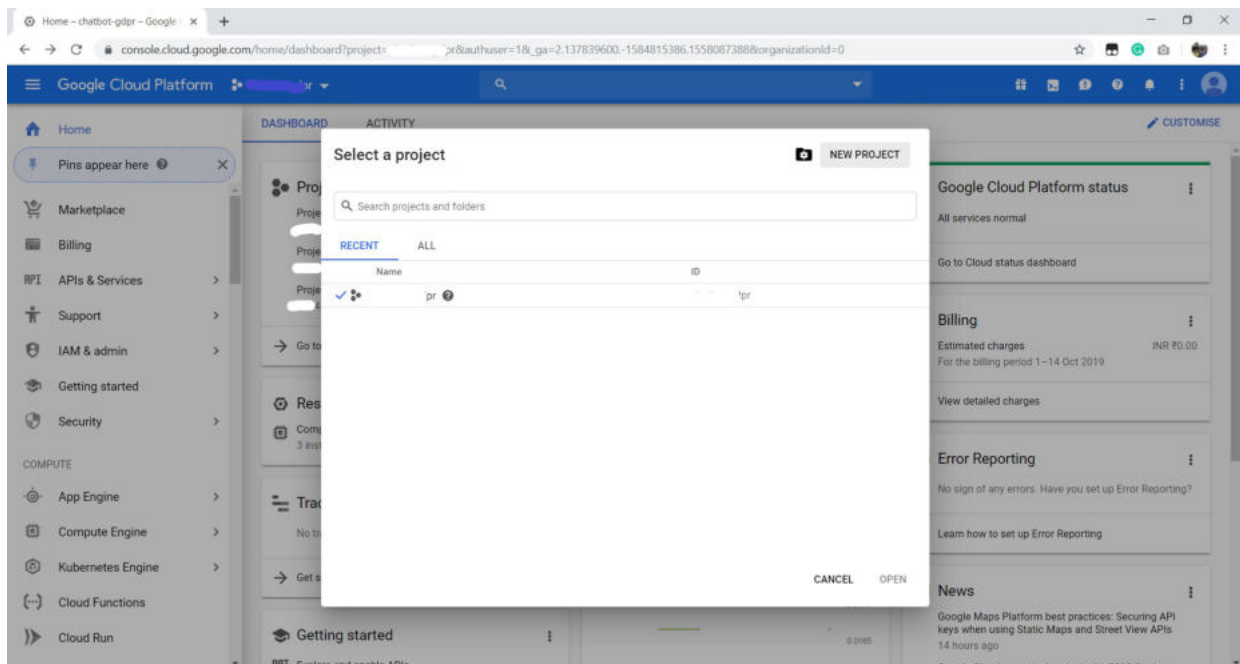


Figure 14.3: Choose the project in the GCP Console

The console has a **Dashboard** and an **Activity** window. Of any selected project, the Dashboard provides quick information such as **Project Info**, **Resources**, **Compute Engine**, **APIs**, **Google Cloud Platform status**, and many more. The **Navigation** panel on the left shows a list of GCP products and services, such as **Compute Engines**, **Storage**, **Networking**, **Stackdriver**, **Tools**, **Big Data**, **AI**, and many more. We will discuss in

detail about these services and products in the below sections. [Figure 14.4](#) is the GCP Dashboard:

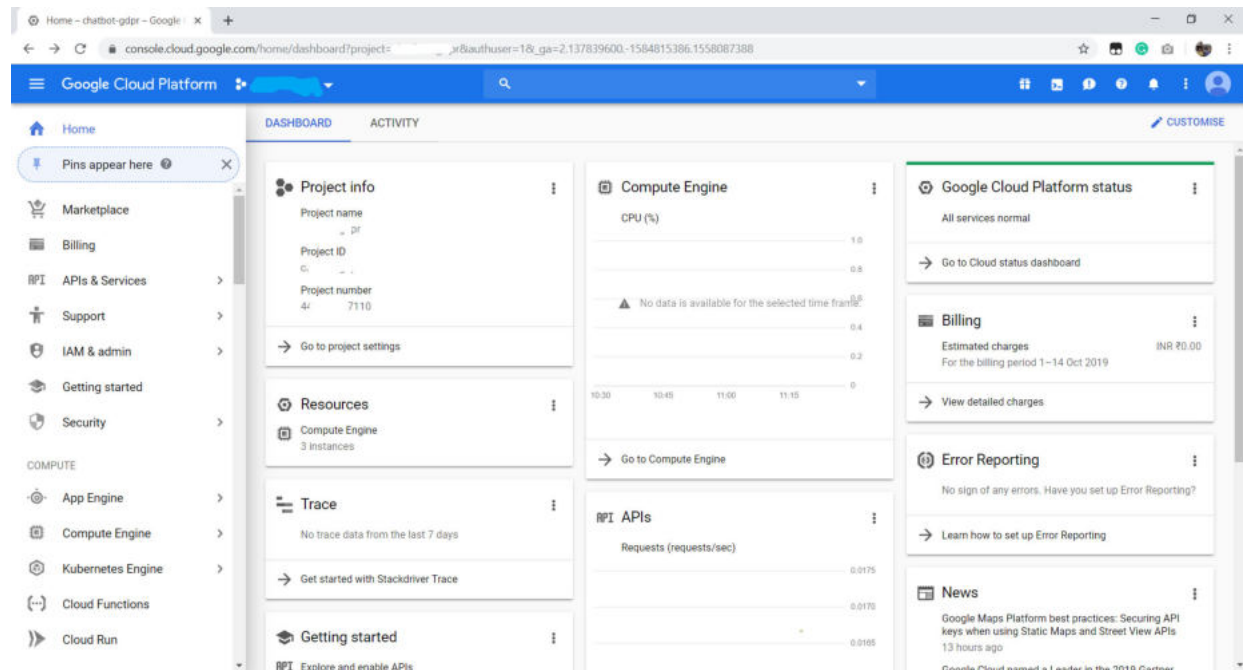


Figure 14.4: GCP Dashboard

The services and products provided by GCP can be accessed in the GCP Console. The App engine allows you to build scalable apps in any language. The Compute Engine shows the list of the VM instances. We will discuss about creating VMs in the next section. The Kubernetes Engine is a managed and production-ready environment for deploying containerized applications. GCP also has **Google Cloud Functions (GCF)**, which is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events, without the need to manage a server or a runtime environment.

Create VM and its properties

In the left navigation panel of the GCP Console, the compute services are listed. Among them, the Compute Engine service allows you to create VM, attach disks, and as well as create snapshots of disks. The other related services can be found in the left navigation panel of the Compute Engine page. The main page shows you the list of VMs created along with its location zone, internal IP, and external IP. Here, you can select any instance

and start, stop, reset, or delete the same. Now let's see how to create instances in the GCP console. [Figure 14.5](#) shows the list of VM instances.

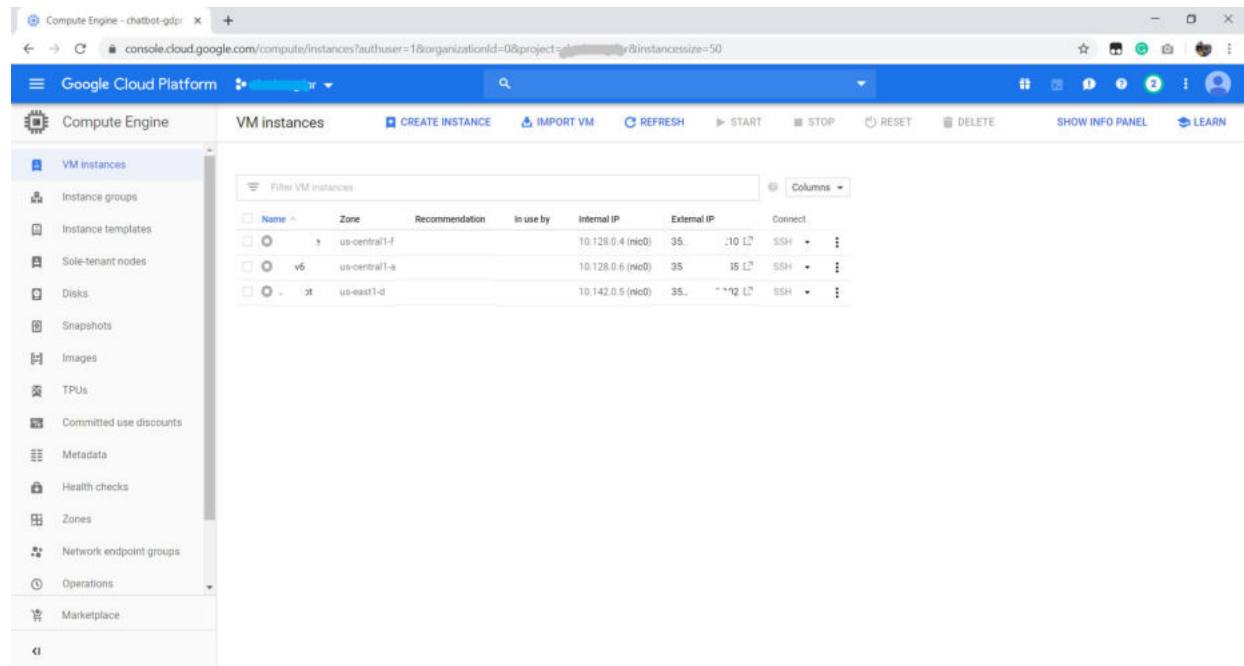


Figure 14.5: VM Instances home page

While clicking the **Create Instance** button in the top navigation bar, you will be directed to a page where you can create a new VM instance. You will need to choose a few important options according to your requirements and fill some of the required information. While following the instructions, you can also view the detailed billing estimate on the right side of the window. It provides you the monthly and hourly estimates in US dollars as per your machine configuration. By default, the region and zone will be us-central1 and us-central1-a. However, you can customize it according to your needs. A region is a specific geographical location where you can run your resources. A zone is an isolated location within a region. The zone determines what computing resources are available and where your data is stored and used. [Figure 14.6](#) shows the window of creating VM instance:

Create an instance

To create a VM instance, select one of the options:

- New VM instance**
Create a single VM instance from scratch
- New VM instance from template**
Create a single VM instance from an existing template
- Marketplace**
Deploy a ready-to-go solution onto a VM instance

Name
Instance-1

Region
us-central1 (Iowa)

Zone
us-central1-a

Machine configuration

Machine family
General-purpose | Memory-optimised
Machine types for common workloads, optimised for cost and flexibility

Generation
First
Powered by Skylake CPU platform or one of its predecessors

Machine type
n1-standard-1 (1 vCPU, 3.75 GB memory)

	vCPU	Memory
	1	3.75 GB

[CPU platform and GPU](#)

Container
☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk
New 10 GB standard persistent disk
Image: Debian GNU/Linux 9 (stretch) [Change](#)

Identity and API access

Service account
Compute Engine default service account

Access scopes
☒ Allow default access
☐ Allow full access to all Cloud APIs
☐ Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the internet.
☐ Allow HTTP traffic
☐ Allow HTTPS traffic

[Management, security, disks, networking, sole tenancy](#)

You will be billed for this instance. [Compute Engine pricing](#)

[Create](#) [Cancel](#)

[Equivalent REST or command line](#)

\$24.67 monthly estimate
That's about \$0.034 hourly
Pay for what you use: No upfront costs and per second billing
[Details](#)

Figure 14.6: Creating a VM instance

For a basic machine configuration, a single first-generation CPU with a 3.75 GB memory is enough. However, you can customize the number of CPUs and memory allocation accordingly. There is also an option to add a GPU in your current configuration. [Figure 14.7](#) shows the VM configurations:

Machine configuration ?

Machine family

General-purpose Memory-optimised

Machine types for common workloads, optimised for cost and flexibility

Generation

First

Powered by Skylake CPU platform or one of its predecessors

Machine type

n1-standard-1 (1 vCPU, 3.75 GB memory)


	vCPU	Memory
	1	3.75 GB

Figure 14.7: VM Configurations

The boot disk allows you to select an image of available operating systems with a minimum disk size of 10 GB. The default disk type is the standard persistent disk, where you can also choose an SSD persistent disk too. [Figure 14.8](#) shows how you can choose a boot disk:

Boot disk ?

 New 10 GB standard persistent disk

Image

Debian GNU/Linux 9 (stretch)

Change

Figure 14.8: Boot disk selection

We usually prefer to allow HTTP and HTTPS option of the firewall. With these instructions, you are now ready to create the VM instance. Once created, you can connect it with SSH. As of now, your new VM instance has an internal IP only. If an instance requires a fixed internal IP address that does not change, you can obtain a static internal IP address for that instance. You can reserve static address by browsing to external IP addresses from the

VPC network services in the left navigation panel of the GCP console. [Figure 14.9](#) shows the options for networking configuration:

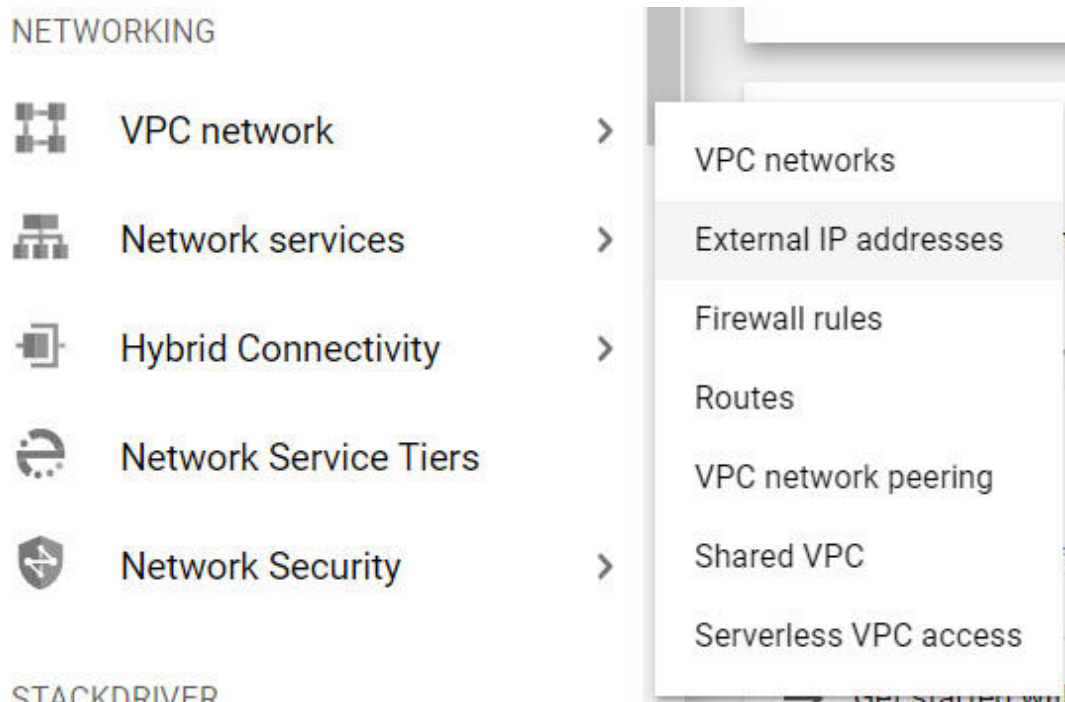


Figure 14.9: Network Configuration

As shown in [Figure 14.10](#), you will see your IP listed. While clicking the **static reserve address** in the top navigation panel, you will be directed to a page. Here you can choose the **Network Service Tier** as **Standard**, **IP version** as **IPv4**, and type as **Regional**:

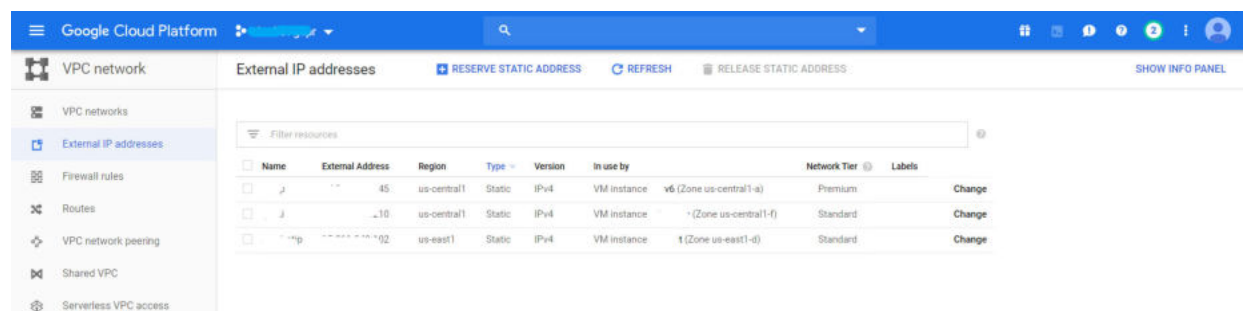


Figure 14.10: External IP Assignment

The region should preferably be the same one as you had chosen while creating the VM instance. You can refer to [Figure 14.11](#). In Attached to option, you should select the VM instance for which you are reserving the static address:

Google Cloud Platform

VPC network

VPC networks

External IP addresses

Firewall rules

Routes

VPC network peering

Shared VPC

Serverless VPC access

← Reserve a static address

Name ?

lowercase, no spaces

Description (Optional)

Network Service Tier ?

☒ Premium (current project-level tier, [change](#)) ?

☐ Standard ?

IP version

☒ IPv4

☐ IPv6

Type

☒ Regional

☐ Global (to be used with Global forwarding rules. [Learn more](#))

Region ?

us-central1 (Iowa)

Attached to ?

Some of the instances may be disabled due to the 'External IPs for VM instances' organisation policy. [Learn more](#)

None

⚠ Static IP addresses not attached to an instance or load balancer are billed at an hourly rate. [Pricing details.](#)

Reserve Cancel

Equivalent [REST](#) or [command line](#)

Figure 14.11: Static IP address assignment

[Connecting and uploading code to VM](#)

Once you created the VM in GCP, go to the VM instances dashboard, as shown in [Figure 14.12](#):

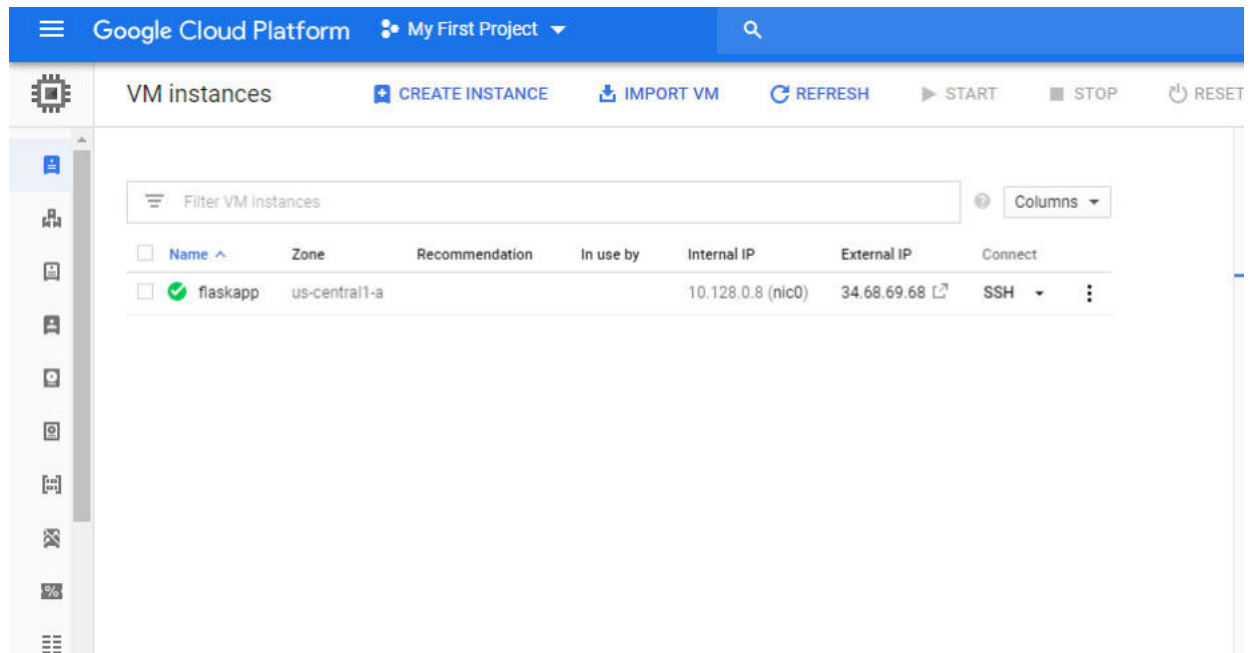


Figure 14.12: GCP VM instances dashboard

Select the VM instance created by clicking the name of the instances:

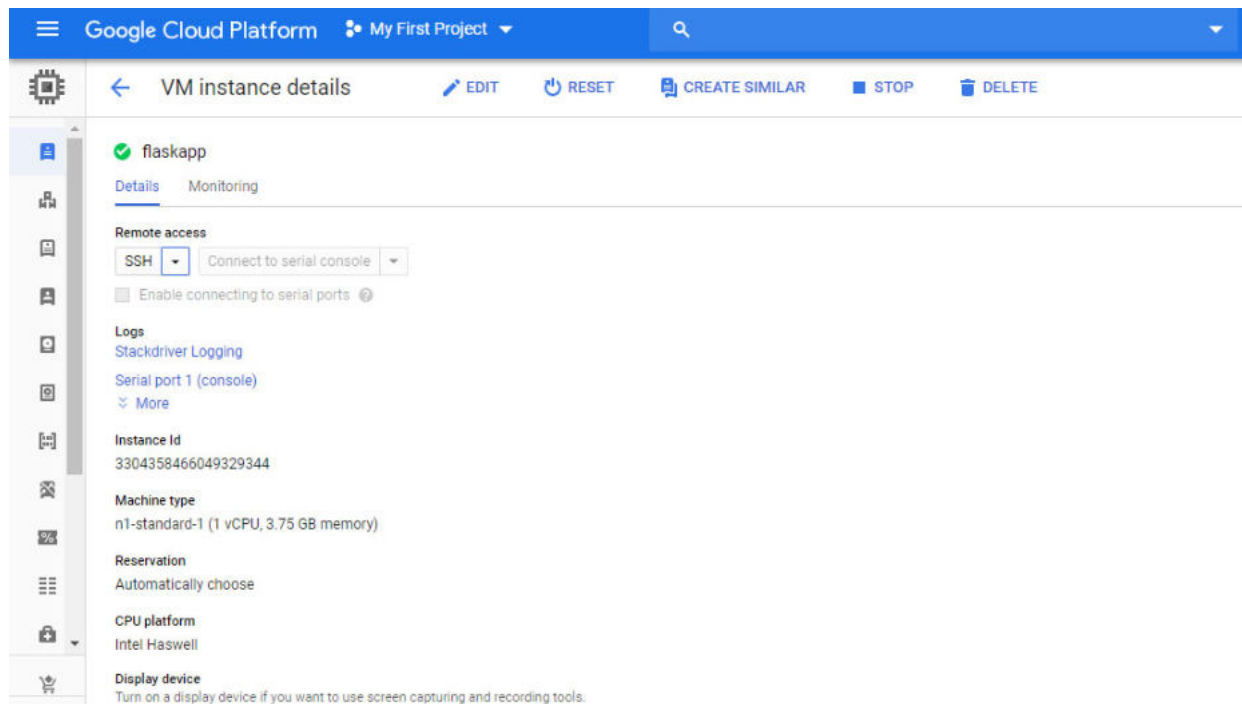


Figure 14.13: Edit the VM instance

Scroll down and go to the place where SSH key will be placed (Refer to [Figure 14.14](#)):

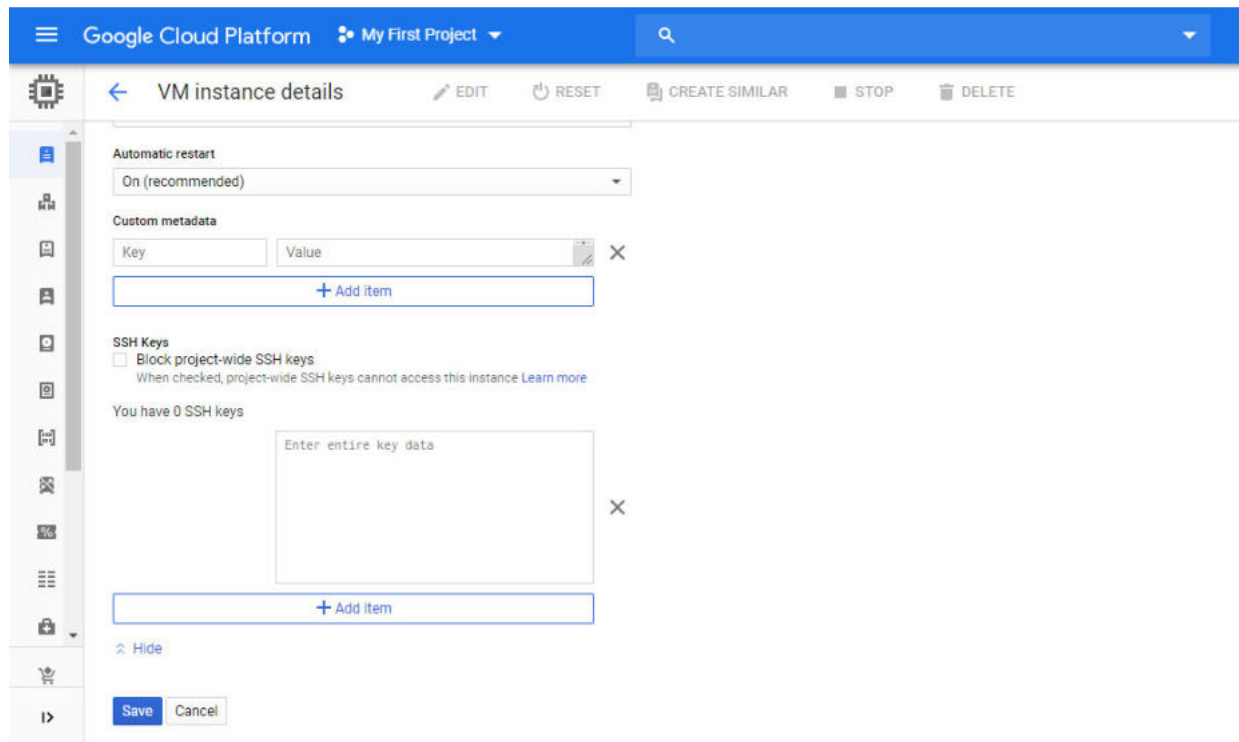


Figure 14.14: Add SSH key in VM instance

Here you can use your SSH key to connect the computer to the VM. If the SSH (Security Shell) is new to you and you may want to use it, it is a network protocol that allows encrypted data communication between two machines that are connected over the Internet.

To build an SSH connection, depending on your operating system, you may need software that can do so. I prefer and recommend PuTTY, as it is open-source and easy to use. Browse to <https://www.putty.org/> to download and install this application.

After installing PuTTY, open PuTTY Key Generator, and click create. It will generate a random key by you moving the mouse over the blank area. Once done, a screen similar to [Figure 14.15](#) can be seen. You can change the key comment field to something easy to type and recognizable, as this will be your username eventually. By clicking the corresponding icons shown in [Figure 14.15](#), you can save both the public and private keys. You then need to copy the key field for the PuTTY Key Generator and paste it in Google Cloud's key data field. [Figure 14.15](#) shows the SSH key generation in PuTTY:

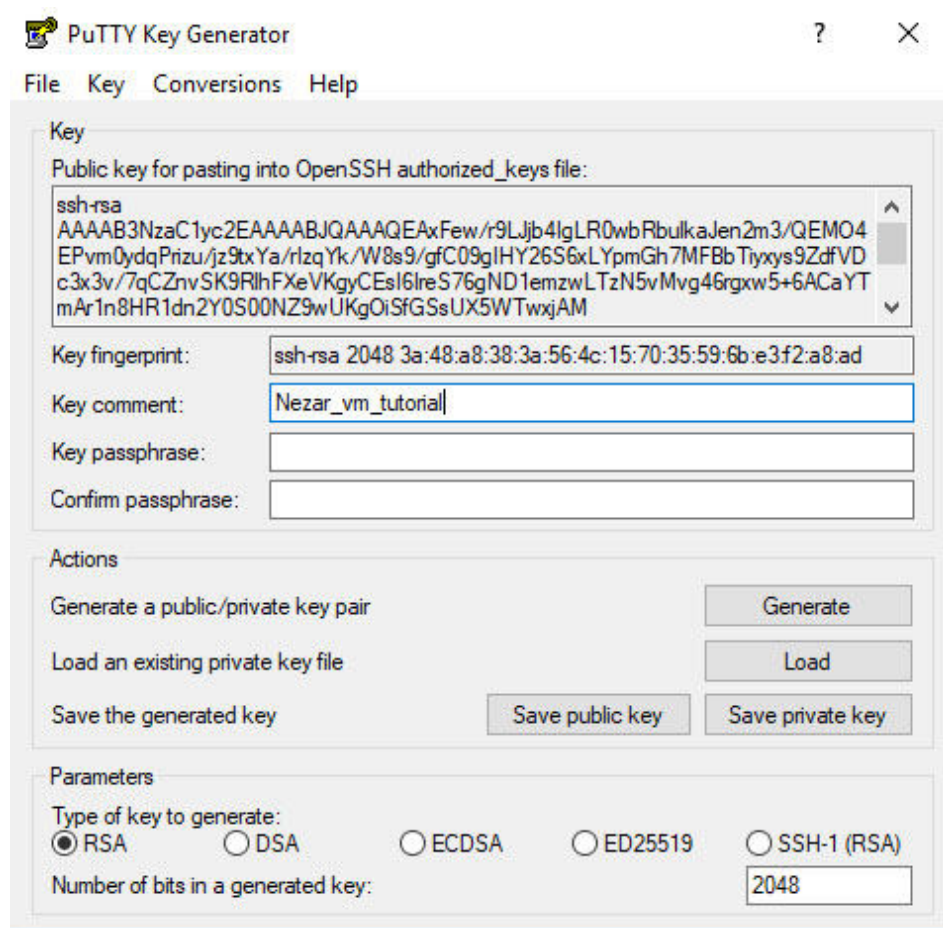


Figure 14.15: SSH key generation using Putty

The SSH key that has been created is then added in the VM instance, as shown in [Figure 14.16](#):

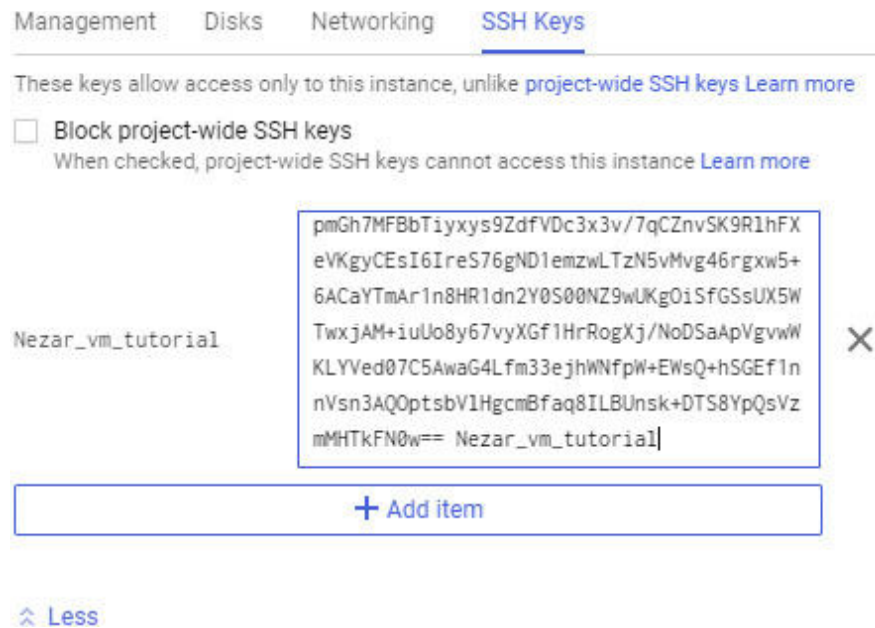


Figure 14.16: Add the SSH key in the VM instance

In the meantime, you can go to PuTTY. Go to **SSH|Auth** and browse for the private key file that you saved. [Figure 14.17](#) shows **Auth** page in the PuTTY, where you can add the private key:

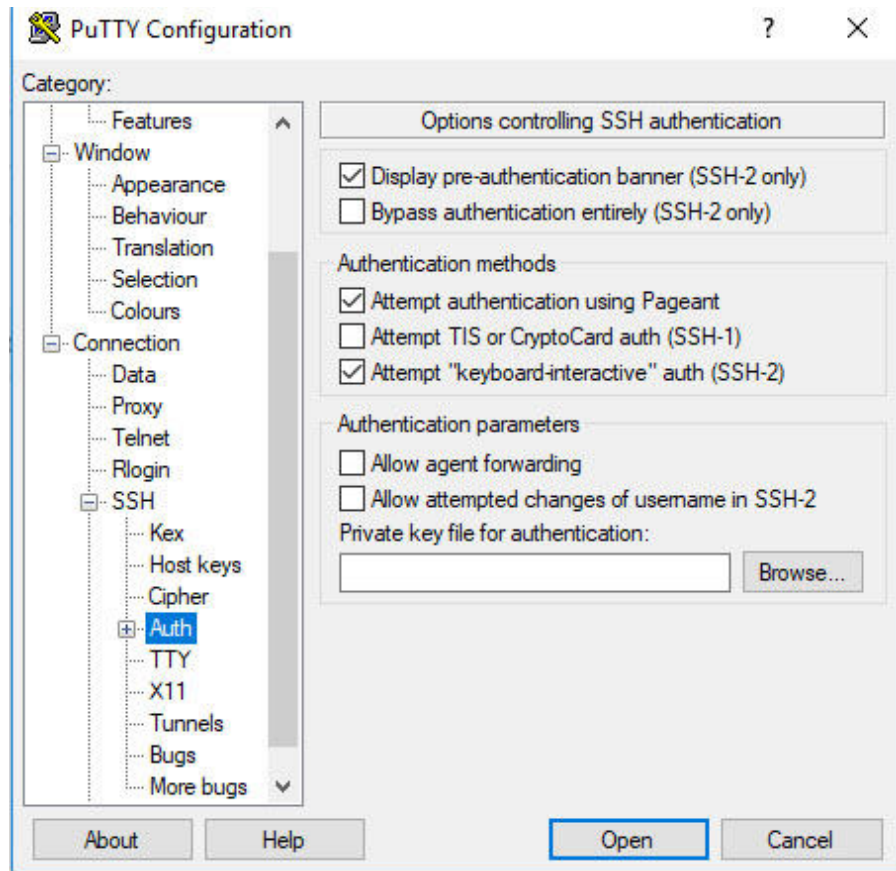


Figure 14.17: Auth Private key in Putty

Next, go to Google Cloud and copy the external IP from the VM instance that you just created, as shown in [Figure 14.18](#):

Zone	Recommendation	Internal IP	External IP	Connect
us-east1-b		10.142.0.2	35.196.10.110	SSH ▾ ⋮

Figure 14.18: Access the VM External IP Address

And paste it on the **Host** field under **Session** in PuTTY and hit Enter:

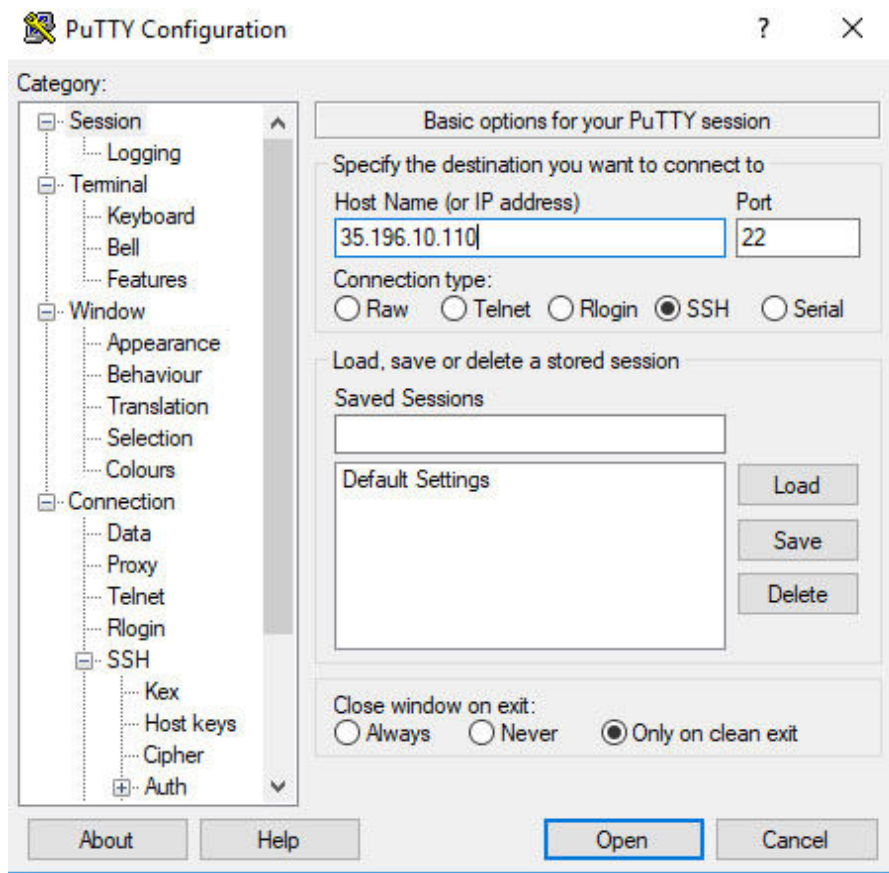


Figure 14.19: Connect with Putty using VM External IP

You might see an error message, to which you can ignore it and click **Yes**. Go ahead and enter your username, which was created while generating the key. Now you can connect with the VM.

You connect to the server using SSH from GCP. [Figure 14.20](#) shows the **Compute Engine** in the GCP window:

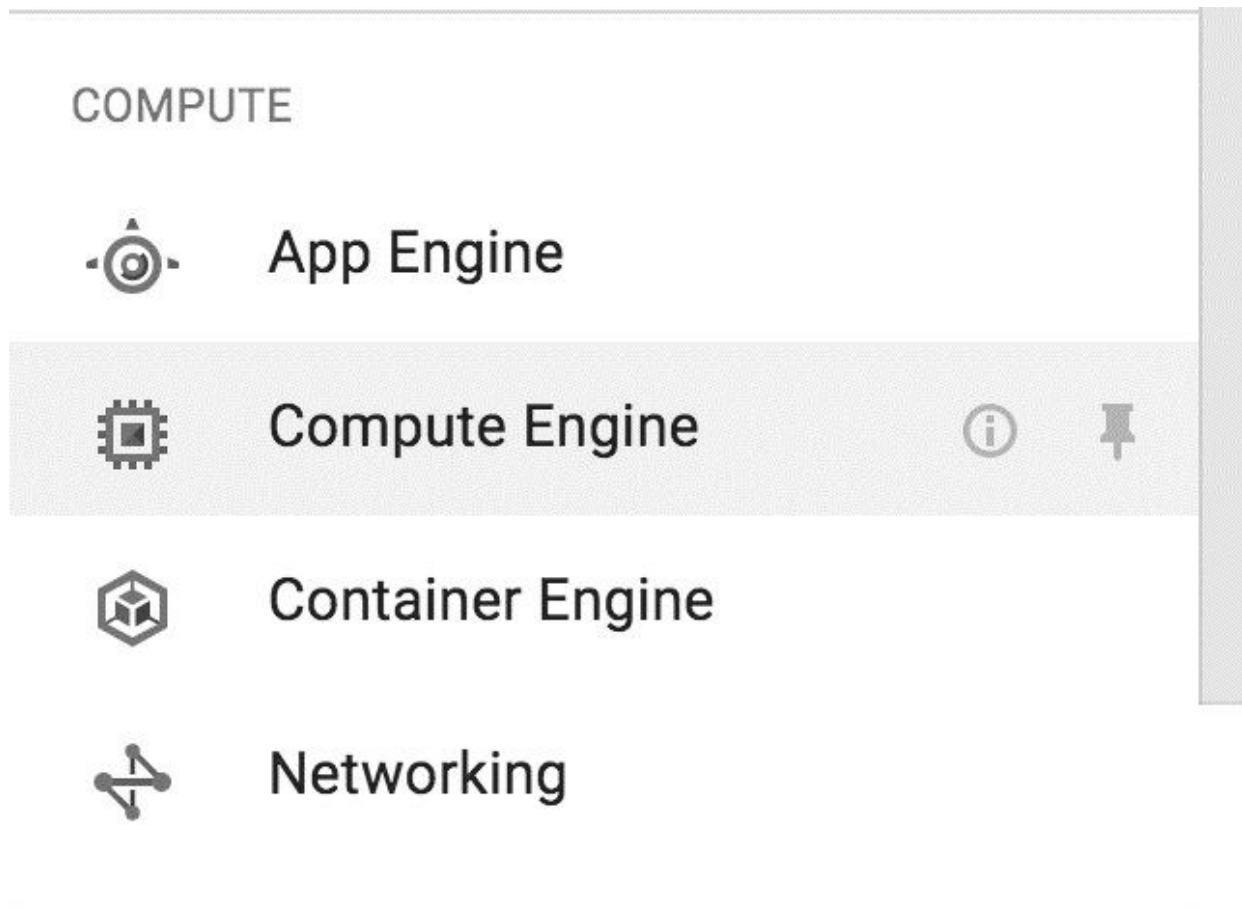


Figure 14.20: Computer Engine Selection in GCP

[Figure 14.21](#) shows the snippet window, where you can see the SSH access:

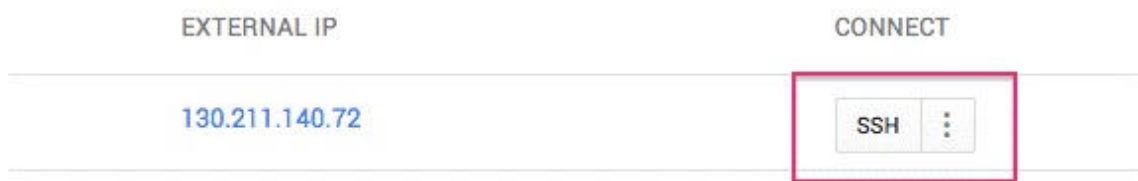


Figure 14.21: Access SSH to upload file

Once you SSH, you will be a similar window, as shown in [Figure 14.22](#). The gear button on the top-right provides access to various settings, including file upload option:

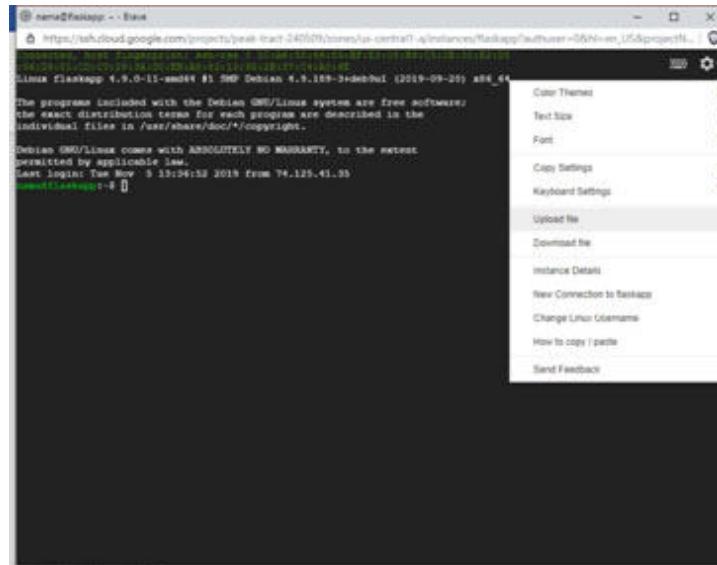


Figure 14.22: Upload the files to the VM instance

Now let's create the following files in the local system, as shown in [Figure 14.23](#):

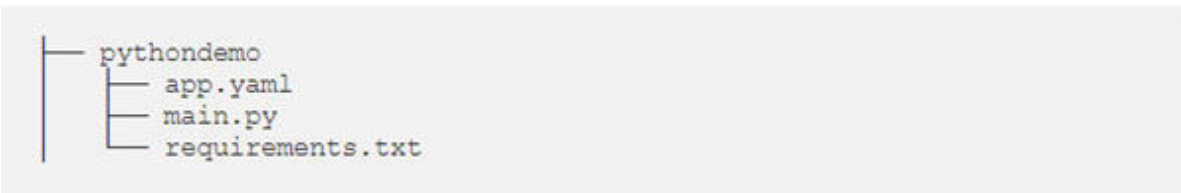


Figure 14.23: Folder structure

You can configure your App Engine app's settings in the `app.yaml` file. The `app.yaml` file contains information about your app's code, Python runtime, entrypoint, and environment variables. [Figure 14.24](#) shows a snippet of the code:

```

runtime: python
env: flex
entrypoint: gunicorn -b :$PORT main:app
runtime_config:
  python_version: 3
  
```

Figure 14.24: Code for `app.yaml` file

The `main.py` is our Python file, which lives on the cloud. Now let us deploy the salary prediction model file. The code will look similar to this:

```

#Import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the datasets
data = pd.read_csv('Salary_Data.csv')

#Load the values on variables in a array
X = data[['YearsExperience']].values
Y = data[['Salary']].values

# Splitting the into the Training set and Test set
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y,
test_size = 0.25, random_state = 0)

# Fitting Simple Linear Regression to the training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)

#Store the model as a pickle object
import pickle
pickle.dump(regressor,open( "linear_reg_salary_model.p", "wb" ))
#Load the Libraries
from flask import Flask,request,jsonify

import pickle
import json
import pandas as pd

#Start a flask app
app = Flask(__name__)

# Load the model
regressor = pickle.load(open( "linear_reg_salary_model.p", "rb"
))

@app.route('/predict', methods=['POST'])
def predict():
    #Retrieve the value of 'YearsofExperince' from the request body

```

```

data = request.get_json()
df = pd.DataFrame([float(data['YearsExperience'])], columns=
['content'])
predict_new = regressor.predict(df)
result = {'predicted_salary': predict_new.tolist()[0]}
return json.dumps((result))

if __name__ == '__main__':
    app.run(port=3000, debug=True)

#import the request library which allows us to make Post/Get
etc. web request
import requests

#Define the address of the host where the application is running
URL = 'http://127.0.0.1:3000/predict'
payload = { "YearsExperience": 3.2 }
res = requests.post(url,json = payload)
The requirements.txt contain Python modules that needed for
main.py:
Sample
Flask==0.12.3
gunicorn==19.6.0

```

Executing Python model on cloud

Go to GCP Console Home, click on Cloud Shell icon. It will open-shell CMD prompt on the bottom screen. [Figure 14.25](#) shows the option to access the GCP Cloud shell:

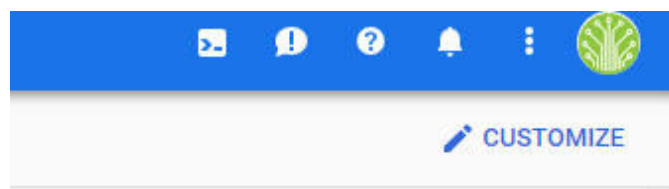


Figure 14.25: Access the GCP cloud shell

Click on the little *pen* icon on the top right of the shell. it will take you to cloud shell UI site. [Figure 14.26](#) is a snippet of accessing the Cloud Shell UI:

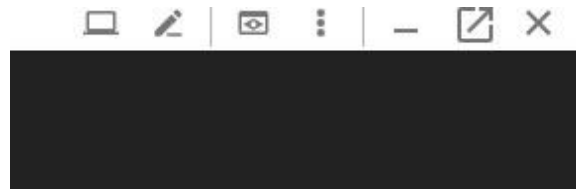


Figure 14.26: Access the Cloud Shell UI

In your folder directory of VM, run following command. This may take some time to deploy your app. [Figure 14.27](#) shows an instant while deploying the app in the Cloud Shell:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to i
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
dabbakuti_salman@cloudshell:~ (1.0.0) $ ls
pythondemo README-cloudshell.txt
dabbakuti_salman@cloudshell:~ (1.0.0) $ cd pythondemo
dabbakuti_salman@cloudshell:~/pythondemo (1.0.0) $ ls
app.yaml main.py requirements.txt
dabbakuti_salman@cloudshell:~/pythondemo (1.0.0) $ gcloud app deploy
```

Figure 14.27: Deploy the App in Cloud Shell

Once deployed, run below command and it will show deployed address and paste this address in your browser. This will show deployed python WebApp response:

```
gcloud app browse
```

[Access the model via browser](#)

Create the HTML file using the below code. Below, you can see a sample HTML page for our salary prediction application:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Salary Prediction ML Model</title>
```



```

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css" integrity="sha384-
gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQU0hcWr7x9JvoRxT2MZw1T
" crossorigin="anonymous">
</head>
<body style="width:60%; margin-left:20%;">
<h1 class="text-center">Salary Prediction ML Model</h1>
<form action="/predict" method="post">
<input class="form-control" type="number" name="YearsExperience"
step="any" min = "0" placeholder="Number of years of
Experience">
<input class="btn btn-primary" type="submit" value="Predict
Salary">
</form>
    {% if salary %}
<p>The Expected Salary is: {{salary}}</p>
    {% endif %}
</body>
</html>

```

Provide the external address of the VM with route /predict.

A webpage can now be accessed at **http://external_ip_VM:3000/predict** and can be used by the end-user to input his experience, and in return, it will show his expected salary. Below is a snapshot of the web page. [Figure 14.28](#) shows the output in a web browser:

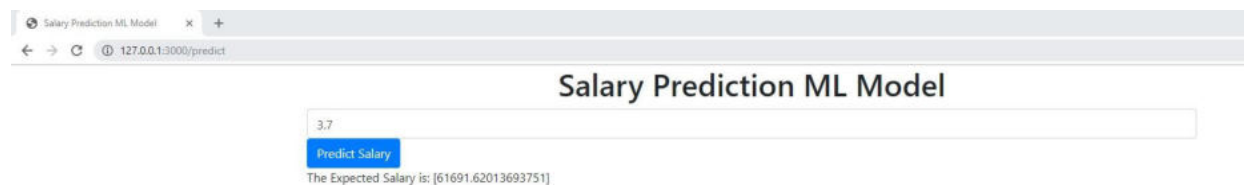


Figure 14.28: View the output in Web Browser

[Scaling the resources in Cloud](#)

Instance groups in Google Cloud are either managed or unregulated. For Compute Engine, we need managed instance groups that have common

features and can scale according to certain circumstances. We have already taken care of the instance template on which the instance groups depends on. Now let's set the instance group. [Figure 14.29](#) is the window snippet of creating an instance group in GCP:

← Create a new instance group

Use an instance group when configuring a load-balancing backend service or to group VM instances. [Learn more](#)

Name ⓘ
autoscaling-instance-group-1

Description (Optional)

Location
Multi-zone groups span multiple zones which assures higher availability [Learn more](#)

☒ Single-zone
☐ Multi-zone

Zone ⓘ
us-east1-b

[Specify port name mapping](#) (Optional)

Group type
☒ **Managed instance group**
Managed instance group contains identical instances, created from an instance template, and supports autoscaling, autohealing, rolling updating, load balancing and more. VM instances are stateless and disks are deleted on VM deletion or recreation. [Learn more](#)

☐ **Unmanaged instance group**
Unmanaged instance group is best for load balancing dissimilar instances, which you can add and remove arbitrarily. Autoscaling, autohealing, and rolling updating are not supported. [Learn more](#)

Figure 14.29: Instance Group Creation

Scaling out relies primarily on load balancing, and this requires one. Now let's configure one. **Loadbalancer** from Google compute is way advanced in contrast to the standard **Loadbalancer** of AWS ELB or Azure. With respect to this content, we will proceed with Network **Loadbalancer**. [Figure 14.30](#) shows the basic setup window of creating Network Load Balancing:

←

New HTTP(S) load balancer

Name ⓘ

geekflarelab

✓

Backend configuration

You have configured 1 backend(s)

✓

Host and path rules

You have created host and path rules

✓

Frontend configuration

Your frontend is configured

ⓘ

Review and finalize

Optional

→

Create

Cancel

Review and finalize

Backend

Backend services

1. geekflarelab

Endpoint protocol: HTTP Named port: http Timeout: 30 seconds Cloud CDN: disabled Health check: geekflarelab

Advanced configurations

Instance group	Zone	Autoscaling	Balancing mode	Capacity
server-uk	europe-west2-c	Off	Max CPU: 80%	100%
server-us	us-central1-c	Off	Max CPU: 80%	100%

Host and path rules

Hosts

Paths

Backend

All unmatched (default)

All unmatched (default)

geekflarelab

Frontend

Protocol

IP:Port

Network Tier ⓘ

HTTP

35.190.65.140:80

Premium

Figure 14.30: Creating Load Balancer

[Figure 14.31](#) shows the window for creating a target pool for load balancing. Target pool distributes traffic among the VM instances:

← Create a target pool

Target pools distribute traffic among their set of healthy VM instances. [Learn more](#)

Name ?

Name is permanent

gd-app-pool

Description (Optional)

Region ?

us-central1

Health check ?

No health check

Session affinity ?

None

[Select existing instances](#)

[Select existing instance groups](#)

VM Instances ?

No available instances

Backup pool ? (Optional)

No target pools available in region us-central1

Failover ratio ?

10

%

Create

Cancel

Figure 14.31: Target Pool creation for load balancing

That's all about it. Your application will now scale up and down depending upon how you configured it.

Conclusion

In this chapter, we have learned how to create an account in GCP. We have also learned the fundamentals of creating VM in GCP and its properties.

Further, we have learned how to deploy a flask application in GCP with the salary prediction example. Finally, scaling the resources in GCP using a load balancer has been explained. From this overall procedure to use a cloud computing platform to implement the machine learning model has been successfully done. In the next chapter, we will give an overview of business intelligence and the key steps in building a business intelligence practice.

CHAPTER 15

Introduction to Business Intelligence

Communication in data science is very important. So far, in our previous chapters, we paid very little attention to the communication of data insights to business users and decision-makers. Data science professionals, specifically those in the role of Analyst, need to be good at understanding data, apply basic analysis, and read the model outputs in the business context.

The business intelligence domain specifically bridges this gap between business, data scientist, and technology, by providing data insights in easy to read and interpret manner. The core practitioners who engage in the role of BI are called *data analyst* or *BI analyst*, and the tools used for BI are typically called visualization tools.

In this chapter, we will give an overview of business intelligence and the key steps in building a business intelligence practice. The next chapter will be fully hands-on on one of the very popular BI Tools, Microsoft Power BI.

Structure

- What is business intelligence?
- Business intelligence analysis
- Business intelligence process
- Business intelligence trends
- Gartner 2019 Magic Quadrant

Objectives

After studying this unit, you should be able to:

- Understand the vast area of business intelligence
- Explore the process of business intelligence

- Get updated with business intelligence trends

What is business intelligence?

Business Intelligence (BI) is an amalgamation of multiple parts of the data universe as it combines visualization, data tools, data mining, and analytics. The system thus provides a unified view of the enterprise data and allows the business to analyze it to drive and communicate insights. A successful BI system in place would mean that any user in the organization can have a 360-degree view of the enterprise's data and make use of that data to make data-driven decisions.

Business Intelligence started as a mechanism to share information across different business verticals to break silos of decision making. Later, with the help of database and improved IT process evolved into micro decision-making systems inside business verticals. Cloud and Big Data have now enabled the BI to take its modern view, which now connects to the enterprise data lake and can provide a complete view of an organization using Visualizations tools with few clicks.

Modern BI tools and data lake technology has now matured to the point that it has become commoditized and available for small and medium companies as well. BI provides flexible self-service, data governance, privacy, accessibility, user-friendly UI, and speed to draw insights and share them across the organization. [Figure 15.1](#) shows a conceptual flow diagram of BI:

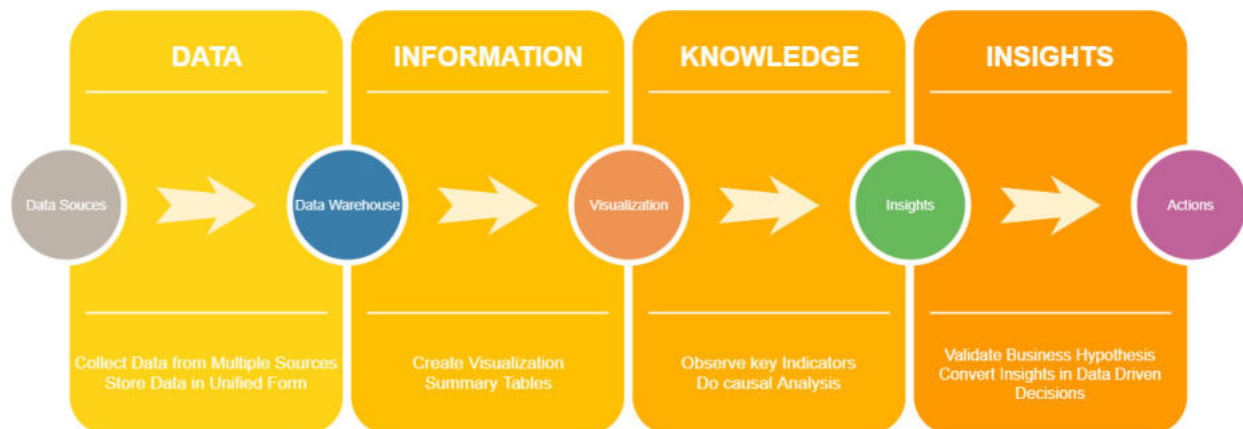


Figure 15.1: BI Conceptual Flow

BI is sometimes seen as a prerequisite for organizations to adopt advanced ML and AI into the organization's decision making. In [Figure 15.1](#), you can

see how the BI conceptual flow brings a new stream of a data-driven approach to decision making. The BI system helps business to discover trends and causation to key business metrics. Businesses can formulate a hypothesis to solve the problems with advanced ML and AI techniques.

The BI conceptual flow can be realized into DIKI framework:

- Data
- Information
- Knowledge
- Insights

And thus, insights lead to decision making based on the data facts rather than intuition. The DIKI framework will impact the organization's people as well as performance and decision making. The framework influences the cultural shift in the organization for taking day to day decisions.

BI adoption is happening at a rapid pace and has created new job roles in companies. A BI analyst is not replacing a **Business Analyst(BA)** role, the key addition to BA role is now extensive use of BI techniques and tools to make decisions. The data scientist also prefers communicating their model results in easy to read and interpret data visualization dashboards. There are many areas where BI is helping business, including:

- Customer behavior analysis
- Sales analysis and buying behavior
- Demographic and customer segmentation
- Financial performance
- Tracking and monitoring marketing campaigns
- Supply chain analysis
- Risk analysis and scenario analysis
- Strategic value driver analysis
- Web analytics
- ML/AI model performance analysis

And many more use cases developed by either self-service or by BI analyst.

Business intelligence analysis

Business intelligence analysis is an evolved way of doing business analysis using empirical evidence. With the proliferation of data and analysis tools, it has been proved that organizations of all sizes benefit from adopting business intelligence best practices and encourage their employee to become data-driven.

While all organizations get benefited from BI, there are some signs when it becomes an absolute necessity for organizations to adopt BI. Few of those signs are listed below:

- We need to integrate data across applications and business verticals.
- Reduced visibility into companies' different departments, that is, Finance, HR, Marketing, operations, and others.
- Delayed decision making due to data gathering.
- End-User requiring analytical capabilities in the system.
- Lack of real-time monitoring of business processes.
- IT environment upgrades to facilitate data-driven decision making.

The list above is a clear sign to adopt BI analysis to empower the business to make decisions driven by data. There are multiple benefits an organization gets from the BI system in place, including:

- Faster analysis of business pain points
- Alignment of strategy to operations by means of tracking dashboards
- 360-degree view of an organization
- Boost in internal productivity by focusing on analysis

While so far, we have been discussing the benefits a BI system brings for the organization, and it's important to understand how that is being made possible by such a system in place.

Tableau is one of the leading BI tools in the market, helping large enterprises to adopt BI technology and create a data-driven culture. The learning series of Tableau lists down 9 ways in which BI helps a business achieve its full potential by using data:

Data mining	Using databases, statistics, and machine learning to uncover trends in large datasets.
--------------------	----------------------------------------------------------------------------------------

Reporting	Sharing data analysis to stakeholders so they can draw conclusions and make decisions.
Performance metrics and benchmarking	Comparing current performance data to historical data to track performance against goals, typically using customized dashboards.
Descriptive analytics	Using preliminary data analysis to find out what happened.
Querying	Asking the data specific questions, BI pulling the answers from the datasets.
Statistical analysis	Taking the results from descriptive analytics and further exploring the data using statistics such as how this trend happened and why.
Data visualization	Turning data analysis into visual representations such as charts, graphs, and histograms to more easily consume data
Visual analysis	Exploring data through visual storytelling to communicate insights on the fly and stay in the flow of analysis.
Data preparation	Compiling multiple data sources, identifying the dimensions and measurements, preparing it for data analysis.

Table 15.1: Tableau's list of BI helping business

Business intelligence process

BI success is critical to data-driven decision making for companies of all sizes. The BI integration to core business use must go through rigorous self-awareness of data and needs. Like garbage in garbage out model of data science models, the same is applicable to BI systems. The BI system generates long term gains if implemented properly with control over a full stack of BI.

The success road map of BI process is described in [Figure 15.2](#):



Figure 15.2: Business intelligence process

Every BI platform needs to be built with the above steps being an essential part of the process. In the last section, we will list down some tools and stacks that make the adoption easier by SaaS and PaaS model for BI intelligence.

There are two ways to look at developing a BI system for business;

- **Business needs are known:** Business defines the needs and expectations of the BI system. The IT, analyst, and database team, work to build those using the best available tools.
- **Business needs are not-known:** The enterprise creates a platform by bringing all enterprise data into one place and enable general users to derive insights by self-service capabilities of tools

The former process works like a typical software development cycle, while later is more of a long-term success platform, liberating end-user from SDLC. Also, in the first case, we lose the agility to pivot the analysis for different purposes. The following sections explain how a BI system is developed for an enterprise, keeping future use cases unknown.

Step 1: Data awareness

Data awareness is the first step in any analysis related to BI. The business can define the data it collects, for what purpose, from what system, and other known factors into its architecture. This data awareness allows the business to understand what it captured and create a meta-store of data for business users to know what is available for analysis. Two key considerations in the data collection stage are; data types and data sources.

Data types

The data types refer to what type of data is flowing through the enterprise. The data types can be generally divided into three buckets:

- **Unstructured data:** Data has no inherent structure and usually stored in a file system. For example, text, PDFs, images, and many more.
- **Semi-structured:** Textual data which can be structured by tools and knowledge of patterns in file. For example, XML, JSON, and more.
- **Structured data:** Data having a defined structure and stored in a pre-defined format. For example, SQL database, CSVs, and more.

The data for analysis may be in any format and type. The BI system needs to be able to define a method, to access the data of different types, subject to business needs.

Data sources

Another flexibility the BI system needs to have is the ability to read data from multiple sources. The data can be coming from multiple types of systems, e.g., flat files, APIs, No-SQL databases, SQL databases, stream data, and other legacy systems. Different parts of the business operate at different maturity of IT in their organization. The diversity in data sources is a challenge, and bringing all this together is the most difficult task in the BI system set-up.

Step 2: Store data

After developing the understanding of data types and sources across the enterprise, it needs to be stored in a system and format that is readily available for analytical purposes. Data storage also requires having data availability for all BI users, controlled by privacy and confidential mechanisms; for example, you don't want the marketing team to see credit card numbers of customers.

Data models

Data models are the structure in which data is stored logically for easier access to desired business purpose. Here, we need to differentiate between how the data was stored in source systems (for application logic purpose) with how we want it to be stored in the analytical database (for BI purpose).

The key considerations include:

- We do not make a simple copy of data from operational to an analytical database; we re-model the data for business needs. (For example, customer profile, product profile, and more.)
- The data is stored in appropriate Normalized form (Modern BI tools capable of faster joins)

Data storage

Data storage is another significant factor to enable on the fly analysis with high agility. In the early days, schema defined data warehouses and data marts were the popular choices for Data storage as the business data growth was slow and predictable. The data warehouse is SQL databases with structured data. In recent years, the data has outgrown both in size and type, hence more powerful data storage in terms of Big Datastore is more appropriate to use. The Schema is also not pre-defined, and data is stored in RAW format; these systems are called Data Lake.

The key considerations include:

- Operational databases to analytics databases
- Apply appropriate transformations and load to a BI tool
- Automate the process of loading (ETL for Data warehouse and ELT for Data Lake)

The process of data loading into storage and keeping it updated with operational databases needs to be an automated process so that the BI always reports near real-time status of actual.

Step 3: Business needs

Data itself cannot provide decisions to the business. Business needs to define what are the basic metrics and process it wants to monitor. Business initiates the strategic goals, and they require decisions to be made to realize those goals. The data help make empirical decisions by using the BI system.

The business requirements also set the basic expectations from the visualization tools and the training requirement of employees to adopt the BI systems. There are two key considerations in capturing business needs; the **Key Performance Indicators (KPIs)** and Data Visuals/Dashboards.

Key Performance Indicators (KPI)

The business performance is monitoring by some selective measures, and these measures are called KPIs. For business, it is neither necessary nor feasible to track too many data points, and they lead to decision paralysis. Some key performance metrics must be defined and then tracked by looking at real data using the BI system. A good selection of KPIs is needed:

- To track company health.

- To measure progress.
- To make adjustments and stay on track.
- To solve problems or tackle opportunities.
- To analyze patterns over time.

Data Visuals

While KPIs will tell us, what needs to be presented by easy to read and impactful manner. The data visuals are the end-user consumable outcome of your BI system. They need to be interactive, clean, sharable, and impactful. One must consider the following points while creating data visuals;

- Plots are just representations of data; they do not give actionable outputs.
- Business-specific KPIs need to be built to quantify in plots.
- Reports are periodic outputs from your BI system to help you make decisions.

Step 4: a Visualization tool

The selection of the right toolset for generating data insights and visualization is an important decision. While all the previous steps and the following steps can be independently implemented using the technology stack as per business needs, the visualization tool sits in between the data and end-user. With plenty of visualization tools available in the market, it is important for us to categorize them for a better understanding of the BI landscape and their capabilities.

The two factors that are very much relevant to the visualization tool adoption are:

- **Time to insight:**How much time it takes to create insights from data using the tool?
- **Ease of use:** How easy is it to use the toolset to derive insights?

The two dimensions cover the complexity to build the solution using these tools and user experience of doing the same. [Figure 15.3](#) compares the powerful BI with two factors, ease of use and time to insight:

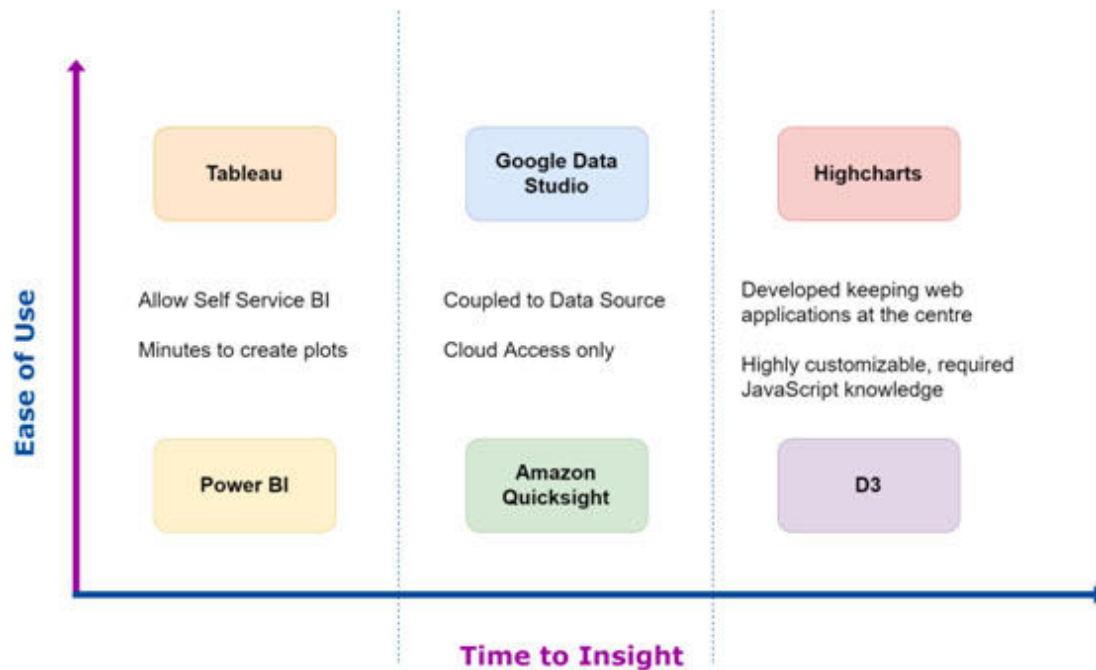


Figure 15.3: BI tool landscape

Time to insight

In a fast-paced business environment, the process of driving insights from raw data must be fast. The dimension of time is important to consider, not just as the development but also time to build a new visual/analysis in the BI system. Business analysis requires a complex drill-down of questions across multiple dimensions; the tool needs to be fast enough to adopt those.

Tools like Power BI and Tableau allows you to get started with data metrics and visuals in no time and hence allows business users to derive insights faster.

The shortcoming of these tools is that they do not allow much customization, less developer-friendly, more user-friendly, and are costly for small businesses.

Ease of use

While a robust dashboard can be built for business needs from scratch using native technology of JavaScript, they remain very developer-friendly. In reality, the user BI system is usually Business Analyst and management executives. They need an easy way to play with data in the visualization

tool. Here as well, Power BI type of tools provides that kind of user interface and interactive features.

Making things easy and exposing with so many options to end-user many times confuses them and, due to excess options, creates a problem in decision making.

Step 5: Enable platform

The end goal of any BI system is to enable a platform that can be used by business users removing the data silos and allow cross-functional analysis. This results in synergy across functions and better decision making at the enterprise level.

The platform creates a conducive environment to facilitate data share among different business users, sharing insights among different business users, present reports, and persist analysis. The two key enablers for BI systems are; data access and business users.

Data access

Data access here is referred to as the ability of the BI system to provide protected access for cross-functional teams for their data. The data storage for the BI system automatically brings data into the common storage and allows users to access the data they require to complete the analysis.

BI system breaks the silos and enables the platform to access any data from authorized users. This leads to the democratization of data inside the enterprise.

Business users

There are multiple stakeholders in any BI system, including IT teams, CXOs, Business Analyst, reporting staff, and many more. The platform should be enabled to the users as per their needs and mandate of their job functions. For example, a CXO might just be a consumer for the dashboard view, while a BI analyst needs to create the dashboard, and both are served by the same platform.

Business intelligence trends

Tableau conducts an annual survey among its huge customer base and open market BI users to understand what is the current status of the BI tools and what advanced users are looking for in the coming years in BI systems. The study emphasizes the growing trend of the BI system is now an integral part of the businesses. There are now dedicated roles of visualizationanalystsand BI analysts.

[Table 15.2](#) shows the findings from the survey of Tableau for 2019:

Trend	Outlook
The rise of explainable AI	Nowadays, the organizations rely more on Artificial Intelligence and its derivative application. This shows the definite growth in the business trends, and the importance of trust in data also comes into the flow.
Natural language humanizes your data	Understanding the people based on their conversation data will be more effective by the use of advanced techniques in NLP systems.
Actionable analytics put data in the context	Evolution in the BI platforms supports the people to visualize the data and take actions based on that.
Data collaborative amplify the social good impact	More private-sectors and public-sector based organizations take efforts on the data to create a good impact on community development.
Codes of ethics catch up to data	Considering regulations like GDPR, leaders judge the future of ethical data practices.
Data management converges with modern BI platforms	The curation of data in a well-governed manner can bridge the gap between business and data.
Data storytelling is the new language of corporations	Visualizing and finding the insights from the data is an art. And communicating those insights in the right way is now like a team sport.
Enterprises get smarter about analytics adoption	Organization leaders have the thought to go more about the adaptation of analytics in their business.
Data democracy elevates the data scientist	Soft skills are essential in business intelligence. So, get to know about the skill is an important addition to the data scientists.
Accelerated cloud data migration fuels modern BI adoption	Putting the data in cloud is a common task in data analysis. This helps the organizations to think moreabout their business based on the strategy that comes through the data.

Table 15.2: Survey of Tableau for 2019

[Gartner 2019 Magic Quadrant](#)

The BI initiative in organizations is now lead by BI tools. The IT teams work with the BI tool vendors to establish the data storage, warehouse, or lakes, and many provide the data layer for faster adoption of their respective BI tools. It becomes important for the reader of the book to be aware of the tools in the market for BI, and they can learn those tools to master data visualization and may prepare themselves for a career in visualization analyst or BI analyst.

Gartner is a credible and globally recognized publishing house for technology trends. They annually publish their Magic Quadrants for Analytics and BI platforms. [Figure 15.4](#) shows the magic quadrant for Analytics and BI platforms:

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms



Source: Gartner (February 2019)

Figure 15.4: Gartner Report Snapshot (Credit:

<https://www.gartner.com/en/documents/3900992/magic-quadrant-for-analytics-and-business-intelligence-p>)

The magic quadrants distribute the BI players into four quadrants;

- **Leaders:** The leading players in the market, to large extent, market makers in the BI domain. For example, Power Bi, Tableau, Qlik, and ThoughtSpot.
- **Visionaries:** The players in the market have a clear vision of their product offering and working on improving the execution—for

example, TIBCO, Salesforce, and more.

- **Nice Players:** The players who are very focused on particular domains and facilitating analysis. For example, Domo, GoodData, and more.
- **Challengers:** The players who are very good at execution and building a clear vision for their offering. For example, Microstrategy.

Tableau and Power BI are the clear winners in the race for large deployments in big corporates, and hence creating a lot of opportunities for job and skill development at the university level as well. In the next chapter, we will show hands-on working in Power BI with real data from our data.gov.in example.

Conclusion

The chapter focused on the introduction of the vast area of Business Intelligence. The BI system allows a business to break silos and build a cross-functional analysis of data. The BI process includes building data awareness and data storage in the appropriate models. The process requires laying down business goals and metrics that they want to track for better decision making; these are called KPIs. The visualization tool then works as a bridge between the end-user and your data to provide a platform for business analysis and drawing insights. The tools are plenty in the market and require careful selection of the business requirements. The end-goal of the BI system is to enable users to use enterprise data and work collaboratively, which in turn leads to data democratization in the enterprise. Towards the end, we have discussed the Tableau report on BI trends for 2019 and also briefly talked about the BI landscape as assessed by Gartner 2019 Magic quadrants. In the next chapter, we will build a dashboard on our data.gov.in data to bring insights from commodity price data using the free version of Power BI.

CHAPTER 16

Data Visualization Tools

Data visualization tools are the main interface that the end-user sees on top of their data. There are types of roles, and a user can play when it interacts with the BI systems; the two main roles are the author and the reader. The author is the person who created visualization or dashboards using BI skills, and the reader is a user who looks at the output to make a decision, usually management or operational professional. In the previous chapter, we covered the whole stack of BI systems, which not only provides the basis of data visualization but also supports decision systems. In this chapter, we will shift our focus on the data visualization field and show you an example of working in the Power BI tool.

Structure

- Introduction to data visualization
- Data visualization tools
- Introduction to Microsoft Power BI

Objectives

After studying this unit, you should be able to:

- Understand the data visualization tools
- Load, view summary, create data visuals and publish it
- Create various types of plots

Introduction to data visualization

Data visualization is a graphical representation of data. The data visualization uses different types of charts, graphs, and spatial maps to present a massive amount of data in more understandable work. There are

more than a dozen types of charts, and hundreds of their variation created for a different purpose and industry problems.

Our brain process colors and shapes very quickly compared to a list of objects. This type of cognizance feature of mind comes into play when we use data visuals to present complex data points. Data visualization is a form of visual art that grabs our attention and keeps our brain focus on the message or critical elements we want to show, for example, trends, outliers, and patterns.

Consider [Table 16.1](#) and observe how much time it takes you to identify top 5 populous cities of India:

City	Population
Agra	1,430,055
Ahmedabad	3,719,710
Bengaluru	5,104,047
Bhopal	1,599,914
Chennai	4,328,063
Delhi	10,927,986
Hyderabad	3,597,816
Indore	1,837,041
Jaipur	2,711,758
Kanpur	2,823,249
Kolkata	4,631,392
Lucknow	2,472,011
Ludhiana	1,545,368
Mumbai	12,691,836
Nagpur	2,228,018
Navi Mumbai	2,600,000
Patna	1,599,920
Pune	2,935,744
Surat	2,894,504
Tirunelveli	1,435,844

Table 16.1: Indian cities with their population size

Now, we represent the same data in a geo-plot. The size of the bubble is a representation of the population size, and a bigger bubble means more population. [Figure 16.1](#) shows the geo-plot of various cities of India with its population size:

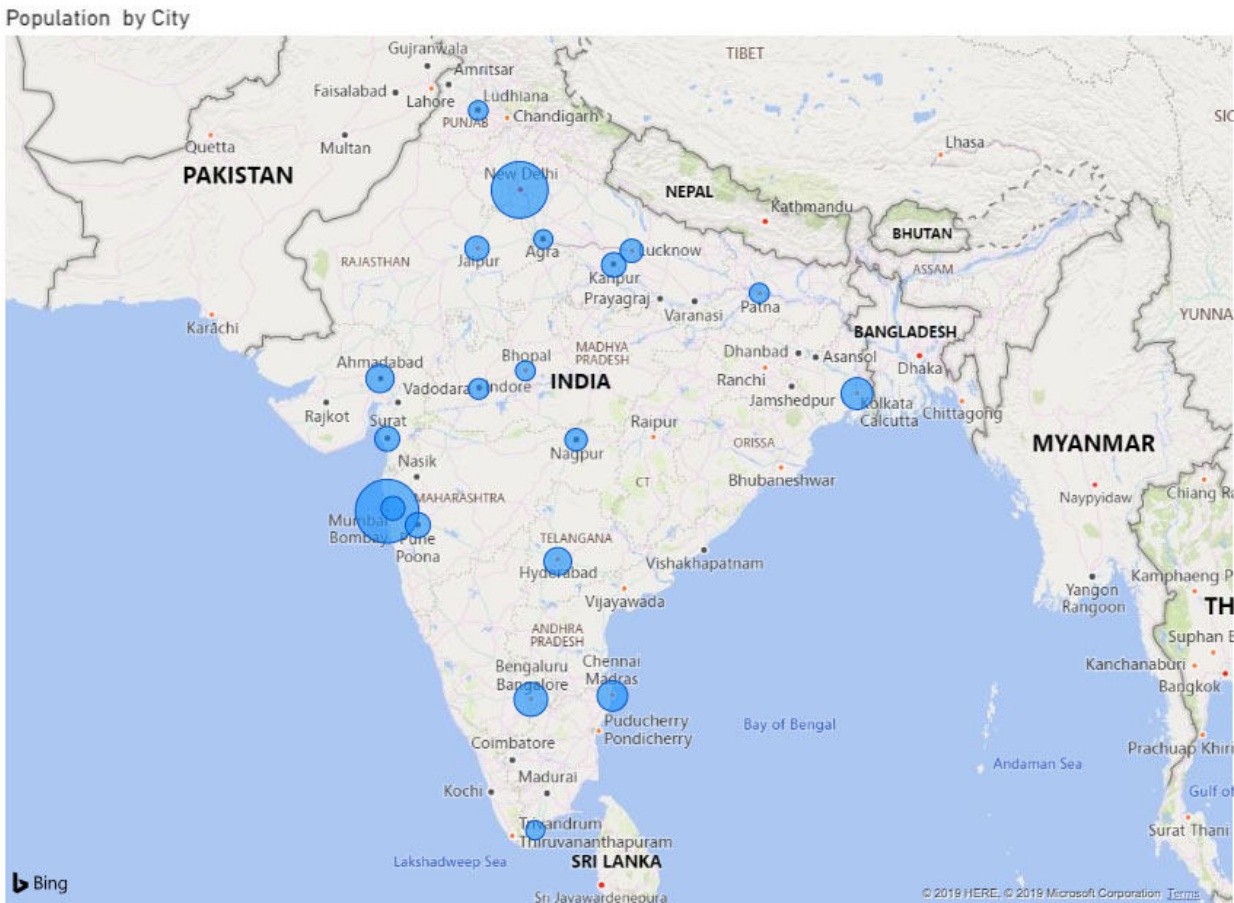


Figure 16.1: Geo-plot of various cities with its population size

You would observe that finding the top 5 cities was quicker, accurate, and less straining in case of visual representation compared to the table. The five most populous cities of India are Mumbai, Delhi, Bangalore, Kolkata, and Chennai.

With the advent of Big Data, the data has grown huge in size and variety. Data visualization is now a necessity to study any data and present it to derive insights.

Data visualization types

There are many types of data charts and visual representations of data. The most common generalization types are:

- Charts
- Tables
- Graphs
- Maps
- Infographics
- Dashboards

These are basic types of visual representations, while charts are a representation of dimensions and measures, the graphs are geotagged data, an infographic is an amalgamation of information and data visuals, dashboards are a combination of all other types of visuals.

In the chapter on EDA, you must have encountered different types of graphs to represent data in different forms. [Figure 16.2](#) shows the most common 4 types of charts:



Figure 16.2: the four types of charts; Bar, Pie, Line, and Cartesian

- **Bar graphs:** show numbers that are independent of each other in a count bar.
- **Pie charts** show the whole quantity divided into categories by the angle of the pie.
- **Line plots:** show continuous variable change over other continuous scales, usually time.
- **Cartesian plots:** you have two or three orthogonal axes to represent different continuous quantities on a different axis.

There are also various types of charts to represent any data. But the above four are the most common.

Data visualization tools

Data visualization tools have been in existence for more than a decade now. Their genesis lies in the performance reports generated from the data warehouses. After the proliferation of Big Data and cloud, the data visualization tools spin-off as a separate toolset providing multiple features to help bring insights faster, and above all, BI is cheaper for small businesses by SaaS and PaaS model.

The market has a plethora of data visualization tools now. From the point of view of technology, there can be three broad categories:

- **Developer friendly:** The tools which are based on JavaScript libraries and allow full flexibility to the developer. They are very popular among SaaS models, where data visualization is built to solve some pre-defined problems.

Example: D3.js, ggplot, Plotly, HighCharts, GoogleCharts, and others.

- **Business-friendly:** The tools which are dragand drop nature, driven by a pre-fixed menu. These pre-defined options cover most of the need for business and quick to learn. They are full-fledged systems deployed on cloud or on-premise.

Example: Power BI, Tableau, Qlik, and others

- **Cloud-Native:** These sets of tools are developed by public cloud providers to allow users to access data visualization of their cloud-native storage. They provide a quick way to sneak into your data stored at databases and filesystems native to the cloud.

Example: Amazon QuickSight, Google Data Studio, and others.

Visualization tool features

The data visualization tool is part of a broader BI stack of an organization. They play an important role in providing an interface to data visualizations and insight generation by business users. While all data visualization tool differs in their offerings, there are some functionalities which user expect from their data visualization tool.

The top 10 features that the model data visualization tool provides for the user are listed below (in order). The following sections will show those features using Microsoft Power BI:

- Robust Data Connectors and Data Model Management.
- Pre-defined charts with drag and drop functionality.
- Allow computed fields and coding in R/Python.
- Filters and actions on charts.
- Ability to create dashboards (collection of charts).
- Publish dashboards via email, social media, and web.
- Host or embed the dashboards in applications.
- Periodic reports and notifications.
- User management.
- Desktop + Mobile + Web Interfaces.

These features are a small set of features compared to the full feature set of the tools provided to the users. You can visit their website or register for a trial to fully explore the tool of interest.

Introduction to Microsoft Power BI

Disclaimer: The authors of this book do not, in any way, endorse or recommend Power BI for end users. The sections and features are listed to help make the reader understand a popular BI tool and kickstart their BI exploration.

As per Gartner Magic Quadrants 2019 for BI tools, the Power BI leads the market by setting itself into Leaders Quadrants along with Tableau and Qlik. For discussion in this chapter, we have picked Power BI, a Microsoft product, for a hands-on explanation of Visualization tool features.

As per the Power BI website (<https://powerbi.microsoft.com/en-us/what-is-power-bi/>), the product is defined as

Power BI is a business analytics solution that lets you visualize your data and share insights across your organization or embeds them in your app or website. Connect to hundreds of data sources and bring your data to life with live dashboards and reports.

You can install Power BI Desktop for free from the Microsoft Store. The desktop version is handy as that does not require cloud infrastructure and can be used on a PC or laptop.

For advanced features and hosting support, it has a different pricing model. The pricing on Power BI is in two-tiers:

- Power BI Pro
 - Self-service and modern BI in the cloud.
 - Collaboration, publishing, sharing, and ad hoc analysis.
 - Fully managed by Microsoft.
- Power BI Premium
 - Enterprise BI, big data analytics, cloud, and on-premises reporting.
 - Advanced administration and deployment controls.
 - Dedicated cloud computes and storage resources.
 - Allow any user to consume Power BI content.

Use case Microsoft Power BI

Building the dashboard in Power BI is a fast and quick process. The important responsibility of critically analyzing the visuals is with the BI analyst. We will define a use case to build a dashboard and show the features we defined in section 4 using Power BI.

Use case: Build a dashboard of commodity prices for a commodity trader using the agricultural market data for daily commodity prices from data.gov.in

Dataset: The dataset can be downloaded from <https://data.gov.in/resources/current-daily-price-various-commodities-various-markets-mandi/>

Top 5 Business Questions to Answer

- What is the costliest commodity of the day?
- Which state mandates are selling tomato, and of which variety?
- The distribution of markets across states by the number of markets they have for agricultural commodities.
- Distribution of onion prices across markets.
- The commodity and their various types sold across markets in India.

Microsoft Power BI console

Once you start the Power BI, you will see a console like this below. The console view is market with a specific area to explain the usage. [Figure 16.3](#) shows the Power BI console:

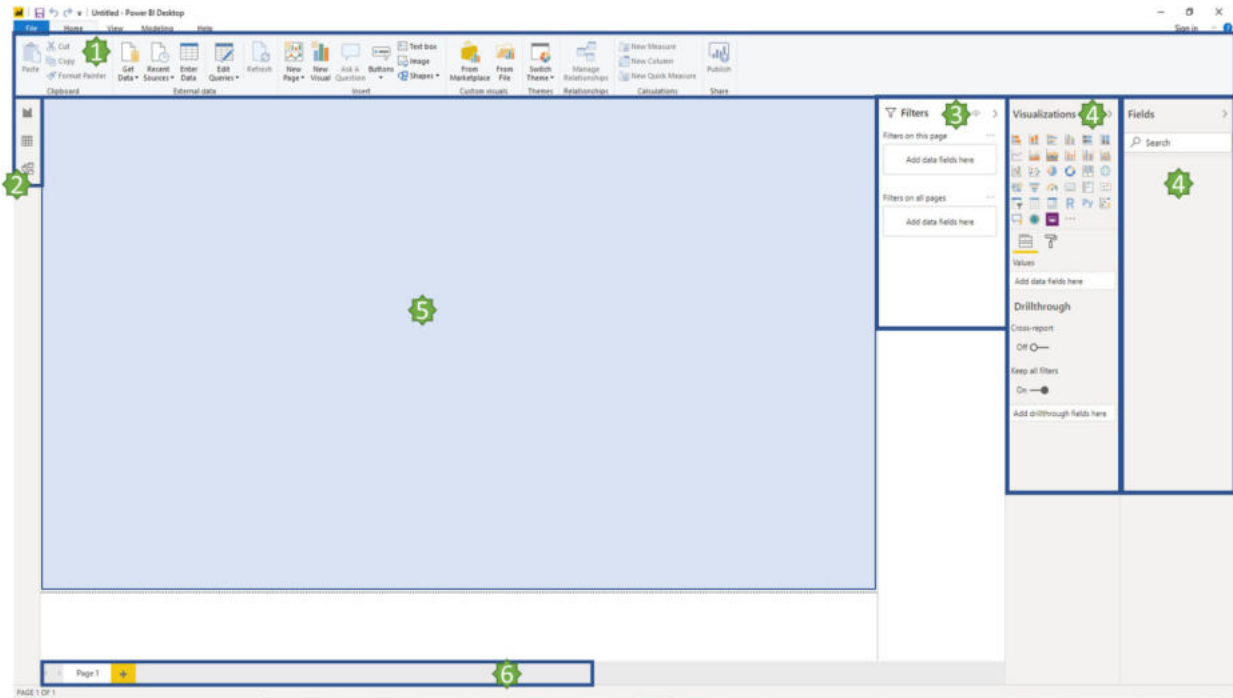


Figure 16.3: Power BI Console

[Table 16.2](#) provides a list of features in the Power BI console.

Area	Description
1	The functionality Ribbon. Here you can save, publish, load data and other functionalities
2	This has the visualization canvas, the data table view, and the data model
3	This place has filters, categorical as well as continuous scale filters
4	This will list all the data points in the datafile and allow the user to drag them to area 5
5	This area is where the visualization is created. It can have multiple visuals in the same canvas
6	This is like sheets in excel. Here separate analysis can be separated by sheets.

Table 16.2: List of features in Power BI console

Load the data

The downloaded data is in a CSV file named `daily_commodity_prices_15102019.csv`.

Go to **getting Data in Ribbon Area** to open a window like below. Select **Text/CSV** to browse the CSV file in your system, as shown in [Figure 16.4](#). You can observe there are numerous data connection options:

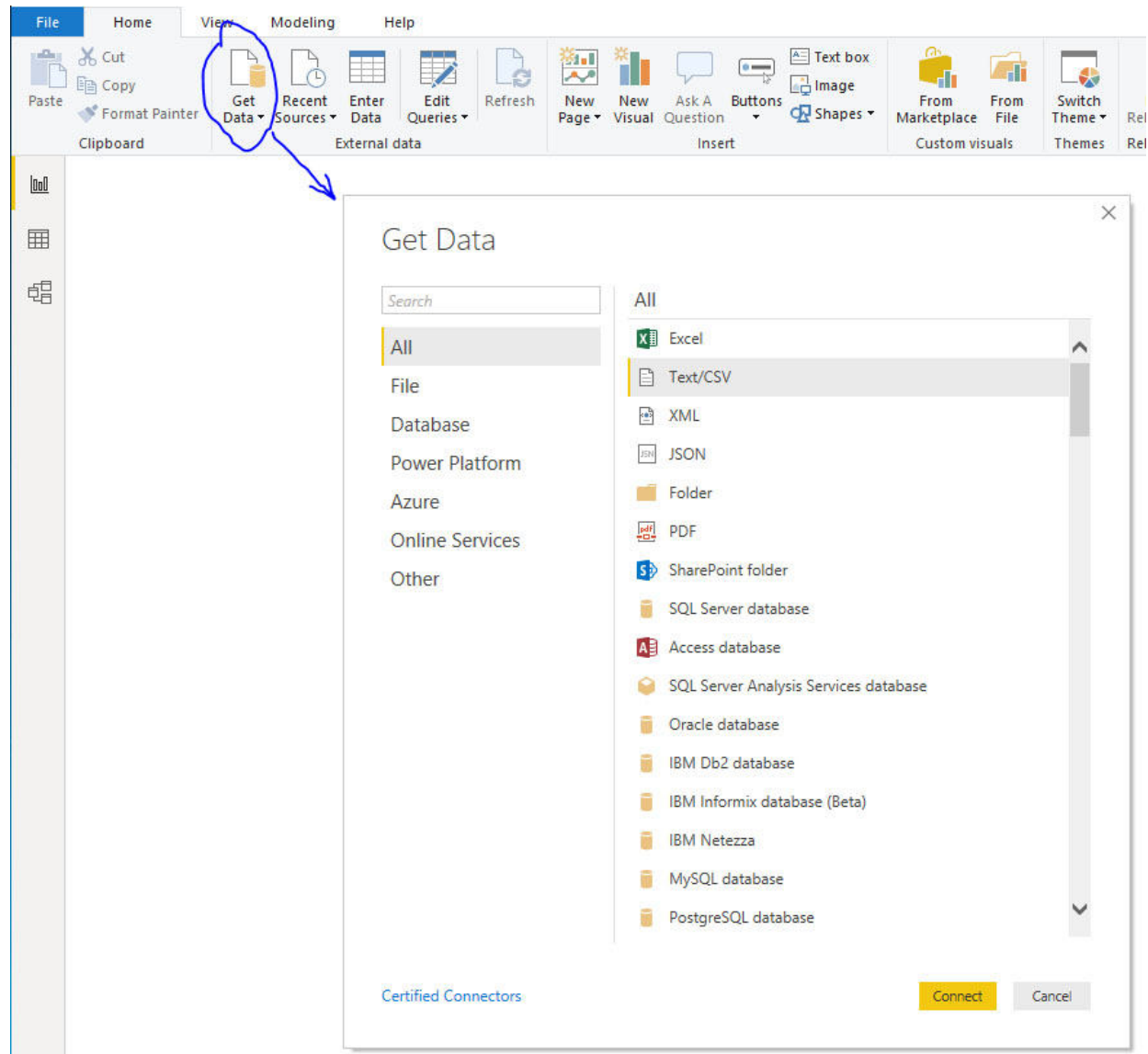


Figure 16.4: Loading data in Power BI

After you load the file, you will see the data summary like below, as shown in [Figure 16.5](#):

daily_commodity_prices_15102019.csv

File Origin: 1252: Western European (Windows) | Delimiter: Comma | Data Type Detection: Based on first 200 rows

state	district	market	commodity	variety	arrival_date	min_price	max_price	modal_price
Andhra Pradesh	Chittoor	Kalikiri	Tomato	Local	15/10/2019	660	2130	1660
Andhra Pradesh	Chittoor	Mulakalacheruvu	Tomato	Local	15/10/2019	700	1800	1400
Andhra Pradesh	Chittoor	Palamaner	Beetroot	Beetroot	15/10/2019	1000	2000	1500
Andhra Pradesh	Chittoor	Palamaner	Bitter gourd	Bitter Gourd	15/10/2019	1000	2000	1500
Andhra Pradesh	Chittoor	Palamaner	Bottle gourd	Bottle Gourd	15/10/2019	600	1200	900
Andhra Pradesh	Chittoor	Palamaner	Brinjal	Brinjal	15/10/2019	1500	2500	2000
Andhra Pradesh	Chittoor	Palamaner	Cabbage	Cabbage	15/10/2019	250	750	500
Andhra Pradesh	Chittoor	Palamaner	Carrot	Carrot	15/10/2019	2500	4500	3500
Andhra Pradesh	Chittoor	Palamaner	Cauliflower	Cauliflower	15/10/2019	500	1500	1000
Andhra Pradesh	Chittoor	Palamaner	Cluster beans	Cluster Beans	15/10/2019	2500	4500	3500
Andhra Pradesh	Chittoor	Palamaner	Cucumbar(Kheera)	Cucumbar	15/10/2019	750	1250	1000
Andhra Pradesh	Chittoor	Palamaner	Green Chilli	Green Chilly	15/10/2019	1000	2000	1500
Andhra Pradesh	Chittoor	Palamaner	Raddish	Raddish	15/10/2019	500	1500	1000
Andhra Pradesh	Chittoor	Palamaner	Ridgeguard(Tori)	Ridgeguard(Tori)	15/10/2019	500	1500	1000
Andhra Pradesh	Chittoor	Palamaner	Tomato	Hybrid	15/10/2019	300	2300	1300
Andhra Pradesh	Chittoor	Piler	Tomato	Deshi	15/10/2019	750	1250	1000
Andhra Pradesh	Chittoor	Vayalapadu	Tomato	Local	15/10/2019	800	4000	2400
Andhra Pradesh	East Godavari	Ravulapelem	Banana	Chakkarakeli(Red)	15/10/2019	1000	1200	1000
Andhra Pradesh	East Godavari	Ravulapelem	Banana	Chakkarakeli(White)	15/10/2019	700	1200	1000
Andhra Pradesh	East Godavari	Ravulapelem	Banana	Karpura	15/10/2019	700	1200	1000

The data in the preview has been truncated due to size limits.

Load Transform Data Cancel

Figure 16.5: Data summary of the loaded data in Power BI

The data summary looks good, and the system has identified the data types accurately. We can load this data into the tool to create visualizations.

Create data visuals

We have been looking for some business questions by means of business questions; we will create the visuals to extract those insights/ answers to the questions:

#	Business question	Supporting data visual
1	What is the costliest commodity of the day?	Bar plot in descending order of average modal price

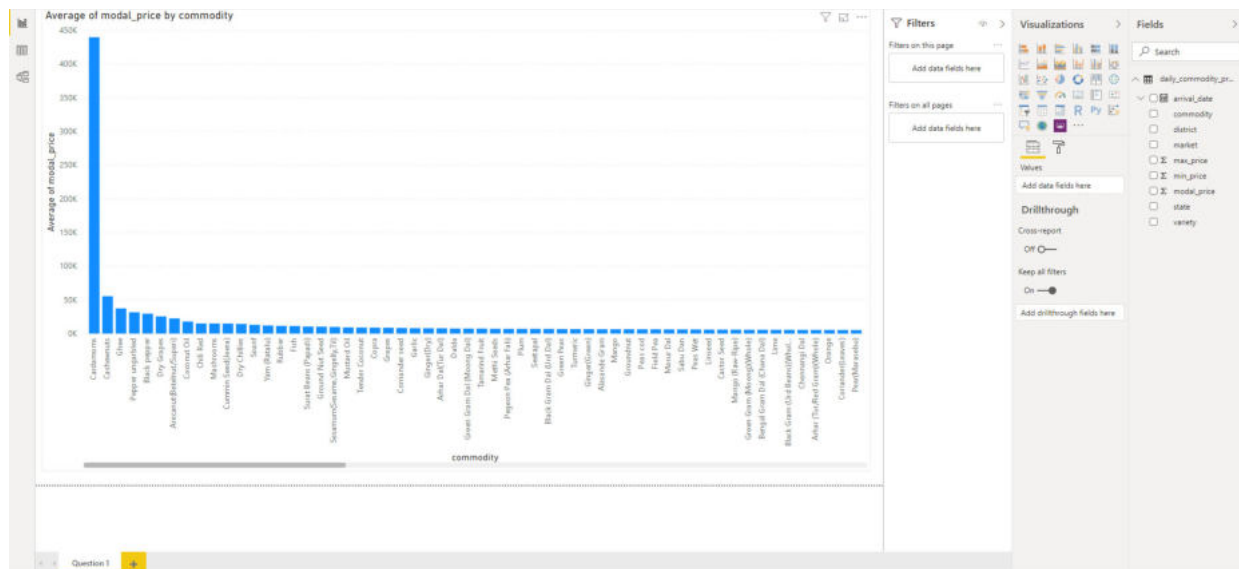


Figure 16.6: Bar Plot

Solution: Cardamoms is the costliest commodity with the average modal price of Rs 440,000/100Kg

#	Business question	Supporting data visual
2	Which state mandis are selling tomato, and of which variety?	A TreeMap with state and variety in dimensions, filter tomato, and shows distinct count as size.

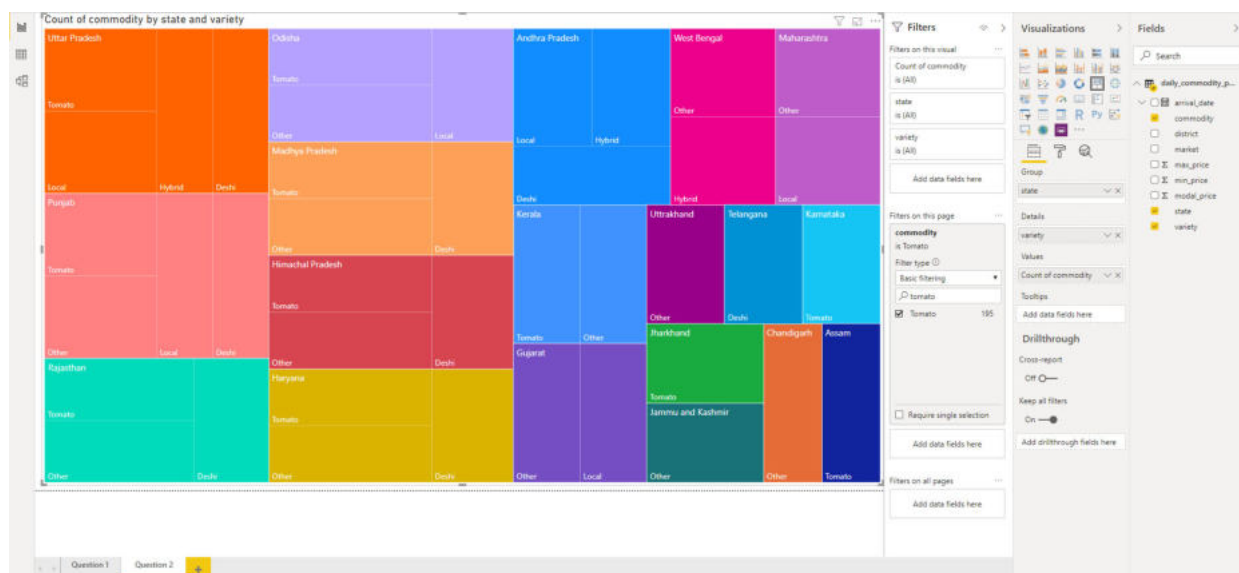


Figure 16.7: TreeMap

Solution: Almost all states are selling the major one being in Uttar Pradesh

#	Business question	Supporting data visual
3	The distribution of markets across states by the number of markets they have for agricultural commodities.	Geo Map on Indian political map with bubble as a number of distinct markets in that state

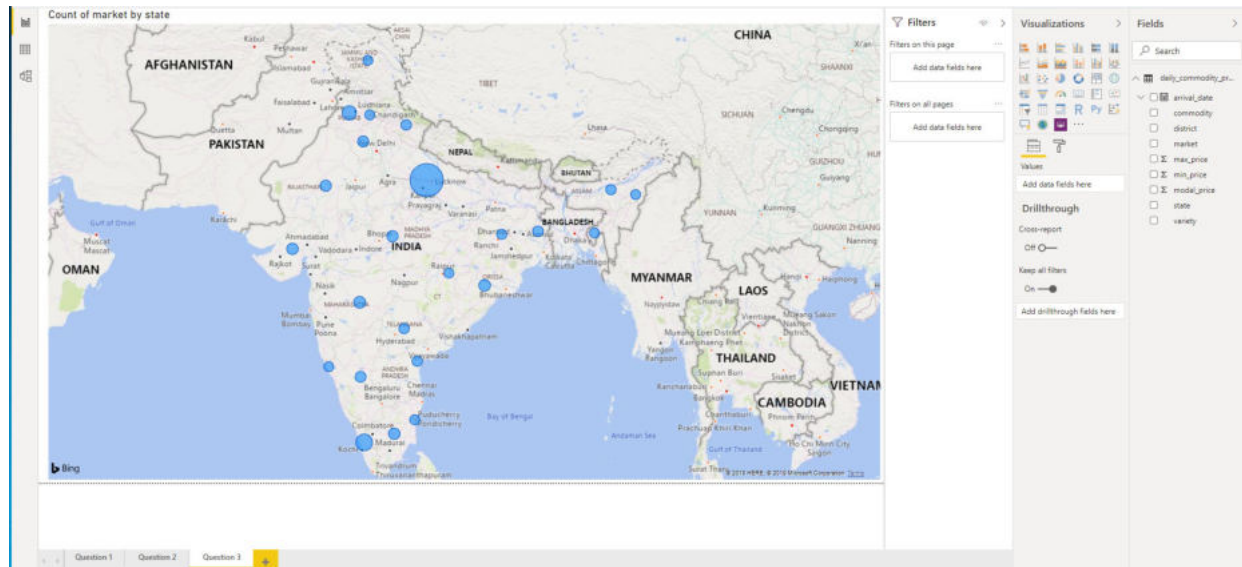


Figure 16.8: Geo Map

Solution: Uttar Pradesh has the highest number of agriculture markets with a count of 1868 markets:

#	Business question	Supporting data visual
4	Distribution of Onion prices across markets	Stacked Area Plot by markets, commodity filter set to Onion

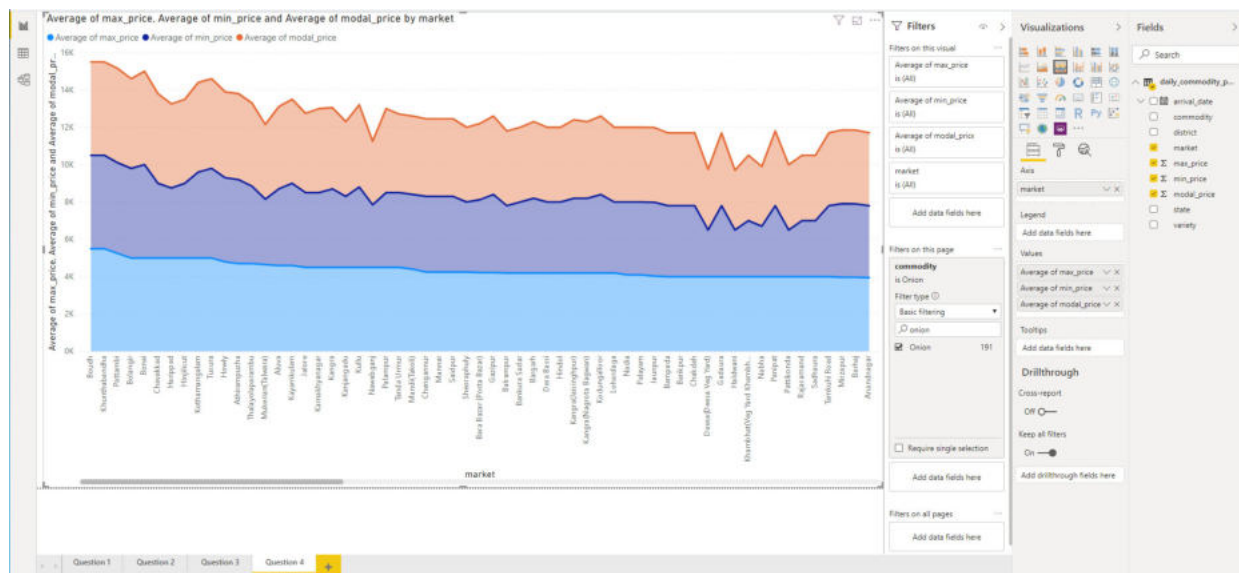


Figure 16.9: Stacked Area Plot

Solution: The distribution of prices of Onion is observed with the highest being at Boudh market:

#	Business question	Supporting data visual
5	The commodity and their various types sold across markets in India.	A Pivot Table with relevant filters

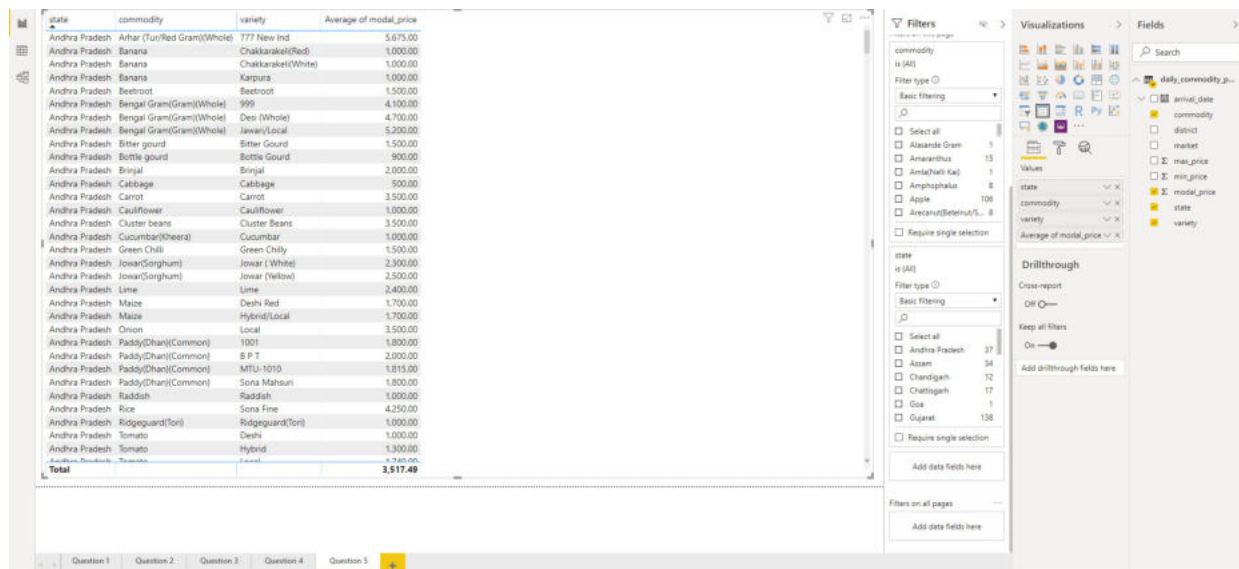


Figure 16.10: Pivot Table

Solution: The pivot table can be filtered to get the exact commodity and variety.

Publish the visuals

The visuals can now be shared with other users using the publish button in the ribbon area. Login to your Power BI account and the visual will be published and then will be available through mobile phone, PC and iPads:

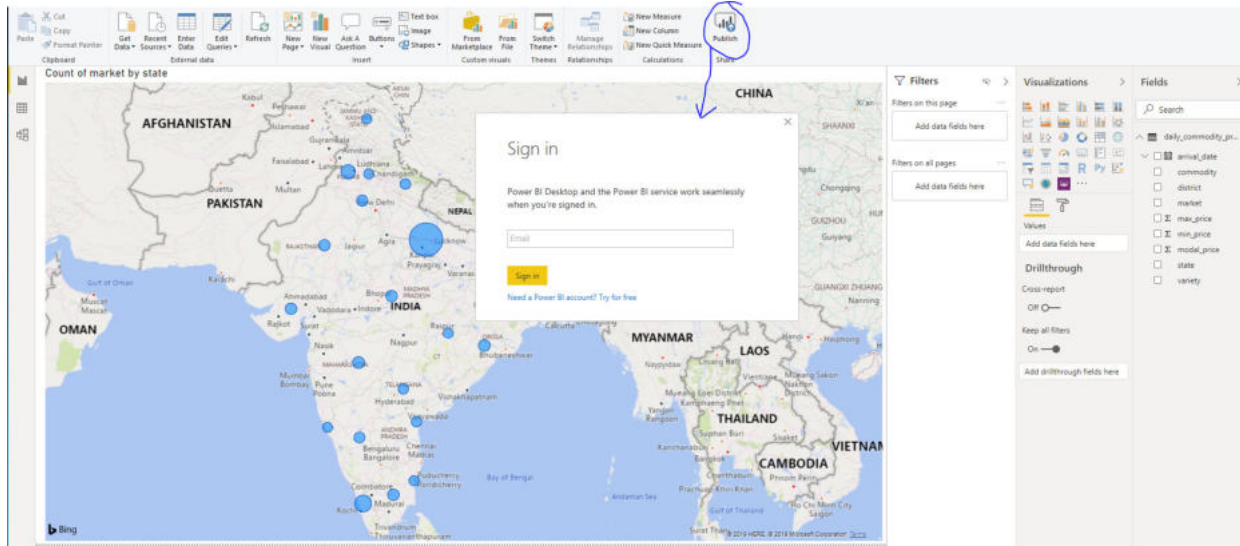


Figure 16.11: Publish Visualization

This concludes the brief hands-on work on Power BI. The data file and Power BI can be further explored by the user to answer more business questions and bring more insights.

Conclusion

In this chapter, we have discussed Data Visualization tools and how they play a central role in BI systems of organizations. There is a type of data visualizations and charts that constitute the summary of data in visual form. Further, the chapter talks about the features of modern visualization tools and how the tool universe can be divided into three types of business user-friendly, developer-friendly, or cloud-native. The chapter then introduces a popular tool, Microsoft's Power BI, and build a use case on data.gov.in dataset. The business questions are then answered using appropriate data visualization in the Power BI tools. The reader is encouraged to learn more about tools and their functionalities. This chapter concludes the section on the Business Intelligence of the book. In the next two chapters, we will

provide you two research papers published around data science solutions on industrial use cases.

CHAPTER 17

Industry Use Case 1 - Form Assist

In the previous chapter, we have learned all the concepts in machine learning and related methods applicable to solve real-time problems. From this chapter, we are going to see the real use cases of application of machine learning technologies in the industry. It will provide enough support to enhance the knowledge of learner how we approach the research problems and what are the steps they can follow while they work on any real-time use cases which need to be solved by using the machine learning algorithms. This chapter gives the idea of how to convert the handwritten forms into digital data format by using deep learning methods.

Structure

- Introduction
- Related works
- Proposed work
- Data augmentation
- Optimization
- Feature extraction
- Image thresholding
- Classifier
- Results
- Conclusion

Objective

After studying this chapter, you should be able to:

- Understand the problem in the conversion of handwritten form to digital format.

- Understand the concepts behind the deep learning algorithm to use it in the real-time problem.
- Evaluate the results based on different visualization techniques.

Abstract

Customer agreement is required to follow statutory and legal requirements, which include agreements to be manually signed. In India, paper forms are still prevalent in the Banking Industry. The paper forms require customers to fill a template form in capital letters and manually sign by agreeing to the terms. This creates a challenge in analytical systems as the data is captured outside the system and requires time to become part of the data pipeline. The future of banks is poised to be digital. However, we still need historical data for train models for current data applications. This limitation is a known bottleneck in designing data applications for real-time decision making. Developing **Optical Character Recognition (OCR)** with capabilities commensurable to that of human is still not achievable, despite decades of excruciating research. Due to the idiosyncrasy of individual form, analysts from industry and scholastic circles have coordinated their considerations towards OCR. The work in this paper shows an efficient model to capture offline handwritten forms and convert them into digital records. The model techniques are based on deep learning methodologies and show higher accuracy for our testing set of real application forms of selected Banks. We have experimented with different feature extraction techniques to extract handwritten characters in the forms. Our experimentation has evolved over time to find a generalized solution and better results. The final model uses the relative position of the characters for extracting characters from the forms and **Convolutional Neural Networks (CNNs)** to predict the characters. The paper also discusses the serverless architecture to host the FormAssist as a REST API with a model calibration feature to accommodate multiple types of forms.

Introduction

Pattern recognition is the science of making inferences from perceptual data based on either a priori knowledge or on statistical information. It is a vital challenge in the field of computer vision and deep learning. It is generally done with feature extraction and classification. The feature extraction

regularly utilizes an assortment of techniques to get a portrayal of the information and afterward utilize the classifier to arrange the information. The procedure is led physically and independently [1].

Handwritten recognition is an area of pattern recognition that characterizes the capacity of a machine to dissect designs and distinguish the character. It has been hailed as a standout amongst the most interesting and testing branch in the field of artificial intelligence and optical character recognition [2]. An assortment of procedures and approaches has been proposed, yet it still an uncertain issue. Notwithstanding, it is a testing errand, particularly handwriting recognition on form documents. A few issues in handwriting recognition are because of the high ambiguity of the information, as the composed characters of every individual are unique, a few characters have fundamentally the same as shape, disengaged, or bending characters [3].

OCR is a procedure that can change over content, exhibit in the computerized picture, to editable content. It enables a machine to perceive characters through optical components. The procedure includes some pre-handling of the picture document and, after that, obtaining vital information about composed content. That learning or information can be utilized to perceive characters [4-8]. OCR comprises of numerous stages, for example, pre-processing, normalization, feature extraction, classification, and recognition. The contribution of one stage is the yield of the following stage [9]. The errand of preprocessing identifies with the expulsion of commotion and variety in manually written.

OCR problems can be solved with various machine learning algorithms. We here have taken a real-life use case of a sample bank form to apply popular techniques to extract handwritten data from it. The numbers from research look exciting, but the real-time scenario is different and more challenging. The first question from a person who needs an OCR solution is the scanner type used. The final output depends on lots of factors starting from the scanner model to the algorithm selection. we have tried to scrutinize them to choose the most economical and accurate solution. In our attempt to provide a complete solution to a data science problem, we have built a user-friendly web portal to read the scanned paper forms and display results.

[Related Work](#)

Handwritten digit recognition has as of late been of extremely enthusiasm among the scientists on account of the development of different machine learning, deep learning and computer vision algorithms. *Anuj Dutt* and *Aashi Dutt* in their paper (IJARCET-VOL-6-ISSUE-7-990-997) [10], analyze the consequences of probably the most broadly utilized machine learning algorithms like SVM, KNN, and RFC and with deep learning algorithms like multilayer CNN utilizing Keras with Theano and TensorFlow. Utilizing these, they could get the exactness of 98.70% using CNN (Keras+Theano) when contrasted with 97.91% using SVM, 96.67% using KNN, 96.89% using RFC.

Darmatasia et al. [1] propose a workflow and a machine learning model for recognizing handwritten characters on form documents. Their learning model depends on CNN as a powerful feature extraction and SVM as a high-end classifier. The proposed technique is more effective than altering CNN with complex architecture. They have assessed some SVM and found that the linear SVM using L1 loss function and L2 regularization, giving the best execution both of the accuracy rate and the calculation time. The recognition rate accomplished by the proposed strategy is 98.85% on numeral characters, 93.05% on capitalized characters, 86.21% on lowercase characters, and 91.37% on the merger of the numeral and capitalized characters, while the first CNN accomplishes a precision rate of 98.30% on numeral characters, 92.33% on capitalized characters, 83.54% on lowercase characters, and 88.32% on the merger of the numeral and capitalized characters. The proposed strategy was additionally validated by utilizing ten folds cross-validation, and it demonstrates that the proposed technique still can enhance the precision rate. The overall system gives an accuracy rate of 83.37% on ten different test form documents.

Balci et al. [11] seek to classify an individual handwritten word so that handwritten text can be translated into a digital form. They have utilized two principle ways to deal with achieve this errand: classifying words directly and character segmentation. For the former, they have used CNN with various architectures to train a model that can precisely classify words. For the latter, they have used LSTM with convolution to build bounding boxes for each character. They, at that point, pass the segmented characters to a CNN for classification, and after that, reconstruct each word as indicated by the results of classification and segmentation.

The choice of classifiers and feature extraction strategies has a prime part in accomplishing the ideal classification exactness in the character recognition system. In *Katiyar et al.* [12], an efficient Support Vector Machine based off-line handwritten character recognition system has been developed. It is obvious from the experimental results that the execution of SVM outperforms other states of the art techniques.

Pai et al [13] present the basics of the OCR method with its parts, for example, pre-processing, feature extraction, classification, post-processing, and many more. This survey additionally examines distinctive thoughts executed before for recognition of a character.

Proposed work

A leading banking institution account opening application form was considered using economical scanning options starting from a phone camera to a daily use scanner. A regular day to day used Flatbed Scanner has been chosen. The snippet of the scanned copy of the original unfilled form is shown in [Figure 17.1](#). As a fresh mind to solve a problem in OCR, we started from a basic level. Initially started with extracting the characters out from the forms and then applied a machine learning model to predict the characters in the form:

PLEASE FILL THE FORM IN BLOCK LETTERS AND BLACK INK

Preferred Home Branch _____ Employee Code (Applicable only for Kotak Bank Ltd) _____

OPTY ID _____

Purpose ☐ Savings Account ☐ Current Account ☐ Deposits ☐ Third Party Products ☐ Other Services

PERSONAL DETAILS * Fields are Mandatory Existing CRN ☐ YES ☐ NO (Please fill the below details)

*CKYCR ☐ New ☐ Existing – No Change ☐ Existing – Update Change ☐ Update CKYCR Change ☐ Local ☐ Global
C-KYCR No. _____ Local change will not be updated in Central KYC Repository (CKYCR) and will only be applicable to Kotak Mahindra Bank Limited

*Name (First Name) _____ (Middle Name) _____ (Last Name) _____ (Spouse Name) _____ (Mention all characters, if any) _____

*Maiden Name (Applicable to married women, documentary proof required) (First Name) _____ (Last Name) _____ *Mother's Maiden Name (Mention Mother's Pre-Marriage Name) (First Name) _____ (Last Name) _____

*DOB (DD MM YYYY) _____ Minor ☐ Senior Citizen ☐ *Father / ☐ *Spouse Name (If PAN not available Father's Name Mandatory) (First Name) _____ (Last Name) _____

*Residential Status ☐ Residential Indian ☐ Foreign National *Citizenship ☐ Indian ☐ Others _____

*Religion ☐ Hindu ☐ Muslim ☐ Christian ☐ Sikh ☐ Zoroastrian ☐ Others _____

Category ☐ General ☐ OBC ☐ SC ☐ ST *Education ☐ Non-Graduate ☐ Graduate ☐ Post Graduate ☐ Others _____

*Gender ☐ Male ☐ Female ☐ Transgender *Marital Status ☐ Single ☐ Married ☐ Others _____

*Annual Income ☐ 0 - 2 lakhs ☐ > 2 - 5 lakhs ☐ > 5 - 10 lakhs ☐ > 10 - 25 lakhs ☐ > 25 lakhs

Facebook ID _____ Twitter ID _____

*Occupation Type Service – ☐ Private Sector ☐ Public Sector ☐ Government Sector ☐ Professional ☐ Self Employed ☐ Retired ☐ Housewife ☐ Student ☐ Business ☐ Not Categorized

Permanent Address (Up to 90 characters only)

Line 1 _____

Line 2 _____

Line 3 / Landmark _____

*City _____ *PIN Code _____

*State _____ Telephone No. (S T D) _____

Figure 17.1: Snippet of the scanned form

Figure 17.2 shows the snippet of the filled form:

*Name MR ALOK KUMAR PANJIYAR

*Maiden Name _____

*Mother's Maiden Name NITU DEVI (Mention Mother's Pre-Marriage Name)

*DOB 26 06 1995 Minor ☐ Senior Citizen ☒ *Father / ☒ *Spouse Name SHATRUGHAN PANJIYAR (If PAN not available Father's Name Mandatory)

Permanent Address (Up to 90 characters only)

Line 1 A T H G A O N

Line 2 _____

Line 3 / Landmark SC PATH

*City GUWAHATI

*State ASSAM

*PIN Code 781001

Telephone No. _____

Figure 17.2: Snippet of the filled form

Work architecture

The proposed work architecture is based on CNN to classify the images. This algorithm takes the image as an input, passes it through a series of convolutional, non-linear, pooling (down-sampling), and fully connected layers, and hence provides an output. The output can be a single class or a

probability of classes that describe the image in a better way. A flowchart is shown in [Figure 17.3](#), which describes the entire proposed model briefly:

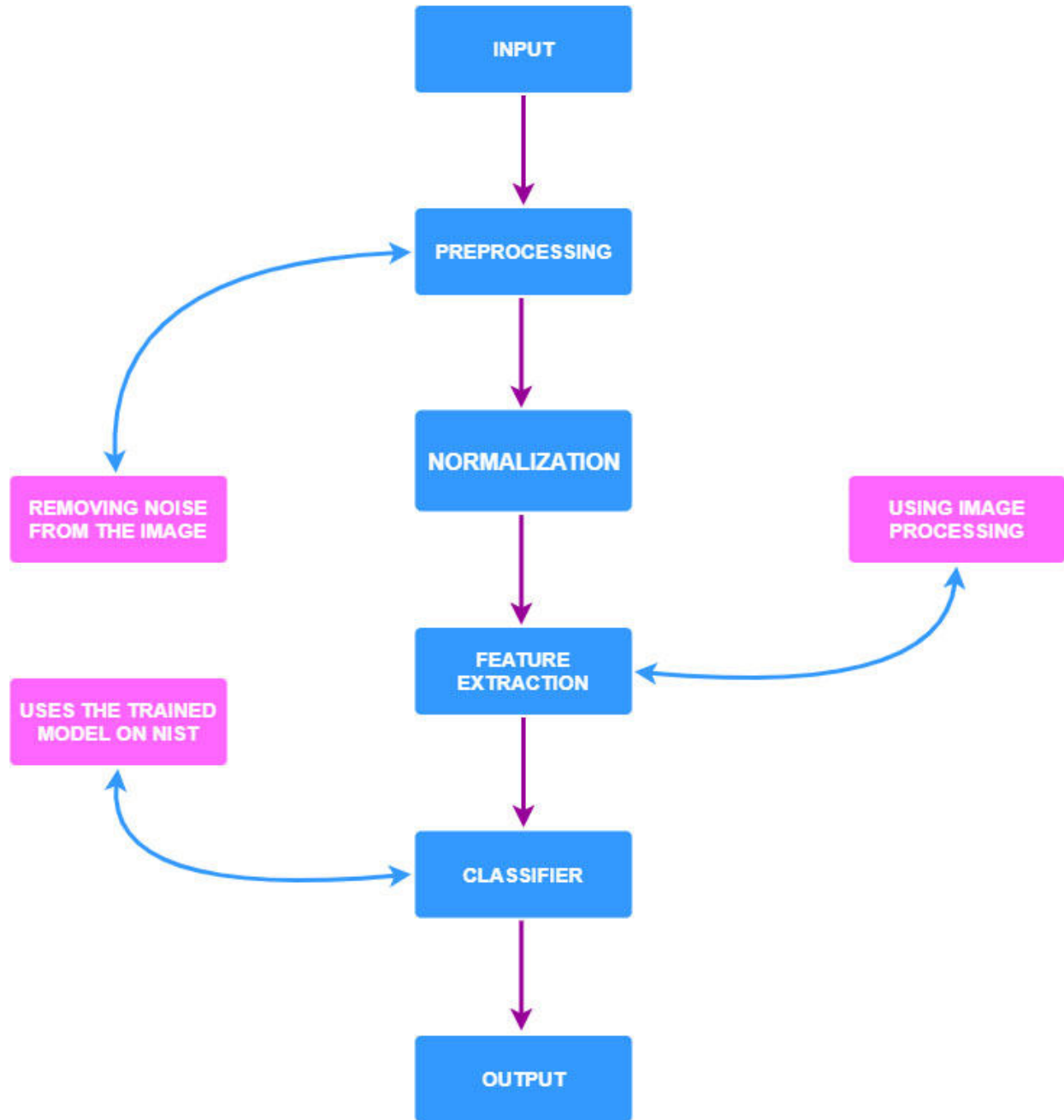


Figure 17.3: Flowchart of the FormAssist Model

The weights(w) in this algorithm are updated with the equation as stated below:

$$w = w_i - \alpha * (\delta L / \delta w) \quad (1)$$

w = Weight, w_i = Initial Weight, α = Learning Rate

An optimal learning rate was targeted so that the algorithm runs smoothly and fast. L is the loss function, and $\delta L / \delta w$ is the derivative of the loss function with respect to w .

NIST dataset

For training the alphabets and numeric models, *NIST Handprinted Forms and Characters Database* was used. The numeric model was trained using the MNIST dataset, which is a subset of the larger NIST dataset. The MNIST dataset consists of the 10 digits. For training the alphabet model, the EMNIST dataset was used. Below in [Figure 17.4](#), an example of the MNIST dataset is shown:



Figure 17.4: Example of the MNIST dataset

There were a few challenges while training the model. Numpy's `np.loadtxt()` was a very slow method for loading the dataset as compared

to `pandas.PD.read_csv()`. Pandas load the dataset exponentially faster than Numpy.

The task of minimizing the loss involves adjusting the weights so that the loss is minimal. Visually, we want to get to the lowest point in the bowl-shaped loss function, as shown in [Figure 17.5](#). This figure shows the idea of minimizing the loss. Therefore, the derivatives of the loss function are taken with respect to the weights in the backpropagation step:

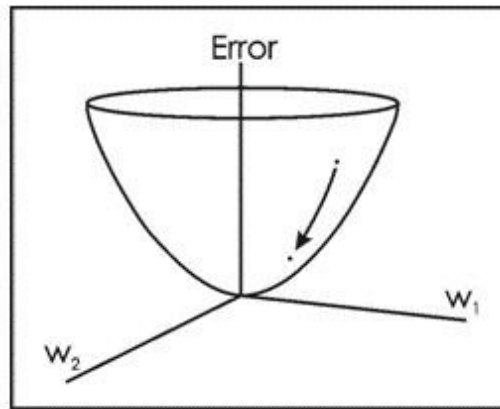


Figure 17.5: Optimization algorithm works to get an optimal value of the weights so as to minimize the cost(error).

Activation function – ReLU

ReLU (Rectified Linear Units) is used as the activation function for the hidden layers in the network. The function is defined in the below-given equation. [Figure 17.6](#) shows the graph of the ReLU function:

$$RELU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

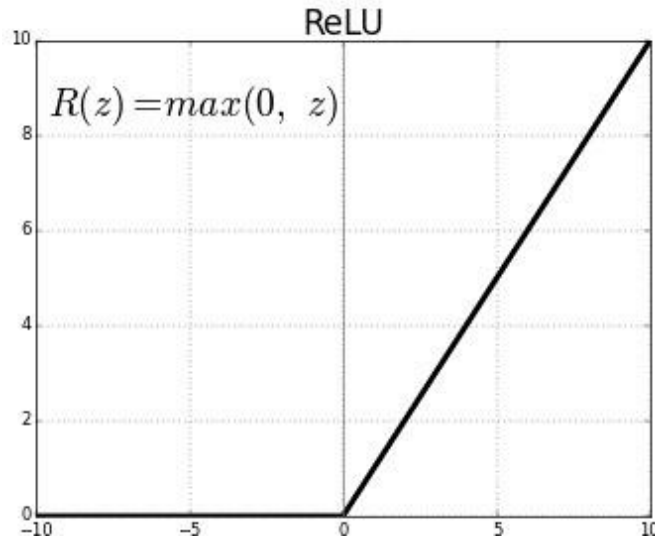


Figure 17.6: ReLU function graph

Dropout

Dropout regularization is used to prevent overfitting. Dropout randomly shuts down some neurons in the network during forwarding propagation and backward propagation. This ensures that weights are not too large, and the model is not overfitting. In [Figure 17.7](#), the dropout process is shown briefly:

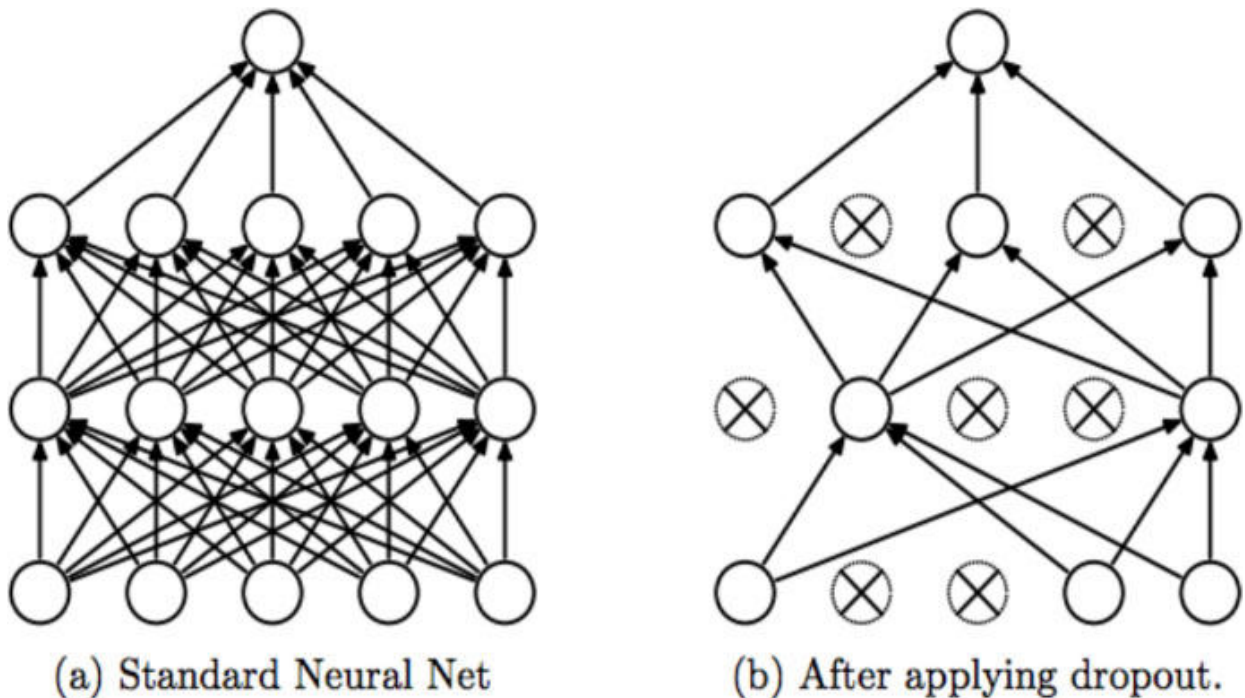


Figure 17.7: Example of the dropout process

Data augmentation

Before training the model, data augmentation is used to make full use of the dataset. In data augmentation, the image is randomly distorted in random manners to create new additional data, which helps to train the model better. In [Figure 17.8](#), digit 6 is shown as an example of data augmentation:

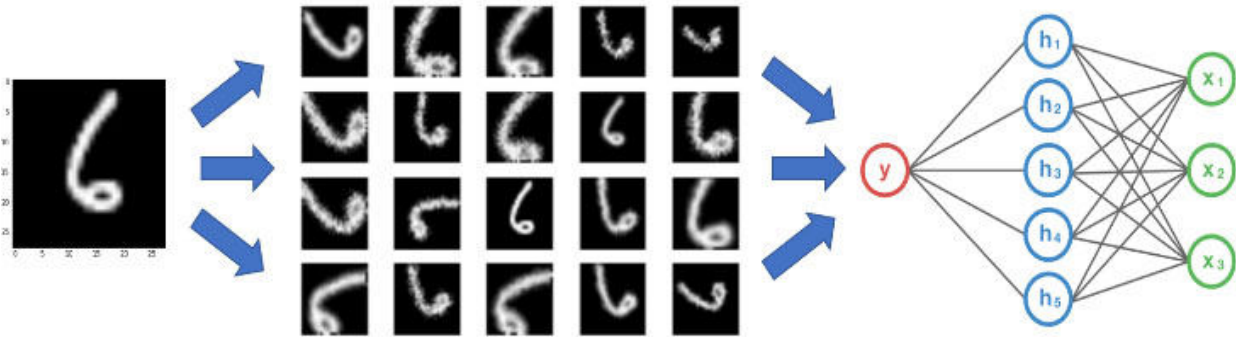


Figure 17.8: Example of Data Augmentation of digit 6

Optimization

Adam is used as the optimization algorithm. It is an extension of Stochastic Gradient Descent and is used to update weights in the neural network by iterating through the dataset. Adam has the combined advantages of optimization algorithms, AdaGrad, and RMSprop.

In the article [\[14\]](#), a good representation of Adam's algorithm can be seen. The advantages of using Adam on non-convex optimization problems are stated below [\[14\]](#):

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Invariant to diagonal rescale of the gradients.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for non-stationary objectives.
- Appropriate for problems with very noisy/or sparse gradients.

- Hyper-parameters have intuitive interpretation and typically require little tuning.

The performance of various optimization algorithms [15] is shown in [Figure 17.9](#). It is clearly seen that Adam performs the best. While training with Adam Optimization algorithm, we get the lowest cost (the pink line):

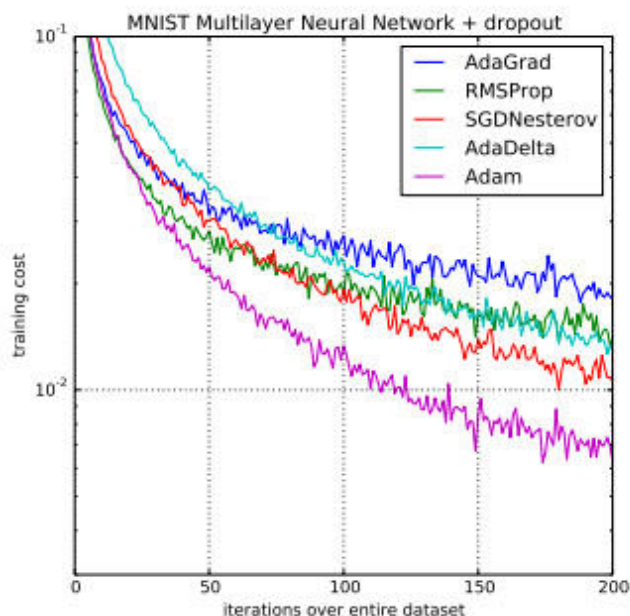


Figure 17.9: Comparison of Adam to Other Optimization Algorithms [15].

Feature extraction

Feature extraction is one of the most important parts of optical character recognition. Here we try to extract the most important features from the image, and a good algorithm for feature extraction can significantly improve the accuracy of the model. The feature extraction method is shown in [Figure 17.10](#):

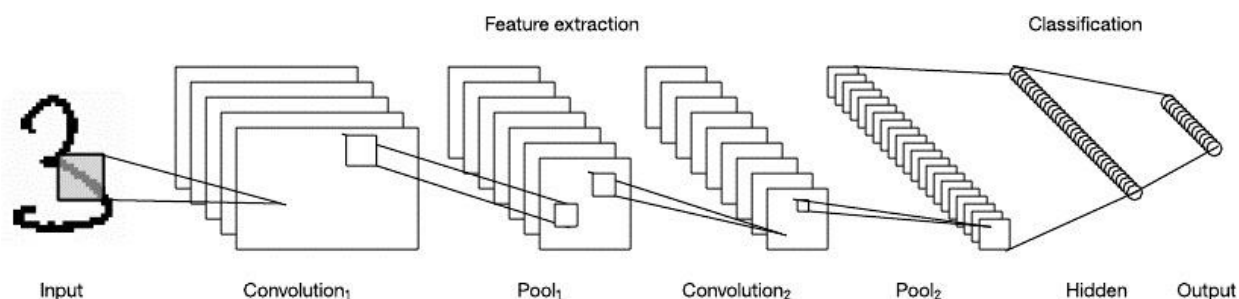


Figure 17.10: Example of Feature Extraction

The challenge in this work was the boundaries of the box, which is not clear in the scanned copy. This becomes hard to extract character based on the boxes. A more robust approach is used to extract the characters based on the relative position of the black markers on the four corners of the form. The position of each character in the form is manually taken into account. OpenCV is used to read the image files, and each image is divided into three equal parts. For matching the black boxes in the original image and the scanned image, OpenCV's `cv2.matchTemplate` is used. *Template Matching* is a technique for looking and finding the location of a template image in a bigger image. It essentially slides the template image over the input image and compares the template and patch of input image under the template image. After template matching, the images are converted into grayscale images using `cv2.cvtColor` and `cv2.BGR2GRAY`.

Image thresholding

There is some noise in the image, which can decrease the efficiency and accuracy of the model. Image thresholding is used to prevent decrease in accuracy and efficiency due to noise in image. In this process, we set a threshold value, and the pixel values that are greater than this threshold value are set to white (a value of 255), and other pixel values are set to black (a value of 0). [*Figure 17.11*](#) is an example of different types of image thresholding:

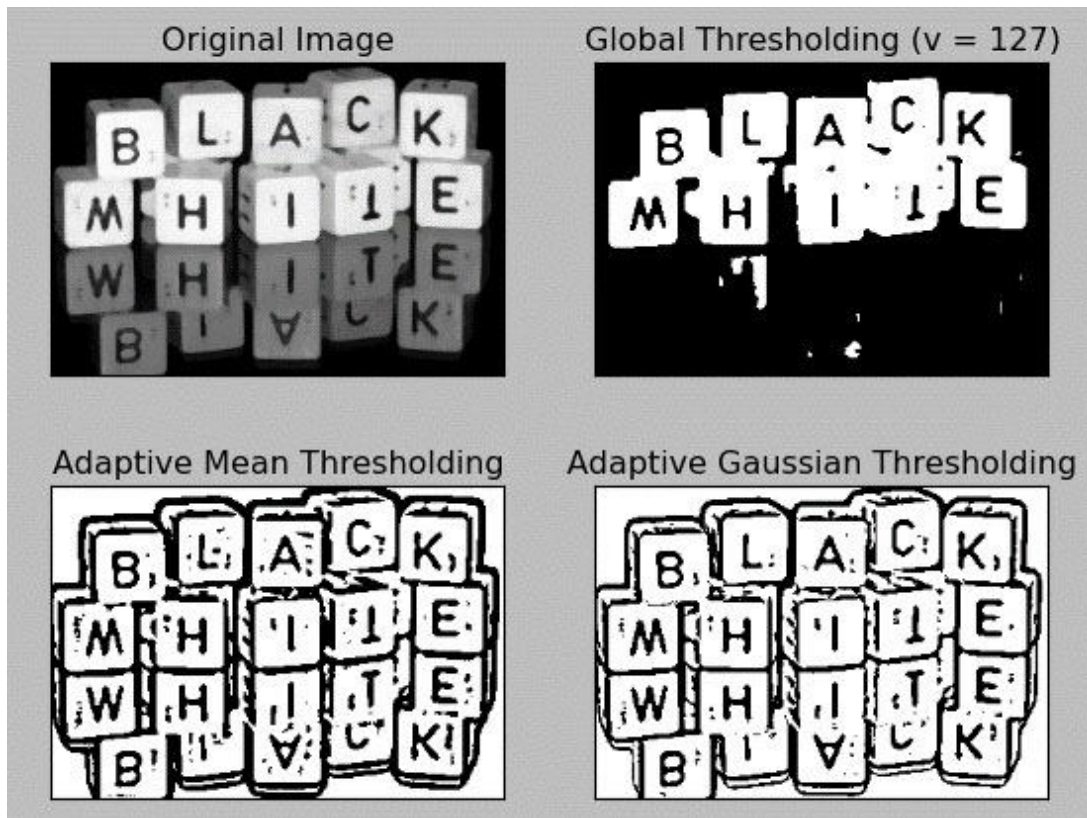


Figure 17.11: Example of different types of image thresholding

Classifier

The training part is done using Keras API. Keras is a high-level neural network API written in Python language, capable of running on top of TensorFlow, CNTK, and Theano.

Results

A confusion matrix is printed that provides the percentage of accuracy with which each digit and each alphabet has been recognized. A confusion matrix defines a specific table that allows the visualization of the performance of an algorithm by providing the accuracy corresponding to each of the input and output classes [16].

The model is tested on a few sample forms, and a confusion matrix is obtained. Confusion matrix of digits and alphabets are shown in [Figure 17.12](#) and [Figure 17.13](#), respectively:

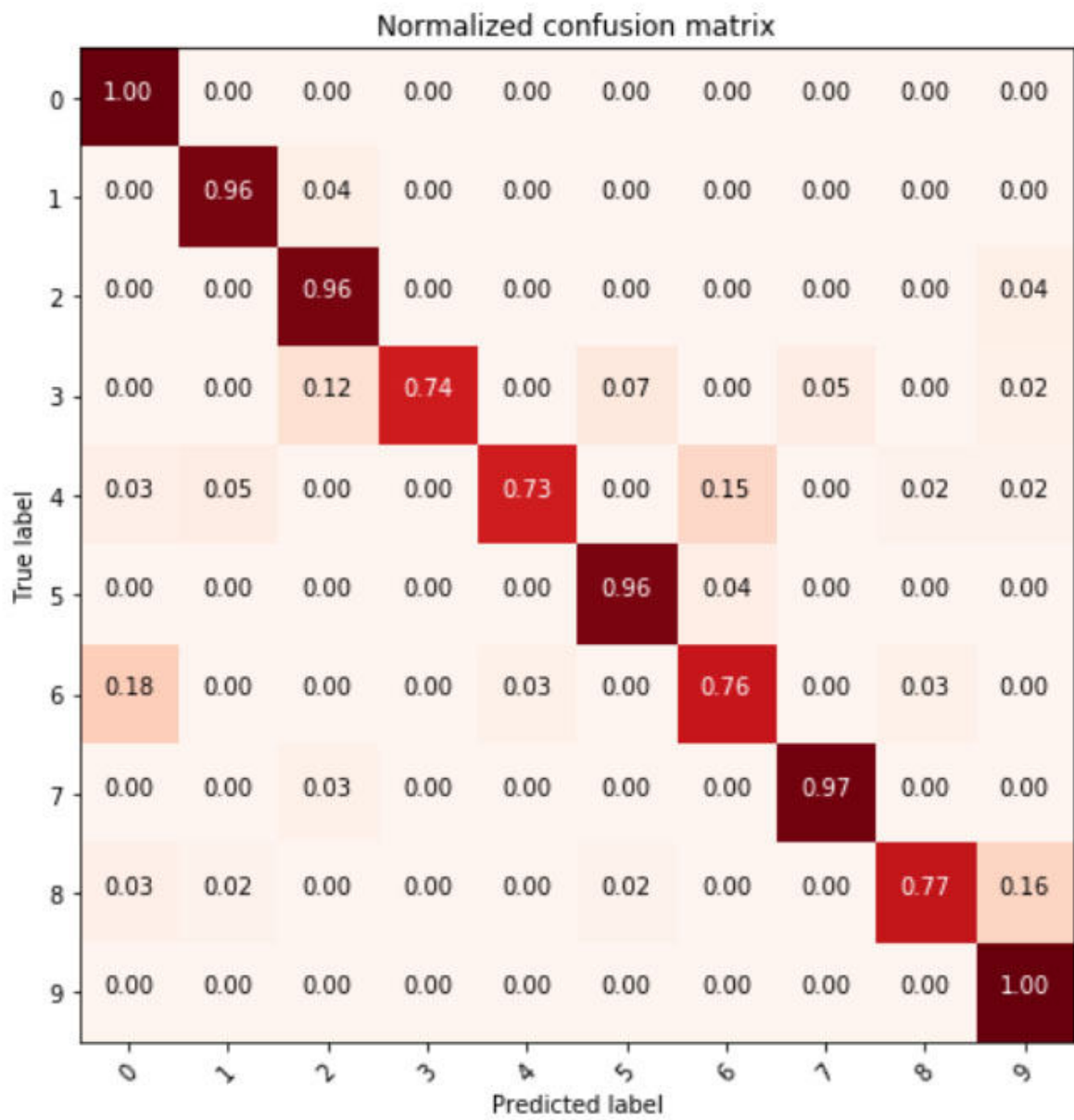


Figure 17.12: Confusion Matrix of 10 digits

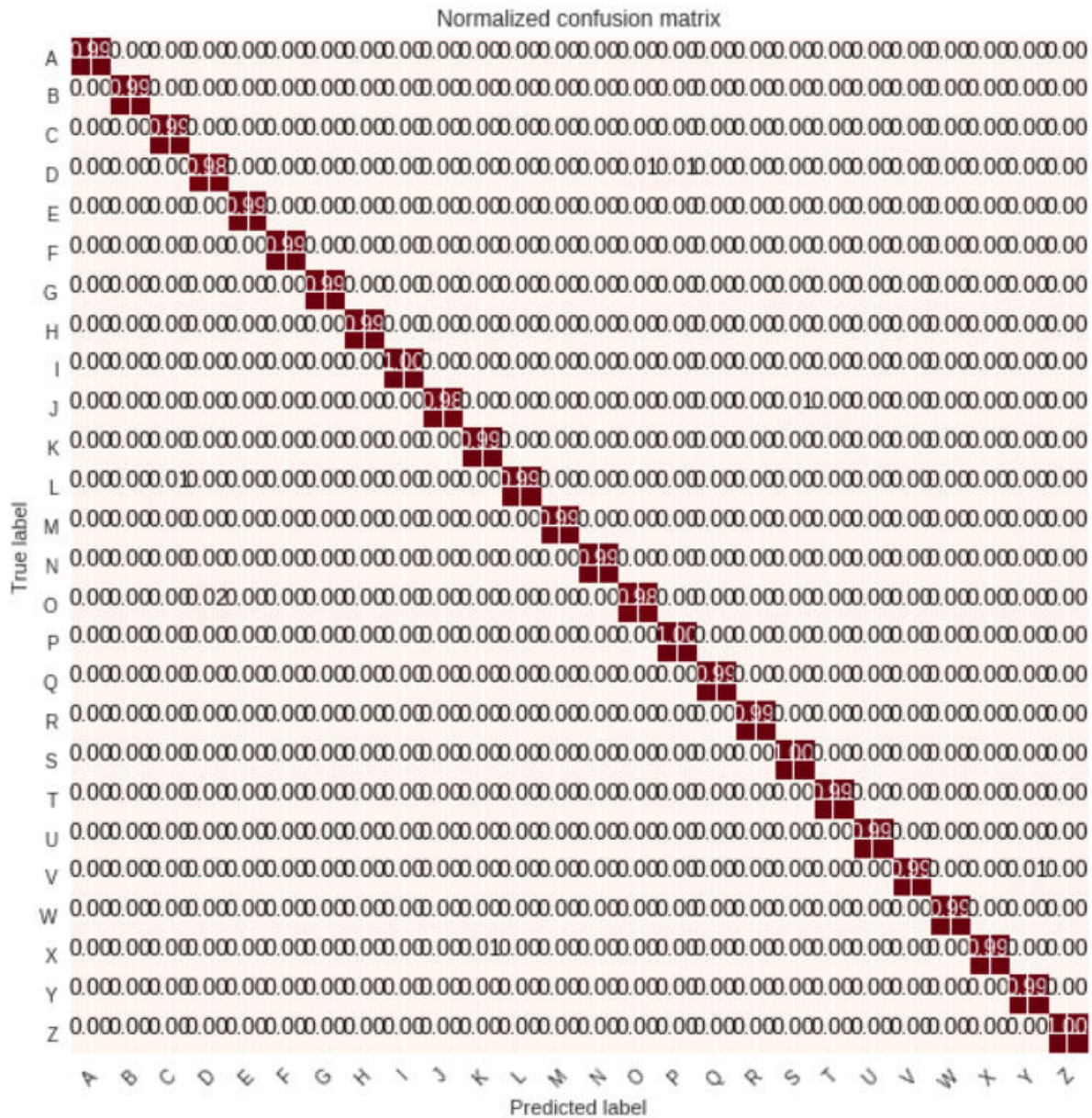


Figure 17.13: Confusion Matrix of 26 alphabets

Most of the characters show excellent accuracy, which more than 90%. Few similar-looking characters were giving a lower but good accuracy above 75%. The details are shown in [Table 17.1](#) below:

CHARACTERS	ACCURACY	ACCURACY (in %)
0, 1, 2, 5, 7, 9 C, D, E, G, H, K, L, N, T, U, W, X, Y, Z	EXCELLENT	90 +

6, 7 J, M, O, R	VERY GOOD	80-90
3, 4 A, B, F, I, P, Q, S, V	GOOD	70-80

Table 17.1: Accuracy of the Characters

A web-based UI is created so that a user can view and edit the response. Then it can be saved in a JSON file. This entire framework is hosted on a webpage, shown in [Figure 17.14](#):

Figure 17.14: Screenshot of the web-based UI of FormAssist

Conclusion

The improved performance of OCR with the latest developments in Deep Learning has opened up scope for high-value business use cases in Banking and Insurance Industry. The impact of improved OCR methodologies will be far-reaching in the near future, mainly due to the need for data for real-time analytics. The above-discussed methodology is implemented in Probyto's Business solution for industrial use – FormAssist.

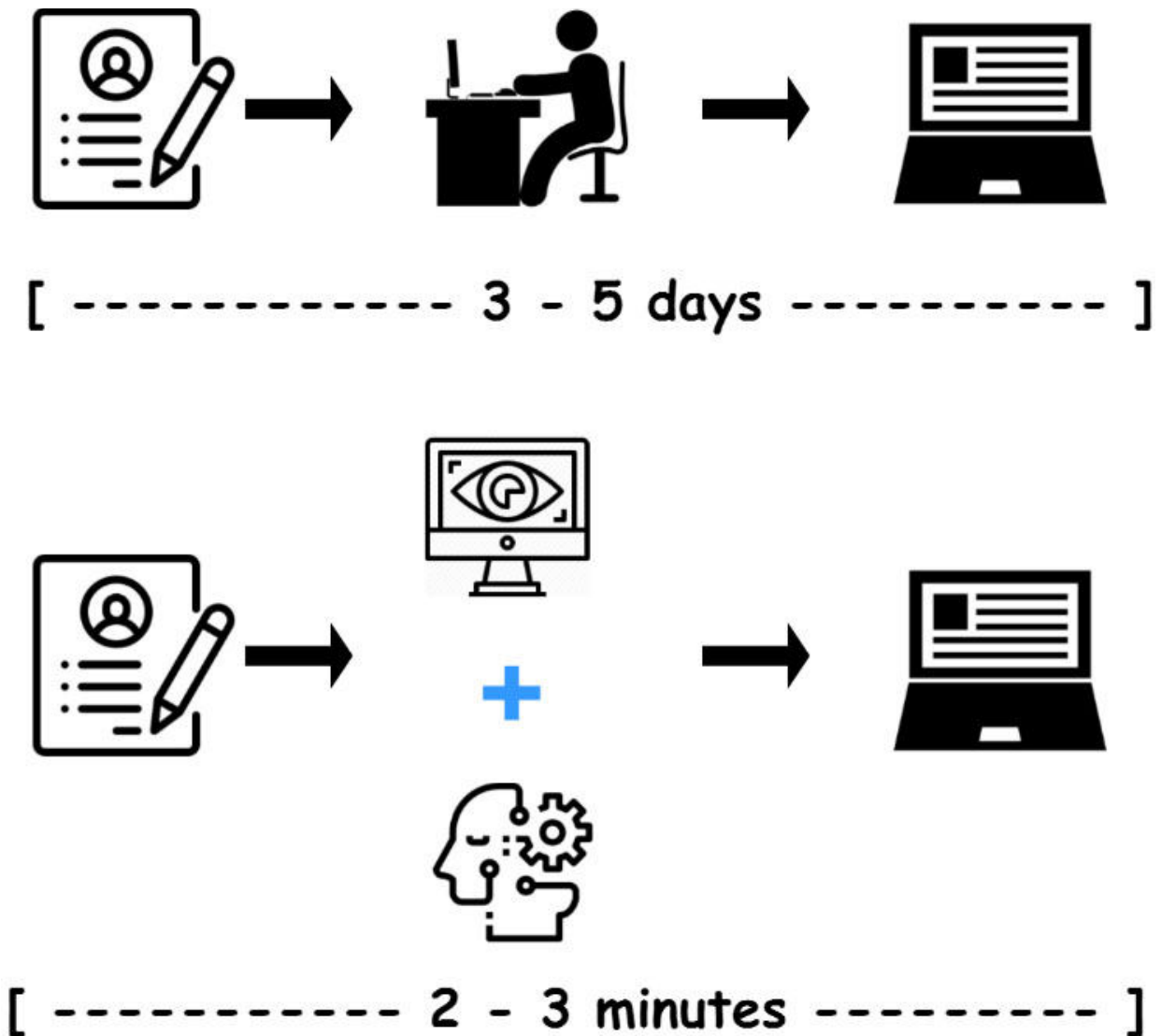


Figure 17.15: FormAssist: End to End Solution for Handwritten Forms using Deep Learning

The application helps businesses to archive the digital data and provide near real-time data for deploying data analytics applications. The algorithm is part of the research team and keeps on updating with the latest developments in deep learning.

[Acknowledgment](#)

This work is supported by PROBYTO Data Science and Consulting Pvt Ltd and is being developed during Data Science Summer Camp 2018.

References

1. *Darmatasia and Mohamad Ivan Fanany*, "Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)"
2. *Jindal and M. Amir*, "Automatic classification of handwritten and printed text in ICR boxes," *Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC*, 2014, pp. 1028– 1032, 2014.
3. *N. Sharma, T. Patnaik, and B. Kumar*, "Recognition for Handwritten English Letters: A Review," vol. 2, no. 7, pp. 318–321, 2013
4. *Dan Claudiu Cires, an and Ueli Meier and Luca Maria Gambardella and Jurgen Schmidhuber*, "Convolutional Neural Network Committees for Handwritten Character Classification," 2011 International Conference on Document Analysis and Recognition, IEEE, 2011
5. *Georgios Vamvakas, Basilis Gatos, Stavros J. Perantonis*, "Handwritten character recognition through two-stage foreground sub-sampling," *Pattern Recognition*, Volume 43, Issue 8, August 2010
6. *Shrey Dutta, Naveen Sankaran, Pramod Sankar K., C.V. Jawahar*, "Robust Recognition of Degraded Documents Using Character N-Grams," IEEE, 2012
7. *Naveen Sankaran and C.V. Jawahar*, "Recognition of Printed Devanagari Text Using BLSTM Neural Network," IEEE, 2012
8. *Yong-Qin Zhang, Yu Ding, Jin-Sheng Xiao, Jiaying Liu, and Zongming Guo*, "Visibility enhancement using an image filtering approach," Zhang et al. *EURASIP Journal on Advances in Signal Processing* 2012
9. *Bhatia*, "Optical Character Recognition Techniques: A Review," *International Journal of Advanced Research in Computer Science and Software Engineering* 4(5), May - 2014, pp. 1219-1223
10. *Anuj Dutt, Aashi Dutt*, "Handwritten Digit Recognition Using Deep Learning," *IJARCET*, Volume 6, Issue 7, July 2017, ISSN: 2278 – 1323
11. *Batuhan Balci, Dan Saadati, Dan Shiferaw*, "Handwritten Text Recognition using Deep Learning."
12. *Gauri Katiyar, Ankita Katiyar, Shabana Mehfooz*, "Off-Line Handwritten Character Recognition System Using Support Vector

Machine," American Journal of Neural Networks and Applications 2017; 3(2): 22-28

- [13.](#) *Nikhil Pai, Vijaykumar S. Kolkure*, "OPTICAL CHARACTER RECOGNITION: AN ENCOMPASSING REVIEW," IJRET, Volume: 04 Issue: 01 | Jan-2015
- [14.](#) *Jason Brownlee*, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," Deep Learning, Machine Learning Mastery
- [15.](#) *Diederik Kingma, Jimmy Ba*, "Adam: A Method for Stochastic Optimization," University of Toronto, 2015 ICLR paper (poster)
- [16.](#) *Alsaad, A.*, 2016. Enhanced root extraction and document classification algorithm for Arabic text (Doctoral dissertation, Brunel University London).

CHAPTER 18

Industry Use Case 2 - People Reporter

Nowadays, we are getting the news very fast through social media, like Facebook; Twitter is the major source of every news. But analyzing that news and evaluating the true value is one of the major problems in real-time. So, in this chapter, we are going to see how that problem can be solved using a machine learning method. From this, we can understand how to use the text analytics to perform the analysis on real-time datasets.

Structure

- Introduction
- Event detection
- Work architecture
- Results
- Conclusion

Objective

After studying this chapter, you should be able to:

- Understand the fundamentals concepts of social media analysis.
- Understand the importance of identifying the credibility of news
- Apply the text analysis to perform real-time analysis.

Abstract

With a total of 4,156,932,140 internet users by 2017, the number of internet users has increased drastically, reaching 54% of the total population and counting. An increase in the total number of users means more user-generated content across several online platforms, which is predominantly in real-time. The user-generated content is being leveraged by applications to

derive insights into customer analyzing, opinion mining, marketing, and for providing niche services like banking in real-time. In recent years, we have also seen a rise in citizen journalism and public posting real-time events on social media channels. Social media has emerged as a supporting player for traditional media as well as powerful stand-alone expression tools for the public, and hence changing the reliance on traditional media for reports and news. Further, the increase in smartphones and better coverage of data networks has shown increased credible news sourced by mainstream media to be from social media. Not only media agencies but the real-time event identification can be used by security departments, disaster management, and others for quick action. The most prominent source of information is the micro-blogging site, Twitter providing geolocation and other features like time, author id, author name, source, link, people's reaction towards that data, etc. and can be easily extracted, stored and analysis using Big Data Tools. Entities extraction in **Natural language Processing (NLP)** is used for identifying the type of event and proceeds further. The fundamental goal of our work is to limit the spread of falsehood by halting the proliferation of fake news in the system. This helps us in taking the lead in collecting information on certain events ahead of local media platforms. For example, when an earthquake occurs, people make many posts related to the earthquake, which enables detection of earthquake occurrence promptly. Our model delivers such notifications of such events much faster than the announcements of other media sources. In this paper, we have utilized the information from the social platforms in real-time based upon some keywords and geolocation and visualized it with powerful BI tools. The continuous monitoring helps us analyzing the events occurring in the respective geolocation and defining its credibility. The credibility of such an event is detected with the help of the credit score factors developed considering multiple factors, including temporal and spatial features of the reported content.

Introduction

Internet-based life has picked up a great deal of consideration today. According to *Kaplan and Haenlain* [1], social media is an application that keeps running on the Internet that enables its clients to make and trade content. Magazines, websites, microblogging, wikis, social networks, and video sharing sites are considered as social media. The ubiquity of social

media is expanding quickly because of the spread of the Internet and the improvement of cell phones. Individuals can access and utilize web-based life at any place and anytime. Many individuals, generally teenagers and youthful grown-ups, joined via social media and utilized them every day. Microblogging and social media sites, like Facebook and Twitter, have expanded in popularity among the others sort of online networking. Facebook, which is considered as the biggest social networking sites right now, reported that it has one billion members around the world, as in October 2012 [2]. People use social media not just for communicating with their friends, but also for many other things. For example, people could use Twitter as a source of information. News organizations could use Twitter to spread more information because people tend to broadcast anything that they think is matter. Even, sometimes, people could share news ahead of newswire [3]. People could also use social media to express their opinion towards something [4, 5].

The new era of the Internet has conveyed a transformation with it. A period where people felt the intensity of articulation. A time where each communicated word holds the possibility to have an effect. Social media particularly had a colossal part to play in this strengthening. With each passing day, the time spent on web-based networking media by people is expanding, to an ever-increasing extent. In April 2018, Facebook had 2.2 billion clients for each month and Twitter 330 million active users, for every month, these numbers just suggest the extent of these online platforms. The data generated from these networking platforms are of extraordinary esteem and can be gathered effortlessly from web crawlers and public APIs.

Data shared by social media users could be used for various purposes like news detection, security, etc. An update by users on these platforms can be checked and hailed as seed for a potential future occasion or present developing event, i.e., detecting news. Concerning news detection proof, there are two points to it, breaking news and the other being trending news. This paper considers both the segments in an enduring movement. The need being an acknowledgment of the news as it happens took after by examination of as it advances. A legitimacy score is delivered after some time to decide the confirmation of the news. For this situation, Twitter and Facebook could be utilized as emergency notification tools about a catastrophe that is going ahead. At the point when the tsunami hit Japan in

2011, Twitter was utilized as a specialized device when alternate methods for correspondence are down [6].

Event detection

Dou et al. [7] outlined event detection is said to spot the primary story on topics of interest through perpetually monitor news streams. Petrovic et al. [3] outlined the goal of event detection is to spot the primary story to debate a selected event that happened at a specific time and place using social media as a source of event detection. There is literally another field of analysis associated with an event. Dou et al. [7] declared that, together with event detection, there are event trailing, event summarization, and event association. Event trailing connected with the event of some events over time. Event summarization produces a summary of the event from given knowledge. Meanwhile, the event association discovers the connection between one event towards another. These three topics are representing **Topic Detection and Tracking (TDT)** fields.

There are 3 reasons why event detection in social media streams is additional challenging: short and howling contents, various and quick changing topics, and enormous data volumes [3, 8]. Besides that reason, text generated from social media streams typically short and use informal literary genre. Totally different approaches must be preferred to handle this type of data. Dou et al. [7] outlined two subtasks associated with event detection: retrospective and on-line event detection. Retrospective event detection detects events from pre-collected sources, whereas online event detection detects events from real-time sources [3, 9]. There are some approaches that use pre-collected data from social media. There is some corpus available to train and test these strategies. Twitter was the source for most of the available because of its popularity as social media sites [10]. A total of 97 million tweets in one corpus come from the *Edinburgh* Twitter corpus [11].

Various event detection approaches are discussed in detail in the paper Event Detection in Social Media: a Survey [12]. Different approaches also developed to process social media streams. *Baldwin et al.* [13] querying data stream with specified keywords directly. Non-English messages then being filtered out and then normalized. After being normalized, the location of the events is detected. There are some problems that arise, such as the little proportions of geotagging usage, user registered location, and no assurance

of the quality of user locations. One approach is to predict the geolocation of the message along with the probabilistic indications of the quality. Detecting events from social media could show great benefits. *Sakaki et al.* [14] showed how updates from Twitter could be used to detect an earthquake and even predict the center of the earthquake in real-time. Another event, such as the death of Michael Jackson, could also detect from social media [15]. In general, there are many kinds of events or trends that could be detected from social media. Researchers have studied disaster, traffic, outbreak, and news detection in social media. The summary of past researches covered in this paper is shown in [Table 18.1](#):

Event	Author	Method
Disaster	<i>Sakki et al. (2010)</i>	The temporal and spatial model
Disaster	<i>Abel et al. (2012)</i>	NER, classification, filtering
Disaster	<i>Abel et al. (2012)</i>	Faceted search and analytics
Disaster	<i>Terpstra et al. (2012)</i>	Geographical display, message content filters, tweet type filters
Disaster	<i>Adam et al. (2012)</i>	Semantic reasoning, event classification / grouping
Traffic	<i>Kosala et al. (2012)</i>	Keyword analysis, abbreviation analysis, location and traffic condition search
Outbreak	<i>Ritterman et al. (2009)</i>	Prediction market: SVM (internal) and n-gram (external)
Outbreak	<i>Achrekar et al. (2011)</i>	Group average clustering (GAC) and incremental clustering (INCR)
News	<i>Petrovic et al. (2010)</i>	Locality sensitivity hashing
News	<i>Osborne et al. (2012)</i>	Streaming model and events cross-checking
News	<i>Ishikawa et al. (2012)</i>	Clustering, burst detection

Table 18.1: Summary of event detection researchers [12].

While testing our model, we took under consideration a plane crash in suburb *Ghatkopar*, Mumbai, on 28th June 2018, claiming multiple lives. The aircraft got airborne from *Juhu Aerodrome* at approx. 12:20 hours IST on June 28th June 2018 with 2 pilots employed by *U.Y. Aviation Private Limited* and 2 technical personnel of *Indamer Aviation Private Limited (MRO)* and flew for approximately 40 minutes as per the Air Test profile.

Contact with Mumbai ATC was reportedly lost at around approx. 13:00 hours IST [16]. At 13:36 hours IST, Media houses came to know about the crash and started publishing about the tragic incident. The model proposed in this paper explains how the time gap between the time delay between the event and its media publication can be decreased. The above-described example took place around 13:25 hours IST, and while running, our model helped us to discover that the first information related to this accident was registered at 13:26 hours IST, which is hardly a minute difference with the accident.

Work architecture

Twitter was considered as the source of study in this case. Data extraction was carried out using APIs. Twitter allows users to interact with its data; that is, Tweets and other attributes of the tweets using APIs. Using server-side scripting language, a request was made, and data extraction was done. In this model, we propose a dashboard where a user can access using a web portal. The keywords and geolocation provided by the user act as a filter to proceed for the data extraction. This extraction is stored as indices and is analyzed continuously and later visualized according to the keyword and geo-location. This leads to the detection of cases that started escalating. All the features which are extracted with the help of the APIs are used for decision making and setting a parameter for decisiveness. The result is displayed on the dashboard, along with all the parameter values and contender news.

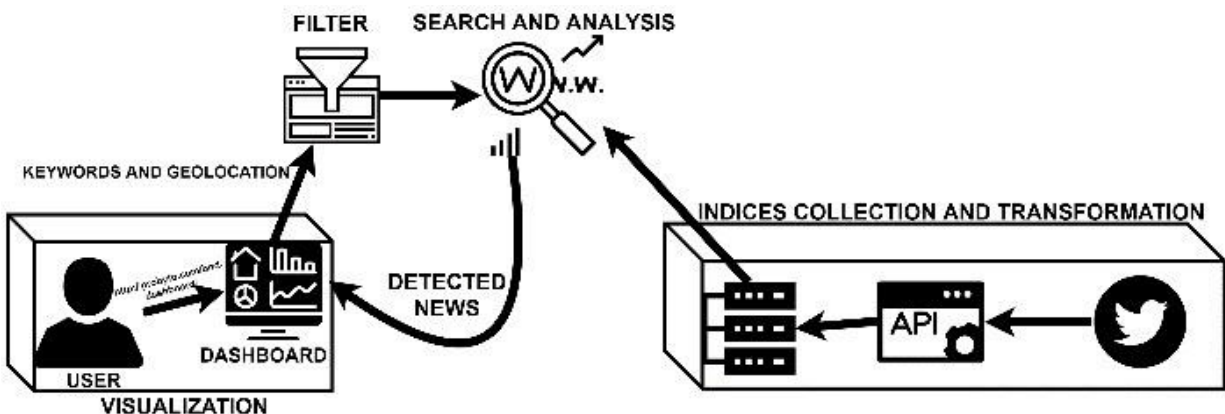


Figure 18.1: Work architecture of PeopleReporter

The work architecture is based on *Real-time Event Detection of Events using Twitter*. The event in the flow chart ([Figure 18.2](#)) defines that the tweets

which are in relation to certain predefined keywords, very specifically generic news keywords. These filtered tweets can come from people, local news agencies, and mainstream news media. These tweets are extracted using Twitter APIs. These APIs extract not only the content of the tweets but also the other attributes, such as number of likes, number of retweets, number of comments, number of followers, verified user or not, and many more, which assists in determining the credibility of news. All these attributes are stored along with tweets. Users from the web portal enter keywords or geolocation or both and filters out all the desired tweets. These sorted out tweets are visualized and analyzed for detection of topics which are just in and are being talked by people:

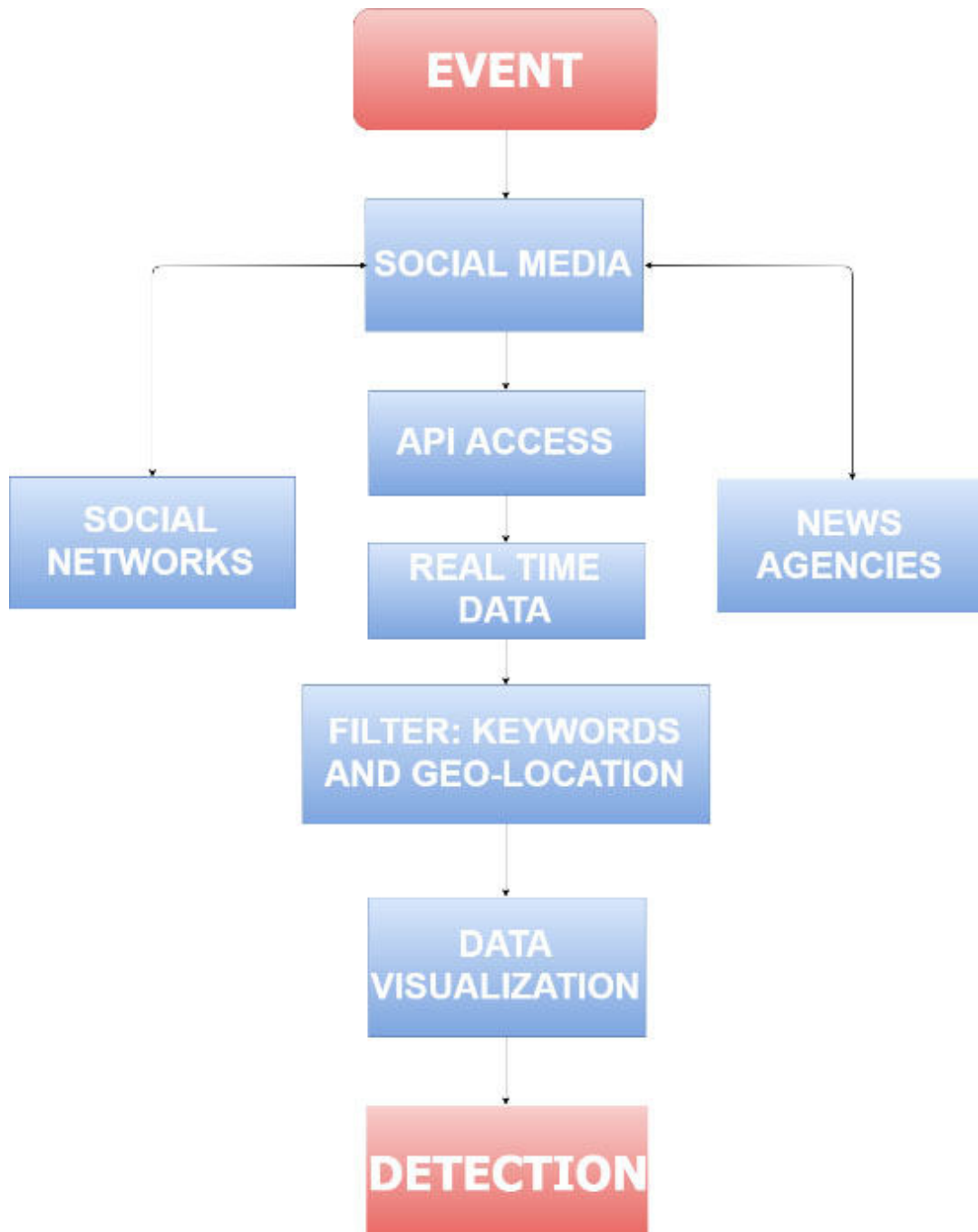


Figure 18.2: Flowchart of the end-to-end solution of the PeopleReporter

Using Twitter APIs, all the tweets and content are taken in real-time, and then the event processing pipeline is used. The event processing pipeline has three stages: Input, Filters, and Output. Input is a collection of all the tweets where its attributes are filtered. Only these filtered outputs can pass for analysis. These input and output files are in JSON format and are stored in indices. This is a real-time data handling, and the data increases over time,

which calls for a search engine which is distributed, fast and can handle such ever-increasing data.

Results

In this paper, we present three examples of real events occurring within the country. [Table 18.2](#) gives an overview of the selected events. These three are considered with respect to the different categories and events happening during the third week of May 2018:

#	Category of event	Title of the event	Reporting date	Reporting region
1	Outbreak	Nipah virus outbreak in Kerala	21/05/2018	Kerala
2	Live	CSK enters the final of IPL 2018	22/05/2018	Chennai
3	News	Oneplus 6 Launched in India	20/05/2018	Bengaluru

Table 18.2: List of events

Nipah virus outbreak in Kerala

On 19 May 2018, three deaths due to Nipah virus infection were reported [\[17\]](#) in *Kozhikode district, Kerala state, India*. As of 28 May 2018, and since the beginning of the outbreak, because of further investigations and contact tracing, 15 people [\[17\]](#) have tested positive for NiV in *Kozhikode* and *Malappuram* districts, *Keralastate*. In this paper, we take this outbreak as an event and utilize the tweets related to it. [Figure 18.3](#) shows the frequency of tweets starting from 21 May 2018 (00:25:43 hours) to 23 May 2018 (07:09:04 hours). The total tweets count was 16242. On the afternoon of 21 May, it is seen that the frequency of tweets reaches maximum to ~140 (yellow), which reflects the detection of an event. But the frequency gets doubled (red) on the next day morning (22/05/2018), which defines the event to be breaking news. This is when people start talking more about the outbreak of the Nipah virus in Kerala. The purple circle shows the decaying of the event gradually:

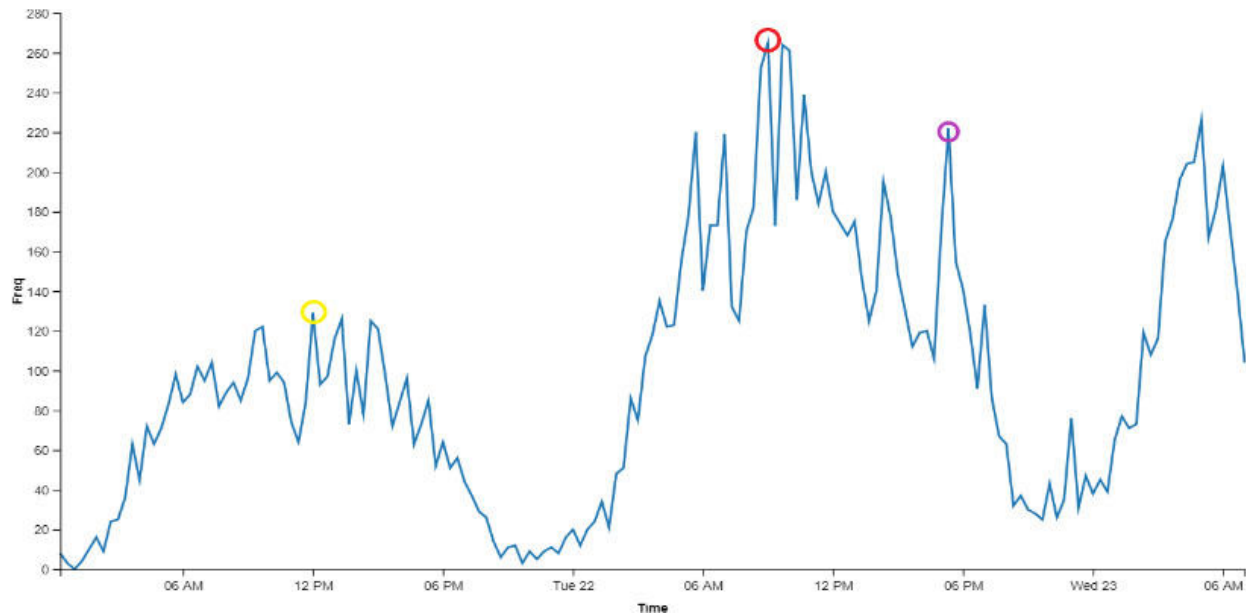


Figure 18.3: the Increasing magnitude of peaks with increasing credibility

CSK enters the final of IPL 2018:

Brilliant innings from Faf du Plessis (67*) [18] helped **Chennai Super Kings (CSK)** beat **SunRisers Hyderabad (SRH)** by two wickets in the first qualifier of the **Indian Premier League (IPL)** 2018 at the *Wankhede Stadium* to enter the IPL 2018 final. Needing 140 runs to win, *Chennai* chased down the target with five balls to spare. [Figure 18.4](#) shows the frequency of tweets starting from 22 May 2018 (01:57:31 hours) to 23 May 2018 (10:56:00 hours). The total tweets count was 495. In the evening (18:00 hours) of 22 May, it is seen that the frequency of tweets reaches maximum to ~35 (yellow), which reflects the starting of the match between CSK and SRH. The frequency almost gets tripled (red) by 23:00 hours when CSK beats SRH. This live event becomes breaking news at this point, and people start talking more about seeing CSK in the finals. The purple circle shows the decaying of the discussions gradually:

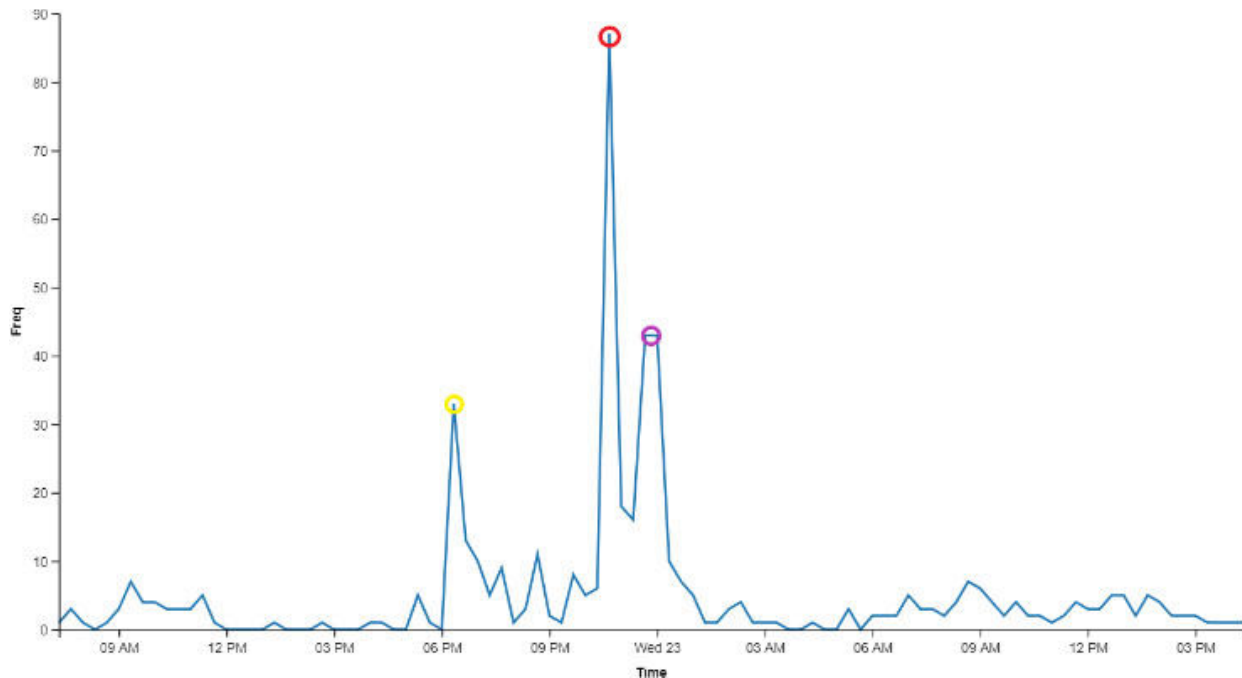


Figure 18.4: Single peak to represent instantaneous joy and decay

OnePlus 6 launched in India

Chinese smartphone player OnePlus launched its next flagship OnePlus 6 in India on 17 May 2018. In *India*, OnePlus 6 was launched in *Mumbai* [19], and the *OnePlus Community* was the first to try out the device at the experience zone at the launch venue. The device will be available for pre-bookings for *Amazon Prime Members* from May 21. The company hosted a live streaming of OnePlus 6 India launch event on its YouTube channel. [Figure 18.5](#) shows the frequency of tweets starting from 20 May 2018 (00:01:34 hours) to 21 May 2018 (23:53:17 hours). The total tweets count was 13424. Usually, the rumors of the most awaited phones, such as OnePlus 6, starts even before a month or two. However, we started analyzing the event, a day before its launches. Descent number of discussions were noted on Twitter on 20 May 2018, reaching a tweet frequency of ~100. On the day of launch, by 05:00 hours, there was a high increase in the frequency (yellow). This shows the excitement of people waiting for the launch. At 10:00 hours, there was a significant increase in the frequency (red) again, which defines the product getting launched in India. The launch event becomes breaking news with all its features listed and along with its price for the customers. The frequency almost increases by 9 times (red) as

compared to the earlier day. The purple circle shows the decaying of the event gradually, and the rumors, discussion and excitements for their OnePlus 6 decreases eventually:

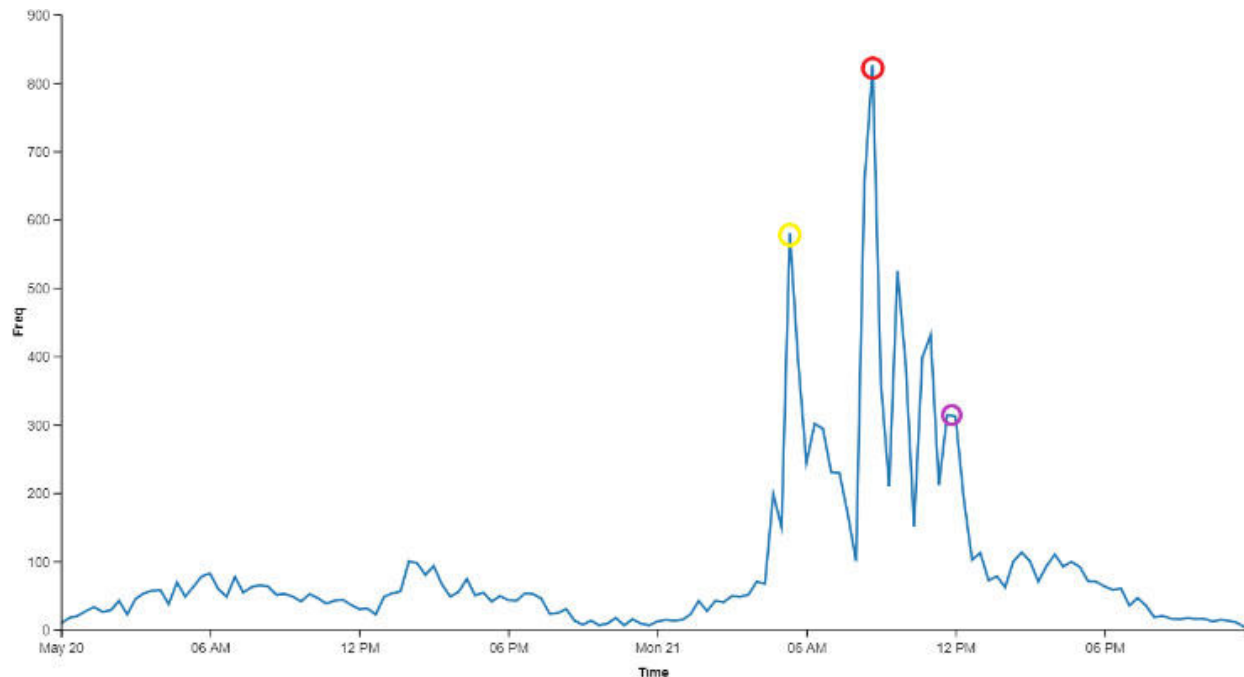


Figure 18.5: Multi peaks as people talks about product features

Conclusion

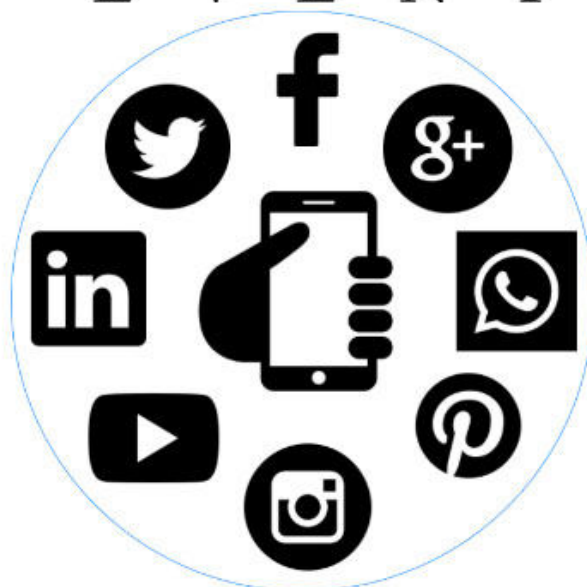
Social media and, in general, the web have become a precious source of information in real-time, contributed by individuals. This new source has created opportunities for media companies, governments, and disaster management authorities to leverage People's reporting or information they broadcast on the web. This paper shows possibilities with a simple metric as frequency over the period can provide credible news information. Further research is being done by the team to develop an ML trained gradient-based credibility threshold to break the news before mainstream media. The future work will also focus on spatial and temporal features to measure the credibility of news.



THE DATA SCIENCE COMPANY

**Breaking
News**

Detection



SOCIAL MEDIA



MAIN STREAM MEDIA



www.probyto.com

Office : TIC No. 5, TECHNOLOGY COMPLEX
INDIAN INSTITUTE OF TECHNOLOGY
GUWAHATI - 781039
contact@probyto.com; +91-844-844-9368

Figure 18.6: Product Design for Social Media monitoring

The working prototype has a powerful engine to continuously scan the web and social media to detect news reported in pre-defined areas. The datastore created and validation from mainstream media sources is a valuable source for future research in the domain.

Acknowledgment

This work is supported by PROBYTO Data Science and Consulting Pvt Ltd and is being developed during Data Science Summer Camp 2018.

References

1. A. Kaplan and M. Haenlaine, "Users of the world, unite! The challenges and opportunities of Social Media," *Business Horizons*, vol. 53, no. 1, pp. 59-68, January-February, 2010.
2. M. Zuckerberg, "Facebook," 04 October 2012. [Online]. Available: <http://newsroom.fb.com/News/457/One-Billion-People-on-Facebook>. [Accessed 6 December, 2012].
3. S. Petrovic and e. al., "Streaming first story detection with application to Twitter," in *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.
4. A. Agarwal and e. al., "Sentiment Analysis of Twitter Data," in *LSM '11 Proceedings of the Workshop on Languages in Social Media*, 2011.
5. E. Kouloumpis and e. al., "Twitter Sentiment Analysis: The Good the Bad and the OMG!" in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.
6. C. Taylor, March 2011. [Online]. Available: <http://mashable.com/2011/03/10/japan-tsunami/>.
7. W. Dou and e. al., "Event Detection in Social Media Data," in *IEEE VisWeek Workshop on Interactive Visual Text Analytics – Task Driven Analytics of Social Media Content*, 2012.
8. C. Li, A. Sun, and A. Datta, "Twevent: Segment-based Event Detection from Tweets," in *The 21st ACM International Conference on Information and Knowledge Management*, New York, 2012.

- [9.](#) *M. Osborne, S. Petrovic, R. McCreadie, C. Macdonald, and I. Ounis*, "Bieber no more: First Story Detection using Twitter and Wikipedia," in In Proceedings SIGIR 2012 Workshop on Timeaware Information Access, Portland, 2012.
- [10.](#) *A. Pak and P. Paroubek*, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," in Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010.
- [11.](#) *S. Petrovic, M. Osborne and V. Lavrenko*, "The Edinburgh Twitter Corpus," in Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media, Stroudsburg, 2010.
- [12.](#) *N. Arif, W. Edi*, "Event Detection in Social Media: a Survey"
- [13.](#) *T. Baldwin, P. Cook, B. Han, A. Harwood, S. Karunasekera, and M. Moshtaghi*, "A Support Platform for Event Detection Using Social Intelligence," in Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, 2012.
- [14.](#) *T. Sakaki, M. Okazaki, and Y. Matsuo*, "Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors," in Proceedings of the 19th International Conference on World Wide Web, 2010.
- [15.](#) *M. Osborne and e. al.*, "Bieber no more: First Story Detection using Twitter and Wikipedia," in Proceedings of the SIGIR Workshop in Time-aware Information Access, 2012.
- [16.](#) <https://www.republicworld.com/india-news/city-news/mumbai-plane-crash-u-dot-y-aviation-issues-official-release>
- [17.](#) World Health Organization (WHO), Emergencies preparedness, response, Disease Outbreak News (DONs); 'Nipah virus – India'
- [18.](#) *Joy Tirkey*, NDTV, Sports Home, IPL 2018 – News, 'Du Plessos Shines as CSK Beat SRH to enter Tournament Final.'
- [19.](#) India.com Business Desk, 'OnePlus 6, launched in India at Rs 34,999.'

CHAPTER 19

Data Science Learning Resources

Data science is a vast interdisciplinary field and requires continued learning to stay relevant with the latest happenings and technology swings. Sitting at the junction of research, technology, and business, it's a highly dynamic and difficult field to stay abreast of. We already discussed as an industry we are learning what the skillsets and resources required to deliver a data-driven product or service are. As you may appreciate by the width of this book as well, it is impractical to find a single person who is an expert in all areas and remain to stay expert.

The industry has recognized this fact and has started grooming talent in specific sub-fields of data science and creating career paths in the industry for them. You must have stumbled upon the data function of large organizations hiring data scientists, data engineers, DevOps, researchers, analysts, visualization experts, and software developers, and so on. and all of these roles are now required to bring an idea into business use. In many cases, the internal teams are also re-organizing. In such an environment, continuous learning is of utmost importance. In this chapter, we will make the reader aware of the type of resources they can engage with and keep learning.

Structure

- Books
- Online courses
- Competitions
- Blogs and magazines
- University courses
- Conferences and events
- Meet-ups and interested groups

- YouTube channels and Podcasts
- Analytic reports and white papers
- Talk to people

Objective

After studying this chapter, you should be able to:

- Find resources to explore more on data science and related works.
- Know more about the industry trends and various opportunities to learn from major resources about the data science field.

Note: The author does not endorse any resource or encourage or soliciting to use that resource. These are just good resources as per a general understanding amount the authors.

Books

Books are very good sources when you want to build the concepts and ideas from scratch. Books provide a structured approach to your learning and act as quick references for the future. There are numerous books published in data science and its sub-domains. Some of them are listed below:

- *The Hundred-Page Machine Learning Book* - by Andriy Burkov
- *Introduction to Statistical Learning* - by Gareth James
- *Practical Statistics for Data Scientists* by Peter Bruce
- *Introduction to Machine Learning with Python: A Guide for Data Scientists* by Andreas Muller
- *An Introduction to Probability Theory and its Applications* – by William Feller

Online courses

Online courses are very popular among corporate learners as they provide the flexibility of time and usually delivered through video lectures. This makes the courses available 24 hours and self-paced as per the capability of learner. Some good courses:

- *Machine Learning* – by Andrew Ng (<https://www.coursera.org/learn/machine-learning>)
- *Introduction to Python for Data Science* – by Datacamp (<https://www.datacamp.com/courses/intro-to-python-for-data-science>)
- *Introduction to Computational Thinking and Data Science* – by MIT (<https://www.edx.org/course/6-00-2x-introduction-to-computational-thinking-and-data-science-3>)
- *Python for Data Science and Machine Learning Bootcamp* – by Udemy (<https://www.udemy.com/course/python-for-data-science-and-machine-learning-bootcamp/>)
- *MicroMasters® Program in Statistics and Data Science*– by MIT (<https://www.edx.org/micromasters/mitx-statistics-and-data-science>)

Competitions

There are many platforms that source good quality problem statements from Industry and roll them out for crowd-solving them as a competition. These competitions are a good place to see how other participants solve the same problem and share their results in a very healthy and competitive manner.

Here we are going to list out top 5 competitions in the data science field:

- Kaggle (<https://www.kaggle.com/>)
- Driven Data (<https://www.drivendata.org/>)
- DataHack (<https://datahack.analyticsvidhya.com/>)
- HackerRank(<https://www.hackerrank.com/categories/ai/machine-learning>)
- Innocentive (<https://www.innocentive.com/ar/challenge/browse>)

Blogs and magazines

Blogs and magazines provide a good source of individuals' experience with different tools, processes, challenges, and other evolving ideas. The blogs are usually reflective of near real-time trends in data science.

Top 5 blogs and magazines participating actively in the development of data science are listed below:

- Towards Data Science (<https://towardsdatascience.com/>)
- Data Science Central (<https://www.datasciencecentral.com/>)
- KDnuggets (<https://www.kdnuggets.com/>)
- Analytics Vidhya (<https://www.analyticsvidhya.com/>)
- Data Science Dream Job (<https://www.datasciencedreamjob.com/blog>)

University courses

If you have very little background with statistics or programming, maybe having a good course from university helps the most to build the foundational concepts and understanding of what to expect in the industry. There are a lot of data science courses in different capacities running across the institutions. Some are given below:

- Data Science: Machine Learning – Harvard University
- MicroMasters Program in Statistics and Data Science - Massachusetts Institute of Technology
- Master of Information and Data Science – The University of California, Berkeley
- Foundations for Data Science – Stanford University
- Master of Computer Science in Data Science – University of Illinois

Conferences and events

There has been a rise in data science and its sub-field specific conferences and events across the world. This shows the growing interest in the business to adopt data science and create an executive-level network by means of events and conferences.

Top 5 conferences most frequently in the data science field are listed below:

- Strata Data Conference – by O'Reilly
- Open Data Science Conference – by Cambridge

- Kaggle Days – by Kaggle
- Machine Learning Developer Summit – by Analytics Magazine
- Data Hack Summit – by Analytics Vidhya

Meet-ups and interest groups

Meet-ups are also events where different groups of similar interest professionals meet and discuss ideas. They can be offline at places or gathered over some open source projects as well.

- Data Science Network - Bangalore
- Hyderabad Data Science Group - Hyderabad
- Practical Data Science - Bangalore
- Datagiri – San Francisco
- Deep Learning Bangalore - Bangalore

YouTube channels and Podcasts

Sometimes you need an individual explaining concepts and delivering through multi-media mode. YouTube and podcast are such sources which provide you bite-sized information on the targeted topics that you want to learn:

- Statistics in Machine Learning - <https://www.youtube.com/playlist?list=PLZoTAELRMXVMhVyr3Ri9IQ-t5QPBtxzJO>
- Data Science by Arpan Gupta IIT,Roorkee - https://www.youtube.com/channel/UCjrGJITO_pggWmjgPvUiHFA/videos
- Intellipaat - <https://www.youtube.com/channel/UCCktnahuRFYIBtNnKT5IYyg>
- Machine Learning Tutorial in Python - Edureka https://www.youtube.com/playlist?list=PL9ooVrP1hQOHUfd-g8GUpKI3hHOwM_9Dn
- 365 Data Science - <https://www.youtube.com/channel/UCEBpSZhI1X8WaP->

Analytic reports and white paper

Active organizations in the field of data science, as well as consulting companies, regularly publish reports and white papers which contain valuable information from a business perspective of how data science is helping them achieve better results or as a market trend.

- Big Data and Analytics Hub - <https://www.ibmbigdatahub.com/whitepapers>
- Inside BigData - <https://insidebigdata.com/>
- Analytics Insight - <https://www.analyticsinsight.net/>
- Data Science Foundation - <https://datascience.foundation/>
- Data Science Journal - <https://datascience.codata.org/>

Talk to people

Talking to people from the same field or other fields is always rewarding where you get to know new things and different perspectives. A data scientist understands these perspectives and builds impact data products for society and organizations.

Conclusion

In this chapter, we have seen several resources based on the different variety of sources, such as a book, online communities, conferences are a few examples. But this not only the resources to learn about data science. These are some of the recommended learning resources to explore more.

CHAPTER 20

Do It Your Self Challenges

This chapter lists out 5 real-life industry-grade challenges for the readers to attempt to solve. Some resources are also provided to get started on the challenges. The reader is expected to start learning from the challenges by doing hands-on problem-solving.

Structure

- Challenge overview
- Challenge statement
- Target users
- Resources
- IP source

Objectives

After studying this chapter, you should be able to:

- Understand the real use cases and their challenges.
- Able to give some useful and effective solutions to the challenges.
- Identify resources to solve them.

DIY challenge 1 – Analyzing the pathological slide for blood analysis

This challenge is from the healthcare industry, where a shortage of lab professionals causes a lot of delay in essential test results in semi-urban and rural areas where the blood slides are manually analyzed.

Challenge overview

Medical diagnosis is the very first step and necessary prerequisites for recommending any medicines and planning further treatment. This process consists of an analysis of the multimedia data of X-rays, ECGs, pathology tests, ultrasounds, and other required tests. But that is not easy to perform for any emergency cases with high efficiency. By involving the Artificial Intelligence technologies, we can assist analyze on the multimedia data quickly to make decisions smartly. Pathology test based medical diagnosis is one of the key areas recently gets the attention towards the usage of AI application development. In this test, we will perform the analysis like counting of WBC and RBC cells, thickness, foreign body detection, shape, and many more. Especially this analysis takes the input as images. Deep learning algorithm-based image analysis techniques show a greater impact on these tasks very effectively.

We can take the use case of analyzing a blood sample to detect the presence of any unwanted foreign body in the blood. Normally, this a standard process performed by a pathologist with the tiny help eye-piece of a microscope to analyze slides manually. But there is a high possibility of error-prone during the analysis, and also it is a complicated process. To assist this process, the involvement of AI has two main components: capturing the slide content as a high-resolution image and an AI algorithm to help the pathologist to analyze the corresponding output. The captured image will be stored in the cloud environment. Through the API calls, we perform the analysis on those images, and we can get the report automatically.

Challenge statement

This challenge is coming under Image analysis techniques. So, computer vision is the perfect technique to adopt this challenge. In short, we can state the challenge statement as below:

Can you build a computer vision-based application to analyze pathological blood samples?

Target users

Pathology labs, clinics, and healthcare service providers.

Resources

The following are the key resources the learning can explore more about this challenge:

- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3996871/>
- <http://www.virtualpathology.leeds.ac.uk/research/analysis/>
- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5556681/>
- <https://blog.athelas.com/classifying-white-blood-cells-with-convolutional-neural-networks-2ca6da239331>

IP source

Probyto AI Lab (<https://ailab.probyto.com>)

DIY challenge 2 – IoT based weather monitoring system

Deterioration weather in big cities and in urban clusters is a problem to tackle for healthy living. This challenge comes from the renewed interest in monitoring the weather to manage the industrial activities to maintain healthy air.

Challenge overview

Nowadays, climate changes happen very fast and drastically. So, the researchers get attention towards forecasting the weather changes because it directly affects humans' life and livelihood. Collecting the temporal dynamics of weather changes is a very important process. From the perspective of any industry, if any certain disaster comes, they must monitor the weather very closely.

IoT based system with the support of machine learning algorithms can be effective in solving this kind of problem. IoT system can be developed by embedding a system to design a weather monitor system with the capability of monitoring the parameters of climate changes in a field or an industry remotely. This system may have components like sensory used to detect

light, temperature, humidity, and many more, and to transmit the data, it needs the module to store the data in clouds.

Challenge statement

Can you build a machine learning application along connected to an IoT weather monitoring sensors to monitor and recommend air quality parameters?

Target Users

Agriculture, food processing, utilities, cold storage, weather/geospatial forecasting, disaster management agencies.

Resources

The following are the key resources the learning can explore more about this challenge:

- <http://nevonprojects.com/iot-weather-reporting-system/>
- <https://www.ijarcce.com/upload/2016/september-16/IJARCCE%2066.pdf>
- <http://ijesc.org/upload/c1ad40d536ccca051ff4af59252ab9d6.Smart%20City%20IoT%20Based%20Weather%20Monitoring%20System.pdf>

IP source

Probyto AI Lab (<https://ailab.probyto.com>)

DIY challenge 3 – Facial image-based BMI calculator

Knowing the human body and its functions from secondary sources have been around for centuries. With the advent of technology, our observable features can be used to understand the health and functioning of different parts of the body. **Body Mass Index (BMI)** is an important metric to

monitor the risk of health lifestyle-related disease; this challenge comes from this domain.

Challenge overview

Research in physiology has shown that facial expressions provide a lot of cues to human psychological and physical features. In today's world, where selfie has become common among millennial, a lot of use-cases in recruitment and user tagging are being built. Life Insurance companies require BMI for underwriting life policies, as they are looking for automating underwriting process they want to have approximate BMI by image.

Challenge statement

Can you create an application that allowsthe user to take a selfie and report the BMI?

Target users

Life insurance, e-commerce, retail, banking, and healthcare.

Resources

The following are the key resources the learning can explore more about this challenge:

- <https://arxiv.org/pdf/1703.03156.pdf>
- <https://www.sciencedirect.com/science/article/pii/S0262885613000462>
- http://www.fasebj.org/doi/abs/10.1096/fasebj.31.1_supplement.955.11
- <http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0169336&type=printable>

IP source

Probyto AI Lab (<https://ailab.probyto.com>)

DIY challenge 4 – Chatbot assistant for Tourism in North East

Chatbots are now ubiquitous. They allow a conversation with a machine in natural language. This has opened the delivery of useful applications to even naïve users as they work on normal spoken language. This challenge finds the opportunity for the same in the tourism domain.

Challenge overview

Visiting beautiful places is the aim of any tourist. During the visit, they want to know more about the places. The tourism development department has the responsibility to enable all the facilities to the tourist. For example, if they want to stay for a night, they must know the availability of hotels around that place. Similarly, related offerings to the tourist must be shown to them. This is common for any travel agency, hotel booking services, and other businesses around the place.

In order to help those situations, we can use a chatbot based digital service to interact the human beings lively and get understanding their expectations, answers to their questions. Consider, product purchase enablement, will help them to get the most precious products. This can be done by providing some rule-based AI systems to converse with them. This is known as chatbots. It can converse with real people via mobile messaging apps such as Telegram and Facebook Messenger or web platforms directly answer their questions.

Challenge statement

Can you develop a chatbot that can help tourists know more about Assam and North East?

Target users

Travel agencies, hotel booking services, and other businesses built around travel and tourism

Resources

The following are the key resources the learning can explore more about this challenge:

- <https://chatbotsmagazine.com/travel-chatbot-how-chatbots-can-help-city-tourism-a2f122c0896d>
- https://medium.com/@onlim_com/chatbots-alexa-co-in-the-tourism-industry-8f8fe0d95662
- https://blogs.msdn.microsoft.com/uk_faculty_connection/2017/09/08/how-to-build-a-chat-bot-using-azure-bot-service-and-train-it-with-luis/

IP source

Probyto AI Lab (<https://ailab.probyto.com>)

DIY challenge 5 – Assaying and grading of fruits for e-procurement

This challenge comes from the agriculture domain, where assaying and grading are still a manual process in mandis. The assay and grading are important to get the real price of a commodity and help farmers and quality testers to have transparency.

Challenge overview

Across India creating a unified national market for processing agriculture commodities can be done by **Electronic National Agriculture Marget (eNAM)**. It is an online based virtual market with the back-end support of mandi (a physical market where directly agriculture products will present). This type of trading can be done across the pan-India. The examination of agricultural produce at the market level is of extreme importance to improve and enhance the marketability of the product and to enable the farmers to realize and get the price appropriate to the quality of their agricultural produce.

A quick analysis of the quality of the products handled by mandis is the essential solution required. Because mandis handle the large volume of products (that is, lots), arrival and smaller lots also present there.

Challenge statement

Can you develop an application based on IoT (Computer Vision) for quick grading and assaying solution for fruits (for example, Apple or orange), which can also be connected to the internet to increase the efficiency of the agricultural chain?

Target users

Farmer, procurement mandis, storage houses, and food companies.

Resources

The following are the key resources the learning can explore more about this challenge:

- <https://www.sciencedirect.com/science/article/pii/S2214317316300385>
- <https://pdfs.semanticscholar.org/4aa0/a0cdfe036b512b9c6a0e9f34c7f47382a27d.pdf>
- <http://pubs.ub.ro/dwnl.php?id=CSCC6201601V01S01A0008>
- <http://tmu.ac.in/college-of-computing-sciences-and-it/wp-content/uploads/sites/17/2016/10/CCSIT114.pdf>

IP source

Probyto AI lab (<https://ailab.probyto.com>)

Conclusion

In this chapter, we have provided five real industry challenges to solve. Each problem statement has an overview and also references to follow. By solving these problems, the reader will have a good hands-on experience in data science/analytics. In the next chapter, we will provide a data science assessment, which tests your aptitude, technology, programming, algorithms, and many other areas.

CHAPTER 21

Qs for DS Assessment

The selection process for data science job roles is very rigorous and goes through multiple rounds. A Data Science assessment exam is a balanced assessment which tests your aptitude, technology, programming, algorithms, and many other areas. To help readers get a realistic assessment of their own learning and a standard data science assessment, Probyto is open-sourcing its recruitment test to the readers. In this chapter, we will showcase the various probable questions from the previous chapters.

Structure

- Data Science Overview
- Mathematics Essentials
- Statistics Essentials
- Exploratory Data Analysis
- Data Preprocessing
- Feature Engineering
- Machine Learning Algorithms
- Productionizing Machine Learning Models
- Data Flows in Enterprises
- Introduction to Databases
- Introduction to Big Data-driven
- DevOps for Data Science
- Introduction to Cloud Computing
- Deploy Model to Cloud
- Introduction to Business Intelligence
- Data Visualization Tools

Objectives

After studying this chapter, you should be able to:

- Assess yourself for a data science / analytics role.
- Identify the knowledge required to build yourself for a specialist data role.
- Prepare yourself for any data science interview.

Data Science Overview

1. What are the 4 transitional ages of data analytics? Explain with examples.
2. Discuss five use cases of data analytics applied in industry.
3. Explain the three key components of data science; domain knowledge, tools and technology, and mathematical and scientific techniques
4. Give at least two examples of supervised learning algorithms and their application?
5. Which of the following is a database technology?
 - a. SQL
 - b. Python
 - c. R
 - d. Postgres
6. What is Software as a Service business model?
7. What is the difference between predictive and prescriptive data analysis?
8. Discuss the responsibilities of the following job roles.
 - a. Data Analyst
 - b. Data Science
 - c. Data Engineer
9. Why is data visualization important for summarizing large amounts of data?

10. What is bias in AI algorithms? Research online and build awareness of Responsible AI

Mathematics Essentials

1. Differentiate between matrices and tensors.
2. Why are eigenvectors and eigenvalues used in data analysis? Select all the correct options.
 - a. Increase the noise in data
 - b. Decrease the noise in data
 - c. Capture significant information
 - d. Reduce the dimensions of data
3. Explain Eigenvalue decomposition with proper example.
4. Explain Single value decomposition with proper example.
5. Explain Principal component analysis (PCA).
6. What is the use of the Principal component analysis technique in data analysis?
7. Differentiate between definite and indefinite variables.
8. Explain the gradient descent algorithm with an example.
9. What are the special rules in differential calculus? Discuss.
10. Write a short note on
 - a. Vectors
 - b. Matrices
 - c. Tensors
 - d. Determinant
 - e. Multivariate Calculus

Statistics Essentials

1. Define the population and sample statistics.
2. Enumerate different sampling techniques and give an example of each.

3. Is numerical data always continuous in nature?
 - a. True
 - b. False
4. What is the typical data we capture in surveys?
 - a. Qualitative
 - b. Quantitative
5. Define the following measure of central tendency and give one example each.
 - a. Mean
 - b. Mode
 - c. Median
6. Define the following measure of variability and give one example each.
 - a. Range
 - b. Variance
 - c. Covariance
 - d. Standard Deviation
7. What is the difference between asymmetry and variability?
8. Define the central limit theorem and its significance in statistical inferences.
9. Give an example of a conditional probability to explain the naive Bayes theorem.
10. Does the coin toss follow binomial distribution?
 - a. True
 - b. False

Exploratory Data Analysis

1. What is Exploratory Data Analysis (EDA) in data science?
2. Discuss the importance of EDA with an example scenario.

3. Explain the steps involved in the EDA process.
4. Discuss in detail the different types of data.
5. Briefly explain about the methods in EDA.
6. What are the objectives of EDA?
7. is the stuff done during the EDA process?
8. Define the following terms:
 - a. Training data
 - b. Validation data
 - c. Testing data
9. How to display the rows in a dataset? Give an example.
10. What is the use of describing () function?

Data Preprocessing

1. What is data preprocessing?
2. What are the important steps in the data preprocessing technique?
3. Enumerate the methods of data preprocessing.
4. Discuss the process of data transformation.
5. What is Normalization?
6. What are the benefits of Normalization?
7. How to handle the missing values in a dataset?
8. What is the use of .info() in data preprocessing?
9. How to visualize the output of preprocessed data?
10. Demonstrate the data preprocessing with sample data.

Feature Engineering

1. What is the core purpose of feature engineering step in Machine Learning development?
2. What is the difference between raw data and features?
3. Do algorithms also influence the choice of features?

- a. True
 - b. False
4. What is an outlier, and how to handle it?
 5. What are the different types of imputation techniques?
 6. How binning is helpful in creating features from continuous data?
 7. When should log-transform be applied?
 8. Does one-hot encoding increase the cardinality of a dataset?
 - a. True
 - b. False
 9. When scaling of a variable is useful? Give an example.
 10. Explain the process flow of features engineering and its impact on the accuracy of the model.

Machine Learning Algorithms

1. What is machine learning?
2. What areas is machine learning being used?
3. How statistics changed the way we viewed machine learning in the 1990s?
4. Differentiate between supervised, unsupervised, and reinforcement learning.
5. Which of the following machine learning algorithms falls under supervised learning?
 - a. K-Means
 - b. Apriori algorithm
 - c. Linear Regression algorithm
 - d. Hierarchical clustering
6. How is linear regression different from logistic regression?
7. Give examples of day to day life problems that can be solved with linear regression.
8. What is the random forest algorithm? Explain.

9. What are the Python modules which are used for machine learning?
10. What is meant by Lemmatization? Explain.

Productionizing Machine Learning Models

1. What is the difference between model training and model scoring?
2. Discuss the model production system and its types with examples.
3. Explain the difference between batch prediction and batch learning.
4. What is the concept of REST APIs and why they are so ubiquitous in modern application development?
5. Give an example of an HTTP URL and its components?
6. What are the different types of HTTP methods and their use?
7. What is a resource in client-server architecture?
8. What is Flask, and what are its key components?
9. Build a simple algorithm and create a flask application to host it as a REST API.
10. Build a simple HTML interface for the application and access your application from the web-browser.

Data Flows in Enterprises

1. What is a Data Pipeline?
2. Define: Extract Transform Load (ETL)
3. Mention the different functionalities of the data pipeline.
4. What are the different categories of data pipelines?
5. Enumerate the key considerations before design any data pipeline.
6. Compare: ETL and ELT
7. Draw the ETL data pipeline flow and explain it in brief.
8. What is a Job in the scheduler?
9. Illustrate the process of job scheduling with an example.
10. Discuss the messaging queue system with its components.

Introduction to Databases

1. Define Query?
2. What do you mean by Structure Query Language (SQL)?
3. What is the difference between relational databases and NoSQL databases?
4. Discuss Acid properties?
5. What is database schema?
6. Write the names of popular DBMS that are available in the market?
7. Discuss different JOIN operations.
8. What is ORM?
9. Describe the installation of MongoDB.
10. Describe the Graph database.

Introduction to Big Data

1. What is Big Data? How can we define Big Data?
2. What is the abbreviation of HDFS?
3. Discuss the Hadoop distributed file system and the principles briefly behind it
4. What is the significance of MapReduce in HDFS? Explain how the algorithm works?
5. How does Hadoop manage resources and job scheduling?
6. What are the key ingredients to start a Hadoop cluster?
7. Write a program to count words in document using MapReduce?
8. What are the steps to follow to run the program you wrote above in an HDFS cluster?
9. What is the default size of a block in an HDFS?
10. Consider you have an HDFS cluster with 1 name node and 2 data nodes? What happens when there is a disk failure in one data node while accessing data using your above-written word count code?

DevOps for Data Science

1. What do you mean by DevOps?
2. What is the difference between Agile methodology and CI/CD?
3. What are the three functions of DevOps?
4. What is the DevOps cycle? Describe it.
5. Write the name of two popular version management systems?
6. Describe QA and its different types.
7. What do you mean by unit testing?
8. What is the difference between unit testing and model testing?
9. What is Docker Container?
10. What are the differences between the virtual machine and Docker containers?

Introduction to Cloud Computing

1. What are the key components of the OSmodel?
2. What is the role of an operating system in the OS model?
3. Can you use the same hardware to run multiple operating systems at the same time? Explain your answer.
4. What is virtualization technology?
5. Define the functionality of the hypervisor layer.
6. What is cloud computing, and how it differs from VM?
7. Explain the following Cloud service models with examples.
 - a. IaaS
 - b. PaaS
 - c. SaaS
8. Explain the types of cloud infrastructure and their use cases.
 - a. Private Cloud
 - b. Public Cloud
 - c. Hybrid Cloud

9. Discuss the applicability of cloud to data science with respect to the four key areas.
 - a. Data Storage
 - b. Computing
 - c. Integration
 - d. Deployment
10. List key cloud service providers and their major services.

Deploy Model to Cloud

1. What options does GCP provide to deploy any application on the cloud?
2. How can we access the resources on GCP?
 - a. GCP web console
 - b. GCP cloud shell
 - c. GCP cloud API
 - d. All the above
3. What is the minimum disk space required to start a VM?
4. How to connect to a VM in GCP?
5. What is the command to deploy an application to an app engine?
6. What is the default config file required by the app engine to recognize an application for deployment?
7. What would you do if you have figured out memory or disk is not enough for your application? How would you achieve it in GCP?
8. What does gcloud app browse do?
9. What is the counterpart of Google loadbalancer in AWS?
10. What is a significant cloud service provides other than Google?

Introduction to Business Intelligence

1. How does business intelligence conceptual flow work?

2. What are the key areas BI is helping the business?
3. What are the benefits of an organization from BI analysis?
4. When does an organization find the necessity for BI analysis?
5. What are the different ways that businesses can achieve their full potential according to Tableau's learning series?
6. Explain briefly the process involved in BI analysis.
7. What are the different types of data available in the real world?
8. What are the key performance indicators? And why is it significant in a BI process?
9. What are the key factors which determine the adoption of a visualization tool?
10. What are the key BI trends identified by Tableau?

Data Visualization Tools

1. What are the different types of data charts and visual representations of data?
2. What is the difference between bar graphs and pie charts?
3. What are line plots?
4. Write the names of different visualization tools that are available in the market?
5. What are the features of Microsoft Power BI?
6. What is the difference between Power BI Pro and Power BI Premium?
7. Write two features of Power BI premium?
8. Write the names of three Business-friendly data visualization tools?
9. Write the definition of Data visualization?
10. What do you mean by Cartesian plots?

Conclusion

In this chapter, we have provided 10 questions each from the previous 16 chapters. By reading this chapter, the reader will be able to assess himself/herself for a data science role. He or she will be able to identify the

skill and knowledge requirements in the current data science world. Thus, the reader can prepare well for a data science interview.