

Python Machine Learning Projects

Learn how to build Machine Learning projects from scratch



Dr. Deepali R Vora

Dr. Gresha S Bhatia

bpb



Python Machine Learning Projects

Learn how to build Machine Learning projects from scratch



Dr. Deepali R Vora

Dr. Gresha S Bhatia

bpb

Python Machine Learning Projects

*Learn how to build Machine Learning projects
from scratch*

**Dr. Deepali R Vora
Dr. Gresha S Bhatia**



www.bpbonline.com

Copyright © 2023 BPB Online

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-89898-27-9

www.bpbonline.com

Dedicated to

*To my parents, in-laws, daughter, husband Mr. Rahul Vora,
and family members*

— Dr. Deepali R Vora

*To my parents, in-laws, husband Mr. Sachin S Bhatia,
and family members*

— Dr. Gresha S Bhatia

About the Authors

Dr. Deepali R Vora is a Professor and Head of Computer Science & Engineering at Symbiosis Institute of Technology, Pune, and has completed her Ph.D. in Computer Science and Engineering from Amity University, Mumbai. She has more than 22 years of teaching, research as well as Industrial experience. She has more than 60 research papers published in Journals and Conferences of International and national repute. She has co-authored three books and delivered various talks in Data Science and Machine learning. She has conducted hands-on sessions in Data Science using Python for students and faculty. She was appointed as a Syllabus Revision Committee member with Mumbai University and developed the course content for B.E. (Information Technology) course. She has received grants for conducting research and organizing training courses for faculties. She acts as a technical advisor and reviewer for many International Conferences and Journals. Her blogs on KnowledgeHut have received wide acknowledgment. She has developed a course on “Deep Learning” on the Unschool platform.

Dr. Gresha S Bhatia is the Deputy Head of the Computer Engineering Department at Vivekanand Education Society’s Institute of Technology (VESIT), Mumbai, and has completed her Ph.D. Technology from the University of Mumbai. She has more than 25 years of industry and teaching experience. She has published more than 50 research papers in International Journals, conferences, and national conferences of repute. She has been awarded Microsoft AI for Earth Grant in the year 2019 as well as Minor Research Grant from the University of Mumbai. She has authored a book and has delivered sessions on Machine learning and Social Computing. She has also been a member of the Syllabus Revision Committee of Mumbai University for the undergraduate and postgraduate programs in Engineering. She has also been a technical advisor and reviewer for the number of International Conferences and Journals.

About the Reviewer

Ravi Choudhary is a Data Scientist from New Delhi, India. He has several years of experience working for the Government of India and Top Global companies. Ravi holds a Master's degree in Artificial Intelligence from IIT and applies his expertise in this field to his work in data science. He is an expert in machine learning and has extensive experience developing and implementing advanced algorithms and models to extract insights from large and complex datasets. He has a deep understanding of various machine learning techniques, such as supervised and unsupervised learning, deep learning, and natural language processing. Ravi has applied his expertise in machine learning to a wide range of industries, including finance, healthcare, and transportation. His work has led to significant improvements in efficiency, accuracy, and performance. He is also known for his ability to communicate complex technical concepts to non-technical stakeholders and effectively collaborate with cross-functional teams.

Acknowledgements

We would like to take this opportunity to thank a few people for their continued support while writing this book. First and foremost, we would like to thank our parents and parents-in-law for encouraging us to author the book.

This journey was possible due to the constant support of our life partner, who has been our pillar of strength.

We are grateful to the BPB publishing house, which showed us the path and initiated our learning process.

Our sincere gratitude to all our family members and well-wishers for always standing by us.

We sincerely thank all those who have directly or indirectly supported us while authoring this book.

Preface

This book covers different aspects of machine learning, and the importance of developing machine learning algorithms that can be applied to real-life projects. The book elaborates on the need and different types of machine learning algorithms. It shows how the right amount of data, and the correct amount of training can help build optimized systems. These optimized systems can help reduce the cost as well as training errors. This book further provides a number of questions to be solved that gives a practical approach. It further gives importance for understanding the python code, its basic concepts as well as the implementation code for machine learning projects.

This book is divided into five chapters. The book elaborates on the need for machine learning, its challenges and limitations. It further explores the development of machine learning algorithms through use of Python code. The working of the machine learning algorithms is emphasized through real-life case studies. The best algorithm that can be applied to a case study is elaborated through the various optimization and meta-heuristic approaches. The details of the chapters are specified below.

Chapter 1: Introduction to ML – introduces the terms artificial intelligence, data science and machine learning and the differences between them. The various models, the types of machine learning algorithms, and the challenges and limitations faced are further explored. The chapter will further dwell on the working and the application areas of machine learning.

Chapter 2: Python Basics for ML – will cover the concepts behind the Python programming language. The syntax and the need for python programming is further emphasized. It will elaborate on python tools and libraries. It further handles the concepts of file and exception handling, which form the crux of the machine learning domain.

Chapter 3: An Overview of ML Algorithms – will introduce the various Machine learning algorithms. The major focus is to provide familiarity with the machine learning programs that can learn from the data provided and improve from experience, without any human intervention. These learning tasks may include learning the function that maps the input to the outputs or

learning through the hidden structure in unlabeled data using algorithms coined under supervised and unsupervised learning. To further understand the core concepts, this chapter will explore the working process of any given machine learning algorithm through a number of examples. As Python forms a handy tool to get started with supervised and unsupervised learning, this chapter will explore its various functionalities. The chapter will further elaborate on the performance measures to be considered for evaluating the machine learning system that would have been developed.

Chapter 4: Case Studies and Projects in Machine Learning – will provide an insight into the various case studies that use the concepts and algorithms of machine learning. This chapter will further introduce recommendation systems, its needs, the process of generating the recommendation systems as well as the current systems incorporating recommendation systems. Another case study will be focused on the application of machine learning algorithms on text mining applications, opinion mining and sentiment analysis. This chapter will further elaborate on applying various machine learning algorithms to image processing applications. More case studies incorporating predictive analysis, Social Media Analytics, Customer churning analytics, and Analytics in Education Systems will be explored in detail.

Chapter 5: Optimization in ML Algorithm – will focus on the improvements in training the algorithms for the most accurate outputs. The chapter will thus focus on determining the best element or path, or techniques to improve the efficiency of algorithms by decreasing the training time as well as the cost incurred for the same. The concept of optimization and the hybrid algorithms that can be utilized will then be explained. The need for optimization techniques in Machine Learning based projects will then be highlighted. This will lead to applying techniques for optimization. The basic optimization techniques will then be explored, and the current research area on meta-heuristic approach will be further elaborated. To conclude the chapter, the various python libraries that could be available for optimization in Machine Learning projects will be explained.

Code Bundle and Coloured Images

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

<https://rebrand.ly/sn6fi4n>

The code bundle for the book is also hosted on GitHub at <https://github.com/bpbpublications/Python-Machine-Learning-Projects>.

In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at <https://github.com/bpbpublications>. Check them out!

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, that may have 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.

Did you know that BPB offers eBook versions of every book published, with PDF and ePUB files available? You can upgrade to the

eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at: business@bpbonline.com for more details.

At www.bpbonline.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit www.bpbonline.com. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit www.bpbonline.com.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

[https://discord\(bpbonline\).com](https://discord(bpbonline).com)



Table of Contents

1. Introduction to ML

Introduction

Structure

Objectives

Introduction to Machine Learning (ML)

Models in Machine Learning

Supervised machine learning model through training

Unsupervised machine learning model: Self-sufficient learning

Semi-supervised machine learning model

Reinforcement machine learning model: Hit and Trial

Types of Machine Learning Algorithms

Working of Machine Learning algorithm

Challenges for Machine Learning Projects

Limitations of machine learning

Application areas of ML

Difference between the terms data science, data mining, machine learning

and deep learning

Conclusion

Questions and Answers

2. Python Basics for ML

Introduction

Structure

Objectives

Spyder IDE

Jupyter Notebook

Python: Input and Output Commands

Logical Statements

Loop and Control Statements

Functions and Modules

Class Handling

Exception Handling

[File Handling](#)
[String functions](#)
[Conclusion](#)
[Questions and Answers](#)

3. An Overview of ML Algorithms

[Introduction](#)
[Structure](#)
[Objectives](#)
[Machine learning modeling flow](#)
 [Terms used in preprocessing](#)
 [Raw data \(or just data\)](#)
 [Prepared data](#)
 [Need for Data Preprocessing](#)
[Preprocessing in ML](#)
 [Researching the best model for the data](#)
 [Training and testing the model on data](#)
 [Evaluation](#)
 [Hyperparameter tuning](#)
 [Prediction](#)
 [Metrics Used](#)
[Regression algorithms](#)
 [Types of regression techniques](#)
 [Linear Regression](#)
 [Logistic Regression](#)
 [Polynomial Regression](#)
 [Stepwise Regression](#)
 [Ridge Regression](#)
 [Lasso Regression](#)
 [ElasticNet Regression](#)
[Classification](#)
 [Terminology used in Classification Algorithms](#)
 [Types of classification algorithms](#)
 [Performance measures for classification](#)
[Clustering](#)
 [Clustering algorithms](#)
 [K-Means Clustering](#)

[Mean-Shift Clustering](#)
[Agglomerative Hierarchical Clustering](#)
[Clustering Validation](#)
[Neural Network and SVM](#)
 [Building Blocks: Neurons](#)
 [Combining Neurons into a Neural Network](#)
 [Training the neural network](#)
 [Neural Network Architectures](#)
[Support vector machine \(SVM\)](#)
[Machine Learning Libraries in Python](#)
 [Numpy](#)
 [Pandas](#)
 [Populating the Dataframe](#)
 [Displaying the Data Using Dataframe](#)
 [Accessing the Data Selectively in Dataframe](#)
 [Basic Descriptive Statistics using Pandas](#)
 [Data transformation](#)
 [Data Preprocessing – Handling missing values](#)
 [Matplotlib](#)
 [Line Graph](#)
 [Scatter Plot](#)
 [Bar plot](#)
 [Histogram](#)
 [Pie Chart](#)
[Evaluation of ML Systems](#)
 [Test and Train Datasets](#)
 [Performance Measure](#)
 [Model Evaluation Techniques](#)
 [Model evaluation metrics](#)
 [Classification Metrics](#)
 [Regression Metrics](#)
[Conclusion](#)
[Questions](#)

4. Case Studies and Projects in Machine Learning

[Introduction](#)
[Structure](#)

Objectives

Recommendation Generation

Importance of recommendation systems

Key terms used

Items/Documents

Query/Context

Approaches for building recommendation systems

Basic recommendation systems

Candidate Generation

Recommendation generation

Information collection phase

Learning phase

Prediction/recommendation phase

Evaluation metrics for recommendation algorithms

Statistical accuracy metric

Case study on Recommendation system: E-learning system domain

Recommender systems

Problem definition

Objective of the case study

Considerations for the case study

System development

Constraints / limitations while developing the recommendation system

Text Analysis and Mining

Importance of text mining

Basic blocks of text mining system using ML

Steps involved in preparing an unstructured text document for deeper analysis

Text mining techniques

Information Retrieval (IR)

Natural language processing (NLP)

Sentiment analysis

Naive Bayes

Linear regression

Support Vector Machines (SVM)

Deep learning

Case study on product recommendation system based on sentiment

analysis

Product recommendation

Problem definition

System development

Considerations for the case study

Opinion mining

Image processing

Importance of image processing

Purpose of image processing

Basic Image Processing System

Image Processing using popular ML algorithms

Real Time case studies of Image Processing using ML

Problem definition

Objective of the case study

Considerations for the case study

System development

Algorithms that can be employed

Tool Utilization

Constraints/Limitations while developing the system

Evaluation Measures

Predictive analytics

Importance of predictive analytics

Need for predictive analysis

Machine Learning vs Predictive Modeling

Building a predictive model

Types of predictive algorithms

Types of Predictive Analytical Models

Comparison of Predictive Modeling and Predictive Analytics

Predictive modeling vs data analytics

Comparison between and Predictive Analytics and Data Analytics

Uses or applications of Predictive Analytics

Benefits of predictive analytics

Challenges of predictive modeling

Limitations of predictive modeling

Case studies

Social media analytics

Case study of Instagram

Case study on Customer churning analytics
Building the hypothesis
Case study on learning analytics in education systems
Challenges faced
Approach
Other case studies
Conclusion

5. Optimization in ML Algorithm

Introduction
Structure
Objectives
Optimization – Need of ML projects
 Types of optimization techniques
 Conventional Approach
 Metaheuristic Approach
 Basic Optimization Techniques
 Backpropagation optimization
 Gradient descent optimization
Metaheuristic approaches to optimization
 Types of metaheuristic algorithms
 Single solution-based algorithms
 Population-based algorithms
Improvisation of ML algorithms by optimizing the learning parameters
 Case study 1: Metaheuristic Optimization Techniques in Healthcare
 Case Study 2: Genetic Algorithm (GA) in Batch Production
Optimization using Python
Conclusion
Questions

Index

CHAPTER 1

Introduction to ML

Introduction

The term *machine learning* was coined by *Arthur Samuel* in 1959. The basic idea behind the coined term was “Can machines do what we as humans can do?” rather than asking “Can machines think?” These questions led to the development of machine learning where, just like human beings, machines tend to learn from experience. The aim is to improve the performance with experience. This chapter introduces the related terms, such as data science, data mining, artificial intelligence, machine learning and deep learning. The major focus is to familiarize you with the methods and techniques used in machine learning. To understand the core concepts, this chapter explores the working process of any given machine learning algorithm. In the recent years, machine learning technology has improved drastically, which is elaborated through the various applications, limitations and the challenges faced while developing the machine learning algorithms.

Structure

In this chapter, we will discuss the following topics:

- Introduction to Machine Learning
- Models of Machine Learning
 - Supervised machine learning model through training
 - Unsupervised machine learning model
 - Semi - structured machine learning model
 - Reinforcement machine learning model
- Working of Machine Learning algorithm
- Challenges for Machine Learning Projects
- Limitations of Machine Learning

- Application areas of Machine Learning
- Difference between the terms data science, data mining, machine learning and deep learning

Objectives

On completion of this chapter, you will be able to understand the various key terms and the fundamentals of machine learning. Additionally, you will be able to understand the types, the models and the mechanism of machine learning. You will become familiar with the challenges and limitations observed while applying machine learning algorithms. These will be elaborated through a number of application areas along with the differences between the various related technologies employed.

Introduction to Machine Learning (ML)

Machine learning is defined as the ability of a computer system to learn from the environment where the data is provided through enabling algorithms. These algorithms gather insights and take data-driven decisions with minimal human intervention. This enables us to make predictions on previously unanalyzed data. So instead of writing code, one just needs to feed the data to the generic algorithm, and the algorithm/machine builds the logic based on the given data. This helps improve the output from its experience without the need for any explicit programming. Thus, it can be said that machine learning is a term closely associated with data science, and it involves observing and studying data or experiences to identify patterns and set up a reasoning system based on the findings. Another way of describing machine learning is to say that it is a science of building hardware or software that can achieve tasks by learning from examples.

The examples come as {input, output} pairs. When new inputs are given, a trained machine can predict the output. For example, recommendations for online products purchased or suggestions for the persons who bought the product is machine learning. The terms used in machine learning include a target that is called a **label**, a variable used in statistics that is called a **feature**, and transformation in statistics that is called **feature creation**.

Models in Machine Learning

Machine learning models can be categorized into three basic models: supervised, unsupervised and reinforcement learning.

Supervised machine learning model through training

Supervised learning is said to be a learning obtained by training a machine or a model. On getting trained, predictions can be made for new data, also known as test data.

This model works on the data provided to the system. The data available is divided into **training** and **testing** data. A supervised learning model analyses the given training data and draws inferences from it. Therefore, mapping between the input and output pair and proper labeling of data is crucial in supervised machine learning models.

The major aim of a supervised model is to utilize historical data, understand its behavior and determine future forecasts based on the historical data available and maintained in the database.

For example, to differentiate between plants and flowers, a few labeled pictures of both categories need to be fed. This will enable the machine learning algorithm to differentiate between, identify and learn about them based on their characteristics. Once the algorithm is trained, classifying the remaining images would become a lot easier for the algorithm.

Unsupervised machine learning model: Self-sufficient learning

Such a model does not use any classified or labelled parameters. It learns through observation and finds structures in the data. It focuses on discovering hidden structures, patterns and relationships from unlabeled data, which enhances the functionality of the system by creating a number of clusters for further analysis.

The unsupervised system or its algorithms are not given the “right answer.” The algorithm is expected to determine, interpret, and review its data and conclude from what is being shown to them through the iterative deep learning approach. Unsupervised learning can use both generative learning models and a retrieval-based approach.

These algorithms use self-organizing maps, nearest-neighbor mapping, k-

means clustering and singular value decomposition as its techniques for operations.

Unsupervised learning also known as neural networks works well for recognizing images, performing operations on transactional data, performing speech to text conversion and natural language generation.

Semi-supervised machine learning model

This model is a combination of supervised and unsupervised learning. It works using a small amount of labeled and large amount of unlabeled data for training to improve the learning accuracy.

Here, all the unlabeled data is fed in, and the machine applies the various algorithms, such as classification, regression and prediction. Then, it understands the characteristics and classifies the information from the data provided.

This learning provides an effective solution when the cost associated with labeling is too high.

Reinforcement machine learning model: Hit and Trial

This learning model interacts with the environment on a trial-and-error basis, determining the best outcome. Reinforcement learning comprises of three primary components: the **agent** (the learner or decision maker), the **environment** (everything the agent interacts with) and **actions** (what the agent can do). The agent here is rewarded or penalized with a point. Based on the actions they take, the output should be maximized over a given amount of time. Steps that produce favorable outcomes are rewarded, and steps that produce undesired outcomes are penalized until the algorithm learns the optimal process.

Thus, the goal in reinforcement learning is to learn the best policy to obtain maximum rewards.

Reinforcement learning is often used for robotics, gaming and navigation. For example, to understand a game of chess, an ML algorithm will not analyze individual moves; it will study the game as a whole.

Types of Machine Learning Algorithms

Based on the various models developed, Machine learning is sub-categorized into three types: supervised, unsupervised and reinforcement-based algorithms, as represented in [Figure 1.1](#):

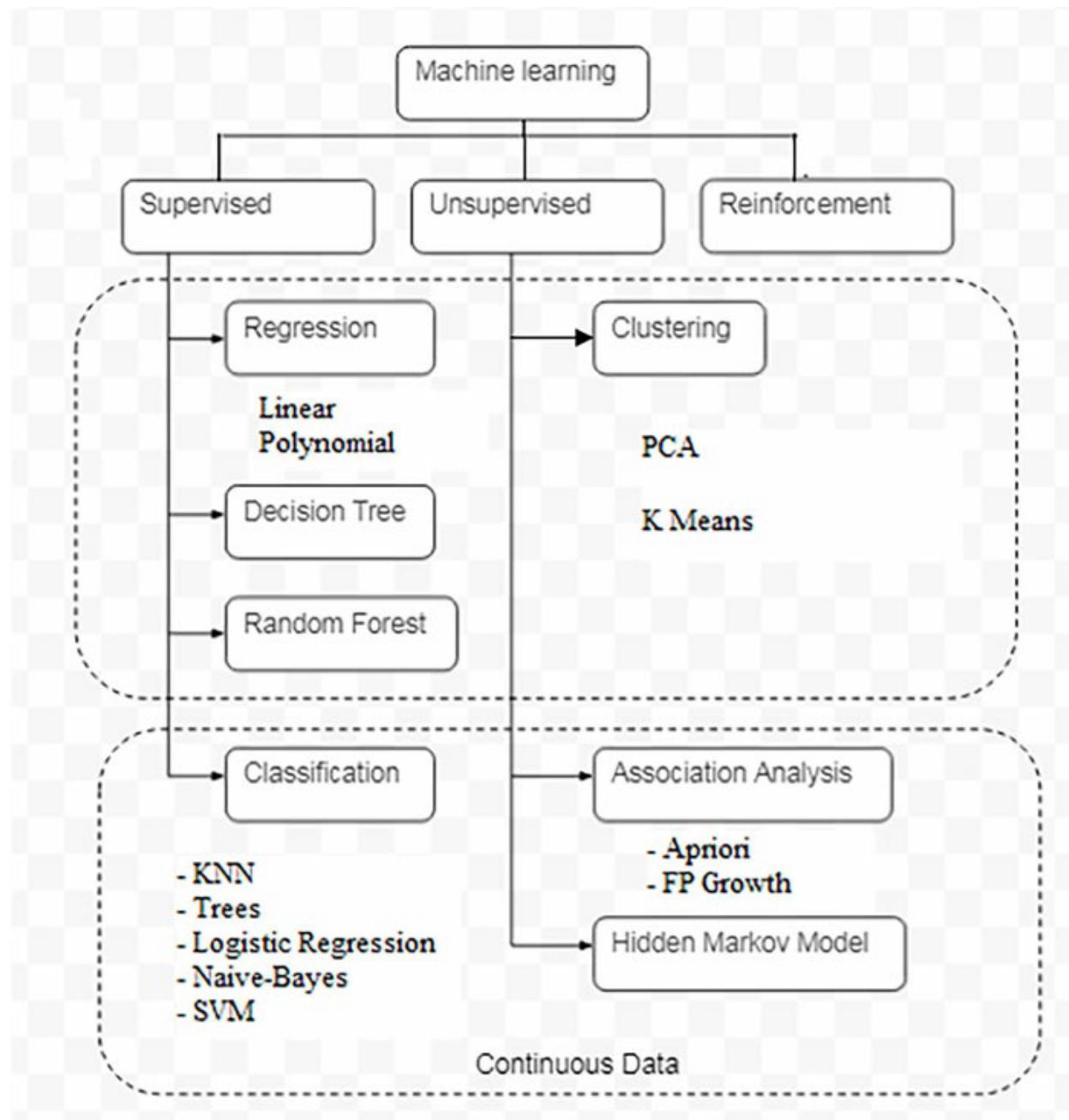


Figure 1.1: Types of Machine Learning

As mentioned in [Figure 1.1](#), supervised and unsupervised algorithms work on continuous and categorical data that is provided. The supervised algorithms that work on continuous data include regression, decision trees, random

forests and classification based methods for categorical data. Unsupervised algorithms include clustering, association analysis and the Hidden Markov model.

Working of Machine Learning algorithm

Machine Learning algorithm is trained using a training data set. This training leads to creation of a model. When new input data is introduced to this model, the ML algorithm makes a prediction.

The following figure depicts the working of ML algorithm:

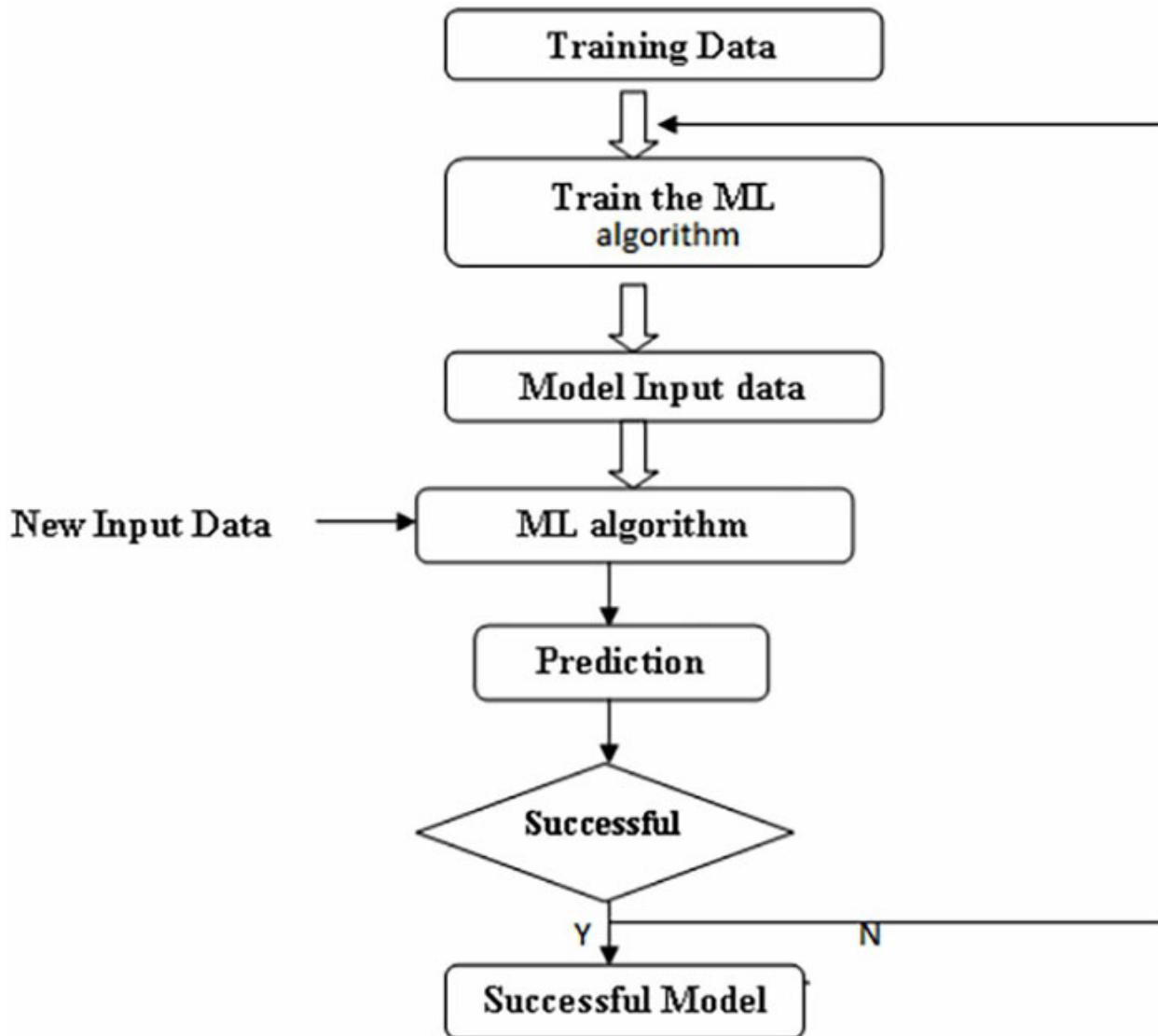


Figure 1.2: Working of ML algorithm

If the accuracy of the prediction obtained is acceptable, the associated machine learning algorithm is deployed. However, if the accuracy obtained is not acceptable, the algorithm is repeatedly trained with an augmented training data set till its accuracy gets acceptable, as represented in [Figure 1.2](#).

[Challenges for Machine Learning Projects](#)

Employing a machine learning method can be extremely tedious, but it can serve as a revenue charger for a company. While machine learning is still evolving, various challenges faced by the organizations include the following:

Understanding the limits of contemporary machine learning technology

Many companies expect the algorithms to learn quickly and deliver precise predictions to complex queries. Therefore, they face the challenge of educating customers about the possible applications of their innovative technology. A business working on a practical machine learning application needs to invest time and resources and take substantial risks. Also, machine learning engineers and data scientists cannot guarantee that the training process of a model can be replicated. Therefore, frequent tests should be done to develop the best possible and desired outcomes.

The black box problem

Understanding how the algorithms work made the ML models simple and shallow methods in the early days. However, as ML algorithms work on large sets of data, algorithms have moved from neural networks to deep learning algorithms. These networks gave good accuracy, but the logic and the models developed to do so was not known. This problem is called a black box problem.

Data collection

Immense amount of data is available to train a machine learning model. Data engineering plays a significant role in collecting and streamlining the data. Though data storage has become cheap, it requires time to collect enough data or buy ready sets of data, which is also expensive. Additionally, preparing data for algorithm training is a complicated process. To do that, one needs:

- To know the problem that the algorithm needs to solve,

- To establish data collection mechanisms and train the algorithm,
- Data reduction with attribute sampling, record sampling, or aggregating;
- To decompose the data and rescale it, a complex and expensive task;

Data privacy is yet another challenge. Differentiating between sensitive and insensitive data is essential to implement machine learning correctly and efficiently.

Feature Selection

In a dataset, you may have 1000 features or more, but only selected features help in the prediction; selecting those features is the challenging task. Here, domain knowledge and data scientist expertise come into the picture.

It is a challenge to select the smallest possible subset of input variables (features) that will give optimal or the best predictions.

Performance prediction

Once optimized predictions are made, the challenge is to predict their generalized performance on new, unseen data.

For measuring the performance, we have parameters to check; for example, we have classification matrix, accuracy, recall, precision, roc-auc curve and so on in classification problems, and we have accuracy, mean squared error, Adjusted R2 squared error and so on in regression problems.

To overcome the various challenges faced, the focus should be on understanding the concepts behind data collection, feature selection and extraction, applying the right algorithm and predicting the performance and accuracy of the applied algorithms.

Limitations of machine learning

The applications of machine learning are observed in various domains, from the financial and retail sectors to agriculture, banking and many more. However, machine learning algorithms have their own limitations, mentioned as follows:

- Each narrow application needs to be specially trained
- Requires large amounts of *hand-crafted, structured* training data
- Learning must generally be supervised: training data must be tagged

- Require lengthy offline/ batch training
- Does not learn incrementally or interactively in real time
- Poor transfer learning ability, reusability of modules and integration
- Systems are opaque, making them very hard to debug
- Performance cannot be audited or guaranteed
- They encode correlation, not causation or ontological relationships
- Do not encode entities, or spatial relationships between entities
- Only handle very narrow aspects of natural language

To obtain the best results, various techniques need to be properly employed while considering all limitations.

Application areas of ML

Various industries have embraced machine learning technologies to gain competitive advantage. Domains that utilize machine learning include the following:

Financial services

Predictions are made by applying appropriate machine learning algorithms to identify two important factors: determining insights into data for identifying investment opportunities or helping investors to know when to trade. Additionally, various data mining and machine learning algorithms are utilized to perform cyber surveillance to prevent frauds from occurring, and to identify clients with high-risk profiles in banking sector and other businesses in the financial industry.

Government

For a government organization, there is a huge collection of data obtained from multiple sources, and records are maintained over years. Such data forms the prime source from where data can be mined, and insights generated to improve public safety and provide utilities for the welfare of the society. Machine learning can play an important role in determining such services. It can also help detect fraud and minimize identity theft.

Health care

Due to the advent of wearable devices, sensor technology incorporated in the

healthcare sector can provide access to a patient's health in real time. This data accumulation can help medical experts analyze the data, and identify trends and critical areas, which can lead to improved diagnosis and treatment. Machine learning models can be developed and trained to provide insights into the patient's health.

Retail

Online shopping is the trend these days. Business organizations are providing services like recommending products and their combinations that have been previously purchased based on the buying history. Retailers rely on machine learning to capture data, analyze it and use it to personalize a shopping experience and provide customer insights along with price optimization.

Oil and gas

Machine learning models are being trained to determine new energy sources, analyze minerals in the ground, predict refinery sensor failure, and streamline oil distribution to make the refineries more efficient and cost-effective.

Transportation

The transportation industry majorly relies on making routes more efficient and predicting potential problems. So, machine learning algorithms to be developed have to analyze, the incoming data needs to be modeled. This can further be used to identify patterns and trends to generate optimized routes. Thus, machine learning is an important tool for various transportation organizations.

Other set of applications used include the following:

Facebook's News Feed

The News Feed uses machine learning to personalize each member's feed. If a member frequently stops scrolling to read a particular friend's posts, the News Feed will start showing more of that friend's activity earlier in the feed. This is done by performing statistical analysis and predictive analytics to identify patterns in the user's data and using those patterns to populate the News Feed.

Enterprise application

In the **Customer Relationship Management (CRM)** system, machine learning utilizes learning models to analyze the emails and informs the sales team members about the most important mails to respond to first. Similarly,

Human resource (HR) systems use machine learning models to identify the characteristics of effective employees and use this to find the best applicants for open positions.

Machine learning software help users automatically identify important data points for providing business insights and intelligence.

Self-driving cars

Machine learning uses deep learning neural networks to identify objects and determine optimal actions for safely steering a vehicle.

Virtual assistant

In order to interpret natural speech, personal schedules or previously defined preferences and take action, smart assistants utilize machine learning technology through several deep learning models.

Difference between the terms data science, data mining, machine learning and deep learning

Although each of these methods have the same goals, i.e., to extract insights, patterns and relationships that can be used to make decisions, each one has a different approach and ability to determine them, as indicated by [Table 1.1](#).

Data Science and Data Mining

Data science deals with the entire scope of collecting and processing data, while data mining involves analyzing large amounts of data to discover patterns and other useful information.

Data mining involves the use of traditional statistical methods to identify previously unknown patterns from data, such as statistical algorithms, machine learning, text analytics and time series analysis. It also includes the study and practice of data storage and data manipulation. Refer the table [Table 1.1](#):

Characteristics	Data Science	Data Mining
Definition	Process of collecting data	Analysis of data
Works on	Structured, semi structured and unstructured data	Structured data
Approach	A multi-disciplinary approach	A part of data science

	consisting of statistics, data visualization and natural language processing	
Process	Finds patterns in present and historical data	Operates on historical data

Table 1.1: Difference between data science and data mining

Machine learning and deep learning

Machine learning is developed based on its ability to probe data for its structures. It uses an iterative approach to learn from data, so the learning is easily automated.

Deep learning combines advances in computing power and neural networks to learn complicated patterns in large amounts of data for identifying objects in images and words in sounds, as indicated by [Table 1.2](#):

Characteristics	Machine Learning	Deep Learning
Optimal data volumes	Thousands of data points	Millions of data points
Outputs	Numerical values	Numerical values and free form data
Working	Uses automated algorithms that learn to model functions and predict future actions based on the data	Uses neural networks
Feature Selection	Requires specific techniques	Automatically done
Management	Algorithms are directed by data analysis to examine specific variables in data sets	Algorithms are self-directed on data analysis

Table 1.2: Difference between machine learning and deep learning

Conclusion

This chapter introduced you to the key terms used in data science, data mining, artificial intelligence, machine learning and deep learning. Various

types of machine learning algorithms, the methods applied and models deployed for learning were also explored here. The working of the machine learning algorithms and the challenges faced while developing the machine learning algorithm were then specified. This was followed by clearing the concepts of machine learning with the help of several examples. We also looked at the difference between data science, data mining, machine learning and deep learning technologies.

In the next chapter, the very popular Python language will be introduced. Then, we will explore code editors, IDE and the working of Python, along with its basic structure, control statements, and exception handling in detail.

Questions and Answers

1. Define the terms machine learning, data science and artificial intelligence with examples.

Ans. **Machine learning (ML)** algorithms enable computers to learn from data and even improve themselves without being explicitly programmed. You may already be using a device that utilizes ML, for example, a wearable fitness tracker like Fitbit or an intelligent home assistant like Google Home.

Data science: Data science is the technology that studies the data trend to make the best possible use of data. Machine learning algorithm, deep learning algorithm and Python libraries make the data science domain more powerful. It includes the following:

- Statistics (traditional analysis you're used to thinking about)
- Visualization (Graphs and tools)

Data science is used in domains like fraud detection, share market prediction, disease prediction, chatbot, social media management, user profiling and speech recognition.

Artificial intelligence: It is a technique that makes machines capable enough to perform tasks that humans are not capable of. These machines are not only trained to do tasks but also improve over time. For example, carrying a tray of drinks through a crowded bar and serving them to the correct customer is something servers do every day, but it is a complex exercise in decision-making and based on a high

volume of data being transmitted between neurons in the human brain.

In broader terms, machine learning and deep learning are the subset of artificial intelligence and data science's tools to get the data ready for the implementation whether it is cleaning, exploratory data analysis, imputation or splitting.

2. Explain the various models used in machine learning.

Ans. Let us talk about the various models in machine learning.

Linear regression: It is used to estimate real values (houses cost, number of calls, total sales and so on) based on continuous variable(s). Here, we establish a relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$.

The best way to understand linear regression is to relive this childhood experience. Let us say, you ask a child in fifth grade to list the people in their class by increasing order of weight, without asking them their weights! What do you think the child will do? They would likely look (visually analyze) at their classmates' height and build and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation.

In this equation:

- Y: Dependent variable
- A: Slope
- X: Independent variable
- B: Intercept

These coefficients A and B are derived based on minimizing the sum of squared difference of distance between data points and regression line.

Look at the following example. Here, we have identified the best fit line having linear equation $y = 0.2811x + 13.9$. Using this equation, we can find the weight, knowing the height of a person. Refer [Figure 1.3](#):

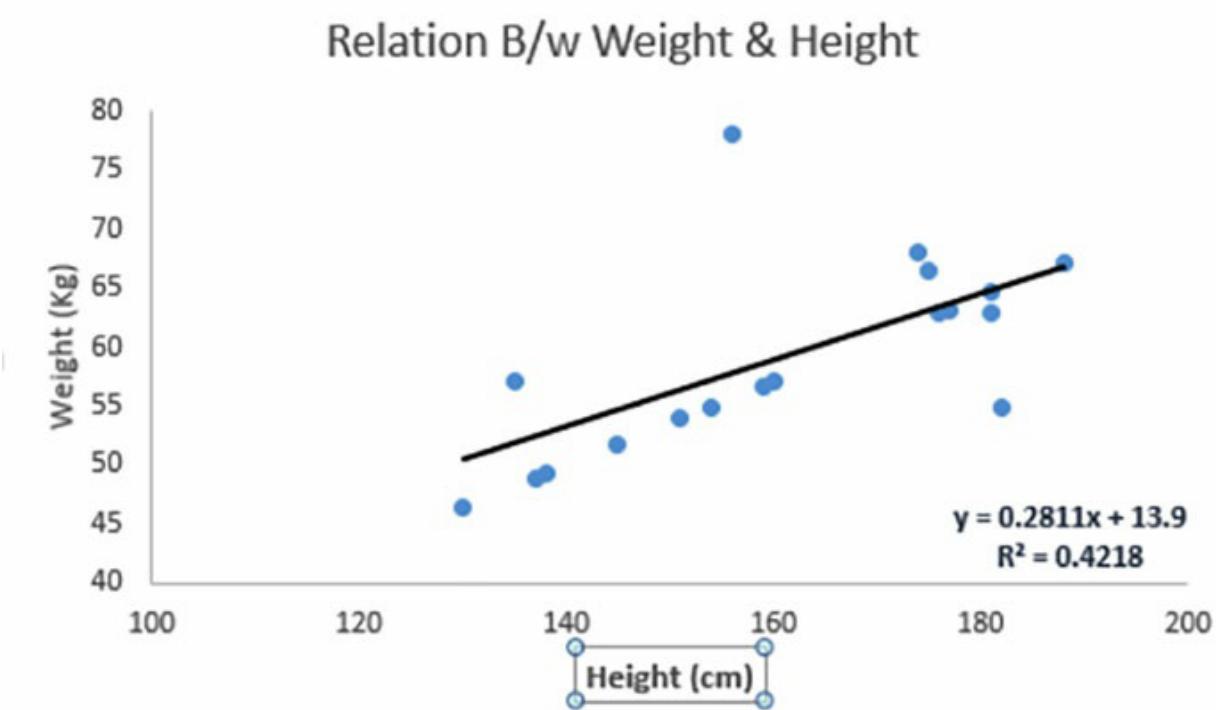


Figure 1.3: Graph presenting the weight

Linear regression is mainly of two types: simple linear regression and multiple linear regression. Simple linear regression is characterized by one independent variable, and multiple linear regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding the best fit line, you can fit a polynomial or curvilinear regression, and these are known as polynomial or curvilinear regression.

Logistic regression: Don't get confused by its name! It is a classification, not a regression algorithm. It is used to estimate discrete values (binary values like 0/1, yes/no, true/false) based on a given set of independent variable (s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as logit regression. Since it predicts the probability, its output values lies between 0 and 1 (as expected).

Again, let us try and understand this through a simple example.

Let's say your friend gives you a puzzle to solve. There are only two outcome scenarios: either you solve it or you don't. Now, imagine that you are being given wide range of puzzles/quizzes to understand which subjects you are good at. The outcome of this study would be something

like this: if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is fifth grade history question, the probability of getting an answer is only 30%. This is what logistic regression provides you.

Coming to math, the log odds of the outcome is modeled as a linear combination of the predictor variables:

$$\text{odds} = p / (1-p) = \text{Probability of event occurrence} / \text{Probability of event non-occurrence}$$

$$\ln(\text{odds}) = \ln(p/(1-p))$$

$$\text{logit}(p) = \ln(p/(1-p)) = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + \dots + b_k X_k$$

Here, p is the probability of the presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than those that minimize the sum of squared errors (like in ordinary regression).

You may ask, why take a log? For the sake of simplicity, let's just say that this is one of the best mathematical ways to replicate a step function. Refer to the following [*Figure 1.4*](#).

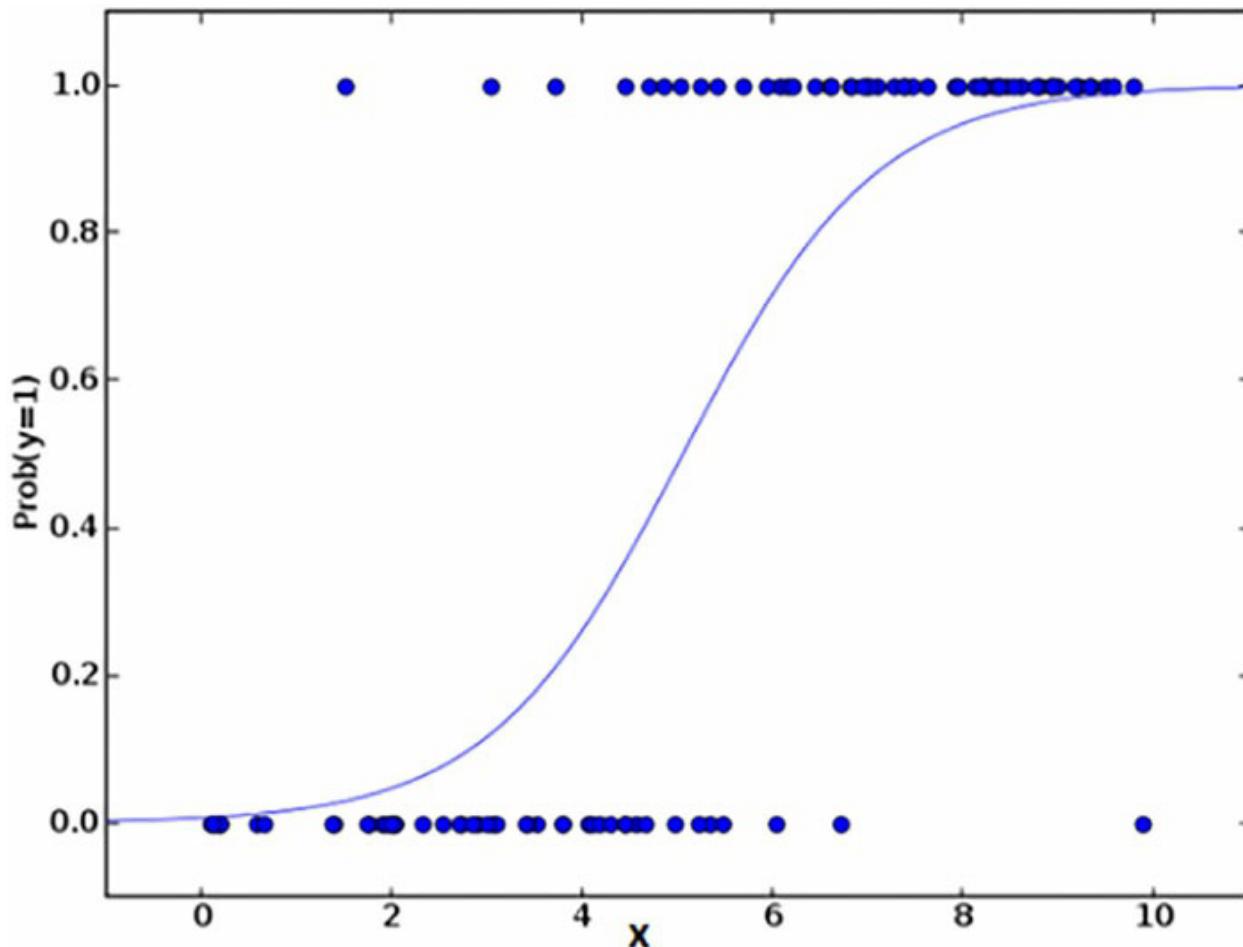


Figure 1.4: Probability of the presence of the characteristic of interest

Decision Tree: This is one of my favorite algorithms and is used quite frequently. It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets based on the most significant attributes/independent variables to make as distinct groups as possible. Refer to the following [Figure 1.5](#).

Dependent variable: PLAY

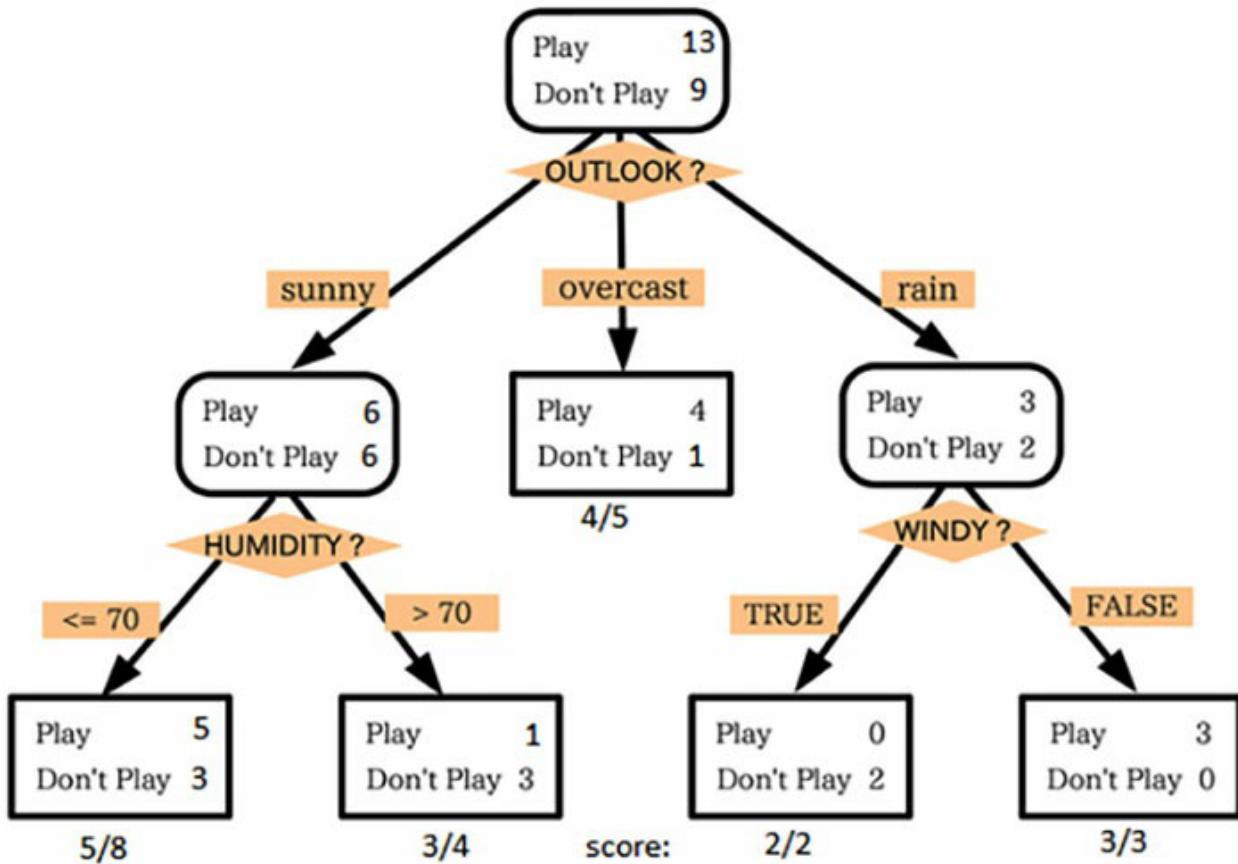


Figure 1.5: Dependent variable
Source: statsexchange

In the preceding image, you can see that population is classified into four different groups based on multiple attributes to identify ‘whether they will play’. To split the population into different heterogeneous groups, it uses techniques like Gini, Information Gain, Chi-square and entropy.

The best way to understand how decision tree works is to play Jezzball, a classic game from Microsoft, as shown in the following figure. Essentially, you have a room with moving walls, and you need to create walls such that maximum area gets cleared off without the balls. Refer the *Figure 1.6*.

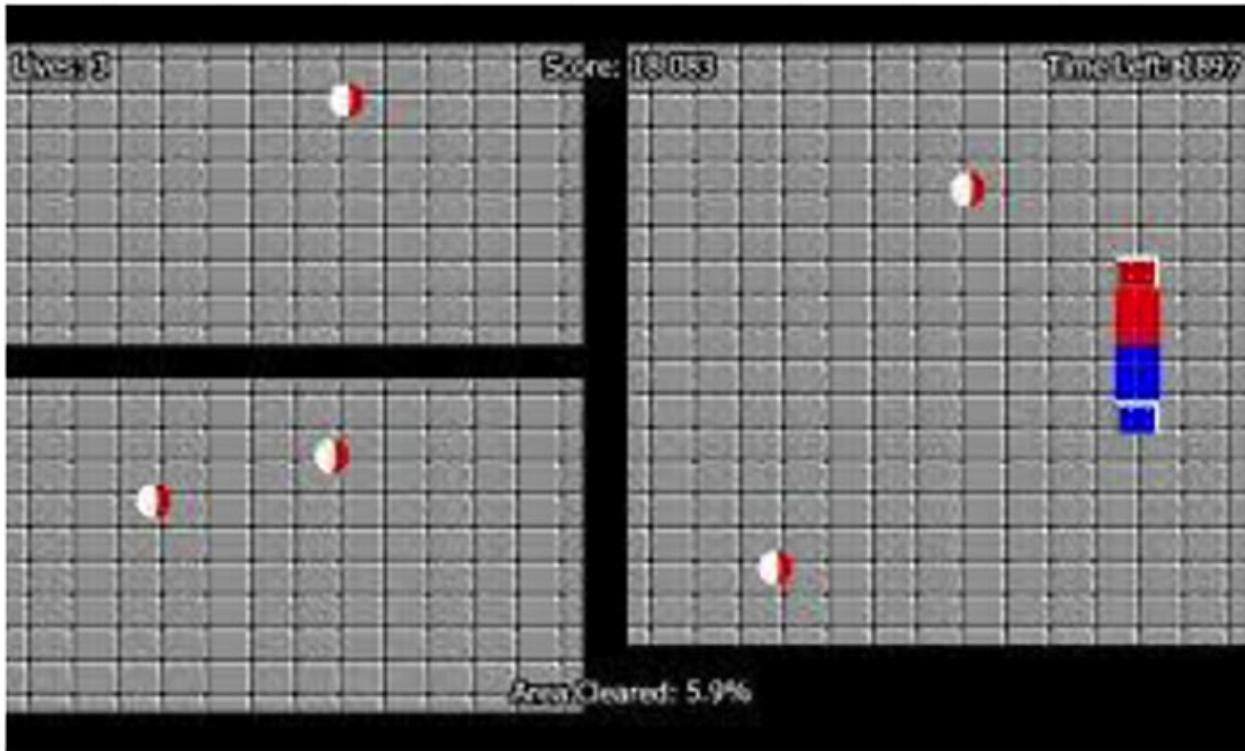


Figure 1.6: Jezzball

So, every time you split the room with a wall, you are trying to create two different populations in the same room. Decision trees work in a similar fashion by dividing a population into groups.

kNN (k-Nearest Neighbors): It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K-nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is the most common among its K -nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. The first three functions are used for continuous function, and the fourth one (Hamming) is used for categorical variables. If $K = 1$, the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling.

KNN can easily be mapped to our real lives. If you want to learn about a person you have no information about, you might like to find out

about their close friends and the circles they move in, and gain access to information about them.

The following are the things to consider before selecting kNN:

- KNN is computationally expensive
- Variables should be normalized, else higher range variables can bias it
- Works on pre-processing stage more before going for kNN, like an outlier or noise removal

K-Means: It is a type of unsupervised algorithm that solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). Data points inside a cluster are homogeneous and heterogeneous to peer groups.

Remember figuring out shapes from ink blots? as shown in [*Figure 1.7*](#), k means algorithm is somewhat similar to this activity. You look at the shape and spread to decipher how many different clusters/population is present.

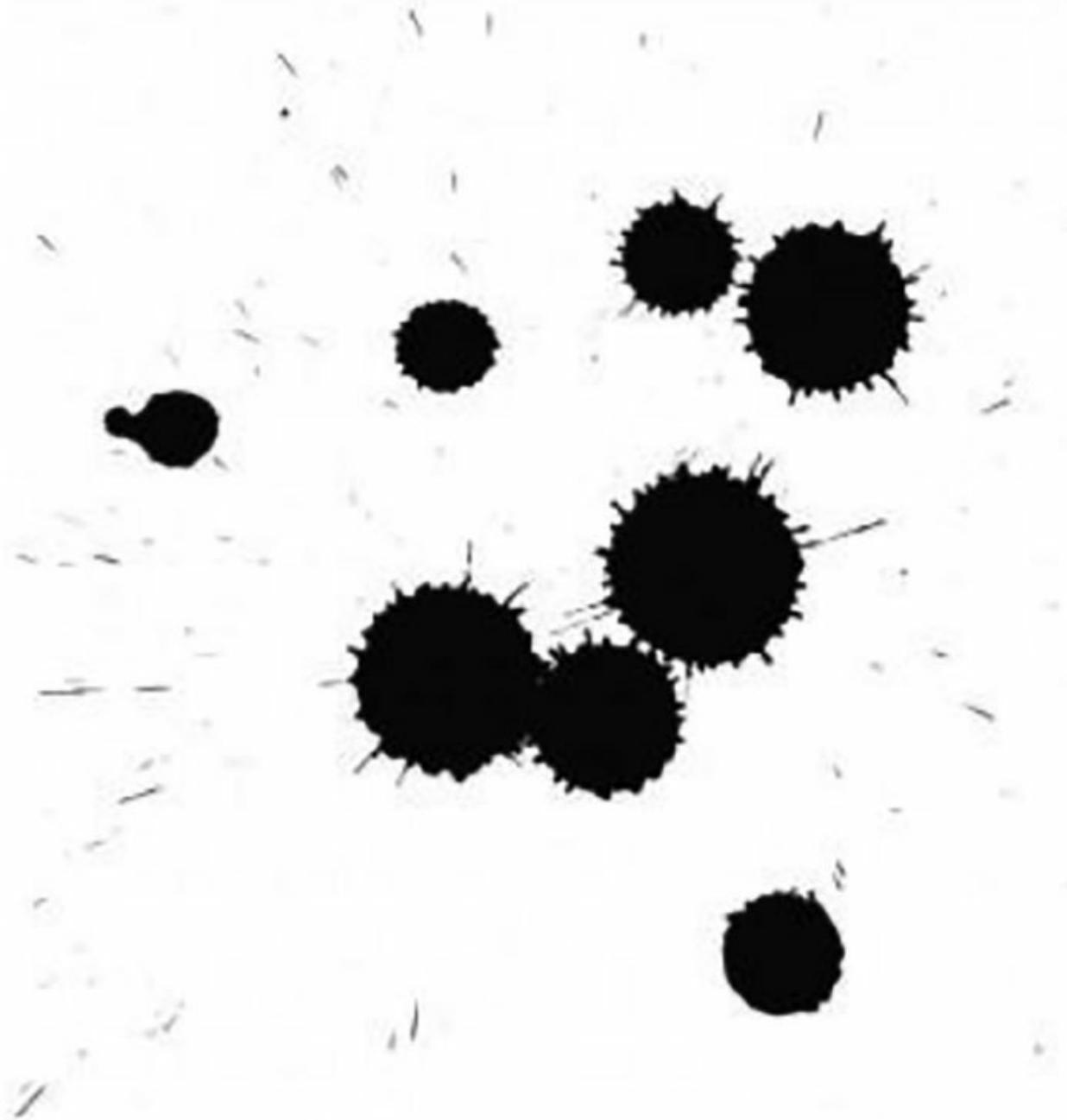


Figure 1.7: Cluster / population present

Here's how K-means forms cluster:

1. K-means picks k number of points for each cluster known as centroids.
2. Each data point forms a cluster with the closest centroids, i.e., k clusters.
3. It finds the centroid of each cluster based on the existing cluster members. Here, we have new centroids.

- As we have new centroids, repeat step 2 and 3. Find the closest distance for each data point from new centroids and get associated with new k-clusters. Repeat this process until convergence occurs, i.e., centroids do not change.

Let's look at how to determine the value of K.

In K-means, we have clusters, and each cluster has its own centroid. The sum of the square of the difference between centroid and the data points within a cluster constitutes within the sum of the square value for that cluster. Also, when the sum of square values for all the clusters are added, it equals the total within the sum of square value for the cluster solution.

We know that as the number of cluster increases, this value keeps on decreasing. However, if you plot the result, you may see that the sum of squared distance decreases sharply up to some value of k and then more slowly after that. Here, we can find the optimal number of clusters. Refer the following [Figure 1.8](#).

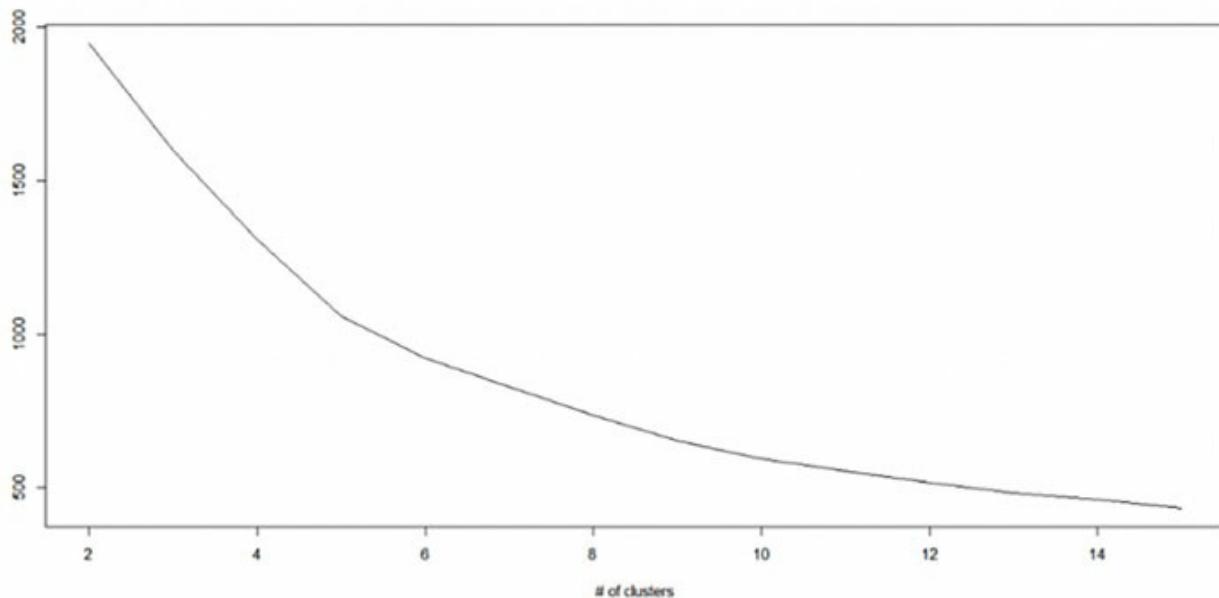


Figure 1.8: Optimal number of clusters

Dimensionality Reduction Algorithms: In the last 4-5 years, there has been an exponential increase in data capturing at all stages. Corporates/government agencies/research organizations are not only identifying new sources but also capturing data in great detail.

For example, e-commerce companies are capturing more details about customers, like their demographics, web crawling history, what they like or dislike, their purchase history, feedback and much more, to give them personalized attention.

As a data scientist, the data we are offered consists of many features; this sounds good for building good robust model, but there is a challenge. How would you identify highly significant variable(s) out of 1000 or 2000 variables? In such cases, dimensionality reduction algorithm helps, along with algorithms like Decision Tree, Random Forest, PCA, Factor Analysis, Identifying based on correlation matrix, missing value ratio and others.

XGBoost: This is another classic gradient boosting algorithm that's known to be the decisive choice between winning and losing in some Kaggle competitions.

XGBoost has immensely high predictive power, which makes it the best choice for accuracy in events as it possesses both linear model and the tree learning algorithm, making it almost 10x faster than existing gradient booster techniques.

The support includes various objective functions, like regression, classification and ranking.

One of the most interesting things about the XGBoost is that it is also called a regularized boosting technique. It helps reduce overfit modelling and has a massive support for a range of languages, such as Scala, Java, R, Python, Julia and C++.

It supports distributed and widespread training on many machines that encompass GCE, AWS, Azure and Yarn clusters. XGBoost can also be integrated with Spark, Flink and other cloud dataflow systems with a built-in cross validation at each iteration of the boosting process.

3. State the challenges faced while implementing the algorithms of machine learning.

Ans. Data unavailability: Despite data abundance, there is shortage of data for various domains, like healthcare and agriculture.

Complex data: With better internet connectivity and social media, we are generating data every second, but this data is not useful for machine learning. There is a lot of data manipulation, extraction and

transformation required to use these data. Data engineering teams play a crucial role in streamlining the data for ML usage.

Computational requirements: In deep learning, we have image datasets, which have millions of images sometimes. To train a model on all the images is a tedious task; it may take months. Transfer learning helps us reduce the burden of training, but there are limitations of transfer learning, which can only be overcome by better computation machines.

Overfitting/Underfitting: Overfitting and underfitting both impact the performance of a model. You will have better accuracy while training the model in both cases, but the accuracy reduces drastically when the model is tested in a real environment.

Overfitting happens when our model is too complex, and underfitting is the opposite; it happens when our model is too simple to learn something from the data.

Feature extraction: The data we received for the model is not actually used for it. We need domain knowledge to debunk the data and prefer only the features that are useful for the model. You may have 500 features but only 80 that are good to be used. To filter out these features, you need knowledge of machine learning models, including the domain that you are working for.

4. Describe the working of the machine learning algorithm.

Ans. The machine learning algorithm is broadly divided into two parts: supervised algorithm and unsupervised algorithm. We can further divide the supervised algorithm based on the type of problem, like regression and classification.

Before implementing the algorithm, we need to get the data ready, understand the data, remove all outliers, clean the data, perform imputation, normalization, divide and split the data into a train and a test set.

According to the problem, we can apply the algorithm. Suppose we need to predict the price of the land. Since this is a regression problem, we need to find all the important features of the land, like location, size and nearby market, based on all the important features. We can use linear regression or random forest to predict the price of the land.

Suppose we have a problem that needs to be answered with “YES” or “NO”. This is typically a classification problem. Here, we can use logistic regression, support vector machines, and decision trees to decide the output. We can also go for deep learning in case of image classification.

In unsupervised learning, we usually have the problem of clustering, like in social and retail domain problems. We need to cluster the users who have similar kind of behavior or purchase history; this help the company determine whom they should provide the offer. Most used algorithms are k nearest neighbor, principal vector composition and singular vector decomposition.

5. Elaborate on any two applications of machine learning for industries.

Ans. For better understanding of machine learning, let's consider the practical usage of ML in different domains of the industry:

Recommendations: Machine learning is widely used by e-commerce and entertainment companies like Amazon and Netflix for product recommendation to users. Whenever we search for some product on Amazon, we start getting advertisements for the same product while browsing the internet using the same browser; this is because of machine learning.

Google understands user interest using various machine learning algorithms and suggests the product as per customer interest.

Similarly, when we use Netflix, we find some recommendations for entertainment series, movies, and such; this is also done with the help of machine learning.

Sentimental Analysis: Applications of sentiment analysis include campaign monitoring, brand monitoring, stock market analysis and compliance monitoring. Let's understand sentiment analysis with the simplest implementation: using a word list with scores ranging from +5 (positive) to -5 (negative). For this, let's use the AFINN scored word list. Say one of your customers wrote, “I loved the product, but the packaging was not good”. In the AFINN word list, ‘loved’ and ‘not good’ have +3 and -2 scores, respectively. If you combine these two scores, you will get +1. This means the user sentiment was mildly positive. Again, this is the simplest example of sentiment analysis.

Complex models combine NLP and machine learning algorithms to analyze large pieces of text. According to Amadeus IT group, 90% of American travelers with smartphones share their photos and travel experience on social media and review services. TripAdvisor gets about 280 reviews from travelers every minute. With a large pool of valuable data from 390 million unique visitors and 435 million reviews, it analyses this information to enhance its service. Machine learning techniques at TripAdvisor focus on analyzing brand-related reviews.

6. Differentiate between machine learning and deep learning.

Ans. While there are many differences between these two subsets of artificial intelligence, here are five of the most important ones:

- **Human Intervention:** Machine learning requires more human intervention to get results. On the other hand, deep learning is more complex to set up but requires minimal human intervention thereafter.
- **Hardware:** Machine learning programs tend to be less complex than deep learning algorithms and can often run on conventional computers, but deep learning systems require far more powerful hardware and resources. This demand for power has meant increased use of graphical processing units. GPUs are useful for their high bandwidth memory and the ability to hide latency (delays) in memory transfer due to thread parallelism (the ability of many operations to run efficiently at the same time).
- **Time:** Machine learning systems can be set up and operate quickly, but they may be limited in the power of their results. Deep learning systems take more time to set up but can generate results instantaneously (although the quality is likely to improve over time as more data becomes available).
- **Approach:** Machine learning tends to require structured data and uses traditional algorithms like linear regression. Deep learning employs neural networks and is built to accommodate large volumes of unstructured data.
- **Applications:** Machine learning is already in use in your email inbox, bank, and doctor's office. Deep learning technology enables more complex and autonomous programs, like self-driving cars or robots that perform surgeries.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 2

Python Basics for ML

Introduction

Python is a language to fall in love with when it comes to machine learning. Since its inception in 1991, it has been gaining popularity for applications in various domains, including machine learning. Its library support is one of the reasons why Python is popular. In recent years, Python has significantly improved its support for libraries suitable for various data analysis and machine learning applications. Python's combination of general-purpose programming language and data analysis support has increased its popularity two-fold.

After looking at machine learning in general, it's time to take a quick look at the basics of Python. In the previous chapter, you learned why machine learning is important and how widely it is used. This chapter takes a quick look at Python from the point of view of a programming language, covering the basics. It introduces two basic popular IDEs for writing Python code: Jupyter and Spyder. To look at machine learning in depth, one needs to know the tools and languages properly. The chapter covers basic data structures, input output commands and file handling using Python. Knowing the basics of Python is important as it is the basic language used in many advanced machine learning tools, like TensorFlow and Keras.

Structure

In this chapter, we will discuss the following topics:

- Spyder IDE
- Jupiter Notebook
- Python: Input and Output
- Functions and Modules
- Class handling

- Exceptions handling
- File handling
- String Functions
- Working with modules in Python
- Simple Input-Output commands
- Logical Statements
- Loop and Control Structures
- Functions and modules

Objectives

The objective of this chapter is to introduce the code editors and IDE in Python. The basic structure of the Python program will also be discussed. You will be able to learn the basic python statements, conditional statements, and control structures. Additionally, you will be able to write functions and create modules, and you will know how to use basic exception handling effectively after completing this chapter. This chapter will also take you through basic file handling.

Spyder IDE

Spyder is the scientific environment written for executing Python scripts. It is written in python by and for scientists, researchers, data analysts and engineers. It is a powerful tool to write, execute and debug Python programs easily. It has many built-in features that make programmers' job easy. The environment, as shown in [*Figure 2.1*](#), has several built-in features like project explorer, coding window, variable explorer, Python console and visualization tool.

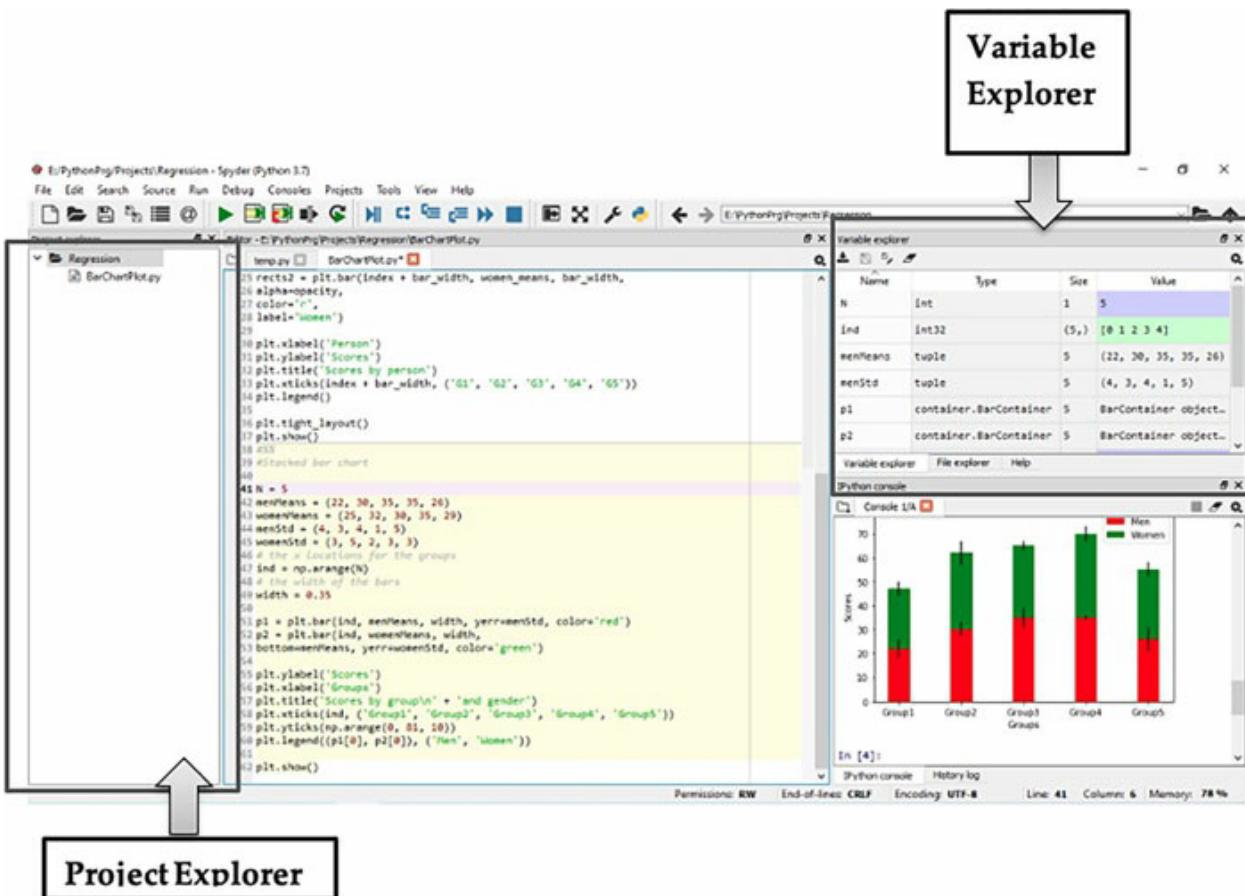


Figure 2.1: Spyder IDE – Project and Variable Explorer

- Project Explorer: As shown in [Figure 2.1](#), one can start a new project in Python, and all the related Python files are available for use under one common resource. The Spyder supports version control using Git.
- Variable Explorer: The variable explorer helps to keep track of variables used in the project. The variable values, their data types and dimensions are visible through this. Also, we can directly open the file without writing any command in coding editor, by double-clicking on the file.
- Code Editor: It helps in writing the code with ease. The editor supports error checking and highlights the syntax. It provides code analysis then and there, as shown in [Figure 2.2](#):

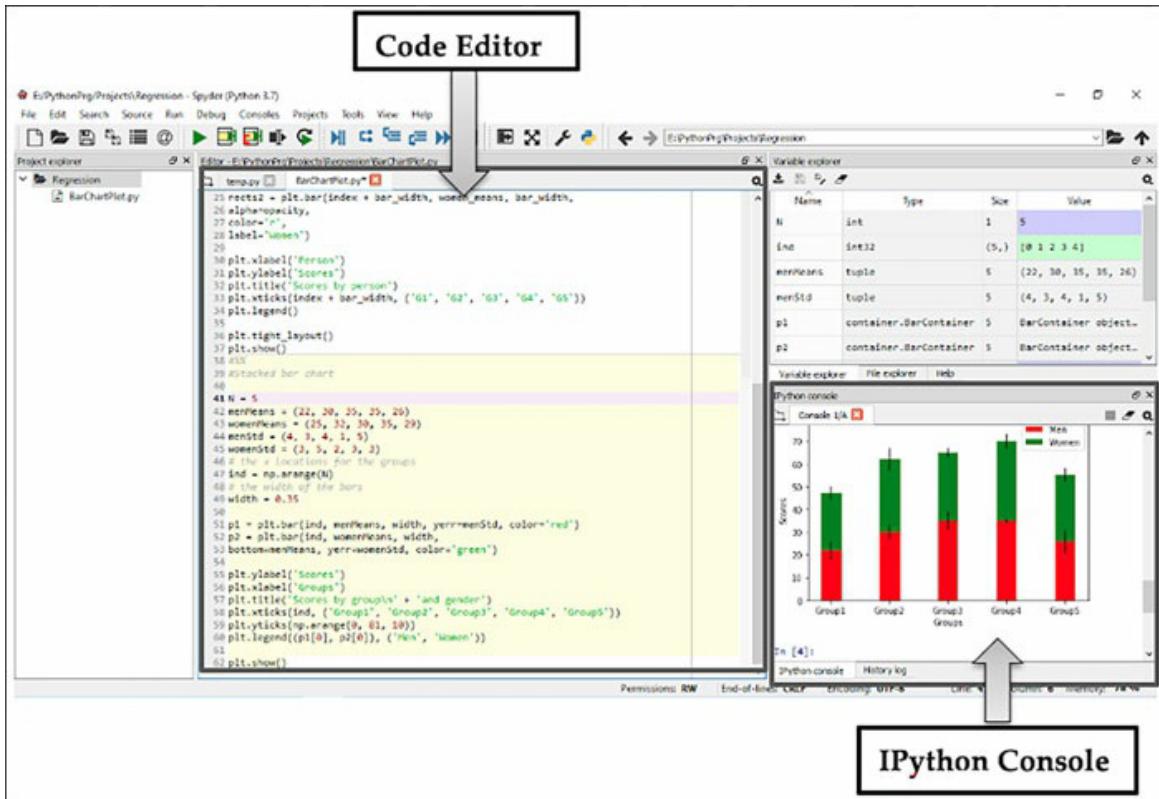


Figure 2.2: Spyder IDE – Code Editor and IPython Console

- **IPython Console:** This is the console that helps execute Python commands and visualize the output. A single independent command or the complete Python program can be executed using this.

Jupyter Notebook

This is one of the most popular editors among beginners for writing and executing Python scripts. These notebooks can be shared easily and are easy to code and understand. The notebook has executable cells, as shown in [Figure 2.3](#), which can execute the Python commands typed in. Jupyter notebook has cell like structure where you can write and execute the command.

The cell type can be code cell, Raw NBconvert or markdown cell. The code cell is used to write Python code, and markdown cell can be used for explaining the code cell. One can use HTML tags to display the details by highlighting the important aspects. The markdown cell is like documenting the comments. The notebook keeps a track of the sequence of execution of Python code. The cell that is going to be executed is highlighted. Refer to the

following figure:

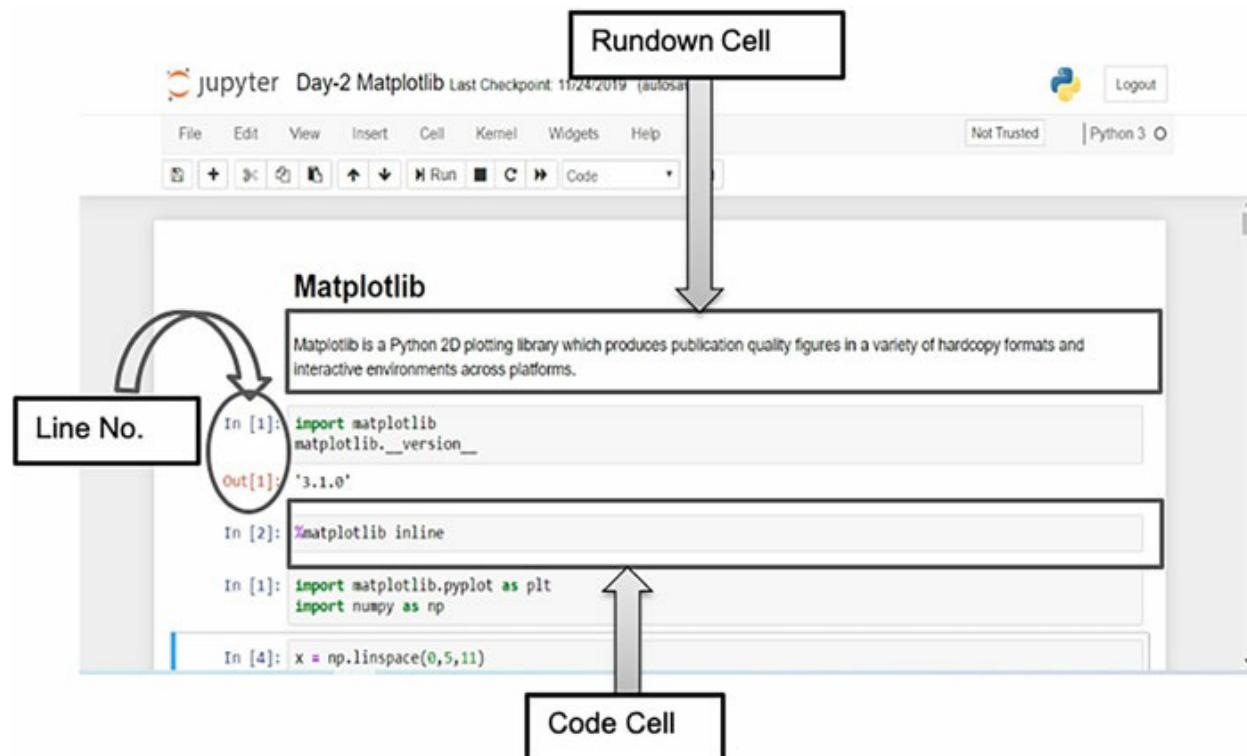


Figure 2.3: Jupyter Notebook

One important advantage of using the Jupyter notebook is that the output is preserved and can be seen later whenever required. As a beginner, the Jupyter notebook acts as a visual guide for the programmer for giving trainings and for efficient learning.

Python: Input and Output Commands

Let's start revising the basics of Python. Any language is rich because of its data structure support. Python provides additional data structures as compared to its counterparts, which makes it suitable for use with machine learning applications. [*Figure 2.4*](#) highlights the important data structures of Python. There are two types of data structures:

- **Primitive data structures:** These data structures are general and common data types that are also part of other programming languages.
- **Non-primitive data structures:** These are special data structures that are native to Python and are developed keeping in mind their use in the data analysis domain.

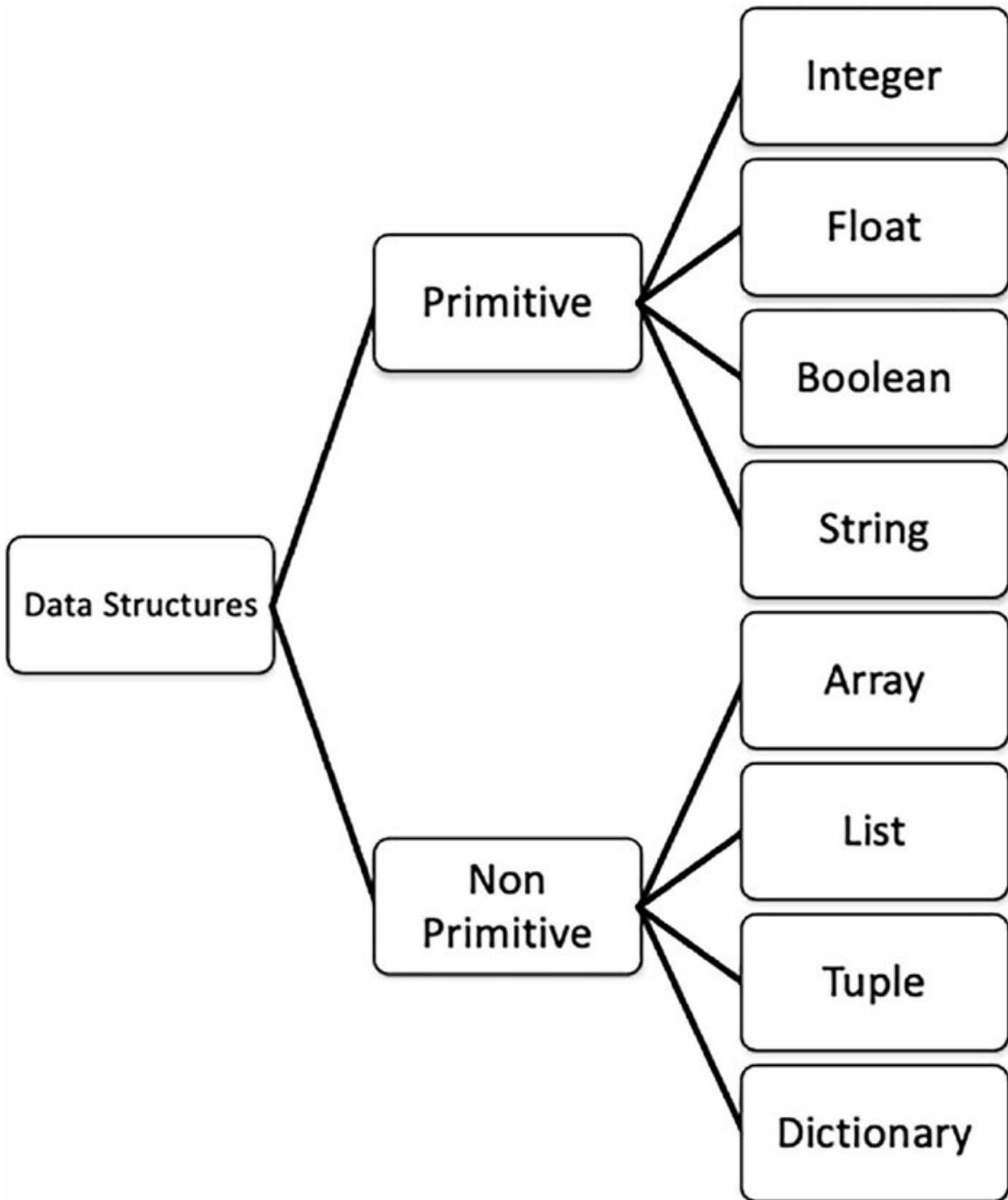


Figure 2.4: Data Structures in Python

The following code shows in Example 1 how to create List, Tuple, and Dictionary.

Example 1:

```
>>> x = [1,'hello', (3 + 2j)] #List creation  
>>> x
```

```
[1, 'hello', (3+2j)]
>>> x = (1,2,3)      #Tuple creation
>>> x[1:]
(2, 3)
>>> d = {1 : 'hello', 'two' : 42, 'blah' : [1,2,3]} #Dictionary
>>> d
{1: 'hello', 'two': 42, 'blah': [1, 2, 3]}
>>> d['blah']
[1, 2, 3]
```

The basic command used to display the output is the **print()** command as seen in Example 2.

Example 2:

```
# print integer and float value
>>>print("Geeks : % 2d, Portal : % .2f" %(1, 05.333))
# Printing String
>>>print "Welcome to Python"
```

Python has a rich set of functions for performing mundane tasks. For reading input from user, Python uses the **input()** function.

Example 3:

```
>>>choice=input("Which fruit do you like")
Apple
>>>print(choice)
Apple
```

Care must be taken while reading numbers. The **input()** function treats all values as string by default. While reading numbers, these values can be converted in proper data type using the **int()** and **float()** functions.

Logical Statements

Like any other programming language, Python provides support for relational and logical operators and logical statements.

The relational operators supported are **<**, **>**, **<=**, **>=**, **!=**, **==**.

The logical operators are **AND**, **OR**, **NOT**.

The logical statement **if, else** is popular while taking the decisions. The Python syntax is as follows:

```
if condition :
    Statements
else:
    Statements
```

Alternatively, to be more decisive, the statement can be used in cascade. One thing to remember here is that Python does not provide any block marking operator. The extra spaces are used to mark or group the statements. Any unnecessary space will be termed as an error in Python.

```
if condition :  
    statements  
elif condition:  
    statements  
else:  
    statements
```

Example 4:

```
>>>a,b=10,20  
>>>if a>b :  
    print(a-b)  
else :  
    print(b-a)  
10
```

Example 5:

```
>>>a = 200  
>>>b = 33  
>>>if b > a:  
    print("b is greater than a")  
elif a == b:  
    print("a and b are equal")  
else:  
    print("a is greater than b")
```

The preceding code will print - **a is greater than b.**

Loop and Control Statements

To aid in repetitive execution of statements in programming, loop statements are used based on conditions. Mainly, there are two loop statements supported in Python: **for loop** and **while loop**.

Example 6:

```
>>>for x in range(1,10):  
    print(x)
```

The **for** statement is used widely to iterate over strings, tuple and lists.

Example 7:

```
>>>a, b, n = 0, 1, 10  
>>>sum=0; count=1
```

```
>>>while (count < n):
    count=count+1
    a=b; b=sum
    sum=a+b
    print(' ',sum)
```

while is used to repeat statements as per requirement. The code in example 7 will print the famous Fibonacci series with 10 terms.

We can use the **break** statement to stop iterating in loop or exit the loop based on some condition. The **continue** statement is used to bypass a few statements from the loop and continue a new iteration.

Example 8:

```
>>>for num in range (1,10):
    print(num)
    if(num ==5):
        break;
```

Example 9:

```
>>>for num in range (1,10):
    if(num ==5):
        continue;
    print(num)
```

The output of example 8 is 12345; the **for** loop is halted after the value reaches 5. On the contrary, the output of example 9 is 1234678910; the **print(num)** statement is not executed for 5 because of the **continue** statement.

Functions and Modules

Functions are the self-contained block of statements that act like a program that performs specific tasks. The functions include a set of statements that are executed whenever we refer to the function in the program. A function can be defined using the **def** keyword in Python.

The generalized syntax for function definition in Python is as follows:

```
def function_name(list of parameters) :
    statements
```

Example 10:

```
>>>def max(x,y) :
    if x > y :
        return x
    else:
```

```
    return y
>>>max(10, 20)
20
```

The function in example 10 takes two numbers as parameters and compares them. The function returns whichever value is greater.

Example 11:

```
>>>def add(a,b):
    print ("ADDING %d + %d" % (a, b))
    return a + b
>>>def subtract(a,b):
    print ("SUBTRACTING %d - %d" % (a, b))
    return a - b
>>>print(add(10,subtract(30,10)))
SUBTRACTING 30 - 10
ADDING 10 + 20
30
```

We may pass function as a parameter to other functions. In example 11, the **subtract()** function is passed as a parameter to the **add()** function. Functions are very useful as the work can be organized as a series of functions.

A module is a collection of functions in short library. Python modules can be saved as **.py** files and can be used within any other regular code files. Here are the steps to create a module:

1. Create a Python file (with **.py** extension) with only function definitions as per requirement.
2. Save the file and add to the path. Now this can be called as module.
3. Create the reference of the module using the **import** statement.
4. Use the functions from the module using the syntax **modulename.functionname**

Example 12 shows how sample library files are created and *Example 13* shows how to use the library functions.

Example 12: MyLib.py

```
def CalDa(basic):
    return(basic*1.1)
def CalHra(basic):
    return(basic*0.3)
def CalIncrement(basic,da):
    return((basic+da)*0.3)
```

Example 13:

```
>>>import MyLib as ml #Create a reference of the library
>>>basic=int(input('Enter the basic :'))
Enter the basic :35000
>>>salary=basic+ ml.CalDa(basic)+ 
ml.CalHra(basic)+ ml.CalIncrement(basic,ml.CalDa(basic))
>>>print('New salary with Increment is ',salary)
New salary with Increment is 106050.0
```

In *Example 12*, a simple **MyLib.py** file is created with necessary functions definitions. *Example 13* illustrates the use of this module for the calculation of the salary. Python uses many built-in and additional modules for the data analysis. These modules are the backbone of Python Data Analysis and are used for the application of machine learning algorithms. Creating the libraries and using them for performing our task is most easy way to handle machine learning programs. The following [table 2.1](#) lists the various ways to include a module in a Python file:

Method	Use
<code>import mymodule</code>	Imports all the elements that module exports
<code>from mymodule import x</code>	Imports x from mymodule right into this namespace
<code>from mymodule import *</code>	Imports all elements of mymodule into this namespace

Table 2.1: Ways to include a module in a Python file

Python allows writing the anonymous function using lambda operator. A lambda expression returns a function object. The body can only be a simple expression, not complex statements.

Example 14:

```
>>> f = lambda x,y : x + y
>>> f(2,3)
5
>>> lst = ['one', lambda x : x * x, 3]
>>> lst[1](4)
16
```

Machine learning algorithms use anonymous functions quite regularly for data conversion or transformation.

Class Handling

Python supports object-oriented programming by supporting the creation and handling of classes. Python is an object-oriented programming language wherein everything that we create is treated as objects. A class is used to create objects. It consists of variables and functions on these variables to perform various tasks. A class can be defined using the **class** keyword.

Example 15:

```
>>>class MyClass:  
    def f(self):  
        return 'Hello World'  
>>> x=MyClass() #Create instance of class  
>>> x.f() #Call the method
```

Here, a simple class is defined with only one function, which displays ‘Hello World’.

Example 16:

```
>>>class MyPerson:  
    def __init__(self,name,age):  
        self.name=name  
        self.age=age  
    def printName(self):  
        print("My name is "+self.name)
```

Here, the class is defined with two functions and two variables. **__init__** is a special function called constructor. When the instance of the class is created, the constructor is called by default. The **self** parameter is a reference to the current instance of the class and is used to access variables that belong to the class. It does not have to be named self; you can call it whatever you like, but it has to be the first parameter of any function in the class.

Exception Handling

It’s a good programming practice to handle an error gently inside the program. This process of error handling embedded in programs is called exception handling. Like other popular languages, Python also supports error handling using **try** and **except** statements as seen in example 17.

Example 17:

```
>>>try:  
    print(a)  
except:
```

```
    print("Variable not defined exception has occurred.")
```

Here, as the variable **a** is not defined, the code will throw an error and the statement in the **except** block will be executed as seen in example 18.

Example 18:

```
>>>try:  
    a=0  
    tot=145  
    c=tot/a  
    print(c)  
except ZeroDivisionError:  
    print("Divide by Zero error.")
```

We can use system-defined standard errors to raise an exception, and then statements after that will be executed. Some useful exceptions are **IndexError**, **ImportError**, **IOError**, **ZeroDivisionError**, and **TypeError**.

File Handling

Python does not explicitly require any library for reading or writing files. It has built-in functions to do so. The steps to read and display file contents are listed here:

1. Open file in read mode:

```
>>>f=open("MyFile.txt","r")
```

2. Read from file:

```
>>> data=f.read()
```

3. Display data read:

```
>>> print(data)
```

4. Close the file:

```
>>>f.close()
```

Here, the **open()** method accepts two arguments as filename and mode in which file is required to be open for which the file is required to be opened. For reading from file, the mode is “r”. The steps followed for writing into the file are as follows:

1. Open file in read mode:

```
>>>f=open("MyFile.txt","w")
```

2. Read from file:

```

>>> data=f.write(s)
3. Display data read:
    >>> print(data)
4. Close the file:
    >>>f.close()

```

Here, in the **open()** method, mode can be “w” or “w+”. In “w+” if file does not exist, then it is created before the string is written. While writing in the file, the contents already present in the file are overwritten. The following [table 2.2](#) summarizes the functions used while reading the file:

Method	Use
f1 = open('filename', 'r')	Opens the file ‘data’ for input
s = f1.read()	Reads whole file into one string
s = f1.readline()	Reads only one line
L = f1.readlines()	Returns a list of line strings

Table 2.2: Functions for Reading the file

The following [table 2.3](#) summarizes the functions used for writing in the file:

Method	Use
f = open('filename', 'w')	Opens the file ‘data’ for writing
f.write(s)	Writes the string S to file
f.writelines(L)	Writes each of the strings in list L to file
f.close()	Closes the file

Table 2.3: Functions for writing the file

Here, the mode can be “r”, “w”, “w+”, “a” and “a+”. The modes ”a” and ”a+” are used to append or add data in the existing file or in a new file.

Example 19:

```

>>>f=open("Numbers.txt","w+")
>>>for num in range(10,20):
    f.writeline("The number is %d",%dnum)
>>>f.close()
>>>f=open("Numbers.txt","r")
>>>f=f.readlines()

```

```
>>>for line in f:
    print(line)
>>>f.close()
```

Here, the file is opened in write mode and numbers from 10 to 20 are written in it. The file is closed and again opened in read mode to read the data. The **for** statement is used to display the data line by line.

String functions

Strings are an important part of any data. The data generally collected consists of maximum categorical data and some numeric data. This makes string handling an important part in machine learning. Let's look at a few important string functions that are used repeatedly for data manipulation.

Consider a string, as follows:

```
>>> s=" Welcome here "
```

Function	Description	Output
s.lower()	Converts to lowercase	welcome here
s.upper()	Converts to uppercase	WELCOME HERE
s.capitalize()	Capitalizes each word	Welcome Here
s.lstrip()	Removes leading spaces	Welcome here
s.rstrip()	Removes trailing spaces	Welcome here
s.strip()	Removes all spaces	Welcome here
s.split(sep,maxSplit)	Splits the string into words depending on separator	['Welcome', 'here']

Table 2.4: Summary of string functions

A few more advanced functions on strings are indicated in table as follows:

Function	Description
s.count(value, start, end)	Counts the occurrences of the value or string
s.find(value,start,end) s.index(value,start,end)	Finds the occurrences of the value or string and returns position or returns -1 if not found

	Index raises an exception when value is not found
s.rfind(value, start, end) s.rindex(value, start, end)	Finds the occurrences of the value or string and returns the last position, or returns -1 if not found Index raises an exception when value is not found
s.replace(oldval, newval, count)	Replaces an old substring with new string; count is optional, indicating how many instances to replace

Table 2.5: Advanced String Functions

Example 20 shows how string functions can be applied.

Example 20:

```
>>>s="Strings and Strings here and there!"  
>>>s.count("Strings",0,30)  
2  
>>> s.find("here",0,40)  
20  
>>> s.rfind("Strings",0,40)  
12  
>>> s.replace ('there','here')  
Strings and Strings here and here!
```

String and text manipulation is an important part of many machine learning applications. The discussed string functions help complete many important steps of the entire machine learning model, like preprocessing of data.

Conclusion

Python is the most popular language for machine learning because of its simple syntax and rich support for library. This chapter focused on the basics of the language required to perform any machine learning task. Python has four basic data structures and three advanced data structures: List, Tuple and Dictionary. Python supports traditional logical statements like **if .. else** and control statements like **for** and **while**. Functions add modularity to the Python code and can be prewritten and utilized many times using modules. Small anonymous functions can be declared using keyword lambda and used effectively whenever required. Basic file handling is provided directly in Python. The support for handling strings using various functions has made

the language more flexible for text processing.

This chapter Helps you to develop simple code to perform many machine learning tasks. It empowered you to use any Python library easily in your code. The text processing, which is an integral part of any machine learning application, is easy because of string functions discussed here.

With the basics of machine learning and the Python language, your journey toward the algorithms in this domain starts. The next chapter will take you through the basic machine learning tasks and algorithms, building a strong foundation for you.

Questions and Answers

1. Why is Python preferred for machine learning?

Ans. Easy and fast data validation

The job of machine learning is to identify patterns in data. An ML engineer is answerable for harnessing, refining, processing, cleaning, sorting out and deriving insights from data to create clever algorithms. Python is easy, but the topics of linear algebra or calculus can be perplexing; they require maximum efforts to solve. Python can be executed quickly, which allows ML engineers to approve an idea immediately.

Different libraries and frameworks

Python is already well-known, so it has many libraries and frameworks that can be utilized by engineers. These libraries and frameworks help in saving time, which makes Python significantly more popular.

Code readability

Since machine learning includes an authentic knot of math, sometimes very troublesome and unobvious, the readability of the code (also outside libraries) is significant if we need to succeed. Developers should think not about how to write but what to write, all things considered.

Python developers are excited about making code that is not difficult to read. Moreover, this language is extremely strict about appropriate spaces. Another of Python's advantages is its multi-paradigm nature, which empowers engineers to be more adaptable and approach issues in the simplest way possible.

Low-entry barrier

There is an overall shortage of software engineers. Python is not difficult to get familiar with, so the entry barrier is low. What's the significance here? That more data scientists can become experts rapidly and thus, they can engage in ML projects. Python is fundamentally the same as the English language, which makes learning it simpler. Because of its easy phrase structure, you can unhesitatingly work with complex systems.

Portable and extensible

This is a significant reason why Python is so mainstream in machine learning. Several cross-language tasks can be performed effectively on Python due to its portable and extensible nature. There are numerous data scientists who favor utilizing **Graphics Processing Units (GPUs)** for training their ML models on their own machines, and the versatile idea of Python is appropriate for this.

2. Explain the non-primitive data structures in Python.

Ans. In contrast to primitive data structures, non-primitive data types store not only values but a collection of values in different formats. Let us look at non-primitive data structures in Python:

Lists: This is the most versatile data structure in Python and is written as a list of comma-separated elements enclosed within square brackets. A list can consist of both heterogeneous and homogeneous elements. Some of the methods applicable on a list are `index()`, `append()`, `extend()`, `insert()`, `remove()`, and `pop()`. Lists are mutable, which means their content can be changed while keeping the identity intact.

Syntax:

```
List1= ['a', 1, 2]
```

Tuples: Tuples are similar to Lists but are immutable. Also, unlike Lists, Tuples are declared within parentheses instead of square brackets. The feature of immutability denotes that once an element has been defined in a Tuple, it cannot be deleted, reassigned or edited. It ensures that the declared values of the data structure are not manipulated or overridden.

Syntax:

```
tuple1=(1, 2, 3)
```

Dictionaries: Dictionaries consist of key-value pairs. The ‘key’ identifies an item, and the ‘value’ stores the value of the item. A colon separates the key from its value. The items are separated by commas, with the entire dictionary enclosed within curly brackets. While keys are immutable (numbers, strings or tuples), the values can be of any type.

Syntax:

```
Dict={‘key1’:‘value1’, ‘key2’:‘value2’}
```

Sets: Sets are an unordered collection of unique elements. Like Lists, Sets are mutable and written within square brackets, but no two values can be the same. Set methods include `count()`, `index()`, `any()`, and `all()`.

Syntax:

```
Set1=set(List1)# List1 converted to set
```

3. Explain the If – Else statement with example.

Ans. If:

An “if statement” is written by using the `if` keyword.

Example:

```
a = 32
b = 100
if b > a:
    print("b is greater than a")
```

In this example, we use two variables, `a` and `b`, which are used as part of the `if` statement to test whether `b` is greater than `a`. As `a` is 32 and `b` is 100, we know that 100 is greater than 32, so we print “`b` is greater than `a`” on screen.

Elif:

The `elif` keyword is Python’s way of saying “*if the previous conditions were not true, then try this condition*”.

```
a = 32
b = 32
if b > a:
    print("b is greater than a")
elif a == b:
```

```
print("a and b are equal")
```

In this example, a is equal to b, so the first condition is not true, but the **elif** condition is true, so we print “**a and b are equal**” on screen.

else

The **else** keyword catches anything that isn’t caught by the preceding conditions.

```
a = 100
b = 32
if b > a:
    print ("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

In this example, a is greater than b, so the first condition is not true; also, the **elif** condition is not true, so we go to the **else** condition and print “**a is greater than b**” on screen.

You can also have an else without the **elif**:

```
a = 100
b = 32
if b > a:
    print ("b is greater than a")
else:
    print("b is not greater than a")
```

4. Explain the use of functions with example and generalized syntax.

Ans. A function is a block of code that only runs when it is called. You can pass data, known as parameters, into a function.

Creating a Function: In Python, a function is defined using the **def** keyword:

Example:

```
def my_function():
    print("Hello from a function")
```

Calling a Function: To call a function, use the function name, followed by parentheses:

Example:

```
def my_function():
    print("Hello from a function")
my_function()
```

Arguments: Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want; just separate them with a comma. The following example has a function with one argument (**fname**). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example:

```
def my_function(fname):
    print(fname + " Refsnes")
my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

5. How can you include Module in Python program?

Ans. Writing Modules: Writing a module is just like writing any other Python file. Let's start by writing a function to add/subtract two numbers in a file `calculation.py`:

```
def add(x,y):
    return (x+y)
def sub(x,y):
    return (x-y)
```

Working with Modules in Python

Modules enable you to split parts of your program into different files for easier maintenance and better performance. As a beginner, you start working with Python on the interpreter, and you start writing scripts when you need to write longer programs. As your program grows in size, you may want to split it into several files for easier maintenance and reusability of code. The solution to this is modules. You can define your most used functions in a module and import it instead of copying their definitions into different programs. A module can be imported by another program to use its functionality. This is how you can use the Python standard library as well. Simply put, a module is a file

consisting of Python code. It can define functions, classes, and variables, and can also include runnable code. Any Python file can be referenced as a module. A file containing Python code, for example, test.py, is called a module, and its name would be test. There are various methods of writing modules, but the simplest way is to create a file with a .py extension that contains functions and variables.

The import statement

To use the functionality in any module, you have to import it into your current program. You need to use the **import** keyword along with the desired module name. When an interpreter comes across an **import** statement, it imports the module to your current program. You can use the functions inside a module by using a dot(.) operator along with the module name. First, let's see how to use the standard library modules. In the example, math module is imported into the program so that you can use the **sqrt()** function defined in it.

import math

```
# you need to put this command, 'import' keyword along with
# the name of the module you want to import
num=4
print(math.sqrt(num))
# use dot operator to access sqrt() inside module "math"
```

Writing modules

Now that you have learned how to import a module in your program, it's time to write your own and use it in another program. Writing a module is just like writing any other Python file. Let's start by writing a function to add/subtract two numbers in a file **calculation.py**:

```
def add(x,y):
    return (x+y)
def sub(x,y):
    return (x-y)
import calculation
# importing calculation module
print(calculation.add(1,2))# calling function defined in
# add module.
```

If you execute **module_test.py**, you will see “3” as output. When the

interpreter came across the **import** statement, it imported the calculation module in your code, and then by using the dot operator, you were able to access the **add()** function.

There are more ways to import modules:

- from .. import statement
- from .. import * statement
- renaming the imported module

from .. import statement

The **from..import** statement allows you to import specific functions/variables from a module instead of importing everything. In the previous example, when you imported calculation into **module_test.py**, both the **add()** and **sub()** functions were imported. But what if you only needed the **add()** function in your code?

Example:

```
from calculation import add  
print(add(1,2))
```

In previous example, only the **add()** function is imported and used. Notice the use of **add()**? You can now access it directly without using the module name. You can import multiple attributes as well, separating them with commas in the import statement. Take a look at the following example:

```
from calculation import add,sub
```

from .. import * statement

You can import all attributes of a module using this statement. This will make all attributes of the imported module visible in your code.

Here is an example to illustrate the use of **from .. import *:**

```
from calculation import *  
print(add(1,2))  
print(sub(3,2))
```

Renaming the imported module

You can rename the module you are importing, which can be useful in cases when you want to give a more meaningful name to the module, or

when the module name is too large to use repeatedly. You can use the **as** keyword to rename it. The following example explains how to use it in your program:

```
import calculation as cal  
print(cal.add(1,2))
```

You saved yourself some time by renaming `calculation` as `cal`. Note that you can't use `calculation.add(1,2)` anymore, as `calculation` is no longer recognized in your program.

6. What is lambda function?

Ans. A lambda function in Python programming is an anonymous function or a function having no name. It is a small and restricted function having no more than one line. Just like a normal function, a Lambda function can have multiple arguments with one expression.

In Python, lambda expressions (or lambda forms) are used to construct anonymous functions. To do so, you will use the **lambda** keyword (just as you use **def** to define normal functions). Every anonymous function you define in Python will have three essential parts:

- The **lambda** keyword
- The parameters (or bound variables)
- The function body

A lambda function can have any number of parameters, but the function body can only contain **one** expression. Moreover, a lambda is written in a single line of code and can also be invoked immediately. You will see all this in action in the following examples.

Syntax and Examples

The formal syntax to write a lambda function is as follows:

```
Lambda x1,x2 : expression
```

Example:

```
adder=lambda x,y :x+y  
print(adder(2,3))
```

Output :

5

7. What is exception handling? Why it is important?

Ans. An exception can be defined as an unusual condition in a program resulting in the interruption in the flow of the program.

Whenever an exception occurs, the program stops the execution, so the code after that point is not executed. Therefore, an exception is the runtime errors that are unable to handle the Python script. An exception is a Python object that represents an error.

Python provides a way to handle exceptions so that the code can be executed without any interruption. If we do not handle the exceptions, the interpreter doesn't execute all the code after the exception.

Python has many built-in exceptions that enable our program to run without interruption and give the output.

The try-except statement

If the Python program contains suspicious code that may throw the exception, we must place that code in the try block. The try block must be followed with the except statement, which contains a block of code that will be executed if there is some exception in the try block.

Syntax:

```
try:  
#block of code  
except Exception1:  
#block of code  
except Exception2:  
#block of code  
#other code
```

Example:

```
try:  
a = 23  
b = 0  
c = a/b  
except:  
print("Can't divide with zero")
```

Output:

```
Can't divide with zero
```

8. Explain functions for reading and writing in files.

Ans. To open the file, use the built-in **open()** function. The **open()** function returns a file object, which has a **read()** method for reading the content of the file:

Example:

```
f= open("test.txt", "r")
print(f.read())
```

By default, the **read()** method returns the whole text, but you can also specify how many characters you want to return:

Example:

```
f= open("test.txt", "r")
print(f.read(5))
```

Return 5 characters of the file.

You can return one line using the **readline()** method:

Example:

```
f= open("test.txt", "r")
print(f.readline())
```

Write to an Existing File

To write to an existing file, you must add a parameter to the **open()** function:

“**a**” – Append; it will append to the end of the file.

“**w**” – Write; it will overwrite any existing content.

Example

Open the “**demofile2.txt**” file and append content to the file:

```
f= open("test2.txt", "a")
f.write("file1!")
f.close()
f= open("test2.txt", "r")
print(f.read())
```

Example

Open the “**test3.txt**” file and overwrite the content:

```
f= open("test3.txt", "w")
```

```
f.write("Oops!")
f.close()
f= open("demofile3.txt", "r")
print(f.read())
```

Create a New File

To create a new file in Python, use the `open()` method with one of the following parameters:

- “x” - Create; it will create a file, returns an error if the file exist.
- “a” – Append; it will create a file if the specified file does not exist.
- “w” - Write; it will create a file if the specified file does not exist.

Example:

Create a file called `test1.txt`:

```
f = open("test1.txt", "x")
```

Result: a new empty file is created!

9. Explain any two string functions.

Ans. There are many strings functions in Python, but we will discuss only two:

`capitalize()`

`count()`

Suppose we have the `st='hello dear'` string.

capitalize string function:

This capitalizes all letters.

Example:

```
st.capitalize()
```

Output:

`HELLO DEAR`

count string function:

It returns total count of elements(words) in the string

Example:

```
st.count(st)
```

Output:

10. What is the use of s.split()? Explain the parameters of this function.

Ans. The **split()** method splits a string into a list. You can specify the separator' the default separator is any whitespace.

Syntax

```
string.split(separator, maxsplit)
```

separator: Optional. It is used to separate the strings using whitespace.

maxsplit: Optional. Specifies maximum number of slips to be performed on the given string. If maxsplit is given as an input then list will be splitted into maxsplit + 1 items

example:

```
txt = "hello, my name is Peter, I am 26 years old"  
x = txt.split(", ")
```

output:

```
[‘hello’, ‘my name is Peter’, ‘I am 26 years old’]
```

Example:

Split the string into a list with max two items:

```
txt = "apple#banana#cherry#orange"  
# setting the maxsplit parameter to 1, will return a list  
with 2 elements!  
x = txt.split("#", 1)  
print(x)  
  
output:  
[‘apple’, ‘#banana#cherry#orange’]
```

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 3

An Overview of ML Algorithms

Introduction

The previous two chapters dealt with the concepts behind *machine learning* and Python. The aim of this chapter is to provide an insight into the flow of information and the operations performed in machine learning. This chapter introduces the various machine learning algorithms and aims to familiarize you to the machine learning programs that can learn from the data provided and improve from experience, without any human intervention. These learning tasks may include learning the function that maps the input to the outputs or learning through the hidden structure in unlabeled data using algorithms coined under supervised and unsupervised learning. To understand the core concepts in detail, this chapter explores the working process of any given machine learning algorithms with the help of several examples. As Python is a handy tool to get started with supervised and unsupervised learning, this chapter explores its various functionalities. It further elaborates on the performance measures to be considered for evaluating the machine learning system that has been developed.

Structure

In this chapter, we will study the following topics:

- Machine Learning modeling flow
 - Terms used in pre-processing
 - Need of Data Preprocessing
 - Training and testing the model on Data
 - Evaluation
 - Hyperparameter tuning
 - Prediction

- Regression Algorithms
 - Types of regression techniques
 - Stepwise Regression
 - Ridge Regression
 - Lasso Regression
 - ElasticNet Regression
- Classification
 - Terminology used in Classification Regression
 - Types of Classification Regression
 - Performance measures for classification
- Clustering
 - Clustering algorithms
- Neural Networks and SVM
 - Building Blocks: Neurons
 - Combining Neurons into a Neural Network
 - Training the neural network
 - Neural Network Architectures
- Data Science libraries in Python
 - Numpy
 - Pandas
 - Matplotlib
- Evaluation of ML systems
 - Test and train datasets
 - Performance measure
 - Model Evaluation Techniques
 - Model Evaluation Metrics

Objectives

After completing this chapter, you will be able to understand the need for preprocessing in ML, the various algorithms and Data Science libraries written in Python which are used in machine learning. You should also be able to know the concepts behind the code editors and Python IDE. The algorithms used for a particular application will be evaluated using several evaluation parameters. On completion of this chapter, you should be able to identify these measures to evaluate the ML system.

Machine learning modeling flow

Machine learning model is a piece of code that has been developed to make a machine smart by training it with data.

As illustrated in *Figure 3.1*, the data that is input to the system comprises of training and testing data, indicated by 1a and 1b, respectively. The training dataset is the one on which the machine learning model is trained, indicated by 2a.

A machine learning model comprises of algorithms utilized based on the problem specified. This algorithm is evaluated based on several performance measures, which will be explained later in the chapter; these are represented by 3a, 3b and 3c, respectively.

When new data comes in, called testing data, the models are evaluated based on the algorithm selected and based on the measures identified, represented by 2b, 3b and 3c.

The obtained model output indicated by 4 is then used for the task of predicting, expressed by 5:

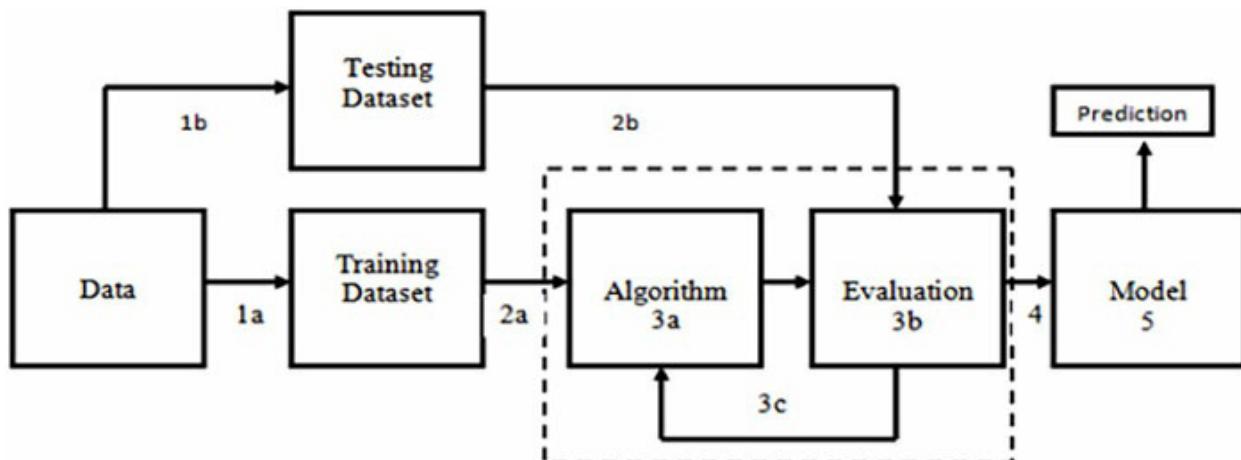


Figure 3.1: Representation of Machine Learning Flow Process

Thus, before training the machine with data, machine learning workflow can be defined in three stages. Let's look at these stages.

Gathering data

This step is considered important because the quality and quantity of data that is gathered would directly determine how good the predictive model could be. Data collected for any ML project usually consists of data gathered from various sources, such as a file or database, and sensor data or real-time data. However, this data cannot be directly used for analysis as most real-world data is messy, which means the data might have one or more of the following issues:

- **Missing data:** This is data that is not continuously created or not available due to technical issues in the application.
- **Noisy data:** This data is referred to as the errors caused by humans during manually gathering the data or error caused by technical problems of the device at the time of collecting the data.
- **Inconsistent data:** This type of data refers to the errors caused due to duplication of data or mistakes in the name or values.

The presence of any of these types of data will degrade the quality of the results. Therefore, data preprocessing or preparation is done to solve this problem.

Data preprocessing

The next step of machine learning is data preparation, wherein the data is loaded into a suitable place and prepared for training. At the same time, pertinent visualizations of the data are done to determine any relevant relationships between different variables or any imbalances between data variables. In machine learning, there is an 80/20 rule: every data scientist is expected to spend at least 80% of the time on data preprocessing and the remaining 20% of the time on performing the analysis.

Data preprocessing is identified as a process of cleaning the raw data, that is, the data collected from the real world, and converting it into a clean data set before it is fed to the algorithm.

Terms used in preprocessing

Before preprocessing data, it is necessary to know the terms utilized: raw data and prepared data.

Raw data (or just data)

The data in its source form, which is the data in a data warehouse or the data sent from streaming systems that are not prepared for the ML task, is termed as raw data.

Prepared data

Data in sources that have been parsed, joined, aggregated and summarized to the right granularity and put into a tabular form for ML operations is referred to as prepared data.

Need for Data Preprocessing

In order to achieve better results from a machine learning model, the underlying data has to be in a proper manner, formatted and preferably in one data set. The data has to move from being raw to structured and then preprocessed, as shown in *Figure 3.2*:

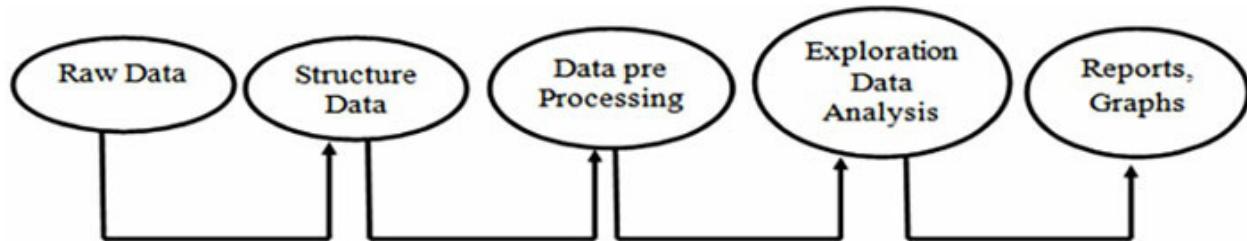


Figure 3.2: Movement of Data

Basic preprocessing techniques used to convert raw data into clean data involve the following steps:

1. **Conversion of data:** Data in any form must be converted into numeric form for machine learning models to handle it.
2. **Handling the missing values:** Whenever any missing data in a data set is observed, one of the following can be considered:

- a. **Ignoring the missing values:** The corresponding row or column of data can be removed as needed.
 - b. **Filling the missing values:** In this approach, the missing data can be filled manually. The value added in the missing data field can be the mean, median or the highest frequency value.
 - c. **Machine learning:** Another approach is to predict the data that can be added in the empty position based on the existing data.
3. **Outlier detection:** Some data might deviate drastically from other observations in the data set; such type of data is said to be error data.

Thus, data preprocessing forms one of the most important steps in machine learning to build accurate machine learning models.

Preprocessing in ML

Preprocessing in an ML algorithm is divided into six steps, as listed here:

1. Importing the libraries
2. Importing the dataset
3. Dealing with the missing data in the dataset
4. Encoding categorical data
5. Splitting the dataset into training set and test set
6. Feature scaling

This entire process can be automated using machine learning algorithms, mathematical modelling and statistical knowledge.

Researching the best model for the data

The next step in the workflow is to choose a model; many models have been created over the years. The main goal of using an ML model is to train the best performing model possible, using the preprocessed data. The various ML models and their categories include the ones represented in *Figure3.3*:

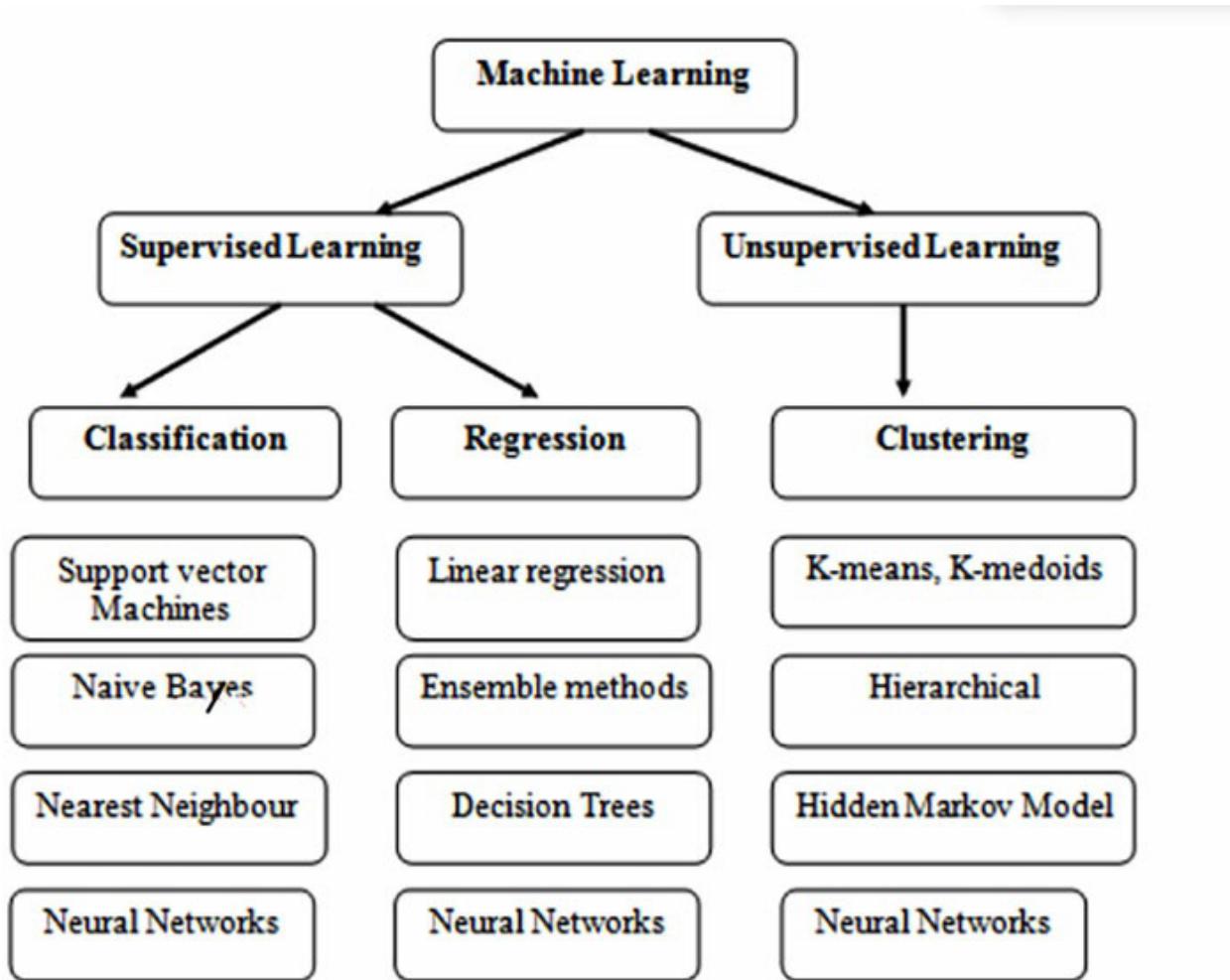


Figure 3.3: Overview of ML models

ML models are bifurcated into the following:

Supervised learning: It comprises determining the mapping function $Y = f(X)$, where Y and X represent the output and input variables, respectively. Supervised learning is split into the following:

- **Classification:** In this method, the output is classified into different classes. This output target variable is said to be categorical. Most important classification algorithms include K-Nearest Neighbor, Naïve Bayes, Decision Trees/Random Forest, Support Vector Machine and Logistic Regression, as represented in [Figure 3.3](#).
- **Regression:** In this approach, the output is observed to be numeric, and the target variable is said to be continuous. Regression algorithms include Linear Regression, Support Vector Regression, Decision Trees/Random Forest, Gaussian Progresses Regression and Ensemble

Methods.

Unsupervised learning: In this method, unlabeled, uncategorized data is input to the system. The respective machine learning algorithm acts on this data without prior training, so the output of the system depends on the coded algorithms.

Unsupervised learning is further divided into two categories:

- **Clustering:** In this, members with similar features are combined. However, the group, labels are not known prior. Methods used for clustering include Gaussian mixtures, K-Means Clustering, Boosting, Hierarchical Clustering, K-Means Clustering, and Spectral Clustering.
- **Association:** In this, the algorithm tries to learn without the availability of labeled data. It aids in discovering interesting relations between variables in large databases. The methods used are Associative rule mining and Mining Multilevel Association Rules from Relational Databases and Data Warehouses.

Training and testing the model on data

To get an accurate output, the input to the system and the algorithm used need to be just right. To achieve an optimized training model, the data is split into three parts:

- **Training data:** The classifier is trained using this data.

The equation for straight line is $y=mx+b$, where x is the input, m is the slope of that line, and b is the y -intercept. As the slope m can take up a number of values, the slope can be represented by “weights” matrix and can have a bias associated with it, as shown in [Figure 3.4](#):

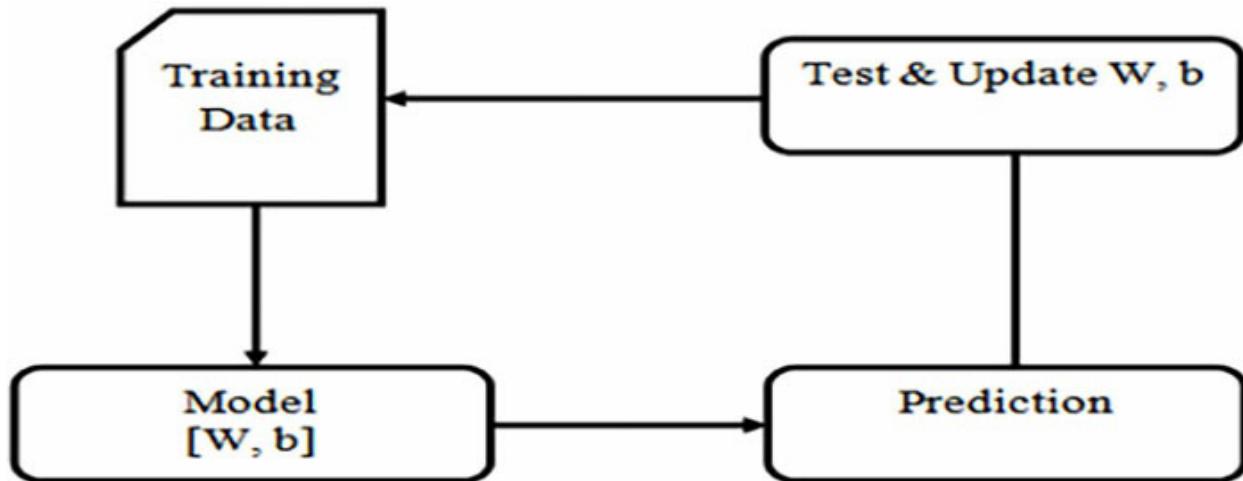


Figure 3.4: Training Process

This process is then repeated. Each iteration or cycle of updating the weights and biases is called one training “step”. When the training is initiated, considering the preceding example, a random line is drawn through the data. As each step of the training progresses, the line moves, step by step, closer to the ideal line. The entire process comprises of working with 2 sets of data namely validation and testing data elaborated as follows:

- **Validation data:** A set of unseen data is used from the training data to tune the parameters of the classifier. Cross-validation is used to estimate the skill of a machine learning model on the unseen data. While training the classifier, only the training or validation set is available.
- **Testing data:** The performance of the classifier is tested on the unseen test data and is available only while testing the classifier. It is used for evaluating the trained model’s performance. Once the data set is determined, a confusion matrix is developed, which indicates how well the classification model is trained. Confusion matrix is represented by four parameters:
 - **True positives:** These are cases in which the predicted TRUE and the actual predicted output is correct. More values are preferred here.
 - **True negatives:** In this case, we predicted FALSE and our predicted output is correct. More values are preferred here.
 - **False positives:** In this, we predicted TRUE, but the actual predicted output is FALSE.

- **False negatives:** In this, the prediction is FALSE, while the actual predicted output is found to be TRUE.

The accuracy of the model can also be determined using the confusion matrix, as follows:

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{\text{total number of samples}}$$

In the case of an imbalanced dataset, accuracy is not the correct evaluation parameter for classification model. For this type of situation, the precision or recall is used as a better evaluation parameter.

Evaluation

On completion of the training, evaluation is done to determine the goodness of the model. Thus, model evaluation helps find the following:

- The best model representing the data that has not yet been seen
- How well the chosen model will work in the future by tuning the hyperparameters of the model
- How to improve the accuracy
- How to increase the number of true positives and true negatives by observing the confusion matrix

Thus, model evaluation forms an integral part of the model development process.

Hyperparameter tuning

Once evaluation is performed, training needs to be improved to enhance the accuracy or reduce the number of iterations required to reach the ideal condition. One such approach is tuning the parameters that would have been implicitly assumed while training. This would require testing those assumptions.

Another parameter under consideration is the “*learning rate*”. It indicates how far the line shifts during each step, based on the information from the previous training step and the initial conditions considered. This value provides information about the accuracy of the model and the amount of time

the training would take.

Prediction

Prediction, or inference, is the step that determines the correctness of the output.

Metrics Used

Though accuracy is one of the metrics for evaluating a machine learning model, fairness and inclusion provides a measure to understand the bias introduced by the data and ensure that the model provides equitable predictions across all demographic groups. Thus, fairness can be said to comprise of three steps:

- Identify dataset imbalances
- Ensure fair treatment for all groups
- Set prediction thresholds

Regression algorithms

These algorithms predict the output values based on input features fed into the system. Regression techniques are mainly categorized based on three metrics: the number of independent variables, the type of dependent variables and the shape of the regression line.

Types of regression techniques

Regression algorithms include linear regression, regression trees, lasso regression and multivariate regression.

Linear Regression

It is a widely known supervised learning algorithm used to predict a continuous real numbered output, that is to predict dependent variable value (y), called the target, based on a given independent variable (x), the predictor. Thus, it determines a linear relationship between x (input) and y(output). Hence, the name is linear regression.

Linear regression is said to be based on a hypothesis that is linear, quadratic, polynomial, or nonlinear. The hypothesis is a function based on some hidden parameters and the input values.

Simple linear regression has only one independent variable, while multiple linear regression has more than one independent variable, as shown in [Table 3.1](#):

Simple linear regression	Multiple linear regression
There is a linear relationship between independent and dependent variables.	It uses several variables to predict the outcome.
It is very sensitive to outliers.	It is sensitive to minor changes in the model and suffers from multicollinearity and auto correlation.
It has only one independent variable.	Forward selection, backward elimination and stepwise approach can be used for selecting most significant independent variables.

Table 3.1: Simple linear regression and multiple linear regression

Logistic Regression

Logistic regression is determined when the dependent variable y is binary (0/1, True/ False, Yes/ No) in nature. In other words, the label is either categorical or discrete in nature. This leads to the value of the output Y to be between 0 and 1, and represented by the following equation using log odds ratio:

$$\text{Odds} = \frac{p}{1-p}$$

= (probability of event occurrence / (probability of not event occurrence))

- Logistic regression can handle various types of relationships as it applies a non-linear log transformation to the predicted odds ratio.
- There should not be any correlation between independent variables, that is, no multicollinearity.
- Ordinal logistic regression has ordinal values of dependent variables.
- Multinomial logistic regression is considered when the dependent

variables belong to multi class approach.

Polynomial Regression

A regression equation is a polynomial regression equation if the power of the independent variable is more than 1, as represented in the polynomial equation. In this regression technique, the best fit line is not a straight line but a curve that fits into the given data points.

Stepwise Regression

This form of regression is used when the data comprises of multiple independent variables. Stepwise regression fits the regression model by adding or dropping covariates one at a time, based on a specified criterion.

Ridge Regression

This technique is employed when the independent variables are highly correlated. Standard errors are reduced by adding a degree of bias to the regression estimates. As mentioned earlier, the equation for linear regression was, where e indicates the error term. This prediction error is said to be affected by bias and variance.

Lasso Regression

Least Absolute Shrinkage and Selection Operator, Lasso, reduces the variability and improves the accuracy of linear regression models.

ElasticNet Regression

ElasticNet is a hybrid of Lasso and Ridge Regression techniques and is useful when there are multiple features that are correlated.

Classification

A classification model attempts to draw conclusions from the observed values. Once data input is given, the classification model would predict one or more output value.

Therefore, the objective of the classification model is to find the decision

boundary that separates the target variable's categories. Training dataset is used to get better boundary conditions; this would aid in determining each target class. Once these boundary conditions are determined, the focus is on predicting the target class. This process is known as classification.

Terminology used in Classification Algorithms

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. The terms that get associated with classification include the following:

- **Classifier:** It is defined as an algorithm that relates the input data to a specific category.
- **Classification model:** A model that tries to draw conclusions from the input values given for training.
- **Feature:** It is an individual measurable property of a phenomenon being observed.
- **Binary Classification:** It classifies a task having only two possible outcomes, for example, gender classification (Male / Female).
- **Multi-class classification:** It classifies a task having more than two classes, with each sample assigned to only one target label.
- **Multi-label classification:** In this, each sample is mapped to a set of target labels (more than one class).

Types of classification algorithms

Classification algorithms are broadly classified into the following:

- **Linear classifiers**

A linear classifier classifies data into labels based on a linear combination of input features. Therefore, it can only be used to classify data that is linearly separable. Linear classifiers are further classified into the following:

- **Logistic regression**

It is most useful for understanding the influence of independent

variables on a single outcome variable. However, it works only when the predicted variable is binary, and it assumes that all predictors are independent of each other and the data does not have missing values.

- o **Naive Bayes classifier**

It is based on the Bayes theorem, assuming that there is independence between every pair of features. It requires lesser amounts of training data and is considered to be very fast compared to other methods. However, it is said to be a bad estimator. In simple terms, a **Naive Bayes classifier** assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Bayes theorem helps to compute posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$, as follows:

$$P(c | x) = P(x_1 | c) * P(x_2 | c) * P(x_n | c) * P(c)$$

Here,

- $P(c|x)$ is the posterior probability of class (target), given predictor (attribute).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood, that is, the probability of a predictor given class.
- $P(x)$ is the prior probability of the predictor.

- o **Support vector machines**

Support vector machines represent the training data as points in space separated by a gap as wide as possible. SVM is considered to be effective in high-dimensional spaces and is also said to be memory efficient. However, the algorithm does not directly provide probability estimates.

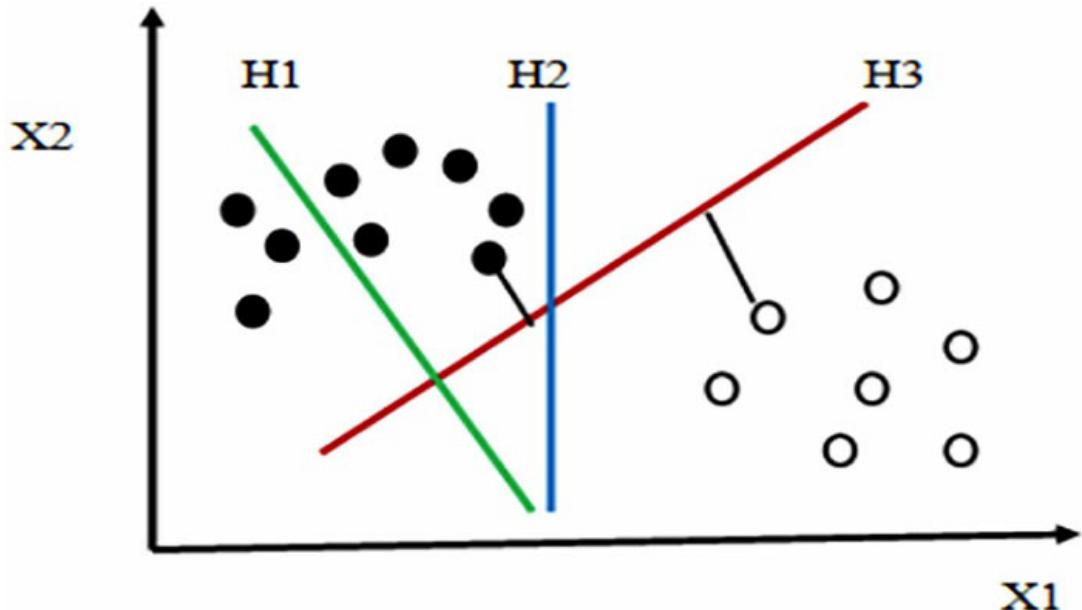


Figure 3.5: Support vector machine

From [Figure 3.5](#), it is evident that H1 is not a good hyperplane as it doesn't separate the classes, H2 forms a better hyperplane, but only with small margin, and H3 separates them with maximum margin (distance).

- **k-nearest neighbor**

It is a type of lazy learning as it simply stores instances of the training data. Classification is done through majority voting of the k-nearest neighbors of each point. It is easy to implement robust to noisy training data and is effective if training data is large. However, it needs to determine the value of K, and the computation cost is high.

- **Decision trees**

These algorithms produce a sequence of rules that enable data classification and make the resultant groups as distinct as possible. These are simple to understand and visualize. They can also handle numerical and categorical data. However, complex decision trees can also be created. However, they would not be able to generalize well, and small variations in its data might result in a completely different tree.

- **Random forest**

It is defined as a meta-estimator that fits several decision trees.

However, the samples are drawn with replacement and are difficult to implement.

- o **Neural networks**

It consists of several units called neurons that are arranged in layers. They convert an input vector into an output. Each unit takes an input, to which a function is applied, and the unit passes the output to the next layer.

Performance measures for classification

The most commonly used performance metrics for classification problem include the following:

$$\text{Accuracy} = (\text{TruePositive} + \text{TrueNegative}) / \text{Total observations}$$

Accuracy is defined as a ratio of correctly predicted observation to the total number of observations:

$$\text{F1-Score: } \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is defined as the weighted average of precision and recall.

Precision: How often the prediction is correct when a positive value is predicted

Recall: The number of times the prediction is correct

Clustering

Clustering is an unsupervised learning method whose objective is to find different groups within the elements in the data. This involves grouping data points based on the specified condition or criteria. Thus, datapoints in the same group have similar properties or features, while those in different groups have different characteristics.

Examples of clustering include market segmentation, image compression and document clustering.

Clustering algorithms

Various clustering algorithms are incorporated to find the grouping between

similar data points. It is further used to differentiate between clusters. These include the following:

K-Means Clustering

It is the most widely known algorithm that is easy to understand and implement. Further, k – means functions fast, with only a few computations. However, in k-Means algorithm, number of groups to be used must be determined beforehand, and it also starts with a random choice of cluster centers; therefore, it yields different clustering results on different runs of the algorithm and lacks consistency.

Mean-Shift Clustering

It determines the density of data points through a sliding-window-centroid – based algorithm.

- **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

It is a density-based clustering algorithm that does not require a pre-set number of clusters. It identifies outliers as noise and can find arbitrarily sized and shaped clusters. However, handling clusters with varying density is a task with DBSCAN because of distance epsilon ε .

- **Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)**

In this algorithm, it is assumed that the data points are Gaussian distributed and can take any elliptical shape due to the standard deviation parameter, which is then assigned to a single cluster. They are said to be flexible and less restrictive. To add to this further, GMM supports mixed membership, that is, GMM belongs to X-percent in class 1 and Y-percent in class 2. In order to find the mean and standard deviation parameters of the Gaussian for each cluster, an optimization algorithm named **Expectation–Maximization (EM)** is undertaken as follows:

1. The number of clusters is first selected by randomly initializing the Gaussian distribution parameters for each cluster.
2. The probability of each data point belonging to a particular cluster is

then determined. When a point is closer to the Gaussian's center, it is considered to belong to that cluster.

3. New sets of parameters are then computed using a weighted sum of the data point positions. The weights specify the probability that the data point belongs to the specific cluster.
4. Steps 2 and 3 are repeated iteratively till convergence.

Agglomerative Hierarchical Clustering

Hierarchical clustering is considered to be an alternative to prototype-based clustering algorithm, where there is no need to specify the number of clusters, it will find it by itself. Hierarchical clustering algorithms are categorized as follows:

- **Top-down algorithm**

This method starts by bringing all datapoints in one single cluster. Then, it splits the cluster iteratively into smaller ones till each one of them contains only one sample.

- **Bottom-up algorithm**

It is also known as hierarchical agglomerative clustering or HAC. In this, each data point is considered as a single cluster, which is then successively merged or paired merged and has all data points. Dendograms, visualization of a binary hierarchical clustering, also known as the hierarchy of clusters, is represented as a tree. The root of this tree represents all the samples gathered, whereas the leaves represent the clusters with only one sample, as represented in [Figure 3.6](#):

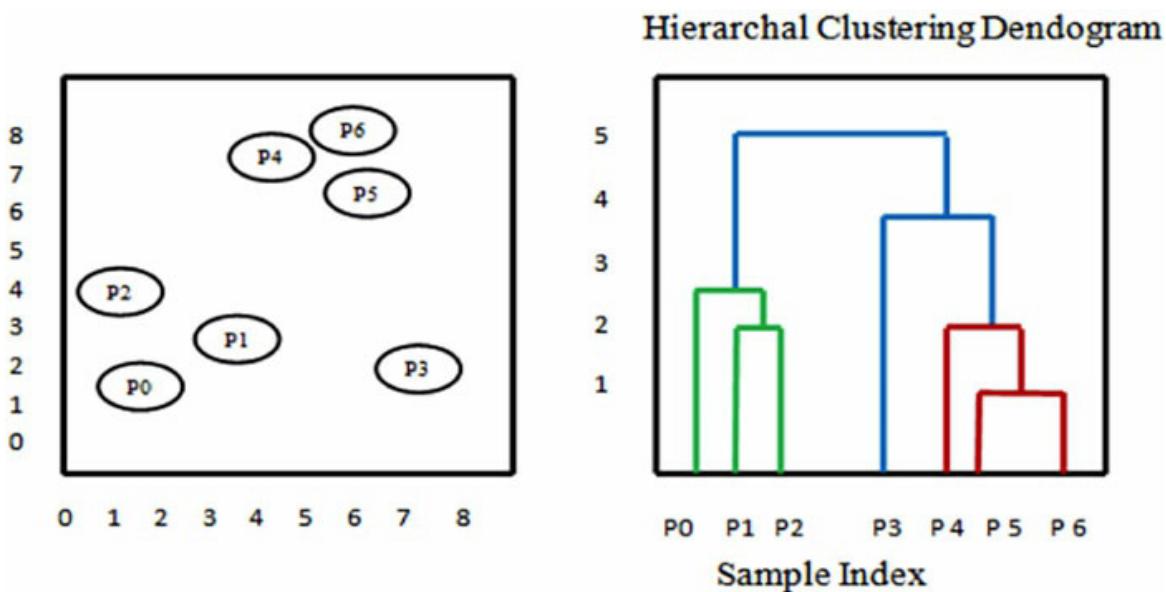


Figure 3.6: Agglomerative Hierarchical Clustering

The algorithm comprises of the following steps:

1. In the first step, every data point is supposed to be a single cluster.
2. A distance metric is specified to measure the distance between two clusters.
3. Two clusters are then combined into one on every iteration based on the smallest distance between them.
4. The previous step is repeated till the root of the tree is reached, that is, only one cluster with all data points is obtained.

Clustering Validation

It is the process of evaluating the result of the cluster objectively and quantitatively by applying cluster validation indices. These indices are divided into three main categories:

External Indices: This scoring method is used if the provided original data is labelled. On this, a clustering structure is matched. The most used index in this case is the Adjusted Rand index.

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}$$

Where **a** is the number of points that are in the same clusters, in C and in K

b is the number of points that are in both clusters

n is the total number of samples

$$ARI = RI - Expected\ Index / Max(RI) - Expected\ Index$$

The value of ARI ranges between -1 to 1. The higher the value of ARI, the better it matches the original data.

Internal Validation Indices: This method is used when the data is unlabeled. The index used for internal validation is the Silhouette Coefficient.

Silhouette Coefficient: For each data point, it is defined as follows:

$$Si = bi - ai / Max(bi - ai)$$

Where **a** is the average distance to the other sample for a in the same cluster.

B is the average distance to other sample i in the closest neighboring cluster.

The **Silhouette Coefficient (SC)** is given by the following:

Sc is the average of the sum of Si for each point

Its value ranges between -1 to 1. The higher the value obtained, the better is the K value selected. It is preferred only for K-Means and hierarchical clustering algorithms and not for DBSCAN.

Neural Network and SVM

A neural network is a class of machine learning algorithm that is used to model complex patterns in the datasets through multiple hidden layers and non-linear activation functions.

Building Blocks: Neurons

The basic unit of a neural network is the neuron. It takes the inputs, processes them with mathematical operations and produces an output as represented by

a 2-input neuron in [Figure 3.7](#):

Inputs

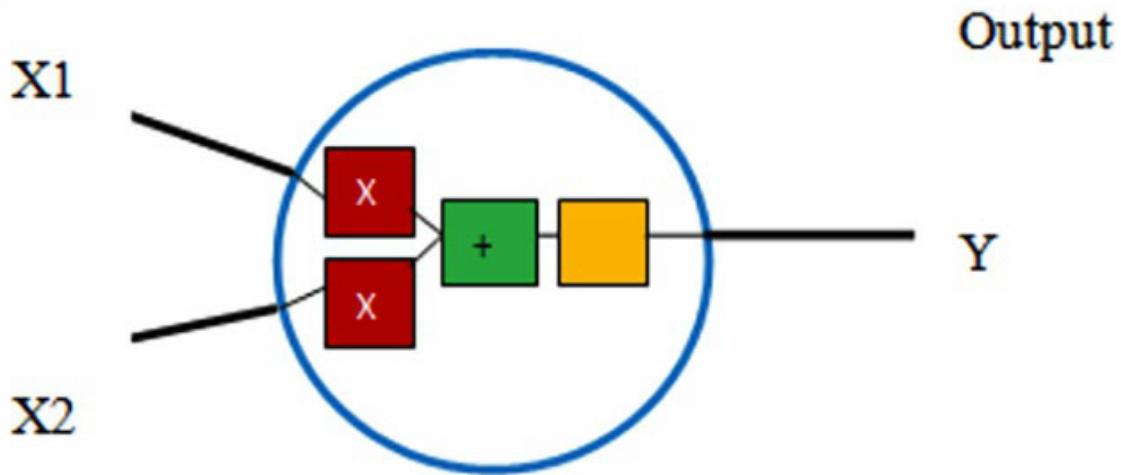


Figure 3.7: Input neuron

The operation that is performed is as follows:

1. Each input is multiplied by a weight, where x_1 and x_2 are the inputs, and w_1 and w_2 are the weights.

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

2. All the weighted inputs are then added with a bias b :

$$(x_1 * w_1) + (x_2 * w_2) + b$$

3. The sum is passed through an activation function that converts an unbounded input into predictable form given in the following equation. This process of passing inputs forward to get an output is known as feedforward. Activation functions are also known as mapping functions. They take some input on the x-axis and output a value in a restricted range.

$$y = f(x_1 * w_1 + x_2 * w_2 + b). \quad y = f(x_1 * w_1 + x_2 * w_2 + b)$$

Combining Neurons into a Neural Network

A neural network comprises of a bunch of neurons connected to each other, as shown in [Figure 3.8](#):

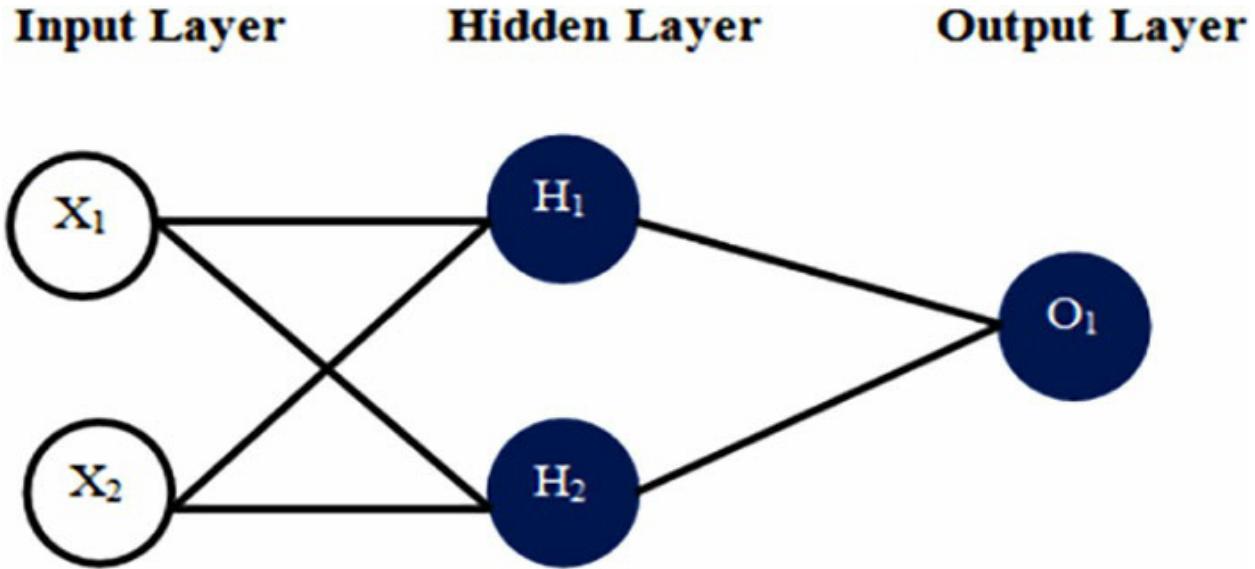


Figure 3.8: Neural Network

There are two layers that every NN has. These layers are the hidden layer that fall between the input and output layers. As represented in *Figure 3.8*, the network has two inputs (x_1 and x_2) and a hidden layer with two neurons (h_1 and h_2). The hidden layer is said to be any layer that is in between the input (first) layer and output (last) layer. The output layer with 1 neuron (O_1) is further represented that has inputs from h_1 and h_2 . The hidden layers H_1 and H_2 then help to develop the output neuron O_1 .

Training the neural network

It refers to training the network in such a way that it minimizes the losses.

The losses are defined in terms of **Mean squared error (MSE)**

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Where n is the number of samples, y represents the variable being predicted, and Y_i is the true value of the variable (the “correct answer”). \hat{Y}_i is the predicted value of the variable the network outputs. $(y_i - y^{\wedge})^2$ (is known as the squared error).

The goal is to minimize the error. A guess, an error measurement, an incremental change in its weights and adjustment to the coefficients are

performed at every step of learning from the neural network.

A model represents the collection of weights. These weights map the inputs that enter the network to a set of outputs, forming the guess made by the network. This is often known as scoring input. Therefore, the guess is being represented by **Guess = input * weight**. The actual data, also known ground truth, is then compared to the guess to determine the error.

Neural Network Architectures

There are different architecture styles of neural networks that can be used. These include the following:

- **Perceptrons or feed-forward neural network**

Perceptron is the first generation of neural networks, proposed by Rosenblatt; it is shown in [Figure 3.9](#). It feeds information from the front to the back. Training perceptrons is done through backpropagation, where the error is backpropagated. This error is the difference between the input and the output data. Training gives the network paired datasets of inputs and outputs. Inputs are sent into the neuron, where they are processed, resulting in an output. With a sufficient number of hidden neurons, the algorithm models the relationship between the input and output. Multilayer perceptrons comprise three or more layers with a nonlinear activation function.

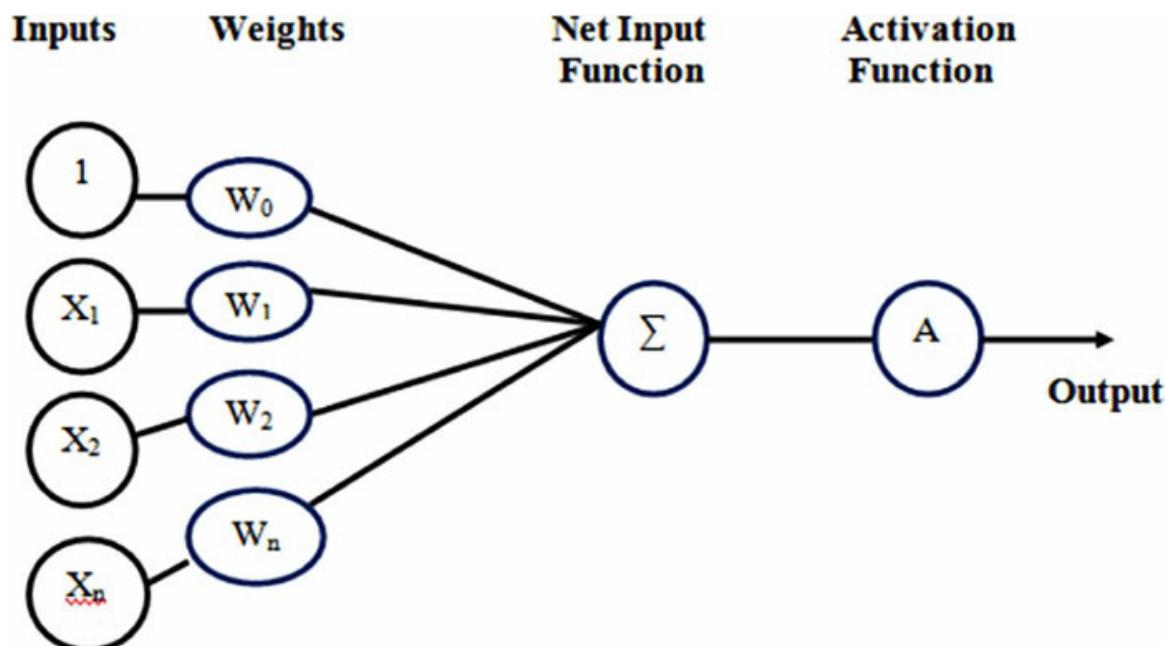


Figure 3.9: Rosenblatt's Perceptron

- **Convolutional neural network**

It is a variation of the multilayer perceptron and uses backpropagation in a feedforward network comprising of several hidden layers, replicated units in each layer and output pooling of nearby replicated units.

- **Recursive neural network**

It uses weights to make structured predictions.

- **Recurrent neural network (RNN)**

It connects neurons in a directed cycle manner without utilizing the activation function. They are said to be stateless perceptrons. RNN combines two properties:

- It comprises of a distributed hidden state that allows the neurons them to store information about the past efficiently.
- It possesses nonlinear dynamics that allow the neurons them to update their hidden state in complicated ways.

- **Long / Short-Term Memory**

It uses the recurrent neural network architecture and combats the vanishing/exploding gradient problem by introducing gates and explicitly defining memory cell. The working of this memory cell comprises of storing the previous value of data till a “forget gate” tells the cell to forget those values.

- **Gated recurrent units (GRU)**

They do not need the cell layer to pass the values. The calculations within each iteration ensure that the probability of retaining the old or new information is high. GRUs are incrementally faster and easier to run, but they are less expressive.

- **Hopfield network**

Recurrent networks of non-linear units are hard to analyze as they settle to a stable state, oscillate or follow chaotic trajectories in different manners. To resolve these issues, John Hopfield introduced the **Hopfield Network (HN)**. In the HN network, every neuron is connected to every other neuron, where the networks are trained by

setting the value of the neurons to the desired pattern, and then the weights are computed. The weights do not change after this. Once trained for a pattern, the network converges to only that learned pattern.

- **Boltzmann machine**

It is a type of stochastic recurrent neural network, which is capable of learning internal representations and can represent and solve difficult combinatorial problems. In this, only some neurons are marked as input neurons, and others act as hidden. Once the output is obtained, the input neurons that have binary activation patterns become output neurons. The process begins by using random weights and learning through backpropagation.

- **Deep belief networks**

Using backpropagation has some problems, such as the following:

- a. Back propagation method performs on labeled training data while most of the data is unlabeled.
- b. It operates in a very slow manner in networks having multiple hidden layers.

To overcome these limitations, unsupervised learning approaches are considered by adjusting the weights.

- **Autoencoders**

These are neural networks that work on unlabeled data and can be used to encode a given input into a representation of a smaller dimension. A decoder then converts the input back from its encoded version.

- **Generative Adversarial Network (GANs)**

It consists of a combination of feedforwards of any two networks. The task of one network is to generate the content (generative), while that of the other is to determine or judge the content, termed as discriminative.

Support vector machine (SVM)

It is a supervised machine learning algorithm where each data item is plotted as a point in n-dimensional space, n being the number of features considered. The value of each feature is said to be the value of a particular coordinate. SVM classifier segregates the classes distinctively. For this, a hyperplane,

also known as the decision boundary, is determined to differentiate between two classes for given classification problem. The key operation of SVM is the identification of the correct hyperplane that separates the data into distinct classes.

Consider the three hyperplanes A, B, and C. In order to identify the right hyperplane and to classify square and circle, the thumb rule is to identify the hyperplane that separates the two classes better. Refer to the following figure:

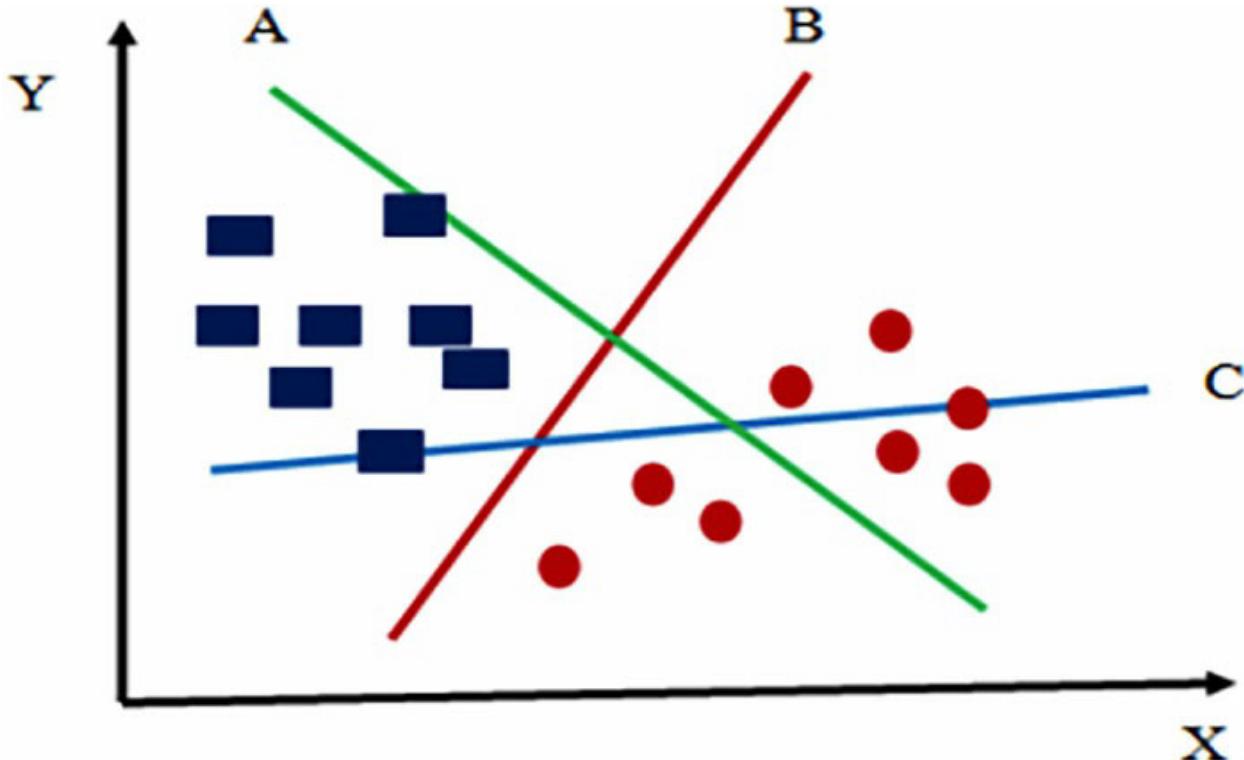


Figure 3.10: Hyperplane B as separator

As observed in [Figure 3.10](#), hyperplane “B” clearly separates the square and the circle.

However, if all the three hyperplanes segregate the classes well, it is essential to identify the best hyperplane. The best hyperplane is identified to be the one that has the maximum distances between their nearest data point and the hyperplane. This distance is known as the margin. Refer to the following figure:

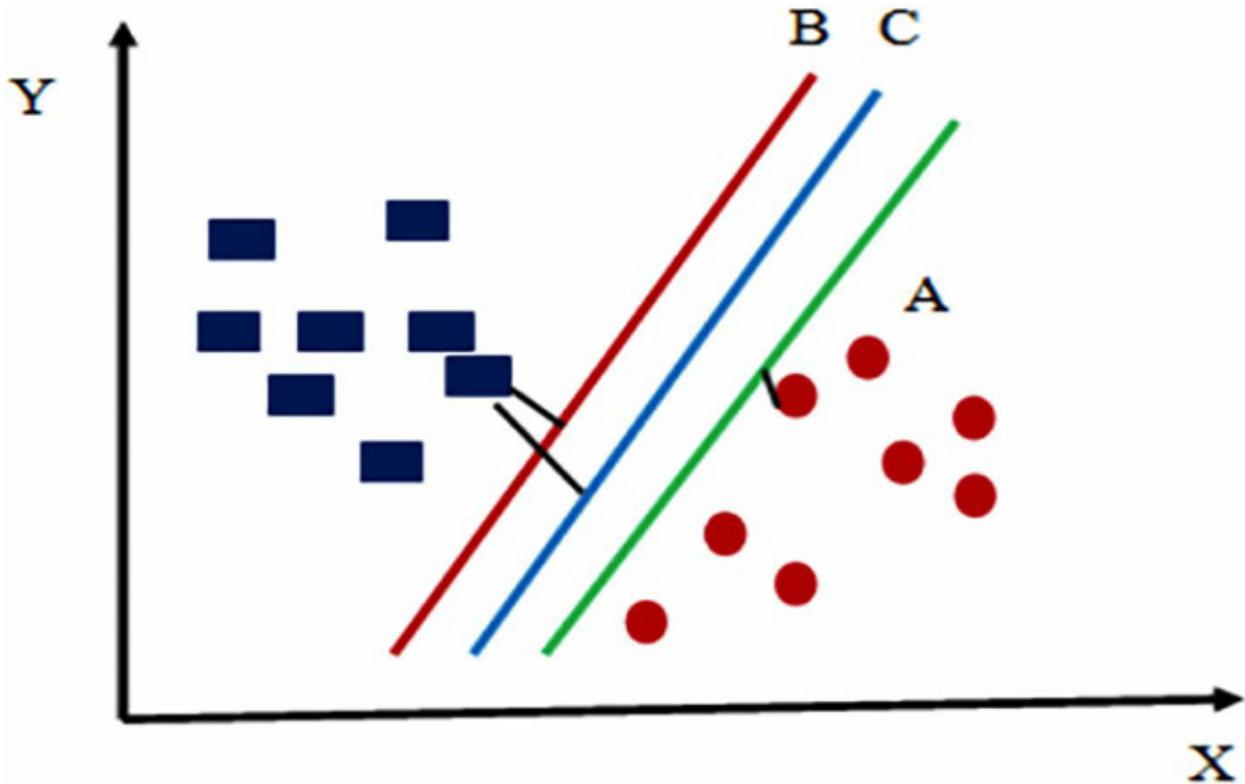


Figure 3.11: Representation of margin

[Figure 3.11](#) shows that the margin for hyperplane C is more as compared to A and B.

SVM and neural networks have several differences between them, as represented in [Table 3.2](#):

Support Vector Machines (SVM)	Neural Networks (NN)
SVMs works by creating one or more hyper planes that separate the data clusters.	Single neural network neuron is an SVM without kernel having different cost functions.
SVMs are conceptually simpler.	Neural networks are considered complex due to the hidden layers.
SVMs are theoretically founded.	NNs are heuristic in nature.
SVMs attempt to maximize the margin between the hyper planes and the clusters.	They try to map m-dimensional inputs to another dimensional space without directly determining a separator margin between classes.

SVMs only rely on the support vectors.	Neural networks are preferred when data points are structured.
SVM implementation finds the optimal set of parameters.	Neural networks get stuck in a local minima.
SVM employs kernel tricks and maximal margin.	They employ back propagation and activation functions.
An SVM is guaranteed to converge towards the best solution.	A neural network would draw any correct line that would separate the samples, but it does not have the best generalization properties.
SVMs perform better on small network or application domains.	Neural networks perform better on very large datasets.
Major problems with SVM are the selection of the model, choosing the kernel and optimizing the kernel.	The statistical distribution of the input keeps changing as training proceeds; therefore, it reduces the training efficiency.

Table 3.2: Difference between SVM and Neural Networks

Machine Learning Libraries in Python

Writing a machine learning application or developing a complete application in data science is easy because of rich support of libraries in Python. Though there are numerous libraries to perform various tasks in machine learning applications, there are a few important and basic libraries that are of great help to start building any machine learning application.

Let's take a tour of a few important functions in these libraries. The three important libraries we are going to discuss are NumPy, Pandas, and Matplotlib.

Numpy

As the name implies, this is the library that supports numerical calculations and tasks, and array operations and basic mathematical functions on the array and other data types in Python. The basic data type of this library is an Array object. **Numpy** contains the following:

- A powerful N-dimensional array object

- Sophisticated (broadcasting/universal) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transforms and random number capabilities
- NumPy can also be used as an efficient multi-dimensional container of generic data

The following example shows how to create a Numpy array.

Example 1:

```
>>>import numpy as np
>>>list1=[1,2,3,4,5]
>>>a=np.array(list1)
>>>print(a)
1 2 3 4 5
>>>print(a.shape)
(5,)
>>>print(a.dtype)
Int32
```

Here the array is created from a list type variable. One can create an array from homogenous data types. Numpy arrays have a fixed size, and modifying the size means we are creating a new array. The size and number of dimensions of the array can be known with the help of the **shape** function. You can use the **dtype** function to know the type of data the array is holding.

We can create an array with different data types, such as integer and float, and with different sizes, such as **int16**, **int32**, **float32**, and **float64**. One may think why prefer array over list? The three main advantages of using array are that less memory is required, it offers fast processing, and it is convenient.

Let's look at a few useful predefined arrays provided by **Numpy** library in the following table:

Syntax	Use	Example
np.zeros(shape, dtype)	Create array of zeros	np.zeros((5,3))
np.ones(shape, dtype)	Create array of ones	np.ones(5)
np.full(shape, value, dtype)	Create array of value provided with required data type	np.full(5, 4, dtype=int32)

np.eye(shape, dtype)	Identity array	np.eye(3)
----------------------	----------------	-----------

Table 3.3: Common Numpy array

Example 2:

```
>>>import numpy as np
>>>list1=[1,2,3,4,5]
>>>print(np.zeros((2,2)))
[[0. 0.]
 [0. 0.]]
>>>print(eye(3,dtype=int))
[[1 0 0]
 [0 1 0]
 [0 0 1]
 [0 0 0 1]]
```

The previous example shows how these common array functions can be used effectively. In machine learning algorithms, these functions play vital roles.

Another useful function to create array in Numpy is the **arange()** function. The syntax of the function is as follows:

np.arange(start, end, step)

The argument **end** and **step** is optional and can be used as per requirement. The following example shows how to use this function in different ways.

Example 3:

```
>>>import numpy as np
>>>np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>np.arange(2, 10, dtype=np.float)
array([ 2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.])
>>>np.arange(2, 3, 0.1)
array([ 2.,  2.1,  2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9])
```

Here, the first use of function generates 10 values starting from 0. The second use gives the starting and ending values, and the third use generates the values with equal stepping value.

Another interesting and widely used function is **linspace()**; its syntax is as follows:

linspace(start, end, no. of values)

Example 4:

```
>>>np.linspace(1., 4., 6)
array([ 1.,  1.6,  2.2,  2.8,  3.4,  4. ])
```

In the previous example, the function generates equal spaced values starting from 1 till 4. The number of values generated is 6.

Now that we are familiar with the creation of an array, let's look at how we can access the array elements. This is also popularly called slicing and dicing operations on an array.

Example 5:

```
>>>import numpy as np
>>>a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
>>>print(a)
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
>>>print(a[0:2,1:3])
[[2 3]
 [6 7]]
```

The previous example creates an array and prints a portion of the array. The rule for handling the elements of array is as follows:

arrayA[start:end:step, start:end:step]

Here, the first **start:end:step** is for the rows, and the second **start:end:step** is for the columns of the array. Here, **end:step** is optional. The start and end indicate the row or column number. One thing to remember is that array indexing in Python starts with zero. For example: array **a[0:2:2,:]** indicates display array with row index starting from 0 till 2 with a step of 2. This means alternate rows are displayed. On the same line, **a[0:2,1:3]** indicates rows starting from 0 till 2 and columns starting from 1 till 3. Another important thing to remember is that the last column or row value is not considered while slicing (vertical chopping) or dicing (horizontal chopping) of an array.

There are several common functions that we can use on arrays directly. A short list of these is given in the following table:

Function	Usage
<code>np.add(array1, array2)</code>	Add the two arrays – the respective elements of the array are added
<code>np.subtract(array1, array2)</code>	Subtract the two arrays – the respective elements of the array are subtracted
<code>np.multiply(array1, array2)</code>	Multiply the two arrays – the respective elements of the array are multiplied

<code>np.divide(array1, array2)</code>	Divide the two arrays – the respective elements of the array are divided
<code>np.sqrt(array1)</code>	Displays the square root of every element of array
<code>np.abs(array1)</code>	Displays the absolute value of every element of array
<code>np.log(array1)</code>	Displays the natural logarithm of every element of array
<code>np.sum(array1, axis)</code>	Displays the sum of all elements of array; one can add the elements of a row or column using the parameter <code>axis=0/1</code> , where 0 indicates row and 1 indicates column

Table 3.4: Summary of Important Functions on Array

There are many more useful functions available in Numpy for performing operations on an array. An online search as per the program will yield fruitful results.

Pandas

This is the most useful library for data preprocessing and preparing the data for the machine learning algorithm. The data from various files like CSV, Excel, and such can be easily read using Pandas. The data is available in a spreadsheet-like manner, which makes processing easily. There are three basic data structures at the core of Pandas library:

- **Series:** One-dimensional array-like object containing data and label (or index)
- **Dataframe:** Spreadsheet-like data structure containing an ordered collection of columns that has both a row and column index
- **Panel:** Collection of dataframes but rarely used data structure

Populating the Dataframe

A dataframe can be populated by reading the data from the file. There are

various functions to read data from different files. We will look at the two most popular functions to read data from CSV and Excel file.

Example 6:

```
>>>import pandas as pd  
>>>df=pd.read_csv("d://myfolder//myfile.csv",headers=None,  
names=['Names', 'birthdates'])
```

In the previous example, the data is read from the CSV file stored on the disk. Generally, CSV files do not have headers, that is, the data starts from the first line in the file. So, no headers use the **headers=None** attribute. The **names** attribute is used to specify the headings of the columns of data. The following [figure 3.12](#) shows the snapshot of the output from a **DataFrame** object when viewed in Jupyter notebook.

	Names	birthdates
0	Bob	1968
1	Jessica	1955
2	Mary	1977
3	John	1978
4	Mel	1973
5	Max	2001

Figure 3.12: Output from a DataFrame object in Jupyter Notebook

On the same line, to read the data from the Excel file, use the `read_excel()` function.

Example 7:

```
>>>import pandas as pd  
>>>df=pd.read_excel("d://myfolder//myfile.xlsx")
```

Here, the file contains the headers for the various columns in the Excel on the

first line, so there is no need to specify the name of the columns while reading the file.

Displaying the Data Using Dataframe

Once the data is available in dataframe, it can be viewed using various functions of the **dataframe** object. The following table show the summary of these functions:

Function	Usage
<code>df.head()</code>	Shows the top 5 rows of the data in Dataframe
<code>df.tail()</code>	Shows the bottom 5 rows of the data in Dataframe
<code>df.sample(5)</code>	Shows randomly chosen 5 rows of the data in Dataframe
<code>print(df)</code>	Will print entire data from the Dataframe; not advisable as the data is mostly too large to be displayed
<code>df['column name']</code>	Displays data from that specific column only
<code>df[['column name 1', 'column name 2']]</code>	Displays data from the specified two columns

Table 3.5: Summary of Display functions of Dataframe

Note: Observe the syntax of multiple column display.

Accessing the Data Selectively in Dataframe

The data in the dataframe can be accessed primarily using two functions: **loc** and **iloc**.

Example 8:

```
>>>df.loc[10,'Name']
Dimple

>>>df.loc[0:3,'Name']
0      Bob
1    Jessica
```

```

2      Mary
3      John
Name: Names, dtype: object

```

Here, the data in the ‘**Name**’ column with index value 10 is displayed. The second command displays the values from the ‘**Name**’ column from the rows with index starting from 0 and ending with 3.

Example 9:

```

>>>df.iloc[10,1]
Dimple
>>>df.iloc[0:3,1]
0    1968
1    1955
2    1977
Name: birthdates, dtype: int64

```

Here, using the **iloc** method, we can provide the column names as integer values. The column indices always start with 0 in Python. In both functions, we can use ‘:’ to provide start and end values for the row index and the column index. In short, the syntax for both is as follows:

df.loc[row_index, column_index] where one can specify **row_index** or **column_index** as **start:end:step**

For example, one can write the following to display the data in odd columns, where the number of columns is 10 and the number of rows is 10:

df.iloc[0:9,0:9:2]

The same thing with **loc** function can be achieved by specifying the start and end column names.

Basic Descriptive Statistics using Pandas

Pandas has many inbuilt functions that help understand the data by running the basic statistical functions. These functions and their descriptions are given in the following table:

df.method()	Description
describe	Basic statistics (count, mean, std, min, quantiles, max)
min, max	Minimum and maximum values
mean, median, mode	Arithmetic average, median and mode

var, std	Variance and standard deviation
sem	Standard error of mean
skew	Sample skewness
kurt	Kurtosis

Table 3.6: Basic descriptive statistics

The following example illustrates the use of one of the statistical functions:

Example 9:

```
>>>df.describe()
   Gender
count    8.000000
mean    0.375000
std     0.517549
min     0.000000
25%    0.000000
50%    0.000000
75%    1.000000
max    1.000000
```

The dataframe has only one column named ‘Gender’ and few rows of data. So, the **describe()** function will display the statistical information about the numeric columns indicating the minimum and maximum values in the column.

Data transformation

The real-time data available is in various forms or is a combination of numeric and textual data. This data need transformation before a machine learning algorithm is applied. One way of transforming data is using Pandas **Apply** or **Map** function.

Example 10:

```
>>>mapping = {'male':0,'female':1}
>>>df['Gender']=df['Sex'].apply(lambda x:mapping[x])
```

Here, mapping is a dictionary type of variable with two key-value pairs. The **apply** function uses this dictionary to convert the categorical values (male/female) into numerical values(0 or 1) from the ‘Sex’ column. Here, the mapping is applied on each row of dataframe. The generalized syntax of **map** and **apply** functions are as follows:

```
df['col'].apply(func, axis=0/1)
df['col'].map(func/series)
```

The **col** indicates the column on which the transformation is to be applied. The difference between the two functions is that **apply** can be used on object types series and dataframe both, and the **map** function can be used with only series object type.

Data Preprocessing – Handling missing values

Pandas provide various functions to handle the missing values in the data. To apply machine learning function, these values require preprocessing. The following table shows various methods to handle missing values.

Function	Description
dropna()	Drops missing observations
dropna(how='all')	Drops observations where all cells is NA
dropna(axis=1, how='all')	Drops column if all the values are missing
dropna(thresh = 5)	Drops rows that contain less than 5 non-missing values
fillna(0)	Replaces missing values with zeros
isnull()	Returns True if the value is missing
notnull()	Returns True for non-missing values

Table 3.7: Handling Missing Values

Matplotlib

Matplotlib is an important library of Python that helps in data visualization. Understanding the data is very important for a data scientist before devising any machine learning-based model. This library helps in understanding data in a visual way. Data can be visualized using various graphical methods, like line graph, bar graph and pie chart.

This is 2D plotting library that offers numerous ways to visualize data. Pyplot is the basic library that has all methods for data visualization. This section discusses a few required plots from the point of view of machine learning

applications.

At the center of most matplotlib scripts is **pyplot**; this module is stateful and tracks changes to a figure. All **pyplot** functions revolve around creating or manipulating the state of a figure. The basic function of **pyplot** is **plot()**, which is used for plotting data.

Line Graph

Example 11:

```
>>>import matplotlib.pyplot as plt  
>>>plt.plot([1,2,3,4,5])  
>>>plt.ylabel('some significant numbers')  
>>>plt.show()
```

The preceding code will plot a simple line graph with the values passed in the **plot()** function, as follows:

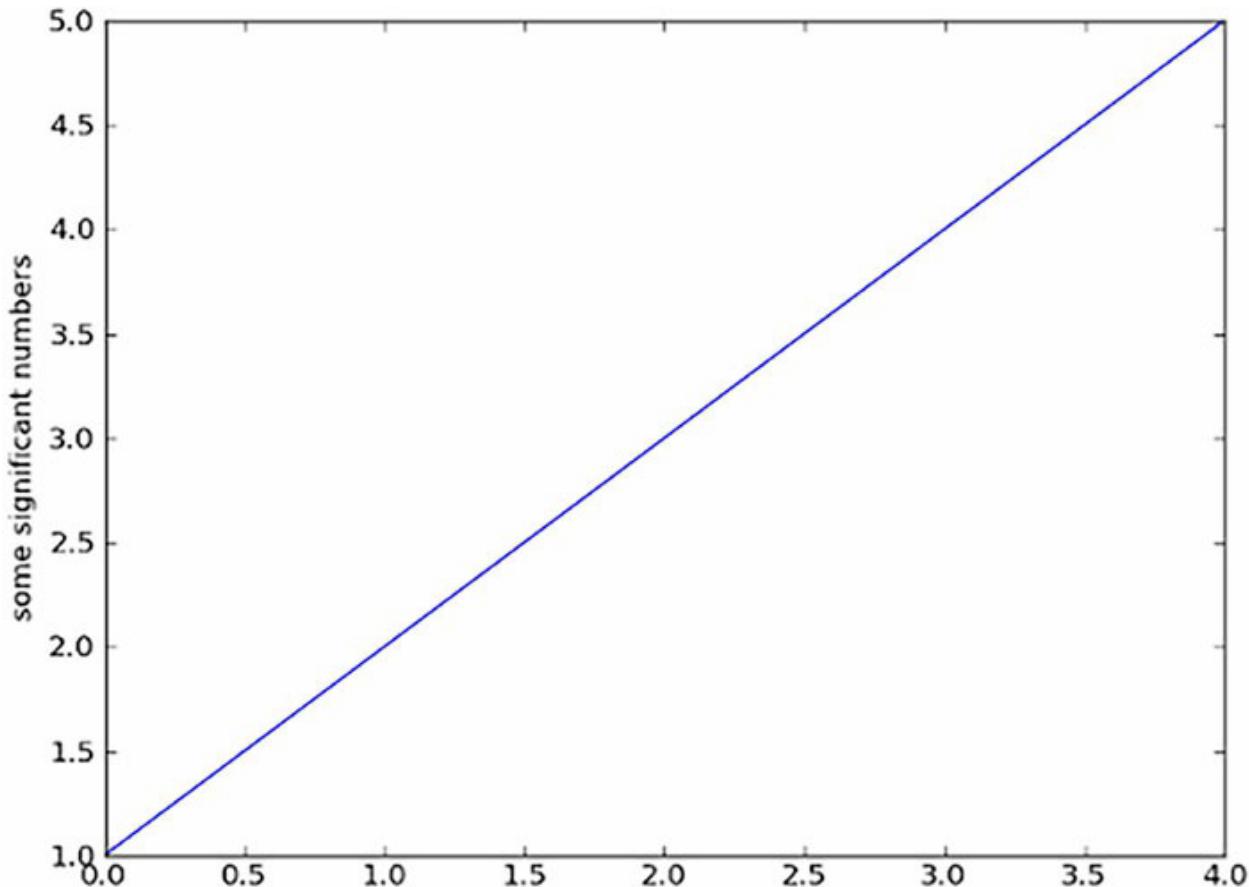


Figure 3.13: A line graph

The following table lists the basic functions required for plotting any graph:

Function	Usage
text(x,y,text)	Adds text in an arbitrary location
xlabel()	Adds text to x-axis
ylabel()	Adds text to y-axis
title()	Adds title to plot
clf()	Clears all figures
axis([left,bottom,width,height])	Indicates the axis dimensions
grid()	Shows grid lines
figure(figsize=[ht, wd in inches], facecolor=backgroundcolor, edgecolor=bordercolor)	Creates a new figure

Table 3.8: Basic functions of pyplot

The following example shows how to use these functions and their effects on the plot.

Example 12:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0,5,11)
y = x**2
plt.figure()
plt.plot(x,y, color = '#4a8c66')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Square of X')
plt.grid()
plt.axis([0,5,0,25])
plt.show()
```

The following figure shows the output of the preceding code:

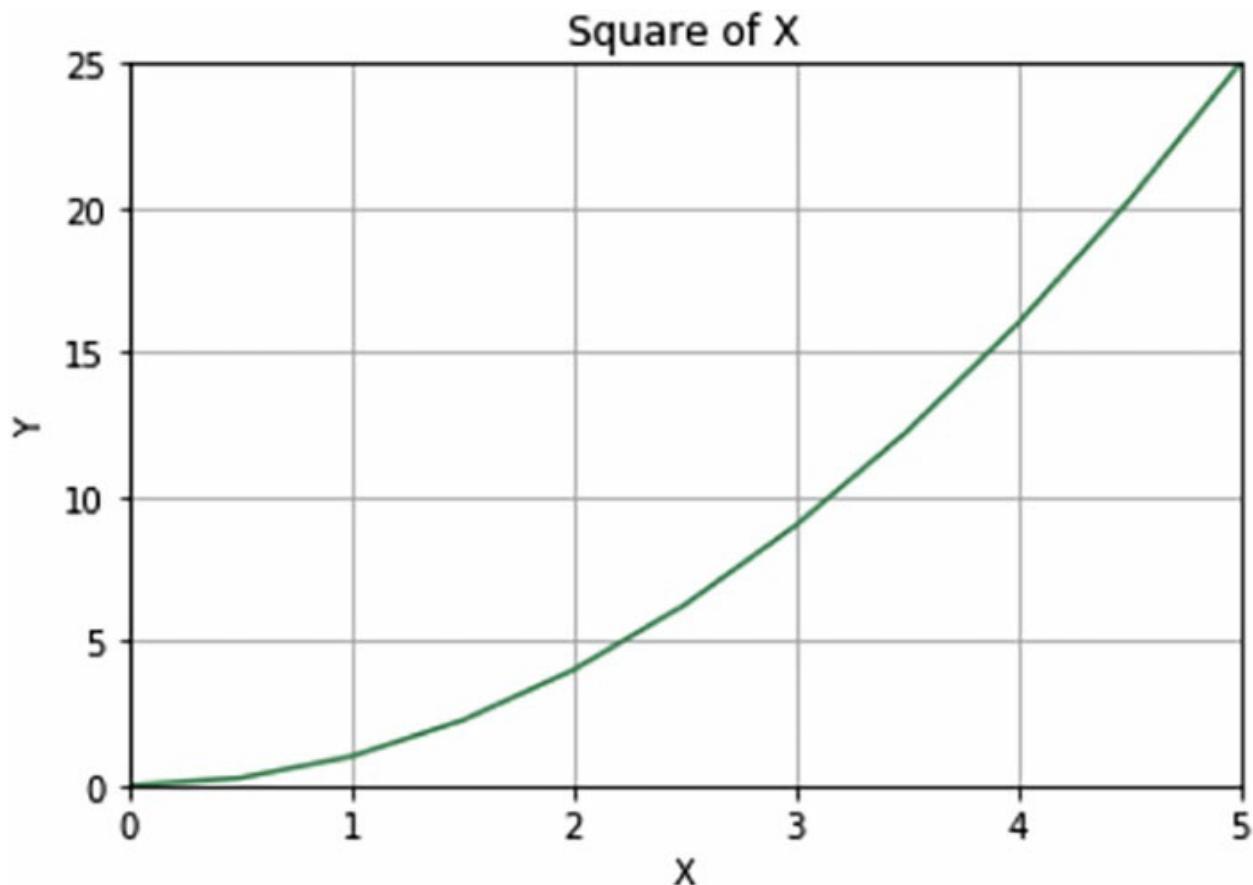


Figure 3.14: Output of example 12

The graph is a simple line graph showing the labels along the X and Y axes, a title of the graph and the grid lines.

Scatter Plot

A scatter plot is a plotting the points along the x and y axes. It is used to show how the data points are placed in two dimensions.

Example 13:

```
x = [5, 10, 7]
y = [12, 6, 15]
x1 = [3, 8, 12]
y1 = [5, 9, 15]
plt.scatter(x,y,color='r', label="Red", marker='o', s=200, alpha=0.2)
plt.scatter(x1,y1,color='g', label="Green")
plt.legend()
plt.show()
```

The preceding code uses the **scatter()**function to plot the data points. One may provide the colors for plotting the points. The plot will not be shown

unless the `show()` function is called.

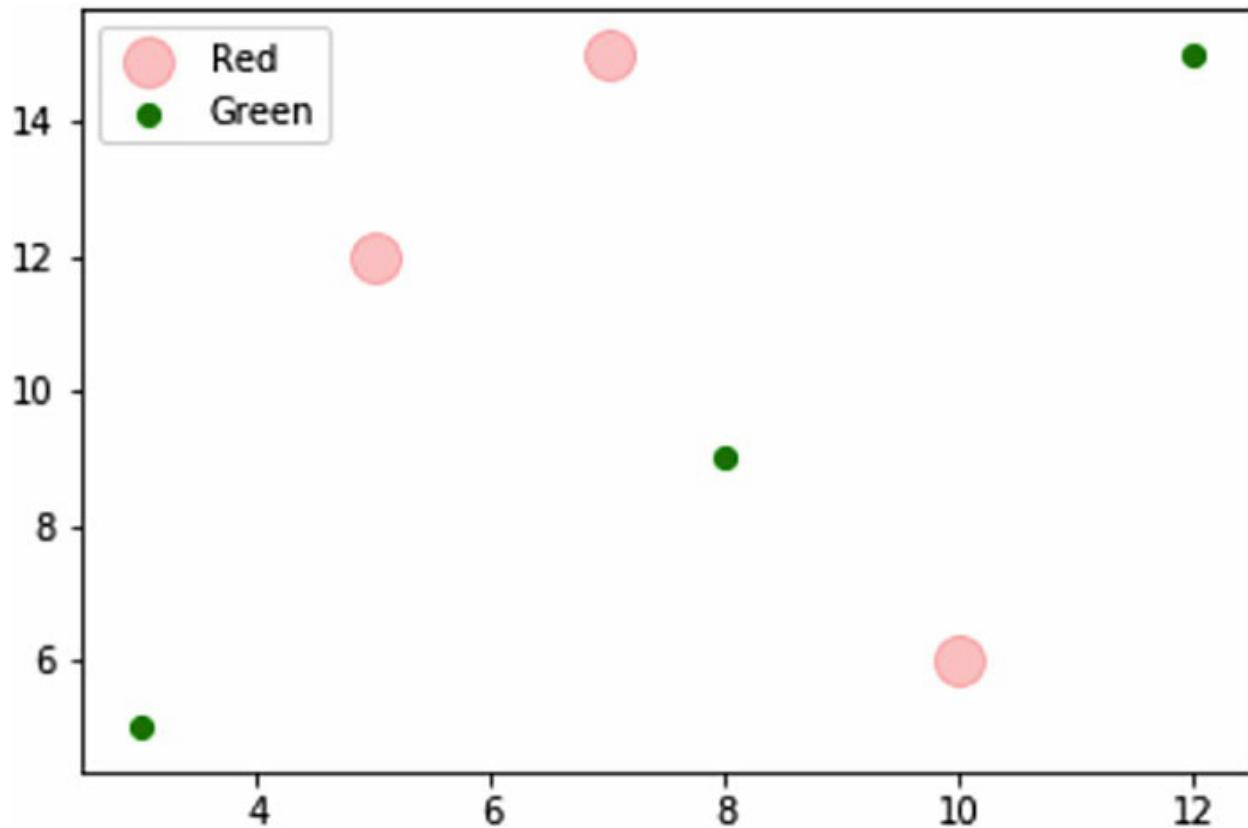


Figure 3.15: Output of example 13 - Scatter Plot

The preceding figure shows the scatter plot. The points with two different colors are plotted.

Bar plot

A bar plot is generally used with categorical data and is the most common type of graph. The length of the bar is proportional to the value of the label. The following example shows the code of a simple bar graph.

Example 14:

```
x = [5, 10, 7]
y = [12, 6, 15]
x1 = [3, 8, 12]
y1 = [5, 9, 15]
plt.bar(x,y,color='r', label="Red")
plt.bar(x1,y1,color='g', label="Green")
plt.show()
```

This example plots two bar graphs with different colors. The output of the

code is shown here:

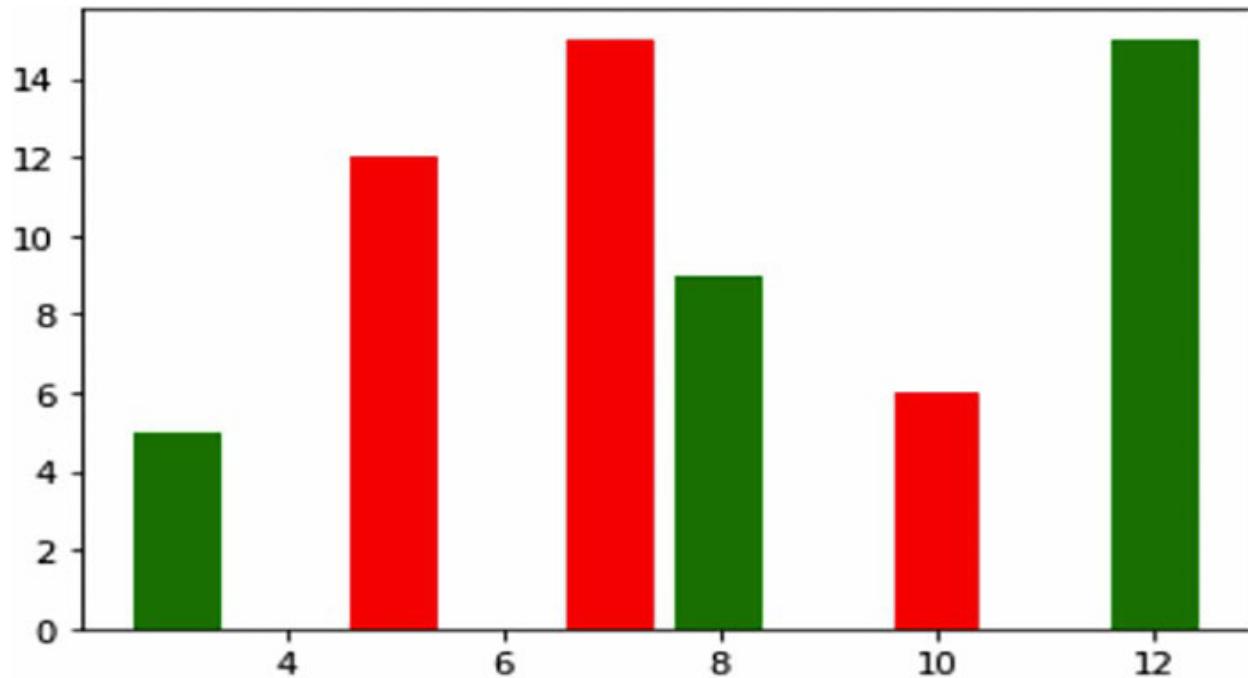


Figure 3.16: Output of example 14 - Bar Plot

The bars represent the locations provided by variable x and variable y represents the length of the bar.

Histogram

Histograms are generally used with images to show the frequency information. This is a very commonly used plot to show the distribution of data. The following example shows the sample code to draw a histogram.

The following code creates 50 random points and plot them. The frequency used here is cumulative, indicating that the next value before plotting is added to the previous value and then plotted.

Example 15:

```
x= 100 + 10*np.random.randn(10000)
plt.hist(x,50, color='g', cumulative=True)
plt.xlabel("People")
plt.ylabel("Frequency")
plt.show()
```

The spread of the data can be visualized very well using a histogram, as shown in the following figure:

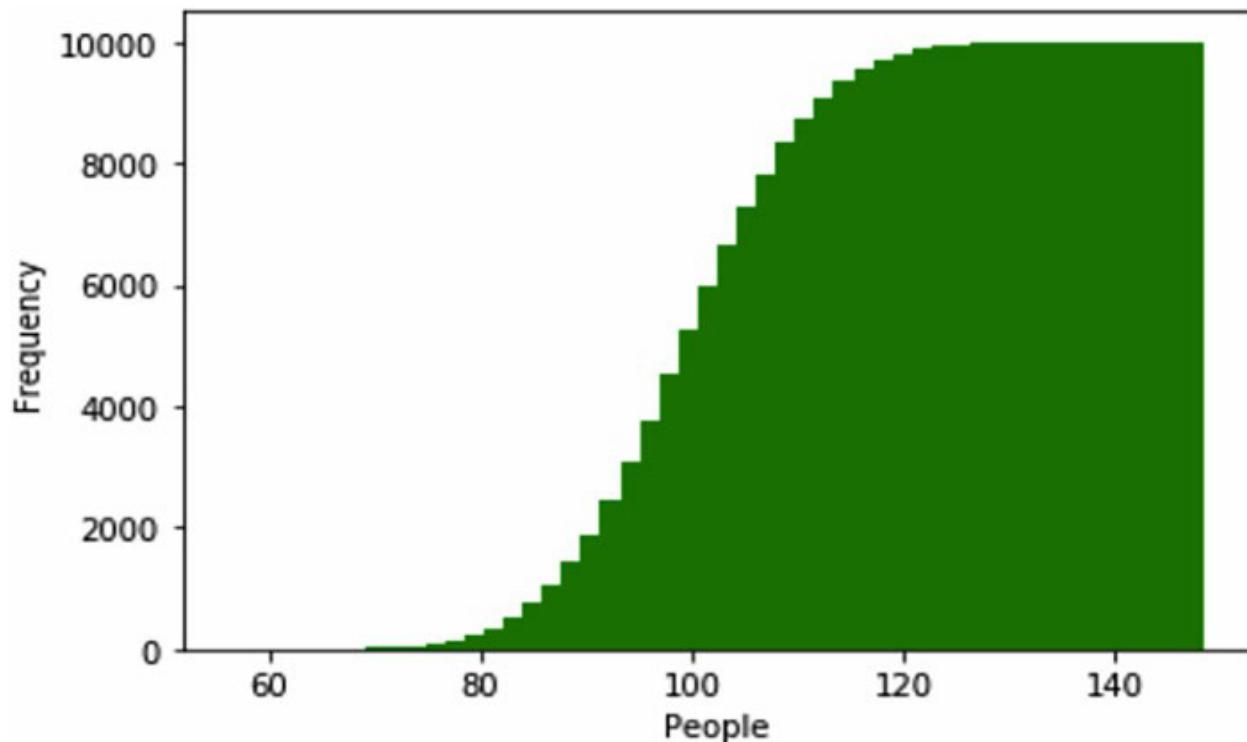


Figure 3.17: Output of example 15 – Histogram

The preceding figure shows the output of the code from example 15.

Pie Chart

The most common plot used is pie chart, which shows size of data using the pie slices. These charts can be best used to show the numerical proportion of the data. The following example shows the Python code to create pie chart using the **Matplotlib** library.

Example 16:

```
country_data= ['UnitedStates', 'GreatBritain', 'China', 'Russia', 'Germany']
medal_data = [46, 27, 26, 19, 17]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#8c564b"]
explode = (0.1, 0, 0, 0, 0)
plt.pie(medal_data, labels=country_data, explode=explode, colors=colors,
autopct='%.1f%%', shadow=True, startangle=140)
plt.title("Gold medal achievements of countries")
plt.show()
```

The output of the previous code is shown in the following figure:

Gold medal achievements of countries

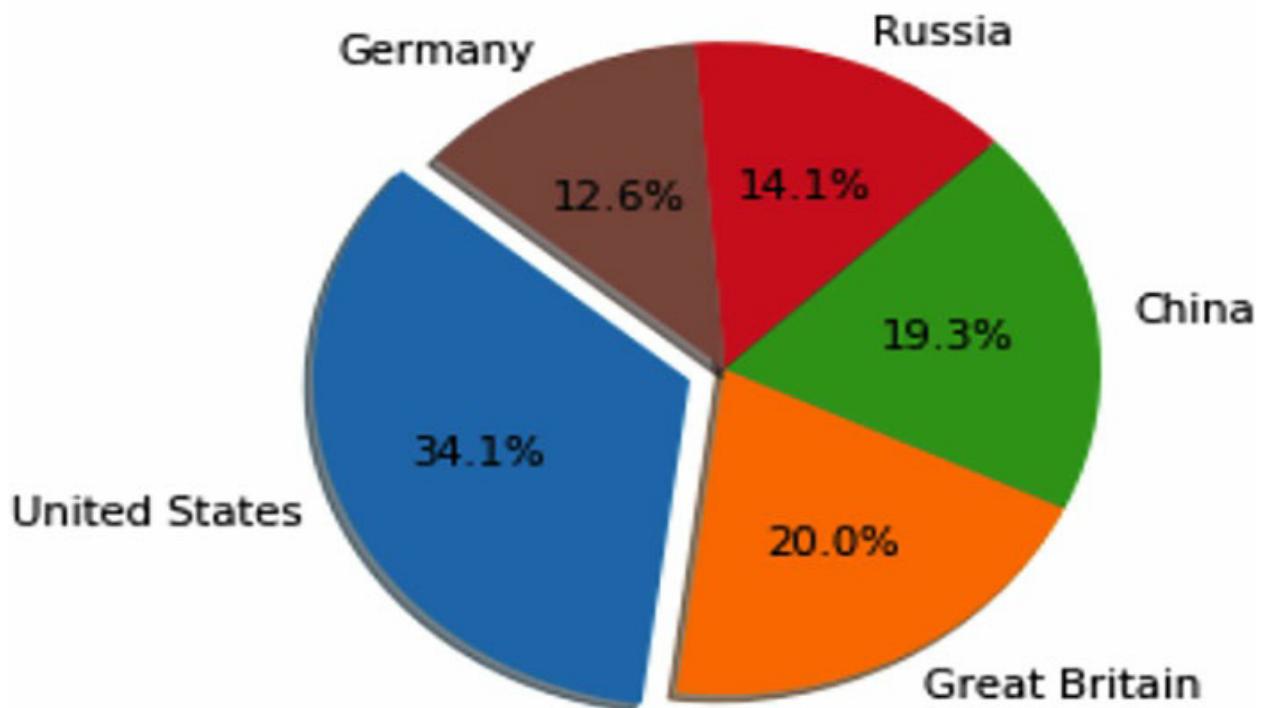


Figure 3.18: Pie Chart

Due to the `explode` property of the `pie()` function, one can easily see the blue portion bulging out. The pie chart started from angle 140, indicated by the `startangle=140` property.

The Matplotlib library provides many other types of visualization techniques, like stacked bar graph and box plot. The plots discussed here are most commonly and widely used to understand or visualize the data effectively.

Evaluation of ML Systems

As discussed in the preceding chapters, once the problem has been defined, the data must be prepared. Data is broadly classified into training and testing data on which various machine learning algorithms can be applied to solve your problem.

Test and Train Datasets

Training and testing data are obtained from the dataset by selecting a random split of 66% of data samples for training and 34% for testing or using other

complicated sampling methods. Trained data are then evaluated against the test set. In order to test the algorithms, we need to define a test harness.

Test harness is the term used to specify the data that would be used for training and testing the algorithm. It evaluates the system against the measure and assesses its performance representing how Learnable the problem statement is. Thus, the major goal of the test harness is to be able to quickly and consistently test the algorithms against a fair representation of the problem being solved and a chosen performance measure.

Performance Measure

It is the measurement of the predictions made by the trained model on the test dataset. Thus, performance measure provides a mechanism to evaluate the solution of the problem under consideration.

A number of performance measures are determined based on the class of problem, such as classification, regression and clustering. These scores determine the relevance and give a meaningful insight into the problem domain.

Model Evaluation Techniques

In order to evaluate the performance of the model, test set data is used as this data has not been seen by the model. Models performance is evaluated based on the following methods:

- **Holdout:** In this method, the model is tested on a data that is different from the training data. The dataset is, thus, randomly divided into three subsets:
 - **Training set** is a subset of the dataset used to build the model.
 - **Validation set** is a subset used to assess the performance of the model built in the training phase.
 - **Test set**, or unseen data, is a subset of the dataset used to assess the likely future performance of a model.

If a model fits the training set much better than it fits the test set, then the model is said to overfit.

- **Cross-Validation:** Cross-validation is a more sophisticated approach

that operates on using the entire transformed dataset to train and test a given algorithm, as compared to using test and train datasets.

- In this technique, the original dataset is divided into two parts: a training set that is used to train the machine learning model, and an independent set used to evaluate the analysis.

Model evaluation metrics

The performance of the machine learning task is obtained by determining how well an algorithm performs on unseen data. The wellness of the algorithm (model) is obtained through its evaluation metrics, which, in turn, depend on the problem to be solved. Most machine learning applications include classification and regression algorithms. Hence, the evaluation metrics used for classification and regression applications include the following.

Classification Metrics

The metrics used in classification problems include the following:

- **Classification accuracy**

Accuracy is defined as the ratio of the number of correct predictions to the total number of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

- **Confusion matrix**

A confusion matrix provides a more detailed split between correct and incorrect classifications for each class. It describes the performance of the model with a matrix as an output.

- **Logarithmic loss or logloss**

Logarithmic loss measures the performance of a multi-class classification model, where the prediction input is a probability value between 0 and 1.

In this evaluation method, the classifier assigns probability to each class for all the samples. Suppose there are N samples belonging to M classes, then the logloss is calculated as follows:

$$\text{Logarithmic loss} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{ij} * \log(p_{ij})$$

Where,

y_{ij} indicates whether sample i belongs to class j

p_{ij} indicates the probability of sample i belonging to class j

Log loss nearer to 0 indicates higher accuracy, while log loss away from 0 indicates lower accuracy.

Area under curve (AUC)

It is one of the most used metrics for the evaluation of binary classification problems, and it can discriminate between positive and negative classes.

True Positive Rate (Sensitivity): True Positive Rate is defined as It specifies the proportion of positive data points that are correctly identified as positive, with respect to all positive data points.

$$\text{True Positive Rate} = \text{True Positive} / (\text{False Negative} + \text{True Positive})$$

False Positive Rate (Specificity) is defined as False Positive Rate It specifies the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$\text{False Positive Rate} = \text{False Positive} / (\text{False Positive} + \text{True Negative})$$

Both False Positive Rate and True Positive Rate have values in the range [0, 1]and are computed at threshold values. AUC has a range of [0, 1]. The greater the value, the better is the performance of the model.

F-Measure: F-measure (also F-score) is a measure of a test's accuracy. It considers the **precision** and the **recall** of the test to compute the score:

- **Precision** is defined as the number of correct positive results divided by the total predicted positive observations.
- **Recall** is defined as the number of correct positive results divided by the number of total actual positives.

F1 Score tries to find the balance between precision and recall, as follows:

Precision: It is the number of correct positive results divided by the number of positive results predicted by the class:

$$\text{Precision} = \text{True Positives} / (\text{True Positives} + \text{False Positives})$$

Recall: It is the number of correct positive results divided by the number of *all* relevant samples (all samples that should have been identified as positive):

$$\text{Recall} = \text{True Positives} / (\text{True Negatives} + \text{False Positives})$$

F1 Score is the harmonic mean between precision and recall. The range for F1 Score is [0, 1]. It indicates the instances when the classifier classifies correctly and does not miss a significant number of instances. The greater the F1 Score, the better is the performance of the model.

Regression Metrics

Regression problems are evaluated by mean absolute error and root mean squared error:

- **Mean Absolute Error (MAE)** is defined as the sum of the absolute differences between predictions and actual values.

It gives the measure of how far the predictions were from the actual output. However, they don't give any idea of the direction of the error, that is, whether the data is under predicted or over predicted.

- **Root Mean Squared Error (RMSE)** measures the average magnitude of the error. This is determined by taking the square root of the average of squared differences between the prediction and the actual observation.

As the square of the error is taken, the effect of larger errors becomes more pronounced than smaller errors. Hence, the model can now focus more on the larger errors.

Conclusion

This chapter focused on various ML-based algorithms and gave an insight into ways programs can be developed. It begins by providing the concepts of ML modelling flow and its preprocessing. Supervised and unsupervised learning algorithms were further elaborated on through the regression, classification and clustering algorithms, followed by the neural networks and support vector machine algorithms. The formats in which the algorithms need to be represented were elaborated on through the various data science

libraries in Python. The chapter concluded by determining the best algorithm by evaluating the algorithms utilized for a given problem statement.

The next chapter deals with application areas where machine learning is applied. The concepts explained in the previous chapters will be elaborated through several case studies.

Questions

1. Explain the steps involved in machine learning modelling flow.

Ans. This method involves six steps:

1. Problem identification
2. Data wrangling
3. Exploratory data analysis
4. Preprocessing and training data development
5. Modeling
6. Documentation

Step One: Problem Identification

This is the process of defining a problem you want to solve and identifying what you want to accomplish by solving that problem.

Step Two: Data Wrangling

Data wrangling involves taking raw data and preparing it for processing and analysis.

Step Three: Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach for summarizing and visualizing the important characteristics and statistical properties of a dataset. Visualizing the data will help you make sense of it to identify emerging themes. Identifying these trends will help you to form hypotheses about the data.

Step Four: Preprocessing and Training Data Development

Data preprocessing refers to the process of removing things like out-of-value ranges and impossible combinations from your dataset. This is an important step because analyzing data that hasn't been properly screened can lead to misleading or incorrect results. This step also

includes the process of splitting the dataset into testing and training subsets.

Step Five: Modeling

Modeling is the fifth step. During this step, you'll leverage your cleaned and processed data to make predictive insights.

Note that modeling involves both model training and selection, and model deployment.

Step Six: Documentation

How you express a message is often just as important, if not more important, than the message itself. Even if you find a really valuable insight through data analysis, you're likely to have a hard time putting that insight into action if you don't communicate your thoughts about it properly. That's where storytelling comes in: when you organize your insights into a good story and tell that story effectively, you'll be much more likely to positively impact the project you're working on.

2. Describe the various categories of ML algorithms.

Ans. When it comes to the taxonomy of types of machine learning, a good account breaks up methods across six dimensions:

1. Supervised vs unsupervised learning
2. Batch vs online learning
3. Instance-based vs model-based learning

In a supervised learning model, an algorithm is given a labeled training dataset and evaluates its ability to understand the data using an answer key. An unsupervised learning model features an algorithm that tries to make sense of an unlabeled dataset on its own without training.

Batch Learning: An ML algorithm performs batch learning if the system can't learn incrementally and must be trained using all the available data. Since this takes both time and computation power, the learning process typically occurs offline (this is known as 'offline learning'). The system is trained and then launched; it doesn't continue to learn after it has been launched. If new data is acquired, a new version of the system needs to be trained to replace the predecessor. Data scientists can automate the training, evaluation, and launch of ML systems that

use batch learning.

Online Learning, this trains the system by breaking the data into small groups and feeding the system those groups over a longer period of time. The learning is broken up into individually cheap and fast steps, which allows for receiving large amounts of data in real time.

You can set the learning rate (that is, the speed at which the system adapts to changing data) yourself, but you need to be judicious. If you make this value too high, your system will quickly adapt to the new data at the expense of learning done on previous data; if you set it too low, it will learn about the new data too slowly to be effective.

Instance-based and model-based learning systems differ in their approach to generalization. The former type of system learns the examples ‘by heart’ before generalizing to new cases using a metric of similarity. The latter generalizes from a set of examples to build a model of those examples before using that model to make predictions.

As the model-based learning form is much more common, this resource will look at a few examples of the little-known instance-based systems. Model-based learning will become more familiar to you as you work through the ML-focused units. As you’ll find, machine learning often involves scrutinizing the data, selecting a model, training it on the training data and applying the model to make predictions about new cases, trusting that the model generalizes effectively.

3. Differentiate between the following:

a. Simple and multi regression

Both are regression problems, and both have the same assumptions, but the major difference comes in terms of the number of independent variables. In simple regression, the number of independent variables is one, but in multi regression, the number is more than one.

In simple regression:

$$Y = mx + c$$

where, m represents the slope.

In multi regression:

$$Y = b_0x_1 + b_1x_2 + \dots + c$$

Equation work.

Multi regression problems have wide scope of practical use, while simple regression is only suitable for understanding the concepts behind regression.

b. SVM and neural networks

SVM: Many advanced algorithms handle more complex datasets, and **Support Vector Machines (SVM)** are among the most popular. SVMs are supervised learning models that analyze data to determine the data's class (you could use an SVM to figure out that 'carrot' belongs to the 'vegetable' class and 'cat' belongs to the 'animal' class).

A kernel generally refers to the kernel trick — when you use a linear classifier to solve a nonlinear problem. You can also use kernels to compute similarities between different data in a set. Refer to the following figure:

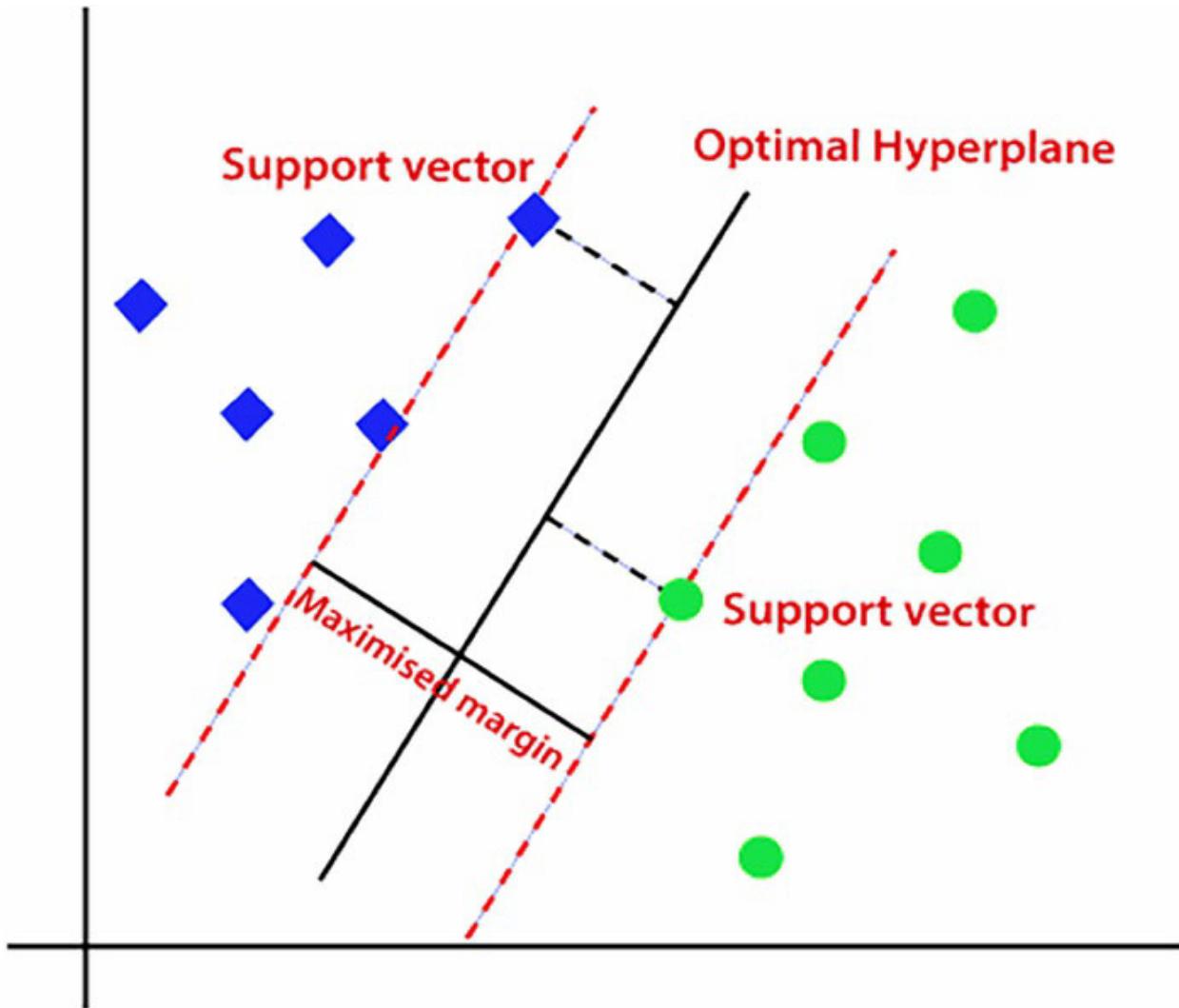


Figure 3.19: Support vector machines

In support vector machines, we use kernel trick, which increases the dimension to make the nonlinear value a linear value. The purpose of doing this is that the algorithms have to segregate the classes by hyperplane. Hyperplane can be 2-D, 3-D or more, depending on the input class dimensions.

The main goal of SVM is to maximize the margin between two support vectors.

Neural Network: Deep learning, which is a subset of machine learning, is the human brain embedded in a machine. It is inspired by the working of a human brain and therefore, is a set of neural network algorithms that tries to mimic the working of a human brain and learn from the experiences.

A neural network is a computational learning system that uses a network of functions to understand and translate a data input of one form into the desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons in the human brain function together to understand inputs from human senses.

Introduction: Deep learning, which is a subset of machine learning, It is inspired by the working of a human brain and therefore, is a set of neural network algorithms that tries to mimic the human brain and learn from the experiences.

In this section, we will learn how a basic neural network works and how it improves itself to make the best predictions.

Artificial Neural Networks and Its components: A neural network is a computational learning system that uses a network of functions to understand and translate a data input of one form into the desired output, usually in another form. In simple words, a neural network is a set of algorithms that tries to recognize the patterns in, relationships between and information from data through a process that is inspired by and works like the human brain/biology. Refer to the following figure:

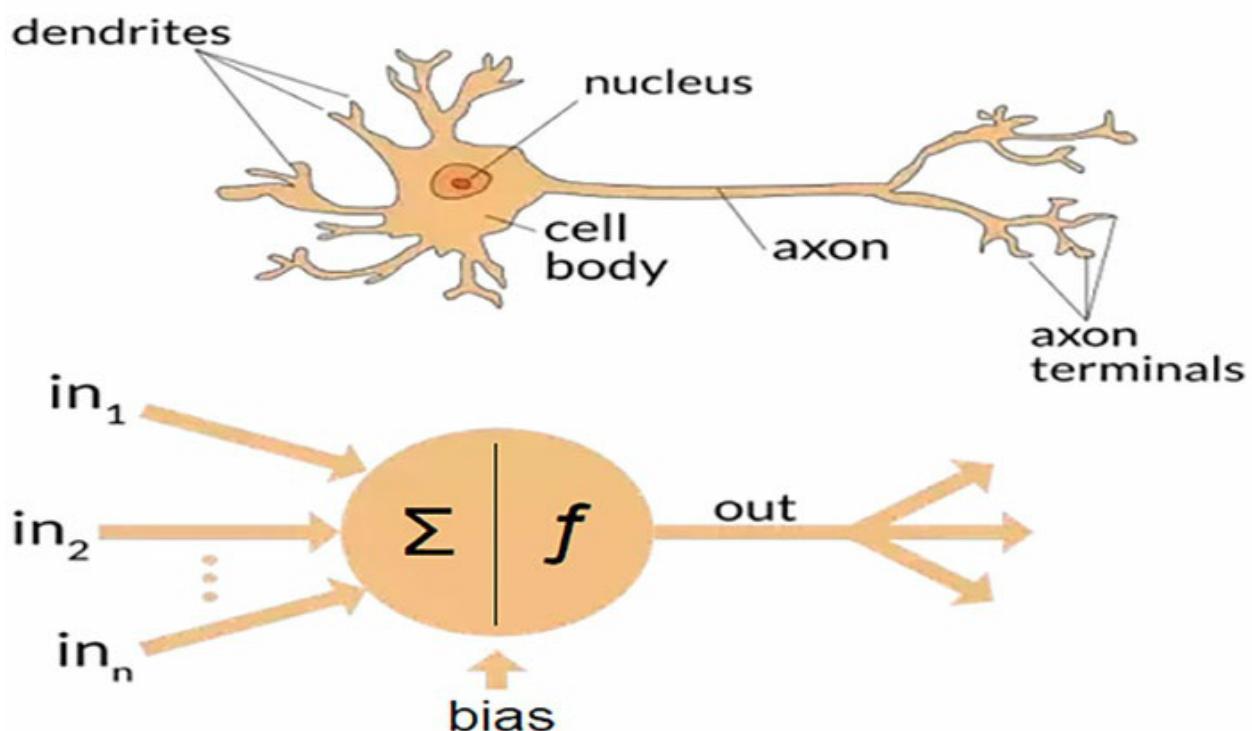


Figure 3.20: Artificial Neural Networks

Components/Architecture of Neural Network: A simple neural network consists of three components:

- Input layer
- Hidden layer
- Output layer

Input Layer: Input nodes are the inputs/information from the outside world provided to the model to learn and derive conclusions from. Input nodes pass the information to the next layer, that is, the hidden layer.

Hidden Layer: This is the set of neurons where all the computations are performed on the input data. There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.

Output layer: The output layer is the output/conclusions of the model derived from all the computations performed. There can be single or multiple nodes in the output layer. If we have a binary classification problem, the output node is 1, but in the case of multi-class classification, the output nodes can be more than 1.

4. Explain the various classification and clustering algorithms in detail.

Ans. Logistic regression

Logistic regression is a calculation used to predict a binary outcome: either something happens, or it does not. This can be exhibited as Yes/No, Pass/Fail, Alive/Dead, and so on.

Independent variables are analyzed to determine the binary outcome, with the results falling in one of two categories. The independent variables can be categorical or numeric, but the dependent variable is always categorical. The variables are represented as:

$$P(Y = 1|X) \text{ or } P(Y = 0|X)$$

It calculates the probability of dependent variable Y , given independent variable X .

This can be used to calculate the probability of a word having a positive or negative connotation (0, 1, or on a scale between). Or it can be used

to determine the object contained in a photo (tree, flower, grass, and so on.), with each object given a probability between 0 and 1.

Naive Bayes

Naive Bayes calculates the possibility of whether a data point belongs within a certain category. In text analysis, it can be used to categorize words or phrases as belonging to a preset “tag” (classification) or not. Consider the following example:

Text	Tag
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

Figure 3.21: Text analysis

To decide whether or not a phrase should be tagged as “sports,” you need to calculate: determine *the probability of A, if B is true, is equal to the probability of B, if A is true, times the probability of A being true, divided by the probability of B being true.*

K-nearest Neighbors

K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples.

When k-NN is used in classification, you place data within the category of its nearest neighbor. If k = 1, then it would be placed in the class nearest 1. K is classified by a plurality poll of its neighbors.

Decision Tree

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it’s able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from “tree trunk” to “branches,” to “leaves,” where

the categories become more finitely similar. This creates categories within categories, enabling organic classification with limited human supervision.

To continue with the sports example, this is how the decision tree works:

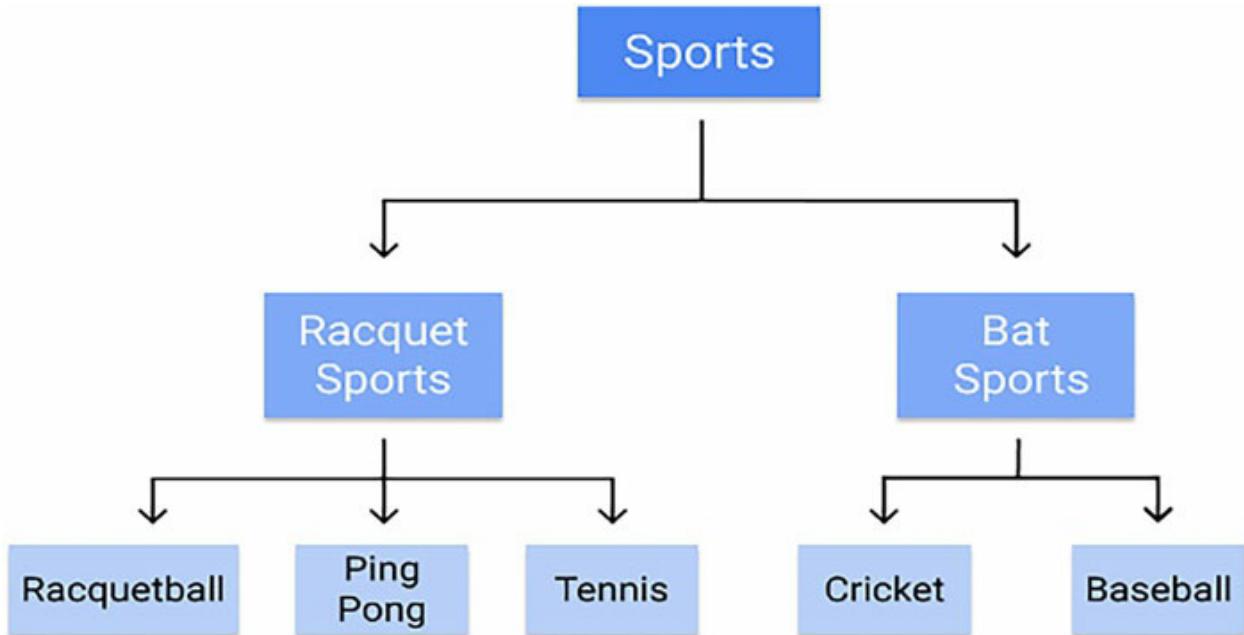


Figure 3.22: Working of a decision tree

Random Forest

The random forest algorithm is an expansion of decision tree in that you first construct a multitude of decision trees with training data and then fit your new data within one of the trees as a “random forest.”

It averages your data to connect it to the nearest tree on the data scale. Random forest models are helpful as they remedy for the decision tree’s problem of “*forcing*” data points within a category unnecessarily.

Support Vector Machines (SVM)

It uses algorithms to train and classify data within degrees of polarity, taking it to a degree beyond X/Y prediction.

For a simple visual explanation, we will use two tags, i.e., *red* and *blue*, with two data features, i.e., *X* and *Y*, and then train our classifier to output an *X/Y* coordinate as either *red* or *blue*.

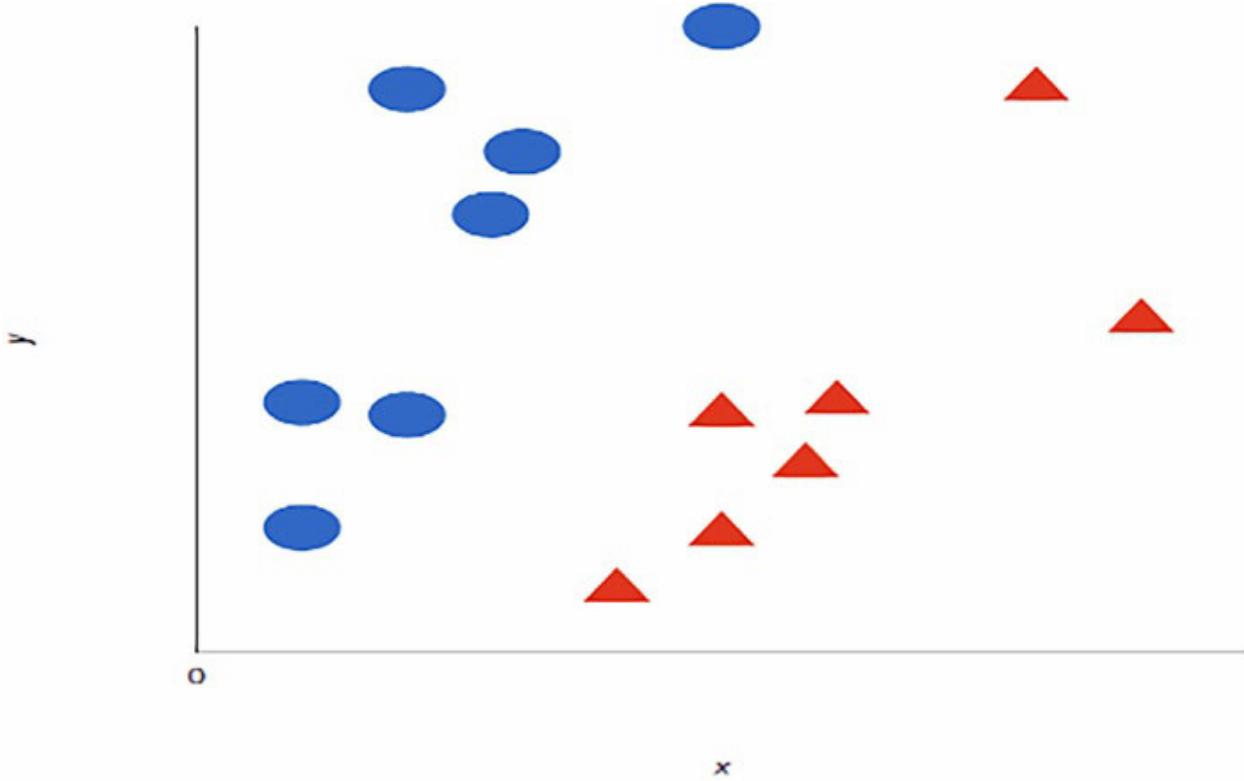


Figure 3.23: red and blue tags with X and Y data features

The SVM then assigns a hyperplane that best separates the tags. In two dimensions, this is simply a line. Anything on one side of the line is *red*, and anything on the other side is *blue*. In sentiment analysis, for example, this would be *positive* and *negative*.

In order to maximize machine learning, the best hyperplane is the one with the largest distance between each tag:

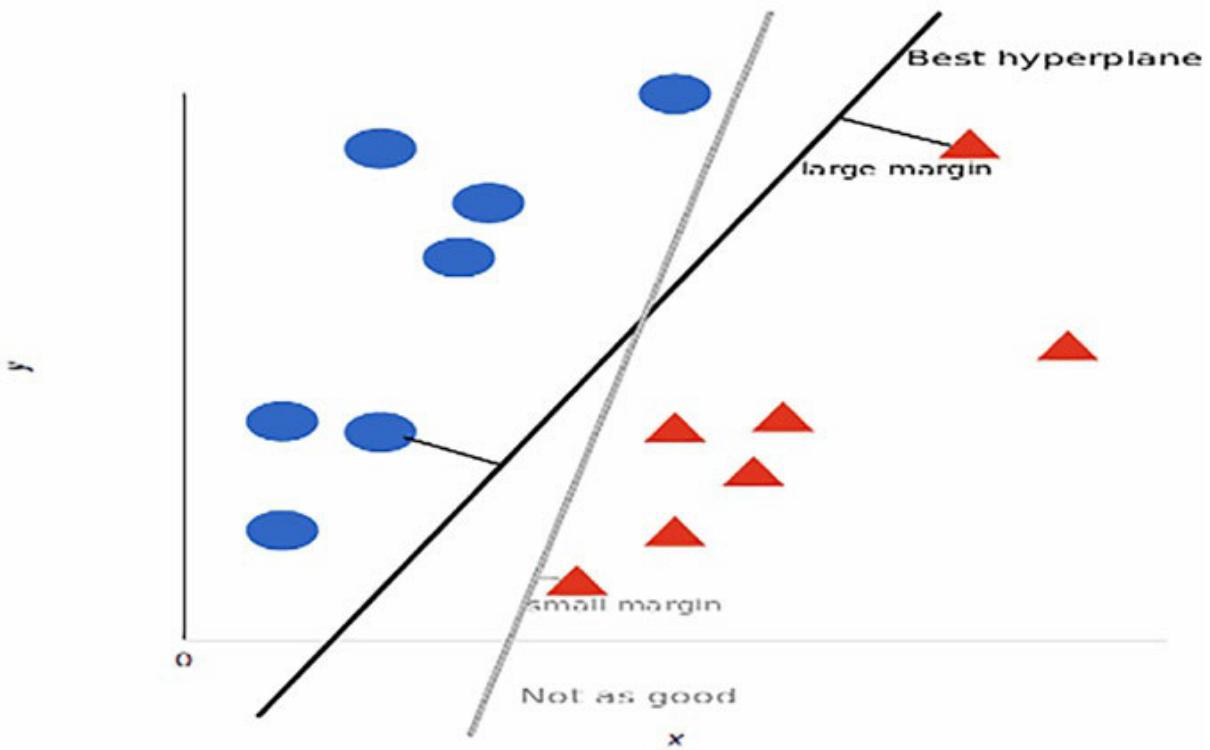


Figure 3.24: Hyperplane with largest distance between each tag

However, as data sets become more complex, it may not be possible to draw a single line to classify the data into two camps:

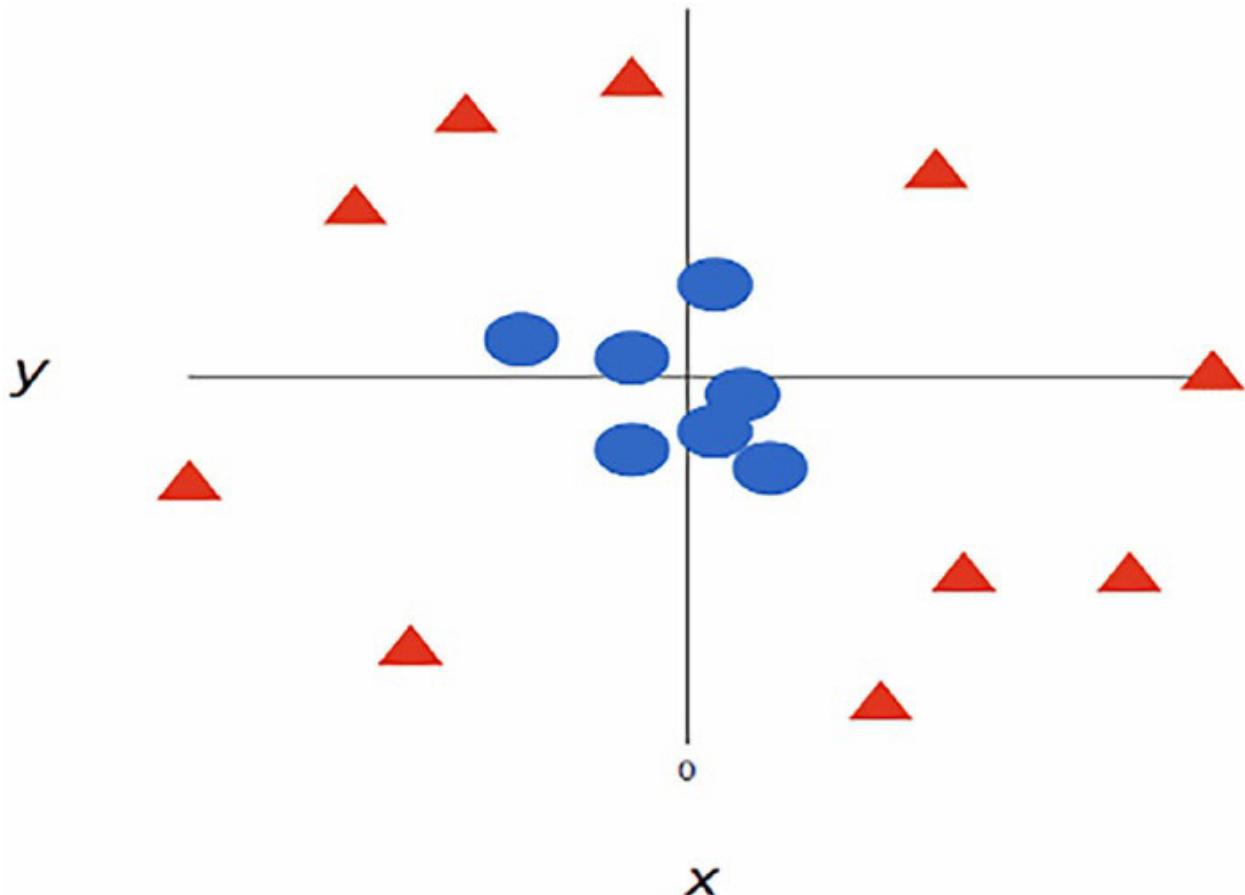


Figure 3.25: More complex dataset

Using SVM, the more complex the data, the more accurate the predictor will become. Imagine the previous figure in three dimensions, with a Z-axis added, so it becomes a circle.

Mapped back to two dimensions with the best hyperplane, it looks as shown in the following image:

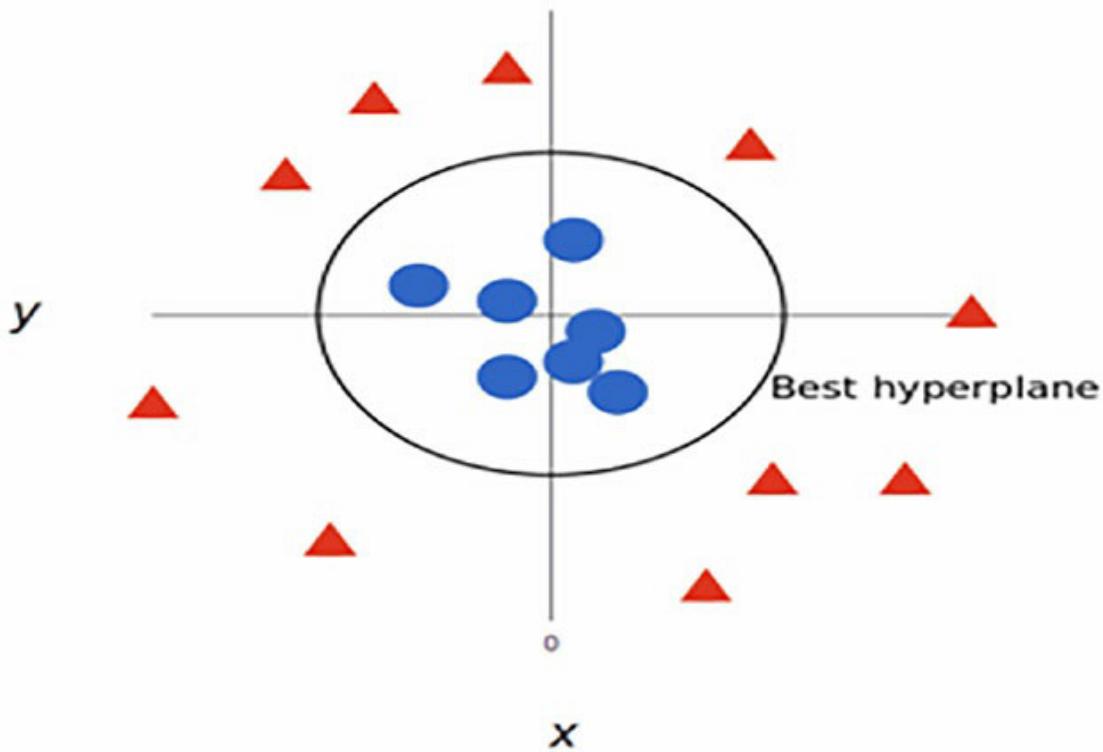


Figure 3.26: Mapped back to two dimensions

SVM allows for more accurate machine learning because it's multidimensional.

Clustering algorithms

Following are the clustering algorithms:

K-Means Clustering

K-Means is probably the most well-known clustering algorithm. It's taught in many introductory data science and machine learning classes and is easy to understand and implement in code:

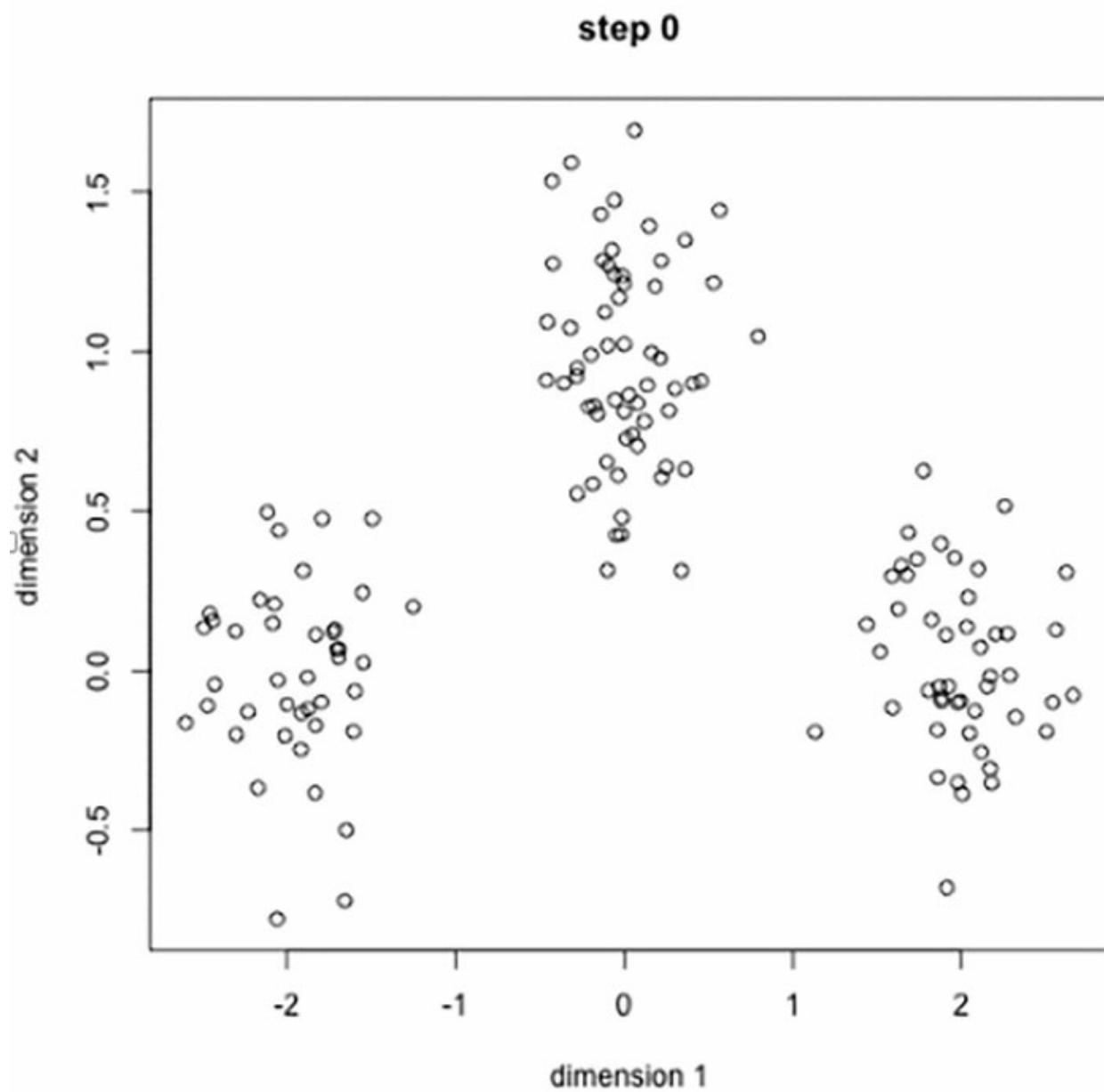


Figure 3.27: K-means clustering

K means algorithm can be represented by the following steps:

1. we select several classes/groups to use and randomly initialize their respective center points. To figure out the number of classes to use, it's good to take a quick look at the data and identify any distinct groupings. The center points are vectors of the same length as each data point vector and are the "X's" in the preceding image.
2. Each data point is classified by computing the distance between that

point and each group center; the point is put in the group whose center is the closest to it.

3. Based on these classified points, we recomputed the group center by taking the mean of all the vectors in the group.
4. Repeat these steps for a set number of iterations or until the group centers don't change much between iterations. You can also opt to randomly initialize the group centers a few times, and then select the run that looks like it provided the best results.

K-Means has the advantage that it's pretty fast, as all we're really doing is computing the distances between points and group centers; very few computations! This, it has linear complexity $O(n)$.

On the other hand, K-Means has a couple of disadvantages. Firstly, you have to select how many groups/classes there are. Ideally, with a clustering algorithm, we'd want to figure it out as the aim is to gain some insight from the data. K-means also starts with a random choice of cluster centers; therefore, it may yield different clustering results on different runs of the algorithm. Thus, the results may not be repeatable and lack consistency. Other cluster methods are more consistent.

K-Medians is another clustering algorithm related to K-Means, except we use the median vector of the group instead of recomputing the group center points using the mean. This method is less sensitive to outliers (because of using the median) but is much slower for larger datasets as sorting is required on each iteration when computing the median vector.

Mean-Shift Clustering

Mean shift clustering is a sliding-window-based algorithm that attempts to find dense areas of data points. It is a centroid-based algorithm, which means the goal is to locate the center point of each group/class; it works by updating candidates for center points to be the mean of the points within the sliding-window. These candidate windows are then filtered in a post-processing stage to eliminate near-duplicates, forming the final set of center points and their corresponding groups:

Find centres

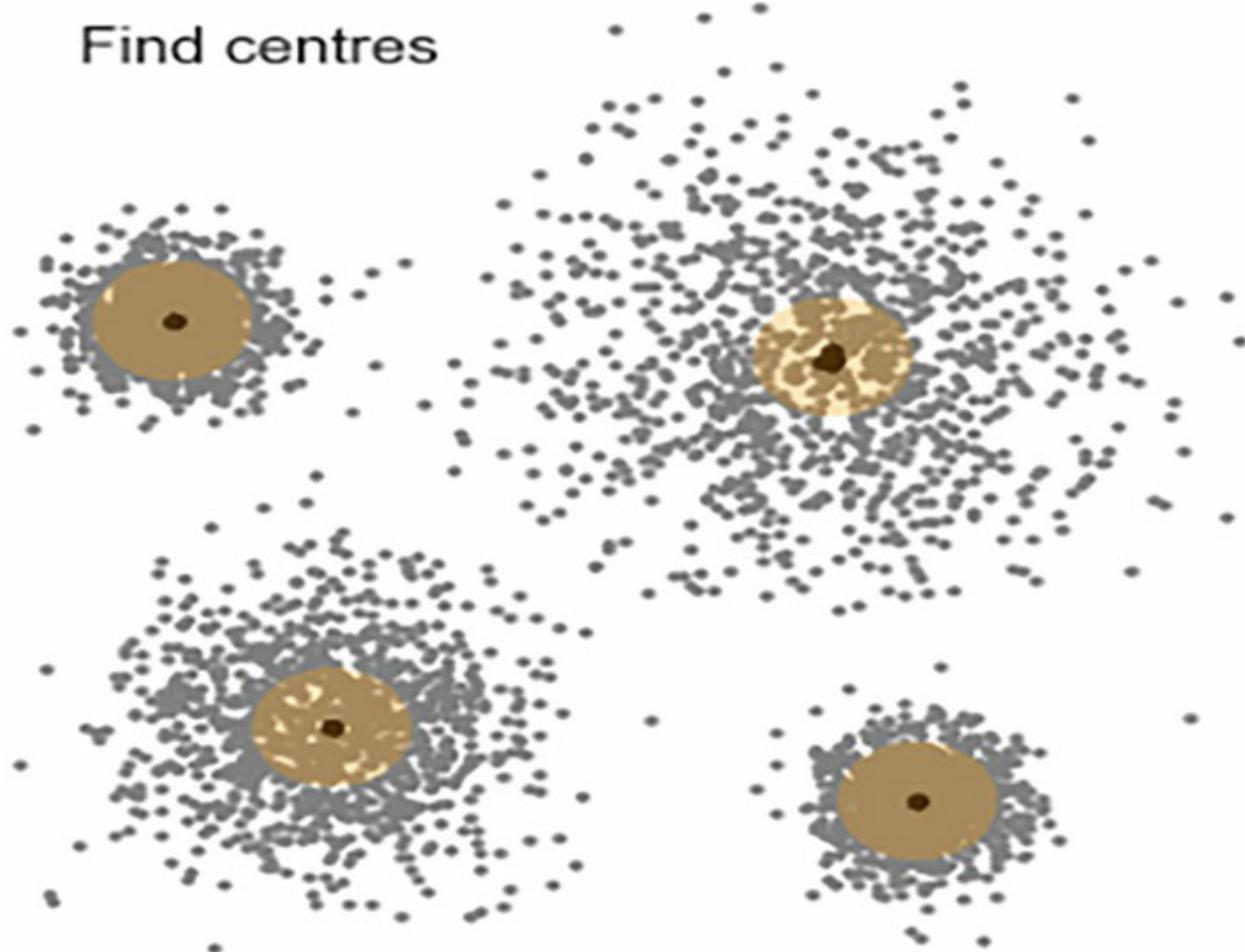


Figure 3.28: Mean-shift Clustering

Mean shift clustering can be represented as follows:

1. To explain mean-shift, we will consider a set of points in two-dimensional space, like in the preceding illustration. We begin with a circular sliding window centered at a point C (randomly selected) and having radius r as the kernel. Mean shift is a hill-climbing algorithm that involves shifting this kernel iteratively to a higher density region on each step until convergence.
2. At every iteration, the sliding window is shifted toward regions of higher density by shifting the center point to the mean of the points within the window (hence the name). The density within the sliding window is proportional to the number of points inside it. Naturally, by shifting to the mean of the points in the window, it will gradually move toward areas of higher point density.

3. We continue shifting the sliding window according to the mean until there is no direction in which a shift can accommodate more points inside the kernel. Check out the preceding graphic; we keep moving the circle until we are no longer increasing the density (that is, number of points in the window).
4. This process of steps 1 to 3 is done with many sliding windows until all points lie within a window. When multiple sliding windows overlap, the window containing the most points is preserved. The data points are then clustered according to the sliding window in which they reside.

In contrast to K-means clustering, there is no need to select the number of clusters as mean-shift automatically discovers this. That's a massive advantage. The fact that the cluster centers converge toward the points of maximum density is also quite desirable as it is quite intuitive to understand and fits well in a naturally data-driven sense. The drawback is that the selection of the window size/radius “r” can be non-trivial.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a density-based clustered algorithm similar to mean-shift, but with a couple of notable advantages.

1. DBSCAN begins with an arbitrary starting data point that has not been visited. The neighborhood of this point is extracted using a distance epsilon ϵ (all points that are within the ϵ distance are neighborhood points).
2. If there is a sufficient number of points (according to minPoints) within this neighborhood, the clustering process starts and the current data point becomes the first point in the new cluster. Otherwise, the point will be labeled as noise (this noisy point might later become part of the cluster). In both cases, that point is marked as “visited”.
3. For this first point in the new cluster, the points within its ϵ distance neighborhood also become part of the same cluster. This procedure of making all points in the ϵ neighborhood belong to the same cluster is then repeated for all the new points that have been just added to the cluster group.
4. Steps 2 and 3 are repeated until all points in the cluster are determined,

that is, all points within the ϵ neighborhood of the cluster have been visited and labeled.

Once we're done with the current cluster, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. This process continues until all points are marked as visited. Since all points will have been visited at the end of this, each point will have been marked as either belonging to a cluster or being noise.

DBSCAN has some great advantages over other clustering algorithms. Firstly, it does not require a pre-set number of clusters at all. It also identifies outliers as noises, unlike mean-shift, which simply throws them into a cluster even if the data point is very different. Additionally, it can find arbitrarily sized and arbitrarily shaped clusters quite well.

The main drawback of DBSCAN is that it doesn't perform as well as others when the clusters are of varying density. This is because the setting of the distance threshold ϵ and minPoints for identifying the neighborhood points will vary from cluster to cluster when the density varies. This drawback also occurs with very high-dimensional data since the distance threshold ϵ becomes challenging to estimate again.

Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

One of the major drawbacks of K-Means is its naive use of the mean value for the cluster center. We can see why this isn't the best way of doing things by looking at the image below. On the left-hand side, it looks quite obvious to the human eye that there are two circular clusters with different radii centered at the same mean. K-Means can't handle this because the mean values of the clusters are close together. K-Means also fails in cases where the clusters are not circular, again as a result of using the mean as cluster center.

Gaussian Mixture Models (GMMs) give us more flexibility than K-Means. With GMMs, we assume that the data points are Gaussian distributed; this is a less restrictive assumption than saying they are circular using the mean. That way, we have two parameters to describe the shape of the clusters: the mean and the standard deviation! Taking an example in two dimensions, this means that the clusters can take any kind of elliptical shape (since we have a standard deviation in both the x

and y directions). Thus, each Gaussian distribution is assigned to a single cluster.

To find the parameters of the Gaussian for each cluster (for example, the mean and standard deviation), we will use an optimization algorithm called **Expectation–Maximization (EM)**. Take a look at the following image as an illustration of the Gaussians being fitted to the clusters. Then we can proceed with the process of Expectation–Maximization clustering using GMMs:

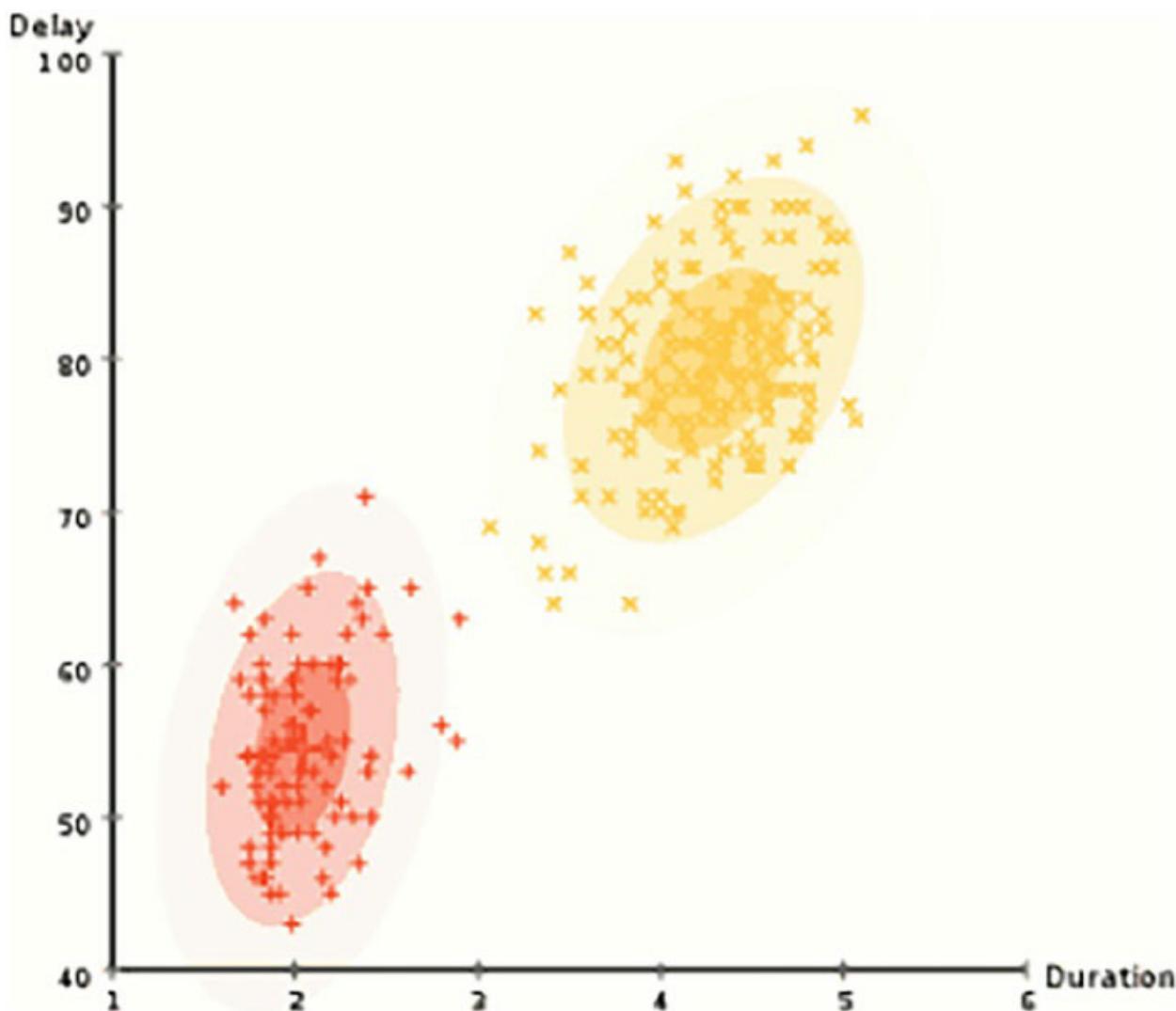


Figure 3.29: Gaussians being fitted to the clusters

1. EM algorithm begins by selecting the number of clusters (like K-Means does) and randomly initializing the Gaussian distribution parameters for each cluster. One can try to provide a good guesstimate for the initial

parameters by taking a quick look at the data too. Given these Gaussian distributions for each cluster, compute the probability that each data point belongs to a particular cluster. The closer a point is to the Gaussian's center, the more likely it belongs to that cluster. This should make intuitive sense since with a Gaussian distribution, we are assuming that most of the data lies closer to the center of the cluster.

2. Based on these probabilities, we compute a new set of parameters for the Gaussian distributions such that we maximize the probabilities of data points within the clusters. We compute these new parameters using a weighted sum of the data point positions, where the weights are the probabilities of the data point belonging in that particular cluster. To explain this visually, we can look at the preceding image, in particular, the yellow cluster, as an example. The distribution starts off randomly on the first iteration, but we can see that most yellow points are to the right of that distribution. When we compute a sum weighted by the probabilities, even though there are some points near the center, most of them are on the right. Thus, naturally, the distribution's mean is shifted closer to those set of points. We can also see that most points are "top-right to bottom-left". Therefore, the standard deviation changes to create an ellipse that is more fitted to these points, to maximize the sum weighted by the probabilities.
3. Steps 2 and 3 are repeated iteratively until convergence, where the distributions don't change much from iteration to iteration.

There are two key advantages to using GMMs. Firstly, GMMs are a lot more **flexible** in terms of **cluster covariance** than K-Means; due to the standard deviation parameter, the clusters can take on any ellipse shape, rather than being restricted to circles. K-Means is actually a special case of GMM in which each cluster's covariance along all dimensions approaches 0. Secondly, since GMMs use probabilities, they can have multiple clusters per data point. So if a data point is in the middle of two overlapping clusters, we can simply define its class by saying it belongs X-percent to class 1 and Y-percent to class 2, that is, GMMs support **mixed membership**.

Agglomerative Hierarchical Clustering

Hierarchical clustering algorithms fall into two categories: top-down or

bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or *agglomerate*) pairs of clusters until all of them have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is, therefore, called *hierarchical agglomerative clustering* or *HAC*. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, and the leaves are the clusters with only one sample:

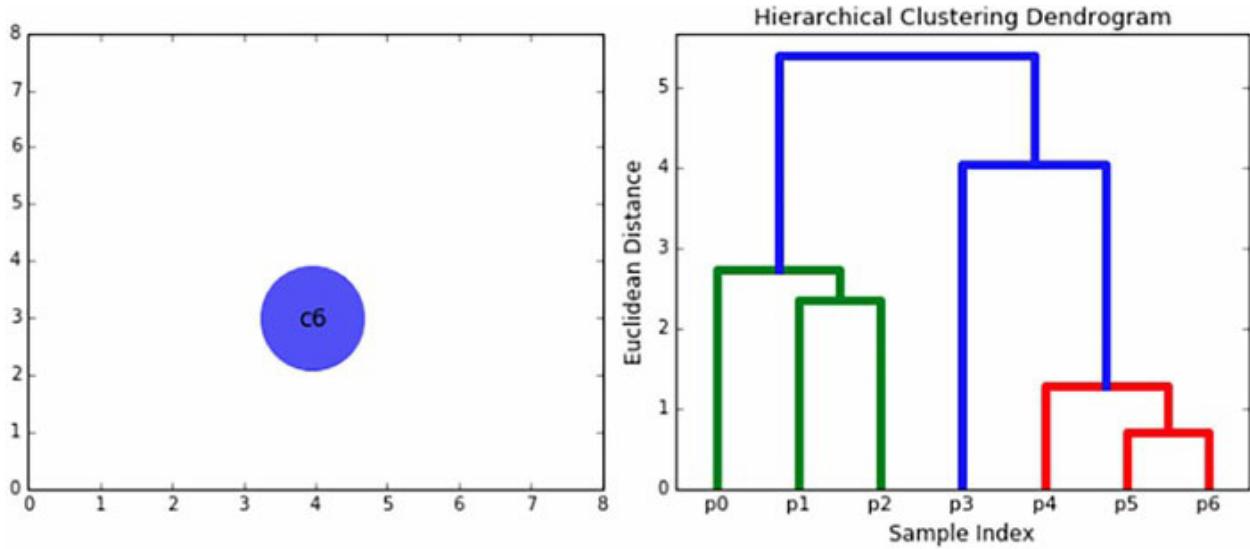


Figure 3.30: Hierarchical clustering dendrogram

1. We begin by treating each data point as a single cluster, that is, if there are X data points in our dataset, then we have X clusters. We then select a distance metric that measures the distance between two clusters. As an example, we will use *average linkage*, which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.
2. On each iteration, we combine two clusters into one. The two clusters to be combined are selected as those with the smallest average linkage, that is, according to our selected distance metric, these two clusters are the closest and, therefore, are the most similar and should be combined.
3. Step 2 is repeated until we reach the root of the tree, that is, until we only have one cluster that contains all data points. This way, we can select how many clusters we want in the end, simply by choosing when to stop combining the clusters, that is, when to stop building the tree!

Hierarchical clustering does not require us to specify the number of clusters, and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric; all of them tend to work equally well. On the other hand, with other clustering algorithms, the choice of distance metric is critical. A particularly good use case of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy; other clustering algorithms can't do this. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of $O(n^3)$, unlike the linear complexity of K-Means and GMM.

5. Describe in detail the python Libraries used for ML.

Ans. The most commonly used libraries in data science are numpy, pandas, matplotlib, and scikit learn.

Numpy

NumPy's main object is the homogeneous multidimensional array.

Its array class is called **ndarray**, and it is also known by the **aliasarray**. Note that **numpy.arrays** not the same as the Standard Python Library class **array.array**, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an **ndarray** object are as follows:

- **ndarray.ndim**

This is the number of axes (dimensions) of the array.

- **ndarray.shape**

This is the dimensions of the array. It is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, the shape will be (n,m). The length of the shape tuple is, therefore, the number of axes, n dim.

- **ndarray.size**

This is the total number of elements of the array. It is equal to the product of the elements of shape.

- **ndarray.dtype**

This is an object describing the type of elements in the array. One

can create or specify `d` type using standard Python types. Additionally, NumPy provides types of its own like `numpy.int32`, `numpy.int16`, and `numpy.float64` are some examples.

- **ndarray.itemsize**

This is the size of each element of the array in bytes. For example, an array of elements of type `float64` has `itemsize` 8 (=64/8), while one of type `complex32` has `itemsize` 4 (=32/8). It is equivalent to `ndarray.dtype.itemsize`.

- **ndarray.data**

This is the buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities. The sample code here explains the usage of array with its attributes.

```
>>>import numpy as np
>>>a=np.arange(15).reshape(3,5)
>>>a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>>a.shape
(3, 5)
>>>a.ndim
2
>>>a.dtype.name
'int64'
>>>a.itemsize
8
>>>a.size
15
>>>type(a)
<class 'numpy.ndarray'>
>>>b=np.array([6,7,8])
>>>b
array([6, 7, 8])
>>>type(b)
<class 'numpy.ndarray'>
```

Following example shows the array creation with various data types like integer, float etc.

```
>>>import numpy as np
>>>a=np.array([2,3,4])
>>>a
array([2, 3, 4])
>>>a.dtype
dtype('int64')
>>>b=np.array([1.2,3.5,5.1])
```

```
>>>b.dtype  
dtype('float64')
```

Basic Operations

Arithmetic operators on arrays apply element wise. A new array is created and filled with the result.

```
>>>a=np.array([20,30,40,50])  
>>>b=np.arange(4)  
>>>b  
array([0, 1, 2, 3])  
>>>c=a-b  
>>>c  
array([20, 29, 38, 47])  
>>>b**2  
array([0, 1, 4, 9])  
>>>10*np.sin(a)  
array([-9.12945251, -9.88031624, -7.4511316, -2.62374854])  
>>>a<35  
array([ True,  True, False, False])
```

Indexing, Slicing and Iterating

One-dimensional arrays can be indexed, sliced and iterated over, much like lists and other Python sequences.

```
>>>a=np.arange(10)**3  
>>>a  
array([ 0, 1, 8, 27, 64, 125, 216, 343, 512, 729])  
>>>a[2]  
8  
>>>a[2:5]  
array([ 8, 27, 64])  
>>># equivalent to a[0:6:2] = 1000;  
>>># from start to position 6, exclusive, set every 2nd element to 1000  
>>>a[:6:2]=1000  
>>>a  
array([1000, 1, 1000, 27, 1000, 125, 216, 343, 512, 729])  
>>>a[::-1]  
array([ 729, 512, 343, 216, 125, 1000, 27, 1000, 1, 1000])  
>>>for i in a:  
... print(i**(1/3.))  
...  
9.99999999999998  
1.0  
9.99999999999998  
3.0  
9.99999999999998  
4.99999999999999  
5.99999999999999  
6.99999999999999  
7.99999999999999  
8.99999999999998
```

Introduction to Pandas: Pandas provides high-performance, easy-to-use data structures and analysis tools for the Python programming language. Open-source Python library provides high-performance. Data manipulation and analysis tool and is also utilized due to its powerful data structures. The name **pandas** is derived from the word Panel Data, an econometrics term for multidimensional data.

Attributes of data

DataFrame.index

To get the index (row labels) of the dataframe

DataFrame.columns

To get the column labels of the dataframe

DataFrame.size

To get the total number of elements from the dataframe

DataFrame.ndim

The number of axes/array dimensions

Indexing and selecting data

DataFrame.head([n])

The function head returns the first n rows from the dataframe.

To access a scalar value, the fastest way is to use the **at** and **iat** methods:

- **at** provides label-based scalar lookups
- **iat** provides integer-based lookups

Numeric types

Pandas and base Python use different names for data types.

Python data type or Pandas data type int64 as well float64, describes data type and storage requirements as:

- ‘64’ simply refers to the memory allocated to store data in each cell, which effectively relates to how many digits it can store in each “cell”
- 64 bits is equivalent to 8 bytes
- Allocating space ahead of time allows computers to optimize storage and processing efficiency

Checking data types of each column

dtypes returns a series with the data type of each column

Syntax: **DataFrame.dtypes**

Count of unique data types

get_dtype_counts() returns counts of unique data types in the dataframe

Selecting data based on data types

pandas.DataFrame.select_dtypes() returns a subset of the columns from dataframe based on the column dtypes

Syntax: **DataFrame.select_dtypes(include=None, exclude=None)**

Concise summary of dataframe

info() returns a concise summary of a dataframe.

unique() is used to find the unique elements of a column
Syntax:
numpy.unique(array)

replace() is used to replace a value with the desired value

– Syntax: **DataFrame.replace([to_replace, value, &])**

To check the count of missing value present in each column

Dataframe.isnull.sum() is used.

Matplotlib: **Matplotlib** is a 2D plotting library that produces good-quality figures.

- Although it has its origins in emulating the MATLAB graphics commands, it is independent of MATLAB.
- It makes heavy use of NumPy and other extension code to provide good performance, even for large arrays.

Scatter Plot: A scatter plot is a set of points that represent the values obtained for two different variables plotted on a horizontal and vertical axis.

When should you use scatter plots?

- Scatter plots are used to convey the relationship between two numerical variables.
- Scatter plots are sometimes called correlation plots because they show how two variables are correlated.

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

```
plt.scatter(x,y,c='red')
plt.title("plot")
plt.xlabel("x_value")
plt.ylabel("y_value")
plt.show()
```

Histogram: It is a graphical representation of data using bars of different heights.

A histogram groups numbers into ranges, and the height of each bar depicts the frequency of each range or bin.

When should you use histograms?

To represent the frequency distribution of numerical variables as shown below:

```
plt.hist(df['x'])
plt.title("plot")
plt.xlabel("x_value")
plt.ylabel("y_value")
plt.show()
```

Bar plot: A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the counts that they represent.

When should you use a bar plot?

- It is used to represent the frequency distribution of categorical variables.
- A bar diagram makes it easy to compare sets of data between different groups.

```
plt.bar(x,y)
plt.title("plot")
plt.xlabel("x_value")
plt.ylabel("y_value")
plt.show()
```

6. How can visualization be effective with matplotlib?

Ans. Data visualization is an important part of business activities as organizations nowadays collect huge amounts of data.

Matplotlib

Matplotlib is a 2-D plotting library that helps in visualizing figures. Matplotlib emulates MATLAB like graphs and visualizations. MATLAB is not free, is difficult to scale and is tedious as a

programming language. So, matplotlib in Python is used as it is a robust, free and easy library for data visualization.

Installing Matplotlib

Type `!pip install matplotlib` in the Jupyter Notebook, or if it doesn't work then in `cmd (that is command prompt)` type `conda install -c conda-forge matplotlib`. This should work in most cases.

Things to follow

Plotting of Matplotlib is quite easy. Generally, while plotting follow the same steps in every plot. Matplotlib has a module called `pyplot`, which aids in plotting figure. The Jupyter notebook is used for running the plots. We import `matplotlib.pyplot as plt` for making it call the package module.

- Importing the required libraries and dataset to plot using Pandas `pd.read_csv()`
- Extracting important parts for plots using conditions on Pandas Dataframes
- `plt.plot()` for plotting line chart. Similarly in place of `plot` other functions are used for plotting; all plotting functions require data, and it is provided in the function through parameters
- `plt.xlabel` and `plt.ylabel` for labeling x-axis and y-axis respectively
- `plt.xticks` and `plt.yticks` for labeling x-axis and y-axis observation tick points, respectively
- `plt.legend()` for signifying the observation variables
- `plt.title()` for setting the title of the plot
- `plt.show()` for displaying the plot

7. Explain the various metrics used to evaluate ML algorithms.

Ans. When we talk about predictive models, we are talking either about a regression model (continuous output) or a classification model (nominal or binary output). The evaluation metrics used in each of these models are different.

In classification problems, we use two types of algorithms (based on the kind of output they create):

1. **Class output:** Algorithms like SVM and KNN create a class output. For instance, in a binary classification problem, the outputs will be either 0 or 1. Today we have algorithms that can convert these class outputs to probability, but these algorithms are not well accepted by the statistics community.
2. **Probability output:** Algorithms like Logistic Regression, Random Forest, Gradient Boosting and Ada boost give probability outputs. Converting probability outputs to class output is just a matter of creating a threshold probability.

In regression problems, we do not have such inconsistencies in output. The output is always continuous in nature and requires no further treatment.

Confusion Matrix

A confusion matrix is an $N * N$ matrix, where N is the number of classes being predicted. For the problem at hand, we have $N=2$, and hence, we get a 2×2 matrix. Here are a few definitions you need to remember for a confusion matrix:

- **Accuracy:** The proportion of the total number of predictions that were correct
- **Positive Predictive Value or Precision:** The proportion of positive cases that were correctly identified
- **Negative Predictive Value:** The proportion of negative cases that were correctly identified
- **Sensitivity or Recall:** The proportion of actual positive cases that are correctly identified.
- **Specificity:** The proportion of actual negative cases that are correctly identified.

Confusion Matrix		Target			
		Positive	Negative	Positive Predictive Value	a/(a+b)
Model	Positive	a	b	Negative Predictive Value	d/(c+d)
	Negative	c	d		
		Sensitivity	Specificity	Accuracy = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Count of ID	Target				
Model	1	0	Grand Total		
1	3,834	639	4,473	85.7%	
0	16	951	967	1.7%	
Grand Total	3,850	1,590	5,440	99.6% 40.19%	88.0%

Figure 3.31: Confusion matrix

The accuracy for the problem at hand comes out to be 88%. As you can see from the preceding two tables, the positive predictive value is high, but the negative predictive value is quite low. The same holds for sensitivity and specificity. This is primarily driven by the threshold value we have chosen. If we decrease our threshold value, the two pairs of starkly different numbers will come closer.

In general, we are concerned with one of the previously defined metrics. For instance, in a pharmaceutical company, they will be more concerned with minimal wrong positive diagnosis. Hence, they will be more concerned about high specificity. On the other hand, an attrition model will be more concerned with sensitivity. A confusion matrix is generally used only with class output models.

F1 Score

In the previous section, we discussed precision and recall for classification problems and also highlighted the importance of choosing precision/recall based on our use case. What if for a use case, we are trying to get the best precision and recall at the same time? F1-Score is the harmonic mean of precision and recall values for a classification problem. The formula for F1-Score is as follows:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Now, an obvious question that comes to mind is, ‘Why are we taking a harmonic mean and not an arithmetic mean?’. This is because HM punishes extreme values more. Let us understand this with an example. We have a binary classification model with the following results:

Precision: 0, Recall: 1

Here, if we take the arithmetic mean, we get 0.5. It is clear that the preceding result comes from a dumb classifier that just ignores the input and predicts one of the classes as output. Now, if we were to take HM, we will get 0, which is accurate as this model is not useful at all.

This seems simple. There are situations, however, for which a data scientist would like to give a percentage more importance/weight to either precision or recall. Altering the previous expression a bit to include an adjustable parameter beta for this purpose, we get the following:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

Fbeta measures the effectiveness of a model with respect to a user who attaches β times as much importance to recall as precision.

Root Mean Squared Error (RMSE)

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that errors are unbiased and follow a normal distribution. Here are the key points to consider on RMSE:

1. The power of ‘square root’ empowers this metric to show large number of deviations.
2. The ‘squared’ nature of this metric helps deliver more robust results, which prevents cancelling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of error term.

3. It avoids the use of absolute error values, which is highly undesirable in mathematical calculations.
4. When we have more samples, reconstructing the error distribution using RMSE is considered more reliable.
5. RMSE is highly affected by outlier values. Hence, make sure you've removed outliers from your data set before using this metric.
6. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors.

RMSE metric is given by the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

Here, N is the total number of observations.

R-Squared/Adjusted R-Squared

We learned that when the RMSE decreases, the model's performance improves. However, these values alone are not intuitive.

In the case of a classification problem, if the model has an accuracy of 0.8, we can gauge how good our model is against a random model that has an accuracy of 0.5. So the random model can be treated as a benchmark. But when we talk about the RMSE metrics, we do not have a benchmark to compare.

This is where we can use R-Squared metric. The formula for R-Squared is as follows:

$$R^2 = 1 - \frac{MSE(\text{model})}{MSE(\text{baseline})}$$

$$\frac{\text{MSE(model)}}{\text{MSE(baseline)}} = \frac{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}{\frac{1}{N} \sum_{i=1}^N (\bar{y}_i - \hat{y}_i)^2}$$

MSE(model): Mean squared error of the predictions against the actual values

MSE(baseline): Mean squared error of mean prediction against the actual values

In other words, how good our regression model as compared to a very simple model that just predicts the mean value of target from the train set as predictions.

Adjusted R-Squared

A model performing equal to baseline would give R-Squared as 0. The better the model, the higher the r2 value. The best model with all correct predictions would give R-Squared as 1. However, on adding new features to the model, the R-Squared value either increases or remains the same. R-Squared for adding features that add no value to the model. So, an improved version of the R-Squared is the adjusted R-Squared. The formula for adjusted R-Squared is given by the following:

$$\bar{R}^2 = 1 - \left(1 - R^2\right) \left[\frac{n-1}{n-(k+1)} \right]$$

Here,

k: Number of features

n: Number of samples

As you can see, this metric takes the number of features into account. When we add more features, the term in the denominator n-(k +1) decreases, so the whole expression increases.

If R-Squared does not increase, it means the feature added isn't valuable

for our model. So overall, we subtract a greater value from 1, and the adjusted r², in turn, would decrease.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 4

Case Studies and Projects in Machine Learning

Introduction

Chapter 1, Case studies and projects with Introduction to Machine Learning, and *Chapter 2, Python Basics for ML*, dealt with the concepts of machine learning and Python, while *Chapter 3, Overview of ML Algorithms* dealt with the various algorithms that are utilized while developing machine learning applications. This chapter aims to provide insights into the various case studies that use the concepts and algorithms of machine learning. It further introduces recommendation systems, their needs, the process of generating the recommendation systems and the current systems incorporating recommendation systems. This is followed by the case study of applying the various machine learning algorithms on text mining applications, its building blocks and case studies on opinion mining and sentiment analysis.

Moving on, the chapter elaborates on image processing applications and the machine learning algorithms employed in these applications. To further understand the core concepts and the deployment of machine learning algorithms, a number of case studies comprising of predictive analysis, social media analytics, customer churning analytics and analytics in the education system has been provided.

Structure

This chapter comprises of:

- Recommendation Generation
 - Importance of Recommendation Systems
 - Approaches for building recommendation systems
 - Recommendation Generation

- Evaluation metrics for recommendation algorithms
 - Case study on Recommendation system
- **Text Analysis and Mining**
 - Importance of Text Mining
 - Basic Blocks of Text Mining System using ML
 - Steps involved in preparing an unstructured text document for deeper analysis
 - Sentiment Analysis
 - Case Study on product recommendation system based on sentiment analysis
 - Opinion Mining
 - Real-Time Text Mining Systems overview
- **Image Processing**
 - Importance of Image Processing
 - Basic Image Processing System
 - Image Processing using popular ML algorithms
 - Real-time case studies of Image Processing using ML
- **Predictive Analytics**
 - Importance of Predictive Analytics
 - Need for Predictive analysis
 - Machine Learning Vs. Predictive analytics
 - Basic Working of Predictive Analytics System
 - Types of predictive analytics models
 - Uses of predictive analysis
 - Benefits of predictive analysis
- **Case studies**
 - Social Media Analytics
 - Customer churning analytics
 - Learning Analytics in Education System

Objectives

In this chapter, we will describe the purpose of recommendation systems. We will also understand the components of a recommendation system including candidate generation, scoring and re-ranking. We will elaborate on the use of machine learning algorithms in text mining and specify the importance of image processing and its applications. You will understand the concepts of predictive analytics and identify the various predictive analytics models. Additionally, we will elaborate on the various case studies employing machine learning algorithms.

Recommendation Generation

Recommendations are the suggestions given to users with respect to items bought frequently or the most popular items.

Importance of recommendation systems

There is explosive growth in the amount of digital content on the internet. Many keep accessing, updating and adding information on the internet. All this leads to the problem of information overload, which, in turn, hinders timely access to items of interest.

Apart from this, there has been a huge increase in the number of choices and timely access to the dynamic generation of information.

There is a need to have a system that segregates, prioritizes, personalizes and efficiently delivers relevant information to the user. One such system is the recommender system.

Recommender systems search a huge amount of dynamically generated information and data usage history. These systems estimate and predict user content preference and provide users with the content and services of their choice and interests. Thus, these systems filter out vital information and predict or recommend an item to the user based on their preference. Thus, recommender systems are an important class of machine learning algorithms that offer “relevant” suggestions to users.

Examples of companies using recommendation systems include Amazon, Facebook, LinkedIn, YouTube and Netflix. Recommendation systems provide customer satisfaction, save user browsing time and increase the

consumption time at the website. All this results in revenue generation for the organizations handling those websites.

Key terms used

Let's look at the key terms associated with recommender systems.

Items/Documents

These represent the entities recommended by the recommender system.

Query/Context

The recommender system determines relevant information to recommend items to the users. This information forms the query that can be a combination of user Information, such as user ID or items the user previously interacted with along with the user's device and their location as additional context.

Embedding: Embeddings represent categorical features as continuous-valued features. In other words, they translate a high-dimensional vector into a low-dimensional space known as an embedding space. In this case, queries or items to recommend have to be mapped to the embedding space.

Two commonly used recommendations are given here:

- **Home page recommendations:** In this, personalized home pages are provided based on users' interest. These are often used as content-based recommendations.
- **Related item recommendations:** These recommendations are based on determining related or similar items. This works on the fundamentals of collaborative filtering.

Approaches for building recommendation systems

Recommender systems are categorized as either content-based system or collaborative filtering.

Figure 4.1 represents the various techniques used for building accurate and efficient recommendation systems:

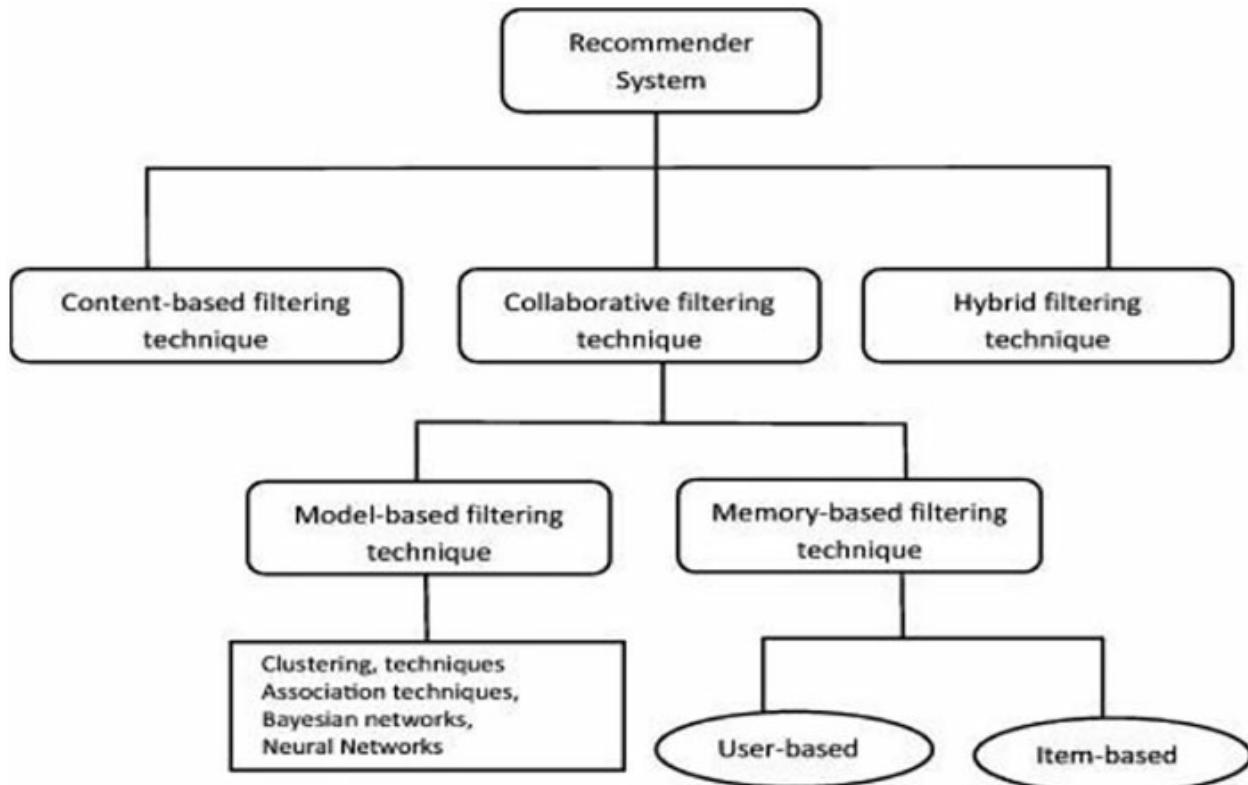


Figure 4.1: Techniques used for Recommendation systems

Various approaches include the usage of the following.

Content-based filtering: In this method, the predictions are based on the information provided by the users/users' profiles, while the choices or similarity of choice of other users are ignored. This information is obtained using the features extracted from the content of the items the user has evaluated in the past. Thus, it can be said that this method matches the user's characteristics. Meaningful recommendations are generated by applying various models that could include the following:

- Vector space models like **Term Frequency Inverse Document Frequency(TF/IDF)**
- Probabilistic models like Naïve Bayes Classifier, Decision Trees or Neural Networks to model the relationship between different documents within a corpus

Collaborative filtering: It is the most commonly used filtering method, and it recommends items based on the identification of the products by similar choices of other users. It uses their opinion to recommend items to the active

user.

It includes building a database in terms of a user-item matrix of preferences by the users. It further matches users with relevant interests and preferences. These are done by calculating similarities, called neighborhoods, between their profiles to make predictions or recommendations. Predictions are represented by a numerical value, R_{ij} , expressing the predicted score of item j for user i while recommendations indicate a list of top N items that the user would prefer the most.

Collaborative filtering is further classified into two categories:

- **Memory-based:** In this, the items already rated by the user play a relevant role in searching for a neighbor that shares similar preferences.
- **Model-based:** This technique analyzes the user-item matrix to identify relations between items; they use these relations to compare the list of top- N recommendations and employ the previous ratings to learn a model.

For example, Amazon generates a table of similar items offline using an item-to-item matrix and then recommends other similar products based on users' purchase history.

Hybrid filtering: In order to increase the accuracy and performance of the recommender system, hybrid filtering is used. It combines two or more filtering techniques. Based on their operations, hybrid systems are classified into weighted hybrid, mixed hybrid, switching hybrid, feature-combination hybrid, cascade hybrid, feature-augmented hybrid and meta-level hybrid.

Applying the concept of machine learning, hybrid approaches for recommender systems can be implemented by utilizing the following concepts:

- Content-based and collaborative-based predictions separately
- Combining the content-based capabilities and collaborative-based approach, or by unifying the approaches into a single model

Combining the content-based capabilities and collaborative-based approach can aid in overcoming the cold start, the sparsity problem and the knowledge engineering bottleneck in knowledge-based approaches. However, out of the various recommendation approaches stated, content-based recommendation

results in an accurate recommendation.

Basic recommendation systems

The architecture for recommendation systems broadly consists of the candidate generation, scoring and re-ranking components.

Candidate Generation

At this stage, from the huge corpus of data available, the system generates a smaller subset of candidates. The aim is that given a query, the system generates a set of relevant candidates.

Two common candidate generation approaches are listed in [*Table 4.1:*](#)

Type	Example
Content based filtering	If user A watches two cute cat videos, then the system can recommend cute animal videos to that user.
Collaborative filtering	If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn't seen any videos similar to video 1).

Table 4.1: Candidate generation approaches

Both content-based and collaborative filtering map each item and query (or context).

These embeddings are used for candidate generation as follows: Given a query embedding $q \in E$, the system looks for item embeddings $x \in E$ that are close to q , that is, embeddings with high similarity $s(q, x)$. To determine the degree of similarity, the recommendation systems rely on cosine, dot product or Euclidean distance measures, where:

- Cosine is the cosine of the angle between the two vectors, $s(q, x) = \cos(q, x)$
- The dot product of two vectors is given by $s(q, x) = \|x\| \|q\| \cos(q, x)$
- Euclidean distance is given as $(q, x) = \|q - x\| = \sqrt{\sum_{i=1}^d q_i - x_i}$ smaller distance means higher similarity

Scoring

Another model then scores and ranks the candidates so that the most relevant information is in the top10 items displayed on the screen.

Re-ranking

Re-ranking is required so that the user's likes/dislikes, and other preferences and constraints are considered.

Recommendation generation

To perform the recommendation, the system undergoes several phases:

Information collection phase

In this phase, information such as the behaviour and the resources the user accesses is collected. Based on which, a user's profile or model is created. This acts as the input to the system. Thus, the user's preference data is collected in two ways:

- **Explicit data:** Here, the users are directly asked whether they like a particular item. Based on this data, the profile is built as per the user's interest. However, the drawback of this type of collection is that not every user gives feedback or rating; even if they do so, it may mean different to different people.
- **Implicit Data:** In this, the information is derived from the users' interaction with the site. This interaction, by observing user behavior, is interpreted as an indication of the users' interest or disinterest.

Learning phase

In this phase, learning algorithms are applied on the data captured in the information collection phase.

Prediction/recommendation phase

It recommends or predicts the items the user may prefer. [Figure 4.2](#) highlights the recommendation phases:

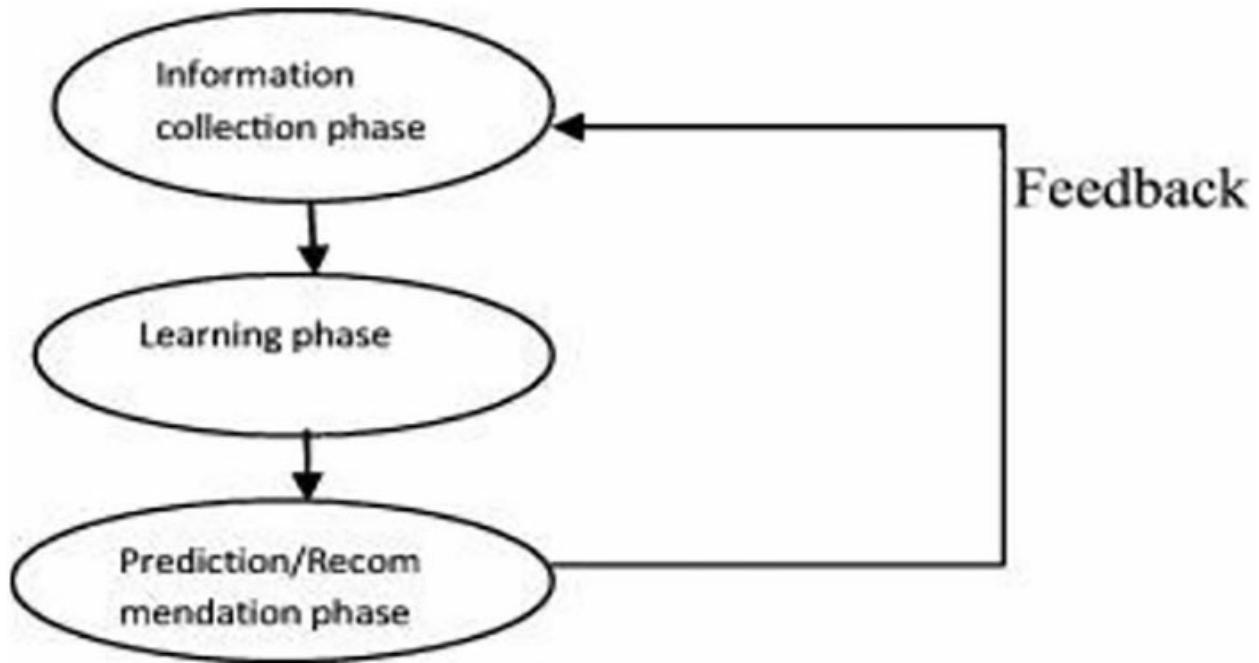


Figure 4.2: Recommendation phases

Evaluation metrics for recommendation algorithms

Accuracy and coverage are mainly considered as the evaluation metrics for recommending users/items based on the type of filtering technique.

Accuracy: *It is defined as the number of correct recommendations out of the total possible recommendations. Statistical and decision support accuracy metrics further determine the accuracy of recommendation systems.*

Statistical accuracy metric

It evaluates the accuracy of the technique used by comparing the predicted ratings directly with actual user ratings through the following:

Mean Absolute Error (MAE): It is determined as follows:

$$\sum_{i=1}^N u, i \mid P_{u,i} - r_{u,i}$$

Where P_{ui} is the predicted rating for user u on item i , r_{ui} is the actual ratio and N is the total number of ratings on the item set. The lower the MAE, the more accurate the recommendation. It is given by the following:

Decision support accuracy metrics: It helps the users in selecting the high-quality items out of the available set of items. These metrics include reversal rate, weighted errors, **Receivers Operating Characteristics (ROC)**, **Precision-Recall Curve (PRC)**, precision, recall and F -measure. Precision and recall are the most preferred evaluation metrics and are determined as follows:

Precision: It is represented as the fraction of recommended items that is actually relevant to the user.

$$Precision = P \text{ where } TP\text{-True Positive, } FP\text{-False Positive}$$

$$TP + FP$$

Recall: It is defined as *the fraction of relevant items that are also part of the set of recommended items*

$$\text{Recall} = R \text{ where } TP\text{-True Positive, } FN\text{-False Negative}$$

$$TP + FN$$

Coverage: *It measures the fraction of objects in the search space the system is able to provide recommendations or predictions for.*

Case study on Recommendation system: E-learning

system domain

In today's scenario, the education system has grown leaps and bounds; it has everything ranging from regular academics and remote to distance learning, on-your-own-pace learning, and certification courses. Therefore, it becomes a challenging task for any student (person) to identify the right type of course, the teaching and learning approach, and the benefits that could be derived from these courses, based on their preferences and requirements.

To overcome this issue, **Recommender Systems (RS)** are used to recommend the course or a list of similar courses that a student can undertake, based on their preferences.

Recommender systems

Many users undertake a number of courses to improve their skill sets. However, there is a lot of confusion and anxiety as to the type of course to be undertaken, the most efficient, in-demand course at a particular time or the most beneficial course for them. Identifying a suitable course for an individual can be tedious as it would involve accessing several platforms, searching for the available courses, selecting courses, reading each course syllabus carefully, and choosing appropriate content.

Problem definition

Gathering and consolidating information from multiple sources to make useful recommendations is a hard problem that many recommendation systems attempt to solve. There is a need for a recommender system to enhance the effectiveness of course choice.

Objective of the case study

The objective of this study is to do the following:

- Design and develop a hybrid Recommender system that can be integrated to enhance the effectiveness of any E-learning system.
- To ease information access and provide personalization to learners.

Need for the system: In order to provide correct, accurate and the best-suited course for an individual, a model needs to be developed. The inputs to the

system have to be based on the requirements of the users, courses and feedback of the correctly recommend courses.

Further, a number of statistical and sentimental analysis parameters will need to be considered.

A hybrid recommendation model is required based on features of users, courses and feedback to correctly recommend the courses. Various statistical and sentimental analysis related parameters are used to give a recommendation based on the existing (prior) dataset. These features could include information (features) about the various courses/professors, ratings of the topic along with the passing percentage and the subject structure to give the best recommendation.

Considerations for the case study

The focus for developing the e-learning recommender system is to guide the learner to select the courses as per their requirement. Therefore:

- a. Deep learning mechanisms need to be incorporated to extract meaningful features and provide learning through multiple levels of representation and abstraction.
- b. Ontology to retrieve useful information and make accurate recommendations.

Therefore, there is a need to apply deep learning and ontology mechanisms to get an accurate recommendation system, leading to what is called the hybrid model.

System development

To develop the recommender system, various steps need to be performed where the individual taking the course can be considered to be a student.

The recommendation system development can follow the **System Development Life Cycle (SDLC)** approach. SDLC is essential to ensure that the appropriate system is developed and meets the requirements in terms of features, usability, cost and time. This approach comprises of developing the system through various phases as represented in [Figure 4.3](#):

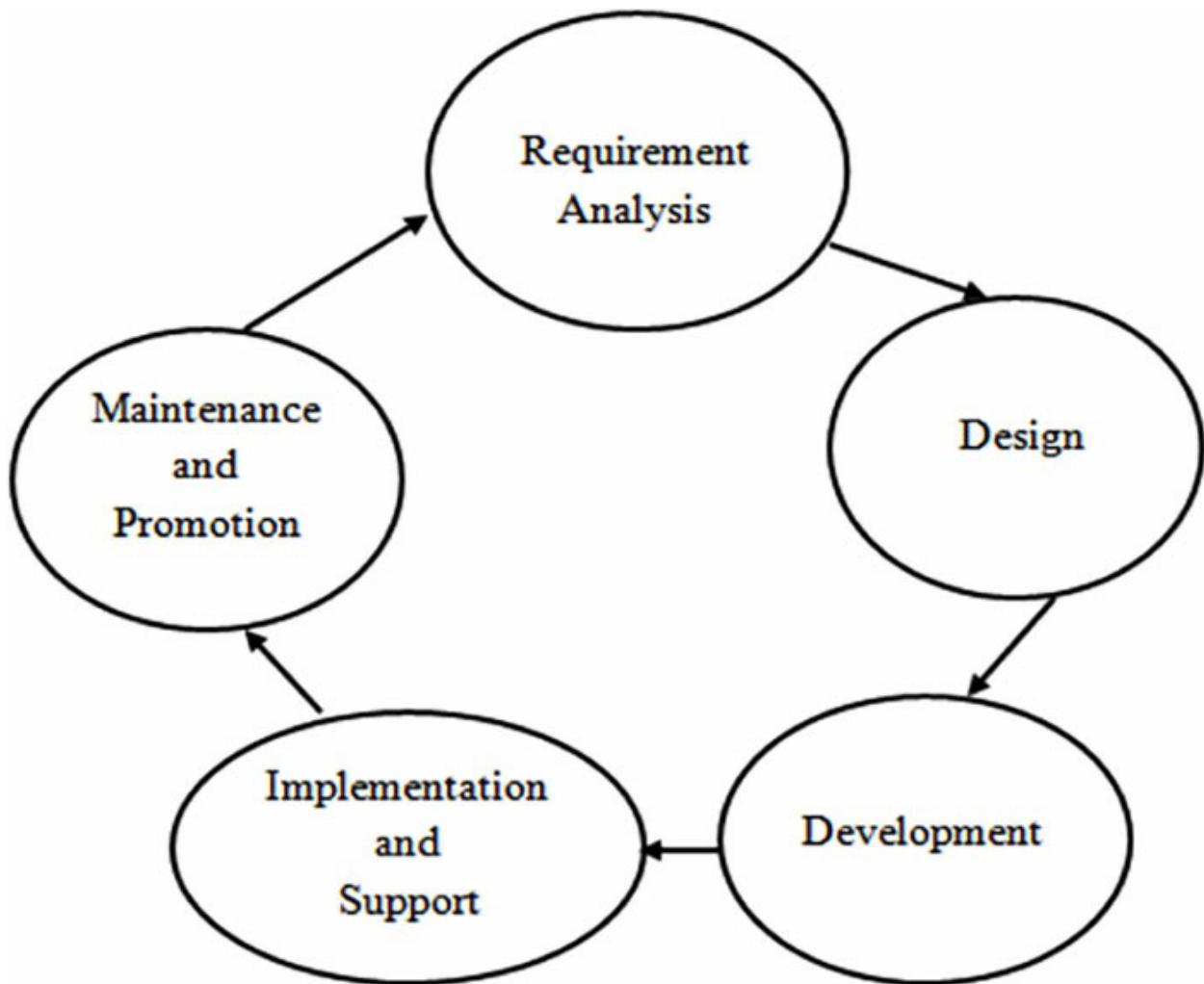


Figure 4.3: System Development Life Cycle (SDLC) approach

Requirement Gathering and Analysis Phase

The basic aim of this phase is to identify the goals, scope constraints, technical and non-technical requirements, risks involved and so on.

Based on the information gathered, analysis is done with respect to the current users of the system and the current system in action to identify new requirements (if any).

System Design

Block diagram and modular diagram-based approaches can be considered for the development of the e-learning recommender system. Its main purpose is to capture the dynamic aspect of the system including internal and external influences that need to be developed, as represented in [Figure 4.4:](#)

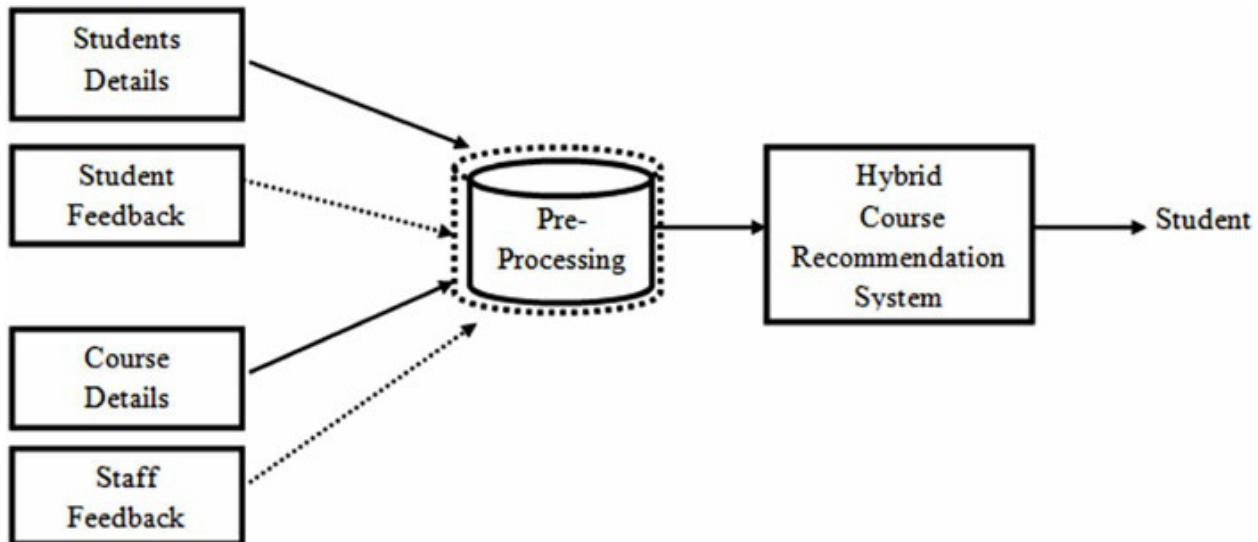


Figure 4.4: Block diagram of the system

As represented in the [Figure 4.4](#), the block representing student details can include the following:

- **Student Details:** Basic information of the student, like name, age, location, the core subject of the student, the subject of interest, email ID, education level, and contact details.
- **Course Details:** Information about the course, minimum qualification required to undertake the course, and graduation required to attend the course.
- **Other details:** Prerequisites for the course, duration of the course, reviews about the course, Passing criteria, course flexibility, fee structure and so on.
- **About the course:** Number of students taking admission for course in previous years, passing percentage age for previous years, and class size for the course.
- **Type of subject:** Analytical/theoretical, passing criteria-theory/practical, part-time/fulltime, financial aid – loan for student scholarship and practical training.
- **Feedback obtained:**
 - **Students feedback:** The feedback data that can be gathered could include the following:

- Courses completed by student
- Students selected for campus after successful completion of the course
- **Staff feedback:** It includes feedback from staff for the particular course, including the following:
 - Faculty details who conduct the course, year of experience, expertise, feedback by student
 - Qualification of faculty
 - Books and research papers published by faculty
- **Alumni feedback about the course**
- **Database/Dataset:** The dataset would comprise all of the data parameters and the factors that would enable accurate recommendations. The preceding information would aid in pre-processing the data that has been received from different entities. The pre-processed data can then be given as input to the proposed system where more than one algorithm has been applied and evaluated to recommend the course to the user.

Algorithms that can be employed

Various algorithms can be applied, including the following:

- **Gradient Descent (GD) or steepest descent algorithm**

It is an iterative first-order optimization algorithm that is used to find a local minimum/maximum of a given function. This is done to minimize a cost/loss function.

The performance of a model is measured through the cost function applied for any given data. It quantifies the error between predicted values and expected values and presents it in the form of a single real number. To achieve the goal of minimizing the cost function, it performs two steps iteratively:

At a point, the gradient (slope) is determined by calculating the first-order derivative of the function.

Take a step (move) in the direction opposite to the gradient. The opposite direction of the slope increases from the current point by alpha

times the gradient at that point. The alpha value indicates the learning rate: a tuning parameter in the optimization process that decides the length of the steps.

- **Evolutionary Algorithms (EA)**

These form a heuristic-based approach to solve problems that cannot be easily solved in polynomial time. They constitute a collection of methods that solve combinatorial optimization problems.

These algorithms mimic natural selection, where only the fittest individuals survive through the process of mutation, selection and crossover. The main classes of EA are as follows:

- **Genetic algorithms (GAs):** It provides the solution for optimization problems where each candidate solution has a set of properties (or genes) that can mutate or change until we find the best solution.
- **Evolution strategies (ESs):** These algorithms provide population-based solutions that are based on concepts of biological evolution. In this, a number of possible solutions is first determined. Each of these solutions is evaluated through a ‘fitness function’ that indicates how good they are. The solution population evolves over time and further determines better solutions.
- **Differential evolution (DE):** These algorithms handle nonlinear and non-differentiable multidimensional objective functions while requiring very few control parameters to go towards the minimization of cost functions.

Estimation of distribution algorithms (EDAs) or probabilistic model-building genetic algorithms (PMBGAs): They help in determining the best candidate solutions for a given problem. Optimization is done over a range of admissible and acceptable solutions to a model that generates only the global optima. The various algorithms expressed previously can be utilized for developing the recommender systems. They can further help to minimize the losses.

Tools utilization

It comprises of utilizing high-level programming language such as :

Python: It makes the code easy to read and its syntax helps programmers

express the concepts in fewer lines of code.

Scikit-learn: Python provides a number of libraries that provide implementations of a range of machine learning algorithms. Scikit-learn provides a number of supervised and unsupervised learning algorithms via a consistent interface in Python.

Deeplearning4j: Deeplearning4j is used as an open-source deep learning library for Java and the JVM. It includes multilayer perceptrons, **Convolutional Neural Networks (CNNs)** for image and text classification, and recurrent neural networks like LSTMs for text and time series data. Additionally, it includes machine learning algorithms, namely, logistic regression and k-nearest neighbors.

TensorFlow: It is a system for building and training neural networks to detect and decipher patterns and correlations.

IDE: Various platforms can be used to develop rich client applications and create integrated development environments. IDE platforms can include Eclipse, spyder, visual studio and so on.

The system can thus be implemented using machine learning and deep learning algorithms like neural networks using Python and their libraries. The dataset can be used to determine the degree of accuracy and precision of algorithms.

Constraints / limitations while developing the recommendation system

Following are the limitations while developing the recommendation systems:

- **Data sparsity**

Data sparsity is the term used to describe the phenomenon of not observing enough data in a dataset.

For example, as mentioned earlier, collaborative filtering methods recommend courses based on user's past preferences. Therefore, new users of the system will need to rate enough courses to enable the systems to capture their preferences accurately and provide reliable recommendations. Similarly, when new courses are added to the system, they need to be rated by a substantial number of users before they can be recommended to users.

Thus, identification of the right type of algorithm for recommending a course is very important.

- **Cold Start Problem**

The cold start problem occurs when the system is unable to form any relation between users and the course for which it has insufficient data. There are two types of cold-start problems:

- **User cold-start problems:** When there is almost no information available about the user
- **Product (course) cold-start problems:** When there is almost no information about the course asked for

The reasons due to which the system fails to give recommendations include the following:

- **Systematic Bootstrapping**

It represents the starting of the system when there is almost no information on which the recommender may rely.

- **Low Interaction (Few instances available)**

When new courses are added to the catalogue, the item cold-start problem occurs when there are none or very few interactions. This forms a problem for collaborative filtering algorithms as they generate recommendations based on the item's interactions. However, if a few interactions are available, a collaborative algorithm can recommend the course. However, the quality of those recommendations would be poor.

- **Entry of new users**

This problem occurs when a new user enrolls in the system and the recommender is required to offer recommendations for a set length of time without depending on the user's previous interactions, as none have been recorded yet. This issue is important as the user who would receive poor-quality recommendations may opt to leave the system before giving enough interaction to allow the recommender to understand their interests. Their preferences are obtained to create an initial user profile, especially when dealing with new users.

- **Lack of Data**

A lot of data is required to make effective recommendations.

Evaluation Metrics

Evaluation is important for assessing the effectiveness of recommendation algorithms. The commonly used metrics can include the following:

- Accuracy
- The mean squared error
- Root mean squared error

Information retrieval metrics that are useful to assess the quality of a recommendation method include the following:

- Precision
- Recall
- Diversity
- Novelty
- Coverage

Text Analysis and Mining

Text mining is defined as the non-trivial extraction of hidden, previously unknown and potentially useful information from a large amount of textual data. Text mining operates on unstructured or semi-structured data sets like emails, text documents and HTML files. Text mining (also referred to as *text analytics*) is an **Artificial Intelligence (AI)** technology that uses **Natural Language Processing (NLP)** to transform the free (unstructured) text in documents and databases into normalized, structured data suitable, identify meaningful patterns and new insights for analysis. Through use of advanced machine learning algorithms like Naïve Bayes, Support Vector Machines (SVM) and other deep learning algorithms, several organizations can explore and discover hidden relationships within their unstructured data.

The terms, text mining and text analytics are used interchangeably. Through the use of machine learning algorithms, statistics and linguistics, textual patterns and trends within the unstructured data can be identified. By utilizing text mining and text analysis, further quantitative insights can be obtained on the structured data.

Importance of text mining

Text mining, also known as text analytics systems, focuses on determining key knowledge from a corpus of texts, like its relevance to the designated subject, and representing the details of its content, such as any specific entities and its linkages. Text mining tools analyze documents to identify entities and extract relationships between them, unlocking hidden information to identify and develop new hypotheses. Thus, text mining can be used to make large quantities of unstructured data accessible. By extracting information, patterns, trends and insights hidden in large amounts of information can be made available. Text being one of the most common data types within databases, its data can be organized as follows:

- **Structured data:** This data consists of maintaining the information in standardized row and column format that is, a tabular format. This makes it easier to process, store and apply various machine learning algorithms on the available data. Structured data can include inputs like names, addresses, and phone numbers.
- **Unstructured data:** Unlike structured data, this data does not have a predefined data format. The data represented can include text information from several sources, such as text information from social media, product reviews or information from rich media formats like video and audio files.
- **Semi-structured data:** This is a combination of structured and unstructured data formats, where an organization might not have enough structure nor a format to meet the requirements of a database; examples include XML, JSON and HTML files.

As 80% of data in the world is present in an unstructured format, text mining forms an important practice in organizations.

Basic blocks of text mining system using ML

Text mining involves considering unstructured text data as input and processing it. This processing deals with handling parsing, along with the addition of some derived linguistic features, deriving patterns within the structured data final evaluation and interpretation of the output, as indicated in *Figure 4.5*:

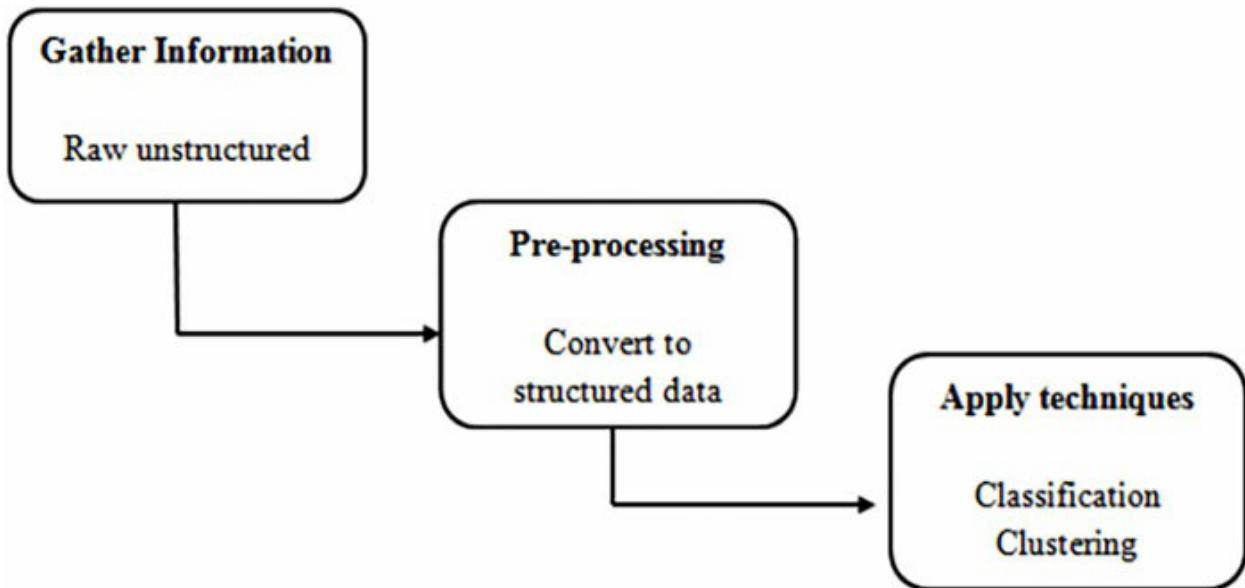


Figure 4.5: Basic Process of Text Mining

Text analysis involves information retrieval, information extraction, data mining techniques like association and link analysis, visualization, and predictive analytics. The goal is to turn text (unstructured data) into data (structured format) for analysis by utilizing **Natural Language Processing (NLP)** methods. The various activities performed to mine the text and determine the patterns are represented in the following [figure 4.6](#):

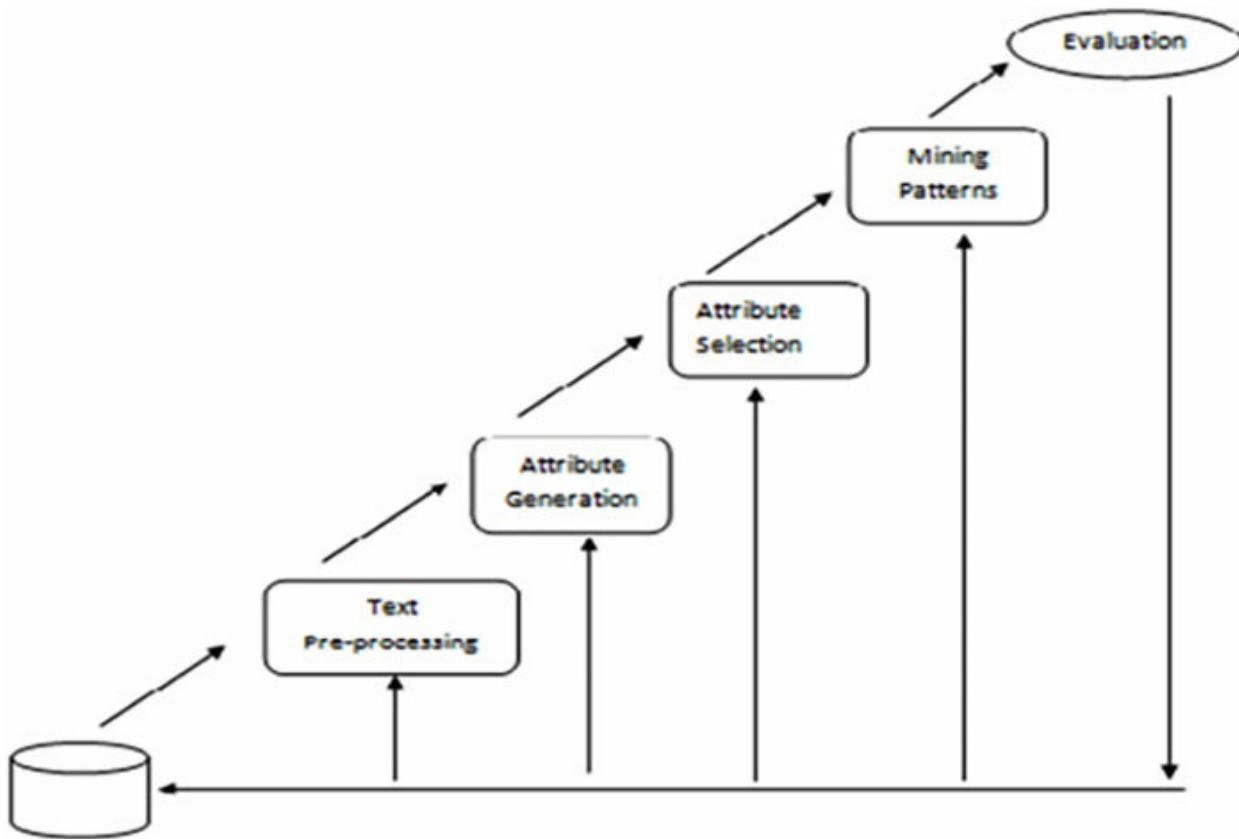


Figure 4.6: Activities performed for efficient mining of information

Before applying various text mining techniques, text preprocessing needs to be performed, as represented in [Figure 4.6](#).

Text preprocessing: This step includes cleaning and transforming text data into a usable format. Once text preprocessing is complete, based on the relevant attribute generation, a number of text mining algorithms can be applied to derive insights from the data.

Text transformation (attribute generation): A text document is represented by the words (features) it contains and their occurrences. Two main approaches of document representation are Bag of words and Vector Space.

Feature or attribute selection: Feature selection, also known as variable selection, is the process of selecting a subset of important features for use in model creation. The main assumption when using a feature selection technique is that the data contains many redundant or irrelevant features that provide no extra or relevant information in any context. Feature selection technique is a subset of the more general field of feature extraction.

Data mining: At this point, the text mining process merges with the

traditional data mining process. Classic data mining techniques are used in the structured database that resulted from the previous stages.

Evaluate: The result is evaluated at this stage. After evaluation, the result can be discarded, or the generated result can be used as an input for the next set of sequence.

Steps involved in preparing an unstructured text document for deeper analysis

There are seven basic steps involved in preparing an unstructured text document for deeper analysis:

1. **Language identification:** It helps determine the language of operation.
2. **Tokenization:** Most text analytics systems rely on rule-based algorithms to tokenize alphabetic languages. The textual information is broken down into pieces called tokens. These tokens can be words, phonemes, or full sentences. These can be obtained by splitting the text on white spaces and at punctuation marks that do not belong to abbreviations identified. The various characteristics and its representations are listed in [Table 4.2](#):

Characteristics	Represents
Feature Extraction	Entities, Themes, Categories, Intentions, Summaries, Sentiments
Text Preparation (Multi layered)	Tokenizing, Sentence breaking, PoS, Chunk, Syntax, Lexical chaining
Base Knowledge	Machine learning models, NLP, Training of words

Table 4.2: Text analytics

3. **Sentence breaking:** It is used to determine the boundaries that exist in a sentence.
4. **Part of Speech Tagging (PoS):** It is the process of determining the part of speech of every token, i.e., whether a given token represents a proper noun, common noun or verb, and then tagging it. **Part-of-Speech (POS)** tagging means word class assignment to each token. Its input is given by the tokenized text. Taggers have to cope with unknown and

ambiguous word-tag mappings. Rule-based approaches like ENGTWOL operate on dictionaries containing words formed with the associated POS labels, morphological and syntactic features. Context-sensitive rules help to choose the appropriate labels during application.

5. **Chunking:** It means assigning PoS-tagged tokens to phrases.
6. **Syntax Parsing:** This step determines the structure of a sentence. It is further considered to be the most computationally intensive step and is a preparatory step in sentiment analysis.
7. **Sentence Chaining or sentence relations:** It is used to draw associations between sentences, run complex analyses, such as comparing, contrasting sentiment scores, and generating accurate summaries of long documents.

Text mining techniques

Text mining techniques include techniques for information retrieval, natural language processing and sentiment analysis, as elaborated here:

Information Retrieval (IR)

IR systems work on applying algorithms to track user behaviors and identify relevant data. Thus, based on a pre-defined set of queries or phrases, this technique returns relevant information or documents. Information retrieval is commonly used in library catalog systems and Google search engines. Common IR sub-tasks include the following:

- **Text cleanup:** Text cleanup is a mechanism that removes any unnecessary or unwanted information, for example, removing advertisements from web pages, normalizing text converted from binary formats, and dealing with tables, figures and formulas.
- **Tokenization:** This deals with the process of splitting the long-form text into sentences and words called “tokens”.
- **Stemming:** This step involves separating the prefixes and suffixes from words to derive the root word form and meaning.

Natural language processing (NLP)

Natural language processing enables computers to understand human language in both written and verbal form. By analyzing sentence structure and grammar, NLP algorithms help the computers to read information. These sub-tasks include the following:

- **Summarization:** As the word suggests, this technique creates a concise summary of the specified document's main points.
- **Part-of-Speech (PoS) tagging:** This step enables semantic analysis on unstructured text. Based on its part of speech, this technique assigns a tag to every token in the document.
- **Text categorization or text classification:** Through this technique, text documents are classified based on predefined topics or categories.
- **Sentiment analysis:** This task is used to provide information about perceptions of brands, products, services and changes in customer attitudes over time. Therefore, it detects positive, negative or neutral sentiments about a product or brand, leading to insights that can improve business processes and user experiences.

Information extraction deals with extracting structured information from free text and storing the various entities, attributes and related information in the database. Common information extraction sub-tasks include the following:

- Feature selection, also known as attribute selection, is a process of selecting the important features (dimensions) that contribute the most to the output of a predictive analytics model.
- In order to improve the accuracy of a classification task, the sub-task i.e., feature extraction, is employed. Its aim is to select a subset of features.
- **Named-entity recognition (NER)** is also known as entity identification or entity extraction. Its major purpose is to find and categorize specific entities in the available text.
- **Data mining:** It is the process of identifying patterns and extracting useful insights from datasets. This technique evaluates structured as well as unstructured data to identify new information. Text mining is said to be a sub-field of data mining.

Sentiment analysis

Sentiment analysis is a machine learning tool that analyzes text for its polarity. The polarity may be positive, negative or neutral. Text analysis techniques are used to interpret and classify the emotions within text data, which, in turn, makes the machines automatically learn how to detect sentiment without any human input.

Sentiment analysis models can be trained to understand the context, sarcasm and misapplied words. The tools used in sentiment analysis help businesses to identify the sentiment of the customer towards a particular product, a brand or a service provided. Once the machines are trained through machine learning tools, they learn to detect the sentiments automatically without any human intervention. A number of techniques and complex algorithms can be used to command and train machines to perform sentiment analysis. Each technique has its own pros and cons, but when used together, they can provide exceptional results, as shown in [*Figure 4.7*](#):

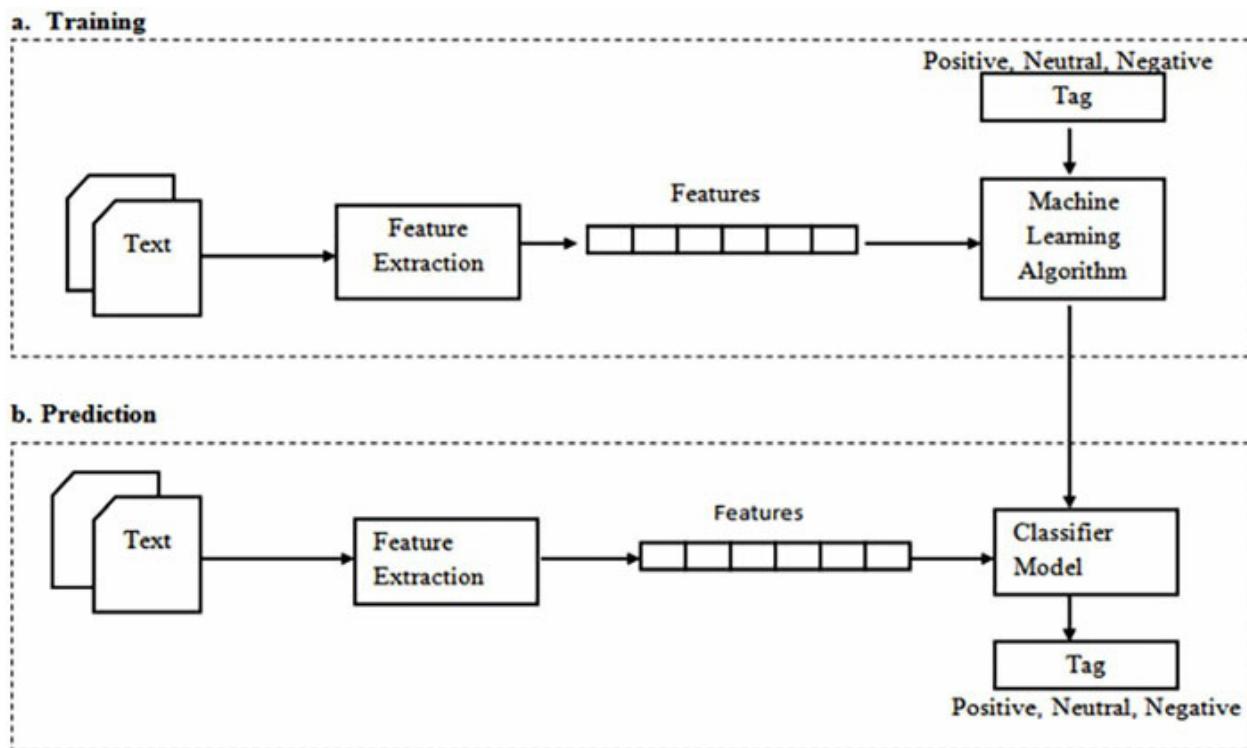


Figure 4.7: Process of Sentiment analysis

Now, let's look at the most frequently used algorithms for sentiment analysis.

Naive Bayes

Naive Bayes algorithms are considered to be one of the simplest groups of probabilistic algorithms. These algorithms help by performing sentiment analysis classification, in which a probability is assigned for the occurrence of the word or text. This assignment represents a polarity of positive or negative to the given word or phrase.

Bayes' theorem works as follows: Naive Bayes algorithm evaluates words against each other. Thus, the likelihood that a word, phrase, or text has occurred positive or negative, can be determined by the well-trained machine learning models. Utilizing techniques like lemmatization, stop word removal or TF-IDF, Naive Bayes provides better accuracy.

Linear regression

Given X features, linear regression, being a statistical algorithm, helps to predict the Y value. Machine learning algorithms work on the datasets provided. These algorithms further determine the relationship between the variables, which are represented through the X-Y axis. The straight-line pattern running across the various coordinates helps determine further predictions in relationships.

With respect to word polarity, linear regression determines the relationship between the words and phrases (X input) and the polarity (Y output). This relationship provides output on a scale of polarity from being “really positive” to “really negative”, along with the values between them.

Support Vector Machines (SVM)

A support vector machine is also a supervised machine learning model that uses its algorithms to train and classify text within the sentiment polarity model. As an example, mentioned in [Figure 4.8](#), consider the red and blue marks as tags that represent two data features: X and Y. The SVM classifier could then be trained to output the X-Y coordinates as either red or blue.

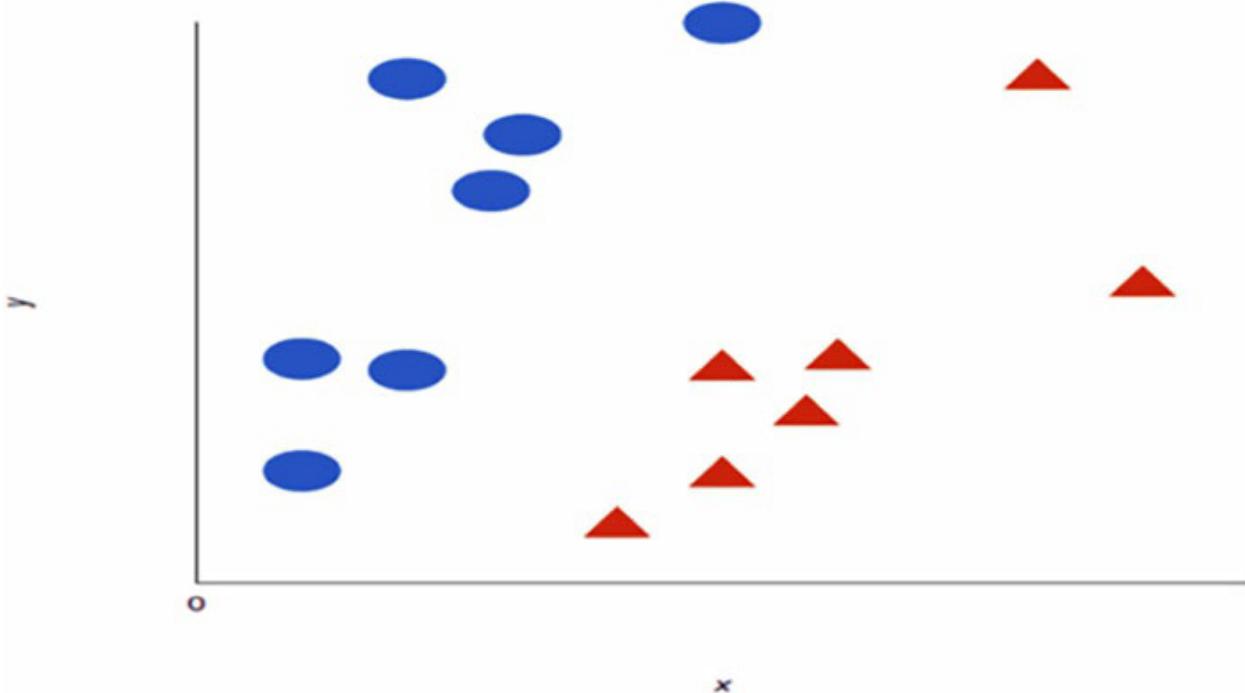


Figure 4.8: Representation of data features through tags

An SVM algorithm further assigns a hyperplane to the data features (X-Y) coordinates, as shown in [Figure 4.9](#). This hyperplane best separates the two tags. For a two-dimensional representation, this hyperplane represents a straight line, similar to a line in linear regression. One side of the hyperplane line represents the *red tag*, while the other side of the line specifies the *blue tag*. This representation works well for sentiment analysis, where one of the tags would represent the positive sentiment, while the other tag would indicate the negative sentiment.

The hyperplane that provides the largest distance between each tag is considered as the best hyperplane in machine learning. Refer to the following [figure 4.9](#):

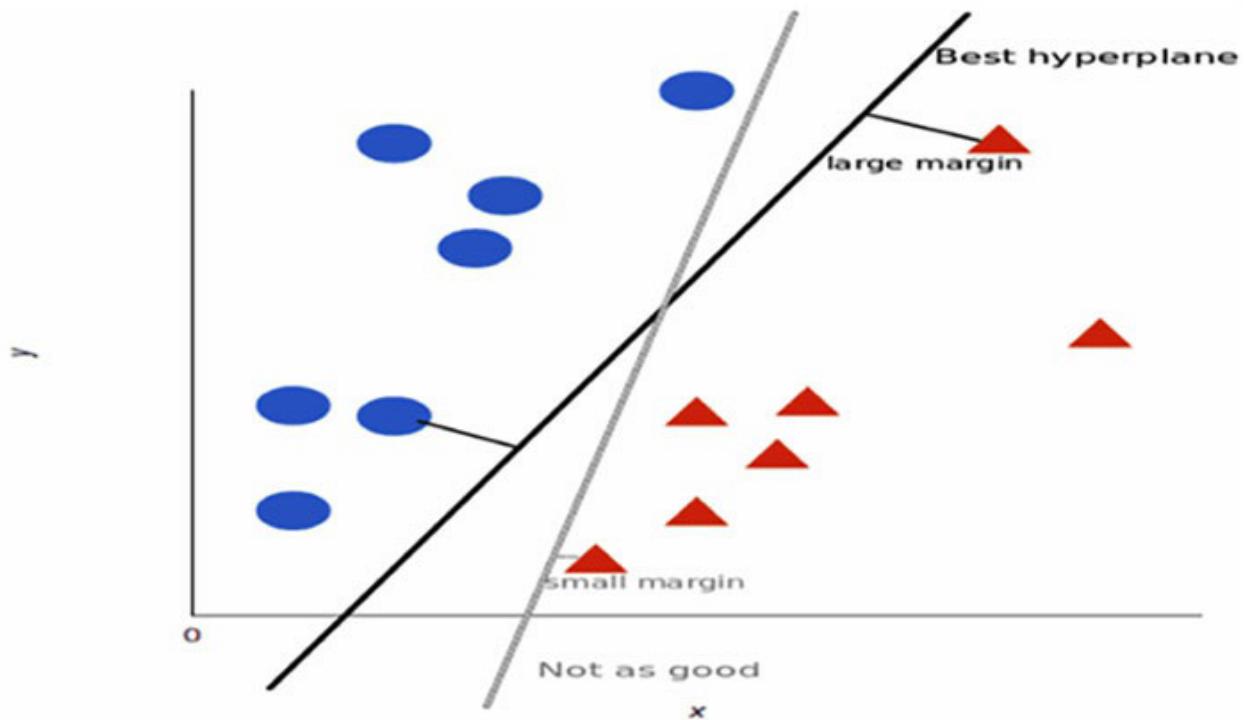


Figure 4.9: Representation of the hyperplane

However, as data sets become more complex, using a single line to separate the data features into two regions is difficult, as represented by [Figure 4.10](#):

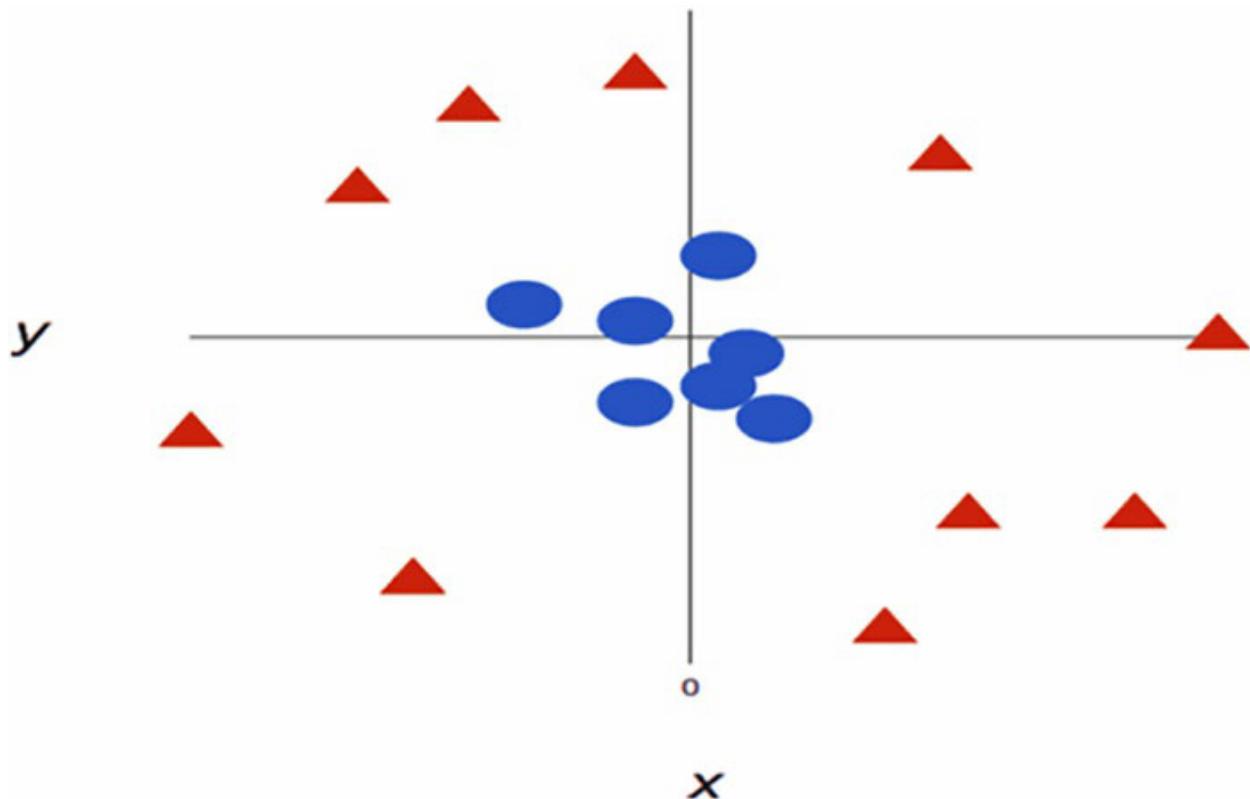


Figure 4.10: Complex data representation

Using SVM, more complex the data, more accurate the predictor will be the predictor will become. Imagine the preceding figure in three dimensions, i.e. with a Z axis added. This leads to make the figure represent a circle. This figure can be mapped to its two-dimensional representation. This would result in the best hyperplane, represented by [Figure 4.11](#):

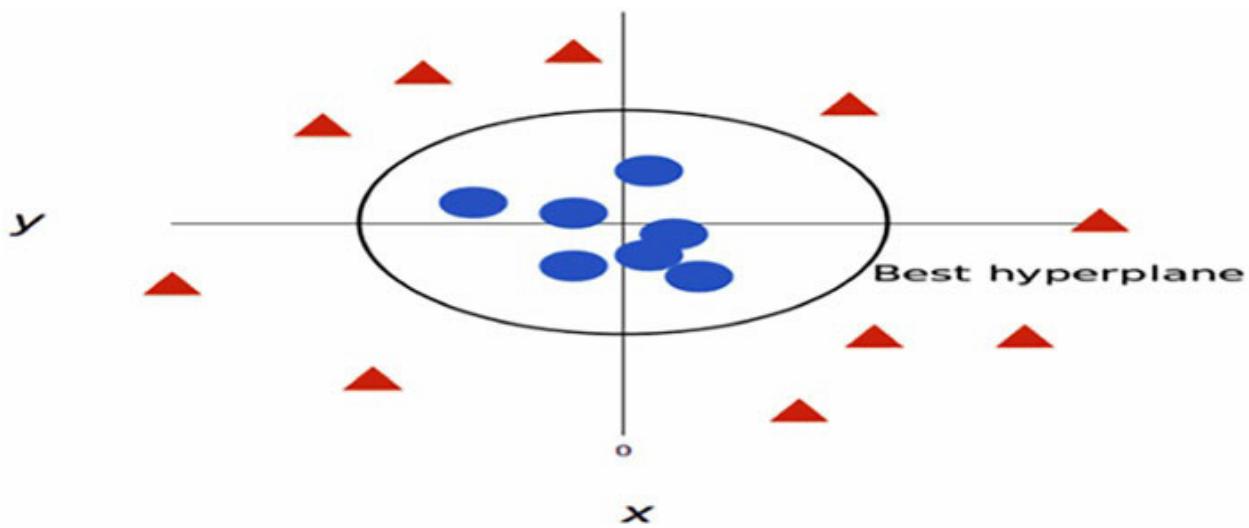


Figure 4.11: Hyperplane represented as a circle

SVM allows for more accurate machine learning because it is multidimensional.

Deep learning

As the name suggests, “deep” represents going into the depth of understanding, while “learning” indicates understanding. Thus, deep learning forms a subfield of machine learning that aims to determine and understand data the way the human brain does, using “artificial neural network.” Deep learning is considered to be a hierarchical machine learning representation. It is considered to be a multi-level learning that permits a machine to automatically chain a number of human-created processes together. By allowing multiple algorithms to be used progressively, deep learning can solve complex problems just like humans do.

Building a sentiment analysis model

Following are the steps to build a sentiment analysis model, assuming that data is available in the system:

1. Choose the model
2. Choose the classifier
3. Import the data
4. Train the sentiment analysis classifier
5. Test the classifier

Case study on product recommendation system based on sentiment analysis

For online shopping, the reviews and rating given by customers lead to further footfalls. However, the current online product recommendation systems that are based on reviews have randomness in their review pattern. This randomness can be due to some fake reviews that would make the buyer confused.

To overcome this issue, a system can be developed that can focus on the genuineness and trustworthiness of the reviews and can enhance the accuracy of the recommendation system.

Product recommendation

Recommender systems, in general, are used in various domains, such as movies, music, news, books, research articles, search queries, social tags, and products. E-commerce sites like eBay, Amazon, Flipkart, and Alibaba use proprietary recommendation algorithms so that they would be in a better position to serve their customers with the products they have. A product recommendation system predicts and shows the items that a user would like to purchase according to likes, recent searches and product reviews. Review helpfulness aids in reducing risks and uncertainty faced by users in online shopping. With the growing amount of information on the internet and a significant rise in the number of users, it has become important for companies to search, map according to the user's preferences and taste, and provide them with the relevant chunk of information. Therefore, there needs to be a system that can scrape the unwanted reviews. These reviews can be from different online websites or from opinion mined and sentiment analysis. These can also work on factors like the buyer's profile, previous purchases, star ratings, whether the review has been given after purchasing, and so on. Based on all the listed factors and reviewer's trustworthiness, the website from which the user should buy the product can be recommended.

Problem definition

There is a need to design a system to recommend the e-commerce website from which the buyer should buy a certain product. The focus needs to be on factors like user trustworthiness, frequent visits, previous purchases, previous reviews and trustworthy review analysis. Depending on user trustworthiness, sentiment analysis can be performed on trusted reviews to generate the number of positive and negative results.

System development

To develop the correct website for product purchase, various steps need to be performed by considering a number of factors prior to the purchase. This recommendation system development can follow the waterfall model of system development. This model is used to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. It is also

referred to as a linear-sequential life cycle model. It is considered to be simple to use and understand. In a waterfall model, each phase must be completed before the next phase can begin, and the phases do not overlap in any way.

Waterfall model is essential to ensure that appropriate system is developed and meets the requirements in terms of feasibility and implementation. This approach comprises of developing the system through various phases, as represented in [Figure 4.12](#):

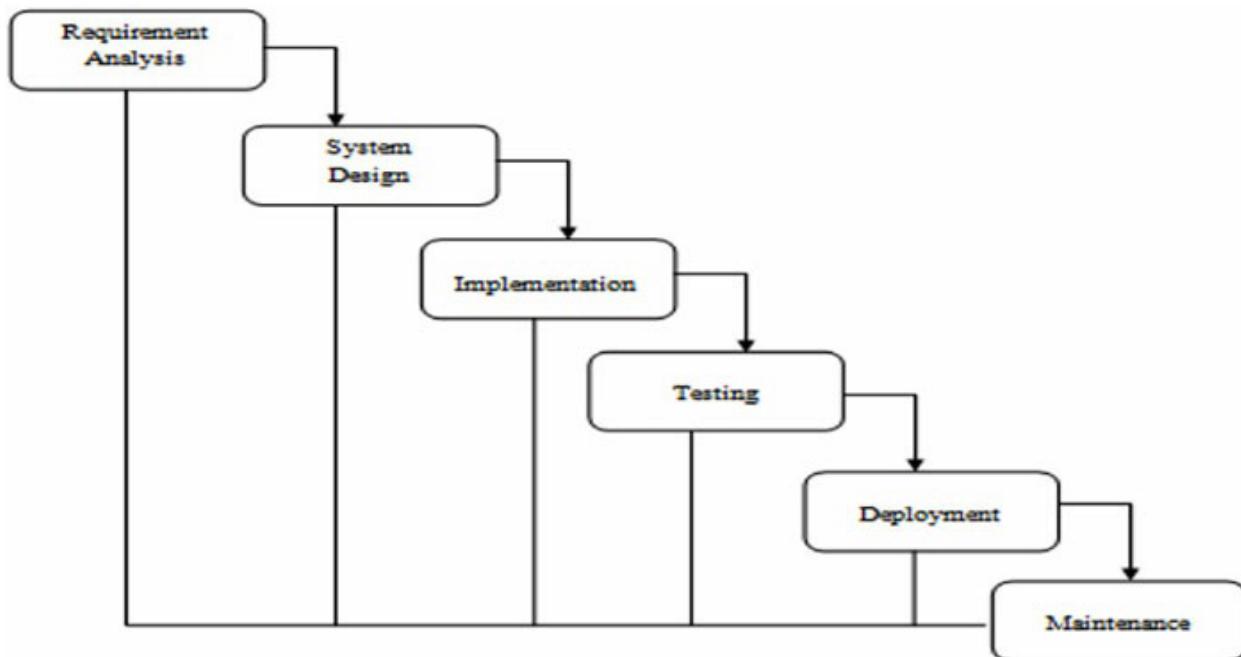


Figure 4.12: WaterFall Model

The phases that represent the Waterfall model include the following:

- **Requirement Gathering and analysis:** All the data requirements are captured in this phase.
- **System Design:** The requirements specified in the first phase are studied and the system design is prepared.
- **Implementation:** Small programs called units are then developed and tested for functionality. These are further moved to the next phase in an integrated manner.
- **Integration and Testing:** All the units developed are then integrated into a system after they are tested. Post integration, the entire system is tested for any faults and failures.

- **Deployment of system:** The product is then deployed in the customer environment or released into the market.
- **Maintenance:** Maintenance of the system or the product developed in the customer's environment is performed.

System Design: The system can start with the user searching for a particular product and end with the recommendation of the website to the user, as represented in [Figure 4.13](#):

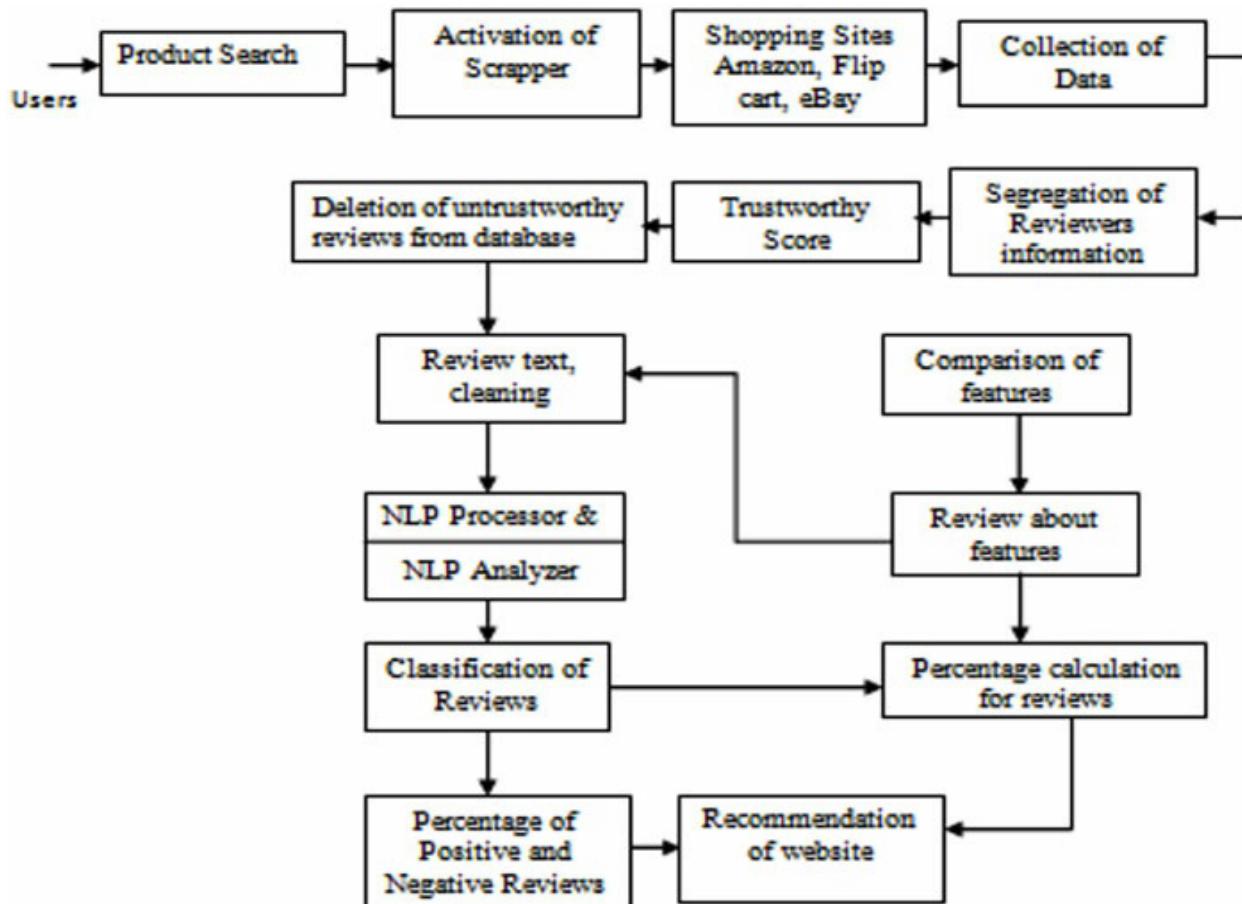


Figure 4.13: The flow of the system

When a user searches for a product on the web page, the reviews from the e-commerce websites can be scrapped. The data collected can include the reviewer's details, the review, trustworthy factors and so on. Untrustworthy reviews or fake reviews can be removed from the database. Further, sentiment analysis can be carried out on the cleaned data using NLP to recommend the website from which the user should buy the product.

Considerations for the case study

The steps that need to be followed while developing this system is shown in [Figure 4.14](#):

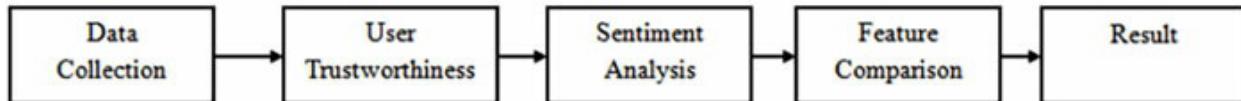


Figure 4.14: Block Diagram

1. Data Collection and Scraping

When the user enters the product, the data is collected from various e-commerce websites, such as Amazon and Flipkart. This data includes the images of the product, features of the product, product ID, star ratings and reviews from each website. All this data is obtained through scrapers, which are different for different websites.

2. User Trustworthiness

This block will determine whether the review given by the specific user is trustworthy based on various factors of the user like previous reviews, previous purchases, visits and so on.

3. Sentiment Analysis

Based on the trusted reviews, sentiment analysis is performed in this block, as shown in [Figure 4.15](#). For this operation, Python libraries and NLTK can be utilized. This analysis would categorize the reviews as positive or negative and can be implemented using the Logistic Regression Model to produce results. Refer to the following [figure 4.15](#):

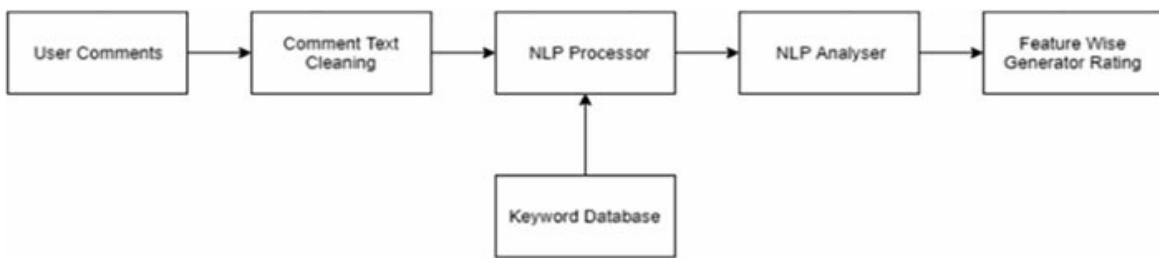


Figure 4.15: Sentiment Analysis

4. Feature Comparison

The features recorded during the scraping process are compared in this block. These features run through the sentiment analysis and produce

much better results.

5. Results

This block finally recommends the website from which the user should buy the product based on the previous analysis done.

6. Algorithms that can be employed

Various algorithms/programming languages can be applied:

- a. **Implementation of Scrapper:** A web scrapper is a tool designed to accurately and quickly extract data from the web pages. This data is vital and becomes the differentiator between the same product on various websites like Amazon and Flipkart. For analyzing and accessing the real-time reviews, scraping is required. Web scraping is the process of automating the fetching of the required data in an efficient and agile manner. This is also known as web extraction or harvesting. The scrapper is implemented in two phases:
 - i. **Scrapper for initial links** can be implemented in JavaScript. This element can automate the process and search for the desired product, name, photo, price and so on to extract the relevant links.
 - ii. **Scrapper for e-commerce websites** can be implemented using Scrapy, which is a scraping framework in Python. It helps in quickly scraping large amounts of data with various features like seamless extraction with IP jumping. These scrappers focus on scrapping the data other than the basic information, like reviewer's name, review title, review text, review date, star rating and whether the purchase is verified for that product on both websites.

7. Implementation of Sentiment Analysis Model:

The sentiment analysis model can be implemented using Python and various Machine Learning libraries such as NumPy, Pandas, Matplotlib, Sklearn, and NLTK. NLTK is a base library for sentiment analysis that contains text processing libraries for basic operation like tokenization, lemmatization and filtering. The algorithm that can be used for Sentiment Analysis is logistic Regression. The model is trained using the Amazon open-

source product review dataset and then can be used on real-time data.

8. **Database Management:** The unstructured data that is scrapped from the web pages needs to be stored for analysis. The data fetched can be stored in the database in CSV format. For database management, Firebase and MongoDB can be used:

- i. **Firebase:** It can be used during the initial scrapper for the links to store the links for the products and their corresponding features, like product dimensions, battery power, connectivity, weight, display, camera, and color.
- ii. **MongoDB:** MongoDB, an unstructured database, can be used to store all the contents of the review. These can be used later when the same product is searched, thus saving computing time.

9. **Trustworthiness Filters:** Following are the trustworthiness filters:

- i. Depending on the user and other properties, the trustworthiness of the review is determined. The output of this determination indicates whether the sentiment should be included for analysis.

For Amazon, only the reviews of verified purchases by the reviewer are considered and scraped. The reviews that have more than 10 helpful votes are considered. For Flipkart, the reviews with certified buyer badge are considered. The importance of the review could be assigned on the basis of likes (upvotes) and dislikes (downvotes).

- ii. **UI Design:** Following are the resources that can be used to design UI:

- i. The user interface can be designed using the React.js framework and Express.js as the backend.
- ii. Flask, a Python framework, can be used to connect the user interface with other elements.

10. **Evaluation Measures:** In order to assess the effectiveness of the recommendation system, three different types of evaluations can be performed:

- a. **User studies:** They are small scale. A few dozens or hundreds of users are presented with the recommendations given by different

recommendation approaches, and then the users judge which recommendations are the best.

- b. **Online evaluations (A/B tests):** In A/B tests, more than a thousand users are shown the recommendations of the real product. Based on the outputs suggested by the recommenders, the recommender system randomly picks at least two different recommendations. These are further used to generate the most appropriate recommendation. The effectiveness is measured with implicit measures of effectiveness such as conversion rate or click-through rate.
- c. **Offline evaluations:** It uses past data to compare how different algorithms have performed.

Opinion mining

Sentiment analysis and opinion mining are almost the same thing. However, opinion mining extracts and analyzes people's opinion about an entity, while sentiment analysis searches for the sentiment words/expression in a text analysis and then classifies them into positive or negative reviews. Further details, features, likes and dislikes are not revealed. To obtain such details, a feature-based opinion mining approach must be applied. This approach typically consists of two steps:

1. Based on the reviewer's opinion, features of an object, topic or event from each sentence are identified and then extracted.
2. The opinion of all the users about the features are determined.

The opinion, be it positive or negative, is determined based on the features considered. This could further provide the users with some insightful information related to opinions on a particular topic.

Image processing

It is a method in which input is provided to the system in the form of images. These images are then processed to either get an enhanced image or extract some useful information from them. It is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. Fixed sequences of operations are performed at

each pixel of an image. The image processor performs the first sequence of operations on the image, pixel by pixel. Once that is done, a second set of operations are performed. The output value can be computed at any pixel of the image. Thus, image processing is a method that tends to either enhance the image or gather alerting insights from it. These insights can further be fed to an algorithm in order to classify, predict or cluster images. The image processing system treats all images as 2D signals where predetermined signal processing methods are applied.

Everyone uses smartphones in their daily lives. With technology, better cameras and visual effects are major considerations while buying a new smartphone. Depending on the viewpoint and illumination, the human brain can recognize many objects, categories and scenes. This is due to the most important sense organ: vision. Any vision task fundamentally relies on the ability to recognize objects. So, object recognition itself forms a part of recognizing an image, transforming and then performing information retrieval on it. Thus, an image has a very important role in image processing. Challenging problems in image processing include classifying and identifying objects in images. Digital image processing techniques help in manipulating these digital images through the use of computers. Thus, image processing forms the core part of computer vision and plays a crucial role in many real-world examples like robotics, self-driving cars and object detection.

Importance of image processing

Following are the points explaining the importance of image processing:

Image: Based on the number of pixels, an image is said to be represented by its dimensions (height and width). For example, if the dimensions of an image are 600 x 400 (width x height), the total number of pixels in the image is 240000.

The pixel forms a point on the image that takes on a specific shade, opacity or color, and is represented as follows:

- **Grayscale:** The pixel is represented as an integer having a value between 0 and 255, where 0 indicates complete black and 255 specifies complete white.
- **RGB:** In this, the pixel is made up of 3 integers between 0 and 255.

Each integer represents the intensity of red, green, and blue colors.

- **RGBA:** Similar to the RGB, this image representation is an extension of RGB with an added alpha field that represents the opacity of the image.

Purpose of image processing

Image processing is divided into five groups based on the purpose:

- **Visualization:** Its purpose is to observe and find the objects that are not visible in the image.
- **Image recognition:** It distinguishes between images or detects objects in the image.
- **Image sharpening and restoration:** Its aim is to create an enhanced image from the original image.
- **Pattern recognition:** It deals with measuring the various patterns around the objects in the image.
- **Image retrieval:** It focuses on browsing and searching images of interest.

Basic Image Processing System

An image processing system has various steps, as shown in [*Figure 4.16*](#):

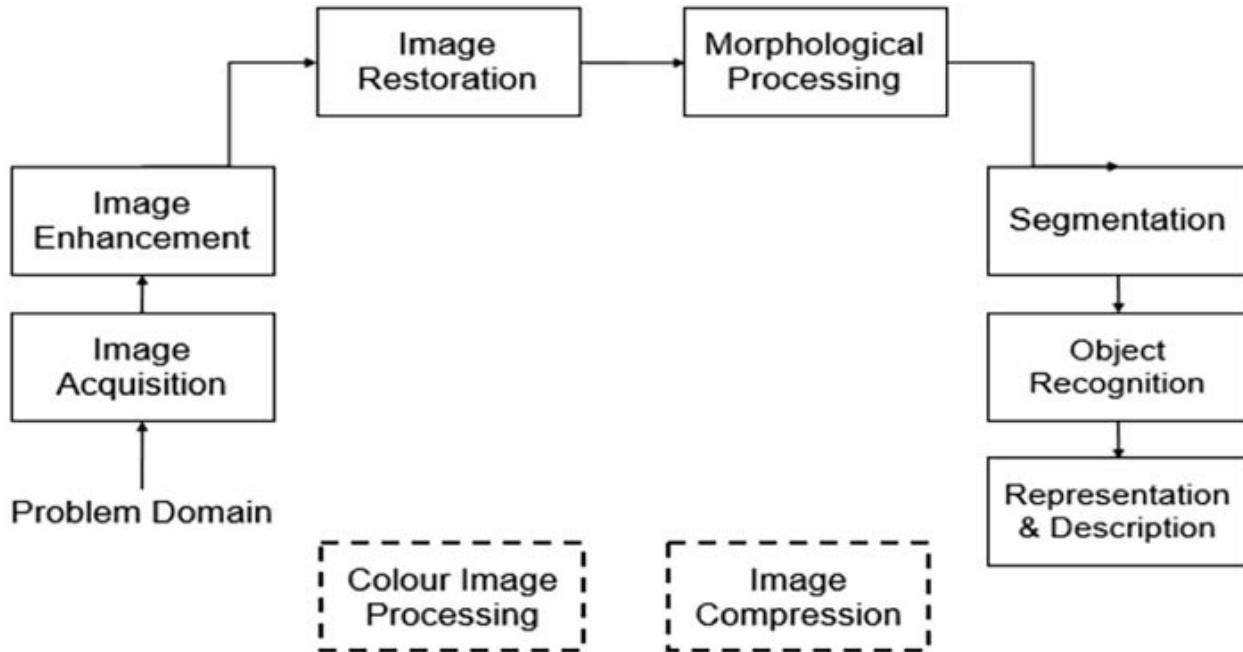


Figure 4.16: Basic Image Processing Steps

1. **Image acquisition:** This forms the first step or process in digital image processing. It is defined as the action of retrieving an image from some source. The image so acquired is a complete raw or unprocessed image. Therefore, pre-processing operations like scaling need to be performed on them.
2. **Image enhancement:** It is a process of adjusting the digital images so that the results are suitable for display or further image analysis. The focus is on modifying the images so that they are more pleasing to the eyes. As the name suggests, the image enhancement techniques aim to bring out the detail that has got obscured or highlight certain features of interest in an image. This step includes sharpening of images, adjusting the brightness and contrast, removing noise and so on. As the features vary from person to person, this step is subjective in nature.
3. **Image restoration:** The appearance of an image can be improved through the image restoration mechanism. However, image restoration is objective in nature as compared to the image enhancement step. Thus, it is a process of recovering an image that has been degraded by some degraded knowledge function, along with addition of noise. These restoration techniques are based on mathematical or probabilistic models of image degradation.

4. **Color image processing:** Color image processing includes processing of colored images, either as RGB or indexed images.
5. **Image compression:** Its main aim is to reduce the storage required to save an image or the bandwidth to transmit it.
6. **Morphological Processing:** Morphological processing is useful in the representation and description of the shape of an image. Basic morphological operations, like dilation and erosion, are performed in this step.
7. **Segmentation:** Segmentation procedures partition an image into multiple segments or objects and is the most difficult part in image processing.
8. **Representation and description:** This step is divided into two parts: the representation step and the description step:
 - a. **Representation:** It converts the data into a suitable form for computer processing in terms of the following:
 - i. **Boundary representation:** It mainly focuses on the external characteristics of the image, such as the corners.
 - ii. **Regional representation:** This part focuses on the internal characteristics of the image, such as the texture of the image.
 - b. **Description:** It deals with extracting the attributes that result in some quantitative information of interest and distinguishes one object from the other.
 - i. **Object recognition:** Based on its descriptors, a label can be assigned to an object through the process of object recognition.
 - ii. **Knowledge Base:** Knowledge about a problem domain is coded into image processing in the form of a knowledge database. This knowledge base can be simple or complex, for example, information about an interrelated list of major possible defects in a materials inspection problem.

Image Processing using popular ML algorithms

In image processing, algorithms are used to identify and detect various vital

components or desired parts and features of the image. The model can be trained to recognize certain patterns through computer vision. This data can be further stored into their artificial memory. Result predictions can later be performed on these stored data values.

Some familiar use cases that leverage ML image processing techniques are listed here:

- **Medical Imaging/Visualization:** Help medical professionals interpret medical imaging and diagnose anomalies faster
- **Law Enforcement and Security:** Aid in surveillance and biometric authentication
- **Self-Driving Technology:** Assist in detecting objects and mimicking human visual cues and interactions
- **Gaming:** Improve augmented reality and virtual reality gaming experiences

Real Time case studies of Image Processing using ML

Let's look at one of the case studies of image processing, which is in the healthcare domain.

Analysis and prediction of hypertension and vein function in hemodialysis: Kidney failure is one of the most dreaded diseases across the globe. The solution for kidney failures is dialysis, which is majorly performed when the kidney functionality goes below a specific threshold. In order to conduct dialysis, native arteriovenous fistulas are constructed to increase blood flow in the superficial vein. One of the major drawbacks of continuous dialysis is that the patient may suffer from hypertension and reduced vein functioning. This may lead to the collapse of the fistula. Thus, checking on the state of the fistula is important, and it can be performed through the ultrasound doppler test. However, these tests are expensive and repeating the tests a number of times is not feasible. There is a need for a system that would take as input all the factors provided in real time by the dialysis machine and use them to make a prediction on the state of a fistula.

Prediction of hypertension: Kidney failure, also known as renal failure or renal insufficiency, is a medical condition in which the kidneys fail to filter from the

These failures are broadly classified into the following:

- **Acute kidney injury:** This is reversible with adequate treatment.
- **Chronic kidney disease:** This is often not reversible and leads to patients being put on dialysis. Over time, as dialysis is conducted, some patients might suffer from hypertension and reduced vein function, leading to the fistula failing. Developing another fistula takes time, and constant ultrasound doppler scans are needed to check the fistula's health.

Problem definition

Currently, the amount of data being output by the dialysis machines is just used for dialysis; it is not stored. Therefore, there is no prediction and most procedures follow an action response method where a new fistula is created once the old one fails catastrophically, so hypertension is not treated proactively.

There is a need for a system that could help nephrologists to treat the patients, prevent them from entering a hypertensive state and keep them informed as to the state of the fistula. This can improve the quality of life of the patient while they undergo dialysis.

Objective of the case study

The major objective of this study is as follows:

- Given the continuous data stream from a dialysis machine, a prediction needs to be made about whether the patient is likely to suffer from hypertension due to dialysis.
- To check the vein function and predict the lifetime of the fistula created for dialysis.

Considerations for the case study

The parameters that need to be considered for predictions include heart rate, systolic BP, diastolic BP, venous pressure, arterial pressure and cumulative blood volume slope. These parameters are further divided into sub-parameters recording the maximum, minimum, mean and standard deviation.

System development

The system can be developed using an overall block diagram, modular diagrams and data flow diagrams for convenience. Refer to the following [figure 4.17](#):

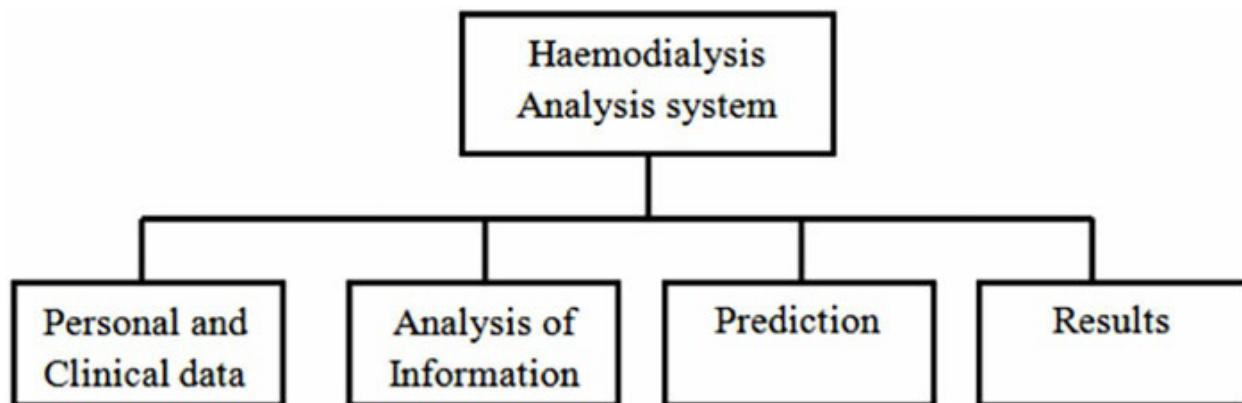


Figure 4.17: Block diagram representation

[Figure 4.17](#) highlights the different subsystems to be built:

- **Personal and Clinical Data interfaces**

This acts as the data collection interface where the technicians and the doctors can upload data file of single or multiple dialysis sessions into the system.

- **Analysis of Information**

Preliminary segmentation of clinical, personal and miscellaneous factors can be done based on the respective conditions and set ranges.

- **Prediction Model**

The algorithms used for the model for the prediction of likeliness of hypertension and life of fistula can be developed here.

- **Results**

Predicted results can be evaluated and compared with the actual data of the patient. Various reports like accuracy, analysis, efficiency and prediction reports can be generated in this module of development.

Algorithms that can be employed

Various algorithms that can be applied include the following:

- **Support Vector Machines:** SVM or Support Vector Machines are a supervised learning method that allows us to perform classification and regression analysis on a dataset. It helps capture more complex relationships between data points without performing any difficult transformations. The kernel takes the data and transforms the data. Several features are provided to the said algorithm. The algorithm runs till the required tolerances are achieved as there is no set limit on the number of iterations that needs to be performed.
- **Random Forest Classifier:** Random Forests is a supervised ensemble classification algorithm, where ensemble classifiers are randomly created decision trees. These trees act as classifiers and the target prediction is based on the majority voting method. They can be used for identifying the most important features from the training dataset. This classifier uses averaging to improve accuracy and have a control on overfitting.
- **Multi-Layer Perceptron:** MLP or Multi-Layer Perceptron is a supervised learning algorithm that is a form of feed forward artificial neural network with at least three layers of nodes. It performs better when it comes to smaller data sets. A MLP with hidden layers of sufficient size can approximate any continuous function to any desired accuracy, does not impose any restrictions on the input variables, can better model heteroskedasticity, that is, data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data.

These algorithms can be utilized as they can handle smaller datasets with high accuracy.

Tool Utilization

- **Scikit Learn:** This open-source software library for machine learning can be used for building and training neural networks to detect and decipher patterns and correlations.
- **Hybrid Application Frameworks:** These application frameworks help develop the software quickly, requiring minimal extra actions on the part of the user.
- **Django, Javascript, CSS, HTML 5:** The said technologies constitute a

full stack. HTML5, CSS3, and JavaScript for the front end, and Python (Django) and SQL can be used for backend.

- **OpenCV:** Open Source Computer Vision Library is used to provide a common infrastructure for various image processing applications.

Constraints/Limitations while developing the system

- The major issue while applying the algorithms is that the data is to be taken directly from the dialysis machines.
- Also, not all dialysis machines can provide the exact same data points.
- The performance of the system is limited by the sheer volume of data. So if the data involved is huge, a significant amount of time would be required to pre-process the data.

Evaluation Measures

The dataset provided can then be run on the previously mentioned machine learning algorithms. These can further be evaluated for imbalances between data of good and bad fistula. Dataset can be split into 80-20 sets for training and testing to maintain a blind test. Evaluating these algorithms is important for assessing the effectiveness of the output obtained. The commonly used metrics for evaluation can include accuracy and precision. This can be followed by determining the accuracies of false positives, false negatives and the overall accuracy of each algorithm. It helps us to get a proper insight of the algorithms to be used.

Predictive analytics

It is defined as a statistical technique that uses machine learning and data mining techniques to predict and forecast likely future outcomes. Historical and existing data is analyzed and a model is generated to forecast likely outcomes. For example, in order to determine the amount of revenue generated for a software company, there would be a need to create a model. This model can be formed based on the historical sales data and marketing expenditures across multiple regions. This enables them to determine the impact of the marketing spend.

As the name suggests, **Predictive analytics** is derived from two terms:

Predictive, which deals with making predictions and **Analytics**, which performs analysis on the data.

Therefore, it is said to be a branch of advanced analytics that is used to make predictions about unknown future events.

Importance of predictive analytics

In order to analyze the current data and make predictions about future outcomes and performances, it uses techniques ranging from data mining and statistics to modeling, machine learning and artificial intelligence. These techniques focus on identifying the likelihood of a possible outcome based on the past or historical data provided. Thus, the aim of predictive analysis is to provide the best assessment of what will happen in the future.

Need for predictive analysis

As mentioned earlier, predictive analytics provides an assessment. Various companies are utilizing it to increase productivity and gain a competitive advantage.

The reason for considering this technique are as follows:

- Huge amount of data is available across the globe.
- Valuable insights can be generated from the various forms of available data.
- The software available for performing predictive analysis are cost-friendly.
- Various insights provide for providing by provide competitive differentiated products.

Major characteristics of predictive analytics are as listed here:

- Some predictive models are complex in nature
- Most predictive models operate at a fast rate
- Predictive analytics perform their calculations in real time
- The predictive model is revised regularly to incorporate changes in its underlying data

Machine Learning vs Predictive Modeling

Table 4.3 represents the comparison between machine learning and predictive modeling:

Characteristics	Machine Learning (ML)	Predictive Modeling (PM)
Definition	Data is provided at the input without applying any set of predetermined rules and regulations	Current and historical sets of data are provided to find patterns and behaviors in predictive modeling algorithms
Operations	Machine learning algorithms are trained to learn from their past mistakes to improve future performance	It makes informed predictions based on historical data about future events only
Principle	Machine learning forms the core set of algorithms for performing predictive modeling	Predictive modelling can be said to be a subset and an application of machine learning
Working	Whenever a new set of data is added to ML systems, they can adapt and learn from it automatically, without the need of being directly programmed	It includes a series of statistical techniques and uses statistics to estimate or predict future outcomes, given a set of input data
Techniques used	ML algorithms are broadly classified into supervised and unsupervised learning algorithms	Predictive modeling is further identified through descriptive analysis, diagnostic analysis, predictive analysis and prescriptive analysis
Approaches and Models	The various approaches considered in machine learning include Decision tree learning, Association rule learning, Artificial neural networks, Deep learning, Inductive logic programming, Support vector machines, Clustering, Bayesian networks, Reinforcement learning, Genetic algorithms and Rule-based learning	Data in predictive modeling is handled using Naïve Bayes, K-nearest neighbor, Support vector machines, Boosted trees, Random forests, Classification and Regression trees(CART) , Neural Networks, Ordinary Least Squares, Logistic and Robust Regression algorithms

Applications	Applications that utilize machine learning algorithms include Bioinformatics, Brain-machine interfaces, Computer vision, Object recognition, Detecting credit card fraud, Linguistics, Marketing, Machine perception, Medical diagnosis, Economics, Insurance, and Online advertising	These include Uplift modeling, Archaeology, Customer relationship management, Auto insurance, Healthcare, Marketing campaigns optimization, Fraud detection, Risk reduction, Customer retention, Sales funnel insights, Crisis Management, Disaster Management, Financial modeling Market trend and analysis, and Credit scoring
Update Handling	Updations are performed automatically in ML algorithms	These models require updations to be run manually multiple times
Utilization of Technology	ML modeling are data-driven models	PM techniques are use case-driven systems
Drawbacks	<ol style="list-style-type: none"> 1. Large amounts of data are needed to train the system 2. Correct ML algorithm needs to be chosen 3. Only the right algorithm will give an optimized solution 	<ol style="list-style-type: none"> 1. It requires a large amount of data that can be applied for predictive analysis 2. For making accurate outcomes, apt method needs to be applied

Table 4.3: Machine Learning Vs Predictive Modeling

Building a predictive model

Prior to building a predictive model, we need to identify the historical data and the outcome that needs to be predicted. This is because the model can infer outcomes from historical data but cannot predict the data it has never seen before.

Here are the steps used to build a predictive analytics model:

1. **Ensure correctness of the input data:** The correctness of the data input, both in terms of its volume and breadth, is critical to determine an accurate prediction.
2. **Clean the data:** Various ways to clean, transform and combine the raw data need to be determined for better predictions. The data so obtained needs to be properly transformed into appropriate data features that are

most suitable for a given model.

3. **Remove unwanted data:** Removing data that is not relevant to a model needs to be eliminated as the additional data can slow the model down or lead to less accurate models.
4. **Determine the predictive modeling approach to be used:** Approaches like parametric and non-parametric approaches need to be chosen to perform predictive modeling and analytics.
5. **Preprocess the data into a form suitable for the chosen modeling algorithm:** The cleaned and transformed data, as represented in step 2, needs to be further processed and readied depending on the approach decided in step 4.
6. **Identify the data to be used for training the model:** Data used for the purpose of training the model has to be identified.
7. **Train, or estimate, model parameters from the training data set:** The actual model building starts here by determining and estimating the training and model parameters for the model to be developed.
8. **Determine the model performance and check the model's adequacy:** The performance of the model developed and considered as per step 4 is verified, and it is validated for accuracy.
9. **Apply the model on the test data and determine its accuracy:** Based on the acceptable level of performance and accuracy of model considered, the model is tested on the test data.
10. **Utilize the model for further predictions:** Once the performance of the model satisfies the accuracy levels and is acceptable, it can be used for other sets of predictions.

Types of predictive algorithms

Predictive algorithms either use machine learning or deep learning for performing predictions, where machine learning works on structured data while deep learning deals with unstructured data like video, audio and text. Let's look at the algorithms commonly used in predictive analytics.

Decision Trees

- They are one of the simplest models and are considered easy to understand.

- These techniques aid in making decisions quickly.
- These represent tree-like structures having branches and leaves, where the branches represent the available choices and the leaves indicate a particular decision.
- The data of the trees are placed into different sections based on certain characteristics or variables.

Random Forest

- A combination of decision trees forms the random forest algorithm.
- The combined trees are distinct and are not related to each other.
- They can use both classification and regression to classify large amounts of data.

Regression

- As mentioned in the previous chapter, in the regression model, the relationship between all the inputs found in the dataset is represented through a formula.
- This model uses statistical analysis to determine patterns in large sets of data when there is a linear relationship between the inputs.

Neural Networks

- Complex data relationships are handled through these.
- These models utilize artificial intelligence and pattern recognition to determine the relationship.
- It is mainly used when there is too much data on hand and no formula suffices to find a relationship between the inputs and outputs in the dataset provided.

Generalized Linear Model (GLM) for Two Values

In this algorithm, the number of variables is reduced to find the “best fit” for the data variables considered.

Gradient Boosted Model

- It uses several combined decision trees that are related to each other.
- It builds one tree at a time.

- The next related tree is built to correct flaws in the previous tree.
- It is used in developing rankings.

K-Means

- It is considered a fast algorithm.
- It groups similar data points together.
- K-means algorithms are often used for the clustering model.

Prophet

- These algorithms are usually used for capacity planning, inventory needs, sales quotas and resource allocations.
- They are highly flexible.

Types of Predictive Analytical Models

Based on the previously mentioned techniques, predictive analytics models are broadly classified into the following:

Classification model: It is considered to be the simplest model that categorizes the data to handle simple and direct query responses. An example use case would be to answer the question “Is this a fake account number?”.

Clustering model: As the name suggests, this model groups data based on common attributes. This grouping can be done by bringing together things or people with shared characteristics or behaviors. For example, based on the operations done by people in the same or similar situations in the past, loan approval can be given to a person.

Forecast model: Based on learning from historical data, this model works on numerical data. For example, at a call center, the number of calls a call center executive should be able to handle per day or week is determined by looking at historical data.

Outliers model: This model works by analyzing abnormal or data points outside a specified range. For example, a bank can use this model to determine whether a transaction done by a customer is outside of their normal buying habits. This would enable them to determine a fraud situation.

Time series model: The evaluation in this model is based on time as a parameter that is applied on a sequence of data points. For example, based on

the number of patients admitted to the hospital over a period of weeks, predictions can be made as to the number of patients the hospital might expect in the near future, over months or the rest of the year.

Comparison of Predictive Modeling and Predictive Analytics

The differences between predictive modeling and predictive analytics are listed in *Table 4.4*:

Characteristics	Predictive Modeling	Predictive Analytics
Steps of operation	1. Data gathering and cleansing 2. Data analysis/transformation 3. Building a predictive model 4. Inferences/evaluation	Business process includes: 1. Defining the project 2. Data collection 3. Data analysis 4. Statistics 5. Modeling 6. Deployment 7. Model monitoring
Process	It is an iterative process	It analyses historic and transaction data
Techniques applied	Runs one or more algorithms on data sets that are reusable	Applies statistics, data mining, machine learning and artificial intelligence techniques to predict the outcome
Classification	Is classified into parametric and non-parametric model	Comprise predictive, description and decision related models
Applications	Used in vehicle insurance, healthcare and so on	Used in project risk management, fraud detection
Type of model/analytics performed	Types of model categories include predictive, descriptive and decision models	Types of analytics performed are regression and machine learning techniques

Table 4.4: Predictive Modeling and Predictive Analytics

Predictive modeling vs data analytics

The difference between predictive modeling and data analytics is elaborated on in [Table 4.5](#):

Characteristics	Predictive Modeling	Data Analytics
Definition	It is defined as a statistical technique to predict future behavior.	It is defined as a science of analyzing raw data to make conclusions about that information.
Known as	It is known as predictive analytics.	It is also known as data analysis.
Term used	It is termed as predictive modelling in academic applications and predictive analytics in commercial applications.	<p>It is termed as:</p> <ol style="list-style-type: none"> 1. Prescriptive analytics on the basis of data analysis 2. Predictive analytics on the basis of the prediction of future possibilities of business events 3. Descriptive analytics on the basis of extracting the description based on the data 4. Diagnostic analytics or diagnostics on the basis of determining the cause behind the glitches in the business processes
Depends on	It depends on the access to sufficient volumes of accurate, clean and relevant data.	It depends on the data being used and the goal of the analysis.
Process	It follows an iterative process, due to which it automatically creates accurate predictive models about the future.	It is used to study the behavior of the created model, which is about to be predicted.
Algorithms Used	Decision trees, K-means clustering and neural network	Linear and logistic regression, clustering, K-nearest neighbors, PCA, time-series and sequencing
Used for	It is extensively used in analytical customer relationship management and data mining to produce customer-level models.	It is used in finding hidden patterns, unidentified correlations, customer preferences and market trends.

Solutions	Predictive modeling solutions are a form of data-mining technology that works by analyzing historical and current data and generating a model to help predict future outcomes.	Business Intelligence (BI) and analytics services provide a foundation for insight and analysis to make more insightful business decisions and execute them quickly.
-----------	--	--

Table 4.5: Comparison between Predictive Modeling Vs Data Analytics

Comparison between and Predictive Analytics and Data Analytics

Table 4.6 compares predictive analytics and data analytics:

Characteristics	Predictive analytics	Data Analytics
Definition	It operates on the current and past data trends.	It deals with refining the data.
Prerequisite	Technical and statistical knowledge is required.	Strong statistical knowledge is required to perform data analytics operations.
Focus	The focus is on inferring a pattern on which predictions could be made.	It focuses on drawing conclusions from the dataset.
Objective	It is a specialized form of analytics whose main purpose is to evaluate the risk and predict future outcomes.	It is a general form of analytics whose purpose is to make data-driven decisions.
Steps for building the model	Model the data, train the model, then predict and forecast the outcome.	Collect, inspect, clean and transform the data in order to reach the conclusion.
Approach	Forecasting and predictions use advanced computational models and algorithms.	In order to build deep insights on data, traditional algorithms are used.
Procedure	Clean data is used to build a predictive model.	Raw data is collected, cleaned and transformed to perform data analytics.

Outcome	Reliable predictive model is generated as an outcome of predictive analytics.	Predictive or non-predictive models can be obtained as an outcome of data analytics.
Usage	It helps to answer questions such as “what will happen if”, for example, ‘What will happen if the demand of a product goes down by 10%?’	It can be used to find hidden patterns, unidentified correlations, customer preferences that could help to make more informed decisions for businesses.
Applications	Sales Forecasting, Crisis Management, Analytical customer relationship management, Clinical decision support systems (CRM)	Fraud and Risk Detection, Delivery Logistics, Customer Interactions, Digital Advertisement and so on

Table 4.6: Comparison between Predictive Analytics and Data Analytics

Uses or applications of Predictive Analytics

Predictive analytics are utilized by various organizations to help them solve difficult problems, uncover new opportunities and aids by acting as a decision-making tool. Predictive analytics is used in a variety of industries, and can be relevant and applied in many sectors, such as banking and financial services, retail, pharmaceuticals, healthcare, insurance, oil gas and utilities, government public sectors, and aerospace. These analytics can be used in a number of areas:

Forecasting: Predictive analytics has an important role in the manufacturing sector. As manufacturing units work on the principle of supply chain management, it requires the right amount of data to be input to the system. This ensures smooth functioning of the manufacturing system and maintaining timelines. Therefore, to ensure optimal utilization of resources, there is a need to forecast the right amount of resources (data) at the input and maintain proper inventory of the raw material required for manufacturing the product. Predictive modeling enables such accurate forecasts by providing clean, good-quality and correct amount of data as input.

Reducing Risks: Based on a consumer's behavior and credit history, their credit ratings are determined. This helps the application categorize the consumers into premium customers, regular, defaulter customers, corporate spying or cyber-attacks. Predictive analytics thus helps to determine the

credit rating of an applicant or consumer. Based on the applicant's credit history, it is determined whether the consumer will be able to repay the extended credit. Thus, predictive analytics is used to predict the risk that the applicant might fail to perform on any credit extended.

Underwriting: Insurance companies need to determine whether they will have to shell out money on claims made by the applicants on various causes, on a case-to-case basis. This would ensure that the company would be able to manage their financial portfolio accordingly. The method employed by the companies is to analyze the past records of the applicant as well as the claims filed by similar policyholders. These predictions can be performed by applying several predictive analytics models.

Optimizing Marketing Campaigns: Various companies research on how their products would be taken by the consumers. The way people will react, the right time to launch a product, forecast future price movements, promote cross-sell opportunities, retain and ensure its profit, and so on are all analyzed. Performing this analysis on a manual basis would be tedious, so companies tend to do so through the predictive analytics models to determine how consumers have reacted to the overall economy and accordingly plan the launch of their products. Thus, it can further be used to unearth new customer insights and predict their behaviors. This allows organizations to tailor marketing strategies and retain their valuable customers.

Detecting fraud: Criminal behavior can be determined and abnormalities like cyber fraud, cyber thefts and other cybersecurity issues can be detected through the application of predictive analytics methods like pattern detection and pattern recognition.

Customer service: Predictive analysis techniques can be used to determine customer behavior and target regular and potential customers.

Other applications of predictive analytics include transaction profiling, predictive search advertising, recommendation engines, decision-making tools for influencing upselling, sales and revenue forecasting, manufacturing optimization, weather forecasts, creating video games, translating voice to text for mobile phone messaging, customer services, and investment portfolio development.

Benefits of predictive analytics

Various benefits of using predictive analysis include the following:

- **Improved decision making:** Better data analysis can be done if there is a lot of data that is relevant and available. The consequences of better data analysis results in better informed decision-making, identifying trends and patterns. This enables organizations to get more detailed insight into its systems for smooth working.
- **Improving efficiency of operations:** The focus of any organization is to increase its profits and minimize its losses. By using predictive analytics, organizations are able to forecast its inventory and manage its resources. This analytic further helps organizations to improve production processes, optimize performances, increase revenues and thereby, make it more efficient in its operations.
- **Reducing time, effort and costs:** Predictive analytics reduce time, effort and costs in forecasting business outcomes like demand forecasting, churn analysis, competitive analysis and financial risks.
- **Improving profit margins:** In order to maximize sales, it is used to forecast inventory, create pricing strategies and predict the number of customers.

Challenges of predictive modeling

The following are the challenges of predictive modeling:

- Too much data availability can develop meaningless or erroneous outcomes.
- Large volumes of data get exposed in predictive modeling.
- Maintaining security and privacy is also a challenge that needs to be handled.
- **Data preparation:** Acquiring and obtaining the correct and right amount of data that could be used for performing predictive modeling is a challenge. Almost 80 % of the development time is spent in getting the right data.
- Data collection is yet another challenge faced to obtain the desired results as output. This is only possible if the data is cleaned and properly managed.

- **Overfitting of information:** There is a need to avoid overfitting of data. If the testing on training data is overdone, it could result in a model that appears very accurate. However, this model is applicable to a single-use case and therefore, would not be able to generalize the model.
- **Technical and cultural barriers:** Organizations tend to avoid sharing their data as it is considered to be their asset. Therefore, systems that store useful data are not connected directly to a centralized pool of information. So, even though predictive modeling is considered to be a mathematical problem, getting the correct, substantial data for modeling may be a challenge faced by the users.
- **Choosing the right business case:** Developing algorithms and insights into a business problem need not address significant business challenges. On investigation, it could be observed that the correlation developed between the results and the problem faced does not actually present an insight that the business can use. Thus, it can be said that predictive modeling initiatives need to have a solid foundation of business relevance.
- **Time as a factor:** Time plays an important role in how predictive modeling systems work for a business. Though a model may be successful at one point in time, customer behavior changes with time, so a model must be updated.

Limitations of predictive modeling

Following are the limitations of predictive modeling:

- **Errors in data labeling:** These can be handled by applying either reinforcement learning or **Generative Adversarial Networks (GANs)**.
- **Shortage of massive data sets:** One of the options that can be applied by a machine is to use a small number of datasets for training as compared to using a massive one.
- **Transparency:** Machines give output without providing an explanation, unlike human beings. Also, the computations that are applied through the machine learning algorithms are complex. Therefore, there is a need to have a model that can fix the errors and is transparent, locally interpretable, safe and secure.

- **Generalizability:** Machine learning algorithms work on a case-by-case basis. It is difficult to apply what has been learnt to a new set of circumstances. The generalizability of learning and applying what is learnt for generalized systems is the need of the hour. Hence, for applying predictive modeling, providing generalized learning, making the systems reusable and applying to more than one use case concept of transfer learning can be employed.
- **Bias in data and algorithms:** Another major challenge faced is the use of biases. Biases are naturally introduced into the system through historical data, and the past outcomes reflect the existing bias. These biases tend to skew the outcomes and are difficult to find and resolve.
- **Predictive modeling tools:** Enhanced with AI, predictive analytics tools are easier to use by data scientists and business users. Various tools used in the industry include the following:
 - **Sisense:** It is a business intelligence software that offers a range of business analytics features, requiring minimal IT background.
 - **Oracle Crystal Ball:** It is a spreadsheet-based application that can be used for predictive modeling, forecasting, simulation and optimization related applications.
 - **IBM SPSS Predictive Analytics Enterprise:** It is a business intelligence platform that supports open-source integration and features descriptive, predictive analysis and data preparation.
 - **SAS Advanced Analytics:** It offers algorithms that identify the likelihood of future outcomes. This can further be used data mining and forecasting.

Case studies

One of the case studies deal with social media and performing analytics on it. Let's look at it.

Social media analytics

Considering the slogan “*too much information, too less understanding*” holds relevance even today with a lot of data pouring in from all quarters. To add to this, there has been a lot of data coming up through social media

applications like Twitter and Facebook, and several internet technologies affecting our daily lives. Rather than turning social media off, there is a need to design new filters for information processing.

Introduction: Social media is trending these days. From an entertainment perspective, almost everyone has an account, interacts and posts their activities on social media platforms like Facebook, LinkedIn, YouTube, and Twitter. From the corporate (business) perspective, usage of social media is seen as a tipping point for gathering information about customers (users) and their interactions, purchases, clicks and so on. Therefore, companies are channelizing their funds in social media with the prime focus on the following:

- Reducing the labor-intensive market research to intelligence data gathering and capturing
- Shifting marketing efforts of brand building to the customer through various channels of social media; by providing the right amount of tuning, applying intelligent filtering, various products can be marketed to customers

Challenges: Challenges faced in social media include the following:

- By analyzing social media data, creating intelligent information is difficult.
- Social media has a lot of information and a high degree of noise.
- Linguistic rules get discarded as Tweets, status updates are small and are usually written using abbreviated words.
- High rate of data demands more online processing of the information that can be captured and summarized.
- Data cleaning and preparing phase is computationally time-consuming.
- Determining the right understanding and correct analysis is difficult. For example, determining whether a social media tweet is just an information or an action to be taken is a decision task to be performed.
- Effectively presenting unstructured intelligence is a real challenge. When we discover a collection of time-varying themes of discussion around a query over various social media streams in real time, representing them meaningfully also requires innovation.

Problem definition: Classification task of labeling a text as expressing either an overall positive, negative, or neutral opinion. Labeling a text as positive, negative, or neutral opinion is a classification task.

Approach: Data for social networking sites can be provided through the interface developed for it. This data can be in the form of a multimedia element. Before establishing the connection, the interaction needs to be authorized. The flow of the system is shown in [Figure 4.18](#):

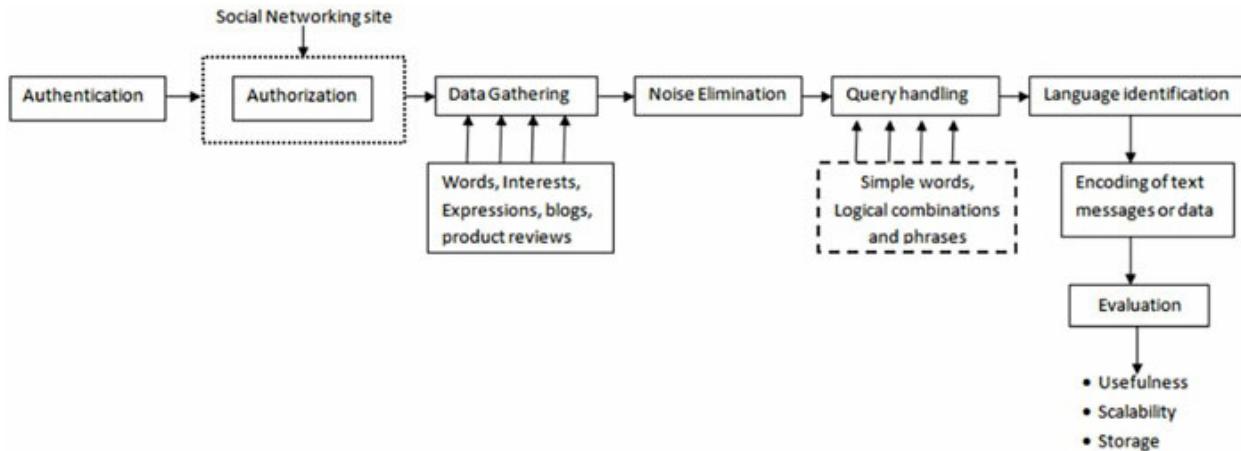


Figure 4.18: Approach for social media analytics

The following are the steps to be considered while designing:

1. **Authorization:** The connection between the system and the user needs to be authenticated and authorized first. The previously mentioned point is handled by building authorization software.
2. **Data collection:** Once authorized by the social networking site, non-private data of users geographic, social and demographic profiles of the people in the conversation can be gathered through API calls. However, various challenges may be faced while developing the system:
 - a. There is a need to maintain a long-duration connection that could stream real-time conversations.
 - b. There is no guarantee on reliability of long duration connection in social networking sites.
 - c. There is also a challenge in effectively identifying spam when messages are short and written with many abbreviations.
3. **Implementation:** Though social media is inherently noisy and filtering

out meaningful information requires throwing out undesirable and repetitive content, there is a need to focus on the following:

- a. Including a mechanism to focus on long-running queries and ensure that no conversation is missed.
- b. With the focus on the queries posed, the replies produced, the interests expressed as words, data through real-time conversations can be gathered; queries could be simple logical combinations of words and phrases.
- c. Information from other social networking sites, blogs and product review sites can be captured.
- d. Once this data is captured, there is a need to identify the language and be able to encode the text messages and the gathered data. This can be done through the language identification technology.
- e. Mathematical model then needs to be created to determine the probability of each word of a message. Another formula can then be used to select the message whose summative probability would indicate its usefulness.
- f. Once this has been done, the words, messages and probabilities can be stored to provide for servicing heavy read/write workloads. This can be done via fault-tolerant systems having efficient storage.

4. **Solution:** Building different sets of metrics and performing the comparative analysis can be used to perform various analytics. This includes the following:

- a. **Daily data:** Based on the day-to-day clicks on social media sites, day-wise information can be gathered and categorized. This categorization could be performed on parameters like geographic locations, demographic measures and others, depending on the query of interest.
- b. **Share of voice:** The opinions raised by people can be clustered together based on some clustering criteria. This clustered data collected can further be presented in terms of volumes and graphs, and the volume of each category can also be presented in a suitable manner.
- c. **Tags assigned by authors:** Authors often assign tags to their own

media through hashtags and blogs. These tags can then be compared based on region or area of interest, and opinions can be formed from them.

- d. **Recurrence of keywords and phrases:** When a number of words, keywords or phrases are repeated, they provide valuable insights into the way brands are perceived by customers over time. This information can then be captured to build a history of trends. Thus, this qualitative analysis provides an understanding of how customer perception changes over time, providing a granular perception of information over time.

Case study of Instagram

Introduction: Instagram is considered to be a social media marketing tool for businesses and a way to foster personal relationships between a brand and its followers. Digitization of content and communication over the internet has made the utilization of social media a source of information and a source of influence. Therefore, there is a need to have some presence on social media platforms. A lack of these avenues is considered unacceptable. So, businesses need to integrate social media into their strategy to attract and interact with customers. As the use of social media, and its patterns and frequencies are changing continuously, it becomes a challenge to assign a right strategy to the right social media platform.

Purpose of case study: To explore the use of Instagram as an app for businesses and consumers

Objectives: To evaluate and compare major social channels in terms of consumer and brand preferences

- To analyze the method of improving business through Instagram.
- To determine the data that can be collected to examine Instagram.

Businesses need to integrate social media and digital marketing strategies to develop their business and interact with customers through social media platforms.

About Instagram: It is an application that enables users to share ideas, connect with those around them, enhance knowledge, and convert the pictures taken into visually appealing images. These images can then be

shared through any social media or any other network.

Different types of social media include social networks, blogs, business networks, collaborative projects, enterprise social networks, forums, microblogs, photo sharing, product/service reviews, social bookmarking, social gaming, video sharing and virtual worlds.

Here are the advantages of using social media:

- Social networks form tools to make a business stand out.
- It provides more exposure to business organizations.
- They increase the awareness of the brand value of the product.
- It helps users review the comments of the purchasers.
- Customers trust social media more than corporate-sponsored messages of product information.
- Companies can earn customer loyalty by creating online communities.
- Social media allows businesses to establish a strong relationship with customers.
- Businesses receive feedback and get new customers.
- Through social media platforms, companies can increase profits.
- Through social media insights about customers' preferences, businesses get leads into creation of new products and informed decisions about the best way to advertise and serve clients.
- Having a presence on Instagram enables direct communication with millions of users across the globe.

Approach: Social media platforms are said to be widely used for communication and information exchange. They are referred to as web applications that facilitate the creation, discussion, modification, and exchange of user-generated content.

The focus of this case study is to employ qualitative content analysis in order to evaluate Instagram's social media strategy. The basic aim of developing Instagram was to combine available pictures with the ease of connecting with people through social media and location services.

The method employed by Instagram was to identify the similarities in how people communicate and enable various aspects of the market. The strategy

incorporated by Instagram is a mirror of Porter's strategy, which focused on creating specialized services in a limited market.

Evaluation: In order to evaluate Instagram, a structured method for organizing and understanding the complex trends, developments, strategy, structure and organizational culture of Instagram were considered. Apart from this, various factors were considered, such as the competitive environment, key obstacles and critical assumptions that impact organizational strategies. The details of the same were as follows:

- A comprehensive strategic analysis included the following:
 - The current competitive environment of the organization
 - Positive and the negative impacts of the actions taken on the systems
 - Various sets of recommendations considered based on the strategic analysis
- Competitor analysis included the following:

Instagram had to gather information about its competitor, and its real and perceived strengths and weaknesses. As the comparisons had to be done, **Strengths, Weaknesses, Opportunities, and Threats (SWOT)** analysis were performed by comparing the SWOT of the competitors. This analysis helped Instagram identify the best and most successful strategy for its future. So, Snapchat and Pinterest being the competitors, a comparison was done on them, as represented in [Tables 4.7](#) and [4.8](#):

Critical Elements	Instagram	Snapchat	Pinterest
Unique Selling Proposition	It is an instant photo sharing/video application.	It is considered to be a video/picture sharing messaging application.	It is said to be a web and mobile sharing application.
Value to Prospective Customers	Customers can view other people's activities, shop and see the latest trends.	Customers can instant message pictures, videos and transfer money through this application.	Customers can search the web and utilize images on a focused scale.
Core Competencies	Instagram	It specializes in	It allows the users

	specializes in advertising and picture sharing.	video/picture sharing	to discover and share creative ideas.
Overall Impressions in the Market	It is considered an industry leader in social media.	It focuses on messaging through picture sharing.	It is said to be a pioneer in web searching.

Table 4.7: Comparison between various social media platforms

Following is the comparison of SWOT between competitors:

Characteristics	Instagram	Snapchat	Pinterest
Strength	Multiple filters enhance and customize pictures.	Constant refreshing of pictures is done.	Virtual bulletins are developed, and it is user-friendly.
Weaknesses	All interactions are performed online.	Lack of discreteness and pictures are only available only for a small amount of time.	Susceptible to spam pictures and is based on user uploads.
Opportunities	Advertisement and technology development opportunities	Advertising and product enhancement opportunities	Advertisement and developing business through social media networking
Threats	Issues with respect to photo rights	There has been a lot of negative publicity	Bookmarking features are being copied by other websites

Table 4.8: Comparison of SWOT between Competitors

Market analysis: It forms an important tool to analyze Instagram, Pinterest and Snapdeal.

Inference drawn: On the basis of analysis with its competitors, Snapdeal and Pinterest, the key improvements Instagram made included the following:

- Instagram allowed people to tweak their photos to their liking
- Similar to Snapchat, Instagram added the feature of sharing a “story”,

along with the addition of face filters for 24 hours, until it disappears.

- Instagram further enhanced its feature through direct messaging, a feature of Facebook where privacy concerns were easily addressed through its private/public feature.
- Despite starting off as a mobile photo-sharing application, Instagram moved into an advertising arena for all companies in social media.
- A major issue identified was about the photo copyrights as people would steal photos/ideas and refer to them as their own.
- Multi-device operability was a challenge: Running the application on multiple devices was identified as its limitation.

Case study on Customer churning analytics

Churning indicates the number of customers leaving the organization or company and procuring products of other companies. In other words, it helps in determining the trend of customer attrition.

Goal of the analysis

- **Churn analysis**

Churn analytics provides insights into the following:

- The reasons for the customers churning out, i.e., the various parameters for customer dissatisfaction
- The customers expected to churn in the coming months
- The various methods that need to be employed to reduce the churns

The churn rate is used to further determine the trend or pattern of the churn and is given by:

$$\text{Churn Rate} = \frac{\text{No. of customers lost over a period of time} \times 100}{\text{Total number of customers at the beginning of that time period}}$$

All this analysis is possible by applying descriptive, diagnostic and predictive analytics. This analytics helps in determining the efforts to be taken towards **customer retention**.

- **Identification of cause for churn**

By tracking the movements of the customers, their behavior and actions, the reason behind their dissatisfaction and canceling a subscription or use of a product can be identified. This analysis can be performed through a number of feedback calls, follow-up surveys and questionnaires. Once proper feedback is obtained, necessary actions can be undertaken to retain the maximum number of customers and also improve the systems.

- **Need for Churn Analysis**

Churn analysis is said to be important as certain processes can be determined by identifying the reasons for the churn, which customers are churning, and which customers are expected to churn next. These processes can further help by:

- Retaining customers who would aid in increasing the profits
- Providing customer loyalty
- Reducing costs as acquiring new customers is said to be more expensive than retaining existing ones
- Developing target marketing campaigns and reducing customer attrition based on the demographics and behavior of the customers
- Building a brand to get repeat customers and referrals, uncover customers' pain points and lead to improved customer experience

Reasons for customer churn in B2B organization

The major reasons for customer churning are as follows:

- The overall customer experience is lacking
- Customers do not get the value they expect from the product
- Absence of onboarding process to help users start with the offerings
- Product is expensive as compared to competitors' products
- UI/UX is not interactive and immersive
- A number of bugs and glitches
- The site takes more processing time and is slow

Case study for a Telecom Company

It is a known fact that there has always been a lot of churning in the telecom

industry. For every 100 people joining as customers, 30-35% opt out in a year. Therefore, a telecommunication company need to capture new customers and prevent contract termination. This termination is also known as churn or attrition. Reasons identified as the causes of churn include better price offers, more lucrative packages and bad service experiences.

So, there is a need for the companies to identify customers who would probably opt out and devise strategies or prediction models to prevent them from doing so. Churn analytics enables the prediction of customer churn and identifies the underlying reasons for the churn. Churn metrics could be used to determine the percentage of customers who cancelled a product or service within a given period.

The Business Challenge

Telecom companies are facing extreme customer attrition and need a lot of funds to acquire a new customer. As soon as a customer leaves the company, future revenue from the customer as well as the resources spent to acquire this customer are lost. So, there is a need to prevent this across the cross-section of the organization. To do this, companies need to do the following:

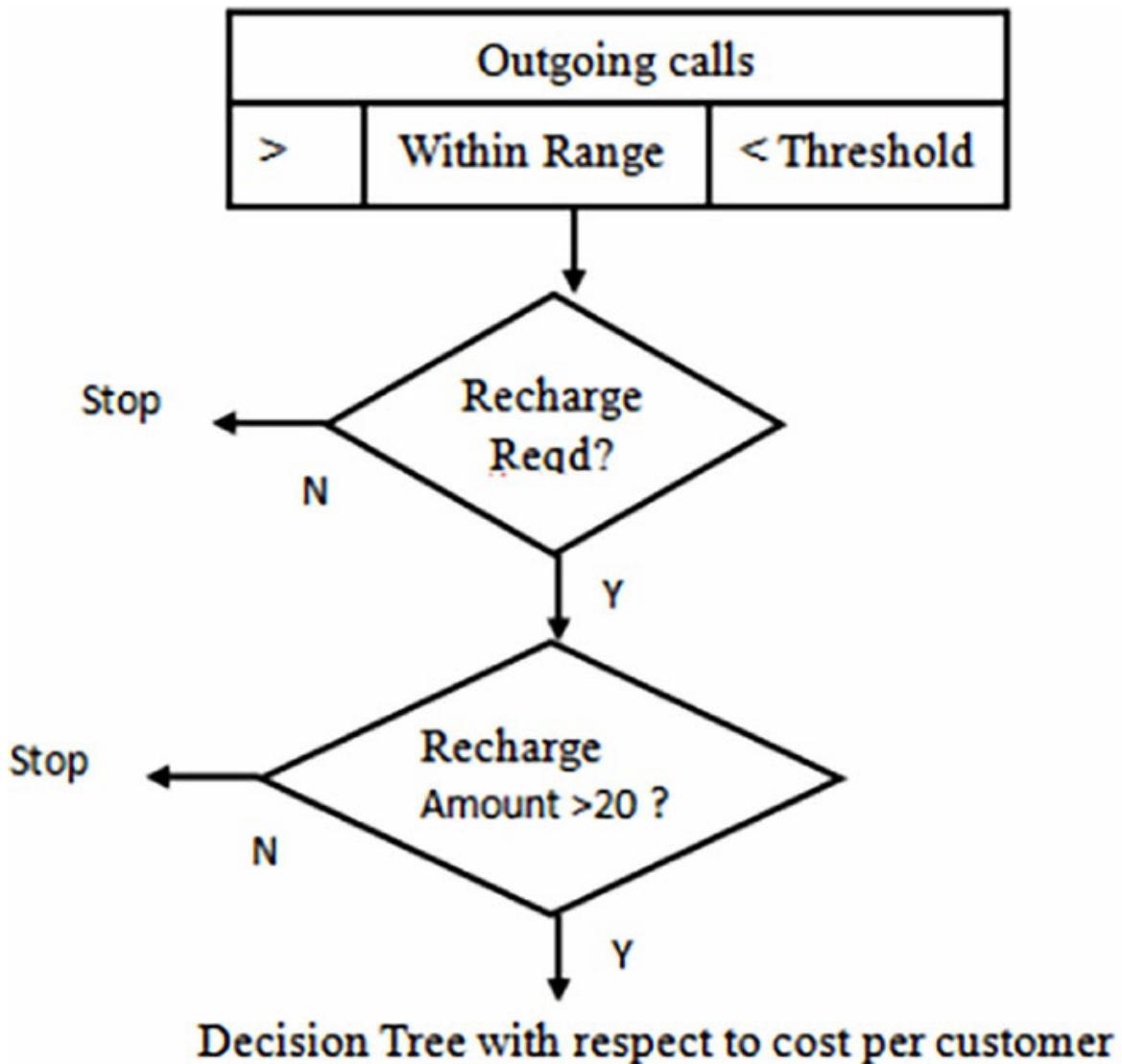
- Identify customers who are likely to churn and deploy interventions to prevent the churning
- Formulate plans and develop strategies for customer retention
- Cater to provide services to customers and improve profits

The approach

Based on customer behavior, demographics and network behavior, a number of classification and hybrid models could be applied to predict the churn with increasing accuracy. Mechanisms such as discounts and special offers need to be devised to retain their customers. In order to perform this prediction, key data parameters that affect customer churn needs to be identified. The various data parameters that can be considered include the following:

- **Satisfaction data:** This data could consist of information about communication, faults, mechanism of installation and so on. It could help determine the impact of different levels of satisfaction data on the overall churn, resulting in exploratory analysis.
- **Service Level data:** Based on service level data like billing, delivery, assurance, customer satisfaction data and complaints, a predictive churn

model can be built, as shown in [Figure 4.19](#)



[Figure 4.19: Decision Tree churn prediction model](#)

This analysis can help determine the factors affecting the churn of customers.

Once the key data parameters are identified, a basic machine learning pipeline is built, and its performance is determined. The best model suitable for a specific telecom company can then be determined by comparing the different model types. Let's look at the approach that could be utilized.

Step 1: Problem Definition

Based on the key challenge, it is identified that there is a need to predict whether a customer would churn. To determine that, machine learning models need to be trained. The training of a model can be performed on 80%

of the data available, while the remaining 20% can be utilized for testing the trained models. This threshold of 80:20 can be made dynamic to 70:30, depending on the application requirement. Further, the testing can be used to assess the prediction with regard to churning or not churning.

Step 2: Data Collection

As mentioned, different types of data and its corresponding dataset needs to be formed. Its use case pipeline using the Panda and NumPy for data handling and processing needs to be further developed. Further, Matplotlib needs to be used for visualization, as explained in detail in [Chapter 2, Python Basics for ML](#) and [Chapter 3, “An Overview of ML Algorithms”](#).

Step 3: Exploratory Data Analysis (EDA)

In order to further explore the data, the concepts behind data structures and the domain knowledge need to be understood, initial preprocessing and cleaning of the data has to be performed, and the topic of investigation must be understood. Further, the various patterns and inconsistencies in the data, such as skewness, outliers and missing values need to be determined using standard Panda functions. This could lead to building and validating the hypotheses. To add to this, the unique values for every feature needs to be understood in this step.

The features can also be clustered into different categories through classification labels, such as the following:

Churning Label

- **Churn, phone service, multiple lines:** Various analysis in terms of whether a customer has churned, has a phone or not are determined. The answer to this classification label would be in a Yes or No format.
- Customer services booked
- **Phone Service:** This indicates whether a customer has a phone service (Yes, No)
- **Type of internet service provider and backup facility:** It indicates the service provided for individual customers, such as DSL, Fiber optic or none. It also indicates whether the customer has opted for online backup, resulting in a yes, no or no service provider.
- **Streaming services:** This indicates whether the customer has a facility of streaming TV, movies and other related operations. This again results

in a Yes, No or No internet service.

Customer account information

It deals with the contract term for which the customers have stayed with the company, billing, payment method and charges to the customers. This provides quantitative information like the number of months, years, credit card payment, bank transfer and so on.

Customers demographic information

This information provides the customers' personal details in terms of their customerID, gender and whether they have dependents.

Building the hypothesis

Hypotheses can then be made with respect to the following:

- The customer would be less likely to churn if the contract duration is longer. The longer the contract duration, the less likely it is that the customer will churn. This is because they are less frequently confronted with the termination or prolongation decision.
- Customers are willing to cancel simple contracts with few associated product components quicker and more often than complex product bundles.
- In order to keep the services running for their families, the customers might churn less.

The major drivers of churn are tenure, contract duration terms and number of additional services.

Feature extraction

Based on the data types and its values, the features could be categorized. This can be done through label encoding into categorical data such as gender, phone service, paperless billing and so on. The outputs of these values is either a yes or no value. These can be transformed to binary integers for further analysis. For variables taking more than two values, such as online security, online backup and so on, payment method, One-hot encoding can be implemented. Here, for each value, a new variable can be created with a binary integer indicating whether this value had occurred in the data entry earlier (1 or 0). Values of numerical features are rescaled between a range of

0 and 1 through the min-max scaling method.

With respect to the preceding example of customer churning, it can be said that customers who churn have a lower tenure of around 10 months with higher monthly charges. Churning is said to be maximum for senior citizens and monthly contract customers.

Step 4: Feature Engineering

In this step, a new feature is generated from the existing ones and a correlation analysis is performed. It could be observed that customers having one additional service could have a higher churn rate as compared to customers with more services.

Step 5: Train/Test Split

The data set gathered can now be divided into training and testing data. This could then be applied for the conduction of model training and testing. Depending on the threshold identified, the information can be split into 80:20 or 70:30, that is, the split between training and testing data. Y axis, that is, the dependent class, can represent the churn, while the other features can be treated as the independent variables, that is, the X axis.

Step 6: Model Evaluation Metrics

Various metrics could be used to evaluate the performance of the chosen models:

- **Feature weights:** This metric could indicate the top features used by the model to generate the predictions.
- **Confusion matrix:** It could indicate a grid of true and false predictions compared to the actual values.
- **Accuracy score:** This could represent the overall accuracy of the model.
- **ROC Curve:** This could specify the **true positive rate (TPR)** along with **false positive rate (FPR)** for different thresholds of class predictions. This would, In turn, help determine the churn prediction.
- **Precision-Recall-Curve:** This could compare the **false positive rate (FPR)** and **false negative rate (FNR)** for different thresholds of class predictions.

Step 7: Selection, Training, Prediction and Assessment of the Model

Machine learning models like K-nearest neighbors, logistic regression, random forest and support vector machines can be tested. Their performances can further be measured, and the models can then be optimized by tuning their hyperparameters. It can be observed that K-Nearest Neighbors could be fast, simple and instance-based as compared to Logistic Regression, which would be a linear-based model. Random Forest could be identified to be slower yet an accurate ensemble model based. Support Vector Machines can be used for non-linear operations. Various machine learning models can be deployed to predict churn. These could include the following:

- **Classification model or supervised learning:** It helps identify whether a customer could churn.
- **Regression analysis:** Separate models using techniques like regression analysis and predictive dashboards can be built to predict customers churning from the business.

Applying these machine learning models could result in developing targeted marketing and reduction in acquisition costs.

Step 8: Hyperparameter Tuning/Model Improvement

The concept of cross-validation is used during hyperparameter tuning to address the Biassing by bias arising due to split of the data in the train-test-split part. Hyperparameter tuning can be incorporated through grid search and randomized search.

The following [*figure 4.20*](#) represents the process of developing the system. It indicates the way of building and deploying the models to determine the highest and lowest customer churn information. As represented in figure A in the following image, building the model consists of performing historical data analysis by identifying significant variables and developing the model. [*Figure 4.20*](#) represents the deployment mechanism that comprises capturing the live feed of historical data and developing the propensity model. This could be used to generate the score that differentiates between the high and low likelihood of customer churn.

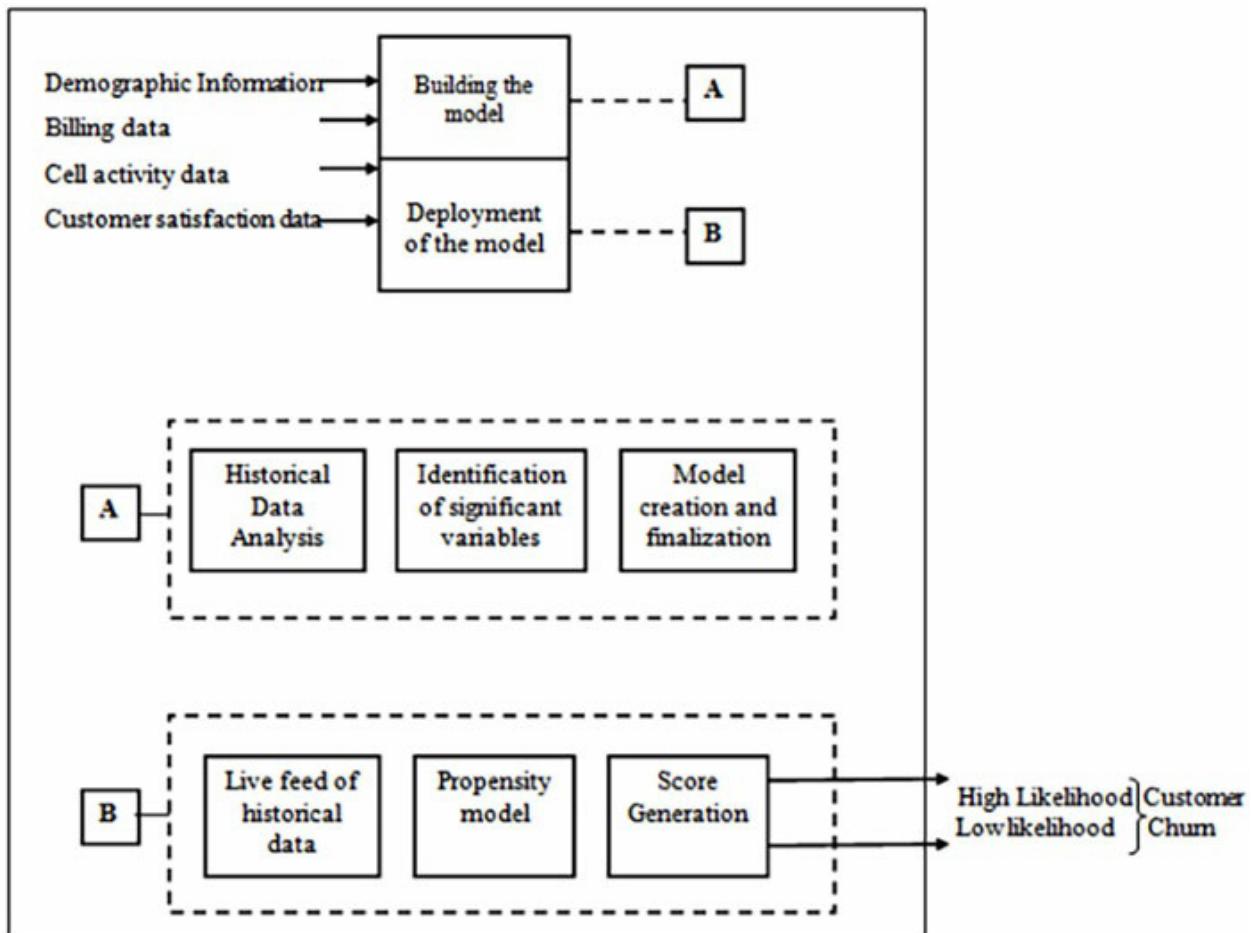


Figure 4.20: System development

Impact analysis

1. Churn analysis could represent the reason for churn to be the likelihood of delay in delivery.
2. The impact of value-added services on churn can be one of the major reasons for churn.

Case study on learning analytics in education systems

As education data is increasing in leaps and bounds, there is a need to have improved data storage, advances in computing resources, algorithms and analytics tools. These advances are required to understand the learning environment, make predictions, optimize the learning processes, make informed and better educational decisions in an organization. This is possible through the concept of learning analytics. Thus, learning analytics is used to

assess and analyze information about the learners and the learning environment.

In order to integrate the learning analytics system into educational organizations, there is a need to develop actionable frameworks and adoption models. However, these models may vary across different countries and organizations. The major stakeholders involve those who communicate, follow and promote learning analytics through secure open communication channels.

Organizational benefits from learning analytics

Various benefits of utilizing learning analytics include the following:

- Identification of students at risk and students who could be successful in their learning.
- Monitoring and improving organizational capabilities
- Development of educational policies and resource allocations across schools. This can be done in order to enhance and optimize the education processes, reduce dropout rates, increase retention, success rates and predict school performance
- **Data Gathering**

The gathered data can be utilized to determine the purpose or use of the data. This data can be broadly classified into the following groups:

- **Summative and descriptive:** This provides detailed insights after completion of a learning phase, which is also known as the study period.
- **Formative and (near) real-time:** This learning captures data from the ongoing information for improving processes through direct interventions.
- **Predictive and prescriptive:** This forecasts the probability of outcomes in order to plan for future interventions, strategies and actions.

Challenges faced

Let's look at the challenges faced.

Adoption

- One of the challenges faced in adopting learning analytics was to determine the need for adopting the learning analytics mechanism either completely or in a phased-up scaling manner.
- Due to involvement of various stakeholders at different levels in the organization, applying learning analytics leads to complexity in developing the processes.
- Developing the preferences for functionalities, intervention strategies, and perceptions of privacy needed tools for development. So, committees needed to be formed for governance that would focus on adopting learning analytics and a group to focus on the private and ethical use of educational data.

Teachers' diagnostic support system

In order to provide diagnostic support, information with respect to the students is to be gathered:

- **Students' personal characteristics:** This could include domain-specific knowledge, competencies, and emotional-motivational characteristics.
- **Descriptions of instructional characteristics:** This could indicate the learning contents and its characteristics.
- Information regarding students' interest in the subject and their knowledge about the topic can also be gathered.

Approach

The focus on development can be in phases: before the session begins, during the session and after the session has ended, as shown in [*Figure 4.21*](#). The development can also be in a client-server model or a mobile app development.

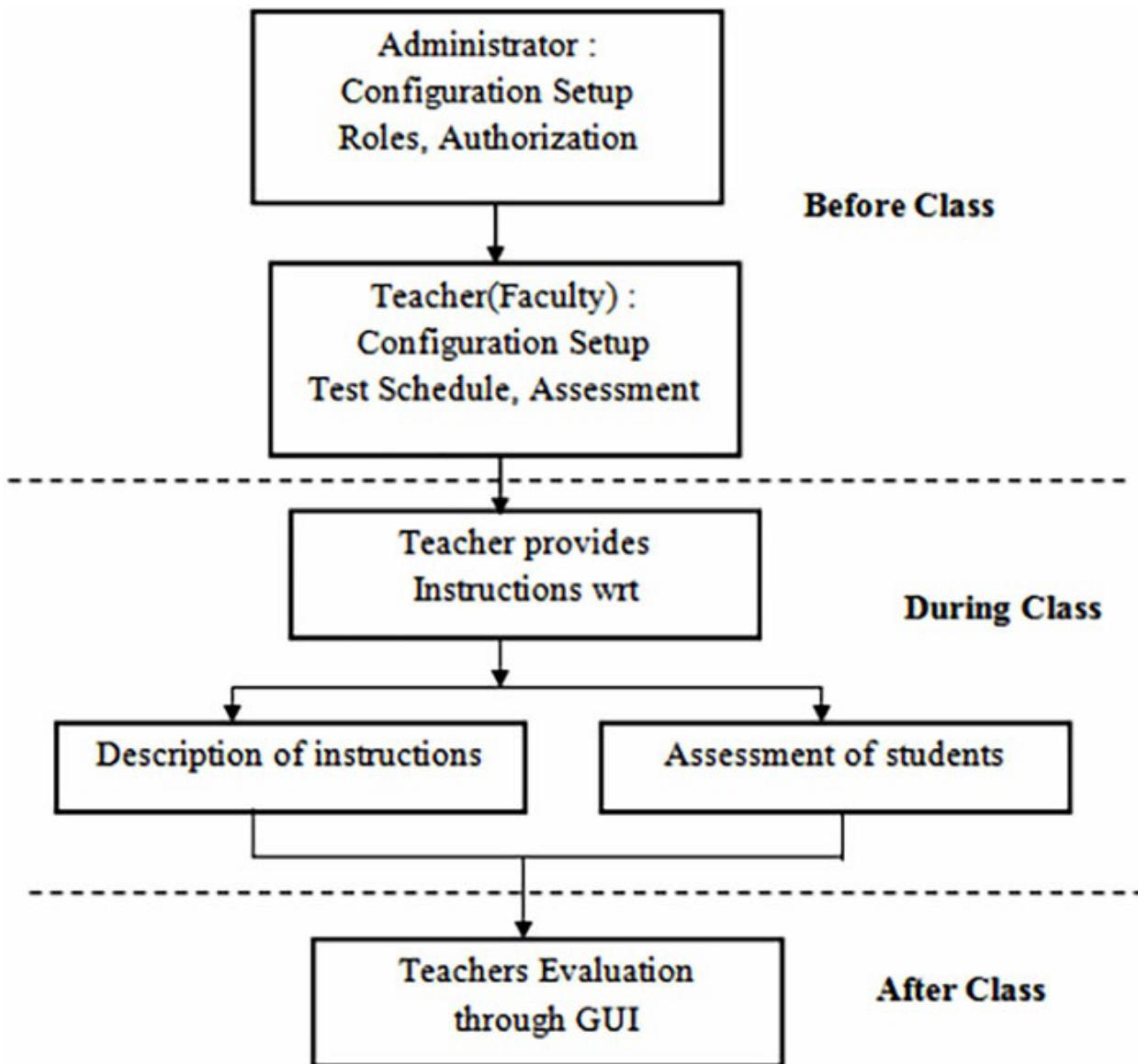


Figure 4.21: Workflow for evaluation

[Table 4.9](#) indicates the information and the source from where the information can be obtained:

Sr.no	Focus Area	Information Description	Information Source
1.	Student	Personal characteristics of students	Student
		Assessment of students	
2.	Teacher	Instruction characteristics	Teacher

Table 4.9: Information Source

Learning analytics can also be used for examinations. Based on the evidences collected, we can determine the success rate, identify the students at risk of failing, the number of enrolled students, drop-outs, average risk possibility, average students age, gender distribution and so on. The process that could be employed is shown in [Figure 4.22](#):

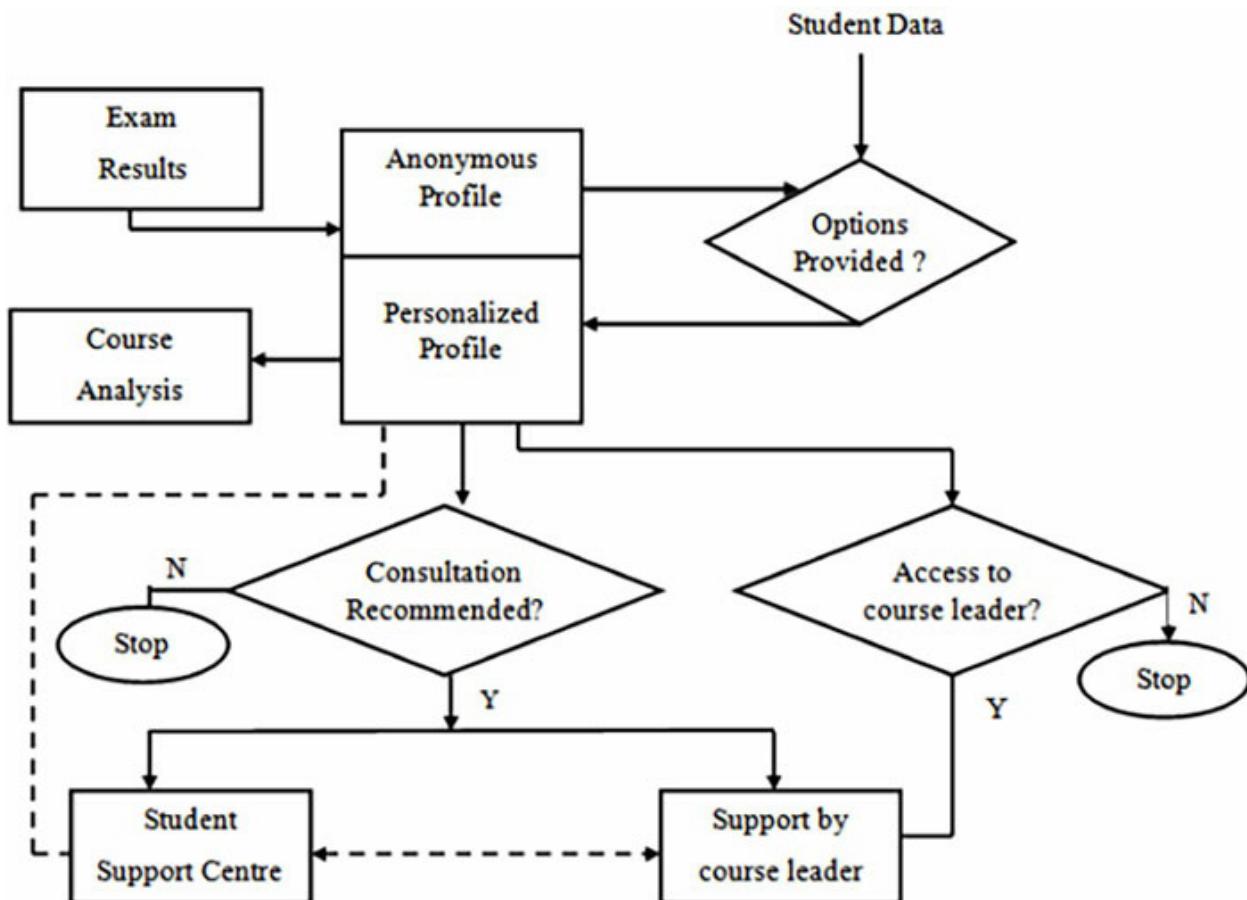


Figure 4.22: Learning analytics process

Based on the processes followed, insights with respect to credits received by students, staff retention, student improvement and student satisfaction could be determined. Various test case scenarios could be considered while employing machine learning algorithms like the following:

- **At the macro-level:** Learning analytics insights can help determine the staff target retention initiatives.
- **At the meso-level:** Machine learning algorithms can be incorporated to improve teaching-learning practices.

- **At the micro-level:** Algorithms that focus on student improvements, determining the students in the failure zone can be determined. Further algorithms can be developed for performing corrective action for individual students.

Utilizing various models, standards visualizations indicators and design guidelines would make learning analytics effective. Also, implementation of various machine learning algorithms, learning management system(LMS) and student information system (SIS) can help develop insights for a better education learning system. Thus, the adoption of learning analytics can be used for the following:

- To develop academic curriculum in tune with job market demand
- To prepare students for their job careers
- To provide flexible learning opportunities through adaptive and on-demand learning approaches
- To support student needs and enhance the learning environment for the students
- To enable flexible teaching and learning methods in the most economical way

Other case studies

Utilization of Predictive Analytics by Netflix: Data collection is critical for a company like Netflix. Based on the behavior and past viewing patterns, Netflix gathers customers' information and uses it to make predictions based on customer preferences.

Conclusion

This chapter focused on the various case studies of the applicability of machine learning. These included recommendation systems that alleviate the problem of information overload and provide access to products and services that are not easily available to users on the system. This was followed by applying machine learning algorithms to the text and image elements of multimedia systems. Further various case studies utilizing machine learning algorithms were explored.

The next chapter elaborates on optimization in ML algorithms and sheds light on how to use various heuristic optimization algorithms to decide the right parameter values for ML algorithms.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 5

Optimization in ML Algorithm

Introduction

Machine learning projects are incomplete without the improvements, specifically when it comes to training of algorithms. Researchers have tried hard to reduce the cost involved in training of algorithms. This is where optimization comes into the picture. In [Chapter 4, “Case studies and Projects in Machine Learning”](#), we have seen projects in the various domains. There is a scope for improvement in these projects using various hybrid algorithms and optimization techniques. Optimization is important when money and time are concerned. The goal of optimization is to find the best element, path or techniques to improve the efficiency of algorithms by decreasing the training time. This chapter introduces the concept of optimization and explains its importance in Machine Learning-based projects. The chapter starts with basic optimization techniques and goes on to discuss metaheuristic approaches. The chapter also throws light on Python libraries available for optimization.

Structure

- Optimization: Need of ML Projects
 - Types of optimization techniques
- Basic Optimization techniques
- Meta-heuristic Approaches to Optimization
 - Types of Metaheuristic algorithms
- Improvisation of ML algorithms by optimizing the learning parameters
 - Case study 1: Metaheuristic Optimization Techniques in Healthcare

- Case Study 2: Genetic Algorithm (GA) in Batch Production
- Optimization using Python

Objectives

In this chapter, you will understand the importance of optimization and learn the basic optimization techniques. You will be introduced to various categories of optimization techniques and will get to know the meta-heuristic approaches of optimization. You will also learn how optimization improves the performance of ML projects and get familiar with various Python libraries for optimization.

Optimization – Need of ML projects

Optimization is the selection of the best element using some criterion from a set of available alternatives. The goal of optimization is to provide near perfect, effective and all possible alternatives. Maximizing or minimizing some function related to application or features of the application is the process of optimization. For example, minimizing the cost function or the mean square error is the goal of optimization in typical ML algorithms.

Optimization is important when money and time are concerned. Machine learning often uses optimization algorithms to enhance algorithms' learning performance. Reducing the cost function is the main aim of any optimization technique in machine learning. Optimization can include optimization of weights, activation nodes, network architecture and learning parameters. Training optimization improves the generalization ability of any model. In complex models, tuning of hyperparameters or finding the most suitable values for hyperparameters is the goal of optimization in Machine Learning.

Types of optimization techniques

Broadly optimization techniques can be divided into two approaches:

Conventional Approach

Conventional optimization approaches like gradient descent and back propagation are very popular and widely used. They can be categorized as

first-order or second-order optimization techniques. The major advantage of using the conventional approaches is that they are very easy to use and implement. These are the direct methods that do not use any derivative information of the objective function. Here, only objective function values are used to guide the process.

A variation in these techniques is also used to improve the performance of machine learning algorithms like momentum-based gradient descent and Stochastic gradient descent depending on the application where these machine learning algorithms are used.

Metaheuristic Approach

A metaheuristic algorithm is evolved to find or generate a good solution to an optimization problem. Metaheuristics works when one has limited computation capacity and incomplete information available. The strength of metaheuristic algorithms lies in two techniques: exploitation and exploration. Exploitation is nothing but searching unknown and new areas in the search space where the current good solution is found. Exploration is nothing but taking advantage of the solutions that are already discovered. A metaheuristic is a procedure that amalgamates these two techniques to find a good solution.

The overall performance of the metaheuristic algorithms depends on the optimum use of these two techniques. The efficiency also depends on how finely these two techniques are balanced. Metaheuristic algorithms work on local and global levels to find a solution. Thus, imbalanced exploration and exploitation will cause a problem. Less exploration and more exploitation will cause trouble in finding the local and global optimum. Also, more exploration and less exploitation will create problems in convergence. Thus, the system will slow down, hampering its overall performance of the system. The epitome of the metaheuristic algorithms is how this balance between the two techniques is achieved.

There are wide varieties of metaheuristic algorithms available, so clear differentiation between them may not be possible. On a broad scale, these algorithms can be divided into two groups:

- **Single solution-based algorithms**

Single solution-based algorithms concentrate on improving a single solution. Simulated Annealing and Tabu search are examples of single

solution-based algorithms. These algorithms concentrate on local search by exploiting the local search space.

- **Population-based algorithms**

As the name suggests, population-based algorithms work on multiple solutions and focus on finding the best among them using heuristics. These algorithms are strong in global search.

These algorithms can be further divided into the following:

- **Evolutionary algorithms:** Genetic algorithms are the most popular evolutionary algorithms. Other than this, Genetic Programming, Evolutionary Programming, and Differential Evolution are some of the popular evolutionary algorithms. Evolutionary algorithms use an evolutionary framework and genetic operators to generate new solutions. Genetic operators are mutation, crossover, selection and so on. These operators are used repeatedly to evolve new and optimal solutions.

These algorithms can be applied to different nonlinear problems. The stopping criteria in evolutionary algorithms are indicated by the fitness function, which, in turn, determines the quality of the solution. Defining a proper fitness function is the key to evolutionary algorithms.

- **Swarm Intelligence:** Particle Swarm Optimization, Artificial bee Colony algorithms, Grey wolf optimization, and bird flocking are some of the widely used examples of swarm intelligence algorithms. These algorithms are inspired by the behavior of various decentralized but self-organized systems. In short, these are nature inspired algorithms. For example, in Particle Swarm Optimization, the behavior of a flock of birds for getting their food is observed. The heuristics are generated by observing this behavior.

In these algorithms, the food indicates the problem solution and the methods these birds use to get food are mimicked in real time to create a global search. In ant colony optimization, the behavior of the ants to search food is observed. Ants in their natural habitat use the shortest path to get to a sample of food. They establish good communication between them to find these shortest paths and

traverse routes in a disrupted manner. This behavior is mimicked in Ant colony optimization algorithms. There are many algorithms that are inspired by the nature and behavior of various animals and birds.

- **Hybrid algorithms**

Conventional algorithms are good at local search and metaheuristic algorithms are good at global search. Hybrid algorithms make the most of both these strengths by combining conventional algorithms with metaheuristic algorithms.

The following [*Figure 5.1*](#) shows the brief classification of the optimization algorithms for easy understanding:

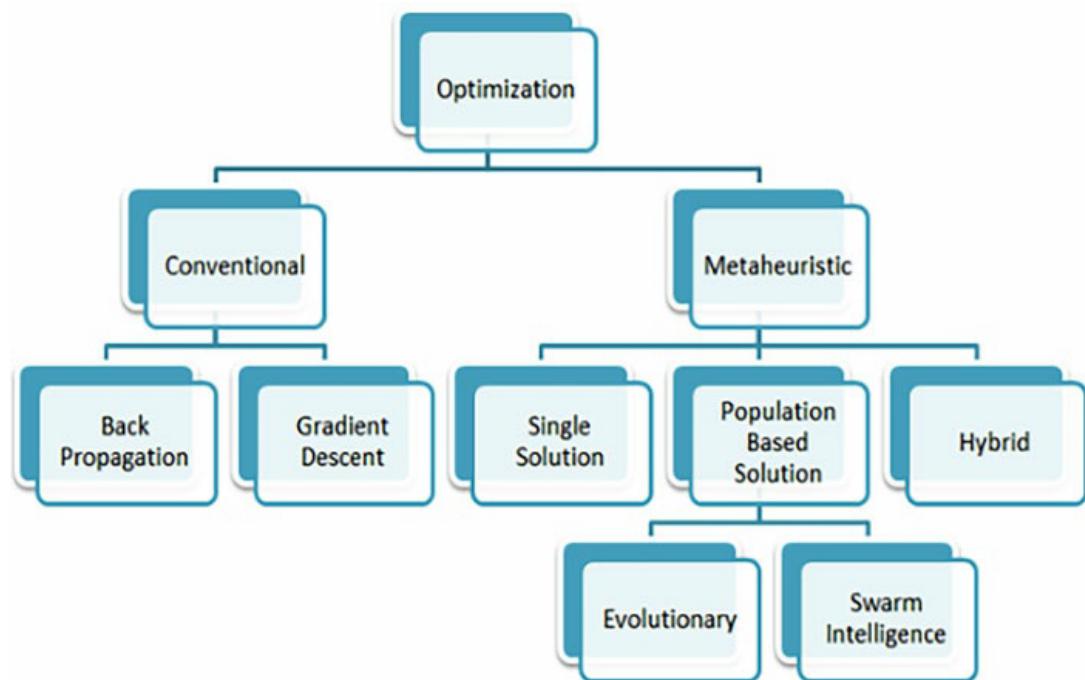


Figure 5.1: Taxonomy of Optimization

For example, Genetic Algorithms with Back Propagation are used to train Artificial Neural Networks giving a better approach to learning. Hybrid algorithms are good at search and converge fast. These algorithms are also called memetic algorithms. Under hybrid models, two different metaheuristic algorithms are also used to get an optimized solution.

The diagrammatic representation of taxonomy of optimization is shown in [*Figure 5.1*](#).

Basic Optimization Techniques

As discussed in an earlier section, there are two basic optimization techniques: backpropagation and gradient descent.

Backpropagation optimization

The backpropagation algorithm is used in a feedforward neural network with supervised training. The input from the input layer is passed through the activation functions present in the hidden layer. While passing through, the input is assigned weights. Now the output from the hidden layer becomes the input for the next layer, and so on. The final output is obtained through the output layer. This resultant output and the initial input are compared, and the error is calculated. This error is used to fine-tune the weights at that layer. This process is followed recursively from the output layer to the input layer. Thus, the weights are modified and adjusted; this process is followed till minimum error is achieved and the final output is computed.

In this entire process, the weights are adjusted. The adjustment of the threshold value is first obtained at the output layer. The two parameters, i.e., the calculated error in the output at the output layer and the learning rate parameter used in the adjustment calculation for weights at subsequent particular layer, are multiplied to get the adjusted value of the threshold. Thus, the backpropagation algorithm learns the correct classification from the training data. Then, it is tested on the test data to find out how well the network classifies the test data. Thus, an important consideration in applying backpropagation learning is how well the network generalizes.

Having said that, there are many problems with back propagation learning:

- **Local minima:** Backpropagation always works toward adjusting the weights of the network so that the error reduces. But while achieving this, the error may start increasing. Thus, the algorithm contradicts itself.
- **Network paralysis:** Network paralysis is the phenomenon where the units are forced to operate at extreme values. These extreme values form a region where the derivative of the activation function is very small. Very large values of weights during training cause units to work at extreme values.

- **Slow convergence:** The multilayer neural network may require many iterations to get optimal values of the weights. Also, if the network gets stuck in local minima, convergence rate still slows down.

Gradient descent optimization

Gradient descent is another optimization method used to find the values of functions' parameters that minimize a cost function. Here, gradient measures the change in the output of a function if the inputs change by a small value. The gradient descent is basic optimization used in all ML projects. There are several variations of gradient descent: momentum-based gradient descent, stochastic gradient descent, Nesterov accelerated gradient descent and so on. But again, it also has a few disadvantages:

- **Slow processing:** The process of gradient descent is very slow as it requires the computation of cost functions based on several parameters.
- **Scaling problem:** The number of iterations to be taken to minimize the cost function depends on the scale of the problem as the direction is not well defined.
- **Local minima:** Gradient descent operation depends on a learning parameter or step. A small step may increase the convergence time and a large step may skip the local minima. Fine-tuning the learning parameter is challenging.

Conventional approach uses squared-based error as cost function for optimization, which may not be suitable for every classification problem. In most classification problems, accuracy and miss-classification rate are used as cost functions to determine whether it is a good or bad classifier.

Metaheuristic approaches to optimization

One may ask what's the difference between heuristic and metaheuristic techniques? Heuristic techniques are problem-based and are specific to a problem, whereas metaheuristic-based techniques are independent of any specific problem and can be applied in large, general domain of problems. Metaheuristic algorithms work fine when the number of solutions is very large. As these are generalized algorithms, they are suitable for various applications. These algorithms are designed to address complex problems.

Nonlinear and non-differentiable problems can also be handled very effectively by these algorithms.

A metaheuristic algorithm follows a step-by-step procedure to generate the solution. These steps are as follows:

1. **Initialization:** Use random values as a seed in population and initialize the population in the search domain.
2. **Evaluation:** Fitness of each one of the population is evaluated.
3. **Generation:** Generate a new population using evolutionary operations as suggested in each algorithm.
4. **Repetition:** Repeat the steps 2 and 3 until stopping criteria are satisfied.

Metaheuristic optimization works with an objective function that assigns a value to each solution in the space. The set of parameters are used to obtain the global maxima or minima of the objective function. The initial solution is derived using some heuristic, and then metaheuristics are used to improve this solution using an iterative process. The iterative process is terminated when the stopping criteria are met. The stopping criteria can be anything, as defined by the user like number of iterations, time completed and so on. The metaheuristic algorithms find a good solution in the existing solution space, but they do not guarantee the best solution.

The main advantage of using metaheuristic approach over the traditional approach is that their global optimal search is better. This property makes them usable in various problems to improve the solution. Metaheuristic algorithms are more adaptive and are intelligent in nature because they learn while they run. On the contrary, heuristic algorithms apply trial and error mechanism, which makes the search dependent on computational capacity.

Metaheuristic algorithms in optimization have its own advantages:

- Easy to understand
- Easy for implementation
- All-encompassing application domains: have applications in various domains as these algorithms are generative in nature
- Can be combined with other approaches making them widely usable
- Complex problems with large solutions can be solved faster

Disadvantages of these algorithms can be as follows:

- No mathematical foundation
- Convergence rate is very slow many times
- Does not provide the best solution
- Their performance is dependent on the fine-tuning of the parameters
- The solution that one gets initially may not be repeated under the same conditions
- It is difficult to decide which algorithm is best suited for which kind of application and under which circumstances. So, it's mostly trial and error for any application.

These optimization algorithms use the exploration and exploitation methods to devise new optimal solutions. Term explorations indicates a process of searching large area to find good, feasible solutions. The time required to search large or wide search space will downgrade the performance of the algorithm; this is where exploitation comes to the rescue. Exploitation restricts the search to a small area so that solutions can be refined. Exploration is introduced at earlier stages and exploitation is introduced at the later stages in the process.

Exploration and exploitation are two important steps in any optimization algorithm. Exploration always ensures that the proper and promising regions of the search space are invaded, whereas exploitation always ensures that optimal solutions are found within the given region. To reach the optimal solution, it is important to fine-tune these components carefully. There is always a tradeoff between these components, so proper balancing is required, which is difficult to achieve. Mostly, the component values are dependent on applications under consideration.

The metaheuristic nature of algorithms helps achieve this balance. Every optimization algorithm has its own strategy for exploration and exploitation, which makes it unique in every aspect. Metaheuristic algorithms are effective for real-life engineering problems but one thing to keep in mind is that due to their nature, they cannot be generic for all kinds of applications. They may provide the best results for one application but not perform well for another. The performance of one optimizer to solve a set of problems does not guarantee to solve all optimization problems with different natures.

Types of metaheuristic algorithms

As indicated earlier, metaheuristic algorithms are broadly classified into single solution-based, population-based and hybrid models. Though there are large number of optimization algorithms, this chapter focuses on a few promising and widely used algorithms.

Single solution-based algorithms

These algorithms are based on applying the heuristics on the single solution and finding the better solution. They work toward the betterment of the solution and use the exploitation method substantially. There are many popular algorithms available; let's look at a few of them in brief:

- Simulated Annealing
- Tabu Search
- Variable Neighborhood search

Simulated annealing

This method is inspired by the slow cooling of metals. Cooling of metals is achieved by reducing the atomic movements, which, in turn, reduces the density of lattice defects till the lowest energy state is reached and the metal gets the desired structure. In algorithms, at every annealing temperature, a new solution is provided to the problem under consideration. Here, the temperature is the most important parameter to get an optimized solution.

The steps of the algorithm are as follows:

1. Start with the initial solution S_0 and set the initial temperature as t_0 .
2. A temperature reduction function, popularly called alpha, is required to be set up, which can be based on Linear Reduction rule, Geometric Reduction rule or Slow Decrease rule. Each reduction rule will decrease the temperature at a different rate.
3. Here, the looping starts, and the next step is repeated until the termination condition is met. The termination condition is based on the value of low temperature with satisfactory performance.
4. From the solutions generated, pick up one solution and calculate the difference between the costs of the old solution and the new solution.

The solutions are generated by altering the parameter values.

5. If the new solution cost is better, then accept that solution. After a certain amount of looping, the performance stabilizes and the final solution is solicited.

Simulated Annealing is a very effective and simple to implement optimization algorithm and is used to solve many classic problems, like Travelling Salesman Problem and scheduling problems. The main advantage is that it sticks with local minima.

Tabu Search

Tabu search is a stochastic optimization method that is effective because of local search methods and is generally used for mathematical optimizations. The strong point of Tabu search is that it uses memory effectively to store and track the previous solutions. This makes local search very effective. Tabu search explores good solutions so that intelligent search mechanisms can be employed to explore new potential regions. It uses adaptive memory, which helps in creating flexible and effective search strategy.

The algorithm steps are as follows:

1. Start with an initial solution S_0 , which is any feasible solution to start with.
2. Find the next neighboring solutions and start populating the Tabu list, which is the memory that stores either moves to solutions or complete solutions generated by the algorithm. The solutions in the Tabu list satisfy the aspiration criteria. Aspiration criteria is optional and decides the quality of the solution. If the current solution is better, it is stored in Tabu List and remains there for a tenure called Tabu Tenure. The aspiration criteria ensures that stagnant solutions are not created.
3. The new solution is compared with the previous solution and is included in the list if it is found to be better.
4. The process is repeated till the termination criteria is met. The diversification in the solution space is ensured by giving a chance to potentially bad solutions as well. Diversification strategies are made so that newer regions are explored and moves that are not part of Tabu List are explored and given a chance.

Tabu search is applied to both discrete and continuous mathematical optimization problems, and it is also used for solving problems in chemical and mechanical streams.

Variable Neighborhood Search

Variable neighborhood search is used for solving combinatorial optimization problems and global optimization problems. The exploration is very strong in the neighborhood search, and a move to a new solution is made only if the improvement in current solution is seen. It uses effective local search methods in neighborhoods to get new solutions till local optima.

The variable search algorithms consider that local minima provide some information about the global one. Thus, an organized search considering local optima is employed and requires very few parameters to be tuned in.

Population-based algorithms

Population-based algorithms are more popular because they work on multiple solutions. These algorithms are inspired by nature, natural activities, behavior, biology, physics and even human activities like teaching and learning. These algorithms use exploration method or global search method substantially. Broadly, they can be categorized as follows:

Evolutionary techniques: Evolutionary techniques are popular for training machine learning models. Evolutionary techniques use the biological concepts of natural selection, evolution and mutation. These techniques have proven beneficial for solving many real-life optimization problems effectively. The techniques are easy to implement and require only a few parameters to train the model. A few popular evolutionary techniques are as follows:

- Genetic algorithm
- Differential evolution
- Evolutionary programming

Genetic Algorithms: Genetic algorithms are popular optimization algorithm proposed by Holland. They are based on genetic, mutation and selection mechanisms. The generalized algorithm can be explained as follows:

1. Generate initial random population.

2. Calculate the fitness of the generated population and select only the fittest.
3. Select the parents from this initial fit population.
4. Select the genetic operator, either crossover operator or mutation operator, to generate the children.
5. Repeat steps 2 to 5 until the stopping condition is met.

The stopping condition is generally based on number of iterations required to get the optimal solution. The major parameters in genetic algorithms are fitness function and genetic operator. The new solution is generated using a genetic operator, either crossover or mutation. The crossover can be one-point crossover, two-point crossover, multi-point crossover or uniform crossover. The basic mutation chooses one or more genes to randomly change, and the inversion operator randomly chooses two gene points to inverse the genes. The selection of mutation operator depends on the encoding used, which can be binary encoding, natural numbers, real number, and so on. The selection operator also plays an important role in generation of new solution, and there are various election operators that can be used, such as roulette wheel selection, stochastic universal sampling, local selection and tournament selection.

The genetic algorithm is the simplest to implement and is an effective optimization algorithm applied to various domains of engineering problems.

Differential Evolution

Nonlinear and non-differential continuous space function optimizations can be solved effectively using differential evolution.

The differential evolution algorithm starts with an initial population of candidate solutions, which are iteratively improved by introducing the mutations into the population and then crossover functions. The fittest candidate solutions that yield a lower objective function value are retained then. The main advantages of differential evolution are that control parameters are few, and it can handle nonlinear and non-differentiable multi-dimensional objective functions. The difference between genetic algorithm and differential evolution is that the solution is only updated when the new solution is better than the old one.

The algorithm steps are as follows:

1. Generate initial random population.
2. Calculate the fitness of the generated population and select only the fittest.
3. Execute the mutation operation and crossover operation and then select the children using selection operation.
4. Add the optimized solutions to the population.
5. Repeat steps 2 to 5 until the stopping condition is reached.

The fitness function here controls the size of population and maintains the fittest values only.

Evolutionary Programming

Evolutionary programming is a stochastic optimization strategy that helps solve numerical and combinatorial optimization problems. The convergence rate of evolutionary programming is rarely slow, and the main variation operator here is mutation.

The basic steps in evolutionary programming are as follows:

1. Generate the initial random population. The speed of optimization depends on the number of populations, so generating number of solutions in the population is important.
2. The generated population is tested for fitness, and only fit solutions are retained. Generally, a stochastic tournament determines the solutions to be retained in the population.
3. New children or offsprings are generated using mutation. The offsprings are mutated according to the distribution of mutation type, which ranges from minor to extreme. The mutation brings functional changes in the parents and is judged based on this change.
4. Steps 2 and 3 are repeated until the stopping criteria is met.

One thing to note here is that evolutionary programming does not use genetic or crossover operator to generate offspring.

Swarm Intelligence

Swarm Intelligence was introduced in 1989 and it uses the concepts of using knowledge of collective group to reach the optimized solution. The basic concept is that a group of objects (people, animals, insects and so on) are

given the same problem to solve. They use all resources to find the best solution for the problem. So, instead of using a single object's intelligence, the algorithms use the group's intelligence to find optimal solution. One thing to keep in mind is that all objects behave individually and try to find the optimal or best solution. Swarm Intelligence has gained popularity as it proved beneficial for large domain of problems. These algorithms are also nature-inspired algorithms. There are numerous algorithms introduced, and some are still being discovered. Let's look at a few popular algorithms that use Swarm Intelligence:

- Particle Swarm optimization
- Ant colony optimization
- Cuckoo Search Optimization

Particle Swarm Optimization: This is one of the most popular optimization techniques introduced in 1995. It is also a stochastic optimization technique that models the behavior of animals or insects to find the optimal solution. For example, birds look or hunt for food in flocks. Particle swarm optimization is a robust and efficient technique to solve various optimization problems.

Here, individual particles are part of solution space and always convey their positions to the neighboring particles. The goal is to find the best position in the search space. Every particle's position is adjusted based on velocity and difference between the current position, the optimal position found by particle itself and the best position found by neighbor. The iterations ensure that the focus of the swarm is more on the search space, which contains high-quality solutions.

It is observed that Particle Swarm optimization performance is best for a wide variety of real-world optimization problems.

The broad steps are as follows:

1. Generate the initial swarm particles randomly.
2. Repeat steps 3 to 5 till the termination criteria are met.
3. For every particle, choose between the current position of the particle and the best position found out by the particle. Let this be p_i .
4. Find the best position of the particle determined by the group of

swarms, that is, neighboring particles. Let this be p_g .

5. Update the velocity and position of the particles based on the values of p_i and p_g , as shown in the following equations:

$$v_i = v_i + c_1 * rand_1 * (p_i - x_i) + c_2 * rand_2 * (p_g - x_i)$$

$$x_i = x_i + v_i$$

Here, v represents the velocity and p represents the position. The i subscript represents the current particle and g represents the group best value.

6. Output the optimized results.

It has been observed that velocity values explode many times, so velocity clipping value can be determined and used to prevent the exploding of velocity.

Particle swarm optimization has been applied to many optimization problems, including the travelling salesman problem and task assignment. It has also been applied to many problems where multi-objective optimization and dynamic optimization are needed. It is effectively used to train the multilayer perceptron and was effective because of its fast convergence. This facilitates saving computation time.

Ant colony optimization

As the name suggests, this algorithm is inspired by the behavior of ants. Ants work systematically and in a disciplined way to search for food. They communicate with each other with the help of an organic compound called pheromone on the ground. When the ant goes for searching the food, they spread the pheromone and other ants follow that path based on how strong the pheromone is. Different ants choose different paths but the shortest path is preferred.

For example, suppose three ants create three paths. The fourth ant will follow any path out of three. The fourth ant will choose the path from where the first ant has reached early. Now when the ants move on these paths, they go on spreading the pheromone highlighting the shortest path. Repeatedly using the same path to reach the food marks more pheromones on the path. Thus, swarm intelligence is used, and the shortest path to reach food and return is found.

The general steps for Ant colony optimization are as follows:

1. Initially all ants generate the paths/ solutions.
2. Compare the paths found by each ant.
3. Based on the best solution/path found, update the value of pheromones and find the best solution. Construction of the best solution using the pheromones models is nothing but constructing parameterized probability distribution over the solution space.
4. The steps are repeated until the stopping criteria or the required number of iterations are complete.

Ant colony optimization algorithms is applied to a wide range of optimization problems and is one of the most preferred optimization algorithms because of its simple implementation and fast convergence rate. There are various variants of this algorithms, which are used over a wide variety of applications. This algorithm is widely applied in biometric and biomedical systems.

Cuckoo Search Optimization

The basic algorithm is explained as follows:

- Each cuckoo selects a nest arbitrarily and lays an egg in it.
- The best nests among the nests with increased quality of eggs are considered for the following generation.
- For a specified quantity of nests, a host bird finds a foreign egg with a probability of $[0, 1]$. Under such conditions, the host bird may either desert the nest or throw the egg and then build a new nest in other places.

The final condition can be approximated by substituting a fraction of the n host nests with novel ones. The fitness or quality B_i of a solution is equivalent to the value of the objective function. From the execution point, the demonstration followed is that every egg in a nest indicates a solution, and every cuckoo can lay only one egg, that is, one solution. In addition, no reverence can be carried out among an egg, a cuckoo or a nest.

The objective is to exploit the novel and capable cuckoo egg (solution) to substitute the worst solution in the nest. The CS approach is effectual for general optimization issues as it sustains a balance between global random

walk and the local random walk. The equilibrium between global and local random walks is adjusted by a switching constraint $\text{Pr} \in [0,1]$. Global and local random walks are demonstrated by Eq. (1) and Eq. (2), respectively.

Accordingly, in Eq. (1), and denote the present positions chosen by arbitrary permutation, β indicates the positive step size scaling factor, indicates the subsequent position, s denotes the step size, indicates the product of two vectors, F signifies the heavy side function, Pr symbolizes a variable that is exploited to switch among global and random walks, and ε denotes an arbitrary variable from uniform distribution.

In Eq. (2), $M(s, \tau)$ indicates Levy distribution exploited to describe the step size of an arbitrary walk. The Levy flight offers an arbitrary walk when the random step length is obtained from a Levy distribution, as shown in Eq. (3). It includes an infinite variance with an infinite mean. In Eq. (3), t indicates the count of iteration and $\lambda = 0, 1, 2, 3$.

$$Y_i^{t+1} = Y_i^t + \beta s \otimes F(\text{Pr} - \varepsilon) \otimes (Y_j^t - Y_k^t) \quad (1)$$

$$Y_i^{t+1} = Y_i^t + \beta M(s, \tau) \quad (2)$$

$$\text{Levy} \approx u = t^{-\lambda} \quad (3)$$

The CS algorithm suggested by Yang and Deb works on three basic conditions:

- Only one egg at one time is laid by the cuckoo. This egg is then dumped in a randomly chosen nest.
- A high-quality excellent egg is carry forwarded to the next generation.
- The numbers of host nests offered are fixed.

Following are the proved characteristics of CS:

- Global convergence requirements are satisfied.
- It is proved that CS has the ability to converge to true global optimum. As compared to other population-based algorithms that either converge to local optima or do not converge fast. CS can converge in the best way to a global optimum.
- Global search and local search abilities are supported. There are two main parts of the CS: local and global. The local component takes care of improving the best solution by taking the directed random walk. The

global component is intended to provide a diverse population. The CS parameters are tuned in such a way that search space is effectively explored and diversity of the population is maintained.

- Enhanced global search with Levy Flight: [Figure 5.2](#) shows the Levy Flight of the CS algorithm. Here, R_i indicates a CS radius. The radius indicates the maximum step that a cuckoo can take to update at one time. The cuckoo will not fly directly to the destined nest if it is within this radius; instead, it will take Levy step to reach the nest. Here, the random walk step is decided using Levy distribution, as shown in Eq. (3)

CS uses Levy flight instead of standard random walk using Gaussian processes. This allows larger changes in magnitude and direction. Levy flight has infinite mean and variance, making these larger changes possible.

Hybrid Algorithms

We have various metaheuristic algorithms in various categories. These algorithms work on basic principles with variations as per the inspiration taken from. Though these basic algorithms are quite successful in various domains, researchers have tried combinations of these algorithms to further improve the optimization process. Such algorithms are called hybrid algorithms.

Some of these algorithms are listed as follows:

- Memetic algorithms
- Combination of GA and BP
- Combination of PSO and GA

Improvisation of ML algorithms by optimizing the learning parameters

One of the observations from this survey is training the deep networks. It is difficult to train the network effectively. Classification accuracy is improved with improvement in training. So, the training of deep learning algorithms can be improved using optimization techniques. There are many optimization techniques, but recent techniques indicate the use of metaheuristic algorithms for training optimization. Metaheuristic techniques are popular because they

can be applied to any generalized problem domain. However, research is rarely conducted on metaheuristic algorithms to optimize deep learning models.

Optimization algorithms can be categorized as stochastic or deterministic. For deterministic algorithms, the output is different every time the program is run with the same environmental conditions. On the other hand, stochastic algorithms inculcate some randomness and give different solutions with the same environmental conditions. But in many applications, this randomness is beneficial and provides optimal solutions.

Here are two major case studies indicating how these optimizations are applied and how they have helped in solving the real-life problem.

Case study 1: Metaheuristic Optimization Techniques in Healthcare

Metaheuristic algorithms are used to solve several problems in the current computing world. One of their applications can be found in the healthcare domain. Metaheuristic algorithms can be used for diagnosis with better accuracy.

- **Topic**

Technological advances along with the use of state-of-the-art tools, has enabled the medical fraternity to perform different types of body examinations, accurate diagnosis and almost 100% successful surgeries. This assistance for diagnosing diseases can be because of the use of the right equipment and computer aided automated processes, the use of correct composition, learning and testing computing tools.

- **Problem Definition**

The amount of data extracted, generated and stored at different healthcare centers is huge, and it is growing continuously. Therefore, there is a need to access this large amount of data at the pace it is being captured. This would help in better, quick diagnosis and treatment of diseases. This requirement needs working on developing automated intelligent processes and systems to diagnose, predict and treat various diseases using multiple machine learning algorithms. One of the key challenges is to determine the algorithms that could be considered

impactful from the many algorithms available for diagnosing the diseases.

- **Objective of the case study**

Metaheuristic algorithms are capable of providing practical solutions in today's computing world. These metaheuristic algorithms can be used in the healthcare domain to diagnose diseases practically with better results. The major objective of this study is to:

- Extract information and patterns for diagnosis
- Determine the techniques along with metaheuristic that can be considered impactful in diagnosing the diseases
- Determine the metaheuristic techniques and its respective algorithms that can be used in a decision support system to select and optimize the features for diagnosing diseases like heart disease, Alzheimer's disease and diabetes.
- Develop the framework for executing metaheuristic techniques to diagnose diseases
- Evaluate the various algorithms for diagnosing the diseases in terms of the accuracy, precision, F1 score, error rate, sensitivity, specificity, an area under a curve, and so on
- Determine the application areas where metaheuristics technique can be used

- **Considerations for the case study**

For collecting the data, different sets of sensors and complex techniques for analyzing, recognition, decoding and forecasting the information are required.

In the healthcare sector, metaheuristics can be applied to improve classification and detection systems, leading to an increase in the early detection and treatment rate of various diseases, decreasing complications and so on.

Metaheuristics techniques can be broadly classified into the following:

- **Single solution-based algorithms:** Here, a single optimal solution is determined from all the randomly generated solutions.
- **Population solution-based algorithms:** Here, values of all the

randomly generated solutions are updated iteratively, and only then the best solution is developed.

- **System development**

The system development can focus on a survey-based approach, followed by the waterfall method, which was explained in a previous chapter. Extensive study of metaheuristic algorithms in the healthcare sector, followed by determining the structure, performing its analysis, and looking at its implementation can lead to a better system.

- **Requirement gathering and analysis phase:** In order to detect the disease, its symptoms and signs on patients need to be determined. For the same, various diagnostic tests are performed on the patient. Further, multiple tests need to be performed for the asymptotic cases where no symptoms are seen.
- **System design:** This consists of taking a number of medical images (different disease images). These images have to be brought to a common platform. Therefore, pre-processing needs to be performed.

Once this input data is ready, there is a need to extract, select and reduce the feature set. This reduction of the feature set can be done by optimizing the selected features. All these processes can be performed by the application of the desired metaheuristic algorithm. Different classes of diseases can then be obtained by classifying the features based on the splitting criteria. [Figure 5.2](#) represents a framework for disease detection using a metaheuristic algorithm:

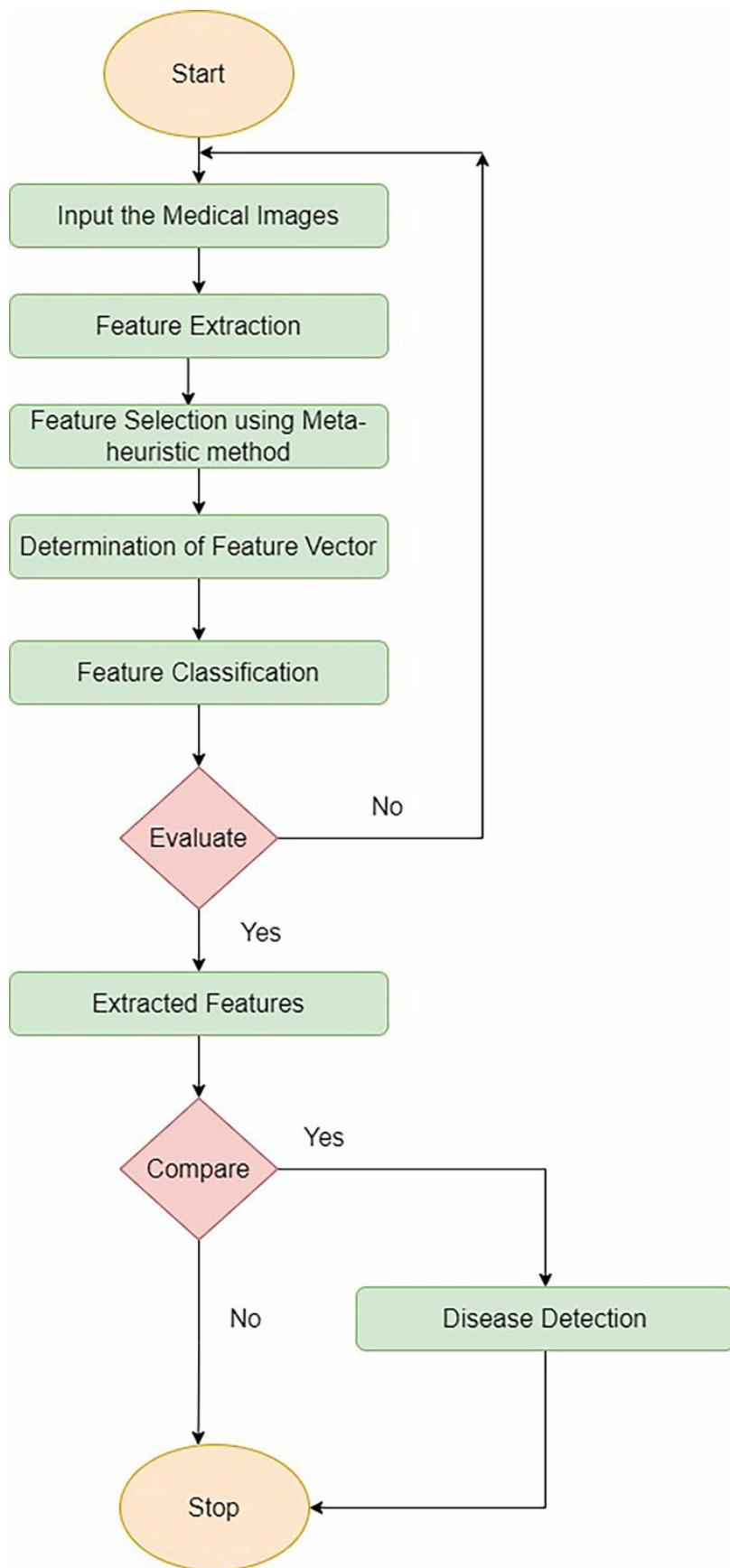


Figure 5.2: Framework for disease detection

Algorithms that can be employed

Different metaheuristic algorithms can be used to detect and classify various health-related diseases. These algorithms can include **Spider Monkey Optimization (SMO)**, **Cuckoo Search Optimization (CSO)**, **Whale Optimization Algorithm (WOA)** and **Dragonfly Algorithm (DA)** for diagnosing various diseases, like cancer, COVID, diabetes, heart disease, genetic disease and Alzheimer's disease. However, for ease of understanding, SMO and CSO have been elaborated on as follows:

- **Spider Monkey Optimization**

In this technique, the societal behavior of the spider monkeys, also known as fusion-fission, is used to find the optimal solutions. These monkeys change their behavior according to the availability of the food. When the availability of food is high, the cluster is bigger; when food is scarce, the cluster is divided into more clusters. The aim of this algorithm is to diagnose diseases by optimizing the dataset with symptoms of diseases. Thus, development of the algorithm comprises of the following:

- **Initialization of the parameters:** The initialization of the population can be done using a random distribution process.

$$SM_{ij} = SM_{\min j} + R(0, 1) \times (SM_{\max j} - SM_{\min j})$$

Where SM_i indicates the ith spider monkey; and SM min j and SM max j refer to the minimum and maximum ranges, respectively.

R (0,1) refers to the numbers that are generated randomly between the values 0 and 1.

- **Determining the fitness function value:** This value is determined using the probability function. This gives the value of position of the spider monkey, as shown in the following equation:

$$P_i = 0.9 \times \frac{fitness_i}{max\ fitness} + 0.1$$

Where max fitness gives the maximum fitness value of the

population, and fitness i gives the fitness value of the i th spider. The following figure shows the steps followed for diseases diagnosis:

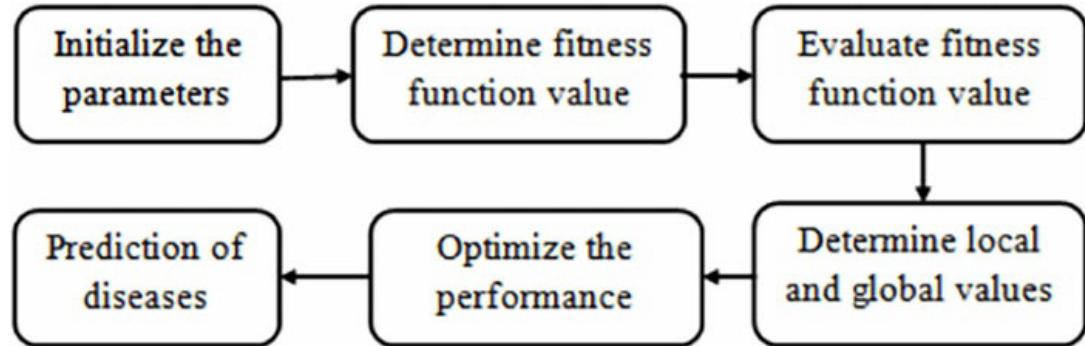


Figure 5.3: Spider Monkey optimization for disease diagnosis

- **Determining the local and global values**

Based on the fitness function, these values can be used to find the best solution. Further, the finalized values can aid in determining the prominent features. These values can further be used to optimize the performance value.

- **Prediction of disease**

Based on the selected features determined in step 3, the diseases can be predicted. The entire process is represented in [Figure 5.2](#).

- **Cuckoo Search Optimization**

This method simulates the behavior of the “Cuckoo” bird, and hence the name. All are aware that the cuckoo bird lays its eggs in some other birds’ nests. To do so, it first aims to identify the best and safest nest for laying its eggs. However, the host bird may realize that the eggs do not belong to them. Hence, they could throw them out or choose a new place for its nest. This best place could be determined by checking the similarity between the eggs of the host bird, and the availability of food.

Based on the same logic, a technique could be developed to check the status of the lungs of patients who suffered from COVID-19. The algorithm developed could be as follows:

- Input the images of the lungs: the images could be X-ray images.
- Identify the important features from the images that would be useful for diagnosis.

- The features can then be selected, extracted and segmented. Extraction and segmentation of the images can be performed through the Gabor filter bank and **Fuzzy C-Means (FCM)**, or by using rough sets.
- Classifier models like Naïve Bayes, K-mean or neuro-fuzzy classifier, along with Cuckoo Search optimization, can be applied to these images.
- Cuckoo Optimization method can then be used to train the neural network (if used) and determine the optimal value of the threshold. The accuracy and the root mean square error can be determined.
- The accuracy of the system can be determined by selecting appropriate features in terms of geometry, texture and statistics.

Tools utilization

For Spider Monkey Optimization, a Jaya Spider Monkey Optimization classifier can be used, based on convolutional long short-term memory. Time series data can be used to further predict confirmed, death, and recovered cases.

Spider monkey classification algorithm can be used to classify various types of diabetes and dengue fever. The system developed for diagnosis can be trained using a Probabilistic Neural Network.

Cuckoo search optimization can be used to determine the initial parameters through a support vector machine, particle swarm and **Principal Component Analysis (PCA)** optimization for early diagnosis of heart diseases. PCA can discard irrelevant features and speed up the convergence.

Constraints / Limitations while developing the metaheuristic system

The focus of this study is on the diagnosis of the disease with respect to the symptoms identified:

- Other parameters like age-related diseases, population density and level of healthcare facilities available can be considered.
- Use of limited computing resources can lead to more time consumption, so focus also needs to be on utilizing state-of-the-art computing resources.
- The systems being developed should be incorporated in clinical settings.

- More datasets and deep learning techniques can be used to optimize the diagnosis and performance of the system being developed.
- The optimizing process can be complex. Also, a lack of information about a person can lead to an erroneous diagnosis.
- Lack of experience of a physician can lead to inaccurate treatment.

Evaluation Measures

Various metaheuristic algorithms, such as spider monkey optimization, grey wolf optimization, and shuffled frog leaping algorithm, can be compared in terms of their average accuracy, average sensitivity, average, standard deviation, mean gradient, specificity, precision, recall, F-measure, Total time for detection, Root mean square errors, mean square errors, number of recovered cases, number of deaths and so on.

Case Study 2: Genetic Algorithm (GA) in Batch Production

This is a case study on job process planning and scheduling in batch production.

- **Topic**

GA enables us to determine the possible schedules for jobs at hand in an iterative manner. This case study elaborates on the application of job process planning and scheduling into the production of turned parts using genetic algorithm (GA) approach to achieve optimum plans.

Every organization, say any manufacturing company, has a plan of developing, implementing and resolving its workflow issues. The term JPPS is often used for job process planning and scheduling problem. The plan represents a sequence of operations made available across machines. These operations tend to produce the items under consideration with an assurance of quick, reliable delivery of the products in expected time.

Production can be categorized into single item and batch production. With respect to batch production, the following criteria hold good:

- The production is in a specific order format.
- Limited resources are available and used for production.

- Several products are manufactured at the same time. However, a single product is manufactured at a single location, which causes scheduling problems:
 - If orders are received after the scheduling has been completed, different tasks are given higher priority.
 - If orders are cancelled or an equipment malfunctions, the entire production needs to be rescheduled.
 - If the response batch production is too slow, unpredictable events might occur.
 - Another cause of concern is the time delay between process planning and the actual beginning of the plan. The more the delay, the more the plan tends to become unworkable. Moreover, fixing the plans can further lead to bottleneck situations as dynamic manufacturing environments tend to modify at least 20 to 30% of all plans.
 - Yet another issue with is that the production plans are made without considering all aspects of production. This results in delivery delays, extra costs and low machine utilization.

Therefore, the planning and scheduling goals should be integrated to create more realistic plans and schedules. This could lead to faster and better response to customers' requests.

- **Problem Definition**

The following problems are discussed in this case study:

- To identify a method for allocating orders in the most optimal way.
- Based on planning and automatic capturing of data in real time, there is a need to create a dynamic planning model.

- **Objectives of the case study**

The following objectives of the case study are mentioned:

- Scheduling model
- Resolving technical difficulties using GA
- Determining the optimum size of a batch using GA
- Integrating the improved scheduling algorithm into existing

- information systems
 - Determining the **Shortest Production Time (SPT)** of all products, while taking account of alternative sequence of operations
- **Considerations for the case study**

Following are the considerations for the case study:

 - In order to set up a dynamic planning and scheduling model, integrated IT support for logistics manipulative processes can be incorporated.
 - Dynamic models can be considered for production planning, taking care of real-time data and acquisition methods.
 - Only one product can be produced on one machine at a time.
 - Each product can be machined independently from the other.
 - Each product can have one to three alternative plans.
 - In the alternative plan, the sequence of operations needs to be predefined for each product.
 - Size of a batch needs to be divided into equal smaller units.
 - Automated, computer-aided planning can lead to integration of process planning and scheduling.

- **System development**

Planning and scheduling of products in the production of turned parts is considered to be one of the complex problems. The process planning and scheduling model that can be utilized is presented in [Figure 5.3](#).

The model can be divided into three steps. The first and second steps could be termed as the static or pre-planning phase. Based on the available resources, production quantities and material needs, various plans can be laid down in these phases.

Step 1: Process planning for new parts

In this step, there is an involvement of the customers and suppliers. Checking resources (machines), estimating the quantity of new parts, new part process plan and estimated quantities, resource optimization and scheduling algorithm can be planned in this step. Feedback can be provided to product designer, and formal part approval can be sought from the customer.

Step 2: Process planning

In this step, alternate plans can be set into the ERP systems. Based on rough calculations of capacities, periodic comparisons between customers' order predictions and resources can be performed, as shown in the previous figure. Once a year, strategic optimization of resources can be undertaken for better utilization of resources.

Step 3: Operational scheduling phase

In this step, realistic conditions and target functions get involved. Comparisons between the available and required resources are performed. In case they differ, an extra plan can be included in the process. Detailed production scheduling of the manufacturing plan can then be determined to ensure timely delivery of materials and tools by suppliers.

Once the schedule has been completed, it forms a working frame. Optimum schedules are those wherein no modifications are to be done. The following diagram shows the planning and scheduling model:

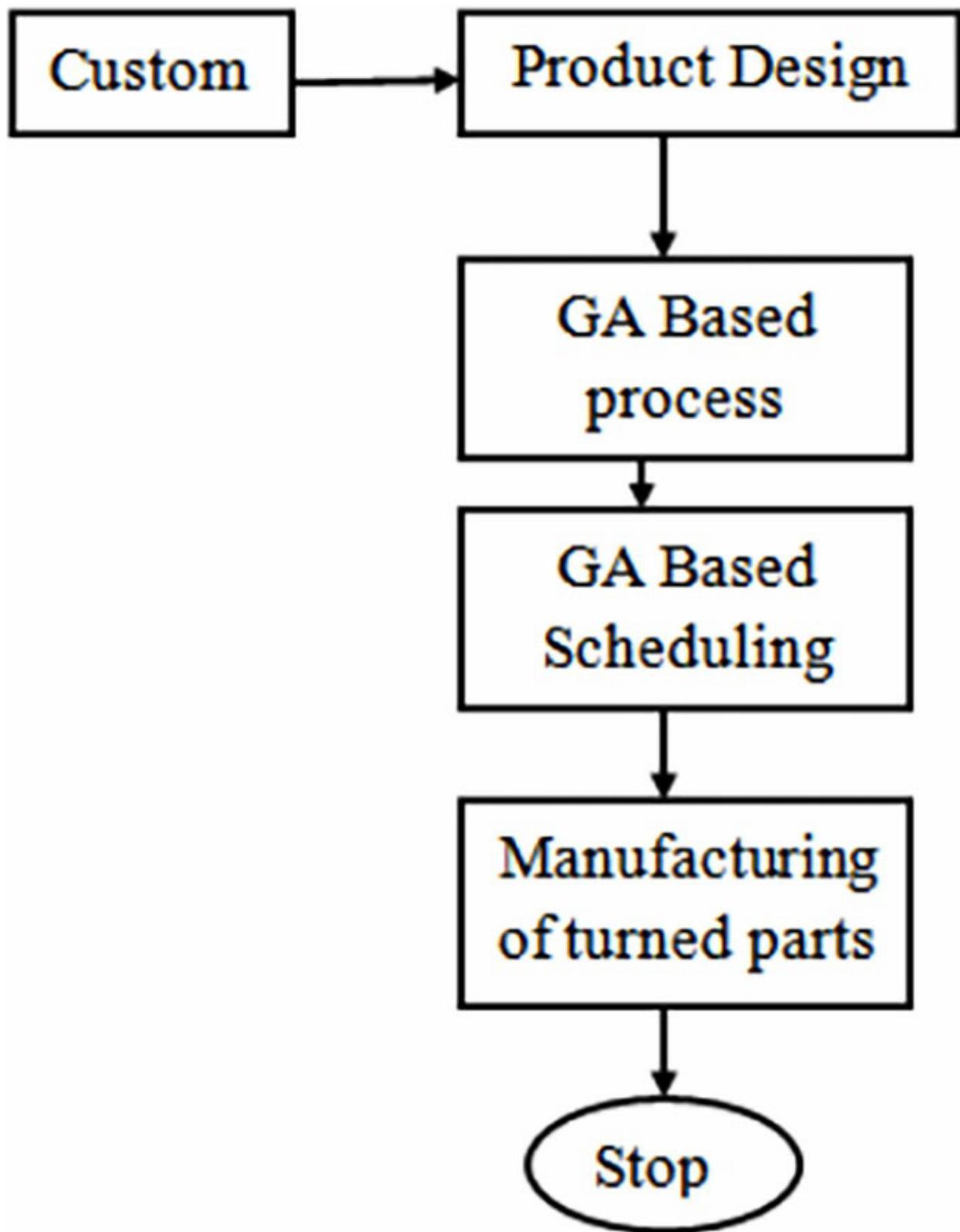


Figure 5.4: Integrated process planning and scheduling model

- Algorithms that can be employed

Manufacturing information systems form the backbone of planning and managing production. This can further integrate the ERP systems and GA scheduling software. The algorithm can comprise the following steps:

- The data stored in the ERP information system can be input to the system.
- In order to calculate the optimum plan and schedule along with the data optimization of the plan with GAs can be performed.
- Once done, the collected data can be returned to the information system.

Figure 5.5 shows the data management system and the integrated process planning and scheduling.

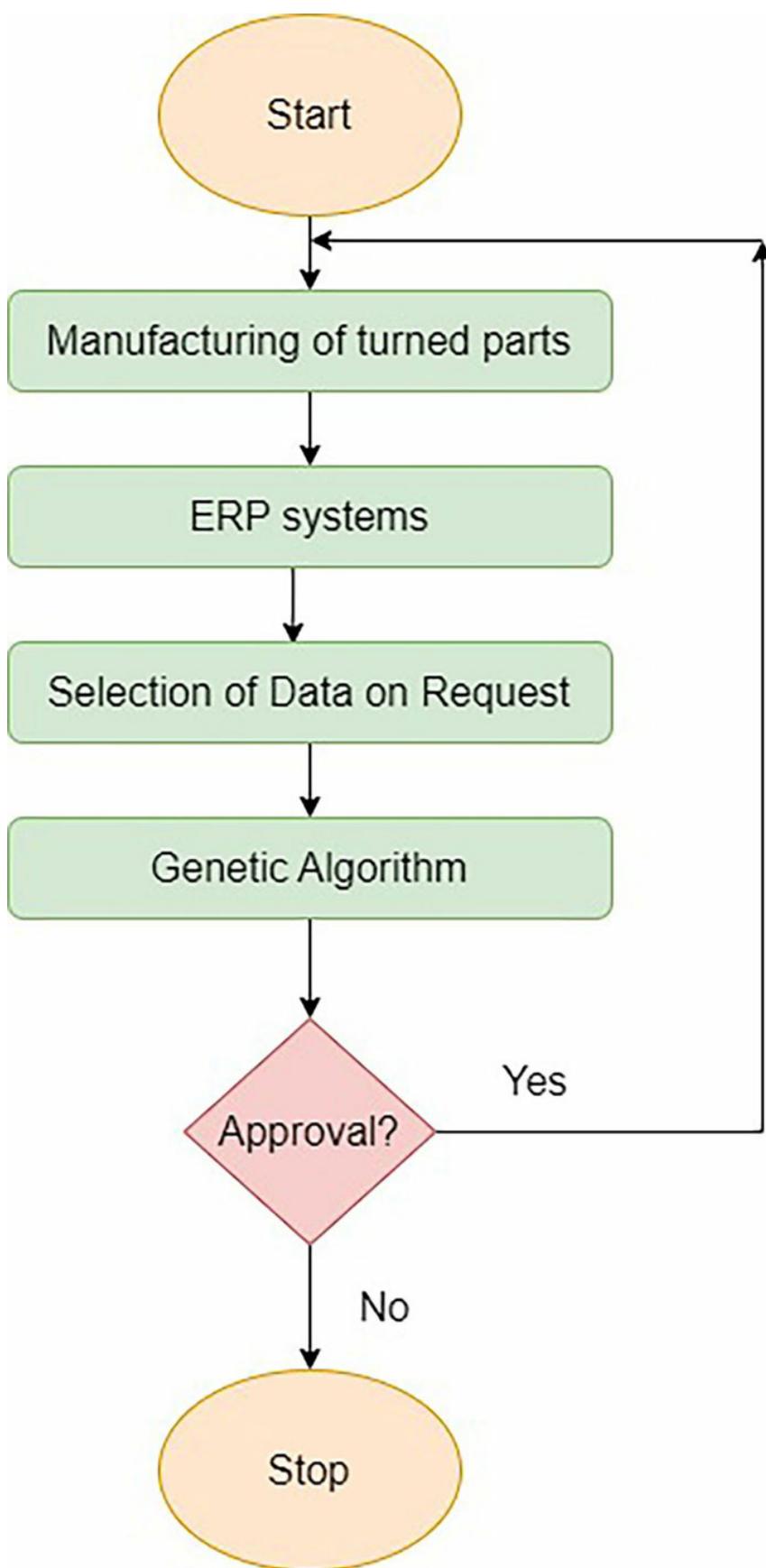


Figure 5.5: Data flow chart for scheduling

- **Software that can be used**

Data can be selected from the ERP system through its ODBC and SQL interface. Process planning and scheduling queries can be handled through relational databases like MS SQL and Visual Basic (VB) interface. A GA can be used to optimize the process.

- **Tools that can be used**

Simulated-based GA can be used to minimize mean tardiness and makespan in **Job Shop Scheduling Problem (JSS)**, and to arrange machines in a flexible processing system using variable transferable costs as their target function. GA can further be used to study job shop scheduling under machine breakdown and unavailability situations. **Advanced Overlapping Production Planning Model (AOPP)** and random key GA with forward-backward improvements can be used for scheduling problems in a **Manufacturing Supply Chain Environment (MSCE)**. Hybrid GA can be used to strengthen the search ability and improve the convergence of the GA. Self-guided algorithm, parallel GA and embedded GA can be used to minimize the makespan.

- **Evaluation parameters**

Product throughput time, close-to-optimum manufacturing schedule, time to manufacture one unit, volume of order, machine usage time, preparation time and finishing time need to be determined. Also, order execution costs, including depreciation costs and labor costs, can be calculated. Thus, GA can be applied to complex problems from production and scheduling to forming a set of rules for turned parts.

Optimization using Python

Python provides good support for optimization algorithms in machine learning, and libraries are available for working with these metaheuristic algorithms. This section will introduce some of these libraries/packages that can be used for optimization.

SwarmPackagePy Library (<https://pypi.org/project/SwarmPackagePy/>)

This is a library in Python that provides implementation of 14 optimization algorithms, including Artificial Bee Algorithm, Cuckoo Search Algorithm,

Firefly algorithm and Particle Swarm algorithm.

You can install the library using the following command:

```
pip install SwarmPackagePy
```

This library works well with Python 3.5 and onward. The code is freely available for understanding and modification on GitHub. The package includes many algorithms that are popular and used frequently for various applications for optimization in machine learning.

For example, the Cuckoo search algorithm can be invoked as follows:

```
SwarmPackagePy.cso(n, function, lb, ub, dimension, iteration,  
pa=0.25, nest=100)
```

The arguments used here are as follows:

- **n**: number of agents
- **function**: test function
- **lb**: lower limits for plot axes
- **ub**: upper limits for plot axes
- **dimension**: space dimension
- **iteration**: number of iterations

The following is example code showing the use of this package to perform Particle Swarm Optimization and then see the animation:

Example 1:

```
import SwarmPackagePy  
from SwarmPackagePy import testFunctions as tf  
from SwarmPackagePy import animation, animation3D  
alh = SwarmPackagePy.pso(50, tf.easom_function, -10, 10, 2, 20, w=0.5, c1=1,  
c2=1)  
animation(alh.get_agents(), tf.easom_function, -10, 10)  
animation3D(alh.get_agents(), tf.easom_function, -10, 10)
```

Scikit-opt Library (<https://pypi.org/project/scikit-opt/>)

This is another metaheuristic optimization library that consists of various popular optimization algorithms. This library includes algorithms like Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, and Simulated Annealing.

One can install this library using this command:

```
pip install scikit-opt
```

After installation, the library functions can be imported and used effectively. The following is another sample example that shows how the library functions can be used.

Example 2:

```
### 1. Import Libraries
import numpy as np
from sko.GA import GA, GA_TSP
from sko.operators import ranking, selection, crossover, mutation
### 2. Create Demo function
demo_func = lambda x: x[0] ** 2 + (x[1] - 0.05) ** 2 + (x[2] - 0.5) ** 2
### 3. Setup the GA environment
ga = GA(func=demo_func, n_dim=3, size_pop=100, max_iter=500, prob_mut=0.001,
         lb=[-1, -10, -5], ub=[2, 10, 2], precision=[1e-7, 1e-7, 1])
ga.register(operator_name='selection', operator=selection.tournament,
            tourn_size=3)
ga.register(operator_name='ranking', operator=ranking.ranking). \
    register(operator_name='crossover', operator=crossover.crossover_2point). \
    register(operator_name='mutation', operator=mutation.mutation)
### 4. Execute the call to GA function
best_x, best_y = ga.run()
### 5. Print the optimal solution
print('best_x:', best_x, '\n', 'best_y:', best_y)
```

In the previous code, the Genetic Algorithm is called to solve the optimization function created. The first three lines import the necessary libraries, the function is set in line 4 that needs to be solved, and line 5 sets the Genetic Algorithm parameters and hyperparameters as required. The next 2 lines talk about which selection and mutation operator is to be used in the Genetic Algorithm to generate the population. Once the parameters and environment is set, `ga.run()` is called and the algorithm is executed. The optimized results are displayed once the stopping criteria is met. One can use multiple optimization algorithms effectively with the help of this library.

MealPy library (<https://pypi.org/project/mealpy/>)

This is one of the most useful and interesting libraries for optimization algorithms. This library has implementation of almost 100 different metaheuristic algorithms based on various principles, like swarm intelligence, human inspired, and physics inspired.

One can install this library as follows:

```
pip install mealpy==2.1.0
pip install --upgrade mealpy
```

This is an extensive library, so it should be installed with Anaconda distribution. This library uses many basic Python libraries like matplotlib and

NumPy, so a correct version of the same is required for the installation and working of this library effectively.

Following is an example code that illustrates how this library can be effectively used to solve the Travelling salesman problem using the Whale Optimization algorithm. The steps are written as comments to better understand the code. Here, the travelling salesman problem for N-cities is constructed, and then the feasible solution is determined using the Whale Optimization algorithm. This library has simple and effective implementation of optimization algorithms, which makes it useful for any type of optimization problem.

Example 3:

```
### 1. Import libraries
import numpy as np
from mealpy.swarm_based import WOA
from models.tsp_model import TravellingSalesmanProblem
from models.tsp_solution import generate_stable_solution,
generate_unstable_solution
### 2. Define data
np.random.seed(10)
N_CITIES = 15
CITY_POSITIONS = np.random.rand(N_CITIES, 2)
TSP = TravellingSalesmanProblem(n_cities=N_CITIES,
city_positions=CITY_POSITIONS)
TSP.plot_cities(pathsave="./results/WOA-TSP", filename="cities_map")
### 3. Design problem dictionary
## Let's take the function that can generate stable solution
LB = [0, ] * TSP.n_cities
UB = [(TSP.n_cities - 0.01), ] * TSP.n_cities
problem = {
    "fit_func": TSP.fitness_function,
    "lb": LB,
    "ub": UB,
    "minmax": "min",           # Trying to find the minimum distance
    "log_to": "console",
    "amend_position": generate_stable_solution
}
solution = np.array([1, 5, 9, 7, 8, 0, 2, 4, 6, 3])
fit_value = TSP.fitness_function(solution)
optimal_solution = np.array([6, 4, 0, 10, 2, 8, 12, 13, 14, 7, 9, 5, 1, 11,
3])
optimal_dist = TSP.fitness_function(optimal_solution)
### 4. Call the model
model = WOA.BaseWOA(problem, epoch=10, pop_size=50)
### 5. Train the model
best_position, best_fitness = model.solve()
### 6. Show the results
print(f"Best solution: {best_position}, Obj = Total Distance:
{best_fitness}")
```

A few sample and popular Python libraries have been discussed here, but there are numerous other implementations of the Optimization Algorithms created by various programmers. Depending on the application at hand, one can choose which implementation or library to use and achieve the results. A cautious decision is always to be employed while using these libraries and their functions.

Conclusion

Optimization is one of the key aspects of any Machine Learning projects as it saves a lot of computation time and thus reduces the cost. A lot of research is done to identify suitable optimization techniques for the various application domains. The techniques are either conventional or metaheuristic. Currently, metaheuristic techniques are most popular as they help in the identification of an optimized solution faster, and the number of parameters to train are fewer. This chapter focused on a few important and widely used optimization techniques. Sample case studies were discussed to help you understand how optimization algorithms are used in real-world problems in varied domains. Numerous packages and implementations are available in Python to use these techniques effectively.

Questions

1. What is optimization in Machine Learning, and why is it important in this domain?

Ans. Optimization is the technique to make a Machine learning model more robust and better its accuracy by reducing the variance of the predicted output.

Optimization is measured through a loss or cost function, which is typically a way of defining the difference between the predicted and actual value of data. Machine learning models aim to minimize this loss function or lower the gap between prediction and reality of output data. Iterative optimization means that the machine learning model becomes more accurate at predicting an outcome or classifying data.

Machine learning models are often trained on local or offline datasets, which are usually static. Optimization improves the accuracy of predictions and classifications, and minimizes error. Without

optimization, there would be no learning and development of algorithms. So, the very premise of machine learning relies on function optimization.

The process of optimizing hyperparameters is vital to achieving an accurate model. The selection of the right model configurations directly impacts the accuracy of the model and its ability to achieve specific tasks. However, hyperparameter optimization can be a difficult task. It is important to get right model, as over-optimized and under-optimized models are both at risk of failure.

The wrong hyperparameters can cause either underfitting or overfitting within machine learning models. Overfitting is when a model is trained too closely to training data, meaning it is inflexible and inaccurate with new data. Machine learning models aim for a degree of generalization to be useful in a dynamic environment with new datasets. Overfitting is a barrier to this and makes machine learning models inflexible.

2. What are the main characteristics of Metaheuristic Optimization techniques?

Ans. A metaheuristic method helps in solving the optimization problem. Problems in optimization can be found in many daily life aspects. The kinds of the metaheuristic method are various: **Ant Colony Optimization (ACO)**, **Genetic Algorithm (GA)**, and **Particle Swarm Optimization (PSO)**.

There are several advantages of using meta-heuristic algorithms for optimization:

- **Broad applicability:** They can be applied to any problems that can be formulated as function optimization problems.
- **Hybridization:** They can be combined with traditional optimization techniques.

Two major components of any metaheuristic algorithms are intensification and diversification, or exploitation and exploration (Blum and Roli, 2003). Diversification means to generate diverse solutions so as to explore the search space on a global scale, while intensification means to focus the search on a local region, knowing that a current good solution is found in this region. A good balance between

intensification and diversification should be found during the selection of the best solutions to improve the rate of algorithm convergence. The selection of the best ensures that solutions converge to the optimum, while diversification via randomization allows the search to escape from local optima and also increases the diversity of solutions. A good combination of these two major components will usually ensure that global optimality is achievable.

3. Compare Backpropagation and Gradient descent approach to optimization.

Ans. Stochastic gradient descent is an optimization algorithm for minimizing the loss of a predictive model with regard to a training dataset.

Backpropagation is an automatic differentiation algorithm for calculating gradients for the weights in a neural network graph structure.

Gradient Descent is an optimization algorithm that finds the set of input variables for a target function that results in a minimum value of the target function, called the minimum of the function.

As its name suggests, gradient descent involves calculating the gradient of the target function.

You may recall from calculus that the first-order derivative of a function calculates the slope or curvature of a function at a given point. Read left to right, a positive derivative suggests that the target function is sloping uphill, and a negative derivative suggests the target function is sloping downhill.

- **Derivative:** Slope or curvature of a target function with respect to specific input values to the function.

If the target function takes multiple input variables, they can be taken together as a vector of variables. Working with vectors and matrices is referred to linear algebra, and doing calculus with structures from linear algebra is called matrix calculus or vector calculus. In vector calculus, the vector of first-order derivatives (partial derivatives) is generally referred to as the gradient of the target function.

- **Gradient:** Vector of partial derivatives of a target function with respect to input variables.

The gradient descent algorithm requires the calculation of the gradient of the target function with respect to the specific values of the input values. The gradient point's uphill, so the negative of the gradient of each input variable is followed downhill to result in new values for each variable that results in a lower evaluation of the target function.

A step size is used to scale the gradient and control how much to change each input variable with respect to the gradient.

- **Step Size:** Learning rate or alpha, a hyperparameter used to control how much to change each input variable with respect to the gradient.

This process is repeated until the minimum of the target function is located, a maximum number of candidate solutions are evaluated, or some other stop condition is met.

- **Backpropagation:** Algorithm for calculating the gradient of a loss function with respect to variables of a model.

You may recall from calculus that the first-order derivative of a function for a specific value of an input variable is the rate of change or curvature of the function for that input. When we have multiple input variables for a function, they form a vector, and the vector of first-order derivatives (partial derivatives) is called the gradient, that is, vector calculus.

- **Gradient:** Vector of partial derivatives of specific input values with respect to a target function.

Backpropagation is used when training neural network models to calculate the gradient for each weight in the network model. The gradient is then used by an optimization algorithm to update the model weights.

The algorithm was developed explicitly for calculating the gradients of variables in graph structures. It works backward from the output of the graph toward the input of the graph, propagating the error in the predicted output. This output is used to calculate gradient for each variable.

4. What is Swarm Intelligence? Determine a case study that uses Swarm Intelligence effectively.

Ans. Swarm intelligence refers to collective intelligence. Biologists and

natural scientists have been studying the behavior of social insects due to their efficiency in solving complex problems, such as finding the shortest path between their nest and food source and organizing their nests.

The **Swarm Intelligence (SI)** algorithms have been proved to be comprehensive methods to solve complex optimization problems by simulating the emergence behaviors of biological swarms. Nowadays, data science is getting more and more attention, and it needs quick management and analysis of massive data. Most traditional methods can only be applied to continuous and differentiable functions. As a set of population-based approaches, recent research has proved that SI algorithms have great potential for relevant tasks in this field.

Traditional model-based methods need the problems that can be written into the form of continuous and differentiable functions. However, in the face of a large amount of data and complex tasks, it is often difficult to achieve.

The population-based metaheuristic algorithms are good at solving problems that the traditional methods cannot deal with or find challenging to solve. Swarm Intelligence, a kind of metaheuristic algorithm, is attracting more and more attention and has been proven to be sufficient to handle the large-scale, dynamic and multi-objective problems in data analytics.

5. Write the Python code using any of the libraries mentioned to implement Particle Swarm Optimization to solve the travelling salesman problem.

Ans. `### 1. Import libraries`

```
import numpy as np
from mealpy.swarm_based import WOA
from models.tsp_model import TravellingSalesmanProblem
from models.tsp_solution import generate_stable_solution,
generate_unstable_solution
### 2. Define data
np.random.seed(10)
N_CITIES = 15
CITY_POSITIONS = np.random.rand(N_CITIES, 2)
TSP = TravellingSalesmanProblem(n_cities=N_CITIES,
city_positions=CITY_POSITIONS)
TSP.plot_cities(pathsave="./results/WOA-TSP",
```

```

filename="cities_map")
### 3. Design problem dictionary
## Let's take the function that can generate stable solution
LB = [0, ] * TSP.n_cities
UB = [(TSP.n_cities - 0.01), ] * TSP.n_cities
problem = {
    "fit_func": TSP.fitness_function,
    "lb": LB,
    "ub": UB,
    "minmax": "min",           # Trying to find the minimum distance
    "log_to": "console",
    "amend_position": generate_stable_solution
}
solution = np.array([1, 5, 9, 7, 8, 0, 2, 4, 6, 3])
fit_value = TSP.fitness_function(solution)
optimal_solution = np.array([6, 4, 0, 10, 2, 8, 12, 13, 14, 7,
9, 5, 1, 11, 3])
optimal_dist = TSP.fitness_function(optimal_solution)
### 4. Call the model
model = WOA.BaseWOA(problem, epoch=10, pop_size=50)
### 5. Train the model
best_position, best_fitness = model.solve()
### 6. Show the results
print(f"Best solution: {best_position}, Obj = Total Distance:
{best_fitness}")

```

6. Explain Tabu search optimization algorithm.

Ans. Tabu Search is a metaheuristic procedure for solving optimization problems designed to guide other methods to escape the trap of local minima. It is used to find optimal and nearly optimal solutions for a wide range of classical and practical problems, from scheduling to telecommunications, character recognition and neural networks. To avoid falling in a local minima, it uses memory so that it can remember moves and solutions that are already exploited. Also, it uses memory functions to allow searching strategies like intensify and diversify in search space.

Tabu Search can be used to guide other processes that use a set of moves for transforming one solution into another, and it provides guidance for measuring the attractiveness of these moves. Example of moves are swapping two tasks, changing the value of a variable (increase or decrease).

Tabu Search is based on three main concepts:

- The use of flexible attribute-based memory structures that allows evaluation criteria and historical search information to be exploited more thoroughly than by rigid memory structures or by memory-less systems. The flexible attribute-based memory structures allow evaluation criteria and historical search information to be exploited in better way.
- A mechanism that evaluates the movement based on specific criteria in tabu search
- The use of memory functions of different time spans from short term to long term, to implement strategies for intensify,
- To focus on a specific region, to diversifications — that drive the search in new regions.

7. What are the main characteristics of Evolutionary Algorithms?

Ans. Evolutionary algorithms are a heuristic-based approach to solving problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems, and anything else that would take far too long to exhaustively process. When used on their own, they are typically applied to combinatorial problems; however, genetic algorithms are often used in tandem with other methods, acting as a quick way to find a somewhat optimal starting place for another algorithm to work off of.

The premise of an **Evolutionary Algorithm** (to be further known as an EA) is quite simple, given that you are familiar with the process of natural selection. An EA involves four steps: initialization, selection, genetic operators, and termination. These steps each correspond, roughly, to a particular facet of natural selection, and they provide easy ways to modularize implementations of this algorithm category. Simply put, in an EA, fitter members will survive and proliferate, while unfit members will die off and not contribute to the gene pool of further generations, much like in natural selection.

Initialization: In order to begin our algorithm, we must first create an initial population of solutions. The population will contain an arbitrary number of possible solutions to the problem, often called members. It will often be created randomly (within the constraints of the problem), or if some prior knowledge of the task is known, roughly centered on

what is believed to be ideal. It is important that the population encompasses a wide range of solutions because it represents a gene pool; so, if we wish to explore many different possibilities over the course of the algorithm, we should aim to have many different genes present.

Selection: Once a population is created, members of the population must be evaluated according to a fitness function. A fitness function is a function that takes in the characteristics of a member and outputs a numerical representation of how viable a solution it is. Creating the fitness function can often be very difficult, and it is important to find a good function that accurately represents the data; it is very problem-specific. Now, we calculate the fitness of all members and select a portion of the top-scoring members.

Genetic Operators: This step includes two substeps: crossover and mutation. After selecting the top members (typically the top two, but this number can vary), these members are used to create the next generation in the algorithm. Using the characteristics of the selected parents, new children are created, which are a mixture of the parents' qualities. Doing this can often be difficult, depending on the type of data, but typically in combinatorial problems, it is possible to mix combinations and output valid combinations from these inputs. Now, we must introduce new genetic material into the generation. If we do not perform this crucial step, we will become stuck in local extrema very quickly, and we will not obtain optimal results. This step is mutation, and we do this, quite simply, by changing a small portion of the children such that they no longer perfectly mirror subsets of the parents' genes. Mutation typically occurs probabilistically, in that the chance of a child receiving a mutation and the severity of the mutation are governed by a probability distribution.

Termination: Eventually, the algorithm must end. There are two cases in which this usually occurs: either the algorithm reaches maximum runtime, or the algorithm reaches some threshold of performance. At this point, a final solution is selected and returned.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech

happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



Index

A

Agglomerative Hierarchical Clustering [111](#)
Ant colony optimization [211](#)
application areas, ML
 Enterprise application [10](#)
 Facebook's News Feed [10](#)
 financial services [9](#)
 government [9](#)
 health care [9](#)
 oil and gas [10](#)
 retail [9](#)
 self-driving cars [10](#)
 transportation [10](#)
 virtual assistant [10](#)
architecture, neural network
 autoencoders [75](#)
 Boltzmann machine [74](#)
 convolutional neural network [73](#)
 deep belief networks [74](#)
 feed-forward neural network [73](#)
 gated recurrent units (GRU) [74](#)
 Generative Adversarial Network (GANs) [75](#)
 Hopfield Network (HN) [74](#)
 Long/Short-Term Memory [74](#)
 perceptrons [73](#)
 recurrent neural network (RNN) [73, 74](#)
 recursive neural network [73](#)
arguments [44](#)
artificial intelligence (AI) [13](#)
auussian Mixture Models (GMMs) [109](#)

B

backpropagation algorithm [202](#)
backpropagation learning
 local minima [202](#)

network paralysis [202](#)
slow convergence [202](#)
bar plot [90, 91](#)
Boltzmann machine [74](#)
break statement [31](#)

C

case studies, predictive analytics
approach [193, 194, 195, 196](#)
challenges [193](#)
customer churning analytics [184](#)
hypothesis, building [189](#)
Instagram [181](#)
learning analytics in education systems [192](#)
Netflix [196](#)
social media analytics [178](#)
Telecom Company [186](#)
class handling [35](#)
classification [65](#)
classification algorithms
 decision trees [67](#)
 k-nearest neighbor [66](#)
 linear classifier [65](#)
 logistic regression [65](#)
 Naive Bayes classifier [65, 66](#)
 performance metrics [67](#)
 support vector machines [66](#)
 terminology [65](#)
classification metrics
 area under curve (AUC) [95](#)
 classification accuracy [95](#)
 confusion matrix [95](#)
 False Positive Rate [96](#)
 F-Measure [96](#)
 logarithmic loss [95](#)
 precision [96](#)
 recall [96](#)
 True Positive Rate [95](#)
classification model [64](#)
clustering [67](#)
 validation [70](#)
clustering algorithms [68](#)

hierarchical clustering [69](#)
k-means clustering [68](#)
mean-shift clustering [68, 69](#)
cold start problem [140](#)
collaborative filtering [129](#)
 memory-based [130](#)
 model-based [130](#)
content-based filtering [129](#)
continue statement [31](#)
conventional optimization approach [199](#)
Convolutional Neural Networks (CNNs) [139](#)
Cuckoo Search Optimization [212](#)
 characteristics [213](#)
customer churning analytics case study
 goal [184, 185](#)
 need for [185](#)
 reasons [185](#)
Customer Relationship Management (CRM) system [10](#)
customer retention [185](#)

D

data gathering [56](#)
data mining [11, 144, 146](#)
data preprocessing [56, 57](#)
 need for [57, 58](#)
 prepared data [57](#)
 raw data [57](#)
data science [11, 12](#)
data structures, Python
 examples [28, 29](#)
 logical statements [29](#)
 non-primitive data structures [27](#)
 primitive data structures [27](#)
DBSCAN [68](#)
decision support accuracy metrics [133](#)
decision tree [15, 16](#)
 neural networks [67](#)
 random forest [67](#)
deep belief networks [74](#)
deep learning [11](#)
Deeplearning4j [139](#)
dictionaries [42](#)

differential evolution algorithm [208](#)
dimensionality reduction algorithms [18, 19](#)

E

ElasticNet Regression [64](#)
e-learning system domain case study, recommendation systems [133](#)
algorithms, using [137-139](#)
considerations [135](#)
constraints / limitations [139-141](#)
objective [134](#)
problem definition [134](#)
Recommender Systems (RS) [134](#)
requirement gathering and analysis phase [136](#)
system design [136, 137](#)
system development [135](#)
elif keyword [43](#)
else keyword [43](#)
embedding space [128](#)
estimation of distribution algorithms (EDAs) [138](#)
evaluation, ML Systems
datasets, testing [93](#)
datasets, training [93](#)
model evaluation techniques [94](#)
performance measures [94](#)
Evolutionary Algorithms (EA) [138, 200](#)
differential evolution (DE) [138](#)
evolution strategies (ESs) [138](#)
genetic algorithms (GAs) [138](#)
evolutionary programming [209](#)
exception handling [36, 47](#)
Expectation–Maximization (EM) [68, 109](#)
Exploratory Data Analysis (EDA) [98](#)

F

feature [3](#)
feature creation [3](#)
feature selection [143](#)
file handling [36-38](#)
for statement [31](#)
functions [32](#)
example [33](#)

G

Generative Adversarial Network (GANs) [75](#)

Genetic Algorithm (GA)

batch production case study [220-225](#)

Gradient Descent (GD) or steepest descent algorithm [137](#)

gradient descent optimization [202](#), [203](#)

Graphics Processing Units (GPUs) [41](#)

H

health care domain case study, image processing [161](#)

algorithms, using [163](#), [164](#)

considerations [162](#)

constraints/limitations [164](#)

evaluation measures [164](#)

objective [162](#)

problem definition [162](#)

system development [162](#), [163](#)

tools, using [164](#)

hierarchical agglomerative clustering [69](#), [70](#)

hierarchical clustering algorithms

bottom-up algorithm [69](#)

top-down algorithm [69](#)

histogram [91](#), [92](#)

Hopfield Network (HN) [74](#)

Human resource (HR) systems [10](#)

hybrid algorithms [201](#), [213](#)

hybrid filtering [130](#)

hypotheses

building [189](#)

feature engineering [189](#)

feature extraction [189](#)

model evaluation metrics [190](#)

train/test split [190](#)

I

IDE [139](#)

if else statement [29](#), [30](#)

image processing [157](#)

algorithms, using [160](#), [161](#)

importance [158](#)

purpose [158](#)
real time case studies [161](#)
image processing system [159](#)
 color image processing [159](#)
 description [160](#)
 image acquisition [159](#)
 image compression [160](#)
 image enhancement [159](#)
 image restoration [159](#)
 morphological processing [160](#)
 representation [160](#)
 segmentation [160](#)
import statement [45](#)
Information Retrieval (IR) [145](#)
 natural language processing (NLP) [145, 146](#)
Instagram case study [181, 182](#)
 approach [182](#)
 evaluation [182](#)
 improvements [184](#)
 market analysis [184](#)
 objectives [181](#)
 purpose [181](#)

J

Jupyter Notebook [26, 27](#)

K

K-means [17, 18](#)
K-means clustering [68](#)
kNN (k-Nearest Neighbors) [17](#)

L

label [3](#)
lambda function [47](#)
Lasso Regression [64](#)
learning analytics case study [192](#)
 organizational benefits [192](#)
linear classifier [65](#)
linear regression [13, 14, 63](#)
line graph [87-89](#)

list [42](#)
logical statements [29](#), [30](#)
logistic regression [14](#), [63](#), [64](#)
loop statements [31](#)
 break statement [31](#)
 continue statement [31](#), [32](#)
 for statement [31](#)
 while statement [31](#)

M

Machine Learning algorithm [12](#)
 preprocessing [58](#)
 types [5](#)
 working [6](#), [20](#)
machine learning libraries, Python [77](#)
 Numpy [77-81](#)
 Pandas [81](#)
Machine Learning (ML) [1](#), [2](#)
 application areas [9](#)
 limitations [8](#)
 versus, deep learning [11](#), [12](#)
 versus, predictive modeling [166](#), [167](#)
Machine Learning model [55](#), [56](#)
 process flow [56](#), [57](#)
 reinforcement machine learning model [4](#)
 semi-supervised machine learning model [4](#)
 supervised machine learning model [3](#)
 unsupervised machine learning model [3](#), [4](#)
Machine Learning projects
 challenges [7](#), [8](#)
margin [76](#)
matplotlib [87](#)
 bar plot [90](#), [91](#)
 histogram [91](#), [92](#)
 line graph [87-89](#)
 pie chart [92](#), [93](#)
 scatter plot [89](#), [90](#)
MealPy library [228](#)
Mean Absolute Error (MAE) [133](#)
mean squared error (MSE) [72](#)
metaheuristic algorithms [199-203](#)
 advantages [204](#)

Ant colony optimization [211](#)
Cuckoo Search Optimization [212](#)
disadvantages [204](#), [205](#)
evolutionary algorithms [200](#)
hybrid algorithms [213](#)
population-based algorithms [200](#), [207-213](#)
single solution-based algorithms [205-207](#)
single solution-based algorithms [200](#)
swarm intelligence algorithms [200](#)
Metaheuristic Optimization Techniques
 healthcare case study [214-220](#)
mixed membership [111](#)
ML algorithms
 improvisation, by optimizing learning parameters [214](#)
model evaluation metrics [94](#)
 classification metrics [95](#)
 Regression Metrics [96](#)
model evaluation techniques
 cross-validation [94](#)
 holdout [94](#)
module [33](#)
 example [33](#), [34](#)

N

Naive Bayes classifier [65](#), [66](#)
named-entity recognition (NER) [146](#)
natural language processing (NLP)
 Part-of-Speech (PoS) tagging [146](#)
 sentiment analysis [146](#)
 summarization [145](#)
 text categorization or text classification [146](#)
neural network [71](#)
 architecture [73](#)
 neurons [71](#)
 neurons, combining [72](#)
 training [72](#)
Numpy [77](#)
 array, creating [78-81](#)
 predefined arrays [78](#), [79](#)

O

open() function [49](#)
opinion mining [157](#)
optimization [198](#)
 performing, with Python [225-228](#)
optimization techniques [199](#)
 backpropagation algorithm [202](#)
 conventional optimization approach [199](#)
 gradient descent optimization [202, 203](#)
 metaheuristic algorithm [199-201](#)

P

Pandas [81](#)
 basic descriptive statistics [85](#)
 data, accessing in dataframe [83, 84](#)
 data, displaying with dataframe [83](#)
 dataframe [82](#)
 dataframe, populating [82, 83](#)
 data preprocessing [86](#)
 data transformation [86](#)
 missing values, handling [86](#)
 panel [82](#)
 series [82](#)
Part-of-Speech (POS) tagging [144](#)
performance metrics
 for classification problem [67](#)
pie chart [92, 93](#)
polynomial regression [64](#)
population-based algorithms [200, 207](#)
 differential evolution algorithm [208, 209](#)
 evolutionary programming [209](#)
 evolutionary techniques [207](#)
 genetic algorithms [208](#)
 particle swarm optimization [210](#)
 Swarm Intelligence [209, 210](#)
Precision-Recall Curve (PRC) [133](#)
predictive algorithms
 decision trees [168](#)
 Generalized Linear Model (GLM) [169](#)
 Gradient Boosted Model [169](#)
 K-Means [169](#)
 Neural Networks [169](#)
 Prophet [170](#)

random forest algorithm [169](#)
regression model [169](#)
predictive analytics [165](#)
 applications [174](#), [175](#)
 benefits [175](#), [176](#)
 case studies [178](#)
 characteristics [165](#)
 purpose [165](#)
 significance [165](#)
 versus, data analytics [173](#), [174](#)
predictive analytics models
 building [167](#), [168](#)
 classification model [170](#)
 clustering model [170](#)
 forecast model [170](#)
 outliers model [170](#)
 time series model [170](#)
predictive modeling
 challenges [176](#)
 limitations [177](#)
 versus, data analytics [172](#)
 versus, Machine Learning (ML) [166](#), [167](#)
 versus, predictive analytics [171](#)
preprocessing, ML
 hyperparameter tuning [62](#)
 metrics, for evaluation [62](#)
 model, evaluating [61](#)
 model, selecting [58](#), [59](#), [60](#)
 model, testing [60](#), [61](#)
 model training [60](#)
 prediction [62](#)
probabilistic model-building genetic algorithms (PMBGAs) [138](#)
product recommendation system case study, sentiment analysis
 considerations [154](#), [155](#), [156](#)
 problem definition [151](#)
 product recommendation [151](#)
 system design [153](#)
 system development [152](#)
Prophet [170](#)
Python [23](#), [138](#)
 class handling [35](#)
 data structures [27](#)
 exception handling [36](#)

file handling [36-38](#)
functions and modules [32](#)
loop statements [31](#)
machine learning libraries [77](#)
string functions [38-40](#)

R

random forest algorithm [169](#)
read() method [49](#)
Receivers Operating Characteristics (ROC) [133](#)
recommendation algorithms
 accuracy [132](#)
 evaluation metrics [132](#)
 statistical accuracy metric [133](#)
recommendation generation
 information collection phase [131, 132](#)
 learning phase [132](#)
 prediction/recommendation phase [132](#)
recommendations [127](#)
 home page recommendations [128](#)
 related item recommendations [128](#)
recommendation systems
 architecture [130](#)
 candidate generation [130, 131](#)
 e-learning system domain case study [133](#)
 embeddings [128](#)
 items/documents [128](#)
 key terms [128](#)
 query/context [128](#)
 significance [127, 128](#)
 techniques, for building [128-130](#)
Recommender Systems (RS) [134](#)
regression algorithms [62](#)
 ElasticNet Regression [64](#)
 Lasso Regression [64](#)
 linear regression [63](#)
 logistic regression [63, 64](#)
 polynomial regression [64](#)
 ridge regression [64](#)
 stepwise regression [64](#)
 types [62](#)
Regression Metrics

Mean Absolute Error (MAE) [96](#)
Root Mean Squared Error (RMSE) [97](#)
reinforcement machine learning model [4](#)
 actions [4](#)
 agent [4](#)
 environment [4](#)
ridge regression [64](#)

S

scatter plot [89, 90](#)
scikit-learn [139](#)
Scikit-opt Library [226, 227](#)
semi-supervised machine learning model [4](#)
sentiment analysis [21, 146, 147](#)
 deep learning [150](#)
 linear regression [148](#)
 Naive Bayes algorithms [147](#)
 product recommendation system case study [151](#)
 support vector machine [148-150](#)
sets [42](#)
Silhouette Coefficient (SC) [70](#)
simulated annealing [205, 206](#)
single solution-based algorithms [205](#)
 simulated annealing [205, 206](#)
Tabu search [206, 207](#)
variable neighborhood search [207](#)
Social media analytics case study [178](#)
 approach [179](#)
 challenges [178, 179](#)
 considerations [179, 180](#)
 problem definition [179](#)
split() method [50](#)
Spyder IDE [24](#)
 code editor [25](#)
 IPython Console [26](#)
 project explorer [25](#)
 variable explorer [25](#)
stepwise regression [64](#)
string functions [38-40, 50](#)
supervised machine learning model
 through training [3](#)
support vector machine (SVM) [75, 76](#)

versus, neural networks [76](#), [77](#)
Swarm Intelligence [209](#)
swarm intelligence algorithms [200](#)
SwarmPackagePy Library [225](#), [226](#)
SWOT analysis [183](#)
System Development Life Cycle (SDLC) approach [135](#)

T

Tabu search [206](#)
techniques, for recommendation systems
 collaborative filtering [129](#), [130](#)
 content-based filtering [129](#)
 hybrid filtering [130](#)
Telecom Company case study [186](#)
 approach [186](#)
 business challenge [186](#)
 churning label [188](#)
 customer account information [188](#)
 customers demographic information [189](#)
 data collection [188](#)
 Exploratory Data Analysis (EDA) [188](#)
 problem definition [187](#)
TensorFlow [139](#)
testing data [3](#)
text analytics [141](#)
text mining [141](#)
 data types [142](#)
 process [142-144](#)
 significance [141](#)
unstructured text document, preparing [144](#), [145](#)
text mining techniques
 Information Retrieval (IR) [145](#)
text preprocessing [143](#)
text transformation [143](#)
training data [3](#)
try-expect statement [48](#)
tuples [42](#)

U

unsupervised machine learning model [3](#), [4](#)

V

variable neighborhood search [207](#)

variable selection [143](#)

W

Waterfall model [152](#), [153](#)

while statement [31](#)

X

XGBoost [19](#)