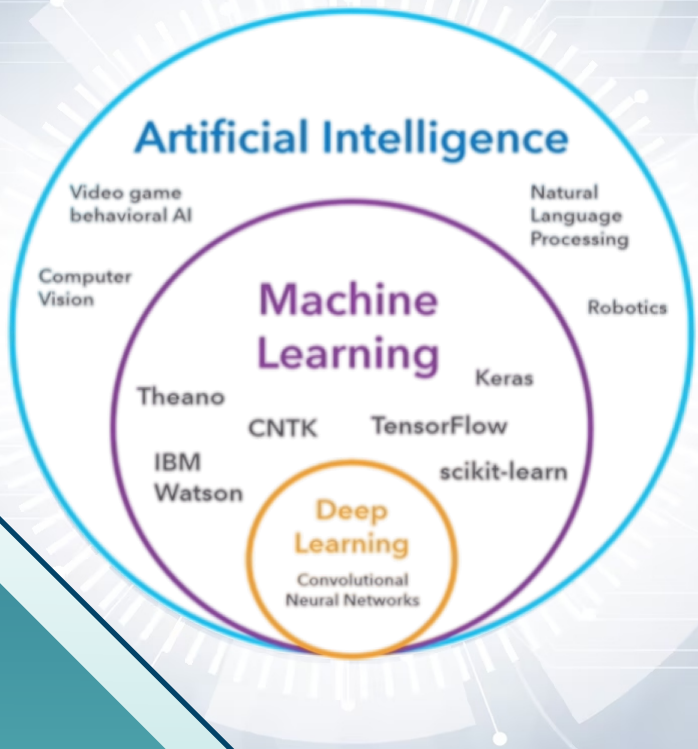
BAOU
Education for All

Dr. Babasaheb Ambedkar
Open University
(Established by Government of Gujarat)

NAAC ACCREDITED A++ GRADE 3.55/4

Machine Learning
MSCDS-302



Master of Science - Data Science (MSCDS)

2024

Machine Learning

Dr. Babasaheb Ambedkar Open University

Expert Committee

Prof. (Dr.) Nilesh Modi Professor and Director, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad	**(Chairman)**
Prof. (Dr.) Ajay Parikh Professor and Head, Department of Computer Science, Gujarat Vidyapith, Ahmedabad	**(Member)**
Prof. (Dr.) Satyen Parikh Dean, School of Computer Science and Application, Ganpat University, Kherva, Mahesana	**(Member)**
Prof. M. T. Savaliya Associate Professor and Head (Retired), Computer Engineering Department, Vishwakarma Engineering College, Ahmedabad	**(Member)**
Dr. Himanshu Patel Assistant Professor, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad	**(Member Secretary)**

Course Writer

Dr. Harshal A. Arolkar Professor & Head, FCAIT-GLS University
Dr. Madhuri Chopade Assistant Professor, FET-GLS University
Dr. Jenny Kasudiya Assistant Professor, FCAIT-GLS University
Ms. Shivangi Gandhi Assistant Professor, FET-GLS University

Content Editor

Dr. Shivang M. Patel Associate Professor, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad

Subject Reviewer

Prof. (Dr.) Nilesh Modi Professor and Director, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad

ISBN- 978-81-986947-1-3

Printed and published by: Dr. Babasaheb Ambedkar Open University, Ahmedabad

Dr. Babasaheb Ambedkar Open University

MSCDS-302

Machine Learning

Block-1: Introduction to Machine Learning Methods

Unit-1: Overview of Machine Learning 02

Unit-2: Techniques of Machine Learning 15

Unit-3: Deep Learning and Algorithms 28

Unit-4: Applications of ML 43

Block-2: Regression, Classification and Ensemble Methods

Unit-1: Introduction to Regression 55

Unit-2: Introduction to Classification 77

Unit-3: Ensemble Methods 113

Unit-4: Model Evaluation 127

Block-3: Neural Networks and Feature Engineering

Unit-1: Introduction to Neural Networks 146

Unit-2: Activation Functions 164

Unit-3: Backpropagation and Optimization 186

Unit-4: Feature Selection and Extraction 213

Block-4: Clustering and Association Rules

Unit-1: Introduction to Clustering 244

Unit-2: Clustering Algorithms 258

Unit-3: Clustering and Association Rules 281

Unit-4: Association Rule Algorithms 291

Block-1

Introduction to Machine Learning Methods

Unit-1: Overview of Machine Learning

1

Unit Structure

1.0. Learning Objectives

1.1. Introduction

1.2 Features of Machine learning

1.3 Characteristics of Machine Learning

1.4 Key Concepts and Terminologies

1.5 History and Evolution

1.6 Let us sum up

1.7 Check your Progress: Possible Answers

1.8 Assignments

1.0 LEARNING OBJECTIVE

After studying this unit students should be able to:

- Comprehend the fundamental concepts of machine learning
- Understand key features and characteristics of machine learning
- Familiarize themselves with essential concepts and terminology in machine learning
- Explore the historical development and advancements in the field of machine learning

1.1 INTRODUCTION

What is Machine Learning?

Machine learning (ML), a core branch of artificial intelligence (AI), enables computers to identify patterns in data, learn from them, and make decisions with minimal human intervention. Unlike traditional rule-based programming, where systems follow explicit instructions, machine learning models improve their performance over time by learning from data. As NVIDIA defines it, machine learning "uses algorithms to analyze data, extract insights, and automatically make educated predictions or decisions without direct human involvement."

A notable milestone in AI history occurred in 1997 when IBM's Deep Blue defeated chess champion Garry Kasparov. While Deep Blue relied on brute-force algorithms executing millions of moves per second, it lacked the ability to learn or adapt from experience; features that distinguish modern machine learning systems. Figure 1.1 represents a Machine Learning System workflow.

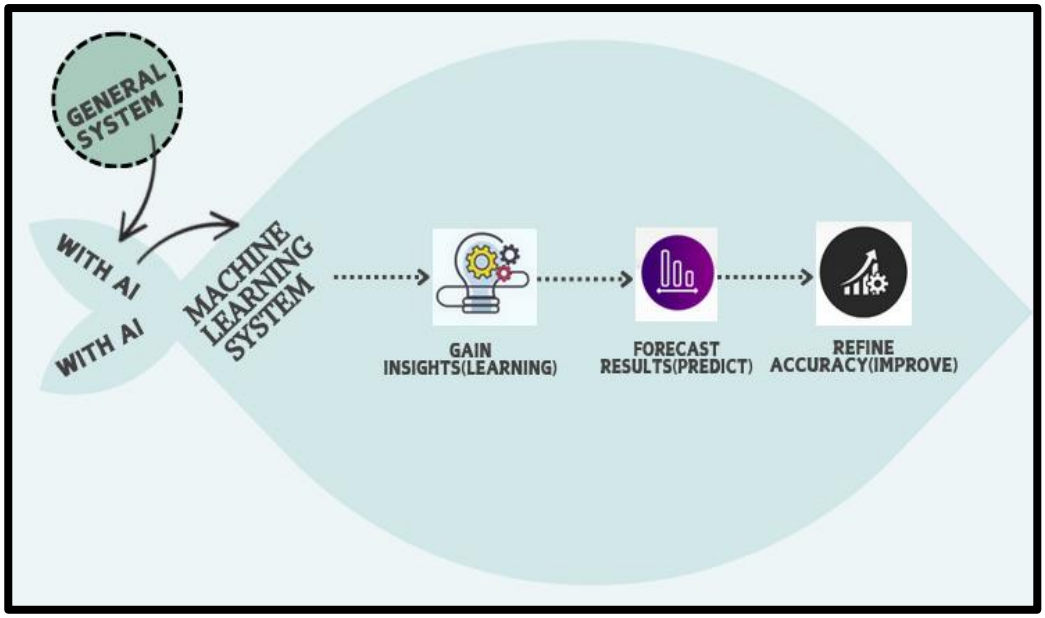


Figure 1.1: Machine Learning System Workflow

Machine Learning Systems

A traditional system works and adheres to established, predetermined regulations. On the other hand, machine learning systems which are integrated with artificial intelligence, evolve and enhance themselves by acquiring knowledge from data. For example; An ATM machine executes a set of predetermined functions like dispensing money, showing bank account details, etc.; but a chatbot when used by a person acquires knowledge through user engagement and is capable of answering differently for the same question asked. As can be seen in the figure, there is a machine learning system that does three things namely; Learn, Predict and Refine. Let us discuss these components in brief:

Gain Insights (Learning):

The machine learning systems evaluate extensive datasets to identify patterns and insights. These findings constitute the basis for forecasts. For example; a medical model may utilize imaging data to facilitate early disease detection.

Forecast Results (Prediction):

The machine learning systems use data-driven insights to make forecasts and recommend actions. The predictive models improve their accuracy over time through continuous learning. For example; Netflix suggests films based on the user's activity and interactions.

Refine Accuracy (Improve):

The machine learning systems consistently refine their predictions by incorporating new data. The model's accuracy keeps on improving with each iteration or interaction. For example; Google's search algorithm improves with every query it analyzes.

1.2 FEATURES OF MACHINE LEARNING

Tom Mitchell further explains that a computer program is said to learn from experience (E) in relation to a task (T) and a performance measure (P) if its performance on T improves with experience E.

Task (T): The task refers to a specific problem that the program is attempting to solve. It might involve tasks like identifying items in photos, forecasting home values, or determining if emails are spam.

Experience (E): The experience refers to the data or examples that are fed to the program as input. The can be labeled, historical, or of any other type from which the program can learn.

Performance Measure (P): It specifies how the program's effectiveness is evaluated. It provides a metric to evaluate how well the program performs the task. For instance, this could represent mean squared error (MSE) in a regression work or accuracy or F1-score in a classification task.

Let us see how the 8-puzzle problem can be explored on the basis of the above three features:

Task: The task in this case is to solve the 8-puzzle, where the goal is to rearrange a 3 x 3 grid of numbered tiles into a specified order by sliding tiles into an empty space. The algorithm's objective is to achieve this arrangement, demonstrating its capability to solve combinatorial puzzles.

Experience: The algorithm enhances its puzzle-solving skills by learning from past attempts. It uses data from previous moves and outcomes to develop more effective strategies, enabling it to make better decisions in future attempts.

Performance Measure: The model's performance is evaluated based on the number of moves it takes to reach the solution and the time required to solve the puzzle. These metrics guide the algorithm to optimize its strategy, aiming for faster and more efficient solutions.

Check Your Progress-1

a) ML systems acquire knowledge from data, but traditional systems adhere to predetermined rules. (True/False)
b) Number of tiles used is crucial in the 8-puzzle problem. (True/False)
c) According to Tom Mitchell, a machine learning program improves its performance on a task if it uses more data for training. (True/False)
d) "Enhancing performance metrics" involves improving model accuracy and precision. (True/False)

1.3 CHARACTERISTICS OF MACHINE LEARNING

Machine learning (ML) is a subset of artificial intelligence (AI) some key characteristics of machine learning are acquiring knowledge from data, ongoing enhancement, automation, identification of patterns, scalability and self-assessment criterion.

Let us look at them in brief:

Acquiring Knowledge from Data - It refers to the process of extracting valuable insights, trends, or patterns from raw data.

Ongoing Enhancement - This is the process of continuously making a system or model better over time.

Automation - It is a process of developing a system that can perform tasks without human intervention.

Identification of Patterns - It involves identifying recurring patterns, correlations, or trends that can be utilized to categorize new occurrences or forecast future events.

Scalability - It refers to the capacity of a system or model to accommodate expansion, whether it be in terms of data volume or user count.

Self-Assessment Criterion - It involves evaluating the performance of a model or system based on predefined standards.

Let us have a look at an E-commerce Recommendation System and see how these characteristics have been imbibed in it.

Learning from Data: An E-commerce recommendation system learns user preferences by analyzing their browsing and purchase history to suggest products tailored to individual shoppers.

Continuous Improvement: The system continuously updates its recommendations based on new user interactions and trends, adapting to changes in consumer behavior without requiring manual adjustments.

Automation: Once trained, the system automatically generates personalized product suggestions for users, enhancing their shopping experience without human intervention.

Pattern Recognition: The recommendation system identifies complex patterns in consumer behavior, such as recognizing that users who buy a certain item often purchase complementary products, which traditional programming might overlook.

Scalability: The system can process massive amounts of user data across millions of transactions, allowing it to provide relevant recommendations to a large user base.

Self-Assessment: The system assesses the effectiveness of its recommendations by tracking user engagement and conversion rates, refining its algorithms based on this feedback to improve accuracy and relevance.

1.4 KEY CONCEPTS AND TERMINOLOGIES

Designing a Machine Learning system is a multi-step process. It involves different processes like data collection, pre-processing of data, feature engineering, model training and evaluation, model deployment and market testing. Figure 1.2 shows these processes as key concepts and terminologies.

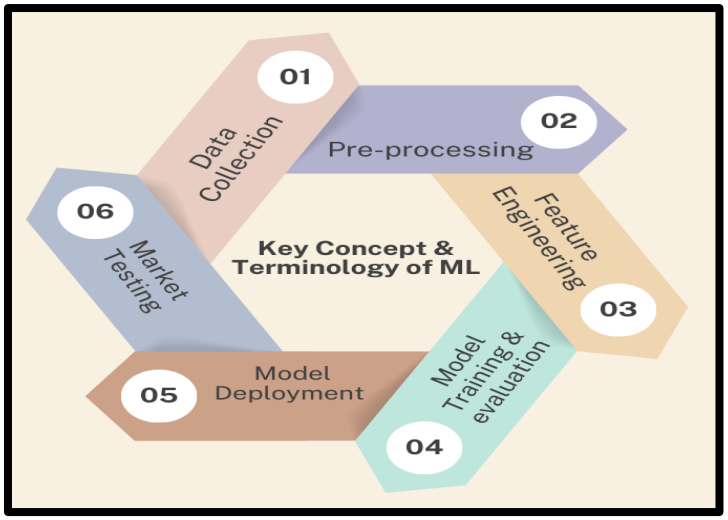


Figure 1.2: Key Concepts and Terminologies

Data Collection: The first step involves gathering raw data relevant to the problem at hand.

Example: If a company wants to predict customer churn; they need to collect data from various sources, including customer demographics, transaction history, and customer support interactions.

Data Preprocessing: In this step, the raw data is cleaned and prepared for analysis. This includes removing duplicates, handling missing values, and normalizing data.

Example: In the customer churn dataset, missing values for age are filled with the average age, and transaction amounts are normalized to a common scale for better model performance.

Feature Engineering: This step involves selecting and transforming raw data into meaningful features that improve the model's ability to learn.

Example: From the customer data, new features such as "total spending" (sum of all transactions) and "average support calls per month" can be created to provide more insight into customer behavior.

Model Training and Evaluation: The dataset is divided into training and testing subsets. The model is trained using the training data, and its performance is evaluated using various metrics.

Example: The training dataset is used to train a logistic regression model to predict churn. The model is then tested on the testing dataset, and metrics like precision and recall are calculated to assess its effectiveness.

Model Deployment and Tuning: Once the model is trained and evaluated, it is deployed into a production environment. Continuous tuning and optimization are performed to enhance its accuracy.

Example: The churn prediction model is integrated into the company's CRM system, where it flags at-risk customers. Hyperparameters that control the behavior and performance of a learning algorithm are adjusted based on real-time feedback to improve predictions.

CASE STUDY:

An illustrative instance of machine learning (ML) use in predictive maintenance within the manufacturing sector. This case study illustrates how a major automobile corporation employed machine learning to forecast equipment malfunctions within its manufacturing facilities.

- **Issue:** The organization had recurrent and unforeseen equipment malfunctions, resulting in expensive downtime and repairs. Conventional maintenance regimens were ineffective, being either excessively frequent, squandering resources, or inadequately timed, leading to equipment failures.
- **Resolution:** The company deployed a machine learning algorithm that examined historical data from sensors affixed to the machines. The sensors gathered real-time data including temperature, vibration, and pressure. Utilizing this data, the machine learning system detected patterns and correlations that suggested the likelihood of machine failure.
- **Effect:** The machine learning algorithm precisely forecasted imminent equipment breakdowns prior to their manifestation. This enabled the organization to conduct maintenance solely when required, so averting unanticipated downtime, reducing expenses, and prolonging the longevity of their equipment. With time, the model enhanced its accuracy by assimilating fresh data, hence streamlining the maintenance process further.

- The above case study illustrates how machine learning may enhance operational efficiency, decrease expenses, and yield actionable insights in manufacturing settings.

Check Your Progress-2

a) Machine learning models continuously improve their predictions by analyzing _____
b) The first step in the machine learning process is _____.
c) Machine learning systems are scalable as they don't need data to function. (True/False)
d) Data collection involves cleaning and preparing data for analysis in machine learning. (True/False)
e) Primary way machine learning models improve precision by analyzing data to identify patterns. (True/False)

1.5 HISTORY AND EVOLUTION

The concept of machine learning (ML) began in the year 1950 when Arthur Samuel, a pioneer in the field of artificial intelligence started development of a checkers-playing program, one of the earliest instances of a machine learning system. The phrase "machine learning" achieved prominence in this decade. In the 1960s, Frank Rosenblatt, a psychologist and computer scientist at Cornell University unveiled the perceptron, a primitive neural network model. Interest in machine learning varied, but it experienced a resurgence in the 1980s with the advent of the back-propagation algorithm, which significantly enhanced neural network training. The 1990s witnessed breakthroughs in algorithms such as Support Vector Machines (SVM), signifying substantial progress in the domain. In 1997, IBM's Deep Blue, a chess-playing computer system, achieved a historic milestone in artificial intelligence by defeating global chess champion Garry Kasparov. The 2000s ushered in the era of large data and advanced GPUs, facilitating the emergence of deep learning. In 2012, deep learning garnered significant attention when a deep convolutional neural network triumphed in the ImageNet competition, transforming domains such as computer

vision and natural language processing. Currently, machine learning persists in its evolution, propelling advancements across several industries.

Figure 1.3 illustrates the diagrammatic representation of the history and evolution of machine learning.

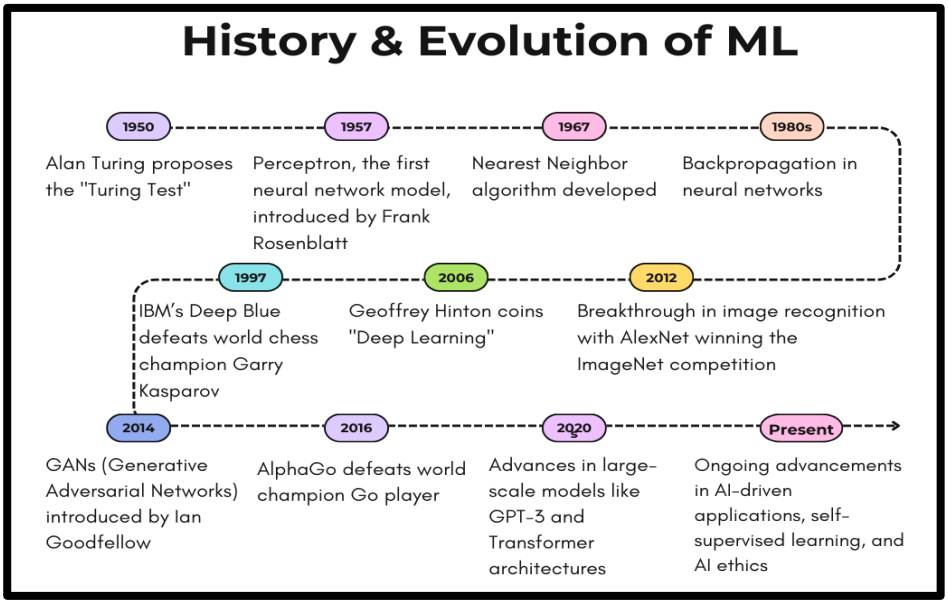


Figure 1.3: Journey of ML

Let us have a brief discussion of how ML evolved over different decades.

1950s - Conceptual Origins: Alan Turing, a British mathematician, logician, cryptanalyst, and computer scientist introduced the idea of machines learning from data in his famous paper "Computing Machinery and Intelligence"; Arthur Samuel coined the term "machine learning" while creating a self-improving checkers program.

1960s - Birth of Neural Networks: Frank Rosenblatt developed the perceptron, an early neural network model, marking the first attempt at mimicking the human brain's structure.

1970s - Shift to Symbolic AI: Artificial Intelligence research focused on rule-based systems and expert systems, where computers used predefined rules to mimic human decision-making.

1980s - Decline of Neural Networks: Due to limited computing power and inefficient learning algorithms, interest in neural networks declined in favor of other artificial intelligence methods like symbolic reasoning.

1990s - Machine Learning Revival: Increased computational power, algorithmic improvements, and larger datasets revived interest in machine learning, with IBM's Deep Blue famously defeating chess champion Garry Kasparov in 1997.

2000s - Rise of Big Data: The explosion of digital data, cloud computing, and advances in hardware accelerated machine learning research, enabling more complex models and applications.

2010s - Deep Learning Revolution: The development of deep learning techniques (Convolutional Neural Networks - CNNs and Recurrent Neural Networks - RNNs) transformed fields like computer vision, speech recognition, and game-playing AI, with milestones like Google's AlphaGo defeating the world Go champion in 2016.

Check Your Progress-3

a) In 1997, IBM's Deep Blue made history by defeating _____, the reigning world chess champion.
b) The ImageNet competition in _____ was a turning point for the rise of deep learning.
c) The term "machine learning" became widely recognized in the _____.
d) The first neural network model, the perceptron, was introduced by Frank Rosenblatt in 1958. (True/False)
e) The development of _____ in the 1980s marked a significant leap in neural network training.

1.6 LET US SUM UP

Machine learning (ML) is a subset of artificial intelligence. It is dedicated towards developing systems that are capable of continuous learning. This continuous learning further enhances the performance of machine learning systems via experience,

without the need of explicit programming. Some of the essential characteristics of machine learning are task classification. The system uses performance indicators like accuracy that allows it to improve over time as it keeps on learning new things. Moreover, machine learning systems depend on large datasets to gain their experience. These large datasets allow us to extract patterns and insights.

Early developments in machine learning, such the perceptron and Arthur Samuel's checkers-playing program, date back to the 1950s. The programs for playing checkers and perceptron were important turning points in the field. The invention of neural networks and the back-propagation algorithm in the 1980s rekindled interest in machine learning. The 1990s saw the development of Support Vector Machines (SVM) and other cutting-edge methods. In 2012, deep learning gained prominence after a convolutional neural network won the ImageNet competition, thanks to large data and greater processing capacity.

1.7 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a True
1-b False
1-c False
1-d True
2-a Data
2-b Data collection
2-c False
2-d False
2-e True
3-a Garry Kasparov
3-b 2012
3-c 1950s
3-d True
3-e backpropagation

1.8 ASSIGNMENTS

- Explain the concept of machine learning.
- How does ML differ from traditional rule-based systems?
- Provide examples of ML systems to illustrate its practical applications in various industries.
- Discuss the essential features of machine learning. How do these features shape the learning process of a model?
- Highlight key milestones and breakthroughs that have shaped the field of ML over time.
- Explore machine learning applications in a field of your choice such as healthcare, finance, or manufacturing. Analyze how machine learning has improved processes, decision-making, or operational efficiency in that domain.

Unit-2:Techniques of Machine Learning 2

Unit Structure

2.0 Learning Objectives

2.1 Introduction

2.2 Supervised Learning

2.3 Unsupervised Learning

2.4 Semi Supervised Learning

2.5 Reinforcement Learning

2.6 Let us sum up

2.7 Check your Progress: Possible Answers

2.8 Assignments

2.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand different types of Machine Learning technique.
- Understand the concept of Supervised Learning.
- Understand the concept of Unsupervised Learning.
- Understand the concept of Reinforcement Learning.

2.1 INTRODUCTION

Depending on the nature of the task, the data available, and the learning process machine learning algorithms are classified into several type. Often the machine learning techniques are classified into three major categories: Supervised Learning, Unsupervised Learning and Semi Supervised Learning. Figure 2.1 shows the classification of machine learning techniques.

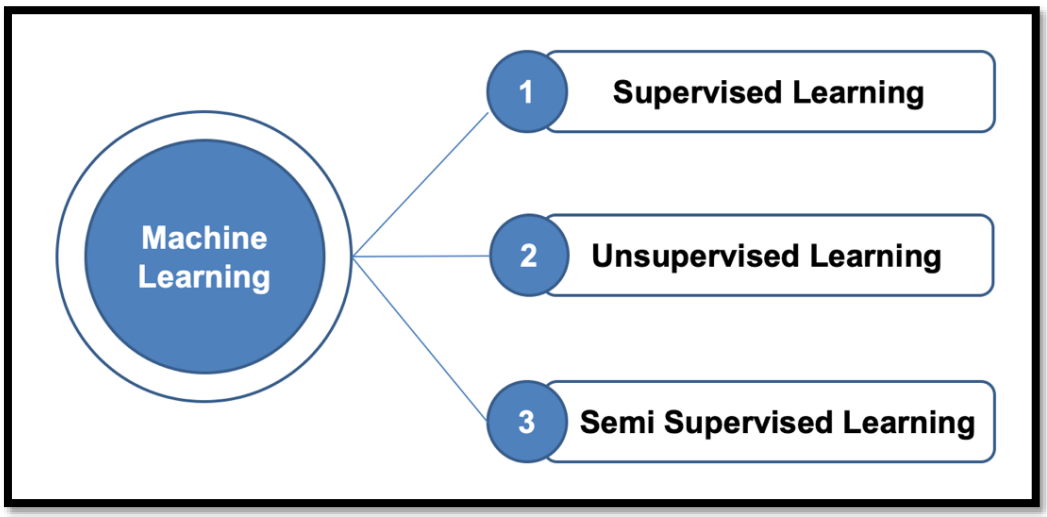


Figure 2.1: Machine Learning Techniques

In this chapter we will have a look at all these machine learning techniques to understand the basic working fundamentals used in each of them.

2.2 SUPERVISED LEARNING

Supervised learning is a technique of machine learning in which an algorithm is trained using labeled data. The model learns from the data by identifying patterns between the input features and known output labels. As machine solely depends on the given data for its training, the data provided for training should be correct and should not contain any false data.

Once trained, the model can use its prior knowledge to predict or categorize new, previously unknown data. The term "supervised" refers to the fact that the learning process is directed by the labels included in the training data.

Let's try to get an idea of how supervised learning works. Assume you are learning to tell the difference between a sparrow and a parrot. The process starts with your teacher showing you a picture of a sparrow and saying, "This is a sparrow". The teacher then again takes a picture of a parrot and says "This is a parrot". You look at several such photographs of sparrows and parrots (labels) until you can tell which is which. Later, if someone shows you a photo of a bird you have never seen before, you will be able to tell if it is a sparrow or a parrot based on what you learned when you were being trained. Figure 2.2 shows the flow diagram of Supervised Learning process.

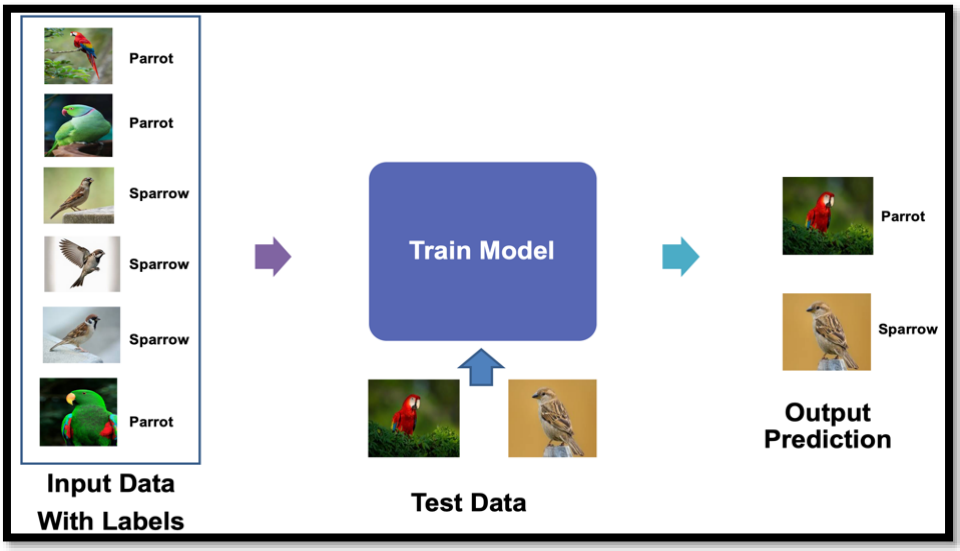


Figure 2.2: Flow Diagram of Supervised Learning

Supervised learning enables a computer to learn same things again and again by training it on labeled examples and allowing it to make estimates about new objects it has never seen before.

Supervised learning can be further categorized in two: **classification** and **regression**. Let us have a look at each of these categories.

Classification

In classification, the model learns to divide data into distinct groups or categories based on some predefined labels.

Example: Imagine you have a basket filled with different fruits, like apples, bananas, and oranges. You teach the computer to classify each fruit into categories based on its type, shape, size or color. Figure 2.3 explains the process of classification.

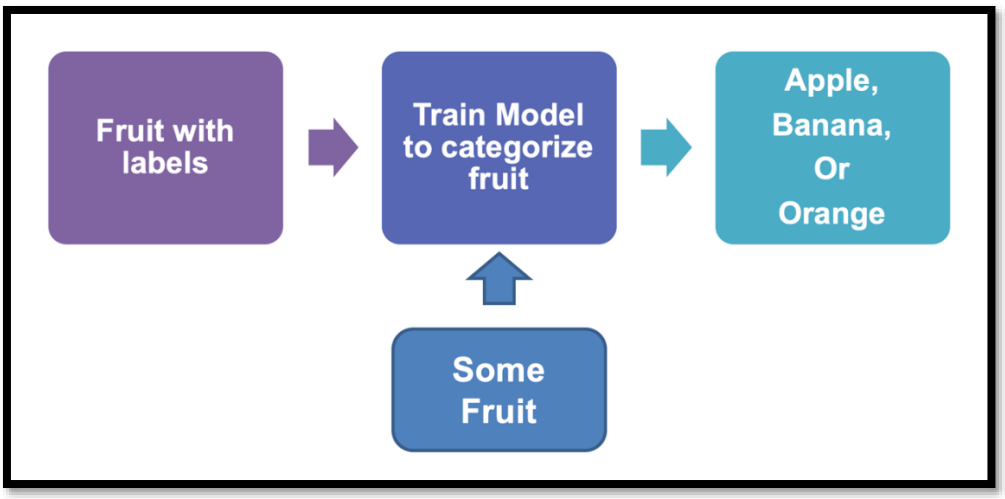


Figure 2.3: Example of Classification

Regression

In regression the model predicts a continuous value, such as a number or a range.

Example: Suppose you keep record of your age and height that you had in that age. Now you want to estimate your future height based on past data. Regression uses the past data to make predictions about your height over time. Figure 2.4 explains the process of regression.

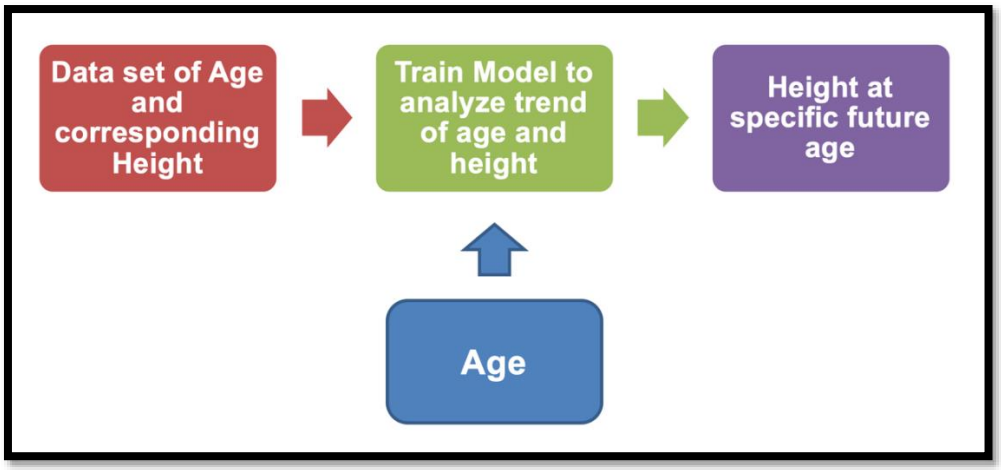


Figure 2.4: Example of Regression

2.3 UNSUPERVISED LEARNING

Unsupervised learning is a machine learning technique in which the model is fed data with no labels. The model examines the data to discover hidden patterns or groups on its own. This method is effective for clustering similar data points or discovering correlations in data without particular prior instructions.

Imagine that kid is given a big pile of mixed LEGO blocks of different colors, sizes and shapes. No one tells the kid what to do with them, so the kid decides to group all the blue blocks of a particular shape in one pile, red blocks into another pile, and green ones in yet another pile. The kid organized the blocks based on the color and shape as they seemed similar without any instructions given. Figure 2.5 shows the flow diagram of unsupervised learning process.

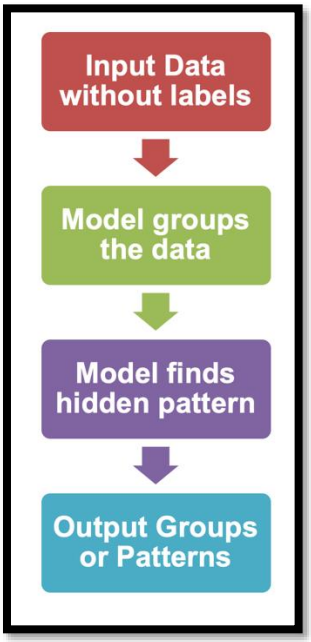


Figure 2.5: Flow Diagram of Unsupervised Learning

Thus, in unsupervised learning the computer groups the data which look similar without being told what to look for.

Unsupervised learning can be further categorized in two: **clustering** and **association**. Let us have a look at each of them.

Clustering

Clustering is an unsupervised learning type where the model organizes similar data points into groups based on some shared features.

Example: Imagine that you have a big pile of crayons without proper labels, and you decide to group them. The options that you have is size, shape or color. Assume that you group them using similar colors, like putting all the blue crayons together, all the red crayons together, and so on. Clustering allows us to group data by similarities, even without knowing or having the exact labels. Figure 2.6 shows the example of clustering.

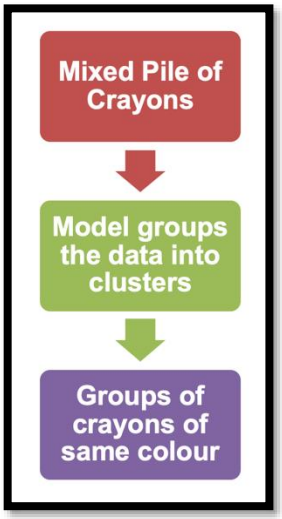


Figure 2.6: Example of Clustering

Association

Association tries to uncover the relationships or patterns between different items within a dataset.

Example: Imagine that you have a stationary store. You want to understand the buying habits of your customers. You start observing your customers and notice that the customer who buys pencil may also buy a notebook, an eraser or a scale. After many such observations it becomes clear that a customer that buys pencil often buys a notebook. This buying pattern of the customers helps you decide to place the pencil and notebooks closer together. Alternatively, you can offer discounts on both these items to encourage sales of these products. This example illustrates what association does: it identifies the patterns between items that frequently appear together. Figure 2.7 shows the example of association.

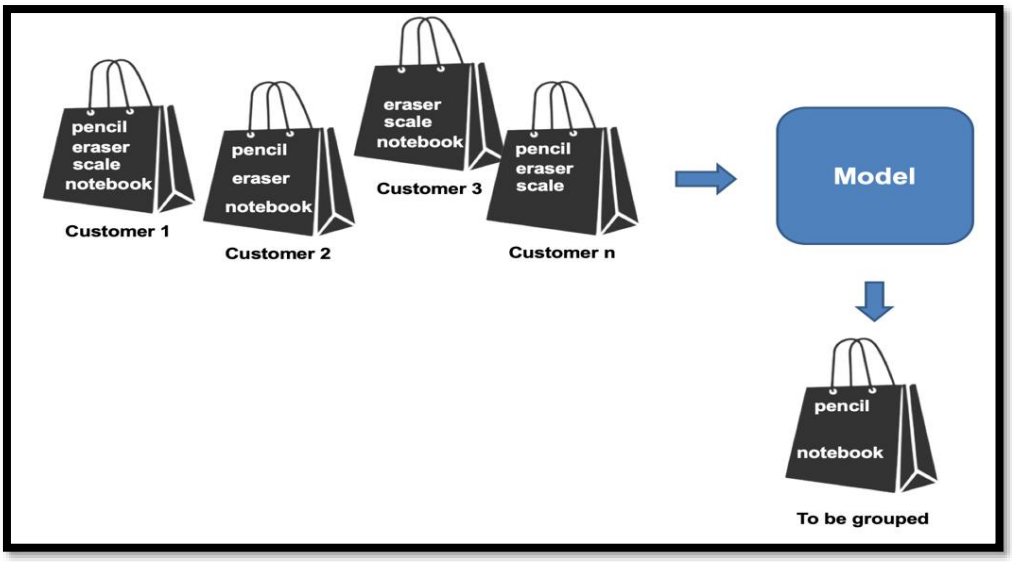


Figure 2.7: Example of Association

Check Your Progress-1

a) In supervised learning the models learn from unlabeled data. (True/False)
b) Regression comes under supervised learning algorithms. (True/False)
c) Unsupervised learning finds patterns in unlabeled data. (True/ False)
d) Association is a supervised learning method that uses different rules to find relationships between variables in a given dataset. (True/False)
e) In a supermarket, finding that customers who buy milk also buy cookies is an example of Classification. (True/False)

2.4 SEMI SUPERVISED LEARNING

Semi-supervised learning as the name suggests is a machine learning technique that uses both unsupervised and supervised learning techniques to obtain the outcome. Acquiring huge volumes of unlabeled data is reasonably easy, but obtaining a substantial amount of labeled data is excessively difficult or expensive at times. In order to create a model, the semi-supervised learning approach uses a little amount of labeled data in addition to a larger pool of unlabeled data. The semi-supervised learning approach can outperform an unsupervised technique in terms of accuracy. It uses the restricted labeled data to direct the model's learning process and then uses the unlabeled data to improve its generalization.

Example: When discussing supervised learning we had talked about image classification. Assume, that you might have a very small set of labeled images (e.g., 200 labeled images of parrots and sparrows) and a much larger set of unlabeled images (e.g., 15,000 images). A semi-supervised learning algorithm can use the labeled images to initially train the model and then leverage the structure and patterns in the unlabeled images to refine the model's understanding of how to distinguish parrots from sparrows, ultimately improving performance even with the small labeled dataset. Figure 2.8 shows the flow diagram of Semi Supervised Learning process.

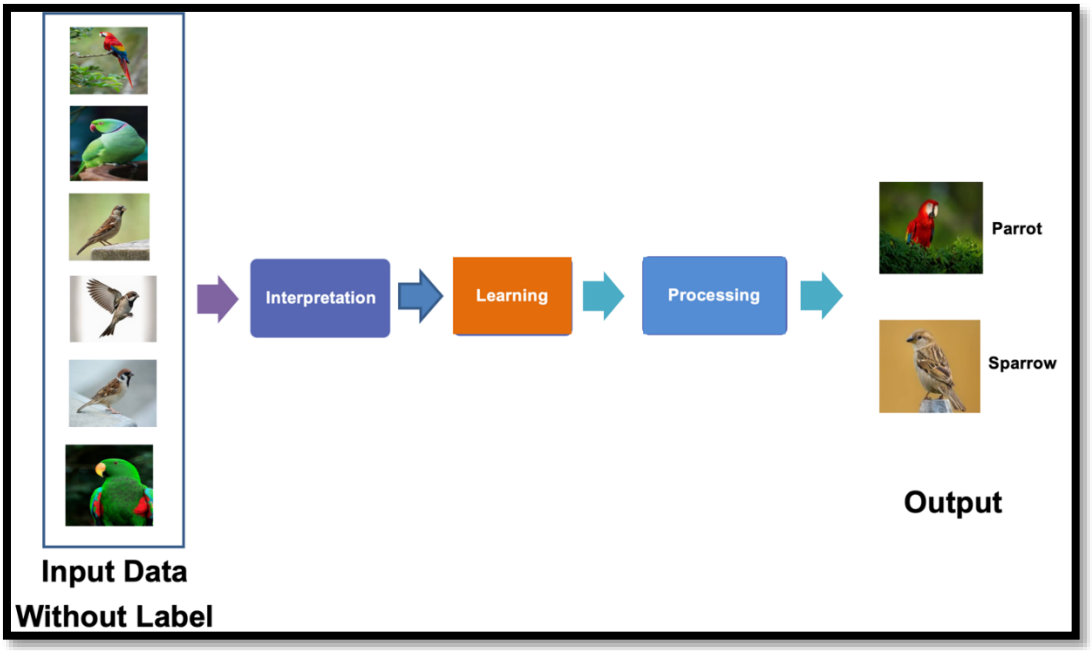


Figure 2.8: Flow Diagram of Semi Supervised Learning

2.5 REINFORCEMENT LEARNING

In machine learning reinforcement learning is a method where we have an agent that learns by interacting with an environment. The agent makes decisions based on certain situations, it then receives a feedback. This feedback is in the form of rewards or a penalty based on the actions taken by the agent. Over a period of time, the agent adjusts its decision-making strategy to maximize the rewards. Thus, it now becomes capable of learning the best actions to achieve a desired goal through trial and error.

Example: Let's say you're interested in learning how to ride a bike. You might initially stumble or fall a few times. But each time you try and stay balanced, you get appreciation from your parents, which makes you feel happy and encourages you to keep trying. If you fall, you know that you need to get up and try again, if you fall again now you know what not to do the next time. Slowly you will learn which movements keep you balanced and which makes you fall. Eventually, after some time you can ride confidently without falling.

Definition: Reinforcement learning is the process of choosing actions that maximize rewards to achieve a goal based on learned experiences.

Let us have a look at another example, imagine you're playing a game where you help a virtual puppy reach a treat in a maze. Every time the puppy goes the right way, it gets closer to the treat and earns a "point." But if it goes the wrong way or hits a wall, it doesn't earn any points, or it may even lose points.

The puppy doesn't know the maze at first, so it tries different paths. Over time, it remembers which moves helped it get closer to the treat and avoids moves that didn't work. The more it plays, the better it gets at picking the right moves because it remembers past actions and the points it got.

In this way, decision-making in reinforcement learning is like helping the puppy learn to reach the treat by choosing the best path, one step at a time, based on what happened after each choice. Figure 2.9 shows the example of the above decision making.

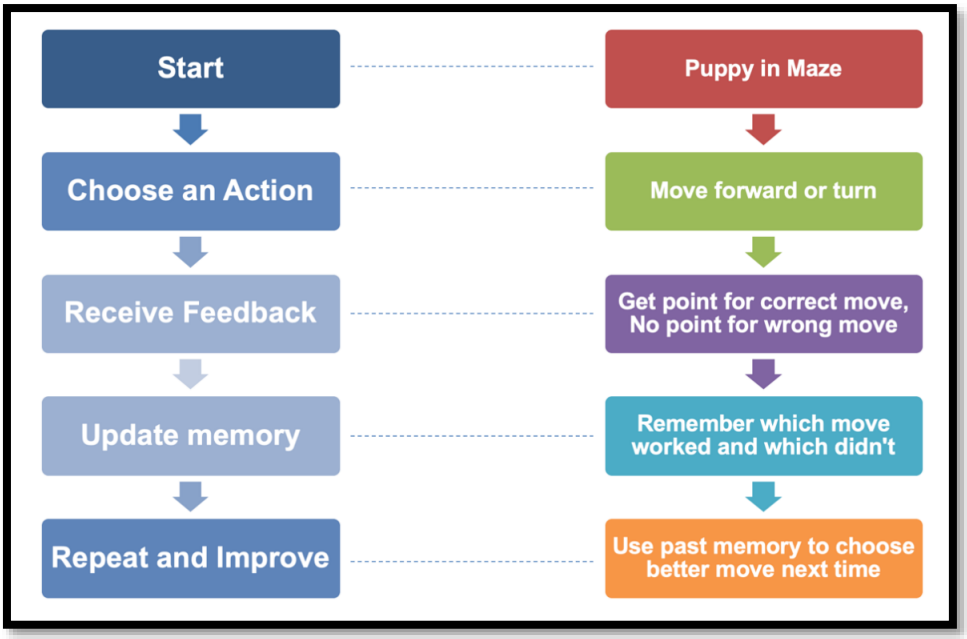


Figure 2.9: Flowchart of decision making

Check Your Progress-2

a) Semi-supervised learning uses only labeled data to train models. (True/False)

b) Semi-supervised learning can outperform unsupervised learning in terms of accuracy by leveraging a small amount of labeled data alongside a larger pool of unlabeled data. (True/False)

c) In reinforcement learning the feedback is given in the form of rewards or penalties. (True/False).

d) In reinforcement learning, the agent learns by receiving feedback only when it takes the right actions. (True/False)

e) Reinforcement learning can be compared to learning how to ride a bike, where you gradually learn the right movements through feedback, such as falling or staying balanced. (True/False)

2.6 LET US SUM UP

In this chapter we learnt about different techniques of machine learning. Supervised learning is a machine learning method where models are trained using labeled data to recognize patterns between inputs and outputs. Supervised learning includes two main types: classification and regression.

Unsupervised learning uses data without labels, allowing the model to discover patterns on its own. This approach helps find natural groupings or associations within the data. Unsupervised learning includes clustering and association.

Semi-supervised learning combines both supervised and unsupervised learning techniques, using a small amount of labeled data and a larger amount of unlabeled data. This approach helps improve model accuracy and generalization.

In reinforcement learning, a model (or agent) learns by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal is to maximize rewards through a process of trial and error.

2.7 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a False
1-b True
1-c True
1-d False
1-e False
2-a False
2-b True
2-c False
2-d False
2-e True

2.8 ASSIGNMENTS

- Explain the concept of supervised learning and give an example.
- What are the two main types of supervised learning, and how do they differ from each other? Provide examples for both.
- Describe the process of unsupervised learning. How does it differ from supervised learning? Include an example to support your explanation.
- What is the role of clustering in unsupervised learning? Provide a real-world example where clustering would be applied.
- In reinforcement learning, how does decision-making work? Illustrate with an example of how an agent learns to make better decisions over time.

Unit-3: Deep Learning and Algorithms

3

Unit Structure

3.0 Learning Objectives

3.1 Overview of Deep Learning

3.2 Five Essential Features of Deep learning

3.3 Neural Network Basics

3.4 Key Algorithms and Architecture

3.5 Let us sum up

3.6 Check your Progress: Possible Answers

3.7 Assignments

3.0 LEARNING OBJECTIVE

After studying this unit students should be able to:

- Comprehend the core principles of Deep Learning
- Identify the five essential features of Deep Learning
- Recognize the distinctive characteristics of Deep Learning
- Understand the fundamentals of Neural Networks
- Explore significant algorithms and architectures in Deep Learning

3.1 INTRODUCTION

One well-known definition of Deep Learning is provided by Ian Goodfellow, Yoshua Bengio, and Aaron Courville in their book *Deep Learning* (2016).

“Deep learning is a type of machine learning that uses multiple layers to progressively extract higher-level features from raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human, such as digits or letters or faces.”

Artificial intelligence that employs neural networks to replicate the human brain's learning process through experience is referred to as Deep Learning. Several layers of interconnected nodes, or "neurons," are used to train models in order to find patterns and make decisions. These networks are particularly skilled at tasks like audio and image recognition, natural language processing, and autonomous driving because they can independently extract and learn complex features from large datasets. Unlike traditional machine learning models that usually require human feature extraction, deep learning networks learn features in a hierarchical fashion, starting with simple elements like lines and curves and working their way up to complex patterns.

Consider a deep learning network that can identify cats in photos. The model first recognizes fundamental components like edges and textures. As it examines more data, it then begins to recognize increasingly intricate traits, like a cat's eyes, ears, and whiskers. The model develops a comprehensive understanding of cat appearance by combining these characteristics on multiple levels, finally attaining high accuracy in

distinguishing cats from other species. Figure 3.1 shows the workflow of a Deep Learning System.

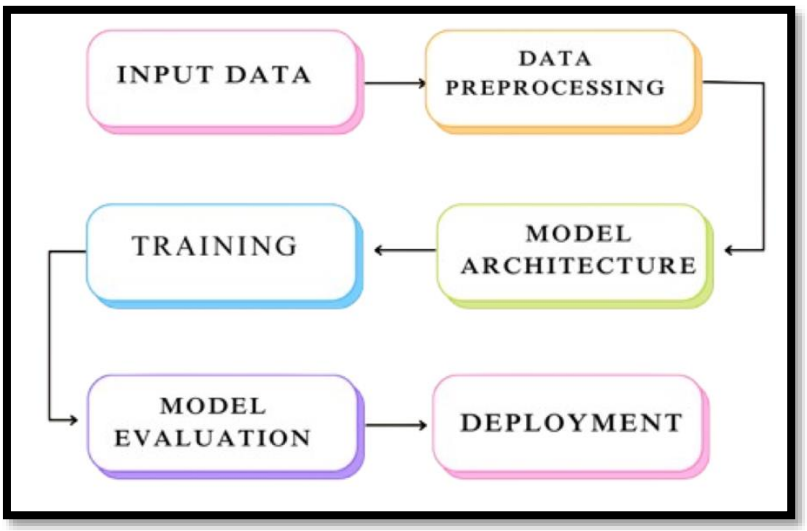


Figure 3.1: Workflow of Deep Learning System

Let us try and associate each component shown in figure 3.1 with a real-life example.

Input Data: The model initiates with input data, specifically. Assume that we have thousands of images of apples and oranges in different colors, shapes, and sizes.

Data Preprocessing: The images have been cleaned and formatted for model comprehension. This may require resizing the photographs, turning them to grayscale, or normalizing pixel values. Assume that we have images of different sizes, we resize them so that the model can process them uniformly.

Model Architecture: The Deep Learning model is built with layers of neurons that learn to extract features from the given images. A common architecture for images is a Convolutional Neural Network (CNN). Assume that the first layer learns about simple features like edges or colors, while deeper layers learn complex features like the shape of the fruit.

Training: The model acquires knowledge by examining the data and generating predictions. When the forecast is mistaken, the model modifies its internal parameters (weights and biases) through a method known as backpropagation to reduce inaccuracies. Assume that the model guesses whether an image is an apple or an

orange. If the output generated is wrong, it learns from this mistake and tries to improve its accuracy the next time.

Model Evaluation: After training, the model is tested on new images to see how well it has learned to identify apples and oranges. The performance is measured using metrics like accuracy or error rate.

Deployment: Once the model performs as per the required expectations, it is deployed in a real-world application. It can then be used to classify images of apples and oranges uploaded by users. The model may be used in a mobile application where users take a picture of a fruit, and gets an output indicating if the uploaded image is of an apple or an orange.

3.2 FIVE ESSENTIAL FEATURES OF DEEP LEARNING

Deep learning can be used in a wide variety of applications such as image recognition, natural language processing, finance, text to image conversions, chatbots and code generators, digital assistants and many more. The five essential features that allows use of Deep Learning in such applications are as mentioned:

1. **Automated Feature Extraction:** The deep learning models find and remove crucial characteristics from unprocessed data by themselves, therefore doing away with manual feature engineering.

   Example: In first layers, a deep learning model finds simple characteristics like edges and textures in image classification; as it goes further, it identifies more intricate ones like shapes and patterns. The model eventually identifies things, such separating a cat from a tree.
2. **Capacity to Manage Unstructured Data:** Deep learning particularly performs well in handling unstructured data including text, audio, and images; conventional machine learning methods sometimes find this difficult.

   Example: Speech recognition systems based on deep learning can convert spoken words into text while managing noise, accents, and different speech patterns, hence making them more robust than previous approaches.

3. **Scalability with Extensive Datasets:** Deep learning models' accuracy gets better with data volume. Learning much from the wealth of knowledge, they may handle huge amounts of data.

   Example: Self-driving vehicles employ deep learning to scrutinize millions of driving photos and videos. This enables the vehicle to enhance decision-making, such as identifying pedestrians or reading traffic signs.

4. **Comprehensive Learning:** By handling end-to-end chores in one process, deep learning models lessen the demand for distinct stages or components.

   Example: For instance, in a language translation system, a deep learning model can immediately convert a sentence from one language to another without requiring intermediary processing stages, hence enhancing the accuracy and fluidity of the translation.

5. **Hierarchical Feature Acquisition:** Building on more and more abstract feature levels, deep learning models learn difficult data patterns. Starting from basic components, the model gradually identifies more advanced structures.

   Example: In facial recognition, initial layers acquire fundamental traits such as eyes and mouth, whereas subsequent layers discern the configuration of these elements to identify an individual's face.

Check Your Progress-1

a) Key advantage of Deep Learning over traditional machine learning is that it handles unstructured data effectively. (True/False)
b) Backpropagation method is used to adjust weights and minimize errors. (True/False)
c) In a voice recognition system, the model learns to convert speech into _____.
d) The ability of deep learning to work well with large datasets is called _____.
e) The process of preparing data for a model, such as resizing or normalizing images, is called _____.
f) Deep learning is particularly effective at handling _____ data like images and sound.

3.3 NEURAL NETWORK BASICS

In neural networks, the computers are programmed to evaluate data as human brains would. Deep learning is a machine learning (ML) technique that uses layered architectures of connected nodes or neurons to imitate the human brain. It sets an evolutionary model whereby computers can always improve upon their faults. Artificial neural networks therefore try to solve difficult problems—face recognition or document summarization, for example—with improved accuracy.

How do neural networks work?

The cells of the human brain, called neurons, create a sophisticated, highly interconnected network and pass electrical signals to assist information processing. An artificial neural network consists of artificial neurons that collaborate to address a problem. Whereas artificial neural networks are algorithms fundamentally using computers systems to perform mathematical calculations, artificial neurons are software parts called nodes.

Basic neural network architecture

A fundamental neural network architecture comprises of interconnected artificial neurons organized into three layers. The basic structure of a neural network is shown in figure 3.2.

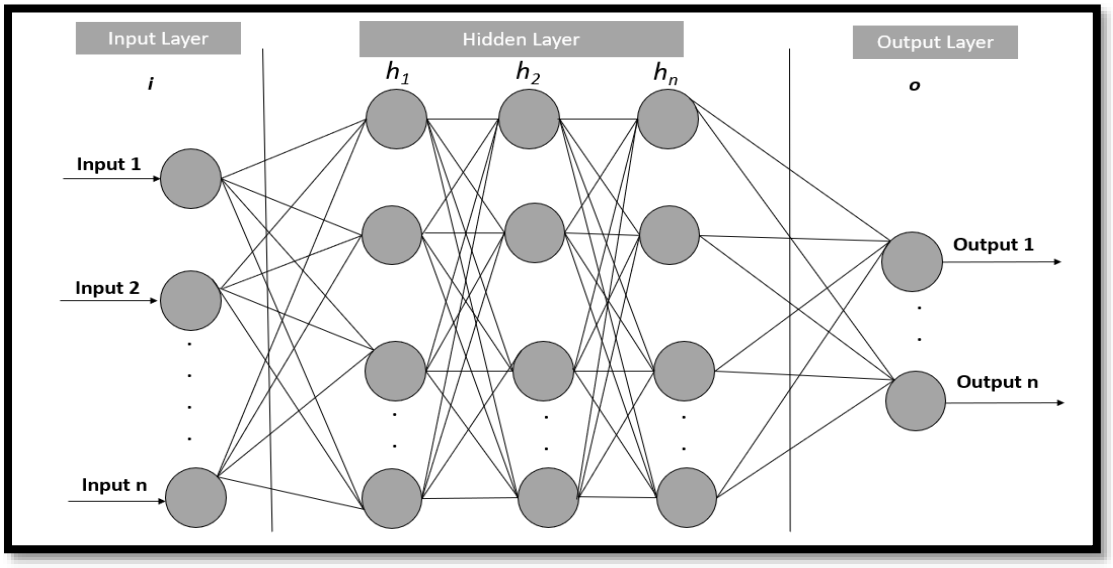


Figure 3.2: Basic Structure of Neural Network

Input Layer: External information is received by the artificial neural network through the input layer. Input nodes process, analyze, or categorize data and transmit it to the subsequent layer.

Hidden Layer: Hidden layers receive their input from either the input layer or other hidden layers. Artificial neural networks may possess numerous hidden layers. Each concealed layer evaluates the output from the preceding layer, processes it further, and transmits it to the subsequent layer.

Output Layer: The output layer gives the definitive result of all artificial neural network data analysis. It could have several nodes or a solitary one. The output layer in a binary classification problem will comprise only one output node generating results either 1 or 0. In a multi-class classification issue, the output layer may contain many output nodes.

Example: Anticipating Student Success or Failure

Input Layer: The input layer acquires data pertaining to the student's study habits and attendance.

- **Feature 1:** Study hours (e.g., 10 hours)
- **Feature 2:** Attendance rate (e.g., 90%)

Hidden Layer: The hidden layer analyzes the input data and discerns the correlation among study hours, attendance, and exam success.

- **Node 1:** Executes computations (weighted sum) and implements an activation function (e.g., ReLU).
- **Node 2:** Executes analogous functions with varying weights and biases.

Output Layer: The output layer provides the conclusive conclusion (pass/fail).

- **Node 1:** The output node produces a result of either 1 (Pass) or 0 (Fail).

Architecture of deep neural networks

Deep learning networks, also called deep neural networks, have many hidden levels each made of millions of linked artificial neurons. One node's connections to another are shown by a numerical value called weight. The weight is positive if one node stimulates another, negative if one inhibits the other. Other nodes are more affected by nodes with more weight values.

In theory, deep neural networks may associate any type of input with any type of output. Nevertheless, they require significantly more training than other machine learning techniques.

Check Your Progress-2

a) The input Layer in a neural network receives the raw input data. (True/False)
b) The output layer in a deep neural network typically has most number of neurons. (True/False)
c) A neural network with more than one hidden layer is often referred to as a _____ neural network.
d) The final layer in a neural network is known as the _____ layer, which generates the output for the classification or regression task.
e) In a deep neural network if the network has more layers then it is more likely to underfit the data. (True/False)

3.4 KEY ALGORITHMS AND ARCHITECTURE

Many different neural network architectures and the algorithms driving them have greatly improved deep learning. Thanks to these algorithms and structures, machines can carry our activities including image recognition, natural language processing, and speech recognition. The primary categories of neural networks are as follows:

Feedforward Neural Networks (FNN): Feedforward Neural Networks (FNN) are the most basic form of neural network, characterized by unidirectional data flow—from input to output—traversing through hidden layers without any feedback loops.

Characteristics:

- Data progresses unidirectionally: from the input layer to the output layer.
- Each layer exclusively obtains information from the preceding layer.
- Frequently employed for classification and regression tasks.

Example: Envision a basic neural network engineered to forecast a customer's likelihood of purchasing a product based on age and wealth.

- **Input Layer:** Age, Income
- **Hidden Layer:** Analyzes features through the application of weights and activation functions.
- **Output Layer:** Determines the likelihood of the client purchasing the product (Yes or No)

Convolutional Neural Networks (CNNs): Convolutional Neural Networks (CNNs) are engineered to process grid-structured data, including pictures and time-series data. Convolutional Neural Networks (CNNs) employ convolutional layers that utilize filters (kernels) on input data to identify patterns such as edges, textures, and forms.

Characteristics:

- Primarily utilized for image processing, video analysis, and time-series analysis.
- Comprises layers such as convolutional layers, pooling layers, and fully linked layers.
- Facilitates the extraction of spatial hierarchies from data (e.g., edges, forms, objects).

Example : For image classification, CNNs categorize photos into classes such as "cat" or "dog" by utilizing filters to identify elements like ears, eyes, and nose within the image.

- **Input Layer:** A 64x64 pixel picture**.**
- **Convolutional Layers:** Extract features, including edges and textures, from the image.
- **Pooling Layers:** Decrease dimensionality while preserving essential features.
- **Fully Connected Layers:** Classify the image according to retrieved features.

Recurrent Neural Networks (RNNs): Recurrent Neural Networks (RNNs) are designed for sequential or time-series data. In contrast to Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs) possess feedback connections that facilitate the retention and transmission of information throughout the network over time.

Characteristics:

- Utilized for time-series analysis, linguistic modeling, speech recognition, and sequential forecasting.
- Can retain prior inputs owing to their feedback linkages.
- Frequently experiences vanishing or inflating gradients; however, technologies such as LSTM (Long Short-Term Memory) assist in alleviating this problem.

Example: An RNN can anticipate the subsequent word in a sentence based on preceding ones.

- **Input Layer:** A sequence of lexemes (e.g., "I am proceeding to the").
- **Hidden Layers:** The network retains prior words and modifies the prediction for the subsequent word.
- **Output Layer:** Forecasts the subsequent word in the sequence (e.g., "market").

Long Short-Term Memory Networks (LSTMs): LSTM is a variant of RNN engineered to address the vanishing gradient issue, enabling the network to acquire long-term dependencies in sequential input.

Characteristics:

- Engineered to retain knowledge over extended durations, a task at which conventional RNNs falter.
- Utilizes memory cells and gates (input, forget, and output) to regulate the flow of information.
- Proficient in activities such as machine translation, speech recognition, and time-series forecasting.

Example : LSTMs are frequently employed in machine translation. For instance, translating the phrase "I am going to the market" from English to French:

- **Input Layer:** A series of words in English.
- **LSTM Layers:** Analyzes the sequence while retaining long-term dependencies.
- **Output Layer:** Produces the translation in French ("Je vais au marché").

Generative Adversarial Networks (GANs): Generative Adversarial Networks (GANs) comprise two neural networks; the generator and the discriminator that engage in competition. The generator produces synthetic data, while the discriminator assesses the authenticity of the data as either real or fabricated. In this adversarial process, the generator acquires the ability to provide progressively realistic data.

Characteristics:

- Consists of two networks that are concurrently trained: the generator and the discriminator.
- Utilized for the generation of authentic data, including photos, music, or text.
- Utilized in domains such as deepfake production, picture generation, and artistic synthesis.

Example: Generative Adversarial Networks (GANs) can produce lifelike representations of non-existent individuals.

- **Generator:** Produces an image of an individual.
- **Discriminator**: Assesses if the image is authentic (sourced from a dataset of real individuals) or fabricated (produced by the network).
- **Output:** Over time, the generator enhances its capabilities and produces very lifelike visuals.

Radial Basis Function Networks (RBFNs): Radial Basis Function Networks (RBFNs) employ radial basis functions as its activation functions. These networks are mostly utilized for classification, regression, and function approximation applications.

Characteristics:

- The hidden layer employs radial basis functions, such as Gaussian functions, to calculate activations.

- Frequently employed for pattern recognition, function approximation, and time-series forecasting.
- May exhibit greater efficiency than alternative network types in specific situations

Example: Radial Basis Function Networks (RBFNs) may classify data points according to their similarity.

- **Input Layer:** Characteristics of data points (e.g., dimensions of an item).
- **Hidden Layer:** Assesses the resemblance of input data points to established patterns.
- **Output Layer:** Categorizes the data point into a certain class (e.g., "circle" or "square").

Autoencoders: Autoencoders are employed for unsupervised learning tasks, specifically for dimensionality reduction or feature extraction. They acquire the ability to compress (encode) input data into a more compact representation and subsequently reconstruct (decode) it to its original form.

Characteristics:

- Consists of an encoder and a decoder.
- Frequently employed for anomaly detection, data compression, or noise reduction.
- Operates with unsupervised learning tasks, indicating it does not necessitate labeled data.

Example : For anomaly detection, an autoencoder may be employed to identify atypical patterns in network traffic.

- **Encoder:** Compresses network traffic data into a reduced latent space.
- **Decoder:** Endeavors to reassemble the original data.
- **Output:** If the rebuilt data substantially deviates from the original data, it may signify an anomaly (e.g., a security violation).

Check Your Progress-3

a) Full Form of FNN is Fit Forward Neural Networks. (True/False)
b) LSTM is a type of RNN that is designed to address the vanishing gradient problem.(True/False)
c) Input layer of a neural network responsible for adjusting weights based on the error between predicted and actual values. (True/False)
d) Feedforward Neural Network (FNN) indicates Data flows only in one direction from input to output. (True/False)

3.5 LET US SUM UP

Deep learning is a branch of machine learning that employs artificial neural networks to model and address intricate issues. It allows machines to autonomously learn and enhance from experience by analyzing data through layers of interconnected neurons.

These networks incrementally derive advanced characteristics from raw input, rendering them suitable for applications including image recognition, speech processing, and natural language comprehension. In contrast to conventional machine learning, deep learning obviates the necessity for manual feature extraction, enabling models to independently identify patterns in extensive datasets, hence facilitating their efficacy in domains such as autonomous driving and AI-driven medical diagnostics.

A deep learning model often adheres to a workflow comprising multiple phases, commencing with the entry of raw data (e.g., pictures), succeeded by preprocessing to ready the data for analysis. The model architecture, often founded on Convolutional Neural Networks (CNNs) for image-related tasks, is designed to extract significant information across several layers. The model is trained by modifying its internal parameters through backpropagation, enhancing predictions by reducing mistakes.

The model is assessed using previously unencountered data following training to determine its performance. Upon success, it is implemented in practical applications, such as mobile applications for picture classification, allowing users to engage with the trained model.

Deep learning is characterized by automated feature extraction, proficiency in managing unstructured data, scalability with extensive datasets, and holistic learning within a singular process. These characteristics render deep learning especially proficient for tasks involving unstructured data, including images, text, and audio.

Neural networks, fundamental to deep learning, function via layers of artificial neurons that analyze information, acquiring knowledge from input data to generate output.

Various neural network architectures, such as Convolutional Neural Networks (CNNs) for image processing and Recurrent Neural Networks (RNNs) for sequential data, have transformed sectors by offering effective answers to challenges previously deemed too intricate for machines.

3.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a True
1-b True
1-c text
1-d scalability
1-e preprocessing
1-f unstructured
2-a True
2-b False
2-c Deep neural network
2-d Output layer
2-e False
3-a False
3-b True
3-c False
3-d True

3.7 ASSIGNMENTS

- What is deep learning and how does it differ from traditional machine learning?
- Describe the role of backpropagation in deep learning training. How does it help improve model accuracy?
- Explain input layer, hidden layer, and output layer in a neural network?
- What are Generative Adversarial Networks (GANs), and how do they generate realistic data?
- Explain the concept of hierarchical feature extraction in deep learning. Provide an example.

Unit-4: Applications of ML 4

Unit Structure

4.0 Learning Objectives

4.1 Real world Applications of Machine Learning

4.2 Issues in Machine Learning

4.3 Let us sum up

4.4 Check your Progress: Possible Answers

4.5 Assignments

4.0 LEARNING OBJECTIVE

After studying this unit students should be able to:

- Understand how machine learning is applied across different industries and sectors.
- Analyze real-world case studies demonstrating successful ML applications.
- Identify key challenges and limitations faced during the implementation of ML models.
- Evaluate potential solutions to address common issues in machine learning.

4.1 REAL WORLD APPLICATION OF ML

Many fields are transforming thanks to machine learning (ML), which allows systems to learn from data, detect patterns, and make judgments with little human assist. Here are some examples of real-world machine learning applications:

Healthcare

The use of machine learning in healthcare have changed the way we diagnose, cure and monitor patients. By generating more precise diagnoses and allowing more personalized therapies, this technology is revolutionizing healthcare research and results. The capacity of artificial intelligence and machine learning in medical to rapidly examine large quantities of clinical data assists doctors to spot disease markers and trends otherwise missed. From early radiological image scanning to forecasting outcomes from digital health records, the possible uses of machine learning in medicine are many and wide-ranging. Using artificial intelligence and machine learning in hospital settings and clinics, healthcare systems can be more intelligent, quicker, and more efficient in delivering millions of people across the world with care.

Example:

- Google's DeepMind developed an ML model that detects over 50 eye diseases from retinal scans with accuracy comparable to top ophthalmologists.
- ML models analyze MRI and CT scans to detect early signs of cancer and neurological diseases.

Impact:

- Faster and more accurate diagnoses.
- Reduction in human error.
- Early detection leading to better treatment outcomes.

Finance

By enhancing decision-making, risk management, and customer experience, machine learning is changing the finance scene. It is widely used for fraud detection by identifying anomalous transactions, enhancing credit scoring through more accurate assessments of financial behaviors, and driving algorithmic trading by analyzing market data for optimal trading strategies. In addition to enabling personalized customer experiences through sophisticated segmentation, ML assists in risk management by forecasting market, credit, and operational risks. Moreover, ML enhances portfolio management by looking at asset performance and forecasting market trends, hence enabling better financial forecast and more knowledgeable investment choices.

Example:

- PayPal uses ML to detect fraudulent transactions by identifying unusual patterns in user behavior.
- Banks employ ML algorithms to automate loan approval processes by analyzing credit history and transaction data.

Impact:

- Increased sales and user engagement.
- Higher customer satisfaction through personalized experiences.

Transportation

The sector of transportation has been transformed by the use of machine learning. By incorporating machine learning in transit, clever transportation systems are being developed to improve safety, performance, and sustainability. By optimizing traffic flow, estimating maintenance demands, and enhancing the general travel experience,

machine learning algorithms can examine huge amounts of data from different sources including sensors, cameras, and GPS.

Example:

- Tesla's Autopilot uses ML to analyze data from vehicle sensors and cameras, enabling self-driving capabilities.
- Google Maps predicts traffic congestion using real-time location data and ML models.

Impact:

- Enhanced safety and convenience in transportation.
- Reduced travel time through efficient route planning.

Manufacturing

Machine learning (ML) is revolutionizing manufacturing by enhancing efficiency, quality control, and predictive maintenance when it comes to operations. By examining data in real time, spotting bottlenecks, and improving supply chain management, ML algorithms help to improve production processes. By knowing ahead of time when machines will fail, predictive maintenance models can help lower maintenance expenses and downtime. Machine learning guarantees better output by helping in quality control by identifying during manufacturing any product flaws. By means of demand prediction and inventory control, ML helps companies to modify production schedules and maximize resource distribution for improved cost efficiency.

Example:

- Siemens uses ML to predict equipment failures by analyzing sensor data from machines, allowing for maintenance before breakdowns occur.

Impact:

- Reduction in downtime and maintenance costs.
- Improved operational efficiency.

Entertainment

The entertainment field uses machine learning (ML) to refine user experiences, streamline content production, and provide better suggestions. From generating lifelike animations to helping with scriptwriting and automating video editing and special effects, ML also significantly helps with content development. By means of machine learning, intelligent game characters and dynamic difficulty levels are built to produce more engaging experiences in gaming. Furthermore, ML is used in marketing to forecast audience engagement and customize advertising campaigns, hence increasing reach and earnings for media businesses.

Example:

- Netflix applies ML to suggest movies and TV shows by analyzing user viewing habits and ratings.

Impact:

- Increased viewer retention and engagement.
- Personalized viewing experience.

Education

Machine learning is reshaping education by customizing learning opportunities, raising student performance, and increasing administrative efficiency. ML algorithms are employed to build adjustable learning systems which customize material to students' learning styles and requirements, hence enabling them to learn at their own pace. Furthermore, it helps predict student performance, find people who might fall behind, and offer tailored therapies. It also helps teachers to concentrate more on instruction as ML can be used to automate administrative duties including scheduling and grading. ML also assists with curriculum planning and resource allocation optimization via data analysis.

Example:

- Duolingo uses ML to personalize lessons based on user progress and areas of weakness.

Impact:

- Improved learning outcomes through tailored content.
- Enhanced student engagement.

Check Your Progress-1:

a) Google DeepMind company developed an ML model that detects over 50 eye diseases from retinal scans. (True/False)
b) ML models allows early detection leading to better treatment outcomes for neurological diseases. (True/False)
c) The primary use of machine learning in PayPal's system is to personalize recommendations. (True/False)
d) In the _____ sector, ML algorithms are used to automate loan approval processes by analyzing credit history and transaction data.
e) ML can be used to predict traffic congestion using real-time location data. (True/False)

4.2 ISSUES IN MACHINE LEARNING

Even though machine learning (ML) has advanced significantly, there are still a number of obstacles to overcome. From data collection and model training to deployment and interpretation, these problems might occur at various phases of machine learning.

The list of typical machine learning problems is as mentioned:

1. **Insufficient or Poor-Quality Data:** The quality and quantity of data are critical to the performance of machine learning models. Insufficient data, noisy data, or imbalanced datasets can lead to inaccurate or biased models.

   Example: A model trained on a dataset of only 100 images of dogs and 100 images of cats will likely struggle to generalize well to new, unseen images, especially if the dataset doesn't include enough diversity (e.g., different breeds or lighting conditions).

2. **Overfitting and Underfitting:**

   Overfitting: When a model learns the training data too well, including its noise and outliers, it performs poorly on new, unseen data.

   Underfitting: When a model is too simple or doesn't capture the underlying pattern in the data, resulting in poor performance even on the training set.

Example: In a spam email classifier, overfitting could occur if the model memorizes specific words in the training set, leading it to misclassify new emails. Underfitting may happen if the model is too simplistic, ignoring important features like the structure of the email.

3. **Bias and Fairness:** If the data used to train a model is biased or unrepresentative of the. population, the model can produce biased predictions. Bias can result in unfair outcomes, such as discrimination against certain groups.
   Example: Even if two candidates are equally qualified, a recruiting algorithm trained on past data from a company that has typically recruited more men than women may make biased decisions that disadvantage the female candidates.

4. **Data Privacy and Security:** The machine learning models, especially those dealing with sensitive personal data, can present privacy concerns. Ensuring that models do not leak private information from training data is a critical issue.
   Example: A model trained on health data could unintentionally reveal information about individuals' health conditions, even if those conditions are not directly included as features in the model.

5. **Model Interpretability and Explainability:** Most of the machine learning models, especially the deep learning models, function as "black boxes". It is thus difficult to understand why a certain decision or prediction was made. Thus, proper explanation of how the model interpret data and provides result is required**.** It may so happen that the lack of interpretability and explainability in machine learning models can hinder trust and adoption, particularly when it’s difficult to understand why a decision was made.
   Example: In a medical diagnosis system, if a model incorrectly diagnoses a patient with cancer, it might be challenging to understand which features led to this decision, hindering trust in the system and its adoption.

6. **Scalability and Computational Resources:** The machine learning algorithms usually require significant computational resources, especially the deep learning models. Training a deep learning model on a large dataset of images may require

hardware like Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs).This can be a bottleneck at times as we usually work with large datasets or in resource-constrained environments.

Example: Training a deep neural network for image recognition on a large dataset of millions of images can take days or weeks, requiring expensive hardware like GPUs or TPUs.

Let us have a look at a simple case study that illustrates how the mentioned issues may appear in real-world scenarios.

Case Study:

A real estate company aims to create a machine learning model to predict house prices based on features such as square footage, number of bedrooms, location, and age of the property. After carefully looking at the dataset they had, following issues were identified and certain solutions were proposed to solve them as mentioned:

Data Quality

Problem: It was observed that some features, like the number of bathrooms and renovation year, had missing values, potentially affecting the model's accuracy.

Solution: The company addressed this by filling in missing values with the mean or median for each respective feature, ensuring data completeness.

Feature Selection

Problem: It was observed that irrelevant features, such as the house color or roofing type, were included in the dataset and didn't contribute to price prediction.

Solution: The company used correlation analysis to eliminate irrelevant features, focusing on important ones like square footage and location.

Overfitting

Problem: A complex decision tree model was performing well on training data but struggled to generalize on new data.

Solution: To counter this, the company reduced the tree’s depth and implemented cross-validation to improve generalization on unseen data.

Model Interpretability

Problem: The company needed a model that could explain the reasoning behind predicted house prices.

Solution: They selected a linear regression model for its simplicity and transparency, enabling easy explanation of how features like square footage influenced price predictions.

Conclusion:

This case study highlights that even straightforward machine learning projects can encounter issues such as missing data, irrelevant features, overfitting, and interpretability concerns. Through the application of appropriate techniques, the company was able to create an effective and practical house price prediction model.

Check Your Progress-2

a) Bias arises when a machine learning model learns the training data too well, including its noise and outliers. (True/False)
b) Poor quality data occurs when the dataset is not diverse enough, like having limited images of only a few breeds of dogs or cats. (True/False)
c) When a model is too simple or doesn't capture the underlying pattern in the data, it leads to _____.
d) Training a deep learning model on a large dataset of images may require hardware _____.
e) The lack of _____ and _____ in machine learning models can hinder trust and adoption, particularly when it's difficult to understand why a decision was made.

4.3 LET US SUM UP

By letting devices learn from data and make decisions with little human intervention, machine learning (ML) is transforming many sectors. Google DeepMind, for instance, has developed a model in healthcare able to identify more than 50 eye issues from retinal scans, therefore lowering human error, offering quick and accurate diagnoses.

By examining unusual user actions, ML algorithms in the financial industry find fake transactions and also help to streamline the loan approval process. Leading the field, businesses like Tesla and Google help the transportation industry benefit from autonomous vehicles and traffic prediction using machine learning. In a similar way, the manufacturing and entertainment industries utilize ML for predictive maintenance and custom content suggestions. Adaptive systems like Duolingo, which adapts lessons based on user advancement, also uses ML in education.

Although ML has great advantages, it also has major obstacles including problems with data quality, overfitting, underfitting, and biased forecasts, which might result in unfair or erroneous results. For example, a real estate firm developing a model to forecast home values ran into issues with overfitting, irrelevant characteristics, and incomplete data. By utilizing techniques such as filling in missing values, removing unimportant features, simplifying the model and going with a linear regression approach that made more sense, the company solved these difficulties. These strategies highlight the need of dealing with data problems, simplifying model complexity, and ensuring openness if machine learning initiatives are to be successful.

4.4 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a True
1-b True
1-c False
1-d finance
1-e True
2-a False
2-b True
2-c Underfitting
2-d GPUs or TPUs
2-e Interpretability and explainability

4.5 ASSIGNMENTS

- Try and find some additional applications areas in real life where machine learning can be implied.
- Discuss how machine learning is transforming healthcare, specifically focusing on the model developed by Google DeepMind. What are the potential impacts of this model on disease diagnosis, and how does it compare to traditional diagnostic methods?
- Explain the common challenges faced by companies when implementing machine learning models.
- Identify and explain at least three sectors where machine learning is being applied. For each sector, describe the problem ML is solving and the impact it has on the respective industry.

Block-2

Regression, Classification and Ensemble Methods

Unit-1: Introduction to Regression

1

Unit Structure

1.0 Learning Objectives
1.1 Overview of Regression
1.2 Regression Algorithms
1.3 Linear Regression
1.4 Polynomial Regression
1.5 Comparative Study
1.6 Let us sum up
1.7 Check your Progress: Possible Answers
1.8 Assignments

1.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the fundamental concept of regression along with its applications.
- Develop an understanding of various types of regression
- Implement linear and polynomial regression techniques.

1.1 INTRODUCTION TO REGRESSION

Ever thought how a child learns to recognize objects? Parent shows the child various pictures (input) of the object along with mentioning its name (output). With time, the child is able to recognize this object. This is exactly how supervised learning works. Here the model learns from the labelled data.

Regression is a type of supervised learning technique which is used to predict continuous values and modeling relationship between these values. Consider the case when the output variable is continuous like Sales of a product. Assume that we have a training set with values of two feature(s), TV advertising budget and the output as Sales figure of TV. Assume that we would like to learn a linear function that relates both these features. For a new value of TV budget, we will use the regression function to predict the sales figure of TV. Figure 1.1 shows an overview of the problem.

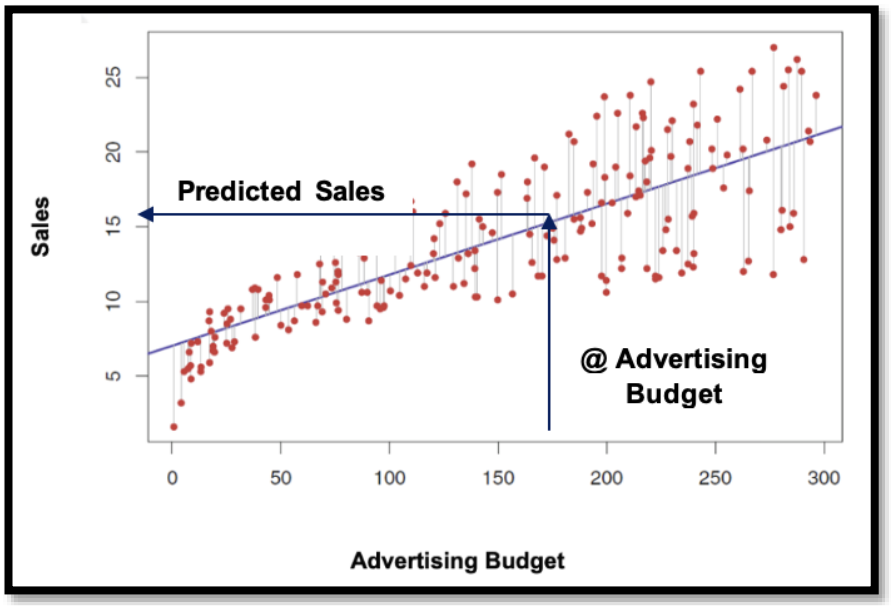


Figure 1.1: Sample Regression Line

Regression plays a pivotal role in understanding relationships and predicting outcomes. Further, it tends to give estimates for possible future sales, demand, or even market performance for worthwhile business planning.

Regression is a stepping-stone to the next level of modeling in advanced machine learning. Simplicity, stability, and the ability to be applied in many scenarios make regression a quintessential methodology for problem-solving and discoveries in various realms.

1.2 REGRESSION ALGORITHMS

Regression algorithms are a group of machine learning techniques used to predict the continuous outcome variables. These algorithms are commonly employed to analyze and model relationship between variables, where one or more input characteristics (independent variables) are utilized to predict the target characteristics (dependent variables).

Using regression algorithms, we are much able to adequately describe a mathematical function-a line or perhaps a curve-that best fits the data, to make predictions about new data and understand how different levels of input characteristics affect the target.

Regression algorithms cover a broad spectrum of application domains due to its accuracy in prediction for solving daily life challenges:

- **Finance:** Estimate stock prices, assess credit risk, and model socio-economics.
- **Healthcare:** Disease progression and approximations of patient out-turns and healthcare expenses.
- **Marketing:** Customer values; sales predictions analysis of trends in the market.
- **Engineering:** Predictive maintenance and minimize downtime.
- **Environment:** Forecasting weather conditions, modelling climate change, and pollution predictions.
- **Retail:** Demand forecasting and price setting.
- **Logistics:** Estimation of delivery times, given distance travelled, and reasoning.

Regression algorithms are a mathematical way to carry out regression analysis. Each algorithm has its own set of advantages and limitations. Some of the popular regression algorithms are as mentioned:

- Linear Regression
- Polynomial Regression
- Lasso regression
- Support Vector Regression
- Decision Tree
- Random Forest Regression

We will be focusing on the algorithms like linear regression and polynomial regression.

Check Your Progress - 1

a) What is the basis of supervised learning?
   a) Random data without labels
   b) Learning from unlabelled data
   c) Learning from labelled data where inputs and outputs are provided
   d) Learning without any examples

b) Primary objective of regression algorithms is
   a) Identifying clusters in data
   b) Predicting continuous outcomes and modeling relationships between values
   c) Reducing the size of datasets
   d) Matching data to predefined categories

c) Which of the following is an example of regression in supervised learning?
   a) Predicting whether an email is spam or not
   b) Forecasting the price of a house based on its size
   c) Classifying animals into different categories
   d) Grouping customers based on buying patterns

d) Why is regression widely used in problem-solving?
   a) Its complexity and unpredictability
   b) Its simplicity, stability, and versatility
   c) Its ability to classify categorical data
   d) Its focus on clustering similar data points

e) Which of the following is NOT an example of regression algorithm applications in healthcare domain?
   a) Predicting disease progression
   b) Forecasting patient outcomes
   c) Diagnosing diseases from X-rays
   d) Estimating healthcare costs

f) From the following, which is an application of regression algorithms in the logistics domain.
   a) Predicting market trends
   b) Estimating delivery times
   c) Modelling climate change
   d) Forecasting product demand

1.3 LINEAR REGRESSION

Linear regression is the most fundamental and widely used supervised learning technique. This technique analyzes the linear relationship between the dependent variable (target) and one or more independent variables (predictors).

Key concepts of Linear Regression are:

1. The General Equation of a Linear Regression Model: This model is the one in which the relationships among variables are represented with a straight line, given as:

$$\mathbf{y} = \beta_0 + \beta_1 x + \epsilon$$

Where,

y: Dependent variable (output)

x: Independent variable (input or feature)

β_0: Intercept of the line i.e. value of y when x=0

β_1: Slope of the line i.e. the rate of change of y with respect to x

ϵ: Deviations between actual values of output y and its predicted values.

The general linear regression model is represented as shown in figure 1.2.

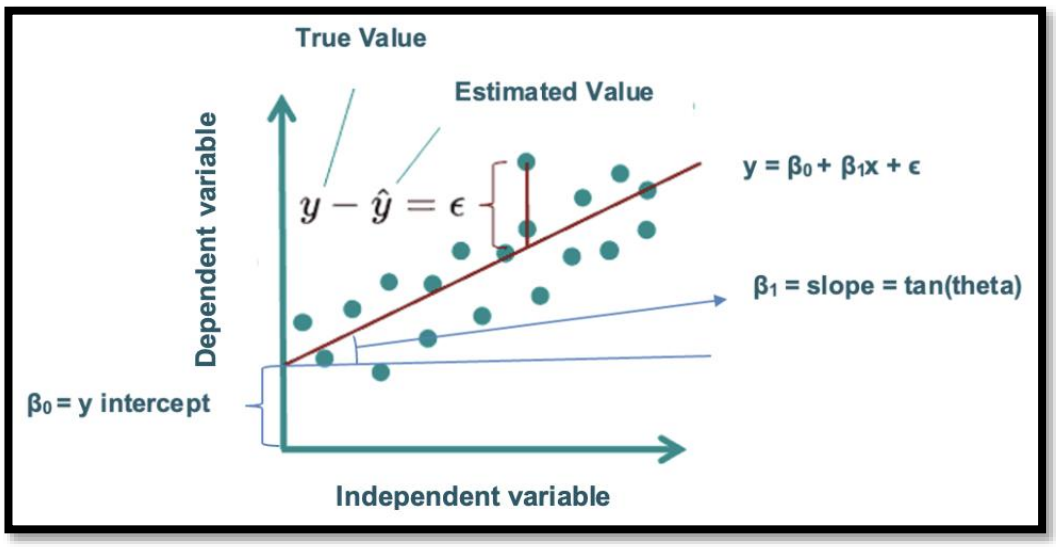


Figure 1.2: Linear Regression Line

The greater the linear relationship between the dependent and independent variables, the more the data points lie on a straight line as shown in figure 1.3.

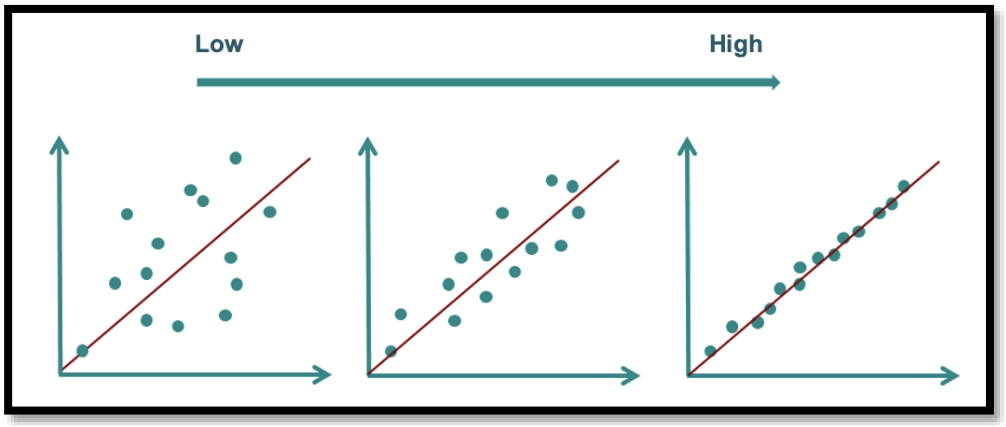


Figure 1.3: Relationship within Linear Regression Line

2. Cost Function: To optimize the model, linear regression uses the Mean Squared Error (MSE) as the cost function:

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

Here, y_i reflects the actual value, $\hat{y}_i$ is the predicted value, and n signifies the total number of observations. Reducing the cost function helps determine the best-fit line.

3. **Optimization Using Gradient Descent:** It is an optimization technique that minimizes the cost function by carrying out the updating of the parameters and iteratively.

Steps to be followed for performing Linear Regression are as mentioned:

1. **Data Gathering:** Gathering input and output variables for any relevant data concerning the problems you want to solve.
2. **Data Pre-processing:** Cleaning the data for replacing empty entries, removing outliers, and normalizing features forms the basis of pre-processing. The aim is creation of a clean data-set ready for analysis.
3. **Data Splitting:** The dataset is further divided into training and test data. The training data form is used to make the model learn, while the testing portion helps to know how well the model performs against unseen data.
4. **Train Model:** A linear regression model is fitted to the training set. The fit is computed for minimizing the cost function, which in this case is that of the mean squared error.
5. **Prediction:** The trained model uses new or unseen input data to predict an unseen or new output variable.
6. **Performance Evaluation:** Performance evaluation can be done with metrics like mean absolute error, root mean squared error, and R-squared values, which help understand how well a model defines the relationship among variables.

Example of Linear Regression:

Assume that we have a dataset pertaining to the size of the house and its price. We are going to estimate the price of the house based on the size given by the user.

Input data: Size of house

Output data: Predicted price of house

Step 1: Data Collection

Let's assume we have the data for house sizes and prices as shown in table 1.1.

House Size (sq ft)	Price (in lakhs Rs.)
1500	30
1700	34
2000	40
2200	44
2500	50

Table 1.1 Dataset of house size and price

Step 2: Formulate the Model

The simple linear regression model can be represented by the equation:

$$\mathbf{y} = \beta_0 + \beta_1 x$$

Where:

- y is the predicted price
- β_1 is the slope of the line (the change in price for each additional square foot)
- x is the house size
- β_0 is the y-intercept (the predicted price when house size is zero)

Step 3: Calculate Mean Values

First, we need to calculate the mean of both the house sizes and prices.

$$\text{Mean of } x(\bar{x}) = \frac{\sum x}{n} = \frac{1500 + 1700 + 2000 + 2200 + 2500}{5} = \frac{10800}{5} = 2160$$

$$\text{Mean of } y(\bar{y}) = \frac{\sum y}{n} = \frac{30 + 34 + 40 + 44 + 50}{5} = \frac{198}{5} = 39.6$$

Step 4: Calculate Slope (β_1)

The formula for calculating the slope is:

$$\beta 1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Table 1.2 shows the calculation of each of the components:

House Size (x_i)	Price (y_i)	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
1500	30	-660	-9.6	6336	435600
1700	34	-460	-5.6	2576	211600
2000	40	-160	0.40	-64	25600
2200	44	40	4.4	176	1600
2500	50	340	10.4	3536	115600

Table 1.2: Calculation of components for finding slope

Sum of products:

$$\sum (x_i - \bar{x})(y_i - \bar{y}) = 6336 + 2576 - 64 + 176 + 3536 = 12560$$

Sum of squares:

$$\sum (x_i - \bar{x})^2 = 435600 + 211600 + 25600 + 1600 + 115600 = 790000$$

Now we can calculate the slope:

$$\beta 1 = \frac{12560}{790000} = \frac{1256}{79000} \approx 0.0158$$

Step 5: Calculate Intercept (β_0)

The formula for calculating the intercept is:

$$\beta 0 = \bar{y} - \beta 1 \bar{x}$$

Substituting the values, we obtained in the above equation we get:

$$\beta 0 = 39.6 - (0.0158)(2160)$$

$$\beta 0 \approx 39.6 - 34.34 \approx 5.26$$

Step 6: Final Model

Now we have our regression equation:

$$y \approx 0.0158x + 5.26$$

Step 7: Making Predictions

Using this model, we can predict the price of a house based on its size. For example, if a house has a size of 1800 sq ft, we can substitute into our model:

$$y \approx (0.0158)(1800) + 5.26$$

$$y \approx 28.44 + 5.26 \approx 33.7$$

Thus, we predict that a house with 1800 sq ft would be priced at approximately Rs. 33.7 lakhs. Figure 1.4 shows the regression line generated in this example.

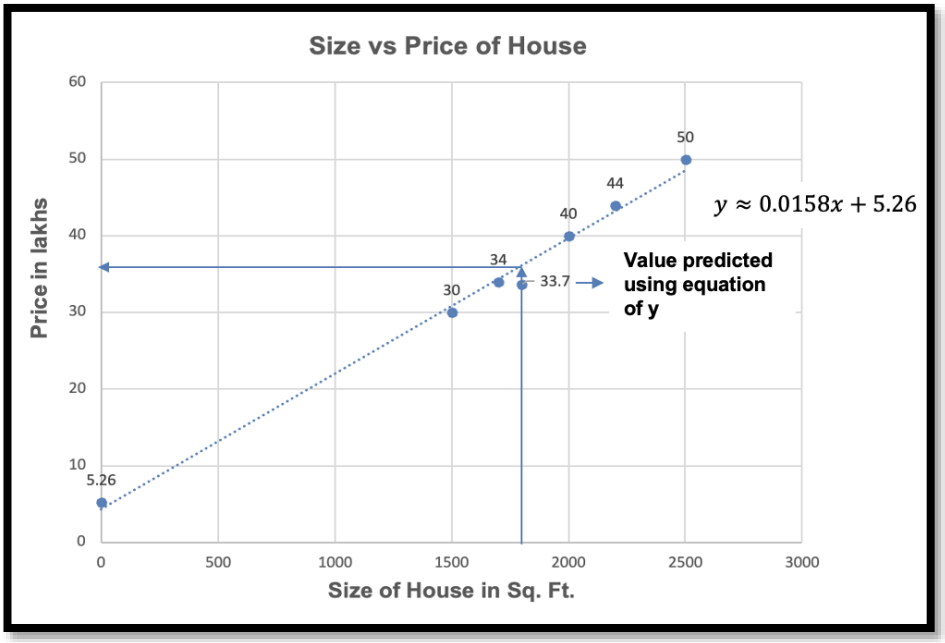


Figure 1.4: Linear Regression Line generated in example

This example illustrates how to perform simple linear regression using a small dataset to establish a relationship between house size and price, allowing us to make predictions based on that relationship.

Assumptions of Linear Regression:

In order to interpret the results of the regression analysis meaningfully, the following conditions must be met.

- **Linearity:** There must be a linear relationship between the dependent and independent variables.
- **Homoscedasticity:** In practice the regression model never exactly predicts the value of dependent variable, there is always an error. This error must have a constant variance over the predicted range.
- **Normality:** The error epsilon must be normally distributed.
- **No multicollinearity:** No high correlation between the independent variables.
- **No auto-correlation:** The error component should have no auto correlation.

Advantages of Linear Regression:

- It is simple to implement and interpret, hence most suitable for beginners in machine learning.
- It is computationally inexpensive and works right for small to medium-sized datasets.
- The nature of the relationship between the variables is straightforward and is easy to be understood.

Limitations of Linear Regression:

- Linear Regression works on the primary assumption of linearity which is never true in the real world.
- Outliers can make the predictions substantially skewed.
- When independent variables are highly correlated, the coefficients of the model may be unstable.

1.4 POLYNOMIAL REGRESSION

Polynomial Regression is the extension of linear regression, the result of which is fitting an object into the polynomial function in order to model the relationship between the independent variable(s) and dependent variable (response). Hence, polynomial regression is much more useful in cases where it would be impossible to draw a straight line due to non-linear relationships among the input and output variables.

General Equation of Polynomial Regression is: Polynomial regression is a form of regression analysis in which the relationship between the independent variable (x) and the dependent variable (y) is modeled as a polynomial of degree n. The generalized equation of polynomial regression can be defined as:

$$\mathbf{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots\ldots + \beta_n x^n + \epsilon$$

Here,

y: Dependent variable (output)

x: Independent variable (input or feature)

$\beta_0, \beta_1, \ldots\ldots, \beta_n$: Coefficients of the polynomial terms

ϵ: Error term

The general polynomial regression model is represented as shown in figure 1.5.

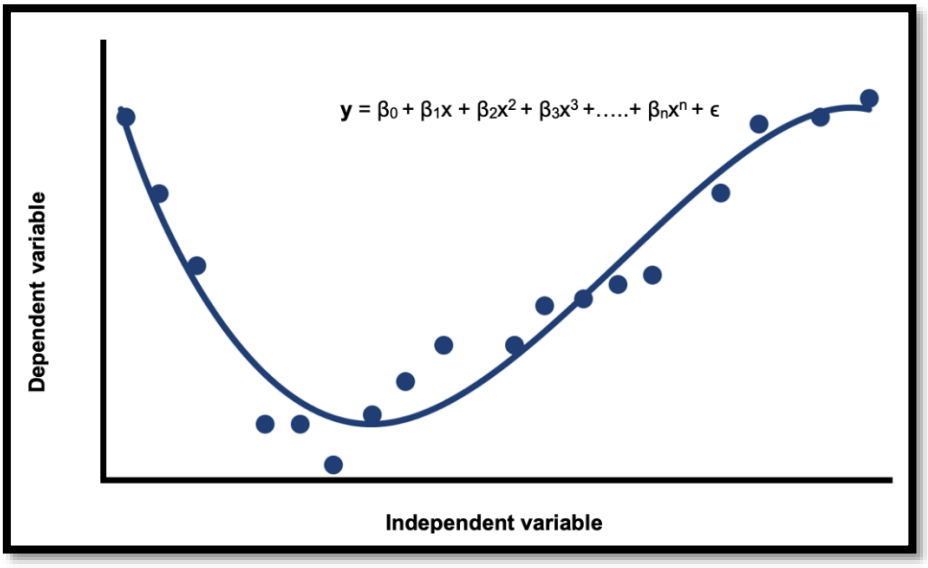


Figure 1.5: Polynomial regression line

Steps to be followed for performing polynomial regression are:

1. **Data Collection:** Prepare a dataset consisting of inputs (independent variables) and outputs (dependent variable), indicating at least one non-linear relationship.
2. **Data Cleaning and Preparation:** Clean the dataset, taking care of missing values and outliers. Normalize or scale the dataset, if needed, so that polynomial terms do not dominate other terms in model fitting.
3. **Feature Engineering:** Convert the independent variable into polynomial features by adding any terms depending on the order (degree) of the polynomial you want to model.
4. **Model Fitting:** Find a line of best fit coefficients for the polynomial equation using the linear regression model and transformed polynomial features.
5. **Prediction:** Feed the dependent variable for new input data to the trained model by substituting the values of these predictors into their respective polynomial equation.
6. **Performance Assessment:** Use evaluation metrics such as Mean-Squared Error (MSE) to check model performance on how accurately it is fit to the provided dataset.

This simple process makes polynomial regression capable of expressing these non-linear trends and providing inferences.

Example of Polynomial Regression:

Assume that we have a dataset pertaining to the age in years and length in millimeters of a fish. We are going to estimate the length of the fish based on the age given by the user.

Input data: Age of fish

Output data: Predicted length of fish

Step 1: Organize the Data

Let's assume we have the data for age and length as shown in table 1.3.

Age (x)	Length (y)
1	67
1	62
2	109
2	83
2	91
3	137
3	122
3	122
2	123
3	122
4	138
4	135
4	146
4	145
4	144

Table 1.3: Dataset for age and length of fish

Let us plot the scatter chart for the dataset given in table 1.3 just to see the way dataset is arranged. The scatter is as shown in figure 1.6.

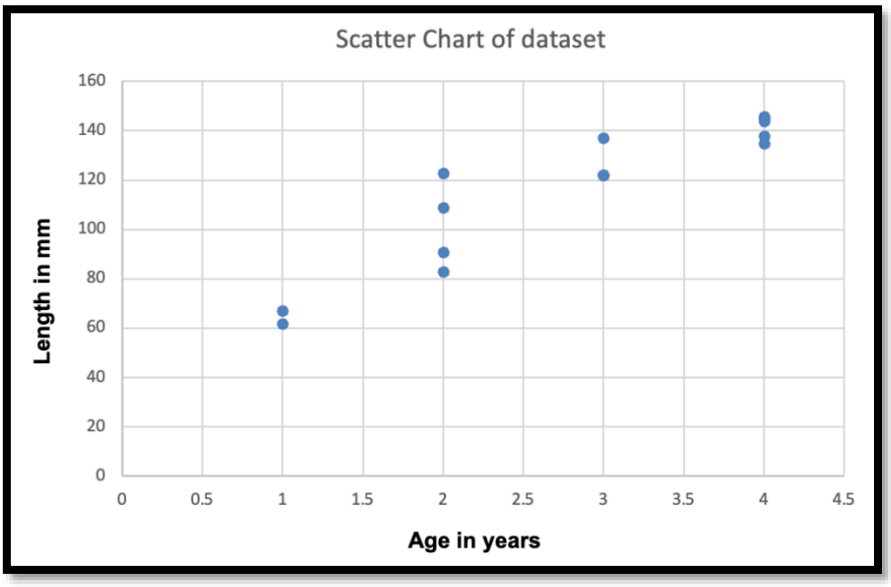


Figure 1.6: Scatter plot of dataset in table 1.3

As can be seen in figure 1.6 the data does not seem to go in straight line, hence we need to fit a polynomial here. Let us try and fit a second-degree polynomial as shown herewith.

$Y = \beta_0 + \beta_1 x + \beta_2 x^{2.}$ Here,

x is the independent variable (age).

y is the dependent variable (length).

β_2, β_1, and β_0 are the coefficients we need to calculate.

Step 2: Compute Summations

To obtain the values of β_0, β_1, and β_2 we need the summations of x, y, x^2, x^3, x^4, xy and x^2y. The calculation is as shown in table 1.4.

Age (x)	**Length (y)**	**x^2**	**x^3**	**x^4**	**xy**	**x^2y**
1	67	1	1	1	67	67
1	62	1	1	1	62	62
2	109	4	8	16	218	436
2	83	4	8	16	166	332
2	91	4	8	16	182	364
3	137	9	27	81	411	1233
3	122	9	27	81	366	1098
3	122	9	27	81	366	1098
2	123	4	8	16	246	492
3	122	9	27	81	366	1098
4	138	16	64	256	552	2208
4	135	16	64	256	540	2160
4	146	16	64	256	584	2336
4	145	16	64	256	580	2320
4	144	16	64	256	576	2304
$\sum x$	$\sum y$	$\sum x^2$	$\sum x^3$	$\sum x^4$	$\sum xy$	$\sum x^2y$
42	**1746**	**134**	**462**	**1670**	**5282**	**17608**

Table 1.4: Calculation of summations

Step 3: Set Up Normal Equations

The coefficients can be calculated by solving the following system of equations:

Equation for Intercept (β_0):

$$N \cdot \boldsymbol{\beta 0} + \boldsymbol{\beta 1} \cdot \sum x + \boldsymbol{\beta 2} \cdot \sum x^2 = \sum y$$

Equation for Linear Coefficient (β_1):

$$\boldsymbol{\beta 0} \cdot \sum x + \boldsymbol{\beta 1} \cdot \sum x^2 + \boldsymbol{\beta 2} \cdot \sum x^3 = \sum xy$$

Equation for Quadratic Coefficient (β_2):

$$\boldsymbol{\beta 0} \cdot \sum x^2 + \boldsymbol{\beta 1} \cdot \sum x^3 + \boldsymbol{\beta 2} \cdot \sum x^4 = \sum x^2 y$$

Where:

N is the number of data points.

Step 4: Solve the System of Equations

Substitute the computed summations into the three normal equations and solve to get values of β_2, β_1, and β_0 using algebraic methods.

$$\begin{bmatrix} N & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix} \begin{bmatrix} \beta 0 \\ \beta 1 \\ \beta 2 \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2 y \end{bmatrix}$$

Thus we need to solve the following

$$\begin{bmatrix} 15 & 42 & 134 \\ 42 & 134 & 462 \\ 134 & 462 & 1670 \end{bmatrix} \begin{bmatrix} \beta 0 \\ \beta 1 \\ \beta 2 \end{bmatrix} = \begin{bmatrix} 1746 \\ 5282 \\ 17608 \end{bmatrix}$$

The matrix equation translates to:

$$15\beta_0 + 42\beta_1 + 134\beta_2 = 1746 \quad ------ \text{ (Eq. 1)}$$
$$42\beta_0 + 134\beta_1 + 462\beta_2 = 5282 \quad ----- \text{ (Eq. 2)}$$
$$134\beta_0 + 462\beta_1 + 1670\beta_2 = 17608 \quad ---- \text{ (Eq. 3)}$$

Eliminate β_0

Subtract 42/15 times Equation 1 from Equation 2:

Multiply Equation 1 by 14/5:

$$42\beta_0 + \frac{588}{5}\beta_1 + \frac{1876}{5}\beta_2 = \frac{24444}{5}$$

Subtract this from Equation 2:

$$\left(134 - \frac{588}{5}\right)\beta_1 + \left(462 - \frac{1876}{5}\right)\beta_2 = 5282 - \frac{24444}{5}$$

Simplify:

$$\frac{82}{5}\beta_1 + \frac{434}{5}\beta_2 = \frac{1966}{5} \implies 41\beta_1 + 217\beta_2 = 983 ---- \text{(Eq. 4)}$$

Eliminate β_0 Again

Subtract 134/15 times Equation 1 from Equation 3

Multiply Equation 1 by 134/15:

$$134\beta_0 + \frac{5628}{15}\beta_1 + \frac{17956}{15}\beta_2 = \frac{233964}{15}$$

Subtract this from Equation 3:

$$\left(462 - \frac{5628}{15}\right)\beta_1 + \left(1670 - \frac{17956}{15}\right)\beta_2 = 17608 - \frac{233964}{15}$$

Simplify:

$$\frac{1302}{15}\beta_1 + \frac{7094}{15}\beta_2 = \frac{30156}{15} \implies 651\beta_1 + 3547\beta_2 = 15078 ---- \text{(Eq. 5)}$$

Solve Equations 4 and 5

Eliminate β_1: Multiply Equation 4 by 651 and Equation 5 by 41:

$$26,691\beta_1 + 141,267\beta_2$$
$$26,691\beta_1 + 145,427\beta_2$$

Subtract the scaled equations:

$$4,160\beta_2 = -21,735 \implies \beta_2 = \frac{-21,735}{4,160} \approx -5.22$$

Solve for β_1:

Substitute $\beta_2 = -5.22$ into Equation 4:

$$41\beta_1 + 217(-5.22) = 983 \implies \beta_1 \approx \frac{983 + 1,133.74}{41} \approx 51.63$$

Solve for β_0

Substitute $\beta_1 \approx 51.63$ and $\beta_2 \approx -5.22$ into Equation 1:

$$15\beta_0 + 42(51.63) + 134(-5.22) = 1,746$$

Simplify:

$$15\beta_0 \approx 277.5 \implies \beta_0 \approx 18.50$$

The final solution thus is

$$\beta_0 \approx 18.50$$
$$\beta_1 \approx 51.63$$
$$\beta_2 \approx -5.22$$

Step 5: Write Final Quadratic Equation

Substitute the values of obtained coefficients into the quadratic equation:

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2.$$

Thus equation that best-fit quadratic curve for our data is

$$Y = 18.50 + 51.63x - 5.22x^2.$$

Step 6: Making Predictions

Using this model, we can predict the length of the fish based on its age. For example, if a fish is 5 years old, we can substitute into our model:

$$y \approx 18.50 + (51.63)(5) - (5.22)(5)^{2.}$$

$$y \approx 18.50 + 258.15 - 130.5$$

$$y \approx 146.15$$

Thus, we predict that fish having 5 years of age would be approximately 146.15 millimetres in length.

Advantages of Polynomial Regression:

- Effectively handles non-linear relationships by fitting curves around the dataset.
- Flexible enough to adapt varying levels of complexity and adjustable by modifying the degree of the polynomial.
- Easy to implement using linear regression methods after feature transformation.

Limitations of Polynomial Regression:

- Very susceptible to overfitting for high-degree polynomials, rendering poor in performance with new, unseen data.
- Being vulnerable to noise and hence results in unstable predictions on new instances.
- Higher computational complexity in processing whenever new polynomial terms are included, may result in slower processing with large datasets.
- Higher-degree equations reduce the interpretability of the relationships making it difficult to understand the relationships between the variables.

1.5 COMPARATIVE STUDY

Having studied both linear and polynomial regression let us have a comparative look at them. Table 1.5 gives the comparison of both the methods.

Aspect	Linear Regression	Polynomial Regression
Relationship Modeled	Assumes a straight-line relationship between variables.	Models non-linear relationships with polynomial terms.
Complexity	Simpler and computationally efficient.	More complex and computationally demanding.
Use Cases	Effective for data with linear trends.	Suitable for data with curved or non-linear trends.
Overfitting Risk	Lower risk of overfitting.	Higher risk, especially with high-degree polynomials.
Interpretability	Easy to interpret and explain.	Harder to interpret due to complex polynomial terms.

Table 1:5: Comparison of regression models

Check Your Progress - 2

a. What is the general equation of a linear regression model?
   a) $y = \beta 0 + \beta 1x^2 + \epsilon$
   b) $y = \beta 0 + \beta 1x + \epsilon$
   c) $y = \beta 0 + \beta 1x^3 + \epsilon$
   d) $y = \beta 0 + \epsilon$

b. What is the primary purpose of the cost function in linear regression?
   a) To increase the number of observations.
   b) To optimize the model by minimizing prediction errors.
   c) To create polynomial features for the model.
   d) To ensure all input variables are normalized.

c. Which of the following is a step in performing linear regression?
   a) Applying clustering techniques.
   b) Transforming features into polynomial terms.
   c) Dividing the dataset into training and testing subsets.
   d) Using cross-entropy loss for optimization.

d. Which of the following is a limitation of linear regression?
   a) It cannot handle small datasets.
   b) It assumes a linear relationship between variables, which may not hold true in the real world.

c) It is computationally expensive for small datasets.

d) It cannot handle missing data.

e. What is the primary purpose of polynomial regression?

a) To model linear relationships between variables.

b) To fit data into a polynomial function and capture non-linear trends.

c) To reduce the computational cost of regression models.

d) To classify data into distinct categories.

f. Which of the following is a limitation of polynomial regression?

a) It cannot handle non-linear relationships.
b) It is difficult to implement using linear regression methods.
c) It is prone to overfitting for high-degree polynomials.
d) It requires no preprocessing of data.

g. What does ϵ represent in the below equation of polynomial regression,
$\mathbf{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots\ldots + \beta_n x^n + \epsilon$?

a) The independent variable.

b) The coefficient of the polynomial term.

c) The error term or residual.

d) The dependent variable.

1.6 LET US SUM UP

This unit primarily focusses on supervised learning techniques and provides an understanding of how relations between variables can be modeled. Regression algorithms are indeed one of the most versatile tools for performing predictions of continuous outcomes, from sales forecasts to predicting outcomes in medicine.

Linear regression is simple, efficient, and widely used; in reality, it is far from sufficient, as the patterns fall outside of a linear relationship. Polynomial regression can easily be said to be a more complex regression that accommodates polynomial terms to accommodate non-linear trends and analyze complex relationships and challenges like overfitting and sensitivity towards outliers.

A thorough understanding of the regression models provides a platform for effective application in predictive analytics and diverse problem-solving domains.

1.7 CHECK YOUR PROGRESS: POSSIBLE SOLUTIONS

1-a Learning from labelled data where inputs and outputs are provided

1-b Predicting continuous outcomes and modeling relationships between values

1-c Forecasting the price of a house based on its size

1-d Its simplicity, stability, and versatility

1-e Diagnosing diseases from X-rays

1-f Estimating delivery times

2-a $y = \beta 0 + \beta 1x + \epsilon$

2-b To optimize the model by minimizing prediction errors.

2-c Dividing the dataset into training and testing subsets.

2-d It assumes a linear relationship between variables, which may not hold true in the real world.

2-e To fit data into a polynomial function and capture non-linear trends.

2-f It is prone to overfitting for high-degree polynomials.

2-g The error term or residual.

1.8 ASSIGNMENTS

- What is the importance of regression algorithms?
- Explain the concept of linear regression.
- Differentiate between linear and polynomial regression.
- List the advantages and limitations of linear regression.
- With an example, explain how polynomial regression is performed.

Unit-2: Introduction to Classification

2

Unit Structure

2.0 Learning Objectives
2.1 Overview of Classification
2.2 Classification Algorithms
2.3 Naive Bayes
2.4 K-Nearest Neighbors (KNN)
2.5 Decision Trees
2.6 Logistic Regression
2.7 Support Vector Machines (SVM)
2.8 Let us sum up
2.9 Check your Progress: Possible Answers
2.10 Assignments

2.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand what is classification, as well as how it forms a part of supervised learning.
- Working mechanisms behind different classification algorithms like Naïve Bayes, KNN, Decision Trees, Logistic Regression, and SVM.
- The student will be able to compare and contrast the strengths, weaknesses, and the appropriate use cases of these classification techniques.
- Apply classification algorithms in solving real-world problems.

2.1 INTRODUCTION TO CLASSIFICATION

Classification is one of the basic machine learning techniques used in supervised learning. It is used to associate the data points with some predefined categories or labels. It implements learning by associating input labels to that with the output labels by recognizing patterns and makes predictions on data not yet seen.

Unlike regression, which focuses on predicting continuous numerical values, classification categorizes data points into distinct, predefined labels or groups. Thus, classification focuses on a discrete set of values by means of attempting to develop a model to conduct actual assignments of new data into previously established categories. For instance, an accurate classification algorithm can identify whether the sent mail is spam or not spam, depending on its content and other features, such as metadata.

Essential characteristics of classification are as follows:

Categorical Predictions: Classification predicts categorical labels, such as “Yes” or “No”, “Spam” or “Not Spam” or many classes, such as “Dog”, “Cat”, or “Bird”.

Defining Decision Boundaries: Classification algorithms create boundaries in the feature space to separate the classes to assure accurate predictions on the unseen data.

Learning from Labeled Data: Models are trained on labeled datasets where each input is associated with a corresponding label.

Steps to be followed for performing classification are:

1. **Data Preparation:** Data is cleaned and measured so as to prepare it for analysis. We might need to handle missing values, normalizing values, and encoding categorical variables.
2. **Model Training:** Using a labeled data set, a classification algorithm is trained to learn the relationship between inputs and the outputs.
3. **Prediction:** The trained model can be used to predict the class label of new data that has never been seen before.
4. **Performance Evaluation:** Evaluate model performance with different metrics to ascertain how accurate, reliable, and efficient the model is.

Classification can have various application areas like:

- **Spam Detection:** Spam is identified via clear division into spam and not spam categories so that users can be spared the unwarranted messages in their inbox.
- **Medical Diagnosis:** Identifying diseases based on the symptoms presented by patients and the results of diagnostic tests.
- **Customer Segmentation:** Groups customers into categories for running particular targeted marketing campaigns.
- **Sentiment Analysis:** It is a process of determining the sentiment (whether it be positive, negative, or neutral) expressed in text data, such as product reviews or social media posts.
- **Image Recognition:** It categorizes images into one of the predefined classes, such as finding objects in a photo.

Classification serves as one of the cornerstones of many decision-making automations, and is remarkably effective in many practical applications since it increases accuracy, thus saving much time. Because it learns from the labeled data, it is one of the most widely used techniques for solving difficult categorization problems.

2.2 CLASSIFICATION ALGORITHMS

Classification algorithms form a major part of supervised learning, effectively applied by the machine to learn from labelled data to perform a forecast on unseen cases or instances. The algorithms are quick to trace out patterns and relations between features and labels so that existing categories are assigned in a reasonable and fast way to new data points. The choice of which algorithm to apply depends on the nature, size, and complexity of the data set. Some of the common classification algorithms are as mentioned:

1. **Naive Bayes:** A probabilistic algorithm based on Bayes' Theorem. Assumes all features are independent. Efficient in cases like text classification and spam detection.
2. **K-Nearest Neighbours (KNN):** An easy, instance-based algorithm that classifies data points according to the majority vote of the nearest neighbours . It works best with small datasets.
3. **Decision trees:** A tree-like structure to divide data based on some feature values using metrics like Gini Index or Information Gain. Are applicable in both binary and multi-class classification instances.
4. **Logistic regression:** A statistical model that can classify binary and multi-class probability using a sigmoid function. It is used mainly for linearly separable data.
5. **Support vector machines:** Identify an optimal hyperplane to separate classes, working through linear and non-linear with the help of kernels. They usually work well with high-dimensional datasets.
6. **Random forest:** An ensemble method comprising several decision trees to avoid overfitting and bolster the accuracy. Best suited for larger datasets.
7. **Neural networks:** Mimicking the processes of human brains via networks of interconnected layers primarily acting to be able to solve highly sophisticated non-linear problems. Requires a substantial amount of data and a high computing power.

Some of the factors that play a crucial role in selection of an algorithm are:

- **Suitability for Dataset Size:** While basic techniques such as Naive Bayes and Logistic Regression suffice for small datasets while larger datasets could be more effectively tackled by Random forest and Neural Networks.
- **Handling Data Complexity:** Logistic Regression and SVMs with linear kernels show empirically that they can successfully separate straightforward datasets, whereas Decision Trees and Neural Networks seem more suitable for non-linear patterns.
- **Achieving High Accuracy:** Ensemble methods such as Random Forest and Gradient Boosting are consistently associated with improved accuracy rates.
- **Need for Speed:** Since algorithms such as Naive Bayes and Logistic regression tend to be computationally efficient, they are naturally practical in applications in which real-time predictions are required.

Check Your Progress - 1

a. What does the term "decision boundaries" refer to in classification?
   a) The limit of a model's computational capacity
   b) The distance between training and testing data
   c) The boundaries that separate classes in the feature space
   d) The number of classes a model can predict

b. What is a common step in the classification process?
   a) Encoding categorical variables in the dataset
   b) Clustering data into groups
   c) Using regression models for predictions
   d) Ignoring missing values in the dataset

c. Which of the following is an application of classification?
   a) Predicting house prices based on square footage
   b) Diagnosing diseases based on patient symptoms
   c) Calculating the average rainfall in a region
   d) Estimating the duration of a video

d. What is the main principle behind K-Nearest Neighbours (KNN)?
   a) Finding the best-fit line between two variables

b) Classifying data based on the majority vote of nearest neighbors

c) Using probability to predict outcomes

d) Building a tree-like structure to split data

e. Which algorithm is best suited for linearly separable datasets?

a) Logistic Regression

b) Random Forest

c) Neural Networks

d) Decision Trees

f. What is the key feature of Support Vector Machines (SVM)?

a) Combines multiple decision trees for classification

b) Finds the optimal hyperplane separating classes

c) Assumes feature independence for predictions

d) Uses majority voting of neighbors for predictions

2.3 NAIVE BAYES

Naive Bayes is a supervised classification algorithm, built from Bayes' theorem and specifically designed for classification problems. It is very effective with other high-dimensional data text categorization problems. As simple and very efficient classifier algorithm, Naive Bayes builds fast models for machine learning that are able to make real-time predictions. The probabilistic classifier Naive Bayes therefore assigns a class label to an object based on the probability that the object belongs to that particular class.

Applications of the Naive Bayes algorithm include spam filtering, sentiment classification, real time predictions and text classification.

The terms "Naive" and "Bayes" in the Naive Bayes algorithm play an important role. "Naive" is named so because the algorithm considers all the features when predicting one variable to be independent of each other. It is basically assuming that the contribution of each feature towards the identification is done separately without consideration of the influence that other features may have. "Bayes" is used because the algorithm is based on Bayes' theorem.

Bayes' theorem, which is also called **Bayes' Rule** or **Bayes' Law**, allows you to update a probability about some hypothesis in light of prior information. It heavily relies on the idea of conditional probability.

The general formula for Bayes' theorem is:

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

Where:

P(A|B) (Posterior Probability): The probability of hypothesis A given the observed event B.

P(B|A) (Likelihood Probability): The probability of observing evidence B given that hypothesis A is true.

P(A) (Prior Probability): The initial probability of hypothesis A before any consideration of the evidence.

P(B) (Marginal Probability): The probability of observed evidence B.

Working of Naive Bayes Algorithm:
The functioning of the Naive Bayes classifier can be illustrated with the following example:

Given a dataset of weather conditions considered in conjunction with a target variable known as "play", it is required to find whether a player will play on a day based on the weather conditions.

We could follow the steps given below to resolve the issue:
1. Create frequency tables from the dataset.
2. Compute the likelihood table through the probabilities of each feature.
3. Apply Bayes' theorem to calculate the posterior probability.

Problem: If it is sunny, will the player play or will be refrained?
Solution: For analysis, we consider the dataset given in table 2.1:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Table 2.1: Dataset of Weather Conditions

The weather conditions frequency would then be as shown in table 2.2:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	**10**	**4**

Table 2.2: Weather Conditions Frequency Table

The weather conditions likelihood would be as shown in table 2.3:

Weather	No	Yes	
Overcast	0	5	5/14 = 0.35
Rainy	2	2	4/14 = 0.29
Sunny	2	3	5/14 = 0.35
All	4/14 = 0.29	10/14 = 0.71	

Table 2.3: Weather Conditions Likelihood Table

Applying Bayes' theorem:

P(Yes|Sunny) = [P(Sunny|Yes)*P(Yes)] / P(Sunny)

P(Sunny|Yes) = 3/10 = 0.3

P(Sunny) = 0.35

P(Yes) = 0.71

So **P(Yes|Sunny)** = [0.3 * 0.71] / 0.35 = **0.60**

P(No|Sunny) = [P(Sunny|No) * P(No)] / P(Sunny)

P(Sunny|NO) = 2/4 = 0.5

P(No) = 0.29

P(Sunny) = 0.35

So **P(No|Sunny)**= [0.5 * 0.29] / 0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny) > P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of the Naive Bayes Classifier:

- Naive Bayes is a quick-and-easy learning-based algorithm for predicting classes to be assigned to the input data sets.
- Naive Bayes can resolve binary as well as multiclass classification problems.
- It is relatively efficient during difficulties of multi-class classification better than other algorithms.
- Naive Bayes is very popular for routinely solving text classification problems.

Disadvantages of the Naive Bayes Classifier:

- Naive Bayes can rely on claiming that each feature is independent, thus meaning that any relation between these features is not measurable in Naive Bayes.

2.4 K-NEAREST NEIGBOURS (KNN)

The K-Nearest Neighbours (KNN) algorithm is a simple and powerful classification method based on the principle of similarity. KNN considers similarity between the new sample data and available cases and assigns it to the most appropriate category based on this similarity. The KNN algorithm keeps track of all available data and can easily classify some new data points using this similarity criterion at the time the new data appear. This means that new data can be classified easily into a well-suited category by using the KNN algorithm.

K-nearest neighbour is sometimes referred to as a lazy learner due to the fact that it does not learn at once from the training set but rather retains the entire dataset and deals with it when asked for classification.

KNN algorithm, during the training phase, only memorizes the dataset, and when it is supplied with new data, it classifies the data based on the categories that resemble that data closely. KNN provides a straightforward procedure for recognizing the category/class of a dataset.

For example, we have an image of an animal that looks like a cat and a dog but, we want to tell, if this animal is a cat or a dog. To carry out this identification, one can surely use the KNN algorithm, since the latter operates on a similarity measure. The KNN model will look for the most similar features of the new image with that of the cat and dog images and will classify it into either a cat or dog category depending on which features were most similar to by this algorithm. Figure 2.1 gives an example of the classification.

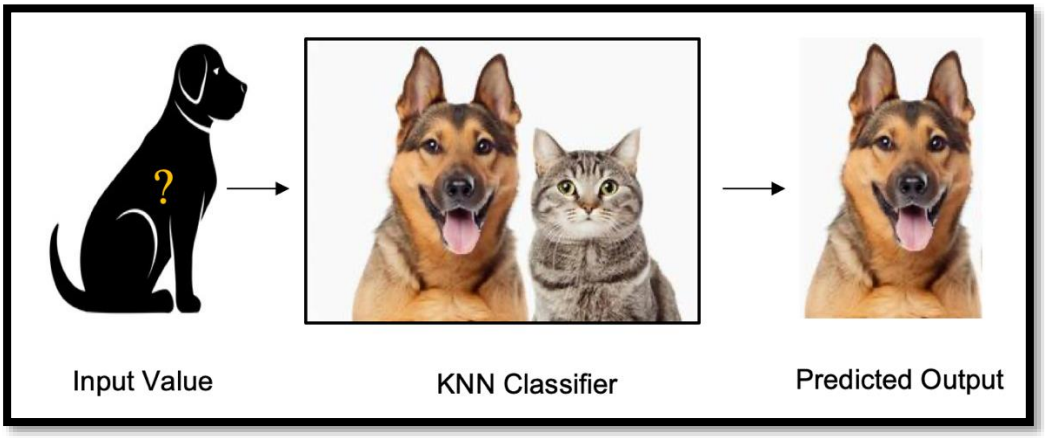


Figure 2.1: Classification Example

Working of KNN Algorithm:

KNN expects new data entry to get compared to the shared feature values with different classes/categories among the data set. Based on its closeness or similarities in a given range (K) of neighbours, the algorithm assigns the new data to the class or category in the data set (training data). The entire process can be broken down in the following way:

Step-1: Decide on K number of neighbouring samples.

Step-2: Calculate their Euclidean distances for K number of neighbouring items.

Step-3: Out of these K neighbouring items, select those that are closest according to the calculated Euclidean distances.

Step-4: Count the number of data points in each category among the K neighbouring contents.

Step-5: Assign the category of the new data point where the number of neighbours is the maximum.

Step-6: The model is ready.

Consider the dataset given in figure 2.2 that consists of two categories: blue and green. Now we have a new data point (orange) for which we need to identify the appropriate category.

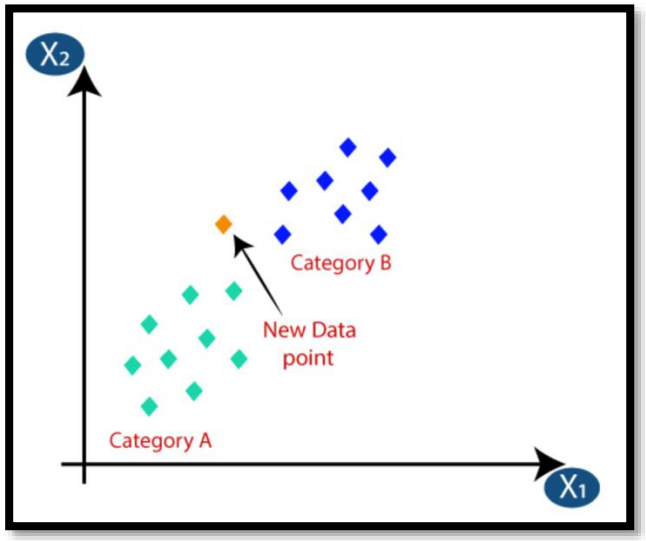


Figure 2.2: Data points corresponding categories - Blue and Green

Here we will assign some value k to represent the number of neighbours that should be taken into account before classifying the new entry. Let K = 3 for this purpose.

In this case, the algorithm will consider only the 3 nearest neighbours of the new point (new entry), represented in the figure 2.2.

Let us now see the working of kNN algorithm. Assume that we have data for two attributes namely brightness and saturation and classes green and blue as shown in table 2.4.

Brightness	Saturation	Class
40	20	Green
50	50	Blue
60	90	Blue
10	25	Green
70	70	Blue
60	10	Green
25	80	Green

Table 2.4: Dataset based on Attributes: Green and Blue

The new data sample has the values for the brightness and saturation attributes as shown in table 2.5.

Brightness	Saturation	Class
20	35	?

Table 2.5: New entry to the existing dataset

Let's identify the class to which it belongs using KNN. To decide its class, we need to find the distance from the new entry to other entries in the data set using some distance formula. The most common distance formula used is the Euclidean distance.

Euclidean Distance = $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

Where:

$x2$ = The brightness of the new entry (20).

$x1$ = The brightness of an existing entry.

$y2$ = The saturation of the new entry (35).

$y1$ = The saturation of an existing entry.

Table 2.6 shows the data of the first row from the dataset used in table 2.4.

Brightness	Saturation	Class
40	20	Green

Table 2.6: First row from the dataset used in table 2.4

Let us now calculate distance 'd1' between the new point (20, 35) and the data in table 2.6 (40, 20).

d1 = $\sqrt{(20 - 40)^2 + (35 - 20)^2}$

= $\sqrt{(-20)^2 + (15)^2}$

= $\sqrt{400 + 225}$

= $\sqrt{625}$

= 25

Now, we have computed the distance from the new data entry to the first entry in the table. Let's update the contents of table 4.2 and add one more field called Distance. The new data will be as shown in table 2.7.

Brightness	Saturation	Class	Distance
40	20	Green	25
50	50	Blue	?
60	90	Blue	?
10	25	Green	?
70	70	Blue	?
60	10	Green	?
25	80	Green	?

Table 2.7: Adding a new column named distance to the dataset

Calculate the distance of all the data points given in table 2.7 as shown in table 2.8.

Point	Calculation
(50, 50)	d2 = $\sqrt{(\mathbf{20} - \mathbf{50})^2 + (\mathbf{35} - \mathbf{50})^2}$ = $\sqrt{(-\mathbf{30})^2 + (-\mathbf{15})^2}$ = $\sqrt{\mathbf{900} + \mathbf{225}}$ = $\sqrt{\mathbf{1125}}$ = 33.54
(60, 90)	d3 = $\sqrt{(\mathbf{20} - \mathbf{60})^2 + (\mathbf{35} - \mathbf{90})^2}$ = $\sqrt{(-\mathbf{40})^2 + (-\mathbf{55})^2}$ = $\sqrt{\mathbf{1600} + \mathbf{3025}}$ = $\sqrt{\mathbf{4625}}$ = 68.01
(10, 25)	d4 = $\sqrt{(\mathbf{20} - \mathbf{10})^2 + (\mathbf{35} - \mathbf{25})^2}$ = $\sqrt{(\mathbf{10})^2 + (\mathbf{10})^2}$ = $\sqrt{\mathbf{100} + \mathbf{100}}$ = $\sqrt{\mathbf{200}}$ = 14.14
(70, 70)	d5 = $\sqrt{(\mathbf{20} - \mathbf{70})^2 + (\mathbf{35} - \mathbf{70})^2}$ = $\sqrt{(-\mathbf{50})^2 + (-\mathbf{35})^2}$ = $\sqrt{\mathbf{2500} + \mathbf{1225}}$ = $\sqrt{\mathbf{3725}}$ = 61.03
(60, 10)	d6 = $\sqrt{(\mathbf{20} - \mathbf{60})^2 + (\mathbf{35} - \mathbf{10})^2}$ = $\sqrt{(-\mathbf{40})^2 + (\mathbf{25})^2}$ = $\sqrt{\mathbf{1600} + \mathbf{625}}$ = $\sqrt{\mathbf{2225}}$ = 47.16
(25, 80)	d7 = $\sqrt{(\mathbf{20} - \mathbf{25})^2 + (\mathbf{35} - \mathbf{80})^2}$ = $\sqrt{(-\mathbf{5})^2 + (-\mathbf{45})^2}$ = $\sqrt{\mathbf{25} + \mathbf{2025}}$ = $\sqrt{\mathbf{2050}}$ = 45.27

Table 2.8: Computing the distance for all the records

The final distance table will be as shown in table 2.9.

Brightness	Saturation	Class	Distance
40	20	Green	25
50	50	Blue	33.54
60	90	Blue	68.01
10	25	Green	14.14
70	70	Blue	61.03
60	10	Green	47.16
25	80	Green	45.27

Table 2.9: Final distance table for all the records

Let us arrange the distances in ascending order as shown in table 2.10.

Brightness	Saturation	Class	Distance
10	25	Green	14.14
40	20	Green	25
50	50	Blue	33.54
25	80	Green	45.27
60	10	Green	47.16
70	70	Blue	61.03
60	90	Blue	68.01

Table 2.10: Rearranging the records in ascending order of distance

Since we have chosen the value of K=3, we will be considering only the first three rows from the table 2.10: Table 2.11 shows the subset.

Brightness	Saturation	Class	Distance
10	25	Green	10
40	20	Green	25
50	50	Blue	33.54

Table 2.11: Considering first three records from the dataset since K=3

From table 2.11 it can be seen that the majority class of 3 nearest neighbours of the new entry is Green. So we classify the new entry as green. So the updated dataset is as shown in table 2.12.

Brightness	Saturation	Class
40	20	Green
50	50	Blue
60	90	Blue
10	25	Green
70	70	Blue
60	10	Green
25	80	Green
20	35	Green

Table 2.12: Newly identified class for the record

There is no specific method to determine the value of K, but here are some general conventions of consideration when we choose it:

- Low values for K will almost always lead to inaccurate predictions.
- Odd values for K should always be maintained.

Advantages of the KNN Algorithm:

- It is easy to implement.
- Classification does not require training beforehand.

Disadvantages of the KNN Algorithm:

- It takes considerable time for a large data set.
- While working with huge data sets, a lot of memory is required.
- Determining the right value of K can be arduous.

2.5 DECISION TREE

A decision tree is a supervised learning algorithm that can be used for both classification and regression problems with a greater preference for classification. It consists of a tree-like structure in which internal nodes represent the features, branches indicate the decision rules, and leaf nodes denote the final outcome.

In a decision tree, there are two kinds of nodes: Decision Nodes and Leaf Nodes. Decision Nodes (i.e. the attributes) are in charge of making decisions and have branches. Meanwhile, Leaf Nodes (i.e. class labels) are the final results of decisions, meaning that they cannot branch anymore. The tree makes decisions or conducts tests based on features of the model.

A decision tree represents all possible solutions to a problem or decision with certain conditions. It is called a tree because it begins from a root node and expands into branches as such forming a shape like a tree. Then it asks a question in which case the tree would split into a smaller subtree.

Example: Predicting whether a person likes computer games?

Suppose you want a prediction on whether or not a person engages in computer games, depending on age and gender. The decision tree goes like this:

Start with the Root Question (Age):

The first question is: “Is the person’s age less than 15?”

- If Yes, go to left.
- If No, proceed to right.

Branch Based on Age:

- If the person is younger than 15, he/she will like them with a prediction score of more than 2 points.
- If the person is 15 or older, ask the next question: Is the person male?

Branch Based on Gender (For Age 15+):

- If the person is a male, he is somewhat likely to enjoy computer games (+0.1 prediction score).
- If the person is a not a male, he is not likely to enjoy computer games (-1 prediction score)

The figure 2.3 illustrates in a broad way the structure of a decision tree for the above problem.

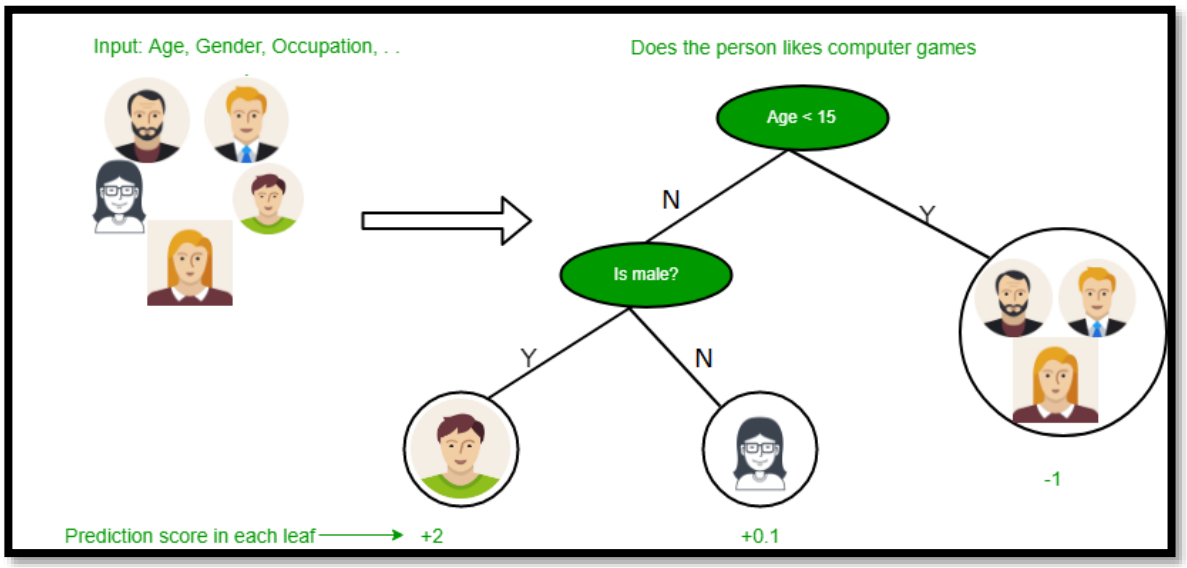


Figure 2.3: Decision Tree

Various categorizations of decision trees along with their distinctive algorithms are as mentioned:

1. Based on the target variable (Classification versus Regression)

This classification pertains to the use of the decision tree, whether for categorical or continuous outputs.

Classification Trees:

These trees are built when the target variable is categorical (for example, Yes/No, Spam/Not Spam). These will predict discrete labels based on input features.

Example: Classifying an email as spam or non-spam.

Splitting Criteria:

- **Gini Index (CART - Classification and Regression Trees) :** It is a measure of impurity or randomness in a dataset. Generally, it calculates probability of misclassifying a randomly chosen element.
- **Entropy and Information Gain (ID3 and C4.5**): It also measures the randomness or uncertainty in a dataset. It is a fundamental concept in information theory.

Regression Trees:

These are the trees are built when the target variable is continuous (for example, predict temperature, predict stock price). This will predict value.

Example: House price prediction based on location and size.

Splitting Criteria:

- **Mean Squared Error (MSE) :** It is a primary splitting criterion for regression trees. It measures average squared difference between predicted and actual values.
- **Mean Absolute Error (MAE):** It is an alternative splitting criterion for regression trees. It measures average absolute difference between predicted and actual values.

2. Based on Specific Algorithms

Most of the algorithms defining ways of building decision trees fall within this category. ID3 (Iterative Dichotomiser 3) works only with categorical data for classification as it uses entropy and information gain to determine the best splits. Nevertheless, it can quickly become an overfitted model. Example: Decision-making based on a set of symptoms as to whether the patient suffers from a certain disease or not.

C4.5 is an improvement over ID3, able to support categorical and numerical data while splitting on subjective entropy and information gain. Additionally, it uses pruning to remove unnecessary branches in order to minimize overfitting. Example: Classifying customers into the different creditworthiness categories.

CART (Classification and Regression Trees) is applicable in invoking both classification and regression, with Gini Index for classification and Mean Squared Error for regression. Example: Defaulting on a loan (Yes/No) as classification, and estimating salary with a regression based on experience.

The Chi-Square Automatic Interaction Detector performs its optimal splits through a Chi-Square test and is extensively used in marketing and survey analysis.

C5.0 is a more advanced version of C4.5, which is more efficient and scalable, capable of producing much smaller and faster decision trees and is thus very well suited for larger business datasets.

3. Ensemble Methods (Combining Multiple Decision Trees for Better Accuracy)

Ensemble techniques combine multiple trees for more accurate predictions and decreased errors, rather than relying on a single decision tree.

One of the very well-known ensemble techniques is Random Forest. In this method, multiple decision trees are made and combined for prediction by averaging over the predicted values in the case of regression or taking a majority vote when it is a classification problem. It reduces the probability of overfitting as against a single decision tree. Example: loan approval decision based on various criteria.

Other ensemble methods are gradient boosting and adaptive boosting.

Some terminologies to be followed while creating a decision tree are:

Root Node: The root node represents the whole population or sample that subsequently gets divided into two or more subsets in a uniform system.

Splitting: It refers to the process of dividing a node into two or more sub-nodes.

Decision Node: A sub-node which further divides into a sub-tree or sub-nodes is designated as a decision node.

Leaf/Terminal Node: The final nodes that do not undergo further branching.

Pruning: It indicates the removal of certain sub-nodes of a tree; an opposite direction to splitting. It can be said that when splitting is done, the cavity made is more of a tree diagram; when pruning is done, that cavity is closed.

Branch/Subtree: A legitimately joined subset of a tree is a branch or a subtree.

Parent and Child Node: The node that takes division into sub-nodes is called the parent of the respective sub-nodes, the latter are termed the child.

Figure 2.4 shows the terminologies used in creating a decision tree.

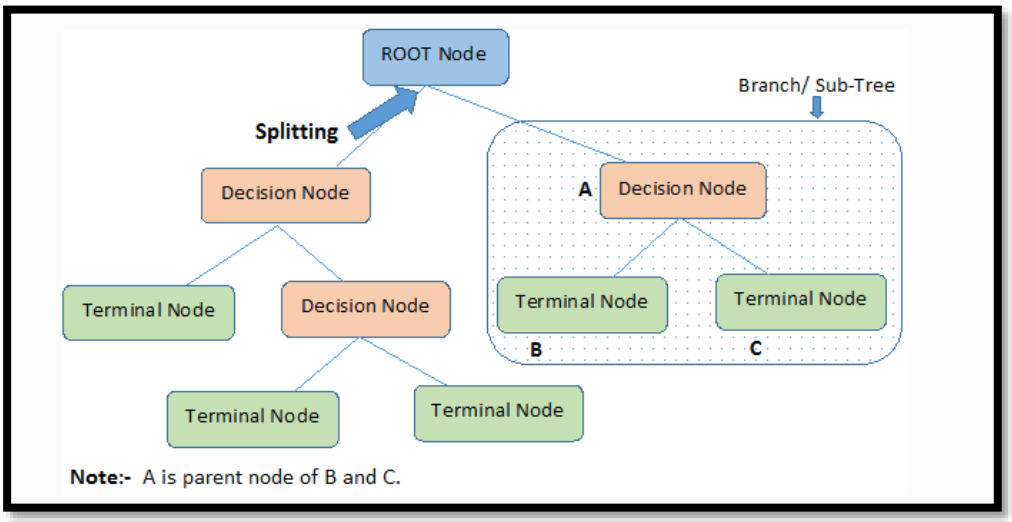


Figure 2.4: Decision Tree Terminology

The decision tree algorithm starts at the root and proceeds in a stepwise fashion to predict the class of a particular dataset like this:

- It takes the attribute value at the root node and compares it with that of the dataset.
- Based on this information, it will follow a particular branch that leads to the next node.

Algorithm for creating Decision Tree:

Step 1. Create a new tree with one node, which is the entire data.

Step 2. Using an Attribute Selection Measure(ASM), select the best attribute.

Step 3. Divide the entire dataset into subsets by using the possible values of the selected attribute.

Step 4. Create a new decision tree node for the best attribute.

Step 5. Grow the child trees, in a recursive manner, of the created nodes until classification is no longer possible.

Step 6. Leaf-nodes are the final nodes in the tree.

One of the main challenges when creating a decision tree is selecting the best attribute for the root node and its subsequent sub-nodes. For such reasons, the concept of attribute selection measure (ASM) is used. ASM can select the appropriate attribute for each node in the tree for efficient decision-making. The two most popular measures applied in attribute selection measure are:

i. Information Gain
ii. Gini Index

Information Gain refers to the effectiveness of a question (or feature) regarding a dataset partitioning. It is essentially a measure of how much uncertainty is reduced after a given split. A good question creates more distinguished groups, and the feature with the highest Information Gain is then used for making the decision.

Example, consider a dataset of people that can be separated into "Young" and "Old" based on their age: in case every young person purchased and no old person

purchased the product in question, the Information Gain will be high, showing that the split separates two groups without any uncertainty left!

If S is a set of instances, A is an attribute, S_v is the subset of S, v represents an individual value that the attribute A can take, and Values (A) is the set of all possible values of A, then

$$\textbf{Gain (S, A) = Entropy (S) } - \sum_{v}^{A} \frac{|Sv|}{|S|} \textbf{ Entropy(Sv)}$$

Here, Entropy is a measurement of uncertainty connected to the variable, which defines the impurity of an arbitrary collection of cases. The greater is the entropy; hence higher is the amount of information contained. Entropy(S) for a set of instances in S can be given by:

$$\textbf{Entropy(S) } = \sum_{v}^{A} - p_v \log_2 p_v$$

Where p represents the proportion of values falling in v.

In simple words, suppose we've got a dataset where there are the same number of "yes" and "no" outcomes-existing, implying 3 people bought a product and 3 did not. Then, in this case, the entropy is really high, as it is uncertain which outcome to predict. If, on the other hand, they are all the same outcome-all "yes" or all "no"-then the entropy equals zero, no uncertainty is left in predicting the outcome.

Example: For the set X = {a, a, b, b, b, b, b}

Total instances: 7

Instances of b: 5

Instances of a: 2

Entropy H(X)= - [$(\frac{2}{7})$ $\log_2(\frac{2}{7})$ + $(\frac{5}{7})$ $\log_2(\frac{5}{7})$]

= - [0.286 x [-1.8074] + 0.714 x [-0.4854]]

= - [-0.863492]

=0.863492

The **Gini index** measures how frequently a randomly selected element would be wrongly classified. It suggests that an attribute with less Gini index value can be preferred.

For example, if in a random selection of people, all bought the product (100% "Yes"), then the Gini Index is 0, which indicates perfect purity; but if an equal proportion of have bought and have not bought the product, the Gini Index is 0.5, indicating that there is a higher impurity or uncertainty.

The formula for the Gini Index is given as follows:

$$\text{Gini} = 1 - \sum_{v}^{A} p_v^{\,2}$$

Advantages of the Decision Tree

- It is easy to interpret since the same process was undertaken by a human when deciding prior to solving a real-life problem.
- Helps in solving decision-related problems.
- It helps to think about all possible outcomes for a problem.
- Less data cleaning is required as compared to other algorithms.

Disadvantages of the Decision Tree

- Decision trees are deep because they have more levels, so they become complex.
- There could be an overfitting issue, which could be tackled with the Random Forest algorithm.
- With an increase in the number of class labels, there is an increase in the computational complexity of the decision tree.

2.6 LOGISTIC REGRESSION

Logistic regression is a well-known and widely used supervised machine learning algorithm mainly employed for solving classification problems. It provides a framework to predict, in terms of probability, whether a particular instance belongs to a certain class. Unlike linear regression, which attempts to predict continuous numerical values,

logistic regression handles problems having a categorical output like Yes/No, True/False, and 0/1.

Logistic Regression hence makes use of the sigmoid function, which transforms the value of the variable into a value between 0 and 1 representing its probability. When this value crosses a particular threshold, usually 0.5, the instance is said to belong to one class; otherwise, it belongs to the other category. Figure2.5 shows the logistic regression S-Curve.

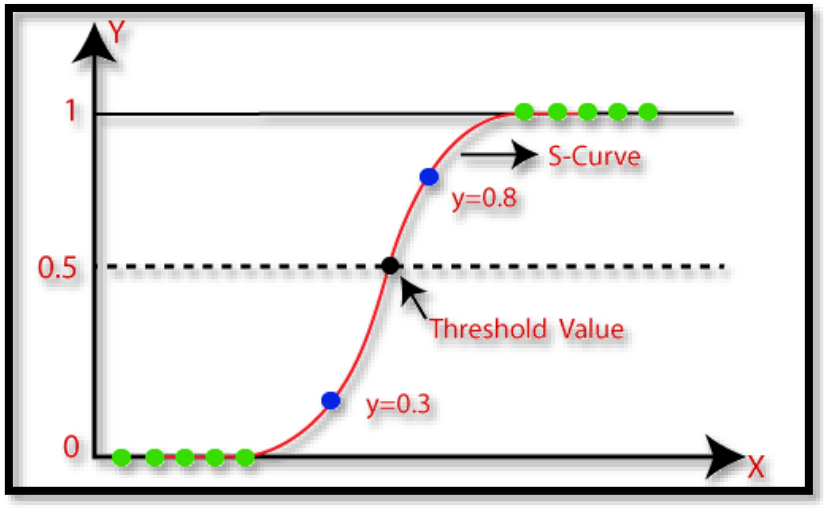


Figure 2.5: Logistic Regression S-Curve

Hence, logistic regression is commonly used for binary classification problems such as spam detection, medical diagnosis, where it has to decide whether a tumour is cancerous or not, and credit risk assessment.

Logistic regression is employed more for classification than for regression tasks. While it is an extension of linear regression, this model applies a logistic transformation to describe the relationship between independent variables and a categorical dependent variable. Thus, it can efficiently deal with both continuous and discrete datasets, allowing for its use in different fields including finance, medicine, and marketing.

Logistic regression is a type of regression analysis used when the dependent variable is categorical. In logistic regression, the dependent variable, Y, is binary (0,1), while X, the independent variables are continuous in nature.

For example, we might want to predict whether a small project will be successful based on the number of years of experience of the project manager who manages that particular project. We believe that the more years of experience the project manager has, the more risk he or she is willing to take in successfully managing projects. Therefore, as X, the number of years of project management experience increases, the probability that Y = 1, where Y equals success on a new project, tends to increase. In work already covered in an example with hypothetical data of 60 previously executed projects, project managers' years of experience were found to lie somewhere between 0 and 20. We described this increasing probability that Y = 1 in graphical form.

To illustrate, it is convenient to segregate years of experience into categories (i.e. 0 - 8, 9 - 16, 17 - 24, 25 - 32, 33 - 40). The mean score on Y (averaging the 0s and 1s) for each category of years of experience is computed as follows:

The data set of table 2.13 shows the years of project management experience (X) and corresponding success ratio (Y).

X	Y
0 - 8	0.27
9 - 16	0.50
17 - 24	0.60
25 - 32	0.66
33 - 40	0.93

Table 2.13: Years of project management experience (X) vs success ratio (Y)

When these X and Y values are plotted on a graph, it will resemble the graph shown in figure 2.6. As can be observed if X increases, so does the probability that Y = 1. This means that as the project manager experiences more years, an increasingly high percentage of projects succeed. This perfect relationship is instead represented by an S-shaped curve rather than by a straight line.

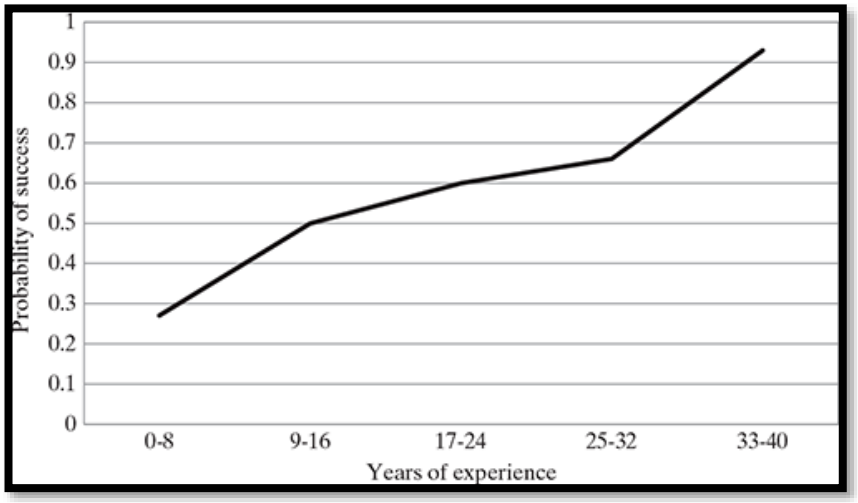


Figure 2.6: Graph representing years of experience vs probability of success

The logistic function can take values ranging from 0 to 1. The formulas of logistic are given in terms of the probability that y=1, called p, and the probability that y=0, given as 1−p. The equation of logistic regression can be obtained from the equation of linear regression. The mathematical steps to get Logistic regression equations:

The equation of the straight line can be written as follows:

$$y = b0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + \ldots\ldots + b_nx_n$$

In logistic regression, the value y can vary between 0 and 1. Hence, we divide this whole equation from (1-y):

$\frac{y}{1-y}$; 0 for y=0 and infinity for y=1

But we need range between -∞ and +∞. So we take logarithm of the equation:

$$\log\left[\frac{y}{1-y}\right] = b0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + \ldots\ldots + b_nx_n$$

Thus the above equation is the final equation for logistic regression.

Based on number of categories, Logistic Regression can be classified broadly into three types:

1. **Binomial:** In Binomial Logistic regression, a dependent variable can take only two possible types, e.g., 0 or 1, Pass or Fail, etc.

2. **Multinomial:** In Multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep".
3. **Ordinal:** In Ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Advantages of Logistic Regression

- Simple and interpretable, logistic regression is easy to implement and understand.
- Efficient for binary classification and works well when the dependent variable is categorical.
- Probability estimates make it useful for decision-making or custom applications.
- Performs well with linearly separable data, ensuring effective classification when classes are well separated.
- It is less prone to overfitting for small datasets with noisy features.

Disadvantages of Logistic Regression

- Linear assumptions make it challenging to find a good decision boundary in cases with too much complexity.
- Seldom usable on large feature sets since the more independent variables there are, the poor its performance.
- It is sensitive to outliers and adversely affects its accuracy.
- For binary classification, it sets apart the extension of one-vs-all to hack through multi-class virtually.

2.7 SUPPORT VECTOR MACHINES(SVM)

Support Vector Machine, or SVM, is one of the most popular Supervised Learning algorithms, applied to Classification and Regression problems. However, it primarily finds its use in Classification problems in Machine Learning.

The SVM algorithm aims to create the best line or decision boundary that can separate the n-dimensional space into classes so that, in the future, we can place the new data point in its correct category easily. This best decision boundary is called a hyperplane.

Extension of a hyperplane depends on the number of used features, that to say, if it has two features, then a hyperplane will consist of a straight line, and if it has three features, then a hyperplane will consist of a two-dimensional plane.

A hyperplane is always created that has maximal-margin, i.e. the greatest distance between the data points.

Those data points that lie close to the hyperplane and thus affect the location of the hyperplane are called **Support Vector.** Since these vectors support the hyperplane, they are thus called a Support vector.

Based upon the nature of the decision boundary, Support Vector Machines can be divided into two major classes:

1. Linear SVM: Linear SVM is for linearly separable data, which means if data can be classified into two classes using a single straight line, this data is called linearly separable data, and the classifier used is called a Linear SVM classifier.

The working of the SVM algorithm can be made clear using an example. Suppose we have a dataset that has two classes (green and blue) and two features x1 and x2. We look for a classifier that can classify the pair (x1, x2) belonging to either the green class or the blue class as shown in figure 2.7.

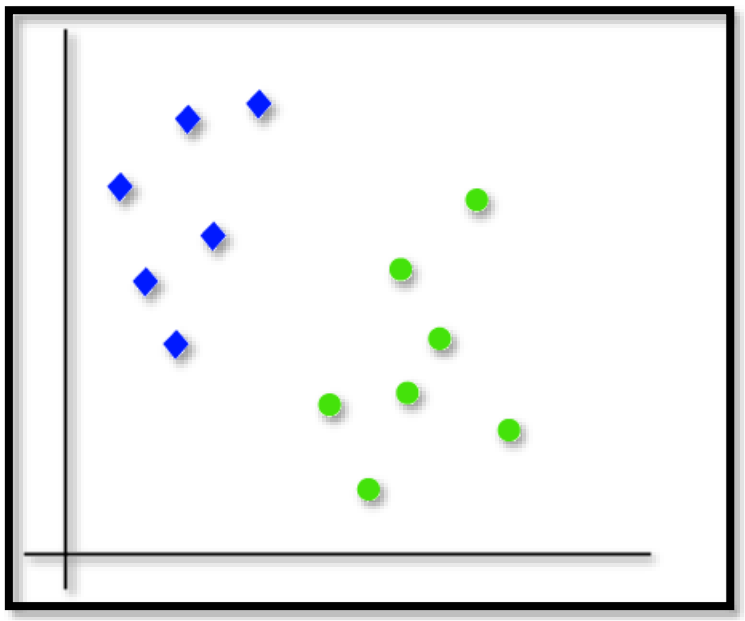


Figure 2.7: Dataset of green and blue class

Since this is a 2-d space, we can easily separate the two classes using a straight line. But there can be many lines that can separate this space as shown in figure 2.8:

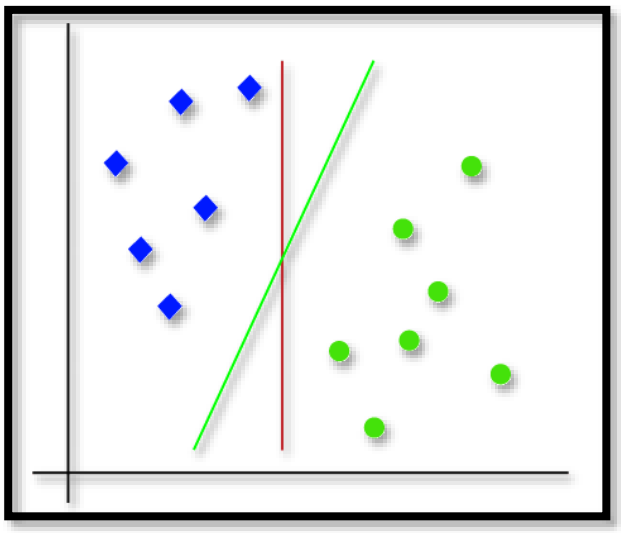


Figure 2.8: Dataset with separated green and blue class

This is where SVM comes in, it finds the line or decision boundary that has the maximum clear path between the two classes; this is called hyperplane. The SVM approach finds the nearest points of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called the margin. And the goal of SVM is to maximize this margin. Hence the hyperplane with maximum margin is called the optimal hyperplane as show in figure 2.9.

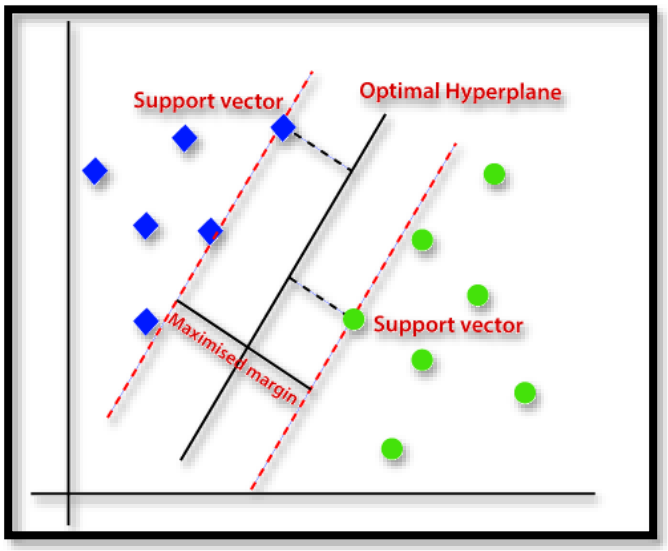


Figure 2.9: Dataset with hyperplane

2. Non-linear SVM: Non-Linear SVM is for non-linearly separated data, which means if data cannot be classified by using a straight line, then such data is called non-linear data, and the classifier used is called a Non-linear SVM classifier.

If the data is arranged linearly, then it can be separated with the help of a straight line, but for nonlinear data it is impossible to draw a unique straight line as can be seen in figure 2.10.

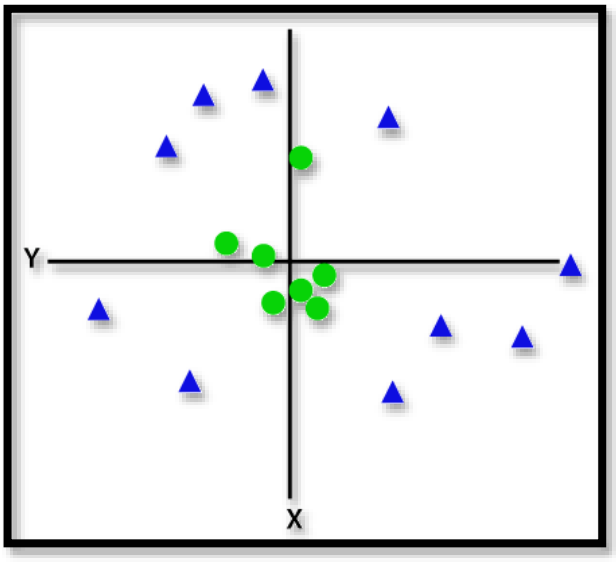


Figure 2.10: Nonlinear Dataset

The data that cannot be linearly separated-SVM employs techniques called kernels to map them to a higher-dimensional space where they become separable. This transformation enables SVM to form a decision boundary even when the data is non-linear.

Kernels are functions through which data points are mapped into some higher-dimensional space without explicitly obtaining the coordinates in that space. Thus, SVM can become increasingly efficient with non-linear data by implicitly performing the mapping.

Consider data points in the above image that are not linearly separable. By applying a kernel function, SVM transforms the data points into a higher-dimensional space where they become linearly separable. To separate these data points, we must add one more dimension. For linear data, we have used two dimensions: x and y, so for

non-linear data, we will add a third dimension, z. With the addition of another dimension, the sample space will now look as shown in figure 2.11.

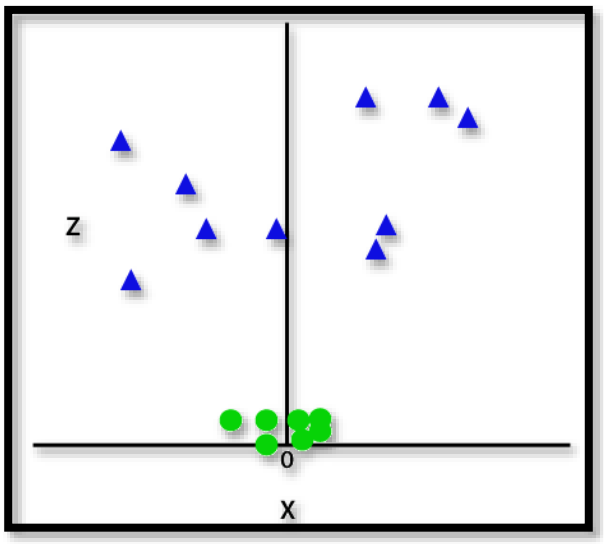


Figure 2.11: Dataset with z dimension

Thus, an SVM separates the datasets into classes in the manner, as illustrated in figure 2.12.

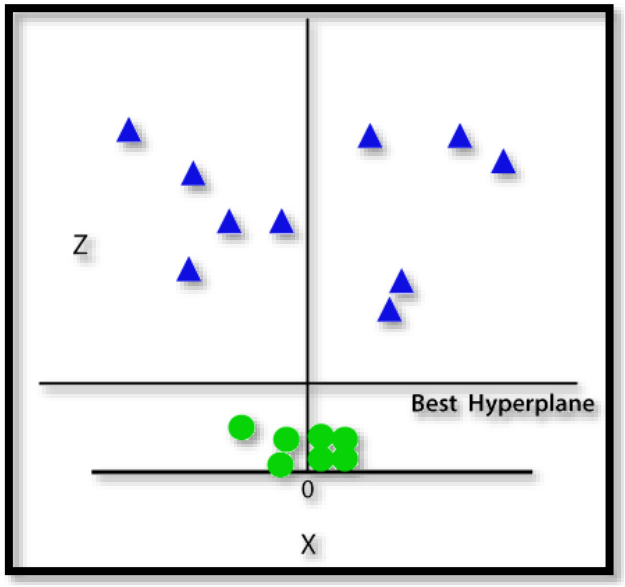


Figure 2.12: Separated Dataset

The hyperplane here appears to be a plane parallel to the x-axis since we are in 3D space. In case we convert this into 2D with z=1, it will take the form as shown in figure 2.13.

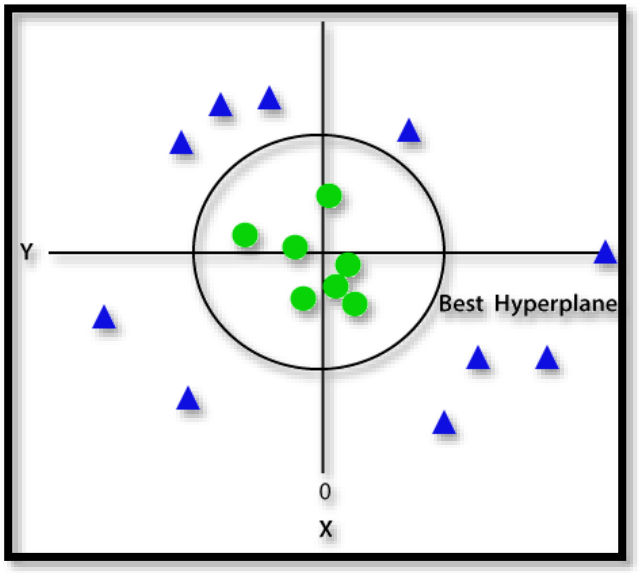


Figure 2.13: Separated Dataset in 2D

Thus we get a circle of a radius of 1 in case of non-linear data.

Advantages of SVM:

- SVM can deal with high-dimensional data effectively, and in addition, performs better where the number of features far exceeds the number of samples.
- Nonlinear data can be effectively handled since the kernel trick can deal with complex decision boundaries.
- Very robust concerning overfitting-for instance, the right regularization in high-dimensional spaces.
- Works well for small datasets; efficient for structured and limited data.

Disadvantages of SVM:

- The computationally expensive training time raises the cost exponentially on the increase of the size of the dataset.
- Careful finetuning is a must since choosing the kernel and the hyperparameters for it is quite a challenge.
- It does not readily admit direct estimates of probability output like logistic regression, hence not very interpretable.
- Memory-consuming particularly on employing the use of nonlinear kernels.

Check Your Progress – 2

a. What assumption does Naïve Bayes make about the features in a dataset?
   a) All features are dependent on each other
   b) All features are independent of each other given the class label
   c) Only numerical features are considered
   d) Features are selected randomly

b. Which of the following is true about KNN?
   a) It is a parametric algorithm
   b) It requires a training phase
   c) It is a lazy learning algorithm
   d) It is not affected by the choice of K

c. What is the primary factor influencing the performance of KNN?
   a) The number of layers in the model
   b) The choice of distance metric and the value of K
   c) The number of neurons in the hidden layer
   d) The activation function used

d. What criterion does a decision tree use to split nodes?
   a) Standard deviation
   b) Entropy or Gini Index
   c) Mean squared error
   d) Root mean square error

e. Which function is used in Logistic Regression to transform linear output into probabilities?
   a) Linear function
   b) ReLU function
   c) Sigmoid function
   d) Softmax function

f. What is the key idea behind the Support Vector Machine (SVM) algorithm?
   a) Maximizing the margin between the decision boundary and the closest data points.
   b) Minimizing the margin between the decision boundary and the closest data points.
   c) Maximizing the number of support vectors.
   d) Minimizing the number of support vectors.

2.8 LET US SUM UP

This unit primarily explored the fundamentals of classification, a key supervised learning technique used to categorize data into predefined labels. We examined various classification algorithms, including Naïve Bayes, K-Nearest Neighbours (KNN), Decision Trees, Logistic Regression, and Support Vector Machines (SVM).

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming feature independence, and is widely used for text classification. KNN is a simple, instance-based algorithm that classifies new data points based on the majority vote of its nearest neighbours. Decision Trees use a hierarchical structure to make decisions based on feature values, making them interpretable but prone to overfitting. Logistic Regression is a statistical model used for binary classification, applying the sigmoid function to predict probability values. SVM finds an optimal hyperplane to separate data points and works well with complex, high-dimensional datasets.

Each of the algorithms discussed has its own advantages and limitations, making them suitable for different types of classification problems. Understanding their working principles, strengths, and challenges enables the selection of the most appropriate model for a given dataset.

2.9 CHECK YOUR PROGRESS: POSSIBLE SOLUTIONS

1-a The boundaries that separate classes in the feature space
1-b Encoding categorical variables in the dataset
1-c Diagnosing diseases based on patient symptoms
1-d Classifying data based on the majority vote of nearest neighbours
1-e Logistic Regression
1-f Finds the optimal hyperplane separating classes
2-a All features are independent of each other given the class label
2-b It is a lazy learning algorithm
2-c The choice of distance metric and the value of K
2-d Entropy or Gini Index
2-e Sigmoid function
2-f Maximizing the margin between the decision boundary and the closest data points.

2.10 ASSIGNMENTS

- What is classification in machine learning, and how does it differ from regression?
- Describe the key assumptions made by the Naïve Bayes classifier and explain how it handles classification tasks.

- How does the K-Nearest Neighbors (KNN) algorithm classify data points, and what are the challenges associated with its use on large datasets?
- In Decision Trees, how does the algorithm decide which attribute to split the data on at each node?
- What is the primary difference between Logistic Regression and Linear Regression, and why is logistic regression used for classification?
- Explain the concept of the hyperplane in Support Vector Machines (SVM) and how it helps in classifying data.
- Compare the strengths and weaknesses of Naive Bayes and Decision Trees for classification problems.

Unit-3: Ensemble Methods 3

Unit Structure

3.0 Learning Objectives

3.1 Overview of Ensemble Learning

3.2 Ensemble Learning Technique

3.3 Simple Ensemble Learning Technique

3.4 Advanced Ensemble Learning Technique

3.5 Difference between Bagging and Boosting

3.6 Let us sum up

3.7 Check your Progress: Possible Answers

3.8 Assignments

3.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the basic ideas of ensemble learning in machine learning.
- Understand several ensemble learning techniques and their utilization.
- Distinguish between bagging boosting techniques in ensemble methods.
- Acquire knowledge regarding essential ensemble learning methods, including Random Forests and Gradient Boosting Machines
- Determine the advantages, disadvantages, and applications of different ensemble approaches through comparing and contrasting them.

3.1 INTRODUCTION

In the previous chapter, we studied classification as a supervised learning technique that classifies data into groups. Our detailed analytically meaningful categorization algorithms were Naïve Bayes, which uses probabilistic approach to make predictions; K-Nearest Neighbour (K-NN), which relies on proximity to the nearest data points; Decision Trees are optimized to repeatedly partition data into branch structures for optimal decisions; Logistic Regression predicted probabilities of binary outcomes effectively; Support Vector Machines (SVMs) reliably determine the correct boundary to accurately distinguish the classes.

This chapter shall cover the basic idea of ensemble learning, a general process that raises the predictiveness of decisions taken from multiple models. It discusses such things as bagging and boosting and proceeds to elaborate a whole range of advanced algorithms such as Random Forests and Gradient Boosting, stating each one's specific advantages and disadvantages in dealing with the complicated nature of real-world machine learning problems.

3.2 ENSEMBLE LEARNING TECHNIQUE

Ensemble learning is a paradigm of integrating and training various base models, usually called “weak learners”, to solve a specific problem. This technique hinges upon the idea that a weak learner is ineffective by itself, yet, when combined with other weak learners, becomes a very strong learner-the ensemble model-which yields much more accurate predictions, the flow of Ensemble learning is shown in figure 3.1.

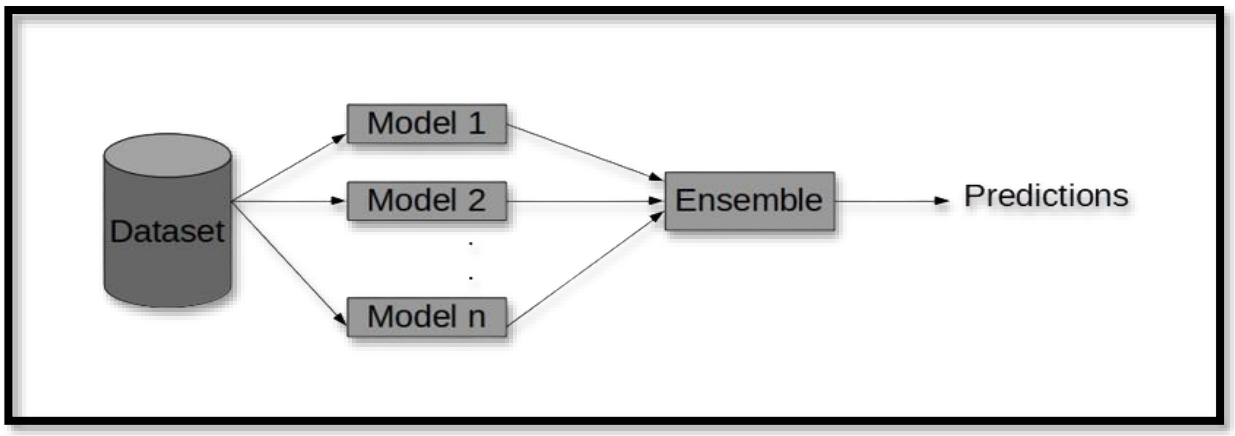


Figure 3.1: Ensemble Learning

Imagine that you want to settle on a movie for the weekend. Instead of just asking one friend, now you're asking for recommendations from five friends. If three out of the five suggest watching the same movie, you go for that particular movie. This is the whole idea behind ensemble learning, voting, or basically integrating multiple individuals or models to come to a better decision.

3.3 SIMPLE ENSEMBLE LEARNING TECHNIQUE

Voting and averaging methods belonging to the ensemble learning class are the simplest and least obscure forms of developing models. For classification problems, one typically uses voting, while averaging is used for regression-type issues.

1. Averaging

This strategy consists of taking the average of the predictions of all models.

Example: Assume that we want to predict the price of the house.

The three base models predict the prices as Rs. 4,52,000/-, Rs. 5,01,000/-, and Rs. 5,00,000/- respectively.

Then the final predicted price of the house would be found by taking the average of the initial three predictions:

Final predicted price = (4,52,000 + 5,01,000 + 5,00,000) / 3
= 1,453,000 / 3
= 4,84,334

2. Max Voting Classifier

Max voting works practically on the same principle as average voting, but it has particular applications in categorization problems. This technique combines predictions from different models, typically termed votes, and the prediction with the greatest number of votes is given as a final outcome.

Example: Assume that we want to predict the price of the house.

If the house price predictions from a set of final prices obtained from multiple models are Rs. 5,00,001/-, Rs. 4,50,600/-, Rs. 6,00,001/-, Rs. 4,50,600/-, Rs. 6,50,000/-, Rs. 4,50,600/-, and Rs. 6,00,000/- respectively.

Then, by applying the maximum voting classifier the prediction of the house price, in this case would be Rs. 4,50,600. (As 3 out of 7 models predicted the price of the house as Rs. 4,50,600)

3. Weighted Averaging

The weighted average is basically a refinement on the averaging process. In other words, if averaging gives equal preference to all base models, this technique gives different weights to all the base models, where the best performing one would have a higher weight relative to others.

Weights for the models will obviously take values between 0 and 1 and are such that the value of the total weights is equal to one.

Example: Assume that we want to predict the price of the house.

The three base models predict the prices as Rs. 4,50,000/-, Rs. 6,00,000/-, and Rs. 6,50,000/- respectively. If the weights for these each of the models are given as 20, 50, and 25 respectively.

Then the final predicted price of the house would be as mentioned:
= 0.20 × 450,000 + 0.50 × 600,000 + 0.25 × 650,000
= Rs. 5,52,500/-

Check Your Progress-1

a) Ensemble learning combines a large number of weak learners to form a strong model that generates better predictions. (True/False)
b) In ensemble learning, a weak learner is competent by itself and does not need combinations with other models. (True/False)
c) Voting and averaging provide the basis for ensemble learning methods. (True/False)
d) Averaging is an operation where the final forecast is computed as the mean of the predictions from all the foundational models. (True/False)
e) In max voting, the final result is determined by the forecast with the least votes. (True/False)

3.4 ADVANCED ENSEMBLE LEARNING TECHNIQUE

Advanced Ensemble learning technique uses two types of methods namely; bagging and boosting. Let us have a look at each of these methods.

1. Bagging (Parallel)

Bagging, an acronym for Bootstrap Aggregating, is a collective learning method which increases the accuracy and stability of any machine learning algorithm. The steps used in bagging are as mentioned:

- **Subset Selection:** Bagging begins with the selection of a random sample from the complete dataset.
- **Bootstrap:** Means for sampling with replacement; given this, a model will invariably learn from the same training examples of that original dataset.
- **Bootstrapped_ aggregation:** Combining models using bootstrapped samples, which might have identical training instances.
- **Independent training:** Each model will be trained independently, using its respective bootstrapped set. This generates an output for each model.
- **Majority Voting:** This is the method of acquiring final output by taking the majority vote. In other words, we identify the most commonly predicted outcome for each model.
- **Aggregation:** In this step, we combine the results obtained from all the models and declare final output through majority voting, known as aggregation.

Figure 3.2 shows the process of bagging.

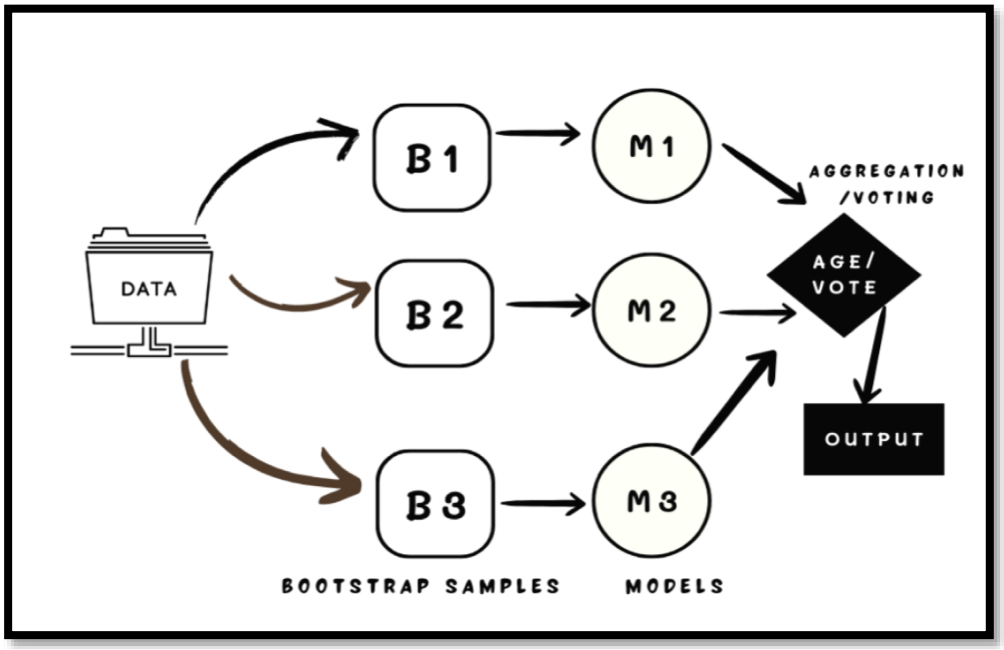


Figure 3.2: Bagging Ensemble Method

Main Benefits:

- **Reducing the Variance:** Bagging reduces the variance for model predictions and reduces the risk of overfitting by averaging the predictions from several predictors.
- **Improves Accuracy:** Usually, an ensemble of models has much improved performance than a single model.

Random Forest Algorithm

The Random Forest Algorithm is the most common application of bagging. In Random Forest, several decision trees are trained on different bootstrapped samples of the dataset, and the predictions of these trees are combined. The method is vibrant because it gives the benefits of decision trees fused together with the stability and robustness of bagging

The steps to implement the random forest algorithm are as mentioned:

Step 1: Randomly select n records and m features from a dataset containing k records to construct each decision tree.

Step 2: Create different decision trees for different samples.

Step 3: Any new test record will be classified based on the majority output among present decision trees in terms of votes.

Step 4: Evaluate the final output using Majority Voting for classification and Averaging for regression, respectively.

Example:

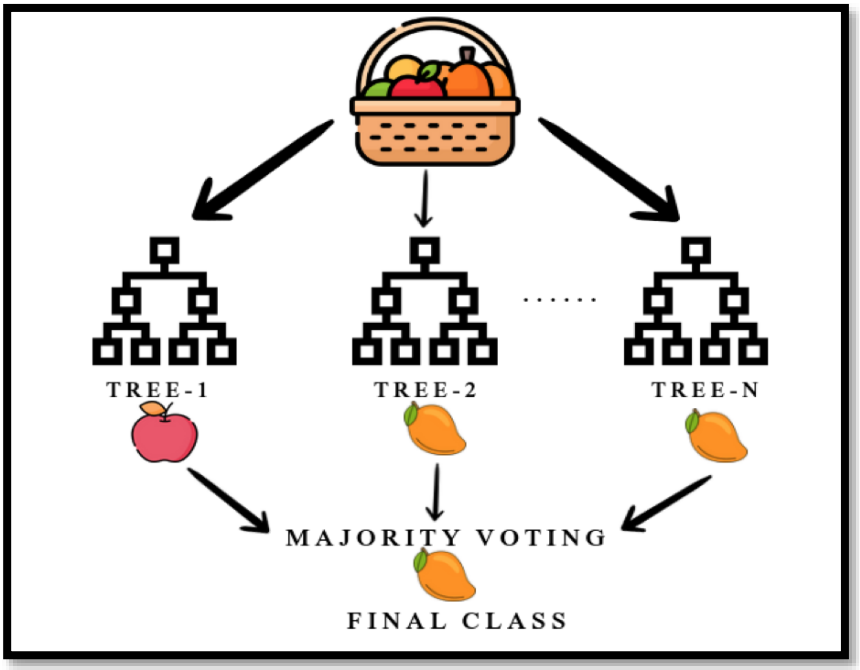


Figure 3.3: Working of random forest algorithm

The fruit basket shown in figure 3.3 is equivalent to data. It specifies how many samples will be extracted from the fruit basket to create a different decision tree for each sample.

Every decision tree produces an output, as seen in the figure 3.3. Finally, the ultimate result is then determined based on majority voting.

In this case, the graphic indicates that the decision tree produces the output of a mango instead of an apple, and therefore it is then designated a mango.

Check Your Progress-2

a) Bootstrap sampling denotes the selection of data samples without replacement. (True/False)
b) In Bagging, each model is trained autonomously utilizing its corresponding bootstrapped dataset. (True/False)
c) Random Forest exhibits less robustness and stability compared to individual decision trees. (True/False)
d) The primary benefit of Bagging is the enhancement of model accuracy by the aggregation of many predictions. (True/False)
e) Majority vote in Bagging is employed to ascertain the ultimate output by choosing the least often expected result. (True/False)

2. Boosting (Sequential)

Boosting is an ensemble method that attempts to put together a strong model with a combination of diverse weak models. The steps of used in boosting algorithm are as mentioned:

- **Sequential Training:** This involves training the models one after each other, with each model trying to solve the mistakes made by the former.
- **Weight Adjustment:** Each example in the training set is assigned a weight. At the beginning, all instances have the same weight. After one model has been trained, the weight of incorrectly classified examples is increased so that the next model will give priority to these difficult examples.
- **Model Combination:** The predictions of all the different models are merged into a single output through either weighted voting or weighted average.

Figure 3.4 shows the process of boosting.

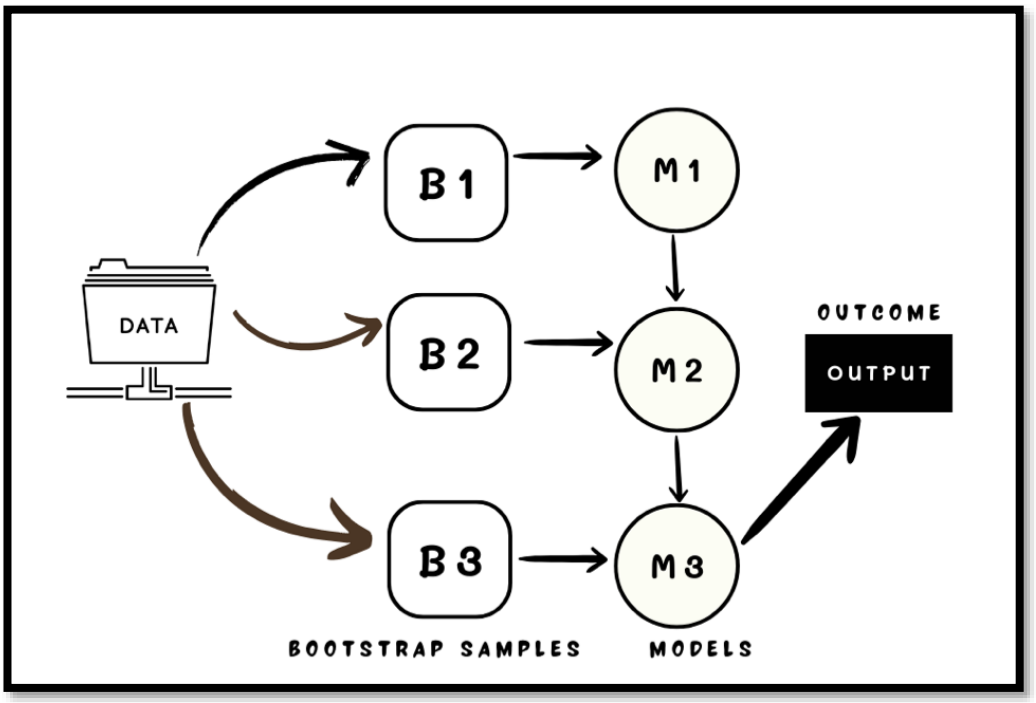


Figure 3.4: The Process of Boosting

Main Benefits

Reduces Bias: The focus on difficult-to-classify instances causes boosting to remove bias from the model, thereby increasing accuracy.

Strong Predictors: The combined effect of weak learners leads to a much more powerful predictive model.

Gradient Boosting Algorithm

Gradient Boosting Machine is a widely utilized forward learning ensemble technique in machine learning. Despite the number of algorithms employed in machine learning, boosting techniques gained a reputation in the global community in machine learning.

The main idea of this boosting is that it follows the ensemble learning principles by combining several weak learners or base estimators to generate the final output. Gradient Boosting Machine is one of the several ensemble techniques employed in machine learning, and it acts by transforming weak learners into strong learners.

Gradient Boosting Machines (GBM) facilitate the development of a predictive model using an ensemble of weak predictive models, such as decision trees. When a decision tree functions as a weak learner, the resultant algorithm is termed gradient-boosted trees.

This aggregation allows us to combine predictions from multiple learner models to develop a final predictive model having accurate predictions. Figure 3.5 shows the process of gradient boosting.

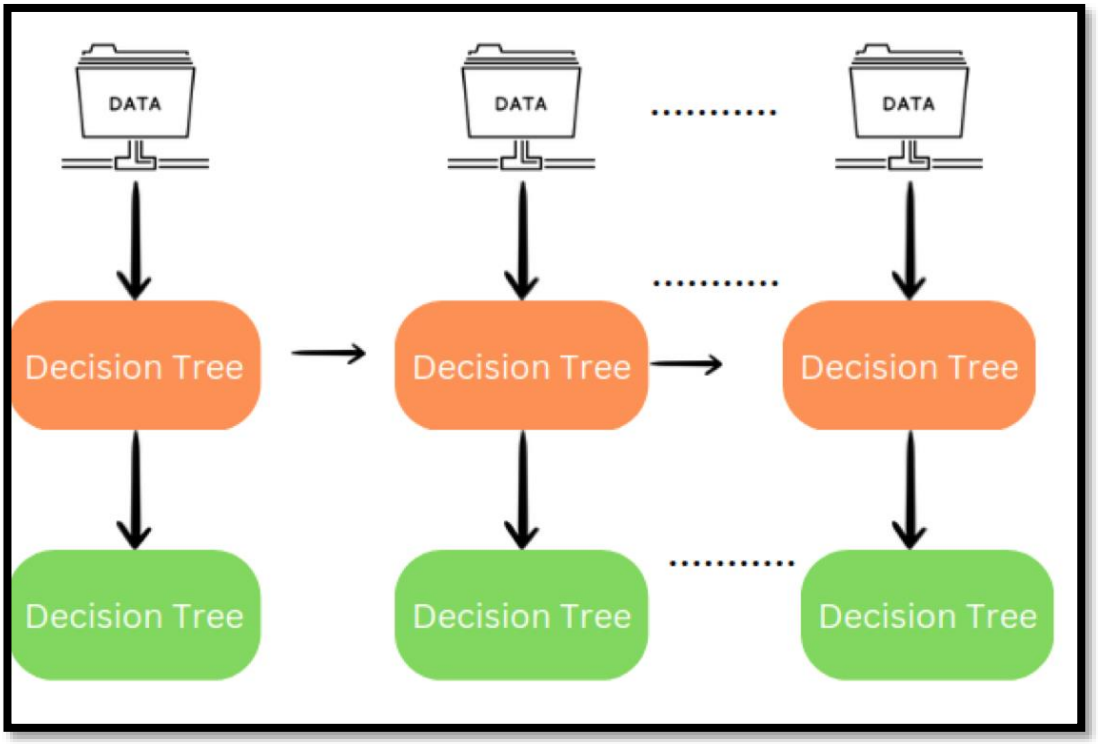


Figure 3.5: The Process of Gradient Boosting

The nodes of the decision tree use different subsets of features to determine their best split. Each tree behaves differently, catching different signals from the entire data.

Gradient Boosting machine consist of 3 elements as follows:

1. Loss Function
2. Weak Learners
3. Additive model

Check Your Progress-3

a) In Boosting, all training occurrences retain uniform weight during the training process. (True/False)

b) Weighted voting or weighted averaging is employed to amalgamate predictions in Boosting. (True/False)

c) A Gradient Boosting Machine (GBM) employs a singular decision tree to produce final predictions. (True/False)

d) Gradient Boosting Machines (GBM) function by converting weak learners into robust learners. (True/False)

e) The three fundamental components of a Gradient Boosting Machine (GBM) are the Loss Function, Weak Learners, and the Additive Model. (True/False)

3.5 DIFFERENCE BETWEEN BAGGING & BOOSTING

The ensemble techniques that are highly adopted are Bagging and Boosting. Figure 3.6 shows both of them, while they both amplify model performance in different ways, Bagging tries to reduce variance through the training of many independent models, whereas Boosting will reduce bias by sequentially training models in such a way that each model learns from the errors of its predecessors.

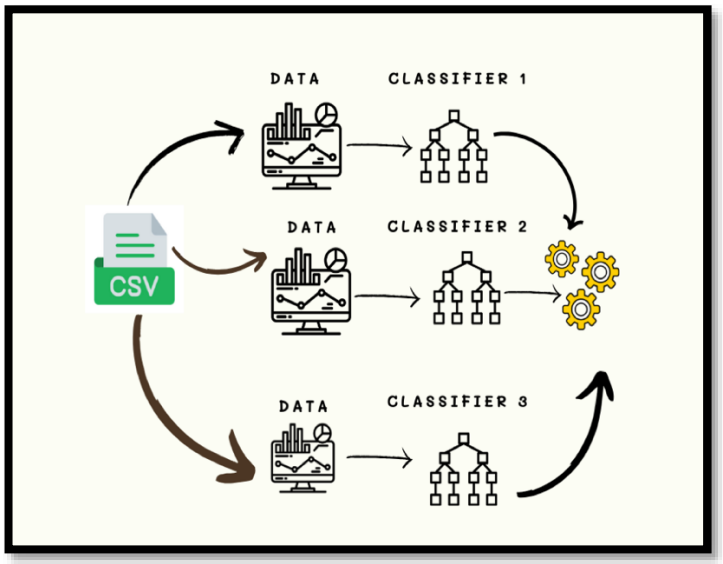


Figure 3.6: Bagging (Parallel) and Boosting(Sequential)

Table 3.1 gives the comparative study of bagging and boosting method.

Aspects	Bagging	Boosting
Objective	Minimize the variance by averaging forecasts from various models.	Trying to reduce bias through learning iteratively from the asymmetric predictions of a model.
Model Development	Models are trained autonomously on distinct subsets of the data .	Models are being built by starting from a single, which learns from the previous model's biases.
Data sampling	Using bootstrapped datasets only.	Using the complete dataset, focusing on instances that were misclassified.
Model Integration	Voting for classification; averaging for regression.	Weighting the models in accordance with their performance against one another.
Risk of Overfitting	Reduced likelihood of overfitting through the building up of the independent models.	More likely to overfit, particularly if the model attains excessive complexity.
Diversity of Models	Several models that are trained simultaneously undergo diversity generally because of bootstrapping.	Models are trained in sequence and rely on the errors of the preceding models.
Fundamental Learner	Generally, employs robust learners such as decision trees.	Generally, use weak learners such as shallow decision trees.

Table 3.1: Comparison of Bagging and Boosting methods

Check Your Progress-4

a) Bagging primarily seeks to reduce bias in models. (True/False)
b) Bagging employs bootstrapped datasets, which entails selecting random samples with replacement. (True/False)
c) Boosting invariably ensures superior performance compared to Bagging in every situation. (True/False)
d) Boosting assigns uniform weights to all training occurrences during the learning process. (True/False)
e) Bagging is generally executed by the Random Forest algorithm. (True/False)

3.6 LET US SUM UP

Ensemble learning is an effective methodology in machine learning that integrates different models to enhance predictive accuracy. The fundamental concept is that combining the outputs of several weak learners yields a more robust and dependable model. Basic ensemble techniques encompass averaging and maximum voting, which improve forecasts by amalgamating results from various models. Ensemble learning is strengthened by advanced methods like bagging and boosting by lessening variation and bias. Whereas bagging, its shortened version which stands for Bootstrap Aggregating, works with models trained independently, boosting is sequential and corrects a preceding model's error.

Bagging techniques, such as Random Forest, train numerous decision trees on various data subsets and consolidate their outputs to formulate a more robust and precise predictive model. Boosting, conversely, emphasizes challenging-to-classify examples by modifying the weights of misclassified samples to improve learning. Gradient Boosting Machines (GBM) are a widely utilized boosting technique that incrementally enhances weak learners into a strong prediction model. Bagging effectively mitigates overfitting and enhances stability, whereas boosting is proficient at decreasing bias but may occasionally result in overfitting if not meticulously calibrated.

A fundamental difference between bagging and boosting is their methodology: bagging mitigates variation through the training of many independent models, whereas boosting lowers bias by successively enhancing weak learners. Both methodologies

have extensive applicability in practical machine learning challenges, ranging from medical diagnosis to financial predictions. Ensemble learning enhances the accuracy, stability, and generalization of machine learning models, becoming essential in contemporary predictive analytics.

3.7 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a True
1-b False
1-c True
1-d True
1-e False
2-a False
2-b True
2-c False
2-d True
2-e False
3-a False
3-b True
3-c False
3-d True
3-e True
4-a False
4-b True
4-c False
4-d False
4-e True

3.8 ASSIGNMENTS

- Explain Ensemble Learning Technique with an example?
- Explain the key differences between Bagging and Boosting?
- Write Main Benefits of Bagging and Boosting?
- Explain Gradient Boosting Algorithm?
- Explain Simple Ensemble learning technique?

Unit-4: Model Evaluation 4

Unit Structure

4.0. Learning Objectives
4.1. Introduction to Model Evaluation
4.2. Evaluation Metrics for Classification and Regression
4.3. Train-Test Split
4.4. Cross-Validation Techniques
4.5. Let us sum up
4.6. Check your Progress: Possible Answers
4.7. Assignments

4.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the purpose and importance of evaluating machine learning models.
- Learn key metrics used to evaluate classification and regression models.
- Understand train-test split and cross-validation techniques for model validation

4.1 INTRODUCTION TO MODEL EVALUATION

Model evaluation is the process of assessing the machine learning model. In order to determine how well, accurately, and efficiently the model can produce predictions, we assess it using specific metrics. Each metric provides unique insights into how well a model is performing and helps in guiding the choice of model and tuning. Evaluation helps us know both the strengths and the weaknesses of the model.

Why is Evaluation important when building a model?

Evaluating a model is not only a formality, it is the key. Without evaluation, we would not know if the model is capable of making reliable predictions on new data in the real-world. A model could be excellent on training data but then perform terribly when encountering unfamiliar data. Evaluation gives us the confidence that the model is precise, and more importantly, generalizable and therefore reliable. Overfitting and underfitting is an example of two issues that could ruin a model's performance.

Overfitting: A model is said to be overfit when the model learns the training data too well, even with the noise and random fluctuations but struggles to learn from new data.

Underfitting: Underfitting is the opposite of overfitting. A model is said to be underfit if it is too basic to understand the underlying patterns in the data. Both on the training and new data, it performs poorly because it lacks the complexity required to accurately depict the relationships that exist.

Right Fit: Occurs when both the training data error and the test data are minimal.

4.2 EVALUATION METRICS

Evaluation metrics are numerical measures used to assess the effectiveness and performance of a statistical model or machine learning model. Evaluation metrics provide not only a basis for comparing different models or algorithms but also insights into model effectiveness. It is crucial to assess the quality, generalizability, and predictability of a machine learning model. Evaluation metrics are selected based on the type of data, the outcome desired, and the specific context.

Evaluation Metrics for Classification:

Classification metrics in machine learning are important for evaluating a model's ability to separate the classes from one another. The most frequent metrics used are accuracy, precision, recall, and the F1-score.

Accuracy indicates the overall proportion of accurate predictions. Precision indicates the percentage of positive predictions that were actually positive. Recall evaluates the percentage of actual positives that were identified correctly. The F1-score indicates a balanced measure of precision and recall.

In addition to these metrics, it is also important to interpret and understand confusion matrices and Area Under the Receiver Operating Characteristic curve (ROC-AUC curves) in order to select the best model for a given classification task and ensure that it will perform properly on new data.

1. Accuracy:

Accuracy is defined as the number of correct predictions to the total number of predictions. This is the most fundamental metric used to evaluate the model. The formula to calculate accuracy is as mentioned:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

$$= \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ \ Negative}$$

Here:

True Positive (TP): Cases where the model correctly predicted the positive class. For example, An OCR predicting that a character is ‘A’ when it actually is character ‘A’.

True Negative (TN): Cases where the model correctly predicted the negative class. For example, An OCR predicting that a character is not ‘A’ when it actually is not character ‘A’.

False Positive (FP): Cases where the model incorrectly predicted the positive class. For example, An OCR predicting that a character is ‘A’ when it actually is character ‘B’. It is also known as a "Type I error" or "false alarm."

False Negative (FN): Cases where the model incorrectly predicted the negative class. For example, An OCR predicting that a character is not ‘A’ when it actually is character ‘A’. It is also known as a "Type II error" or "miss."

Strengths:

- It is easily interpretable.
- Good for balanced classes, meaning when you have roughly the same character of each class (e.g., accuracy is a good overall performance metric when you have at least reasonably balanced class distributions within a dataset).

Limitations:

Sensitive to imbalanced class: In severe scenarios (e.g., fraud, when most of the transactions are not fraud) a model would get high accuracy just predicting the majority class.

2. Precision (Positive Predictive Value):

Precision measures the proportion of predicted positive cases that are truly positive. Precision tells you how reliable the model is when it says something is positive. A high-precision model is less likely to produce false alarms. The formula for calculating precision is as mentioned:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

3. Recall (Sensitivity)

Recall measures the proportion of actual positive cases the model correctly identifies. Recall reflects how good your model is at not missing out on the class you are truly interested in. A high recall model minimizes false negatives. The formula to calculate recall is as mentioned:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

4. F1 score:

F1 score is the harmonic mean of precision and recall. It is seen that during the precision-recall trade-off if we increase the precision, recall decreases and vice versa. Unlike a simple average, the harmonic mean is more sensitive to low values. The formula is given by:

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

5. ROC (Receiver Operating Characteristic) Curve:

ROC is a graph illustrating a classifier's performance at all possible classification thresholds.

Axes:

X-axis: False Positive Rate (FPR) = 1 - Specificity

It measures how often a negative instance is wrongly classified as positive. It can also be calculated using the formula as mentioned:

$$FPR = \frac{False\ Positive}{False\ Positive + True\ Negative}$$

Y-axis: True Positive Rate (TPR) = Recall

It measures how many of the positive cases the model catches. It can also be calculated using the formula as mentioned:

$$TPR = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figure 4.1 shows the graph for Receiver Operating Characteristic) Curve

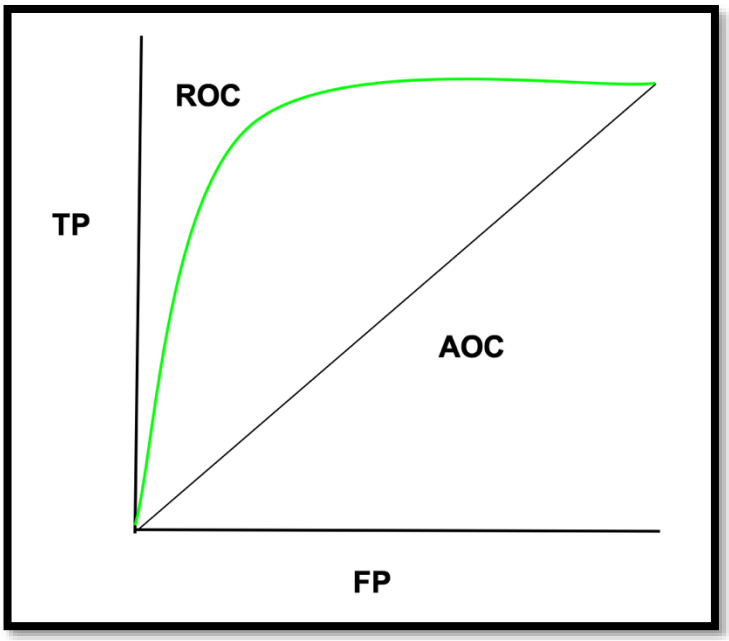


Figure 4.1: Representation of ROC Curve

6. Confusion Matrix:

It is a table that summarizes the performance of a classification model based on the values of true positives, false positives, true negatives and false negatives. It's especially helpful when comparing the actual results with the predictions. Figure 4.2 shows the confusion matrix.

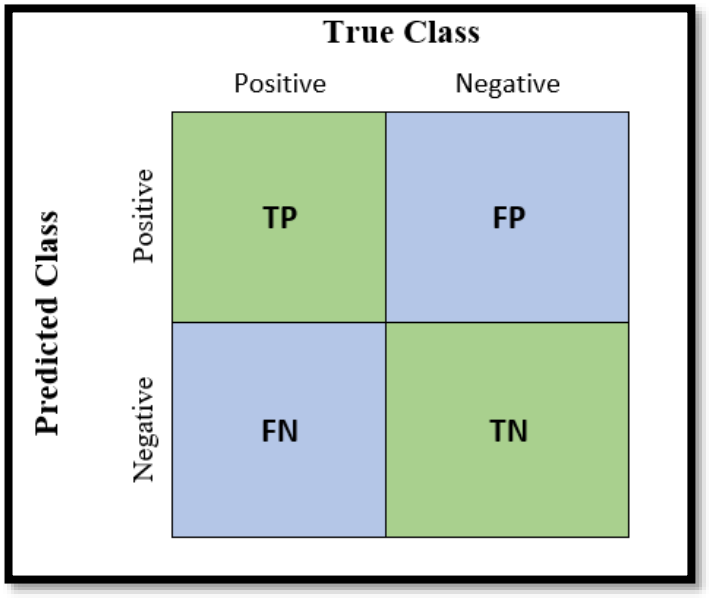


Figure 4.2: Confusion Matrix

Evaluation Metrics for Regression:

Regression identifies continuous values. Regression is primarily used to evaluate the relationship between a dependent and an independent variable. With classifiers, we can summarize the prediction quality with a confusion matrix, accuracy, f1 score, etc. Since regression is predicting a numerical value, it may differ from the actual prediction, therefore, we perform an error calculation, providing a summary measure of the prediction value to the actual value. There are many metrics available for evaluating the regression model.

1. Mean Absolute Error (MAE):

It is the average of the absolute differences between the actual value and the value predicted by model.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \quad | y_i - \hat{y_i} |$$

where,

N = total number of data points

y_i = actual value

$\hat{y}_i$ = predicted value

2. Mean Squared Error (MSE):

It is the average of the squared differences between the actual and the predicted values. Lower the value, the better the regression model.

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

3. Root Mean Squared Error (RMSE)

It is the mean squared difference between the actual value and the predicted value. Then we take the square root of MSE to get the Root Mean Square Error. The lower RMSE is, the better the model is with its predictions. A higher RMSE means that there are larger deviations between the predicted and actual value.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$$

4. R-squared or the coefficient of determination (R^2):

R-squared (R^2) indicates the proportion of variance of the dependent variable that the independent variable explains. R^2 is a common measure of model accuracy which indicates how close data points are to the fitted line created by a regression algorithm. A larger R squared indicates a better fit. This helps us identify the relationship of the independent variable towards the dependent variable.

R^2 score ranges from 0 to 1. The closer to 1 the R^2, the better the regression model is. If R^2 is equal to 0, the model is not performing better than a random model. If R^2 is negative, the regression model is erroneous.

It is the ratio of the sum of squares and the total sum of squares.

$$R^2 = 1 - \frac{SSE}{SST}$$

where

SSE **:** is the sum of the square of the difference between the actual value and the predicted value

$$SSE = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

SST **:** is the total sum of the square of the difference between the actual value and the mean of the actual value.

$$SST = \sum_{i=1}^{m} (y_i - \underline{y_i})^2$$

Check Your Progress - 1

a) Overfitting occurs when a model performs well on both training and test data. (True/False)
b) Mean Squared Error (MSE) is a suitable metric for evaluating classification problems. (True/False)
c) Recall measures the proportion of actual positives correctly identified by the model. (True/False)
d) Mean Absolute Error (MAE) is commonly used to evaluate regression models. (True/False)
e) F1-Score is the average of accuracy and recall. (True/False)
f) ROC is used to evaluate classification models by analyzing the trade-off between sensitivity (recall) and specificity. (True/False)
g) After adding a feature in the feature space, whether that feature is an important or unimportant one, the R-squared always increases. (True/False)

4.3 TRAIN-TEST SPLIT

Train test split is a validation method used for a model that simulates how a model would perform on unseen data through a simple split of a dataset into two groups. The first group, the training set, comprises the data used to train the model. The second group of data, the testing set (which is considered new to the model), is used to test

the model's performance (accuracy). Train test split can also include a validation set, which is data used to adjust hyperparameters and optimize the model during the training process. Figure 4.3 give the general overview of the train test split procedure.

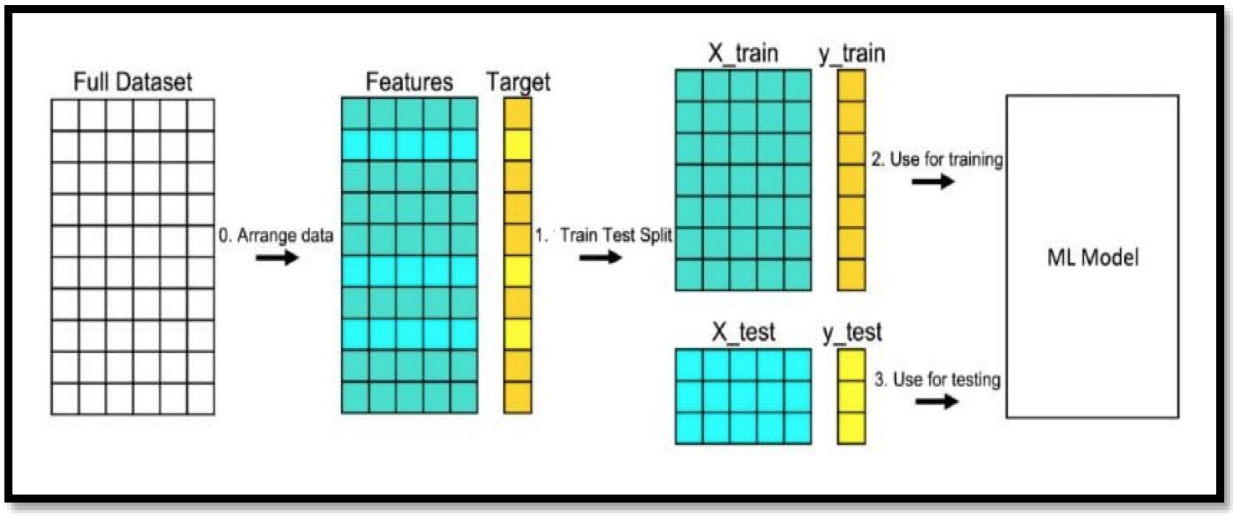


Figure 4.3: Train test split procedure

Training Set:

Training set is a portion of the dataset designated for model fitting. To put it another way, the model is given the training set of data so that it can learn from it and adjust its parameters.

The training set must be

i. sufficiently large to yield decent results (without being so huge that the model overfits)
ii. representative of the entire dataset in order to maximize the model's performance during training.

This will improve the trained model's future predictive power on unobserved data. When a machine learning model becomes so specialized and tailored to the training data that it is unable to generalize or predict accurately with any new data, this is known as overfitting. Thus, an overfit model will overperform with the training set, but underperform when introduced to the validation sets and test sets.

Test Set:

When assessing a trained model's ultimate efficiency, the test dataset is utilized. It offers an objective evaluation of how effectively the finished model generalizes to fresh data and, eventually, to the actual world. An accurate assessment of the model's performance is successfully provided by keeping a distinct test dataset up to date during the model-building process.

Additionally, the test dataset indicates how well the trained model can handle fresh data. An objective indicator of how well the model will probably generalize to the real world is provided by assessing model fit on the test dataset, which consists of new data that the model has never seen before. This assessment enables us to determine if the trained model has successfully learned relevant patterns and can make accurate predictions beyond the training and validation contexts.

Validation Set:

Data used to evaluate and improve a machine learning model during training is known as the validation set. This is done in order to evaluate the model's output and maximize its performance. We can determine how effectively the trained model generalizes to unknown data by evaluating it on the validation set.

This evaluation may highlight issues like overfitting, which could significantly affect the model's practical applicability. Hyperparameters can also be adjusted using the validation set.

The model's performance is controlled by hyperparameters, which include regularization strength and learning rate. We can identify the hyperparameters that get the best results by experimenting with different values during training on the training set and assessing on the validation set.

Methods for Splitting Data in a Train Test Split:

There are various methods of splitting datasets for machine learning models. The selected data split method and data split ratios will differ based on the following factors: (1) the intended use case, (2) the available data amount, (3) the data quality, and (4) the parameters that one might consider. Here’s some common methods of splitting data in a train test split:

1. Random Sampling

Random splitting refers to the practice of shuffling the dataset at random and creating training and testing datasets according to a specific percentage (for example, 80% training and 20% testing). It is one of the most common ways to split data into train test splits because of its simplicity and ease of execution. Random splitting is effective on large datasets with categories that are typically equally represented in the dataset.

2. Stratified Splitting:

Splitting a dataset using a stratified approach is often utilized with datasets that are imbalanced, which means certain classes/categories have considerably less instances to other classes/categories. In this situation, it is important to properly reflect the class distribution of the dataset in the splits for training, validation, and test set, so that the final model will not be biased.

With stratified splitting, we divide the dataset while maintaining the relative sizes of each of the classes in each of the splits. The training, validation, and test set will then have a representative portion of each of the classes, maintaining the same proportions navigated. This way, the model is able to learn to identify patterns and predict across all classes, which will result in a more robust, reliable machine learning algorithm.

Cross-Validation Splitting

Cross-validation sampling refers to the method of partitioning a dataset into training and validation data as part of cross-validation. The cross-validation sampling process involves creating multiple copies of a dataset subset, with each subset serving as training or validation data in the cross-validation process.

This process is very easy to understand and to implement, and it usually has lower estimated bias than available alternatives for tracking the efficiency scores of a machine learning model. There are a lot of different techniques that may be used to cross-validate a model.

4.4 CROSS-VALIDATION TECHNIQUES

To ensure that the model is correctly trained on the data provided without much noise, you need to use cross-validation techniques. These are statistical methods used to estimate the performance of machine learning models. Let's the different types of cross-validation techniques:

1. K-fold cross-validation

In this method, the entire dataset is split into k equal-sized portions. Each portion is referred to as a fold. It is referred to as k-fold because the dataset is partitioned into k distinct portions where k is an integer of 3, 4, 5, etc.

One fold will be used for validating the model, and the other K-1 folds will be used for training the model. This method is carried out k times until each fold is used once as a validation set and the remaining portions are used for training the model. Figure 4.4 shows the general working of K-fold cross-validation.

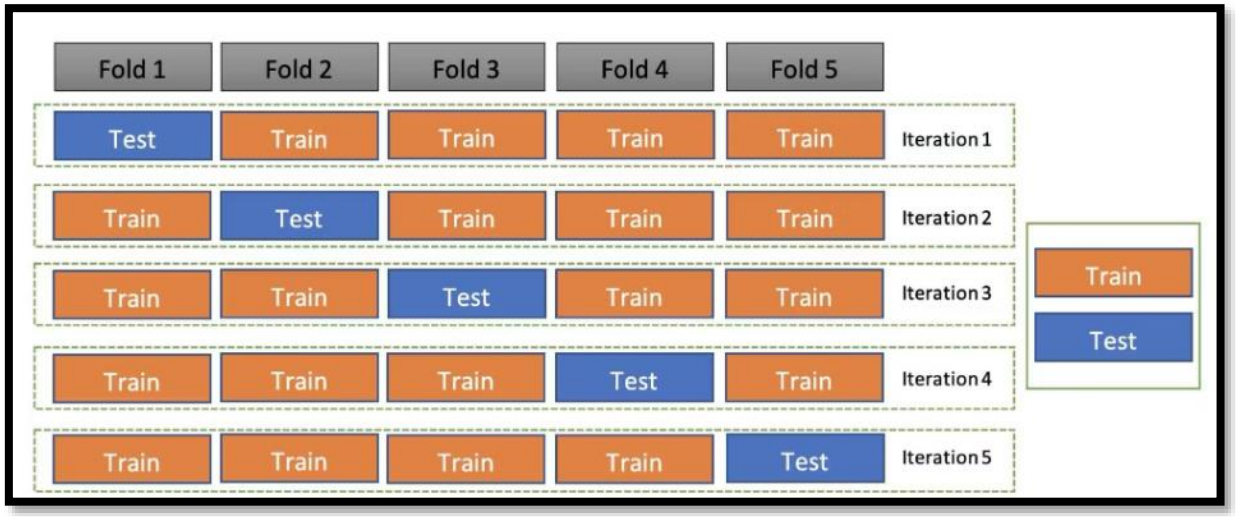


Figure 4.4: K-fold cross-validation

The illustration in figure 4.4 shows 5 folds (k), or 5 iterations. In every instance, one fold is the test set/validation set and the other k-1 sets (4 sets) are the train set. To get the final accuracy, you take the accuracy of the k-models validation data.

Note that this validation method is not suitable for imbalanced datasets because the model will not get trained properly due to not having the correct ratio of each class's data.

2. Hold-out cross-validation

This is the most straightforward evaluation technique and is popular in any Machine Learning projects, where the full dataset (population) is divided into 2 sets - train set and test set. The portion of train and test data could be split in the ratio of 70-30 or 60-40 or 75-25 or 80-20 or even 50-50 depending on use case.

As a rule, however, the proportion of training data has to be greater than test data. The data is split randomly and essentially, we do not know which data ends up in the train or test bucket when we split the data unless you set random_state. The random assignment generates very high variance, and every time we carry out the split, we get a different accuracy score. Figure 5.5 shows the block diagram of hold-out method.

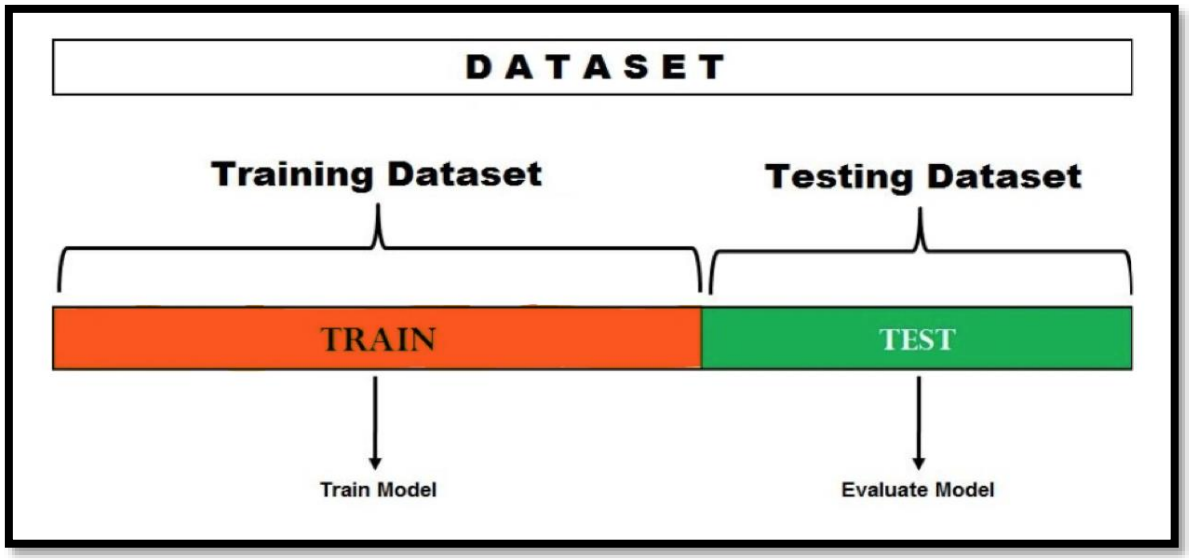


Figure 4.5: Hold-Out method

Drawback of hold-out methods are:

- Test error rates are highly variable (high variance) and entirely dependent on which observations end up in the training set and test set.
- With hold out methods, also only a portion of the dataset is used for training the model (high bias) which is not a good idea when data is not ample, and will lead to overestimation of test error.

The major advantage of this method holds in its computational efficiency with respect to all cross-validation methods.

3. Stratified k-fold cross-validation

As illustrated previously, k-fold validation cannot be performed with imbalanced datasets because the data is split into k-folds with a uniform probability distribution. However, this is not true for stratified k-fold, which is a better version of k-fold cross-validation. It also splits the dataset into k equal parts with each part having the same ratio of instances of target variables that are found in the entire dataset. This allows it to work perfectly for imbalanced datasets, but not work for time-series data. Figure 5.6 shows the process of Stratified k-fold method.

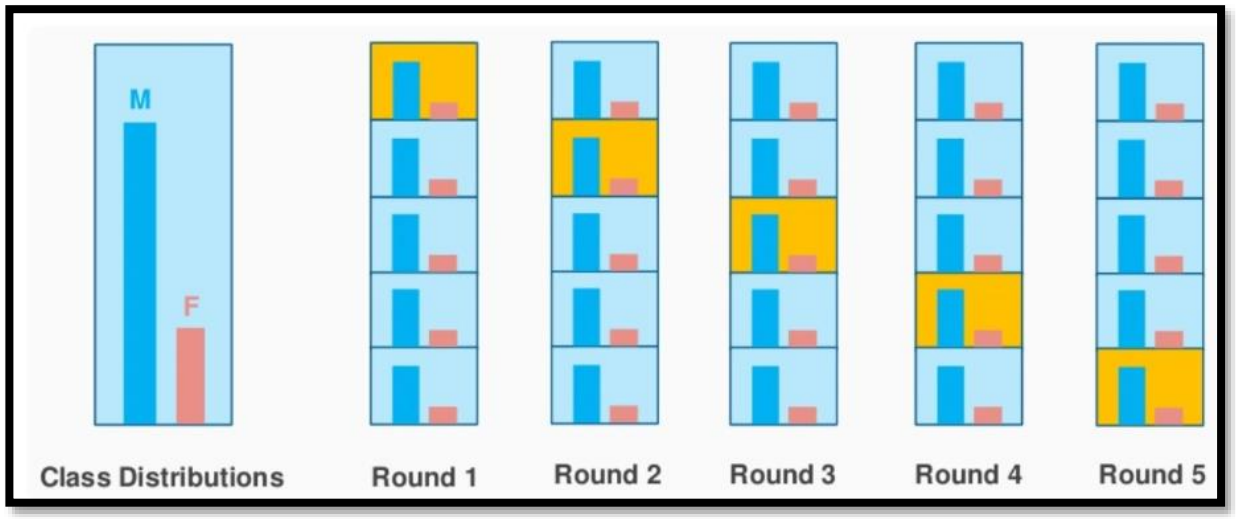


Figure 4.6: Stratified k-fold method

In the aforementioned illustration the dataset consists of Male (M) and Female (F). The original dataset has females that are exponentially lower than males, meaning this target variable has an imbalanced distribution. Stratified k-fold cross-validation maintains this target variable ratio of instances in all folds.

4. Leave-p-out cross-validation

An extensive cross-validation method – for example, if a dataset has n samples, p samples will be used as the validation set, and n-p samples will be used as the training set. One would then repeat this process until all samples have been used for each run, where the samples change each time so the last n-p samples will be the training set and p samples will be the validation set and so forth, until all samples have been used in a running process and considered each as a validation set.

The method is computationally intense, yet yields good results. However, it is not suitable for an imbalanced dataset, and is impractical as a mode of computational use. If all training set samples are of one class, the model will be unable to generalize properly and will be biased towards the other class.

5. Leave-one-out cross-validation

In this approach, only 1 observation datum is used as the validation set and all remaining n-1 observations are used to construct the training set. You can think of this as a more specific version of the leave-p-out cross-validation approach in which P=1. Figure 5.7 shows the process of Leave-one-out method.

Figure 5.7: Leave-one-out method

To better understand what this means, consider the following example:

Suppose you have a dataset with 1000 observations. In each iteration, you will leave 1 observation for the validation set and will use all remaining 999 observations to construct a training set.

This process is then repeated until every observation from the dataset has been used as a validation sample. The leave-one-out cross-validation method can be computationally expensive to conduct and ought to not be used with very large datasets. On the plus side, this approach is very simple to use and requires no parameter configuration to specify a single validation observation. Additionally, it can provide a trustworthy and unbiased estimation of your model performance.

Check Your Progress – 2

a) Train-test split helps detect overfitting and underfitting. (True/False)

b) Cross-validation involves splitting the dataset into multiple subsets or folds. (True/False)

c) Leave-One-Out Cross-Validation uses all data points for testing at once. (True/False)

d) A classifier that attains 100% accuracy on the training set and 70% accuracy on the test set is better than a classifier that attains 70% accuracy on the training set and 75% accuracy on test set. (True/False)

4.5 LET US SUM UP

This unit primarily explored the fundamentals of model evaluation in machine learning and its critical role in assessing model performance. It covered key classification metrics such as accuracy, precision, recall, and F1-score, along with regression metrics like MAE, MSE, and R^2. The train-test split technique was introduced as a basic evaluation method to test model generalization. To overcome limitations of single splits, cross-validation methods like k-fold were discussed for more reliable evaluation.

Overall, the unit emphasized selecting appropriate metrics and validation strategies to build robust and effective models.

4.6 CHECK YOUR PROGRESS: POSSIBLE SOLUTIONS

1-a False

1-b False

1-c True

1-d True

1-e False

1-f True

1-g True

2-a True

2-b True

2-c False

2-d False

4.7 ASSIGNMENTS

- Explain the importance of model evaluation in machine learning.
- Why is it necessary even after training the model with high accuracy on training data?
- Differentiate between the following metrics with examples:
  a) Precision b) Recall c) F1-Score d) Accuracy
- What is the role of Mean Absolute Error (MAE) and Mean Squared Error (MSE) in regression tasks?
- Explain R-squared (R^2) with an example.
- Discuss the concept of Train-Test Split.
- What problems may arise from using only a single train-test split to evaluate a model? How can they be addressed?
- Explain k-Fold Cross-Validation with a diagram.
- What is Stratified k-Fold Cross-Validation and when is it preferred over regular k-Fold?

Block-3

Neural Networks and Feature Engineering

Unit-1: Introduction to Neural Networks

1

Unit Structure

1.0. Learning Objectives

1.1. Overview of Artificial Neural Networks

1.2. Basic Concepts: Neurons, Layers, and Activation Functions

1.3. Key Architectures

1.4. Let us sum up

1.5. Check your Progress: Possible Answers

1.6. Assignment

1.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the foundational principles of Artificial Neural Networks (ANNs).
- Explain the basic concepts related to ANNs, including neurons, layers, and activation functions.
- Identify and describe key architectures such as Feedforward Neural Networks.

1.1 OVERVIEW OF ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) have significantly transformed the domain of artificial intelligence, becoming a foundational element in numerous AI-driven applications. Over the years, advancements in neural network architectures and methodologies have enhanced the capabilities of ANNs, enabling them to deliver superior outcomes across various AI projects.

ANNs are instrumental in empowering developers to create models that produce reliable and scalable results. As a result, they are often seen as an improvement over traditional machine learning algorithms in various tasks, particularly when dealing with complex and high-dimensional data. For example Neural networks are highly effective in tasks such as facial recognition and object detection in images. where traditional machine learning algorithms find it challenging.

What is an Artificial Neural Network?

An Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks in the human brain function. It is composed of interconnected layers of nodes (called neurons), where each node processes input data and passes the information to the next layer, ultimately producing an output.

As shown in figure 1.1, Artificial Neural Networks are composed of neurons, which act as basic processing units similar to neurons in the brain. After receiving information, neurons perform a mathematical operation to produce an output. The input layer of the network accepts data; hidden layers perform computations to identify patterns; and the output layer generates predictions. Weights and biases regulate the connections between neurons by determining the relevance of the input and modifying the output. By introducing non-linearity, activation functions like sigmoid, ReLU, and tanh enable the network to record complex interactions.

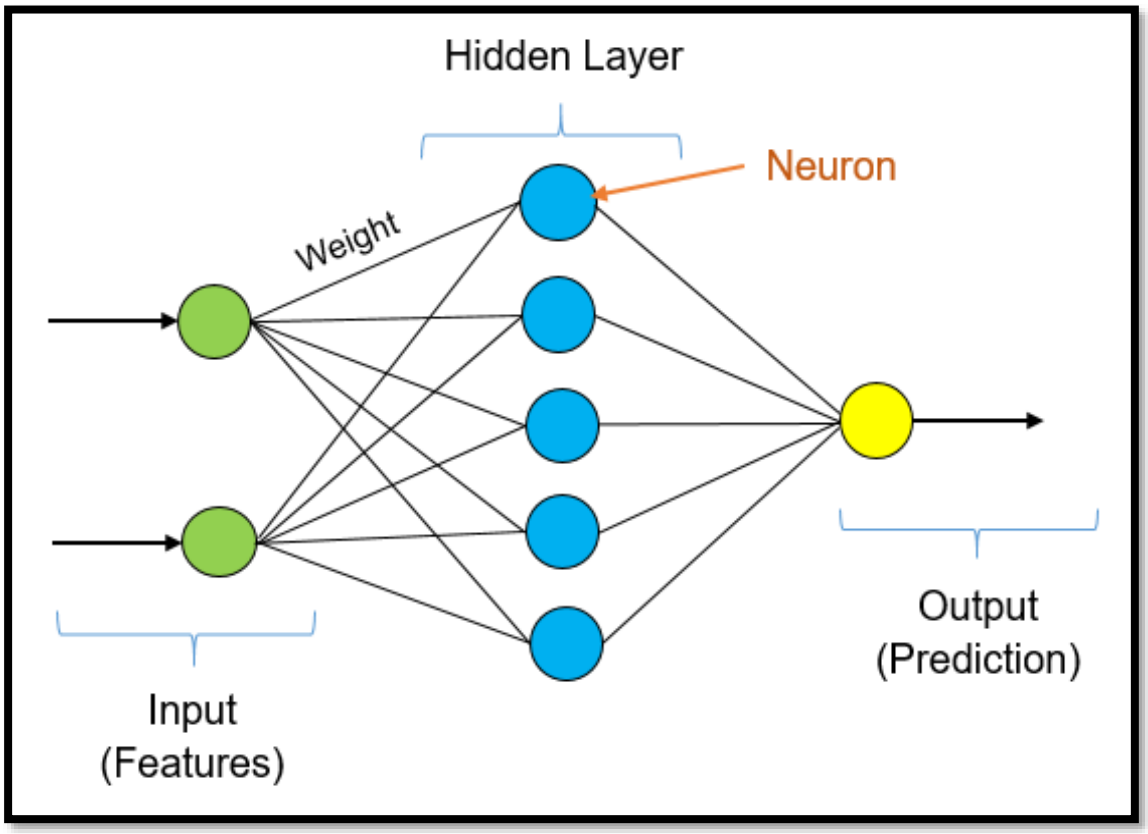


Figure 1.1: Artificial Neural Network

1.2 BASIC CONCEPTS: NEURONS, LAYERS, AND ACTIVATION FUNCTIONS

Neural network consists of several concepts that work together to process and learn from data. The concepts have been discussed herewith:

Neuron

Artificial neural networks are widely used machine learning methods that mimic the learning process of biological organisms. In the human nervous system, cells known as neurons are interconnected by axons and dendrites. The junctions where these axons and dendrites meet are called synapses, as shown in figure 1.2. The strength of synaptic connections typically adjusts in response to external stimuli, facilitating the learning process in living beings. This biological phenomenon is emulated in artificial neural networks, where computational elements, referred to as neurons, perform similar functions.

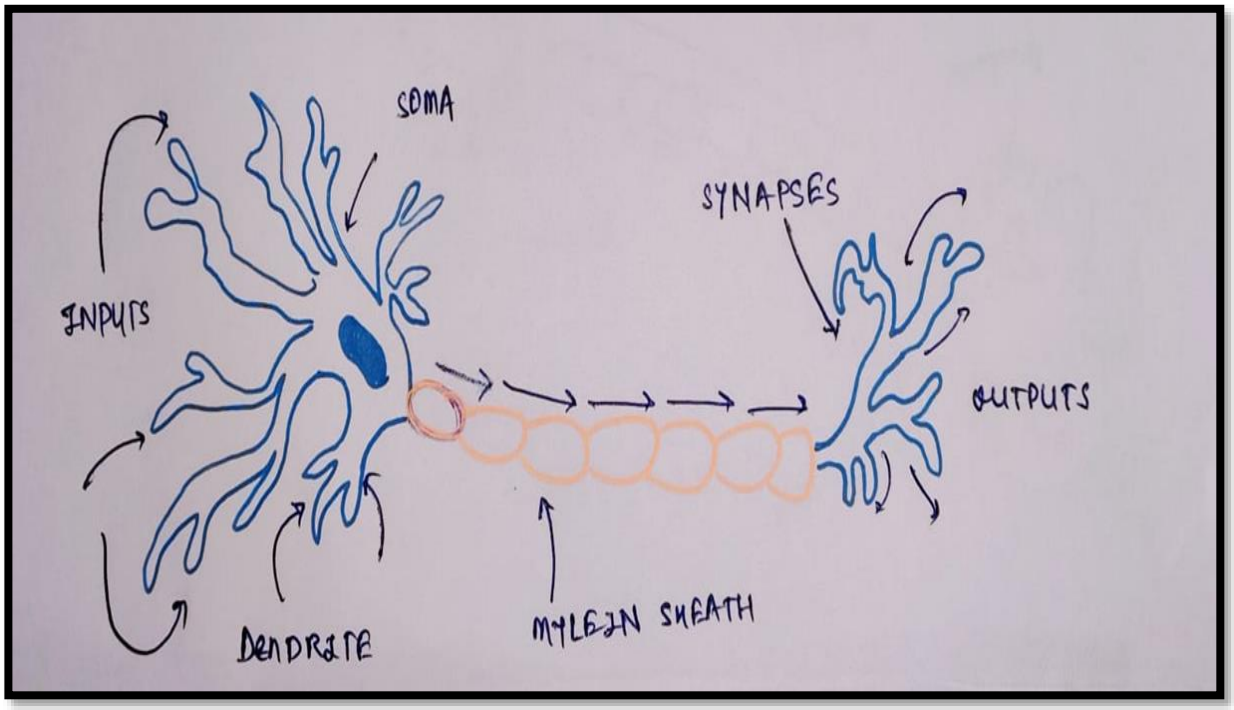


Figure 2. A Biological Neuron

An artificial neuron (also referred to as a perceptron) shown in figure 1.3 is a mathematical function. It takes one or more inputs that are multiplied by values called “weights” and added together. This value is then passed to a nonlinear function, known as an *activation function*, to become the neuron’s output.

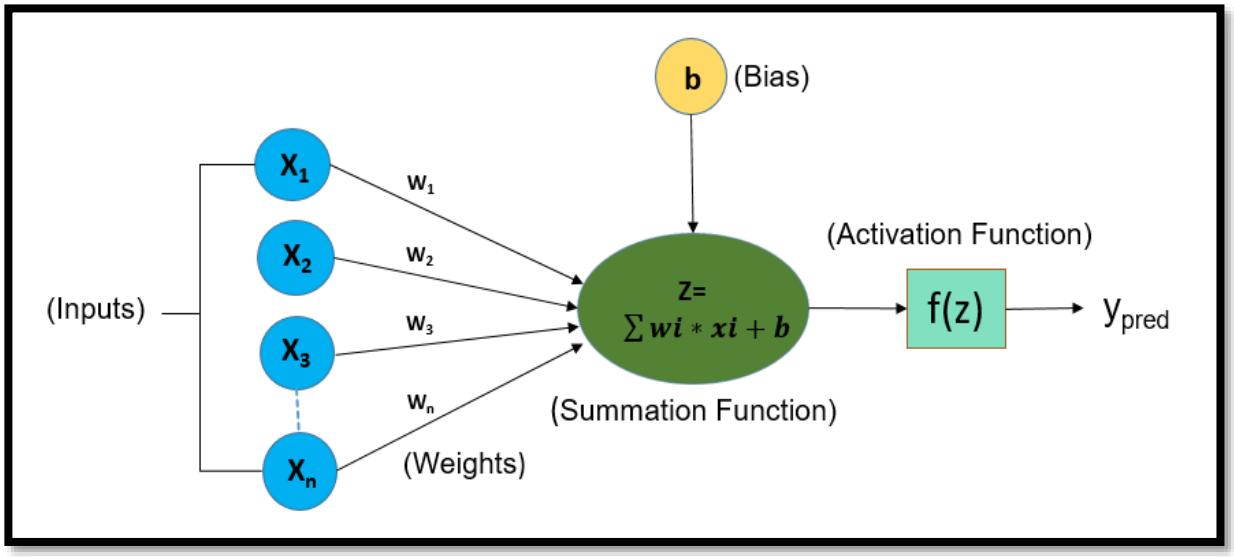


Figure 1.3: An Artificial Neuron

Components of Artificial Neuron:

The artificial neuron consists of five components namely; input values, weights, bias, weighted sum and activation function. This section describes them in brief.

1. **Input Values (x)**: An artificial neuron takes input values (also known as features) from the preceding layer or from the input data directly. Each input is assigned a weight that signifies its importance.
2. **Weights (w):** Each input value is multiplied by a weight, which affects its importance in the computation of the neuron. During training, weights are modified to allow the network to learn from data.
3. **Bias (b):** To the weighted total of inputs, a bias factor is applied. It enables the neuron to control its output independently of its inputs. Bias aids in shifting the curvature of the activation function.
4. **Weighted Sum (z)**: The following formula is used to determine the weighted sum of inputs and bias:

   $z = (w_1 * x_1) + (w_2 * x2) + ... + (w_n * x_n) + b$, where w is the weight, x denotes the input value, n denotes the number of inputs, and b denotes the bias.
5. **Activation Function(f(z)):** To incorporate non-linearity, the weighted sum is processed via an activation function (also known as a transfer function). ReLU (Rectified Linear Activation), sigmoid, and tanh are examples of common

activation functions. The neuron's output is determined by the activation function. The neuron's output, represented as y_{pred}, is the result of applying the activation function to the weighted sum and bias:

$$y_{pred} = \text{activation\_function}(f(z))$$

Calculating the weighted sum of inputs, adding the bias, sending the result through an activation function, and producing the output are all steps in the computation of an artificial neuron. This output is subsequently used as input by other neurons in the neural network's succeeding layers.

An artificial neural network's architecture is made up of numerous interconnected neurons organized into layers. Based on the data it is trained on, this network can learn to approximate complex functions, recognize patterns, and make predictions.

Layers

A basic neural network has interconnected artificial neurons in three layers:

1. **Input Layer:** Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.
2. **Hidden Layer:** Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
3. **Output Layer:** The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

Activation Functions

In neural networks, an activation function is a mathematical function that, given an input, determines the output of a neuron. It is a function that is intended to "activate" the neuron, as the name implies.

Neural network neurons function similarly to brain neurons in that they receive signals from the body and decide how best to process them. They perform the role of transfer functions, taking in values as inputs and generating equivalent outputs.

Why Are Activation Functions Essential?

Adding non-linearity to the neural network is the aim of an activation function. Activation functions add a step at every layer in the forward propagation process, but the computational cost is justified. Here's the reason:

Assume for the moment that the activation functions of a neural network are absent. In that scenario, the only operation that each neuron does on the inputs is a linear transformation using the weights and biases. The reason behind this is that, since the composite of two linear functions is a linear function in and of itself, all hidden layers in a neural network will act in the same manner regardless of the number of layers we attach.

By incorporating non-linear behaviors through activation functions, neural networks are able to learn these non-linear correlations. This significantly boosts neural networks' adaptability and capacity to represent intricate and nuanced input.

The Activation Functions can be basically divided into three types:

1. **Binary Step Function:** The most crucial factor to take into account when using an activation function is a threshold-based classifier, which indicates whether or not the value of the linear transformation is required to activate the neuron. Alternatively put, we can state that a neuron is activated if its input exceeds a

threshold value, and deactivated otherwise. The output is then not used as an input by the next hidden layer. The binary step function is shown in figure 1.4.

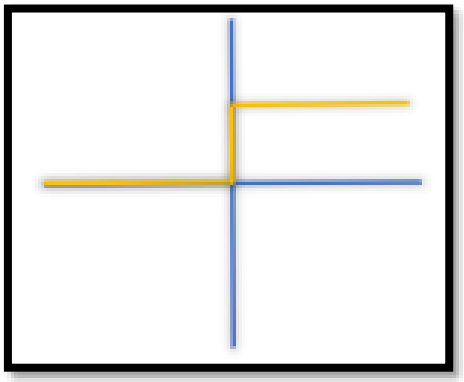


Figure 1.4: Binary Step Function

2. **Linear Activation Function:** Because the output of a linear activation function as shown in figure 1.5. is always proportionate to the input, it is often referred to as an "identity function" or "no activation."

   Therefore, in theory, such a function returns the value fed into the network and gives you the weighted sum of the input.

   Mathematically represented as: **f(x)=x**

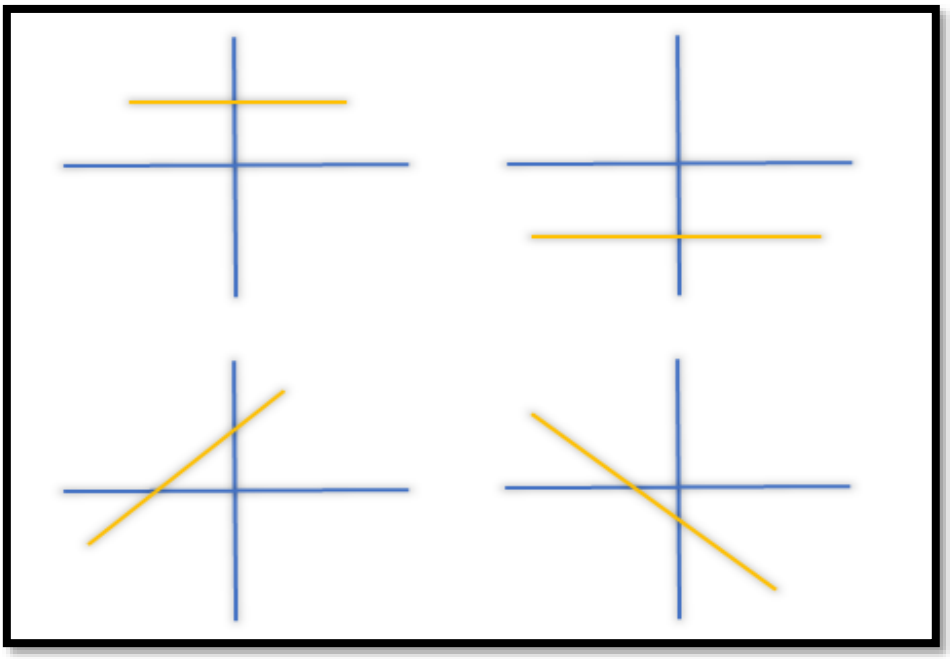


Figure 1.5: Linear Function

3. **Non-linear Activation Functions:** The most popular activation functions are the nonlinear ones shown in figure 1.6. It facilitates the model's ability to distinguish between outputs and to generalize or adapt to a variety of data.

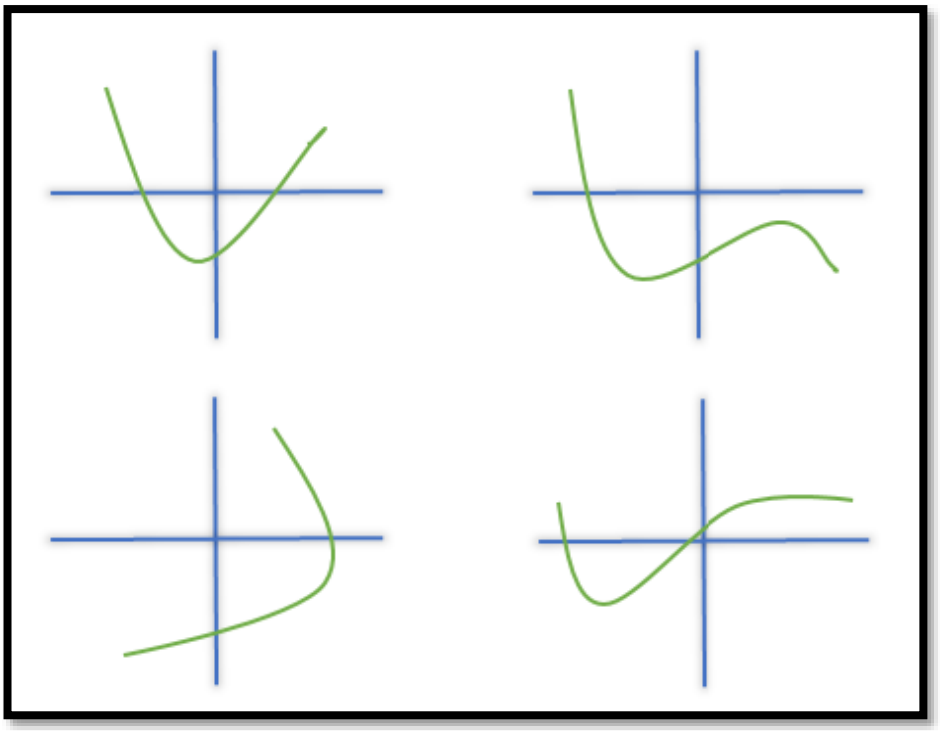


Figure 1.6: Non-Linear Function

Check Your Progress-1

a) What is the primary function of the input layer in an Artificial Neural Network?
   a) To process data
   b) To receive initial data for processing
   c) To provide final predictions
   d) To adjust weights

b) What is the purpose of hidden layers in an ANN?
   a) To store the final output
   b) To receive input data
   c) To perform computations and learn complex patterns
   d) To initialize the network

c) In an ANN, what do weights determine?
   a) The number of neurons
   b) The importance of inputs in processing

c) The final output
d) The learning rate

d) What process is used to minimize error in an ANN during training?
   a) Forward propagation
   b) Weight initialization
   c) Backpropagation
   d) Activation function adjustment

e) Which layer in an ANN provides the final output or prediction?
   a) Input Layer
   b) Hidden Layer
   c) Output Layer
   d) Activation Layer

f) What is the role of an activation function in a neuron?
   a) To adjust weights
   b) To determine the importance of the neuron
   c) To introduce non-linearity in the output
   d) To store input data

1.3 KEY ARCHITECTURES OF NEURAL NETWORK

The number and kinds of layers in a neural network, as well as their organization, are referred to as its architecture. Let's dive in and explore key Architectures of Neural Network:

1. Feedforward Neural Networks

One of the more fundamental types of neural networks is a feedforward, and its architecture is frequently used to build more specialized networks. Feedforward neural networks, as the name implies, transfer data forward; that is, without loops or circles, from input to output. Even though it's among the most basic neural network designs, the hidden layers that connect input and output can nevertheless be quite intricate. This kind of neural network can be used for a number of tasks, including classification, regression analysis, and pattern and picture recognition.

How Feedforward Neural Networks Work?

An input layer precedes a number of hidden layers in a feedforward neural network, and it concludes with an output layer. Via the input, data enters the algorithm and moves through the first layer's nodes. By using each node's weight to calculate the data, the first layer of nodes transfers the computation to the subsequent layer of nodes. The data can only move in the direction of the output, even though every node in one layer is connected to every other layer's node.

Benefits:

- **Simplicity:** The construction of FNNs is simple; data flows from input to output in a single direction.
- **Versatility:** May be applied to a variety of issues, including regression, classification, and (with some adjustments) unsupervised learning.
- **Faster to Train:** Compared to RNNs, training may be done more easily and quickly because there is no feedback or recurrent link.

Challenges:

- **Lack of Memory:** FNNs are not appropriate for sequential data such as time series, speech, or text since they are unable to remember the past.
- **Data Requirements:** Overfitting occurs when the dataset is small, and requires a significant amount of labeled data for training.
- **Complexity Growth:** The number of parameters may increase dramatically as input dimensions increase, making training more challenging.

2. Convolutional Neural Networks

Convolutional networks, also known as convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional Networks are simply neural networks that use convolution in place of general matrix multiplication in at least one

of their layers. CNNs are very useful for applications like object identification, picture segmentation, and image classification because of their capacity to extract hierarchical representations and local patterns.

How Convolutional Neural Networks Work?

CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers in addition to input and output layers. Convolutional Neural Networks (CNNs) work by hierarchically learning features from input data, typically images, to make predictions or classifications.

The process begins with the **input layer**, where the image, represented as a 2D or 3D grid of pixel values, is fed into the network. The first key operation occurs in the **convolutional layers**, where small filters (kernels) are applied to the image. These kernels perform convolution, a mathematical operation that extracts features like edges, textures, and patterns by sliding over the image and computing weighted sums at each position. The output of this operation is a feature map, which represents the presence of detected features at various locations in the image. To introduce non-linearity and enable the network to learn complex patterns, an **activation function**, typically ReLU (Rectified Linear Unit), is applied after the convolution.

The network then uses **pooling layers** to reduce the spatial dimensions of the feature maps, typically using max pooling or average pooling, which simplifies the representation and reduces computation while retaining important information. As data progresses through deeper layers, the network extracts more abstract and high-level features. In the **fully connected layers**, the high-level features from the convolutions and pooling are flattened into a vector and passed through one or more layers of neurons to map the features to the final output, such as class probabilities for classification tasks.

The **output layer** applies an activation function like softmax (for classification) or a linear function (for regression) to produce the final prediction. During **training**, the network's parameters, such as the kernels, are optimized using algorithms like

backpropagation and **gradient descent**, which adjust the parameters to minimize the error between the predicted outputs and the actual labels. This process enables the network to automatically learn relevant features from data, making CNNs highly effective for tasks like image recognition, object detection, and segmentation. Figure 1.7 shows the architecture of a CNN.

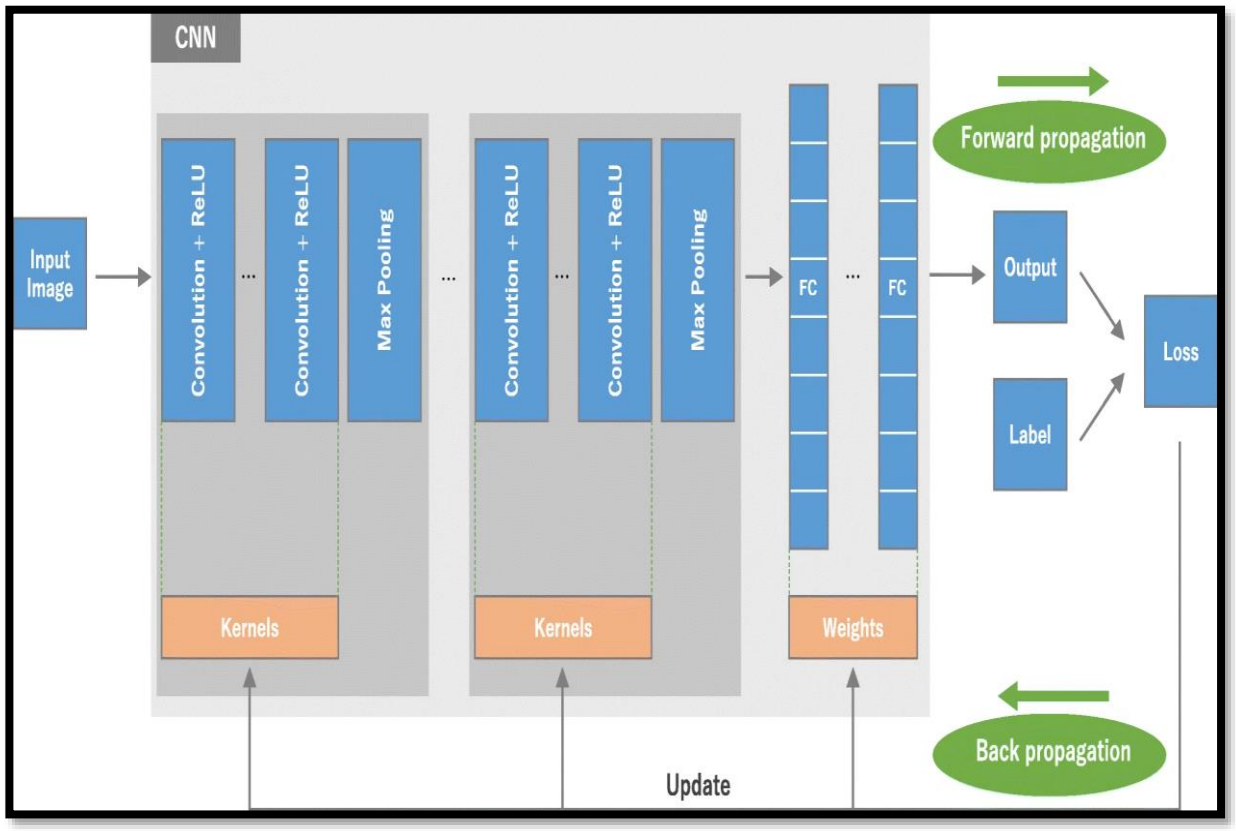


Figure 1.7: Architecture of CNN

Benefits:

- **Spatial Hierarchy:** CNNs are perfect for computer vision tasks like object recognition and picture classification because of their superior ability to capture spatial characteristics in images through convolutional layers.
- **Sharing of parameters:** By using filters, the network can share parameters, which drastically lowers the number of parameters and improves computing performance.
- **Translation Invariance:** CNNs' filters are resilient to translation since they can identify patterns in any part of the input.

Challenges:

- **Application-specific:** RNNs perform better than CNNs for sequential tasks (such as text or time-series data) even if CNNs perform better in jobs like image and video processing.
- **Requirement for Big Datasets:** For training, CNNs, particularly deep CNN architectures, usually need big volumes of labeled data as well as a lot of processing capacity.
- **Overfitting:** On small datasets, CNNs with deep architectures are prone to overfitting, which may call for regularization strategies like dropout and data augmentation.

3. Recurrent Neural Network

A Recurrent Neural Network (RNN) is a specialized type of neural network designed for time series predictions. It learns patterns from previous data points and uses this knowledge to predict future values. The neurons in the hidden layers act as memory units, retaining outputs from prior steps to inform subsequent predictions. In an RNN, data points from earlier steps are continually utilized for each prediction, making it a recurrent process. While the network can store a limited sequence of past outputs, it is not well-suited for handling longer sequences.

How Recurrent Neural Networks Work?

Recurrent networks have a design similar to feedforward neural networks, but they also use loops to circle back through the hidden layers with the data and return an output. Context layers are specific hidden layers that are occasionally included in recurrent neural networks. These layers give feedback to the neural network and aid in its accuracy.

Benefits:

- **Memory Retention:** RNNs are perfect for time-series forecasting, speech recognition, and natural language processing because they can remember past information in sequences.
- **Sequential Data:** Ideal for applications such as handwriting recognition and language modeling where the context of earlier inputs is important.

- **Recurrent Connections:** These enable RNNs to modify the current output by utilizing data from earlier time steps.

Challenges:

- **Vanishing/Exploding Gradient Problem:** The gradients can either become too small (vanish) or too large (explode), during backpropagation through time, making training difficult.
- **Long-Term Dependencies:** Standard RNNs struggle with learning long-term dependencies (addressed to some extent by LSTMs and GRUs).
- **Training Complexity:** RNNs generally take longer to train compared to FNNs and CNNs due to their recurrent structure.

Understanding the Differences in Neural Networks

Neural networks come in various architectures, each designed to handle different types of data and tasks. Table 1.1 gives a comparison between FNN, CNN and RNN.

Category	Feedforward Neural Networks (FNN)	Convolutional Neural Networks (CNN)	Recurrent Neural Networks (RNN)
Definition	A type of artificial neural network where information moves in only one direction, from input to output.	A class of deep neural networks most commonly applied to analyzing visual imagery.	A class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence.
Data Type	Suitable for structured data, where inputs do not have temporal or spatial dependencies.	Suitable for spatial data like images.	Suitable for temporal data (sequential data), such as time-series or text.

Structure	Simple feed-forward architecture with no feedback connections.	A feed-forward artificial neural network with variations designed to use minimal amounts of preprocessing, using convolutional layers	RNNs have internal memory (feedback connections) that allow them to process arbitrary sequences of inputs.
Power Comparison	FNN is generally simpler but less powerful compared to CNN and RNN in handling complex data types.	CNN is considered to be more powerful than RNN for spatial data analysis.	RNN has less feature compatibility compared to CNN but is superior in sequential processing.
Input/Output Handling	Typically handles fixed-size inputs and outputs, lacking the flexibility of RNNs for sequential or variable data.	Takes fixed-size inputs and generates fixed-size outputs.	Can handle arbitrary input/output lengths, making them flexible for sequence data.
Memory Retention	No memory retention; each input is treated independently of others.	No memory capability; processes data independently of previous inputs.	Retains memory of previous inputs due to its recurrent connections, useful for time-dependent patterns.
Applications	Binary Classification, Multiclass Classification, Regression Tasks, Anomaly Detection, and recommendation Systems.	Image Recognition, Image Classification, Medical Image Analysis, Face Detection, and Computer Vision.	Text Translation, Natural Language Processing (NLP), Sentiment Analysis, Speech Recognition.

Table 1.1 FNN vs. CNN vs. RNN

Check Your Progress-2

a) Data in a feedforward neural network flows in a looped or cyclic manner. (True/False)

b) The main challenge for RNN is gradient vanishing and exploding. (True/False)

c) CNN is most suitable for tasks like image recognition and classification. (True/False)

d) Pooling layers in CNNs are used to increase the spatial dimensions of the feature maps. (True/False)

e) The purpose of the ReLU activation function in CNNs is to introduce non-linearity and enable the network to learn complex patterns. (True/False)

1.4 LET US SUM UP

In this unit, we explored how Artificial Neural Networks (ANNs) have transformed the field of artificial intelligence, providing enhanced performance in numerous AI applications. ANNs, composed of interconnected layers of neurons, emulate the structure of the human brain's neural networks. Each neuron processes input data, which is then passed through activation functions to introduce non-linearity. These networks are made up of input, hidden, and output layers that collaborate to make predictions. The training process involves adjusting neuron weights through backpropagation to reduce errors and improve overall performance. Feedforward neural networks (FNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) are examples of neural network designs. Depending on the type of data and the purpose, each design offers unique advantages and disadvantages.

1.5 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a to receive initial data for processing

1-b to perform computations and learn complex patterns

1-c the importance of inputs in processing

1-d Backpropagation

1-e Output Layer

1-f To introduce non-linearity in the output

2-a False

2-b True

2-c True

2-d False

2-e True

1.6 ASSIGNMENTS

- What is the importance of artificial neural networks?
- What are the most important advantages of using neural networks?
- Explain the architecture of an Artificial Neural Network (ANN). Include a description of the different layers and their functions.
- Discuss the role of activation functions in ANNs. Why is it important to introduce non-linearity in the network? Provide examples of common activation functions.
- What are different layers of neural network?
- Differentiate between CNN and RNN.
- Explain different layers of CNN.

Unit-2: Activation Functions 2

Unit Structure

2.0. Learning Objectives
2.1. Introduction
2.2. Need of Activation Function
2.3. Need for Nonlinearity in Neural Network
2.4. Types of Activation Functions
2.5. Choosing right activation function
2.6. Let us sum up
2.7. Check your Progress: Possible Answers
2.8. Assignment

2.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the role of activation functions.
- Explore different types of activation functions.
- Learn Strategies for choosing the right activation function.

2.1 INTRODUCTION

Activation functions are integral to artificial neural networks, it transforms the input signals into output signals passed on to subsequent layers. In a neural network, the output of each layer is determined by computing the weighted sum of inputs, followed by applying an activation function. This output is then used as input for the following layer. The accuracy of neural networks in making predictions is influenced by both the number of layers and the type of activation function utilized. While no strict guidelines exist regarding the minimum or maximum number of layers, it is generally recommended to use at least two layers to achieve meaningful outcomes.

2.2 NEED OF ACTIVATION FUNCTION

An activation functions are essential to the functioning of neural networks, as they introduce the capacity to learn and model complex data patterns. Without them, neural networks would be reduced to simple linear regression models, offering limited ability to handle intricate real-world data. Activation functions enable non-linearity, allowing the network to capture complex relationships within the data and perform more advanced tasks. By controlling whether neurons should fire or not, activation functions simulate decision-making processes, improving the network's ability to generalize predictions.

Moreover, activation functions influence how efficiently a neural network is trained. The choice of activation function determines the gradient in backpropagation, which affects weight adjustments during learning. Various activation functions, such as ReLU, sigmoid, and tanh, have distinct properties that cater to different types of neural networks. The right activation function not only enhances feature learning but also helps prevent common training issues like vanishing or exploding gradients, which can impede the learning process.

2.3 NEED FOR NONLINEARITY IN NEURAL NETWORK

Functions with a degree greater than one, which exhibit curvature when plotted, are referred to as non-linear functions. For a neural network to effectively learn, represent, and process various forms of data, it must be capable of handling any arbitrary, complex function that maps inputs to outputs. Neural networks are also recognized as Universal Function Approximators, meaning they have the ability to compute and learn any given function. Essentially, any conceivable process can be represented as a functional computation within neural networks.

To enable neural networks to dynamically extract finer details and complex information from data, an activation function is necessary. This activation function introduces non-linearity, allowing the network to model the non-linear relationships between inputs and outputs, which is crucial for representing the complex functional mappings in real-world data. By incorporating non-linear activation functions, neural networks can perform non-linear transformations from input to output. Additionally, an activation function must be differentiable, as this allows the implementation of backpropagation to compute errors or losses concerning the weights. This enables the network to optimize the weights using techniques like gradient descent, ultimately reducing errors and improving performance.

Check Your Progress-1

a. What does the term ‘activation function’ refer to in neural networks?
   a) A function that scales input data
   b) A function that determines the output of a neuron
   c) A function that initializes weights
   d) A function that optimizes the model

b. What is the primary purpose of an activation function in a neural network?
   a) To adjust the learning rate of the network
   b) To transform the input data to a normalized form
   c) To introduce nonlinearity to the network, enabling it to learn complex patterns
   d) To reduce the size of the neural network

c. In practice, what is the most accurate description for activation functions used in neural networks?
    a) They must be differentiable.
    b) They can be non-differentiable, but only for a small amount of points
    c) They can be any continuous functions.
    d) They must be non-linear to be learnable.

2.4 TYPES OF ACTIVATION FUNCTIONS

The key elements within a neural network are its net inputs, which undergo processing and are transformed into outputs through a scalar-to-scalar transformation, often referred to as the activation function, transfer function, or threshold function. Squashing functions are used to confine the output of a neuron within a specific range and limit the signal's amplitude. These functions effectively compress the output signal to a finite value.

The Activation Functions can be basically divided into 3 types:

1. Binary Step Function
2. Linear Activation Function
3. Non-linear Activation Functions

1. Binary Step Function

The most basic activation function available is the Binary Step Function, which is implemented in Python using basic if-else expressions. It is common practice to employ binary activation functions while developing binary classifiers. However, in the event that the target carriage has many classes, the binary step function is not applicable.

Also, the gradient of the binary step function is zero which may cause a hindrance in back propagation step i.e if we calculate the derivative of f(x) with respect to x, it is equal to zero.

Mathematically the binary step function can be represented as shown herewith:

$$f(x) = 0 \, for \, x < 0$$

$$f(x) = 1 \, for \, x \geq 0$$

Figure 2.1 is the visual representation of the binary step function.

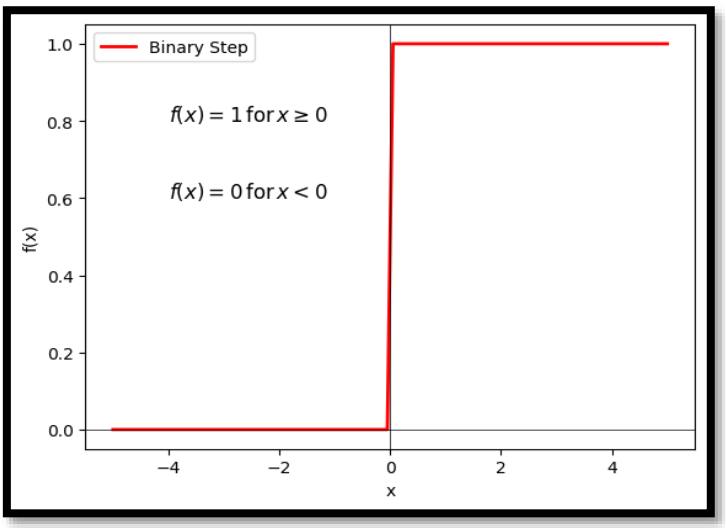


Figure 2.1: Binary Step Function

2. Linear Activation Function

When data scientists want the output of a neural network to be identical to the input signal, they employ linear activation functions, sometimes referred to as identity functions. Internal layers of a neural network do not apply this activation function because identity is differentiable and, like a train moving through a station without stopping, it does not alter the signal in any way.

This may not seem particularly helpful in most situations, but it is when you want your neural network's outputs to be continuous instead of discrete or changed. Nothing declines, and the data does not converge. The layers in a neural network would collapse into one if this activation function were applied to each layer. Therefore, it isn't particularly helpful unless you really need it or the successive concealed layers have different activation functions.

Mathematically it can be represented as shown herewith:

$$f(x) = x$$

Figure 2.2 is the visual representation of the linear activation function.

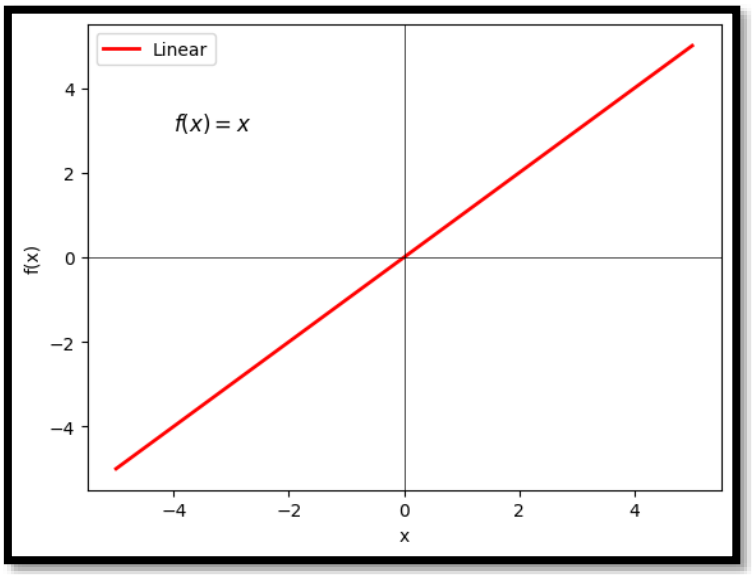


Figure 2.2: Linear Activation Function

However, a linear activation function has two major problems :

- It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

3. Non-linear Activation Functions

A linear regression model is all that the linear activation function above represents. This limits the model's ability to create complex mappings between the network's inputs and outputs.

The constraints of linear activation functions are addressed by non-linear activation functions.

- Given that the derivative function is now connected to the input, they permit backpropagation, which makes it feasible to determine which input neuron weights are most suited for making predictions.
- As a result of the input being carried via many layers in a non-linear fashion, they enable the stacking of numerous layers of neurons. Any output can be represented in a neural network as a functional computation.

Non-linear activation functions are mainly divided basis on their range or curves as **Sigmoid, Tanh, ReLU, Leaky ReLU, Parametrized ReLU, Exponential Linear Unit, Swish, SoftMax, GeLU** and **SeLU.** Let us look at each of them in brief.

1. Sigmoid / Logistic Activation Function

Sigmoid Activation function is very simple which takes a real value as input and gives probability that's always between 0 or 1. It looks like 'S' shape.

Mathematically it can be represented as:

$$f(x) = 1 / 1 + e^{-x}$$

The sigmoid/logistic activation function is one of the most used functions for the following reasons:

- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

Figure 2.3 is the visual representation of the Sigmoid Activation Function and Its Derivative.

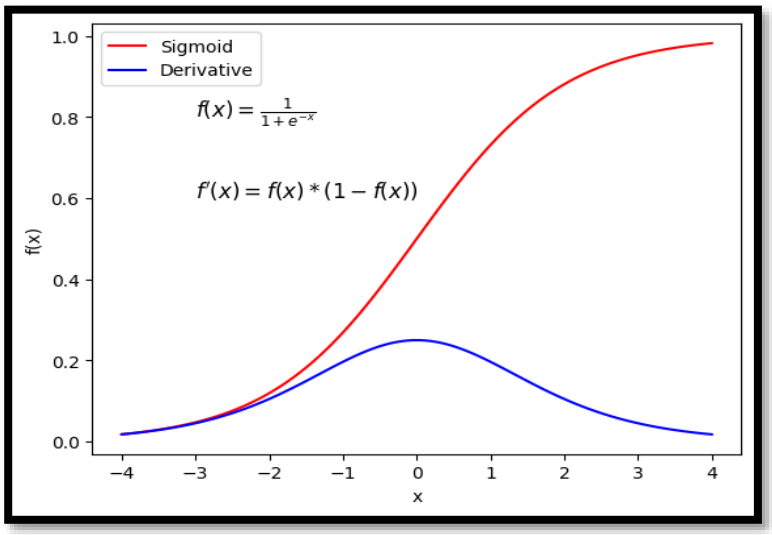


Figure 2.3: Sigmoid Activation Function and Its Derivative

The following paragraph describes the sigmoid function's limitations:

- The derivative of the function is $f'(x) = f(x) * (1 - f(x))$
- According to the above figure, the gradient values are only significant for the range -3 to 3, and the graph gets much flatter in other regions.
- It implies that for values greater than 3 or less than -3, the function will have very small gradients. The network experiences the Vanishing Gradient Problem and stops learning when the gradient value gets closer to zero.
- Around zero the logistic function's output is not symmetrical. Consequently, every neuron's output will have the same sign. As a result, neural network training becomes more challenging and unstable.

2. Tanh Function (Hyperbolic Tangent)

The Tanh function closely resembles the sigmoid or logistic activation function, sharing an S-shaped curve. However, a key distinction lies in its output range, which spans from -1 to 1. For larger (more positive) input values, the output of Tanh approaches 1.0, while for smaller (more negative) inputs, the output nears -1.0.

Mathematically it can be represented as:

$$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$

Figure 2.4 is the visual representation of the Tanh Activation Function and Its Derivative.

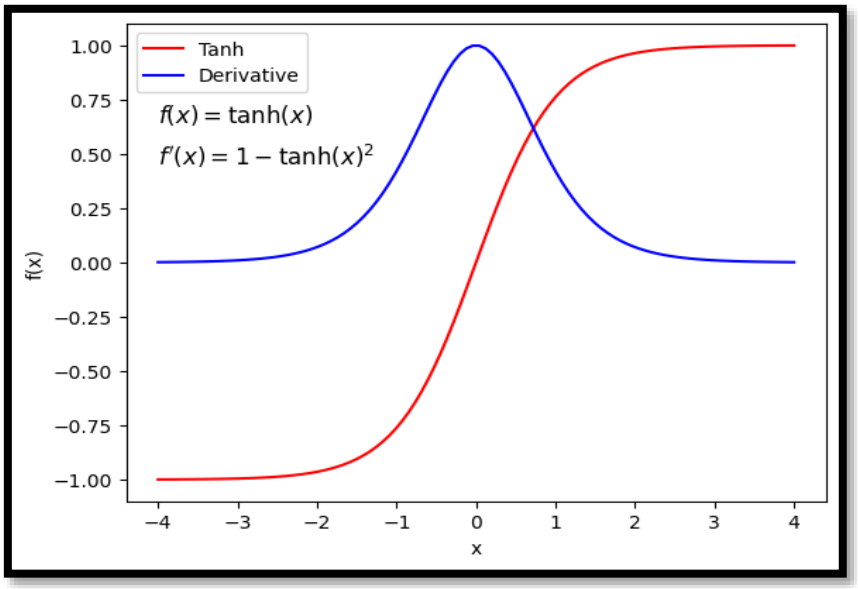


Figure 2.4: Tanh Activation Function and Its Derivative

Advantages of using the Tanh activation function include:

- Its output is zero-centered, which allows for easier interpretation of the results as strongly negative, neutral, or strongly positive.
- It is commonly used in the hidden layers of neural networks because its output values range from -1 to 1. This property helps ensure that the mean of the hidden layer is close to zero, facilitating better data centering and simplifying learning for subsequent layers.

Examining the gradient of the Tanh activation function reveals its limitations.

- Like the sigmoid function, Tanh also suffers from the vanishing gradient problem.
- Additionally, the gradient of the Tanh function is steeper compared to that of the sigmoid function.

Note:
Although both sigmoid and Tanh functions experience the vanishing gradient problem, Tanh has the advantage of being zero-centered, allowing gradients to move more freely without being constrained in a specific direction. As a result, Tanh nonlinearity is generally preferred over sigmoid nonlinearity in practice.

3. ReLU Function

ReLU, which stands for Rectified Linear Unit, may seem like a linear function, but it has a derivative, enabling backpropagation and making it computationally efficient. One notable aspect of ReLU is that it does not activate all neurons simultaneously. Neurons become deactivated only when the output of the linear transformation is less than zero.

Mathematically it can be represented as:

$$f(x) = max(0, x)$$

The advantages of using ReLU as an activation function include:

- ReLU is more computationally efficient than sigmoid and Tanh functions because only a subset of neurons is activated at any given time.
- Its linear, non-saturating nature speeds up the convergence of gradient descent towards the global minimum of the loss function.

Figure 2.5 is the visual representation of the ReLU Activation function.

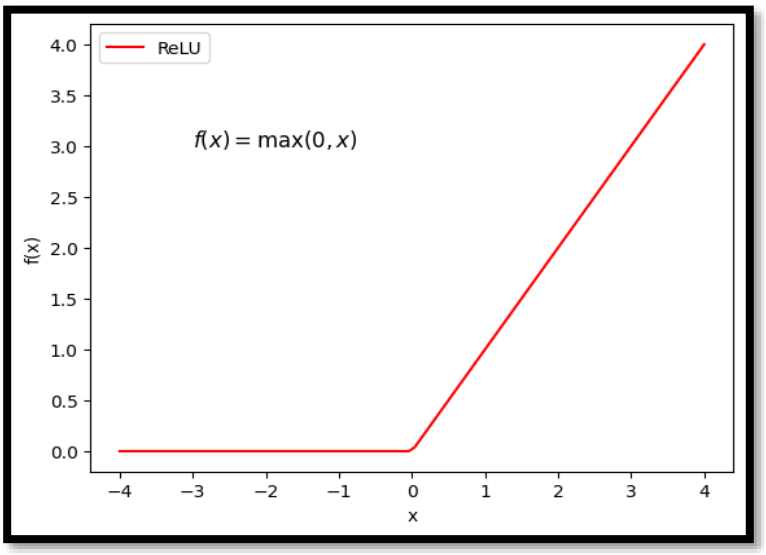


Figure 2.5: ReLU Activation Function

The limitation faced by this function is:

- This function encounters a limitation known as the "Dying ReLU" problem. In this issue, neurons can become inactive if they consistently output zero for certain inputs, effectively "dying" and ceasing to contribute to the learning process. This occurs because the gradient becomes zero, preventing any further updates to those neurons during training.

Figure 2.6 is the visual representation of the dying ReLU problem.

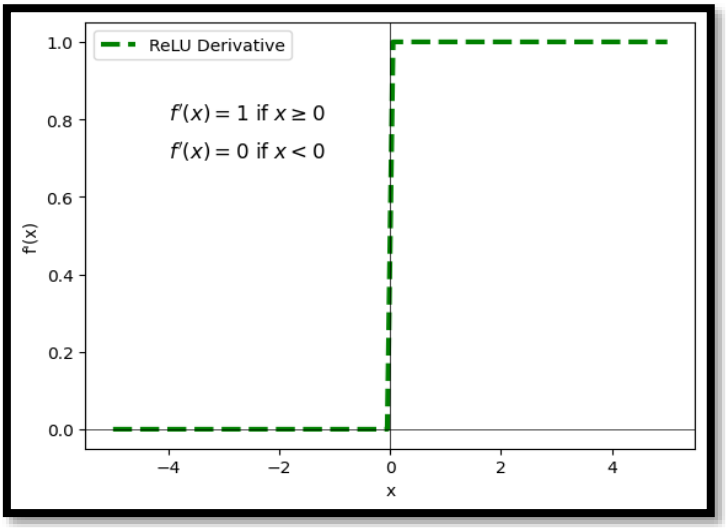


Figure 2.6: The Dying ReLU Problem

4. Leaky ReLU Function

Leaky ReLU is a modified version of the ReLU function designed to address the Dying ReLU problem. It introduces a small positive slope for negative input values, ensuring that neurons do not become completely inactive.

Mathematically it can be represented as:

$$f(x) = x \quad if\ x \geq 0$$

$$f(x) = \alpha x \quad if\ x < 0$$

where α is a small positive constant (e.g., 0.05) that defines the slope for negative values of x. Figure 2.7 is the visual representation of the Leaky ReLU activation function.

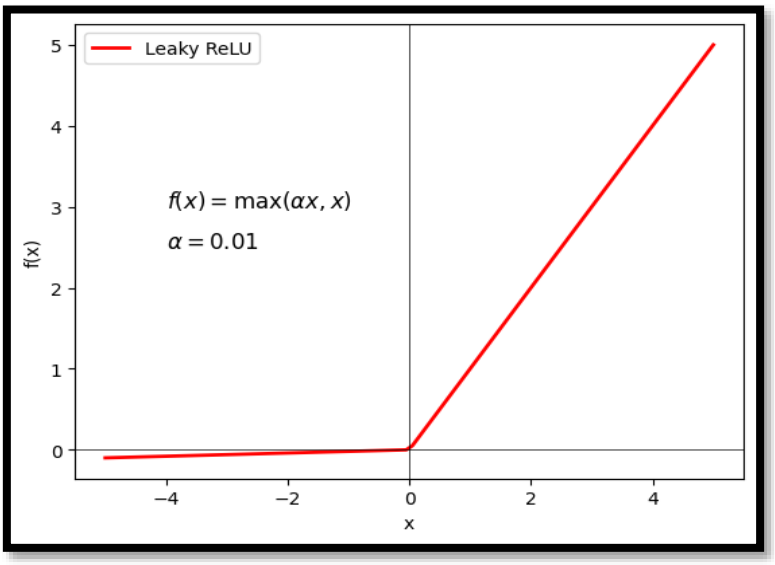


Figure 2.7: Leaky ReLU Activation Function

The advantages of using Leaky ReLU as an activation function include:

- Leaky ReLU helps prevent the Dying ReLU problem by allowing a small gradient for negative inputs, keeping neurons active.
- It maintains non-linearity, which is essential for modeling complex data relationships.

The limitations that this function faces include:

- The predictions may not be consistent for negative input values.
- The gradient for negative values is a small value that makes the learning of model parameters time-consuming.

5. Parametrized ReLU

Parametric ReLU is a variation of the ReLU function designed to address the issue of gradients becoming zero for negative input values.

This function allows the slope of the negative side to be defined by a parameter a. During backpropagation, the model learns the optimal value of a for improved performance.

Mathematically it can be represented as:

$$f(x) = max(ax, x)$$

Where *"a"* is the slope parameter for negative values. Figure 2.8 is the visual representation of the PReLU activation function.

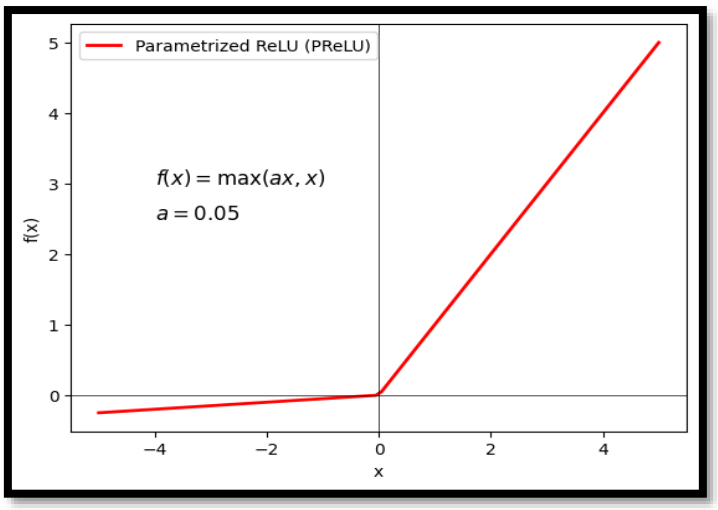


Figure 2.8: Parametrized ReLU (PReLU) Activation Function

The advantage of using parameterized ReLU as an activation function include:

- The parameterized ReLU function is utilized when the leaky ReLU function does not fully address the issue of dead neurons, resulting in relevant information not being effectively transmitted to the next layer.

The limitation that this function faces include:

- The performance can vary across different problems, depending on the chosen value of the slope parameter a.

6. Exponential Linear Unit

Another variation of ReLU that alters the slope of the function's negative portion is the Exponential Linear Unit, or ELU for short.

In contrast to the leaky ReLU and parametric ReLU functions, which utilize a straight line to determine the negative values, ELU uses a log curve.

Mathematically it can be represented as:

$$f(x) = x \qquad if\ x \geq 0$$
$$f(x) = \alpha(e^x - 1) \quad if\ x < 0$$

Figure 2.9 is the visual representation of the Exponential Linear Unit (ELU) Activation Function.

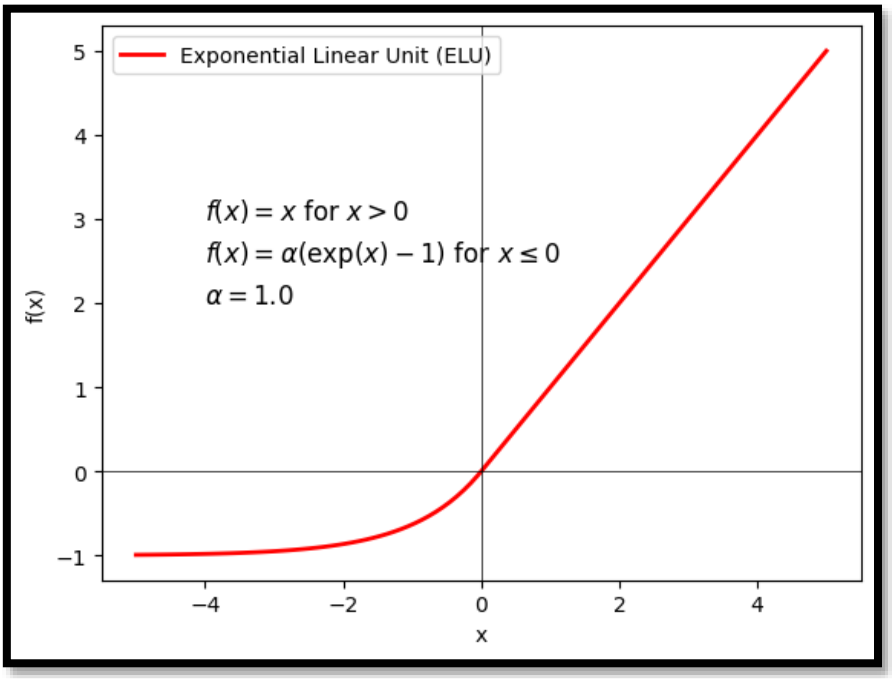


Figure 2.9: Exponential Linear Unit (ELU) Activation Function

ReLU can be effectively replaced with ELU due to the following advantages:

- ELU becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes.
- The log curve is introduced for negative input values to prevent the dead ReLU problem. It facilitates the network's adjustment of weights and biases.

The limitations of the ELU function are as follow:

- It increases the computational time because of the exponential operation included
- There is no learning of the "a" value.
- The problem of the exploding gradient

7. Swish

The Swish function is a relatively recent activation function developed by researchers at Google. Its unique characteristic is that it is not monotonic, meaning that the function's value can decrease even as the input values increase. In certain situations, the Swish function has been shown to outperform the ReLU function.

Mathematically it can be represented as:

$$f(x) = x * sigmoid(x)$$

where the sigmoid function defined as:

$$f(x) = 1/(1 + e^{-x})$$

Thus, the Swish function can also be expressed as:

$$f(x) = x/(1 + e^{-x})$$

Figure 2.10 is the visual representation of the Swish Activation Function.

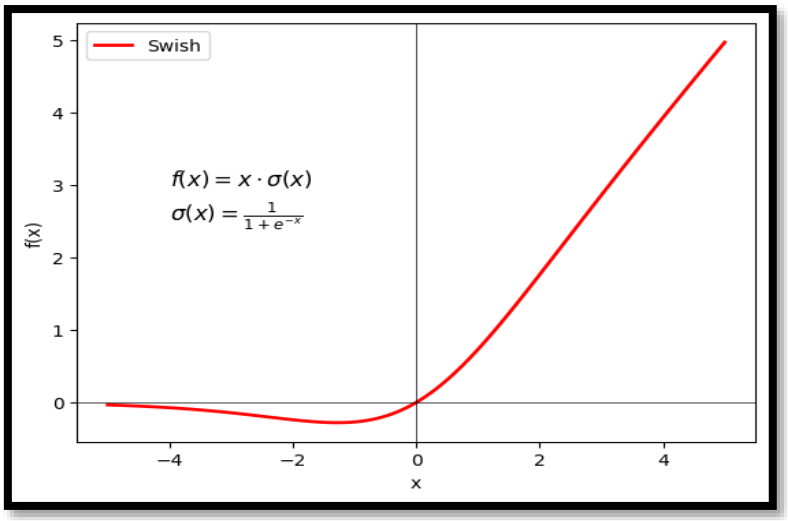


Figure 2.10: Swish Activation Function

Advantages of the Swish activation function over ReLU are as follows:

- Since Swish is a smooth function, it doesn't change direction suddenly like ReLU does when x = 0. Instead, it bends smoothly from 0 to values less than 0 and back up.
- The ReLU activation function was wiped out for small negative values. Nevertheless, those negative values could still be important for identifying underlying patterns in the data. It is a win-win situation because large negative numbers are zeroed out for sparsity reasons.
- The non-monotonous nature of the swish function improves the expression of input data and learning weight.

8. SoftMax

The Softmax function can be viewed as a generalization of multiple sigmoid functions. While a sigmoid function outputs values between 0 and 1, which can be interpreted as probabilities for a binary classification problem, the Softmax function extends this concept to multi-class classification.

In Softmax, each output can be treated as the probability of a particular class given a set of data points, ensuring that the sum of all output probabilities equals 1. This allows the model to provide a clear probabilistic interpretation of the likelihood of each class, making it suitable for tasks involving multiple classes.

Mathematically it can be represented as:

$$f(x_i) = e^{x_i} / \sum_{j=1}^{K} e^{x_j}$$

Figure 2.11 is the visual representation of the Softmax Activation Function.

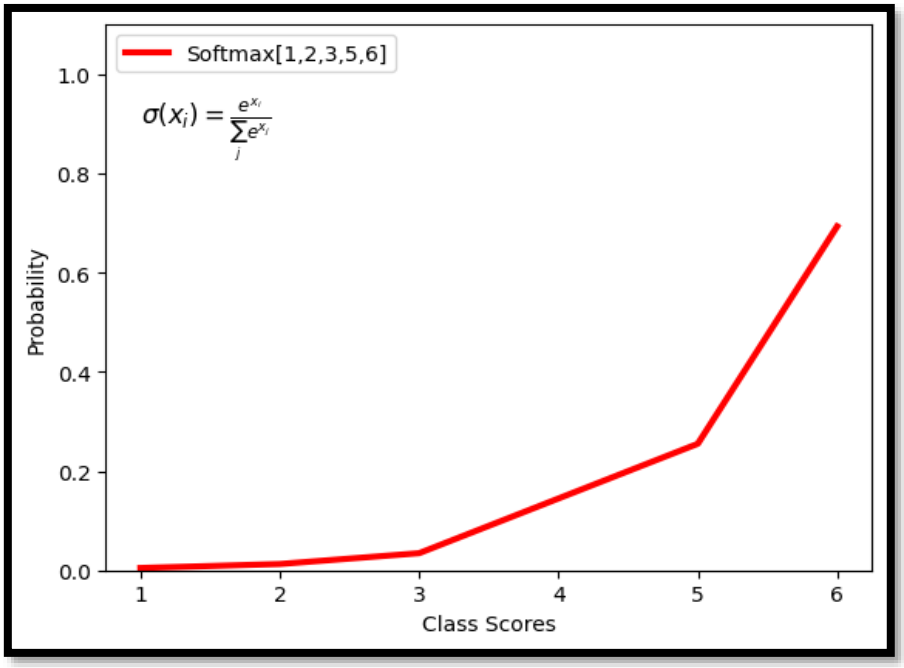


Figure 2.11: Softmax Activation Function

When implementing a neural network for multi-class classification, the output layer typically employs the Softmax function to determine class probabilities. The final layer will have a number of neurons equal to the number of target classes, and after passing through the network, the Softmax function is applied to the raw outputs to produce probabilities for each class. This output can then be used to make predictions, selecting the class with the highest probability as the model's prediction.

9. GeLU

Top NLP models like as BERT, ROBERTa, and ALBERT are compatible with the Gaussian Error Linear Unit (GELU) activation function. The combination of features from zoneout, dropout, and ReLUs motivates this activation function.

Together, ReLU and dropout produce the output of a neuron. ReLU does it stochastically by multiplying by zero and deterministically by multiplying the input by either zero or one, depending on whether the input value is positive or negative.

Zoneout is an RNN regularizer that randomly multiplies inputs by one.

We merge this functionality by multiplying the input by either zero or one which is stochastically determined and is dependent upon the input. We multiply the neuron input x by

m ~ Bernoulli(Φ(x)), where Φ(x) = P(X ≤x), X ~ N (0, 1) is the cumulative distribution function of the standard normal distribution.

This distribution is chosen since neuron inputs tend to follow a normal distribution, especially with Batch Normalization.

Mathematically it can be represented as: $f(x) = xP(X \leq x) = x\phi(x)$

$$= 0.5x(1 + tanh[\sqrt{2/\Pi}\ (x + 0.044715x^3)])$$

or

$$f(x) = 0.5x(1 + erf(x/\sqrt{2}\ )$$

where erf is the error function. Figure 2.12 is the visual representation of the GELU Activation Function.

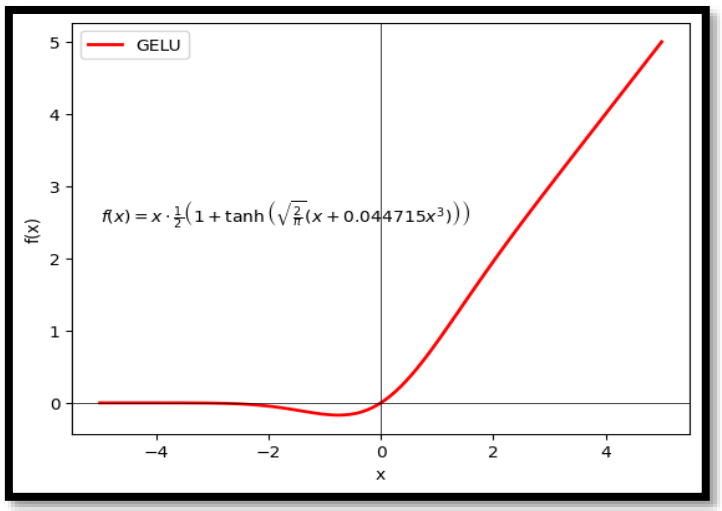


Figure 2.12: GELU Activation Function

In computer vision, natural language processing, and speech recognition, GELU nonlinearity outperforms ReLU and ELU activations and improves performance on all tasks.

10. SeLU

Scaled Exponential Linear Unit (SeLU), which was defined in self-normalizing networks, handles internal normalization, meaning that every layer maintains the variance and mean from the layers before it. This normalization is made possible by SeLU, which modifies the variance and mean.

SeLU has both positive and negative values to shift the mean, which was impossible for ReLU activation function as it cannot output negative values.
To modify the variance, gradients can be employed. To raise it, the activation function requires a region with a gradient greater than one.

Mathematically it can be represented as:

$$f(x) = \lambda\alpha(e^x - 1) \quad for\ x < 0$$
$$f(x) = \lambda x \quad for\ x \geq 0$$

SeLU has values of α and λ predefined. Figure 2.13 is the visual representation of the SeLU Activation Function.

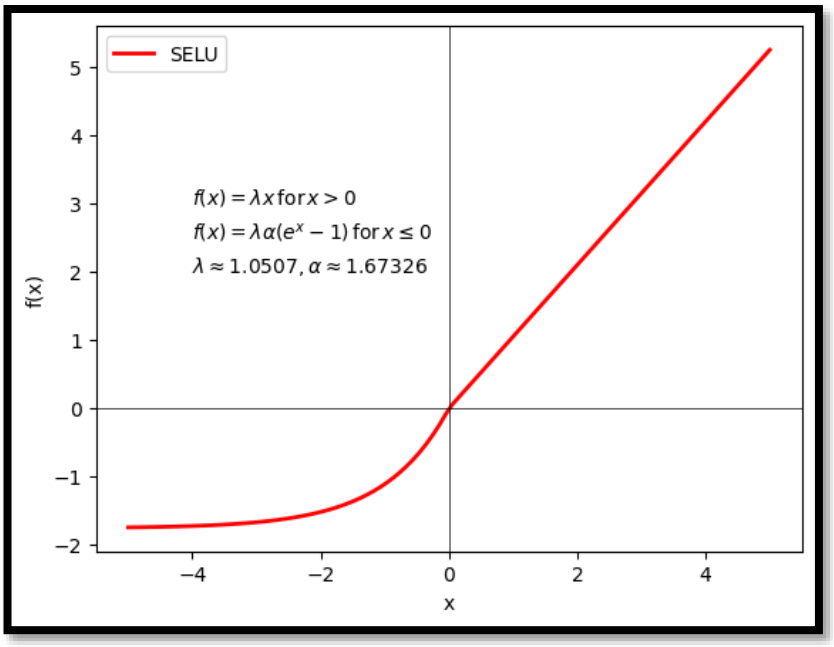


Figure 2.13: SELU Activation Function

SELU is a relatively newer activation function and needs more papers on architectures such as CNNs and RNNs, where it is comparatively explored.

Table 2.1 lists all the activation functions discussed along with their derivatives.

Activation Function	Equation	Derivative
Linear	$f(x) = x$	$f'(x) = 1$
Sigmoid	$f(x) = 1 / 1 + e^{-x}$	$f'(x) = f(x) * (1 - f(x))$
TanH	$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$	$f'(x) = 1 - f(x)^2$
ReLU (Rectified Linear Unit)	$f(x) = max(0, x)$	$f'(x) = 1 \quad if\ x \geq 0$ $f'(x) = 0 \quad if\ x < 0$
Leaky ReLU	$f(x) = x \quad if\ x \geq 0$ $f(x) = \alpha x \quad if\ x < 0$ where α is a small constant (e.g., 0.01)	$f'(x) = 1 \quad if\ x \geq 0$ $f'(x) = \alpha \quad if\ x < 0$
PReLU	$f(x) = max(ax, x)$ or $f(x) = x \quad if\ x \geq 0$ $f(x) = ax \quad if\ x < 0$ where a is a learnable parameter	$f'(x) = 1 \quad if\ x \geq 0$ $f'(x) = a \quad if\ x < 0$
Exponential Linear Unit (ELU)	$f(x) = x \quad if\ x \geq 0$ $f(x) = \alpha(e^{x} - 1) \quad if\ x < 0$ where α is a constant (e.g., 1)	$f'(x) = 1 \quad if\ x \geq 0$ $f'(x) = f(x) + \alpha \quad if\ x < 0$
Swish	$f(x) = x/(1 + e^{-x})$	$f'(x) = f(x) + \sigma(x) * (1 - f(x))$ where σ(x) is the Sigmoid function
SoftMax	$f(x_i) = e^{x_i} / \sum_{j=1}^{K} e^{x_j}$	$f'(x_i) = f(x_i) * (1 - f(x_i))$ for each output in multi-class probability distribution
Gaussian Error Linear Unit (GeLU)	$f(x) = 0.5x(1 + erf(x/\sqrt{2}))$	$f'(x) \approx 0.5(1 + erf(x/\sqrt{2}) + x e^{-x^2/2} / \sqrt{2\Pi})$
Scaled Exponential Linear Unit (SELU)	$f(x) = \lambda\alpha(e^{x} - 1)\ for\ x < 0$ $f(x) = \lambda x \quad for\ x \geq 0$	$f'(x) = f(x) + \lambda\alpha \quad for\ x < 0$ $f'(x) = \lambda \quad for\ x \geq 0$

Table 2.1: Activation Functions and their Derivatives

Table 2.2 gives the pros and cons of activation functions discussed.

Activation Function	Pros	Cons
Linear	Simple and useful for regression; direct proportionality between input and output.	Lacks non-linearity, so complex patterns can't be captured; may cause deep networks to act linearly.
Sigmoid	Maps input to a 0–1 range, making it useful in binary classification problems.	Prone to vanishing gradients, slowing training in deeper layers.
Tanh	Centered around zero, reducing bias in activation; stronger gradient than Sigmoid.	Still affected by vanishing gradients, slowing convergence for high values.
ReLU (Rectified Linear Unit)	Computationally efficient; mitigates vanishing gradient issue; faster convergence.	Can result in dead neurons (permanent zero output) when inputs are non-positive.
Leaky ReLU	Provides a small gradient for negative inputs, reducing the likelihood of dead neurons.	May result in instability if the negative slope (α) is too high; requires fine-tuning.
Parametric ReLU (PReLU)	Adaptive negative slope, learning the best parameter for each neuron.	Higher computational cost due to additional parameters; may risk overfitting.
Exponential Linear Unit (ELU)	Avoids dead neuron issue by allowing small negative values; helps gradients stay strong.	More computationally intensive; requires tuning of the α parameter.
Swish	Enhances learning in deeper models; combines the advantages of Sigmoid and ReLU functions.	Computationally heavier than ReLU; benefits may be minimal for shallow networks.
SoftMax	Converts outputs to probabilities for multi-class classification tasks.	Sensitive to outliers; gradients may saturate for large inputs, slowing optimization.
Gaussian Error Linear Unit (GeLU)	Used in advanced architectures; combines smoothness with probabilistic properties.	Complex to compute; may require specialized functions (e.g., error function) in implementation.
Scaled Exponential Linear Unit (SELU)	Enables self-normalization, preserving stable activations; beneficial for deep networks.	Requires specific parameters; performance may vary on non-scaled or non-centered data

Table 2.2: Pros and Cons of Activation Functions

2.5 CHOOSING RIGHT ACTIVATION FUNCTION

To achieve better performance and minimize errors in neural networks, several factors must be taken into account, such as the number of hidden layers, training techniques, hyperparameter tuning, and the selection of an appropriate activation function. Among these, the activation function is one of the most critical elements. Choosing the right activation function for a specific task can be a challenging process that often requires extensive research. There is no universal rule for selecting an activation function, as the choice is highly dependent on the task at hand. Different activation functions come with their own advantages and disadvantages, and the best one depends on the design of the system.

For example,

- sigmoid functions tend to work well in classification tasks.
- However, both sigmoid and tanh functions are avoided in certain cases due to the vanishing gradient problem, where gradients approach zero, slowing down the learning process. T
- The ReLU function is the most commonly used activation function and typically outperforms others in many scenarios.
- When facing issues like dead neurons in the network, leaky ReLU can be applied as an alternative.
- ReLU, however, should only be used in hidden layers and not in the output layer.

Studies have shown that both sigmoid and tanh functions are not ideal for hidden layers, as their slopes become negligible for large or small inputs, slowing down gradient descent. ReLU, with a derivative of 1, is often the preferred choice for hidden layers, and leaky ReLU can be used when zero derivatives are encountered.

Selecting an activation function that approximates the desired function efficiently and enables faster training is also essential for improving model performance.

Check Your Progress-2

a. The Sigmoid Activation function f(x) is defined as:
   a) $f(x) = 1/e^{-x} + e^{x}$
   b) $f(x) = x * e^{x}$
   c) $f(x) = 1 / 1 + e^{x}$
   d) $f(x) = 1 / 1 + e^{-x}$

b. Which activation function is commonly used in the hidden layers of a deep neural network?
   a) ReLU
   b) Sigmoid
   c) Tanh
   d) Softmax

c. Which of the following functions can be used as an activation function in the output layer if we wish to predict the probabilities of n classes (p1, p2..pk) such that sum of p over all n equals to 1?
   a) ReLU
   b) Sigmoid
   c) Tanh
   d) Softmax

d. Sigmoid and tanh activation functions cannot be with many layers due to the _____ problem.

e. _____ function overcomes the vanishing gradient problem, allowing models to learn faster and perform better

f. _____ Functions are most often used as the output of a classifier, to represent the probability distribution over n different classes

2.6 LET US SUM UP

In this unit we have discussed Activation Functions which are essential components in neural networks, transforming inputs to meaningful outputs for each layer. This process adds nonlinearity, making the network better at recognizing complex data patterns. Without it, the network would just act like a basic linear model and miss important details in data. We explored different types of activation functions—such as

Sigmoid, Tanh, and ReLU - each with its own properties. Choosing the right activation function is critical and often depends on the data and the network's design, making it important to experiment to find the best option for improving accuracy.

2.7 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a A function that determines the output of a neuron

1-b To introduce nonlinearity to the network, enabling it to learn complex patterns

1-c They must be differentiable

2-a $f(x) = 1 / 1 + e^{-x}$

2-b ReLU

2-c Softmax

2-d vanishing gradient problem

2-e ReLU

2-f Softmax

2.8 ASSIGNMENTS

- Explain what an activation function is and its purpose in a neural network.
- Why is nonlinearity important in neural networks? Describe what would happen if only linear functions were used.
- List three common activation functions. Describe each one, including when and why it is typically used in neural networks.
- What factors should be considered when selecting an activation function for a neural network?

Unit-3: Backpropagation and Optimization

3

Unit Structure

3.0. Learning Objectives
3.1. Introduction
3.2. Backpropagation : Understanding the Process,
3.3. Optimizer
3.4. A Model-Optimization Algorithm
3.5. Gradient Descent
3.6. Stochastic Gradient Descent (SGD)
3.7. Mini-batch Gradient Descent
3.8. AdaGrad (Adaptive Gradient Algorithm)
3.9. RMSprop (Root Mean Square Propagation)
3.10. Adam (Adaptive Moment Estimation)
3.11. Let us sum up
3.12. Assignment
3.13. Solution Check your Progress

3.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the Backpropagation Process.
- Derive and analyze the mathematical equations governing backpropagation.
- Identify Gradient Descent Variants.
- Compare and evaluate various optimization techniques, such as SGD, RMSProp, Adam, and Adagrad, based on their effectiveness and efficiency.

3.1 INTRODUCTION

An effective technique for training an ANN was found in 1986. This approach propagates errors; that is, the discrepancy between the output layer's values and the expected values backward from the output layer to the layers that came before it. Backpropagation, or propagating the errors backward to the preceding layers, is the name given to the algorithm that uses this technique.

The backpropagation technique can be used in feed forward networks with multiple layers. The goal of this supervised learning approach is to minimize the output signal's deviation from the desired output by continuously modifying the weights of the connected neurons. There are several iterations, or epochs, in this process. Every epoch has two distinct phases forward and backward:

- In the **forward phase**, signals move through the hidden layers from the input layer's neurons to the output layer's neurons. Throughout the flow, the activation functions and connectivity weights are utilized. The output signals are produced in the output layer.
- In the **backward phase** the expected value and the output signal are compared. From the output to the layers before it, the calculated errors are propagated backwards. The connectivity weights between the layers are modified using the propagated back faults.

3.2 BACKPROPAGATION : UNDERSTANDING THE PROCESS

The figure 3.1 shows the workflows of the backpropagation process which includes various computation activities such as:

- **Find Error rate:** Here we need to calculate the model output with actual output
- **Minimum Error:** Cross verifying whether the error is minimized or not.
- **Update the Weights:** The error is more than the acceptable range then, update the weights and biases. After that, again check the error. Repeat the process until the error becomes low.
- **Neural Network Model:** Once the error rate was acceptable range then the model is ready to use for forecasting the data

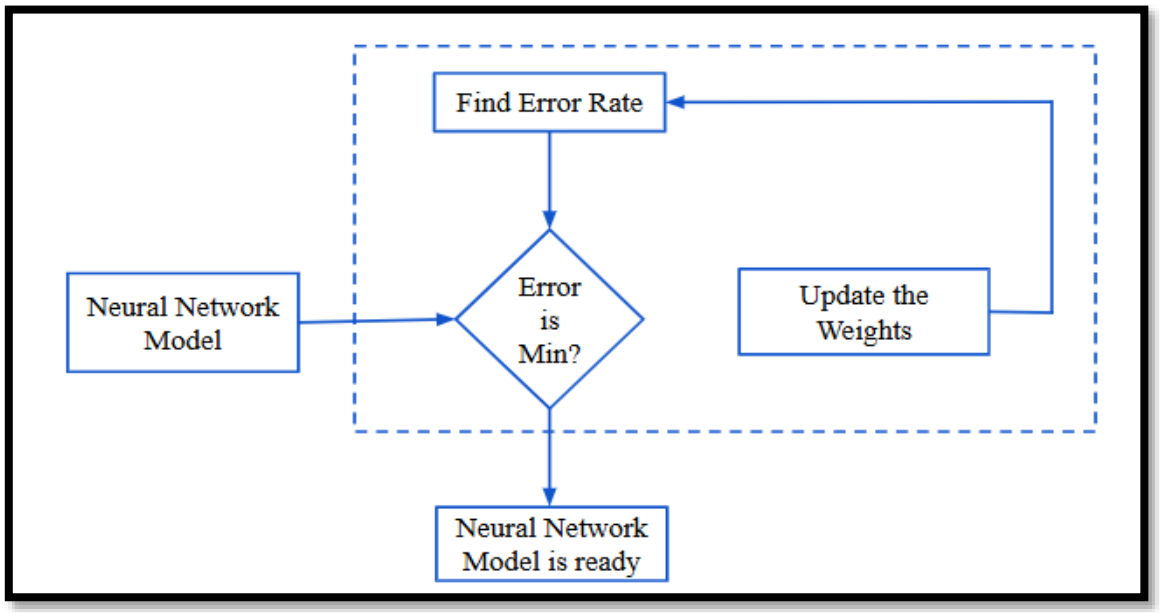


Figure 3.1:Process of backpropagation

Working of Backpropagation Algorithm

The Forward Pass and the Backward Pass are the two primary steps of the Backpropagation algorithm. The forward pass involves feeding the input layer with the data. Figure 3.2 shows the general working of backpropagation algorithm.

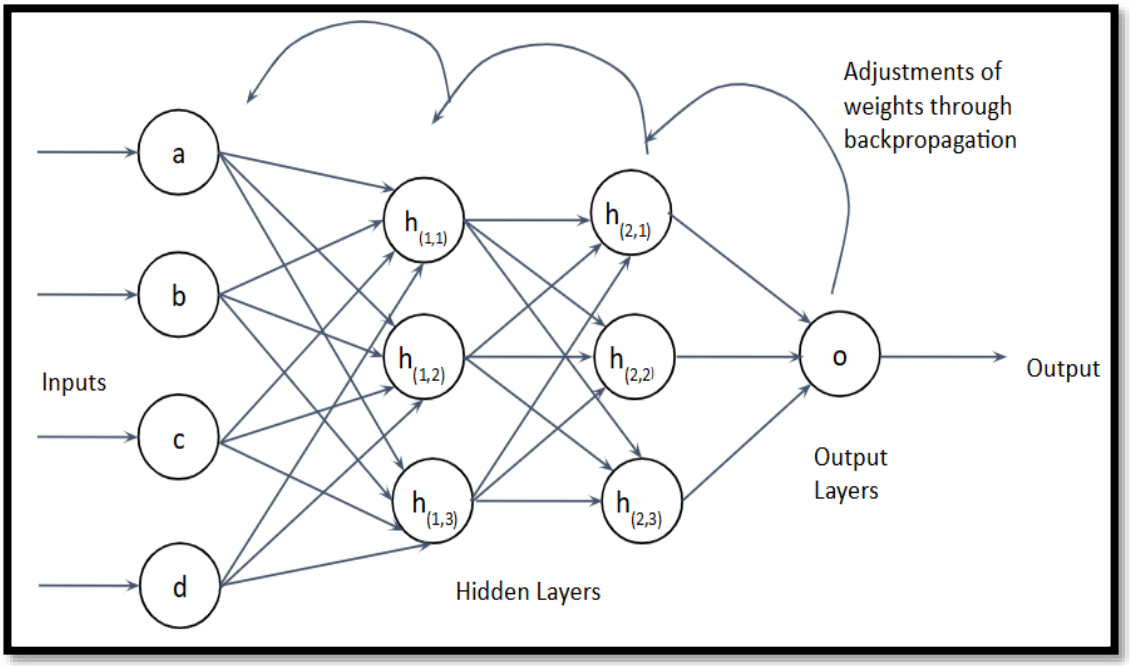


Figure 3.2: Working of Backpropagation Algorithm

It passes these inputs to hidden layers along with their corresponding weights. The output of h1 is used as the input for h2 in a network with two hidden layers (h1 and h2, as seen in figure 3.2). Before applying an activation function, a bias is added to the weighted inputs.

An activation function such as ReLU (Rectified Linear Unit), which returns the input if it is positive and 0 otherwise, is applied by each hidden layer. By adding non-linearity, the model is able to discover intricate linkages within the data. The output layer receives the outputs from the final hidden layer and uses an activation function, like softmax, to transform the weighted outputs into classification probabilities.

The backward pass modifies the weights and biases in the network by propagating the error, the discrepancy between the expected and actual output backwards. The Mean Squared Error (MSE), which is calculated using the formula as follows, it is a popular technique for calculating errors:

$$\mathbf{MSE} = (\mathbf{Predicted\ Output} - \mathbf{Actual\ Output})^2$$

After calculating the error, the network uses gradients which are determined by the chain rule to modify weights. These gradients show the amount that each bias and

weight should be changed in order to reduce the error in the subsequent iteration. In order to make sure the network learns and gets better at what it does, the backward pass keeps going layer by layer. During backpropagation, the activation function is essential in calculating these gradients through its derivative. Figure 3.3 shows an example of a neural network.

Example:

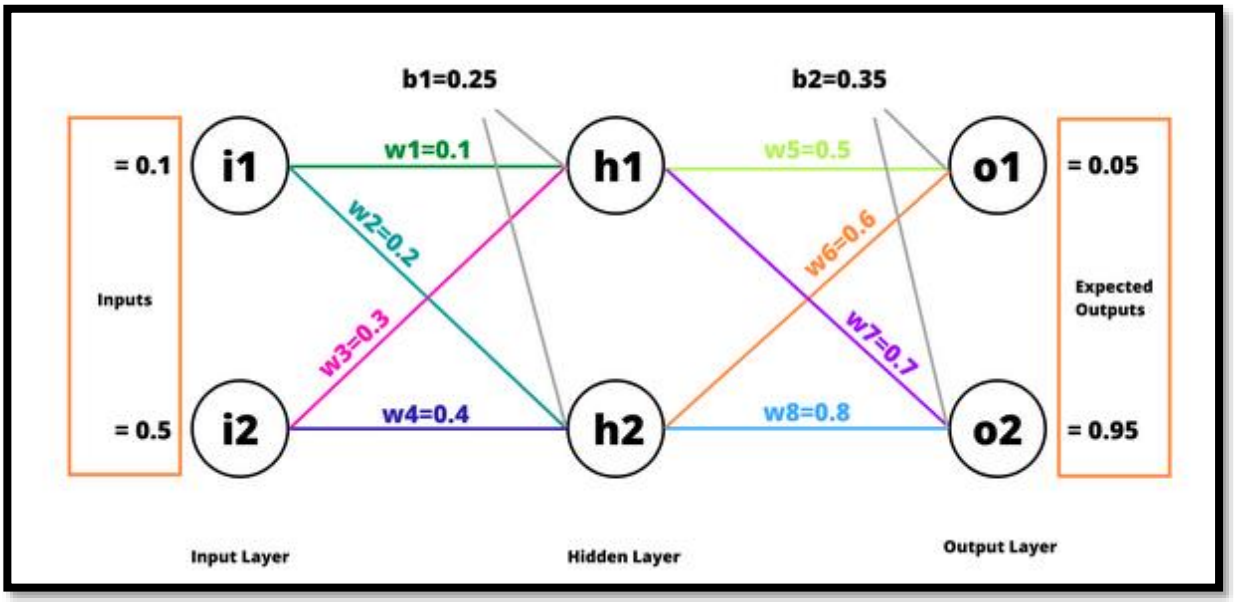


Figure 3.3: Example of A Neural Network

Above example of a neural network is self-explanatory. There are two units in the Input Layer, two units in the Hidden Layer and two units in the Output Layer. The w1, w2, w2, …, w8 represent the respective weights. The b1 and b2 are the biases for Hidden Layer and Output Layer, respectively.

We'll be passing two inputs i1 and i2, and perform a forward pass to compute total error and then a backward pass to distribute the error inside the network and update weights accordingly.

Before getting started, let us deal with two basic concepts which should be sufficient to comprehend this point.

Peeking inside a single neuron as shown in figure 3.4

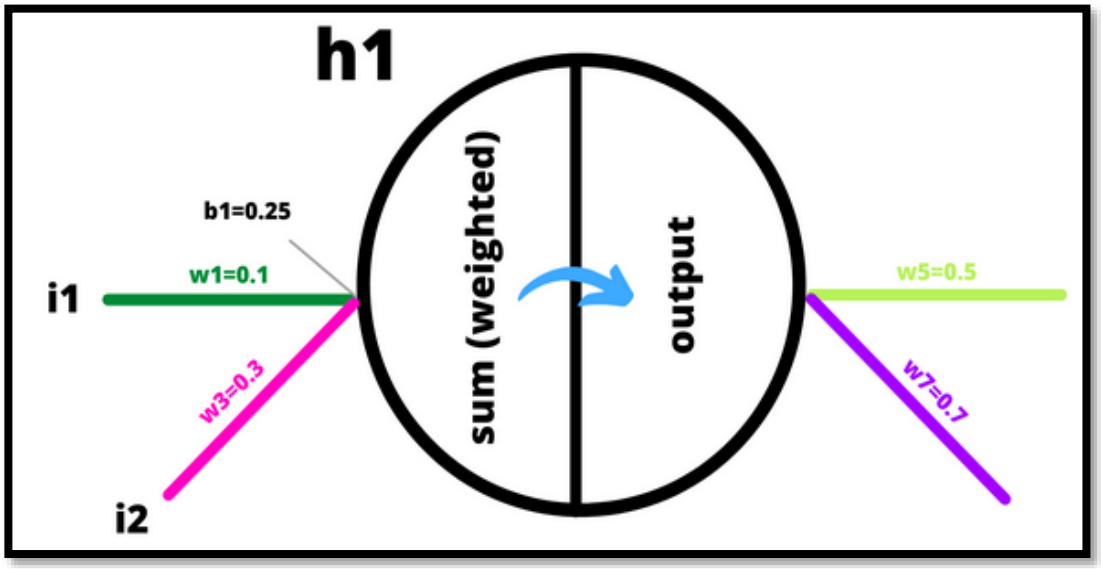


Figure 3.4: Inside h1 (first unit of the hidden layer)

Two actions take place within a unit: (i) the weighted sum is computed, and (ii) an activation function is used to squash the weighted sum. Until the subsequent layer is an output layer, the activation function's result serves as an input. The Sigmoid function, often known as the logistic function, is used as the activation function. In essence, the Sigmoid function restricts an input value between 0 and +1. However, the two actions mentioned above take place inside a neural network unit. Assuming that the input layer's linear function yields the same value as the input, we can proceed.

Chain Rule in Calculus

If we have functions $y = f(u)$ and $u = g(x)$ then we can write the derivative of y as:

$$\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx}$$

The Forward Pass

Each unit of a neural network performs two operations: compute weighted sum and process the sum through an activation function. The outcome of the activation function determines if that particular unit should activate or become insignificant.

Let's get started with the forward pass, the values of all the parameters is as shown in figure 3.4.

For h1,

$$sum_{h1} = i1 * w1 + i2 * w3 + b1$$

$$sum_{h1} = 0.1 * 0.1 + 0.5 * 0.3 + 0.25 = 0.41$$

Now pass this weighted sum through the logistic function (sigmoid function) so as to squash the weighted sum into the range (0 and +1). The logistic function is an activation function for our example neural network.

$$output_{h1} = \frac{1}{1 + e^{-sum_{h1}}}$$

$$output_{h1} = \frac{1}{1 + e^{-0.41}} = 0.60108$$

Similarly, for h2, we perform the weighted sum operation sum_{h2} and compute the activation value outputh2outputh2.

$$sum_{h2} = i1 * w2 + i2 * w4 + b1$$

$$sum_{h2} = 0.1 * 0.2 + 0.5 * 0.4 + 0.25 = 0.47$$

$$output_{h2} = \frac{1}{1 + e^{-0.47}} = 0.61538$$

Now, $output_{h1}$ and $output_{h2}$ will be considered as inputs to the next layer.

For o1,

$$sum_{o1} = output_{h1} * w5 + output_{h2} * w6 + b2 = 1.01977$$

$$output_{o1} = \frac{1}{1 + e^{-sum_{o1}}} = 0.73492$$

For o2,

$$sum_{o2} = output_{h1} * w7 + output_{h2} * w8 + b2 = 1.26306$$

$$output_{o2} = \frac{1}{1 + e^{-sum_{o2}}} = 0.77955$$

Computing the total error

We started off supposing the expected outputs to be 0.05 and 0.95 respectively for $output_{o1}$ and $output_{o2}$ Now we will compute the errors based on the outputs received until now and the expected outputs.

Use the following error formula,

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

To compute E_{total} , we need to first find out the respective errors at o1 and o2.

$$E_1 = \sum \frac{1}{2}(target_1 - output_{o1})^2$$

$$E_1 = \sum \frac{1}{2}(0.05 - 0.73492)^2 = 0.23456$$

Similarly, for E_2,

$$E_2 = \sum \frac{1}{2}(target_2 - output_{o2})^2$$

$$E_1 = \sum \frac{1}{2}(0.95 - 0.77955)^2 = 0.01452$$

Therefore,

$$E_{total} = E_1 + E_2 = 0.24908$$

The Backpropagation

The aim of backpropagation (backward pass) is to distribute the total error back to the network so as to update the weights in order to minimize the cost function (loss). The weights are updated in such a way that when the next forward pass utilizes the updated weights, the total error will be reduced by a certain margin (until the minima is reached).

For weights in the output layer (w5, w6, w7, w8)

For w5,

Let's compute how much contribution w5 has on E1. If we become clear on how w5 is updated, then it would be really easy for us to generalize the same to the rest of the weights. If we look closely at the example neural network, we can see that E1 is affected by $output_{o1}$,$output_{o1}$ is affected by sum_{o1}, and sum_{o1} is affected by w5. It's time to recall the Chain Rule.

$$\frac{dE_{total}}{dw5} = \frac{dE_{total}}{doutput_{o1}} * \frac{doutput_{o1}}{dsum_{o1}} * \frac{dsum_{o1}}{dw5}$$

Let's deal with each component of the above chain separately.

Component 1: partial derivative of Error w.r.t. Output

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$E_{total} = \frac{1}{2}(target_1 - output_{o1})^2 + \frac{1}{2}(target_2 - output_{o2})^2$$

Therefore,

$$\frac{dE_{total}}{doutput_{o1}} = 2 * \frac{1}{2} * (target_1 - output_{o1}) \quad * -1 = output_{o1} - target_1$$

Component 2: partial derivative of Output w.r.t. Sum

The output section of a unit of a neural network uses non-linear activation functions. The activation function used in this example is Logistic Function. When we compute the derivative of the Logistic Function, we get:

$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-x}}$$

$$\frac{\boldsymbol{d}}{\boldsymbol{dx}}\boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})(\mathbf{1} - \boldsymbol{\sigma}(\boldsymbol{x}))$$

Therefore, the derivative of the Logistic function is equal to output multiplied by (1 – output).

$$\frac{\boldsymbol{doutput_{o1}}}{\boldsymbol{dsum_{o1}}} = \boldsymbol{output_{o1}}(\mathbf{1} - \boldsymbol{output_{o1}})$$

Component 3: partial derivative of Sum w.r.t. Weight

$$sum_{o1} = output_{h1} * w5 + output_{h2} * w6 + b2$$

Therefore,

$$\frac{\boldsymbol{dsum_{o1}}}{\boldsymbol{dw5}} = \boldsymbol{output_{h1}}$$

Putting them together,

$$\frac{dE_{total}}{dw5} = \frac{dE_{total}}{doutput_{o1}} * \frac{doutput_{o1}}{dsum_{o1}} * \frac{dsum_{o1}}{dw5}$$

$$\frac{dE_{total}}{dw5} = [\ output_{o1} - target_1\ ] * [\boldsymbol{output_{o1}(1 - output_{o1})}][\boldsymbol{output_{h1}}]\frac{dE_{total}}{dw5}$$

$$= 0.68492 * 0.19480 * 0.60108$$

$\frac{dE_{total}}{dw5}$=0.08020

The new_w5 is,

$new_{w5} = w5 - n * \frac{dE_{total}}{dw5}$, where n is learning rate

$$new_{w5} = 0.5 - 0.6 * 0.08020$$

$$new_{w5} = 0.45187$$

We can proceed similarly for w6,w7,w8.

For w6,

$$\frac{dE_{total}}{dw6} = \frac{dE_{total}}{doutput_{o1}} * \frac{doutput_{o1}}{dsum_{o1}} * \frac{dsum_{o1}}{dw6}$$

The first two components of this chain have already been calculated. The last component $\frac{dE_{total}}{dw5} = output_{h2}$

$\frac{dE_{total}}{dw6}$ =0.68492*0.19480*0.61538=0.08211

The new_w6 is,

$new_{w6} = w6 - n * \frac{dE_{total}}{dw6}$, where n is learning rate

$$new_{w6} = 0.6 - 0.6 * 0.08211$$

$$new_{w6} = 0.55073$$

For w7,

$$\frac{dE_{total}}{dw7} = \frac{dE_{total}}{doutput_{o2}} * \frac{doutput_{o2}}{dsum_{o2}} * \frac{dsum_{o2}}{dw7}$$

The first component of the above chain Let's recall how the partial derivative of Error is computed w.r.t. Output.

$$\frac{dE_{total}}{doutput_{o2}} = output_{o2} - target_2$$

For the second component,

$$\frac{doutput_{o2}}{dsum_{o2}} = output_{o2}(1 - output_{o2})$$

For the third component,

$$\frac{dsum_{o2}}{dw7} = output_{h1}$$

Putting them together,

$\frac{dE_{total}}{dw7} = -0.17044 * 0.17184 * 0.60108$=-0.01760

The new_w7 is,

$$new_{w7} = w7 - n * \frac{dE_{total}}{dw7}$$

$$new_{w7} = 0.7 - 0.6 * -0.01760$$

$$new_{w7} = 0.71056$$

Proceeding similarly, we get new_w8=0.81081(with $\frac{dE_{total}}{dw8} = -0.01802$

For weights in the hidden layer (w1, w2, w3, w4)

Similar calculations are made to update the weights in the hidden layer. However, this time the chain becomes a bit longer. It does not matter how deep the neural network goes, all we need to find out is how much error is propagated (contributed) by a particular weight to the total error of the network. For that purpose, we need to find the partial derivative of Error w.r.t. to the particular weight. Let's work on updating w1 and we'll be able to generalize similar calculations to update the rest of the weights.

For w1 (with respect to E1),

For simplicity let us compute $\frac{dE_1}{dw_1}$ and $\frac{dE_2}{dw_1}$ separately, and later we can add them to compute $\frac{dE_{total}}{dw_1}$.

$$\frac{dE_1}{dw_1} = \frac{dE_1}{doutput_{o1}} = \frac{doutput_{o1}}{dsum_{o1}} * \frac{dsum_{o1}}{doutput_{h1}} * \frac{doutput_{h1}}{dsum_{h1}} * \frac{dsum_{h1}}{dw1}$$

Let's quickly go through the above chain. We know that E_1 is affected by $output_{o1}$, $output_{o1}$ is affected by sum_{o1}, sum_{o1} is affected by $output_{h1}$, $output_{h1}$ is affected by sum_{h1}, and finally sum_{h1} is affected by w1. It is quite easy to comprehend.

For the first component of the above chain,

$$\frac{dE_1}{doutput_{o1}} = output_{o1} - target_1$$

For the third component,

$$sum_{o1} = output_{h1} * w5 + output_{h1} * w6 + b2$$

$$\frac{dsum_{o1}}{doutput_{h1}} = w5$$

For the fourth component,

$$\frac{doutput_{h1}}{dsum_{h1}} = output_{h1} * (1 - output_{h1})$$

for the fifth component,

$$sum_{h1} = i_1 * w1 + i_2 * w3 + b1$$

$$\frac{dsum_{h1}}{dw1} = i_1$$

Putting them all together,

$$\frac{dE_1}{dw_1} = \frac{dE_1}{doutput_{o1}} = \frac{doutput_{o1}}{dsum_{o1}} * \frac{dsum_{o1}}{doutput_{h1}} * \frac{doutput_{h1}}{dsum_{h1}} * \frac{dsum_{h1}}{dw1}$$

$$\frac{dE_1}{dw_1} = 0.68492 * 0.19480 * 05 * 0.23978 * 0.1 = 0.00159$$

similarly, for w1(with respect to E2),

$$\frac{dE_2}{dw_1} = \frac{dE_2}{doutput_{o21}} = \frac{doutput_{o2}}{dsum_{o2}} * \frac{dsum_{o2}}{doutput_{h1}} * \frac{doutput_{h1}}{dsum_{h1}} * \frac{dsum_{h1}}{dw1}$$

$$\frac{dE_2}{dw_1} = -0.17044 * 0.17184 * 0.7 * 0.23978 * 0.1 = -0.00049$$

Now we can compute

$$\frac{dE_{total}}{dw_1} = \frac{dE_1}{dw_1} + \frac{dE_2}{dw_1} = 0.00159 + (-0.00049) = 0.00110$$

The new_w1 is,

$$new_{w1} = w1 - n * \frac{dE_{total}}{dw_1}$$

$$new_{w1} = 0.1 - 0.6 * 0.00110 = 0.09933$$

Proceeding similarly, we can easily update the other weights (w2, w3 and w4).

$$new_{w2} = 0.19919$$

$$new_{w3} = 0.29667$$

$$new_{w4} = 0.39597$$

Once we've computed all the new weights, we need to update all the old weights with these new weights. Once the weights are updated, one backpropagation cycle is finished. Now the forward pass is done and the total new error is computed. And based on this newly computed total error the weights are again updated. This goes on until the loss value converges to minima. This way a neural network starts with random values for its weights and finally converges to optimum values.

Check your progress-1

a) Backpropagation is used to minimize the _____ function in a neural network.
b) The _____ pass of backpropagation calculates the gradients of the loss function with respect to the network parameters.
c) The derivative of the _____ function is used during backpropagation to compute gradients.
d) The _____ rate determines the size of weight updates during training.
e) Backpropagation calculates the gradients for all layers of the network using the chain rule. (True/False)
f) The vanishing gradient problem is more common in deep networks with sigmoid or tanh activations. (True/False)

3.3 OPTIMIZER

In deep learning, optimizers are crucial as algorithms that dynamically fine-tune a model's parameters throughout the training process, aiming to minimize a predefined loss function. These specialized algorithms facilitate the learning process of neural networks by iteratively refining the weights and biases based on the feedback received from the data. Well-known optimizers in deep learning encompass Gradient Descent, Stochastic Gradient Descent (SGD), Adam, and RMSprop, each equipped with distinct update rules, learning rates, and momentum strategies, all geared towards the overarching goal of discovering and converging upon optimal model parameters, thereby enhancing overall performance.

3.4 A MODEL-OPTIMIZATION ALGORITHM

A deep learning model comprises multiple layers of interconnected neurons organized into layers. After calculating an activation function from the incoming data, each neuron sends the result to the layer below. Complex mappings between inputs and outputs are made possible by the non-linearity introduced by the activation functions. Weights and biases parametrize the strength of the link between neurons and their activations. To reduce the difference between the model's output and the intended output provided by the training data, these parameters are iteratively changed throughout training. A loss function measures the difference. Figure 3.5 shows the mechanism of model optimization.

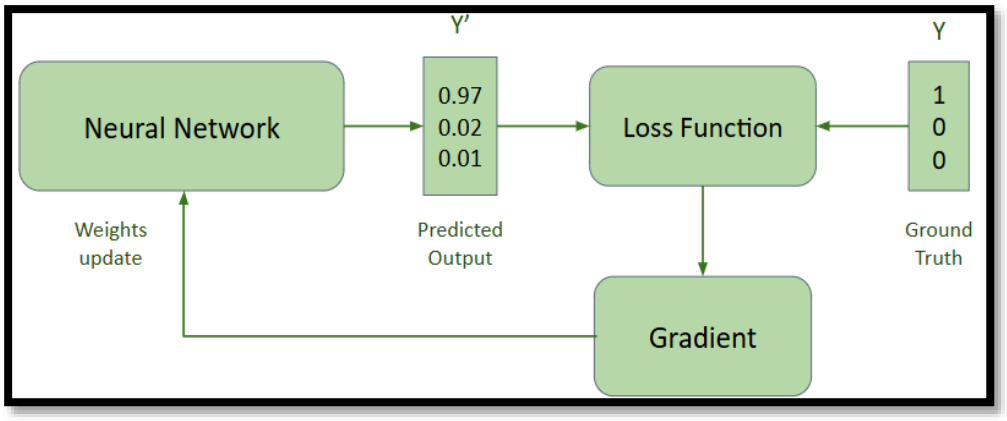


Figure 3.5: Model Optimization

There is an optimization algorithm that controls this modification. In order to efficiently traverse the model's high-dimensional parameter space, optimizers use gradients calculated via backpropagation to identify the direction and magnitude of parameter changes. In order to avoid local minima and converge to optimal or nearly optimal solutions, optimizers use a variety of tactics to strike a balance between exploration and exploitation.

Any data scientist training deep learning models must be aware of the advantages and disadvantages of various optimization strategies. To get the greatest training outcomes in the shortest period of time, it is crucial to choose the appropriate optimizer for the task.

Maxima and Minima

Maxima is the largest and Minima is the smallest value of a function within a given range. The maxima and minima is represented as shown in figure 3.6:

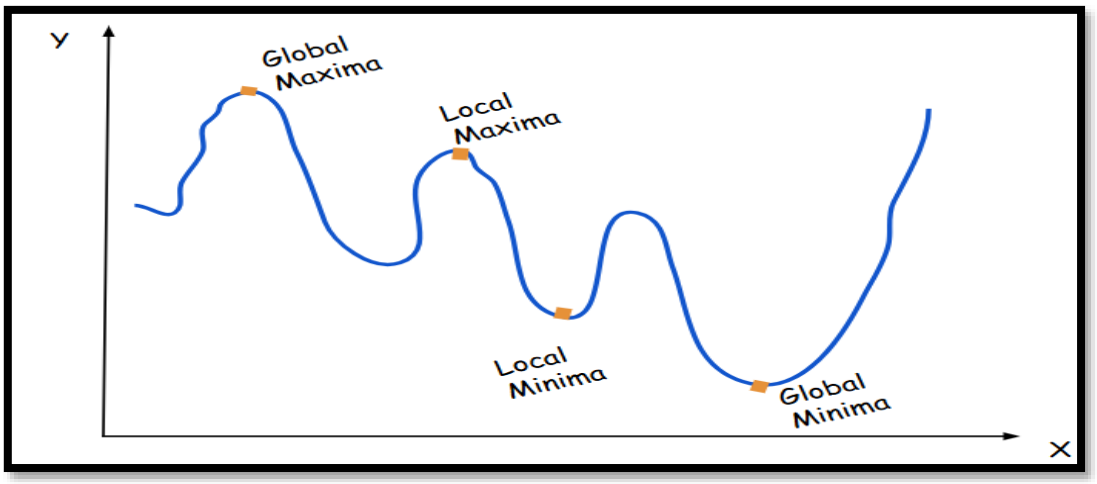


Figure 3.6: Maxima and Minima

- Global Maxima and Minima: It is the maximum value and minimum value respectively on the entire domain of the function
- Local Maxima and Minima: It is the maximum value and minimum value respectively of the function within a given range

There can be only one global minima and maxima but there can be more than one local minima and maxima.

3.5 GRADIENT DESCENT

Gradient Descent is an algorithm designed to minimize a function by iteratively moving towards the minimum value of the function. It's akin to a hiker trying to find the lowest point in a valley shrouded in fog. The hiker starts at a random location and can only feel the slope of the ground beneath their feet. To reach the valley's lowest point, the hiker takes steps in the direction of the steepest descent.

All deep learning model optimization algorithms widely used today are based on Gradient Descent. Hence, having a good grasp of the technical and mathematical details is essential. So, let's take a look:

- **Objective:** Gradient Descent aims to find a function's parameters (weights) that minimize the cost function. In the case of a deep learning model, the cost function is the average of the loss for all training samples as given by the loss function. While the loss function is a function of the model's output and the ground truth, the cost function is a function of the model's weights and biases. Figure 3.7 shows the graphical representation of gradient descent.

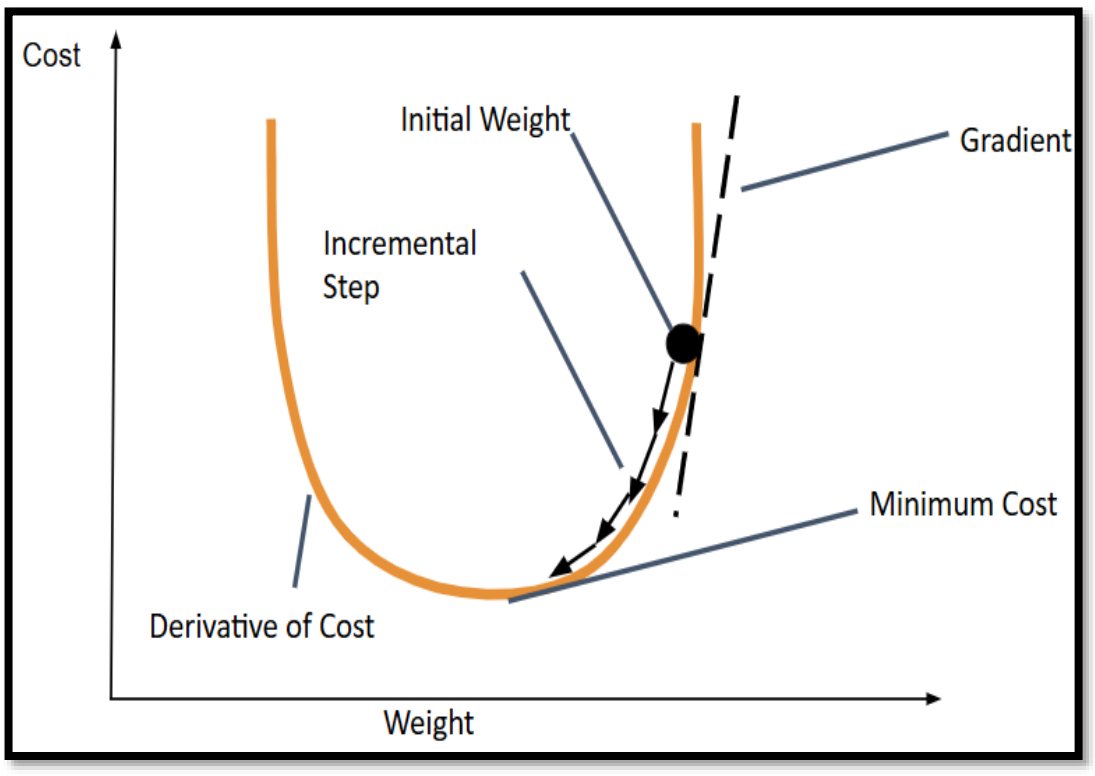


Figure 3.7: Gradient Descent

- **How it works:**
    - **Initialization:** Start with random values for the model's weights.
    - **Gradient computation:** Calculate the gradient of the cost function with respect to each parameter. The gradient is a vector that points in the direction of the steepest increase of the function. In the context of optimization, we're interested in the negative gradient, which points towards the direction of the steepest decrease.
    - **Update parameters**: Adjust the model's parameters in the direction opposite to the gradient. This step is done by subtracting a fraction of the gradient from the current values of the parameters. The size of this step is determined by the learning rate, a hyperparameter that controls how fast or slow we move toward the optimal weights.
- **Mathematical representation:** the update rule for each parameter $\boldsymbol{w}$ can be mathematically represented as:

$$_{\omega} := \omega - \alpha \nabla_{\omega} J(\omega)$$

where w represents the model's parameters (weights) and α is the learning rate. ΔwJ(w) is the gradient of the cost function J(w) with respect to w.

The learning rate is a crucial hyperparameter that needs to be chosen carefully. If it's too small, the algorithm will converge very slowly. If it's too large, the algorithm might overshoot the minimum and fail to converge.

Challenges:

- **Local minima and saddle points:** In complex cost landscapes, Gradient Descent can get stuck in local minima or saddle points, especially in non-convex optimization problems common in deep learning.
- **Choosing the right learning rate**: Finding an optimal learning rate requires experimentation and tuning.

Figure 3.8 shows the convergence and divergence scenario of gradient descent.

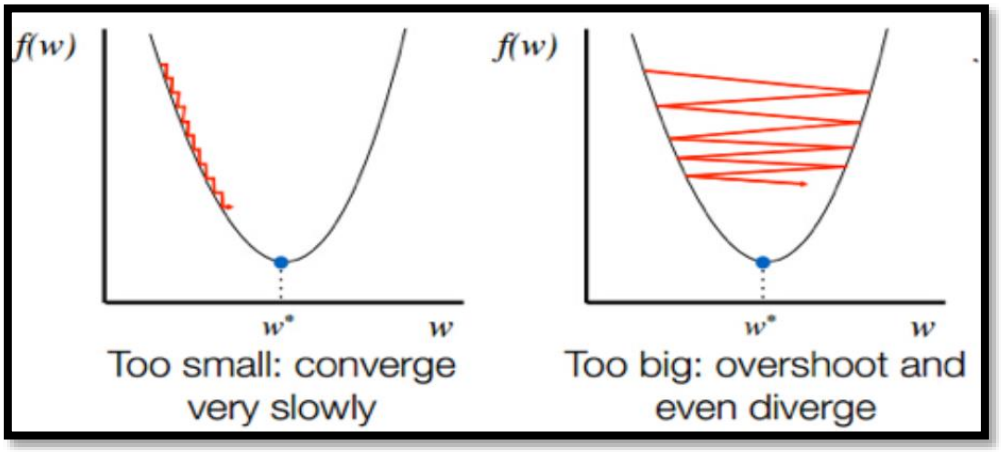


Figure: 3.8: Convergence and Divergence Scenario

3.6 STOCHASTIC GRADIENT DESCENT (SGD)

Stochastic Gradient Descent (SGD) is a variant of the traditional Gradient Descent optimization algorithm that introduces randomness into the optimization process to improve convergence speed and potentially escape local minima. To understand the intuition behind SGD, we can again invoke the analogy of a hiker descending a foggy valley. If Gradient Descent represents a cautious hiker who carefully evaluates the slope around them before taking a step, Stochastic Gradient Descent is akin to a more impulsive hiker who decides their next step based only on the slope of the ground immediately beneath their feet.

This approach can lead to a quicker descent but might involve more meandering. Let's take a closer look at the specifics of Stochastic Gradient Descent:

- **Objective**: like Gradient Descent, the primary goal of SGD is to minimize the cost function of a model by iteratively adjusting its parameters (weights). However, SGD aims to achieve this goal more efficiently by using only a single training example at a time to inform the update of the model's parameters.
- **How it works:**
    - **Initialization:** Start with a random set of parameters for the model.
    - **Gradient computation**: Instead of calculating the gradient of the cost function over the entire training data, SGD computes the gradient based on a single randomly selected training example.

    - **Update parameters:** Update the model's parameters using this computed gradient. The parameters are adjusted in the direction opposite to the gradient, similar to basic Gradient Descent.
- **Mathematical representation:** The parameter update rule in SGD is similar to that of Gradient Descent but applies to a single example i:

$$\omega := \omega - \alpha \nabla_{\omega} J_i(\omega)$$

Here, w represents the model's parameters (weights), α is the learning rate, and $\Delta wJi(w)$ is the gradient of the cost function Ji(w) for the i^{th} training example with respect to w.

Challenges:

- **Variance:** The updates can be noisy due to the reliance on a single example, potentially causing the cost function to fluctuate. As a result, the algorithm does not converge to a minimum but jumps around the cost landscape.
- **Hyperparameter tuning:** Correctly setting the learning rate requires experimentation.

Advantages:

- **Efficiency:** Using only one example at a time, SGD significantly reduces the computational requirements, making it faster and more scalable than Gradient Descent.
- **Escape local minima:** The inherent noise in SGD can help the algorithm escape shallow local minima, potentially leading to better solutions in complex cost landscapes.
- **Online learning:** SGD is well-suited for online learning scenarios where the model needs to update continuously as new data arrives.

3.7 MINI-BATCH GRADIENT DESCENT

Mini-batch Gradient Descent strikes a balance between the thorough, calculated approach of Gradient Descent and the unpredictable, swift nature of Stochastic Gradient Descent (SGD). Imagine a group of hikers navigating through a foggy valley.

Each hiker independently assesses a small, distinct section of the surrounding area before the group decides on the best direction to take.

Based on a broader but still limited view of the terrain, this collective decision-making process allows for a more informed and steady progression toward the valley's lowest point compared to an individual hiker's erratic journey. Here's a deep dive into Mini-batch Gradient Descent:

- **Objective:** Similar to other gradient descent variants, the aim of Mini-batch Gradient Descent is to optimize the model's parameters to minimize the cost function. It seeks to combine the efficiency of SGD with the stability of Gradient Descent by using a subset of the training data to compute gradients and update parameters.
- **How it works**:
  - **Initialization**: Start with initial random values for the model's parameters.
  - **Gradient computation**: Instead of calculating the gradient using the entire dataset (as in Gradient Descent) or a single example (as in SGD), Mini-batch Gradient Descent computes the gradient using a small subset of the training data, known as a mini-batch.
  - **Update parameters**: Adjust the parameters in the direction opposite to the computed gradient. This adjustment is made based on the gradient derived from the mini-batch, aiming to reduce the cost function.
- **Mathematical representation:** The parameter update rule for Mini-batch Gradient Descent can be represented as

$$_{\omega} := \omega - \alpha \nabla_{\omega} J_{mini-batch}(\omega)$$

  where w represents the model's parameters (weights), α is the learning rate, and $\Delta wJi(w)$ is the gradient of the cost function $Jmini\text{-}batch(w)$ for the current mini-batch of training samples with respect to w.

Challenges:

- **Hyperparameter tuning:** Like with the other variants we've discussed so far, selecting the learning rate requires experimentation. Further, we need to choose the batch size. If the batch size is too small, we face the drawbacks of SGD, and if the batch size is too large, we're prone to the issues of basic Gradient Descent.

Advantages:

- **Efficiency and stability:** Mini-batch Gradient Descent offers a compromise between the computational efficiency of SGD and the stability of Gradient Descent.
- **Parallelization:** Since mini-batches only contain a small, fixed number of samples, they can be computed in parallel, speeding up the training process.
- **Generalization:** By not using the entire dataset for each update, Mini-batch Gradient Descent can help prevent overfitting, leading to models that generalize better on unseen data.

3.8 ADAGRAD (ADAPTIVE GRADIENT ALGORITHM)

AdaGrad (Adaptive Gradient Algorithm) introduces an innovative twist to the conventional Gradient Descent optimization technique by dynamically adapting the learning rate, allowing for a more nuanced and effective optimization process. Imagine a scenario where our group of hikers, navigating the foggy valley, now has access to a map highlighting area of varying difficulty. With this map, they can adjust their pace, taking smaller steps in steep, difficult terrain and larger strides in flatter regions to optimize their path toward the valley's bottom.

Here's a closer look at AdaGrad:

- **Objective**: AdaGrad aims to fine-tune the model's parameters to minimize the cost function, similar to Gradient Descent. Its distinctive feature is individually adjusting learning rates for each parameter based on the historical gradient information for those parameters. This leads to more aggressive learning rate adjustments for weights tied to rare but important features, ensuring these parameters are optimized adequately when their respective features play a role in predictions.
- **How it works:**
    - **Initialization:** Begin with random values for the model's parameters and initialize a gradient accumulation variable, typically a vector of zeros, of the same size as the parameters.
    - **Gradient computation:** Square and accumulate the gradients in the gradient accumulation variable, which consequently tracks the sum of squares of the gradients for each parameter.

- **Adjust learning rate:** Modify the learning rate for each parameter inversely proportional to the square root of its accumulated gradient, ensuring parameters with smaller gradients to have larger updates.
- **Update parameters:** Update each parameter using its adjusted learning rate and the computed gradient.

- **Mathematical representation:** The parameter update rule for Mini-batch Gradient Descent can be represented as

$$_{\omega}:=\omega-\frac{\alpha}{\sqrt{G_t+\epsilon}}\nabla_{\omega}J_{\blacksquare}(\omega)$$

Where w represents the model's parameters (weights), α is the initial learning rate, G is the accumulation of the squared gradients, $\in$ is a small smoothing term to prevent division by zero, and $\Delta wJ(w)$ is the gradient of the cost function J(w) for the training samples with respect to w.

Advantages:

- Adaptive learning rates: By adjusting the learning rates based on past gradients, AdaGrad can effectively handle data with sparse features and different scales.
- Simplicity and efficiency: AdaGrad simplify the need for manual tuning of the learning rate, making the optimization process more straightforward.

Challenges:

- **Diminishing learning rates**: As training progresses, the accumulated squared gradients can grow very large, causing the learning rates to shrink and become infinitesimally small. This can prematurely halt the learning process.

3.9 RMSPROP (ROOT MEAN SQUARE PROPAGATION)

RMSprop (Root Mean Square Propagation) is an adaptive learning rate optimization algorithm designed to address AdaGrad's diminishing learning rates issue.

Continuing with the analogy of hikers navigating a foggy valley, RMSprop equips our hikers with an adaptive tool that allows them to maintain a consistent pace despite the terrain's complexity. This tool evaluates the recent terrain and adjusts their steps accordingly, ensuring they neither get stuck in difficult areas due to excessively small steps nor overshoot their target with overly large steps. RMSprop achieves this by

modulating the learning rate based on a moving average of the squared gradients. Here's an in-depth look at RMSprop:

- **Objective:** RMSprop, like its predecessors, aims to optimize the model's parameters to minimize the cost function. Its key innovation lies in adjusting the learning rate for each parameter using a moving average of recent squared gradients, ensuring efficient and stable convergence.
- **How it works**:
  - **Initialization:** Start with random initial values for the model's parameters and initialize a running average of squared gradients, typically as a vector of zeros of the same size as the parameters.
  - **Compute gradient**: Calculate the gradient of the cost function with respect to each parameter using a selected subset of the training data (mini-batch).
  - **Update squared gradient average:** Update the running average of squared gradients using a decay factor, γ, often set to 0.9. This moving average emphasizes more recent gradients, preventing the learning rate from diminishing too rapidly.
  - **Adjust learning rate:** Scale down the gradient by the square root of the updated running average, normalizing the updates and allowing for a consistent pace of learning across parameters.
  - **Update parameters**: Apply the adjusted gradients to update the model's parameters.
- **Mathematical representation:** The parameter update rule for RMSprop can be represented as follows:

$$\omega := \omega - \frac{\alpha}{\sqrt{E[(g)^2]_t + \epsilon}} \nabla_\omega J_{\blacksquare}(\omega)$$

Here, w represents the parameters, α is the initial learning rate, $E[g^2]_t$ is the running average of squared gradients at a certain time, ϵ is a small smoothing term to prevent division by zero, and $\Delta wJ(w)$ is the gradient of the cost function $J(w)$ with respect to w.

Advantages:

- **Adaptive learning rates:** RMSprop dynamically adjusts learning rates, making it robust to the scale of gradients and well-suited for optimizing deep neural networks.
- **Overcoming AdaGrad's limitations:** By focusing on recent gradients, RMSprop prevents the aggressive, monotonically decreasing learning rate problem seen in AdaGrad, ensuring sustained progress in training.

3.10 ADAM (ADAPTIVE MOMENT ESTIMATION)

Adam (Adaptive Moment Estimation) combines the best properties of AdaGrad and RMSprop to provide an optimization algorithm that can handle sparse gradients on noisy problems.

Using our hiking analogy, imagine that the hikers now have access to a state-of-the-art navigation tool that not only adapts to the terrain's difficulty but also keeps track of their direction to ensure smooth progress. This tool adjusts their pace based on both the recent and accumulated gradients, ensuring they efficiently navigate towards the valley's bottom without veering off course. Adam achieves this by maintaining estimates of the first and second moments of the gradients, thus providing an adaptive learning rate mechanism.

Here's a breakdown of Adam's approach:

- **Objective:** Adam seeks to optimize the model's parameters to minimize the cost function, utilizing adaptive learning rates for each parameter. It uniquely combines momentum (keeping track of past gradients) and scaling the learning rate based on the second moments of the gradients, making it effective for a wide range of problems.
- **How it works**:
- **Initialization:** Start with random initial parameter values and initialize a first moment vector (m) and a second moment vector (v). Each "moment vector" stores aggregated information about the gradients of the cost function with respect to the model's parameters:
    - The first moment vector accumulates the means (or the first moments) of the gradients, acting like a momentum by averaging past gradients to determine the direction to update the parameters.

- The second moment vector accumulates the variances (or second moments) of the gradients, helping adjust the size of the updates by considering the variability of past gradients.

Both moment vectors are initialized to zero at the start of the optimization. Their size is identical to the size of the model's parameters (i.e., if a model has N parameters, both vectors will be vectors of size N).

Adam also introduces a bias correction mechanism to account for these vectors being initialized as zeros. The vectors' initial state leads to a bias towards zero, especially in the early stages of training, because they haven't yet accumulated enough gradient information. To correct this bias, Adam adjusts the calculations of the adaptive learning rate by applying a correction factor to both moment vectors. This factor grows smaller over time and asymptotically approaches 1, ensuring that the influence of the initial bias diminishes as training progresses.

- **Compute gradient:** For each mini-batch, compute the gradients of the cost function with respect to the parameters.
- **Update moments:** Update the first moment vector (m) with the bias-corrected moving average of the gradients. Similarly, update the second moment vector (v) with the bias-corrected moving average of the squared gradients.
- **Adjust learning rate:** Calculate the adaptive learning rate for each parameter using the updated first and second moment vectors, ensuring effective parameter updates.
- **Update parameters:** Use the adaptive learning rates to update the model's parameters.
- The second moment vector accumulates the variances (or second moments) of the gradients, helping adjust the size of the updates by considering the variability of past gradients.
- **Mathematical representation:** The parameter update rule for Adam can be expressed as

$$_{\omega}:= \omega - \frac{\alpha.\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}$$

Where w represents the parameters, α is the learning rate, and m_t and v_t are bias-corrected estimates of first and second moments of the gradients, respectively.

Advantages:

- **Adaptive learning rates:** Adam adjusts the learning rate for each parameter based on the estimates of the gradients' first and second moments, making it robust to variations in gradient and curvature.
- **Bias correction:** The inclusion of bias correction helps Adam to be effective from the very start of the optimization process.
- **Efficiency**: Adam is computationally efficient and is well-suited for problems with large datasets or parameters.

Check your progress-2

a. The _____ rate controls how fast or slow Gradient Descent moves toward the optimal weights.
b. In Stochastic Gradient Descent, gradients are computed based on a _____ training example.
c. RMSprop addresses AdaGrad's issue of diminishing learning rates by using a _____ average of squared gradients.
d. Adam combines the features of AdaGrad and _____ to provide an adaptive learning rate optimization algorithm.
e. The second moment vector in Adam accumulates the _____ of gradients.
f. Gradient Descent is guaranteed to converge to the global minimum in all optimization problems.(True/False)
g. Mini-batch Gradient Descent strikes a balance between the computational efficiency of SGD and the stability of traditional Gradient Descent.
h. AdaGrad ensures that parameters associated with frequently occurring features are updated more aggressively than those associated with rare features.
i. Bias correction in Adam ensures that the algorithm performs well even in the early stages of training.
j. Mini-batch Gradient Descent can parallelize computations, leading to faster training.

3.11 LET US SUM UP

In this unit, we have discussed the backpropagation process and its role in training neural networks. We explored the mathematical derivations involved in backpropagation, focusing on gradient calculations and parameter updates. Additionally, we examined various gradient descent variants, including Stochastic

Gradient Descent (SGD), RMSProp, Adam, and Adagrad. We compared these optimization algorithms in terms of their efficiency and effectiveness in training deep learning models.

3.12 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a *loss/cost*
1-b backword
1-c Activation
1-d learning
1-e True
1-f True
2-a learning
2-b single
2-c moving
2-d RMSprop
2-e variance
2-f False
2-g True
2-h False
2-i True
2-j True

3.13 ASSIGNMENTS

- Explain the concept of backpropagation in neural networks.
- Derive the mathematical equations used for calculating the gradients during backpropagation.
- Compare and contrast the Gradient Descent, Stochastic Gradient Descent (SGD), and Mini-batch Gradient Descent algorithms.
- Explain maxima and minima.
- What role do optimizers play in deep learning?

Unit-4: Feature Selection and Extraction

4

Unit Structure

4.0. Learning Objectives
4.1. Introduction
4.2. Curse of Dimensionality
4.3. Dimensionality Reduction
4.4. Feature Selection
4.5. Feature Extraction
4.6. Techniques: Principal Component Analysis (PCA),
4.7. Linear Discriminant Analysis (LDA)
4.8. Let us sum up
4.9. Solution Check your Progress
4.10. Assignment

4.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand dimensionality reduction, feature selection, and feature extraction techniques.
- Understand the principles of Principal Component Analysis (PCA) for dimensionality reduction.
- Understand the key ideas behind Linear Discriminant Analysis (LDA) and its role in feature extraction for classification tasks.
- Evaluate the effectiveness of PCA and LDA in improving model performance and reducing computational complexity.

4.1 INTRODUCTION

Nowadays, data mining and knowledge discovery have a great role in several digital applications. Knowledge is detected by processing and analyzing a large amount of the previously collected data. Data generated in a huge volume in different fields, and it is on continuous growth in size, complexity, and dimensionality. A dataset with high dimensionality features its numerous features, but few samples have a direct relation with data mining and machine learning tasks. Therefore, these issues of data become a big challenge for extracting potentially useful, and ultimately understandable patterns or information in almost every data mining task. Also, working in high dimensional data increases the difficulty of knowledge discovery and pattern classification because there are a lot of redundant and irrelevant features. Reducing high dimensional datasets to a low dimensional dataset by filter or remove redundant and noise information is a method to solve this problem, and this is known as dimensionality reduction.

4.2 CURSE OF DIMENSIONALITY

In machine learning, increasing the number of features or observables in a problem results in higher computational costs, more memory usage for storing inputs and intermediate results, and a greater need for data samples for effective learning. While, in theory, adding more features could improve model performance, in practice, the opposite often occurs. This is due to the "curse of dimensionality," where the number of training examples required grows exponentially as the number of features increases, leading to diminished performance and efficiency.

4.3 DIMENSIONALITY REDUCTION

Data mining and machine learning face significant challenges when handling large datasets with many attributes. The feature space's dimensionality, also known as model attributes, plays a crucial role in the complexity of these processes. As the number of dimensions increases, processing algorithms become more difficult to implement and more time-consuming. These attributes, which can be either variables or features, represent the fundamental characteristics of the data. When there are numerous features, it becomes increasingly challenging to analyze them all, making the training process more complicated. The complexity increases further when a large number of features are highly correlated, leading to issues such as irrelevant classifications.

In such cases, dimensionality reduction techniques become highly valuable. Dimensionality reduction is essentially the process of transforming a large set of random variables into a smaller set of major variables, preserving essential information while eliminating unnecessary complexity. As a preprocessing step in data mining, dimensionality reduction helps mitigate the effects of noise, correlations, and excessive dimensionality, ultimately improving the performance and efficiency of machine learning models.

There are two main approaches to dimensionality reduction namely; feature selection and feature extraction.:

- **Feature Selection:** Feature selection is a means of selecting the input data set's optimal, relevant features and removing irrelevant features.
    - **Filter methods:** This method filters down the data set into a relevant subset.
    - **Wrapper methods**: This method uses the machine learning model to evaluate the performance of features fed into it. The performance determines whether it's better to keep or remove the features to improve the model's accuracy. This method is more accurate than filtering but is also more complex.
    - **Embedded methods:** The embedded process checks the machine learning model's various training iterations and evaluates each feature's importance
- **Feature Extraction:** Feature extraction involves creating new features by combining or transforming the original features. The goal is to create a set of features that captures the essence of the original data in a lower-dimensional space. There are several methods for feature extraction, including principal

component analysis (PCA), linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE). PCA is a popular technique that projects the original features onto a lower-dimensional space while preserving as much of the variance as possible.

4.4 FEATURE SELECTION

Feature selection is utilized to reduce the dimensionality impact on the dataset through finding the subset of features which efficiently define the data . It selects the important and relevant features to the mining task from the input data and removes redundant and irrelevant features. It is useful for detecting a good subset of features that is appropriate for the given problem. The main purpose of feature selection is to construct a subset of features as small as possible but represents the whole input data vital features. Feature selection provides numerous advantages: reduce the size of data, decrease needed storage, prediction accuracy improvement, overfitting evading, and reduce executing and training time from easily understanding variables.

Feature selection algorithm phase is divided into two-phase such as
(i) Subset Generation: we need to generate subset from the input dataset.
(ii) Subset Evaluation: we have to check whether the generated subset is optimal or not.
Figure 4.1 shows the overall method of the feature selection process.

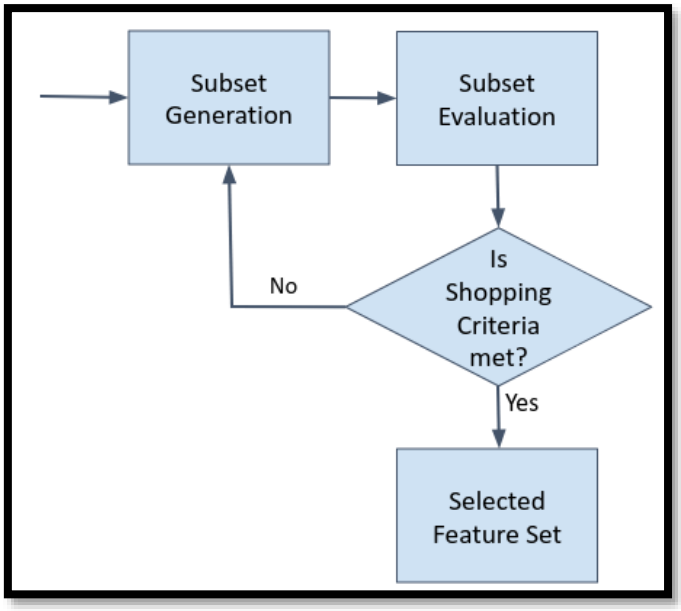


Figure 4.1: Feature Selection Process

Feature Selection Methods:

It is the process of choosing a few characteristics from a given set of potential features and then eliminating the other characteristics. Feature selection can be used to avoid having duplicate or irrelevant features or to obtain a small number of attributes to avoid overfitting. The capacity to select features is a critical skill for data scientists. The machine learning algorithm's performance depends on your ability to select the most pertinent features for analysis. The learning performance, accuracy, and processing cost of an algorithm might be negatively impacted by features that are noisy, redundant, or irrelevant. As the amount and complexity of the usual dataset continue to grow at an exponential rate, feature selection will become ever more crucial. Figure 4.2 gives the list of available feature selection techniques.

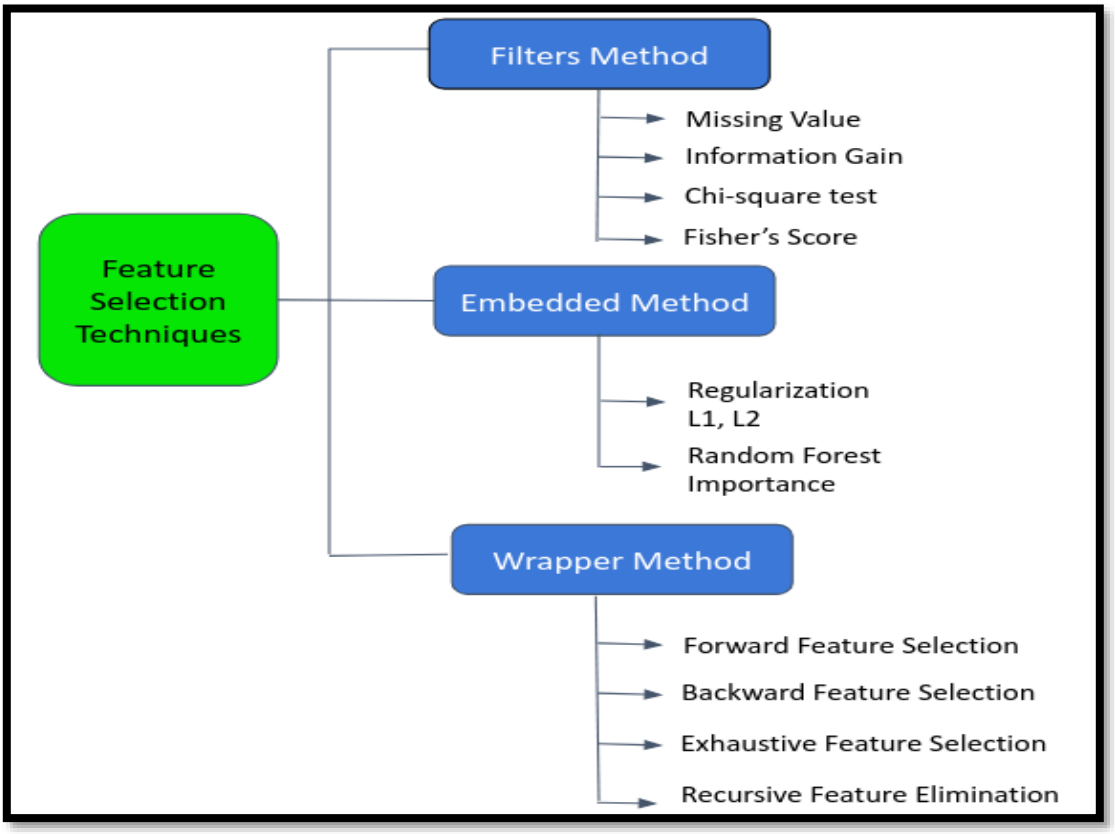


Figure 4.2: Feature Selection Techniques

Filter Method:

Filter, commonly referred to as an open-loop method, is regarded as the oldest approach. Before beginning the learning activities, it verifies the features based on the inherent properties. Information, dependency, consistency, and distance are the four types of measurement criteria that are primarily used to measure the feature qualities.

The feature selection procedure in the filter approach is carried out separately from the data mining algorithm. It assesses the subset's ranking using statistical criteria. The method can compute with high efficiency and good performance, and is easily scalable in high-dimensional datasets. This method's main drawback is that it ignores the relationship between the induction algorithm's performance and the chosen subset. The advantage of using filter methods is that it needs low computational time and does not overfit the data. Figure 4.3 shows the general approach of applying filter method.

Figure 4.3: Filter Method

Some common techniques of Filter methods are as follows:

- **Missing Value Ratio: T**he feature set can be assessed against the threshold value using the missing value ratio value. The missing value ratio can be calculated by dividing the total number of observations by the number of missing values in each column. If the variable's value exceeds the threshold, it can be removed.

$$Ratio\ of\ missing\ values = \frac{Number\ of\ missing\ Values}{Total\ number\ of\ observations} * 100$$

- **Information Gain:** Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

- **Chi-square Test:** Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

- **Fisher's score:** It is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

Embedded Method:

The embedded approach is an integrated feature selection mechanism that guides feature evaluation by integrating feature selection into the learning algorithm and utilizing its properties. While maintaining comparable performance, the embedded approach is computationally more efficient and tractable than the wrapper method. This is due to the fact that the embedded approach bypasses the need to run the classifier repeatedly and analyze each feature subset. The advantages of the filter and wrapper approaches are combined in the embedded method. It is less computationally expensive since it chooses features as the mining technique is being implemented. Figure 4.4 shows the general approach of applying the embedded method.

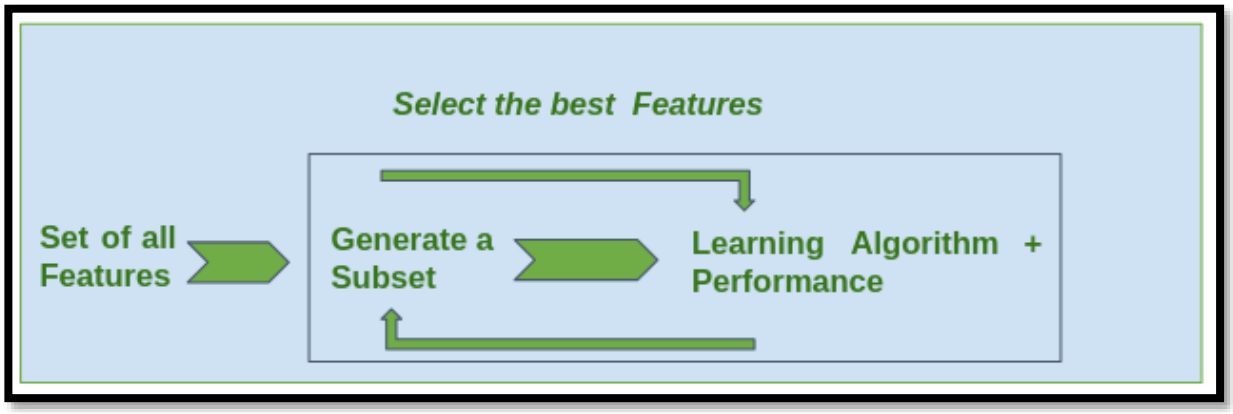


Figure 4.4: Embedded Method

Some techniques of embedded methods are:

- **Regularization:** To prevent overfitting in the machine learning model, regularization applies a penalty term to various model parameters. Some coefficients are shrunk to zero because this penalty term is added to the coefficients. It is possible to eliminate features from the dataset that have zero coefficients. Elastic Nets (L1 and L2 regularization) or L1 Regularization (Lasso regularization) are the two types of regularization techniques.

- **Random Forest Importance:** Various tree-based feature selection techniques assist us in determining the significance of features and offer a method for feature selection. In this case, feature importance indicates which feature is more crucial for model construction or has a significant influence on the target variable. One such tree-based technique is Random Forest, a kind of bagging algorithm that combines a variety of decision trees. Across all trees, it automatically ranks the nodes according to their performance or the decline in the impurity (Gini impurity). Trees below a certain node can be pruned because nodes are ordered according to impurity levels. A subset of the most significant features is produced by the remaining nodes.

Wrapper Method:

Also known as a close-loop approach, it bases feature selection on the learning algorithm and evaluates features based on performance accuracy or classification process error rate. A classifier selects the most discriminative collection of characteristics by lowering its estimation error. Based on the learning algorithm's performance, the wrapper technique chooses the best feature for the prediction algorithm. Therefore, it outperforms the filter algorithm and achieves high accuracy. In contrast to the filter strategy, this method's primary drawbacks are its increased computational complexity and susceptibility to overfitting. On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

Figure 4.5 shows the general approach of applying wrapper method.

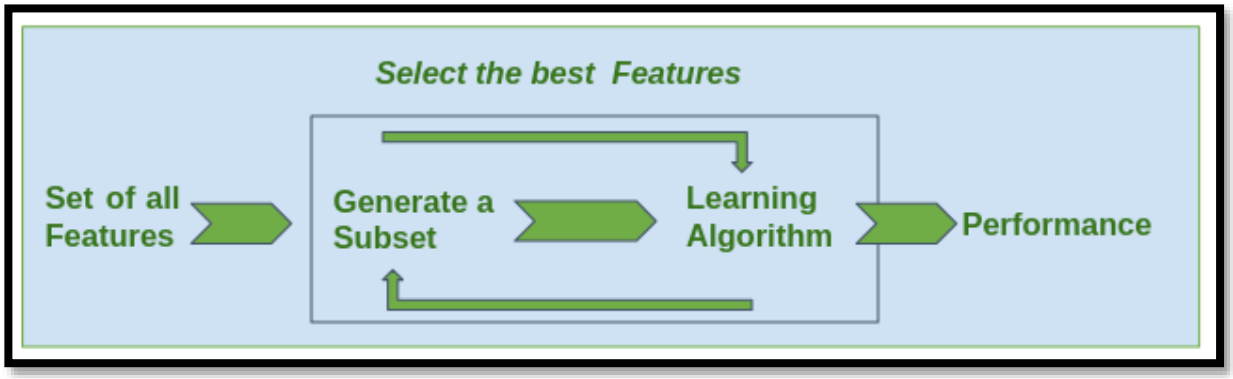


Figure 4.5: Wrapper Method

Some techniques of wrapper methods are:

- **Forward selection:** Forward selection starts with a blank set of features and is an iterative process. It continues to add features after each iteration and assesses performance to determine whether or not it is improving performance. Until adding a new variable or feature does not enhance the model's performance, the process is repeated.
- **Backward elimination: T**he reverse of forward selection, backward elimination is also an iterative process. This method starts by taking into account every aspect and eliminates the least important one. Until deleting the characteristics does not enhance the model's performance, this elimination step is repeated.
- **Exhaustive Feature Selection:** One of the greatest feature selection techniques is exhaustive feature selection, which uses a brute-force evaluation of every feature set. It indicates that this approach returns the feature set with the best performance after trying and creating every feasible feature combination**.**
- **Recursive Feature Elimination:** Recursive feature elimination is a recursive greedy optimization technique in which a decreasing subset of features is iteratively taken in order to pick features. Each set of features is then used to train an estimator, and the significance of each feature is ascertained using either feature_importances_attribute or coef_attribute.

Having seen the three methods let us have a comparative study of advantages and disadvantages of these methods as shown in table 4.1.

Method	**Advantages**	**Disadvantages**
Filter	• Works faster than wrapper methods • Scalable • Classifier independent • Lower computational complexity compared to wrapper • Better generalizability	• Neglects interaction between classifiers • Ignores feature dependency

Embedded	• Interacts with the classifier • Better computational complexity than wrapper • Higher accuracy than filter • Less prone to overfitting than wrapper • Considers feature dependency	• Classifier-specific
Wrapper	• Interacts with the classifier • Considers dependence among features • Higher performance accuracy than filter	• Prone to overfitting • Classifier-specific • Requires expensive computation

Table 4.1 Comparison of Filter, Embedded and Wrapper Methods

4.5 FEATURE EXTRACTION

"Feature extraction" is the process of minimizing the resources required to describe a significant volume of data. Many variables need to be monitored, which is one of the primary issues with complex data analysis. A classification method that has a lot of variables may overfit to training examples and not generalize to new samples, which also uses a lot of memory and processing power. A general name for many approaches to combining variables to circumvent these issues while maintaining an accurate representation of the data is feature extraction.

Many people who work in machine learning believe that the secret to creating effective models is to extract features as best as possible. The features of the data must display the information in a manner that satisfies the requirements of the algorithm that will be applied to solve the issue. While it is possible to extract certain "inherent" features directly from the raw data, most of the time we must use these "inherent" features to identify "relevant" aspects that we may utilize to address the issue.
In simple terms "feature extraction." can be described as a technique for defining a set of features, or visual qualities, that best show the information.

Common examples in machine learning include feature extraction techniques like PCA, ICA, LDA, LLE, t-SNE, and AE.

Check Your Progress-1

a. Which of the following is a technique for dimensionality reduction?
   a) Decision Trees
   b) Principal Component Analysis (PCA)
   c) k-Nearest Neighbors (k-NN)
   d) Logistic Regression

b. What is the main purpose of dimensionality reduction?
   a) To increase the number of features in the dataset
   b) To generate synthetic data points
   c) To ensure data follows a Gaussian distribution
   d) To improve interpretability and reduce computational complexity

c. Feature extraction is best described as:
   a) Selecting a subset of relevant features from the original dataset
   b) Removing irrelevant data points
   c) Transforming raw data into a set of derived features
   d) Clustering features into groups

d. What is the key difference between feature selection and feature extraction?
   a) Feature selection creates new features, while feature extraction removes features
   b) Feature selection retains original features, while feature extraction generates derived features
   c) Feature selection is used for classification tasks, while feature extraction is used for clustering tasks
   d) Feature selection is computationally more complex than feature extraction

e. What is the main advantage of using filter methods for feature selection?
   a) They have low computational time and avoid overfitting
   b) They ensure high accuracy
   c) They work directly with the learning algorithm
   d) They test all possible feature combinations

f. What is the main drawback of wrapper methods?
   a) They ignore statistical criteria
   b) They are computationally expensive and prone to overfitting
   c) They cannot handle high-dimensional datasets
   d) They do not use learning algorithms for feature selection

4.6 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA), originally developed by Karl Pearson, is a method used to reduce the number of dimensions in a dataset. It does this by mapping the data to a lower-dimensional space that captures the maximum variance. Essentially, PCA identifies the main components that represent the data's most important patterns, focusing on the directions with the greatest spread. This technique is especially valuable when dealing with datasets with three or more dimensions.
The PCA computation process is summarized in the steps below, showing that how the principal components are calculated and how they relate to the original data:

1. **Standardize the Data**: Since PCA requires the data to be on a comparable scale, the first step is to standardize it. This involves adjusting each variable so that it has a mean of 0 and a standard deviation of 1.
2. **Compute the Covariance Matrix**: Next, calculate the covariance matrix for the standardized data. This matrix indicates how each variable in the dataset correlates with the others.
3. **Determine Eigenvectors and Eigenvalues**: After obtaining the covariance matrix, calculate its eigenvectors and eigenvalues. Eigenvectors reveal the primary directions of data variation, while eigenvalues show the magnitude of variation along these directions.
4. **Select Principal Components**: The principal components are chosen based on the eigenvectors with the largest eigenvalues, as they capture the most significant data variance. These components define the new, reduced-dimensional space.
5. **Project the Data**: Finally, map the original data into the lower-dimensional space formed by the selected principal components. This transformation reduces dimensionality while preserving the most important patterns in the data.

Why Choose PCA?

Principal Component Analysis (PCA) is a popular method for feature extraction, primarily because it is straightforward, requires minimal hyperparameter tuning, and is easy to implement. PCA simplifies data by identifying eigenvectors and eigenvalues of the covariance matrix, making it accessible and widely understood due to its strong theoretical underpinnings.

PCA is computationally efficient, making it ideal for large datasets. It also works well across a range of data types, including continuous, categorical, and binary. Additionally, PCA provides easily interpretable results, as principal components are ranked by the variance they explain, allowing users to quickly determine which features hold the most significance in representing data variation.

PCA's simplicity, efficiency, and versatility make it a favored choice for dimensionality reduction. However, alternative techniques may better serve specific tasks, depending on data type and analysis objectives. For instance, Linear Discriminant Analysis (LDA) often outperforms PCA in classification, as it focuses on maximizing the separation between classes rather than variance. LDA includes the target variable, enhancing its effectiveness for categorically labeled data in classification tasks.

Considerations When Choosing a Dimension-Reduction Technique

1. **Combined Approaches**: Combining feature selection with feature extraction can further improve machine learning model performance.
2. **Type of Data**: Each technique suits different data types. PCA is particularly effective for continuous data, while LDA is more suitable for categorical data.
3. **Purpose of Analysis**: The aim of the analysis is crucial; PCA is designed to maximize data variance, while LDA focuses on distinguishing between different classes.
4. **Data Dimensionality**: Some techniques excel at visualizing high-dimensional data, like t-SNE, whereas PCA is better suited for reducing the dimensions of large datasets.
5. **Complexity of the Data**: The complexity of each method varies; for example, PCA is simpler to implement, whereas techniques like Independent Component

Analysis (ICA) are more complex and may require additional expertise. Choosing the right technique based on data complexity can enhance analysis quality.

Let us have a look at a working example.

Example

Step - 1 Collect the data

Assume that the following data points or features are given. Let us start with 4 data points;

Feature	*E1*	*E2*	*E3*	*E4*
X_1	2	3	4	5
X_2	4	6	8	10

$$\underline{X}_1 = \frac{2+3+4+5}{4} = \frac{14}{4} = 3.5$$

$$\underline{X}_2 = \frac{4+6+8+10}{4} = \frac{28}{4} = 7$$

Step - 2 Covariance Matrix Computation

$$S = [\ cov(X_1, X_1)\ cov(X_1, X_2) \quad cov(X_2, X_1)\ cov(X_2, X_2)\ ]$$

$$cov(X_1, X_1) = \frac{1}{N-1}\sum_{k=1}^{N} \quad (X_{1k} - \underline{X}_1)(X_{1k} - \underline{X}_1)$$

$$= \frac{1}{N-1}\sum_{k=1}^{N} \quad (X_{1k} - \underline{X}_1)^2$$

$$= \frac{1}{3}[(2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2]$$

$$= \frac{1}{3}(2.25 + 0.25 + 0.25 + 2.25) = \frac{1}{3} * 5.5 = 1.83$$

$$cov(X_1, X_2) = \frac{1}{N-1}\sum_{k=1}^{N} \quad (X_{1k} - \underline{X}_1)(X_{2k} - \underline{X}_2)$$

$$= \frac{1}{3}[(2 - 3.5)(4 - 7) + (3 - 3.5)(6 - 7) + (4 - 3.5)(8 - 7) + (5 - 3.5)(10 - 7)]$$

$$= \frac{1}{3}[-1.5 * -3 + 0.5 * -1 + 0.5 * 1 + 1.5 * 3]$$

$$= \frac{1}{3}[4.5 + 0.5 + 0.5 + 4.5]$$

$$= \frac{1}{3} * 10 = 3.33$$

$$cov(X_2, X_1) = 3.33$$

$$cov(X_2, X_2) = \frac{1}{N-1}\sum_{k=1}^{N} (X_{2k} - \underline{X_2})(X_{2k} - \underline{X_2})$$

$$= \frac{1}{N-1}\sum_{k=1}^{N} (X_{2k} - \underline{X_2})^2$$

$$= \frac{1}{3}[(4 - 7)^2 + (6 - 7)^2 + (8 - 7)^2 + (10 - 7)^2]$$

$$= \frac{1}{3}(9 + 1 + 1 + 9) = \frac{20}{3} = 6.67$$

The final covariance matrix is the following

$$COV\ matrix = [\ 1.83 \quad 3.33 \quad 3.33 \quad 6.67\ ]$$

Step - 3 Eigen values and Eigen vector Computation

Next, we will compute the eigen values of this covariance matrix. The characteristic equation of the covariance matrix is the following.

$$0 = det\ det\ (S - \lambda\ I)$$

$$= | \ 1.83 - \lambda \ 3.33 \quad 3.33 \ 6.67 - \lambda \ |$$

$$= (1.83 - \lambda)(6.67 - \lambda) - 3.33 * 3.33$$
$$= 1.83 * 6.67 - 1.83\lambda - 6.67\lambda + \lambda^2 - 11.09$$

$$= 12.21 - 8.5\lambda + \lambda^2 - 11.09$$

$$= \lambda^2 - 8.5\lambda + 1.12$$

To find the roots of the quadratic equation we can apply the formula as mentioned:

Quadratic Formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

By applying the formula, we will get two roots and these are the ***eigen values.***

$$Roots = \frac{8.5 \pm \sqrt{8.5^2 - 4 * 1 * 1.12}}{2}$$
$$= \frac{8.5 \pm \sqrt{72.25 - 4.48}}{2}$$
$$= \frac{8.5 \pm \sqrt{67.77}}{2}$$
$$= \frac{8.5 + 8.23}{2}, \frac{8.5 - 8.23}{2}$$
$$= 8.37, 0.14$$

Next step is to calculate eigen vectors. To calculate this the initial step is to calculate the Characteristic Equation:

The characteristic equation for finding eigenvalues (λ) of a matrix C is given by:

det(S — λI) = 0

Where:

-S is the covariance matrix.

-λ (lambda) represents the eigenvalues.

-I is the identity matrix of the same dimensions as C.

$$U = [u_1\ u_2\ ]$$
$$[\ 0\ \ 0\ ] = (s - \lambda I)U$$
$$= [1.83 - \lambda_1\ \ 3.33\ \ 3.33\ \ 6.67 - \lambda_1\ ][u_1\ u_2\ ]$$
$$= [(1.83 - \lambda_1)u_1 + 3.33u_2\ 3.33u_1 + (6.67 - \lambda_1)u_2\ \ ]$$

Equating the above to vector 0, we will get the following

$$(1.83 - \lambda_1)u_1 + 3.33u_2 = 0$$
$$3.33u_1 + (6.67 - \lambda_1)u_2 = 0$$

Taking the first equation into consideration

$$\frac{u_1}{-3.33} = \frac{u_2}{1.83 - \lambda_1} = t$$
$$u_1 = -3.33t, u_2 = (1.83 - \lambda_1)t$$

Consider the value of t as 1, the U vector or eigenvector will become

$$U = [-3.33\ 1.83 - \lambda_1\ ]$$

Step - 4 Sort Eigen Values

Whenever we want to calculate the principal components, we have to consider the largest eigen value. Here it is 8.37. To calculate the unit eigen vector, we have to calculate the length of U1.To obtain unit eigenvectors (eigenvectors with a magnitude of 1), we normalize the eigenvectors. To normalize a vector, divide each component by its magnitude (Euclidean norm).

$$\|u_1\| = \sqrt{-3.33^2 + (1.83 - 8.37)^2} = \sqrt{11.09 + 42.77} = \sqrt{53.86} = 7.34$$

$$e_1 = \left[\frac{-3.33}{7.34}\ \frac{1.83 - 8.37}{7.34}\right] = [-0.45\ -0.89\ ]$$

The below represents the eigenvector corresponding to Principal component 2, if we are calculating PC2 we can consider the below eigenvector.

$$u_2 = [-3.33\ 1.83 - \lambda_2\ ]$$

$$\|u_2\| = \sqrt{(-3.33)^2 + (1.83 - 0.14)^2} = \sqrt{11.09 + 2.86} = 3.73$$

$$e_2 = \left[\frac{-3.33}{3.73}\ \frac{1.83 - 0.14}{3.73}\right] = [-0.89\ 0.45\ ]$$

Step - 5 Select First Principal Component

The equation to calculate the principal component is as follows.

$$e_1^T = \left[X_{1k} - \underline{X}_1\ X_{2k} - \underline{X}_2\ \right]$$

Applying the above equation on the first eigen vector we will get the following.

$$[-0.45\ \ -0.89]\left[X_{1k} - \underline{X}_1\ X_{2k} - \underline{X}_2\ \right]$$

$$= -0.45\left(X_{1k} - \underline{X}_1\right) - 0.89(X_{2k} - \underline{X}_2)$$

Step — 6 Projection onto Principal Components

Applying the equation on each data point.

$$(2,4) = -0.45(2 - 3.5) - 0.89(4 - 7) = 0.68 + 2.67 = 3.35$$

$$(3,6) = -0.45(3 - 3.5) - 0.89(6 - 7) = 0.23 + 0.89 = 1.12$$

$$(4,8) = -0.45(4 - 3.5) - 0.89(8 - 7) = -0.23 - 0.89 = -1.12$$

$$(5,10) = -0.45(5 - 3.5) - 0.89(10 - 7) = -0.68 - 2.67 = -3.35$$

The final answer,

X_1	2	3	4	5
X_2	4	6	8	10
PC_1	3.35	1.12	-1.12	-3.35

The final result can be visualized approximately as shown in figure 4.6.

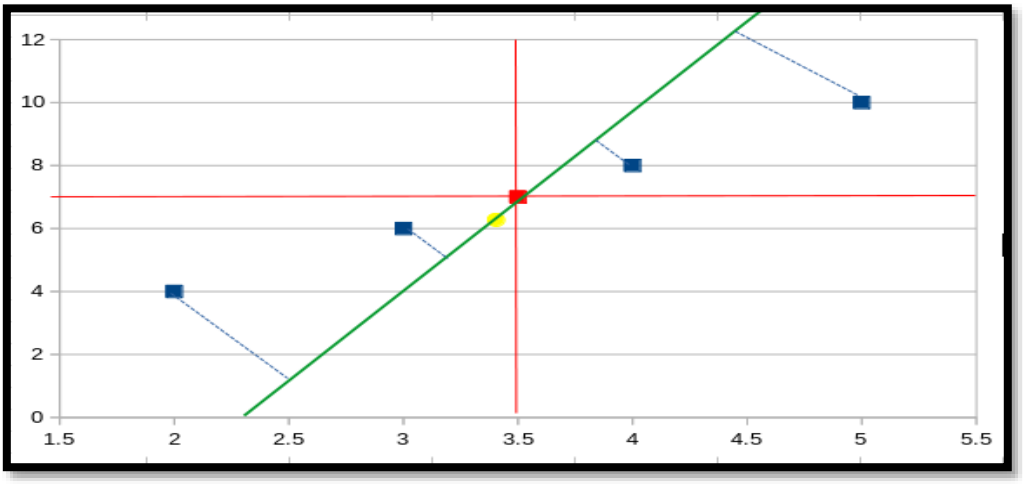


Figure 4.6: Projection onto Principal Components

Blue points are the data points, the red point is the average point, the yellow circle is the eigen vector point (-0.45,-0.89) and the green line is the first principal component.

Advantages and Disadvantages

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and feature extraction, with its own set of strengths and limitations.

Advantages

- **Dimensionality Reduction:** PCA reduces the number of features while retaining most of the variance in the data, simplifying analysis and visualization.

- **Elimination of Multicollinearity:** It transforms correlated features into a set of uncorrelated principal components.
- **Improved Model Performance:** By removing noise and redundancy, PCA can enhance the performance of machine learning models.
- **Versatility:** PCA can be applied to a wide range of domains, from image processing to financial modeling.
- **Efficiency in Large Datasets:** PCA handles high-dimensional data effectively, reducing computational complexity.

Disadvantages

- **Loss of Interpretability**: The transformed features (principal components) lack direct interpretability as they are linear combinations of the original features.
- **Sensitivity to Scaling**: PCA is sensitive to the scale of the data, requiring standardization or normalization before application.
- **Assumption of Linearity**: It assumes linear relationships among features, making it less suitable for non-linear datasets.
- **Impact of Noise**: PCA can capture noise as part of the principal components, especially if the noise has significant variance.
- **Dependence on Variance**: PCA prioritizes components with the highest variance, which may not always align with the most relevant features for the specific problem.

Applications

PCA is extensively used across industries to simplify data and uncover hidden patterns. Here are some key applications:

- **Image Compression and Processing**: PCA reduces the dimensionality of image data while retaining essential features, making it useful for compressing and denoising images.
- **Genomics and Bioinformatics**: In biological studies, PCA is employed to analyze high-dimensional genomic data, identifying patterns and relationships among genes or individuals.
- **Stock Market Analysis**: PCA is used in finance to identify principal components in stock price movements, helping to uncover underlying trends and reduce noise.
- **Customer Behavior Analysis**: In marketing, PCA helps group customers based on behavioral data, facilitating targeted campaigns and strategic planning.

- **Speech and Audio Signal Processing**: PCA simplifies audio data for tasks like speaker identification and audio classification.
- **Fault Detection in Manufacturing**: PCA analyzes sensor data to identify anomalies, enabling the early detection of equipment malfunctions.
- **Healthcare Analytics**: PCA helps process medical imaging data or clinical datasets to extract meaningful features for diagnostics and prediction.

By reducing data complexity while preserving essential patterns, PCA serves as a powerful tool in both exploratory data analysis and predictive modeling.

Check Your Progress-2

a. In PCA, why is data standardization necessary before applying the technique?
   a) To increase the number of dimensions
   b) To make the data non-linear
   c) To ensure variables are on a comparable scale
   d) To decrease the computational time

b. The covariance matrix in PCA represent the correlations between variables. (True/False)

c. _____defines the primary directions of data variation in PCA?

d. The smallest eigenvalues determine the selection of principal components in PCA. (True/False)

e. In PCA, the principal components are linear combinations of _____.

f. Why is PCA sensitive to scaling of data?
   a) It relies on the variance of features, which depends on their scale
   b) It assumes data points are scaled to integers
   c) It is designed to work with binary data
   d) Scaling increases computational complexity

g. Which equation is used to calculate eigenvalues in PCA?
   a) $A \times B = C$
   b) $\det(S - \lambda I) = 0$
   c) $X^T X = Y$
   d) $P^2 = X + Y$

4.7 LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis (LDA) is a supervised learning technique used for classification tasks. It helps distinguish between different classes by projecting data points onto a lower-dimensional space, maximizing the separation between those classes.

LDA performs two key roles:

1. **Classification:** It finds a linear combination of features that best separates multiple classes.
2. **Dimensionality Reduction:** It reduces the number of input features while preserving the information necessary for classification.

LDA is very similar to Principal Component Analysis (PCA), but there are some important differences. PCA is an unsupervised algorithm, meaning it doesn't need class labels ***y*** . PCA's goal is to find the principal components that maximize the variance in a dataset. LDA, on the other hand, is a supervised algorithm, which uses both the input data ***x*** and the class labels ***y*** to find linear discriminants that maximize the separation between multiple classes.

Key Assumptions of Linear Discriminant Analysis (LDA)

1. **Gaussian Distribution**: LDA presumes that the features within each class are distributed normally (i.e., follow a Gaussian distribution).
2. **Equal Covariance Matrices**: It assumes uniformity in the variance (spread) of data points across all classes.
3. **Linearity**: The method assumes a linear relationship between the features and the target variable.
4. **Feature Independence**: LDA operates under the assumption that the features are ideally independent of one another.

Limitations Stemming from These Assumptions

1. **Sensitivity to Data Distribution**: LDA's performance can decline if the features deviate significantly from a Gaussian distribution or if the class covariances differ substantially.

2. **Effect of Multicollinearity**: Highly correlated input features can negatively impact LDA's effectiveness.
3. **Challenges with Complex Data**: Datasets with intricate, non-linear relationships may not be well-suited for LDA, resulting in lower classification accuracy.

Preparing for Linear Discriminant Analysis Implementation

To achieve reliable results, proper data preparation is crucial before implementing LDA. Below are the key steps for effective preparation:

Steps for Data Preparation

1. **Data Cleaning**: Identify and eliminate any missing, inaccurate, or inconsistent data points that could skew the results.
2. **Feature Selection**: Select features relevant to the classification problem to avoid unnecessary model complexity.
3. **Addressing Multicollinearity**: For highly correlated features, either remove redundant ones or combine them to enhance model performance.
4. **Feature Scaling**: While LDA is relatively robust to variations in feature scale, standardizing features (mean=0, variance=1) is beneficial when their ranges differ significantly.

By carefully preparing the data, LDA can better identify and represent the relationships between the features and class labels, leading to improved classification outcomes.

How Does LDA Work?

Linear Discriminant Analysis (LDA) aims to identify a new axis or subspace that effectively separates different classes by maximizing inter-class differences and minimizing intra-class variation. It achieves this while reducing the data's dimensionality and preserving class-discriminative features.

Key Concepts

- **Maximizing Between-Class Variance**: LDA focuses on maximizing the distance between the mean vectors of different classes, ensuring they are as distinct as possible.

- **Minimizing Within-Class Variance**: It minimizes the variance within each class, keeping data points of the same class close together.
- **Projection to a Lower-Dimensional Space**: LDA projects the original dataset onto a new axis or subspace. For instance, in a problem involving three classes, it can reduce the dimensionality to two or even one dimension while retaining information critical to class separation.

Working Mechanism

LDA involves the following steps:

1. **Compute Mean Vector**: Calculate the mean vector for each class in the dataset.

$$m_i = \frac{1}{n_i} \sum_{i=1}^{n_i} \quad x_i$$

2. **Calculate Scatter Matrices**: Determine the within-class scatter matrix (representing the variance within each class) and the between-class scatter matrix (representing the variance between class means).

   Within-class scatter matrix S_W

$$S_W = \sum_{i=1}^{c} \quad S_i$$

   where S_i is the scatter matrix for a specific class

$$S_i = \sum_{x \epsilon D_i}^{n} \quad (x - m_i)(x - m_i)^T$$

   and m_i is the mean vector for that class

$$m_i = \frac{1}{n_i} \sum_{i=1}^{n_i} \quad x_i$$

Alternatively, the class-covariance matrices can be used by adding the scaling factor 1/N-1 to the within-class scatter matrix.

$$\sum_i \quad = \frac{1}{N_i - 1} \sum_{x \epsilon D_i}^{n} \quad (x - m_i)(x - m_i)^T$$

$$S_W = \sum_{i=1}^{c} \quad (N_i - 1) \sum_i$$

Between-class scatter matrix S_B

$$S_B = \sum_{i=1}^{c} \quad N_i (m_i - m)(m_i - m)^T$$

Where

m is the overall mean,
m_i is the mean of the respective class, and
N_i is the sample size of that class.

3. **Find Eigenvectors and Eigenvalues**: Solve $S_w^{-1} S_b^V = \lambda V$ to find eigenvalues and eigenvectors. The eigenvector with the largest eigenvalue provides the direction of maximum separability.
4. **Select Top Eigenvectors**: Rank the eigenvectors by their eigenvalues in descending order and select the top k eigenvectors that capture the most significant variance.

5. **Transform Data**: Use the selected eigenvectors to project the original data onto the new subspace, resulting in a lower-dimensional representation optimized for class separation. After selecting the ***k*** eigenvectors, we can use the resulting ***d * k***-dimensional eigenvector matrix ***W*** to transform data onto the new subspace via the following equation:

$$Y = X * W$$

Example:

Compute the Linear Discriminant projection for the following two-dimensional dataset X1=(x1,x2)={(4,1),(2,4),(2,3),(3,6),(4,4)} & X2=(x1 x2)={(9,10),(6,8),(9,5),(8,7),(10,8)}

Solution:

Step 1: Compute Mean Vectors

1. Mean Vector for class $\boldsymbol{X_1(\mu_1)}$

$$\mu_1 = \frac{1}{5}\sum_{i=1}^{5} (x_{i1}, x_{i2})$$

$$= \frac{1}{5}[(4+2+2+3+4)], (1+4+3+6+4)]$$

$$= \left(\frac{15}{5}, \frac{18}{5}\right)$$

$$= (3\ , 3.6)$$

2. Mean Vector for class X2($\boldsymbol{\mu_2}$)

$$\mu_2 = \frac{1}{5}\sum_{i=1}^{5} (x_{i1}, x_{i2})$$

$$= \frac{1}{5}[(9+6+9+8+10)], (10+8+5+7+8)]$$

$$= \left(\frac{42}{5}, \frac{38}{5}\right)$$

$$= (8.4\ , 7.6)$$

Step 2: Compute Scatter Matrices

Within-Class Scatter Matrix (Sw):

$$S_W = \sum_{i \epsilon X_1} (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{i \epsilon X_2} (x_i - \mu_2)(x_i - \mu_2)^T$$

Now for class X_1: Compute each term $(x_i - \mu_1)(x_i - \mu_1)^T$

For (4, 1):

$$x_i - \mu_1 = [4 - \ 3\ 1 - 3.6\ ] = [1\ -2.6\ ]\ , (x_i - \mu_1)(x_i - \mu_1)^T$$

$= [1\ -2.6\ ][1\ -2.6] = \ [1\quad -2.6\ -2.6\ \ 6.76\ ]$ -------eq.1

For (2, 4): $[1\quad -0.4\ 0.4\ \ 0.16\ ]$ ----------eq.2

For (2, 3): $[1\quad 0.6\ 0.6\ \ 0.36\ ]$ ----------eq.3

For (3, 6): $[1\quad 0\ \ 0\ \ 5.76\ ]$ ----------eq.4

For (4, 4): $[1\quad 0\ 0.4\ \ 0.16\ ]$ ----------eq.5

Now adding eq. 1, 2, 3, 4, 5 and taking average we get covariance matrix S1

$S1 = [0.8\quad -0.4\ -0.4\quad 2.6\ ]$

Similarly, for class X_2**:** Compute each term $(x_i - \mu_2)(x_i - \mu_2)^T$ for each point in X_2 and sum

$$S2 = [1.84\quad -0.04\ -0.4\quad 2.64\ ]$$

$$\boldsymbol{Sw = S1 + S2}$$

$$S_W = [2.64\quad -0.44\ -0.44\quad 5.28\ ]$$

Between-Class Scatter Matrix (SB):

$$[\mathbf{-5.4\ -4}\ ][\mathbf{-5.4\ -4}] = \ [\mathbf{29.16\ \ 21.6\ 21.6\ \ 16.0}\ ]$$

Step 3: Find the best LDA Projection Vector.

We find this using Eigen vectors having largest Eigen value. We calculate this using formula:

$$S_w^{-1}\ S_b^V = \lambda V$$

$$i.e\ |S_W^{-1}\ \ S_b - \lambda\ I| = 0$$

$$|11.89-\lambda \quad 8.81\ 5.08 \quad 3.76-\lambda\,|=0$$

$$(11.89-\lambda)(3.76-\lambda)-5.08*8.81=0$$

$$44.7-11.89\lambda-3.76\lambda+\lambda^2-44.7=0$$

$$\lambda^2-15.65\lambda=0$$

$$\lambda\,(\lambda-15.65)=0$$

$$\lambda=0\ or\ \lambda=15.65$$

Two eigen values we get. Now we take biggest value to reduce dimension

$$[11.89\ \ 8.81\ 5.08 \quad 3.76\,][V_1\ \ V_2\ ]=15.65\,[V_1\ \ V_2\ ]$$

$$We\ get\ [V_1\ \ V_2\ ]=[0.91\ \ 0.39\ ]$$

Step 4 : Dimension reduction

$$y=W^T X$$

$$y=[0.91\ 0.39][4\ \ 2\ \ 2\ \ 3\ \ 4\ 1\ \ 2\ \ 3\ \ 6\ \ 4\,]`=[4.03\ 3.38\ 2.99\ 5.07\ 5.2]$$

Here we can see 2d data sample is changed to 1D data samples

Advantages and Disadvantages

While LDA is a robust method for classification and dimensionality reduction, it has both strengths and limitations.

Advantages

- **Ease of Use**: LDA is straightforward to implement and understand, making it ideal for those new to machine learning techniques.
- **Interpretability**: It provides clear insights into how features contribute to the classification process.
- **Efficiency**: LDA is computationally efficient, making it suitable for handling large datasets.
- **Effective for Linearly Separable Data**: It performs well when classes are linearly separable.

Disadvantages

- **Reliance on Assumptions**: LDA assumes a Gaussian distribution of features, which may not always be realistic.
- **Challenges with Non-linearity**: It struggles to handle datasets with complex, non-linear relationships.
- **Sensitivity to Class Imbalance**: When class distributions are imbalanced, LDA may favor the majority class, leading to biased results.
- **Impact of Outliers**: The presence of outliers can adversely affect its performance.

Applications

LDA finds applications in various fields due to its ability to effectively classify data. Below are some prominent use cases:

- **Face Recognition**: LDA is utilized to extract key features from facial images and classify them based on individuals. This is often employed in biometric systems for user identification or verification.
- **Medical Diagnostics**: In healthcare, LDA helps analyze medical data to classify diseases. For instance, it is used to differentiate between cancer stages or predict heart disease.
- **Customer Segmentation in Marketing**: By grouping customers based on their profiles, LDA supports targeted marketing campaigns and personalized service strategies.
- **Credit Risk Analysis**: Financial institutions apply LDA to predict creditworthiness by analyzing customer data, helping to assess the likelihood of loan defaults.
- **Quality Assurance in Manufacturing**: LDA is used to analyze sensor data and detect defects in products, enabling early identification of issues in the production process.
- **Marketing Campaign Optimization**: It aids in evaluating customer interactions to identify the most effective marketing strategies and improve campaign outcomes.

Check Your Progress-3

a. How does LDA differ from Principal Component Analysis (PCA)?
   a) LDA maximizes class separation, while PCA maximizes variance
   b) PCA uses class labels, while LDA does not
   c) LDA handles unsupervised data, while PCA handles supervised data
   d) Both use the same mathematical approach

b. _____ measures the variance within each class in LDA.

c. LDA aims to project data onto a new axis or subspace by maximizing _____ variance and minimizing _____ variance.

d. LDA assumes that the features in each class follow a _____ distribution.

e. One key limitation of LDA is its sensitivity to _____, which can negatively affect its performance.

4.8 LET US SUM UP

In this unit, we have discussed various techniques for dimensionality reduction, feature selection, and feature extraction. We explored the curse of dimensionality and its challenges. We delved into Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), understanding their principles and performing practical examples to illustrate how PCA and LDA can be applied to real-world data for improving model performance.

4.9 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a Principal Component Analysis (PCA)

1-b To improve interpretability and reduce computational complexity

1-c Transforming raw data into a set of derived features

1-d Feature selection retains original features, while feature extraction generates derived features

1-e They have low computational time and avoid overfitting

1-f They are computationally expensive and prone to overfitting

2-a to ensure variables are on a comparable scale

2-b True

2-c Eigenvectors

2-d False

2-e the original features

2-f It relies on the variance of features, which depends on their scale

2-g $\det(S - \lambda I) = 0$

3-a LDA maximizes class separation, while PCA maximizes variance

3-b Within-class scatter matrix

3-c between-class, within-class

3-d Gaussian

3-e outliers

4.10 ASSIGNMENTS

- Define the term feature selection.
- What is the purpose of feature extraction in machine learning?
- Expand the following terms : PCA,LDA
- Name components of dimensionality reduction.

Block-4
Clustering and Association Rules

Unit-1: Introduction to Clustering

1

Unit Structure

1.0. Learning Objectives

1.1. Introduction

1.2. What is Clustering

1.3. Clustering Approaches

1.4. Clustering Techniques

1.5. Let us sum up

1.6. Check your Progress: Possible Answers

1.7. Assignments

1.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the concept of clustering in machine learning.
- Identify the need for clustering in various real-world applications.
- Learn about different approaches of clustering.
- Understand the concepts of Partition-Based, Hierarchical-Based, Density-Based Clustering and Distribution-Based Clustering.

1.1 INTRODUCTION

The machine learning algorithms are frequently used in various real life applications like market segmentation where companies try to group customers based on their purchasing behaviour. Document classification wherein one tries to organize similar documents together. Image segmentation wherein one tries to identify distinct parts in an image. Anomaly detection wherein one tries to identify rare events or outliers. Many more examples of use cases are there.

To support such decision making the machine learning domain uses a concept known as Clustering. In this chapter we will learn what is clustering, need of clustering and different types of clustering mechanisms.

1.2 WHAT IS CLUSTERING

Clustering is a technique used to arrange similar or dissimilar data points into distinct groups known as clusters based on some specific criteria. In clusters, data points within the same cluster share common features. It is possible that two clusters formed after clustering may have significantly different characteristics. The primary goal of clustering is to uncover or discover concealed knowledge underlying large data sets without predefined labels. The process of clustering involves dividing large data sets into smaller more informative data sets based on some specific criteria. Figure 1.1 shows an example of basic clustering.

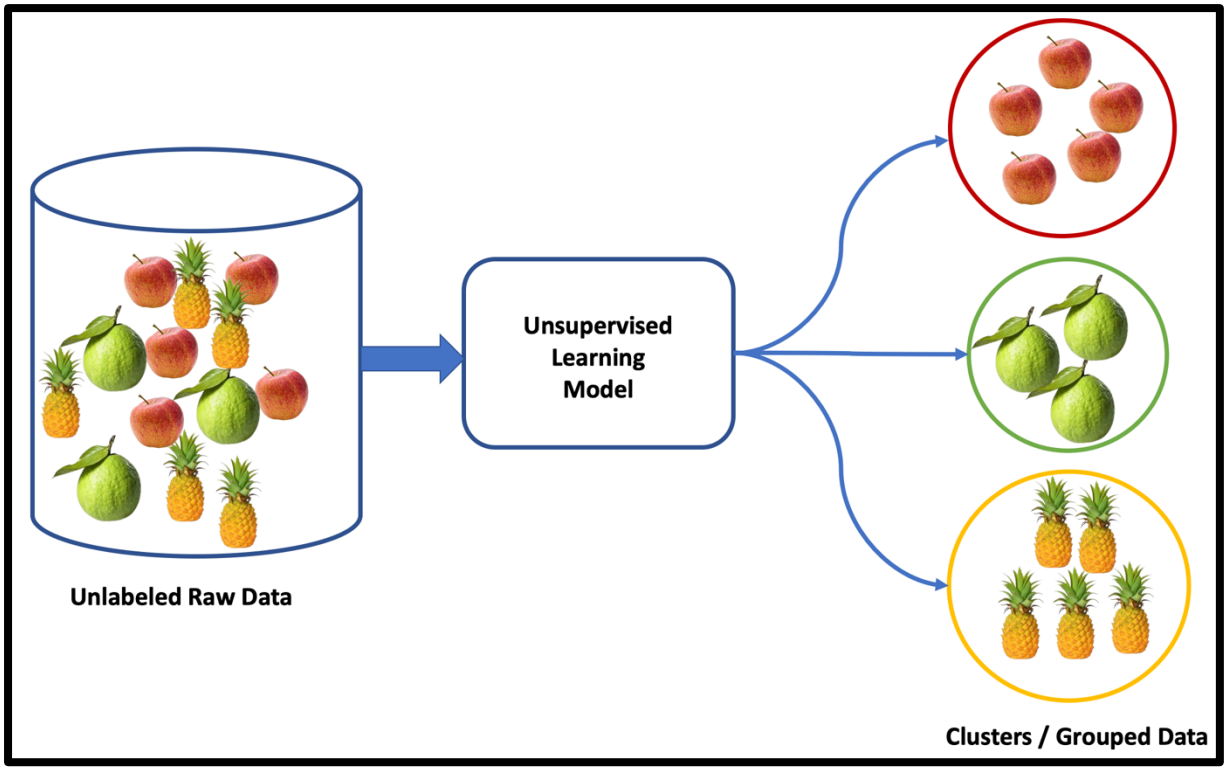


Figure 1.1: Example of Basic Clustering

Depending on the need of an application the grouping can lead to different outcomes such as partitioning of data, re-organization of data, compression of data or data summarisation. Let us have a look at the simple forms of these outcomes.

Partitioning of Data

Assume that for an E-Commerce application we have a table named ORDERS that contains the information about orders placed by customers. Let the table contain 50000 fixed length records. If we want to evenly distribute the orders in the group of months then we will need 12 distinct values of MONTH each representing the unique month of the year.

If the data within the table is clustered by MONTH then to access all the orders for the month of February (MONTH = "February") will require $\log_2(12) + 4167$ accesses. Here the first term $\log_2(12)$, involves accessing the index table that is constructed using MONTH. This could be achieved by applying a binary search. The second term, 4167 (or 50000/12), involves fetching the 4167 records from the clustered ORDERS table.

Without clustering, accessing 4167 orders for a particular month would require, on an average, 4167 · 50000/12 accesses as the records are randomly distributed.

As can be observed here the proper partitioning of the dataset improves the efficiency of query execution. Large datasets can be managed more efficiently by storing each partition separately. Since each partition is independent, database systems can process partitions in parallel also.

Re-organization of Data

Assume that we have data pertaining to orders as shown in Table 1.1. The table contains data of 10 customers. It has features like age and order total. Let us try to divide the given data set into three clusters. The first cluster will be called "Young and High Spending", wherein the age of the customer would be below 40 years and the order total would be greater than 20000. The second cluster will be called "Middle-Aged and Low Spending", wherein the age of the customer would be between 40 to 59 years and the order total would be less than 20000. The third cluster will be called "Senior", wherein the age of the customer would be 60 years and above regardless of the spending.

Table 1.1: Sample data of order

Order ID	Customer ID	Age	Order Total
1	101	22	12000
2	102	29	25000
3	103	35	7500
4	104	42	18000
5	105	52	50000
6	106	30	12000
7	107	60	30000
8	108	40	22000
9	109	25	15000
10	110	50	35000

When the process of clustering begins the data will be divided into clusters as shown in Table 1.2, 1.3 and 1.4 respectively.

Table 1.2: Data of Cluster 1 - Young and High Spending

Order ID	Customer ID	Age	Order Total
2	102	29	25000
8	108	40	22000

Table 1.3: Data of Cluster 2 - Young and High Spending

Order ID	Customer ID	Age	Order Total
1	101	22	12000
3	103	35	7500
4	104	42	18000
6	106	30	12000
9	109	25	15000

Table 1.4: Data of Cluster 3 – Senior

Order ID	Customer ID	Age	Order Total
7	107	60	30000

Observe that by creating the three clusters we have gained valuable insights into the customer behaviour. For example we can now target the young and high spending customers with premium products and exclusive offers. The middle aged and low spending customers may benefit from value based marketing or discounts to encourage them for higher spending. The senior customers may be interested in products that focus on long term comfort and lifestyle.

Such a segmentation thus allows businesses to customize their marketing strategies and product offerings, ensuring that each group receives the most relevant promotions and services.

Data Compression

The goal of data compression is to reduce the size of the dataset while still preserving its key information. This can be done by removing redundancy, representing data more efficiently, or consolidating information. It is possible to use simple techniques for compression, such as categorical encoding which allows us to reduce the size of the text field by encoding it with numerical values. Proper feature selection allows us to reduce the dataset by only keeping the most relevant features.

Data Summarization

Data summarization allows us to create a small but better representation of a large dataset. The goal of data summarization is to highlight key trends, patterns, and relationships in the data. For example we can summarize the data of Table 1.1 as shown in Table 1.5 so as to contain summarized data like Count of Orders, Average Order Total, Min Order Total and Max Order Total.

Table 1.5: Summary of data by Age group

Age Group	Count of Orders	Average Order Total	Min Order Total	Max Order Total
Young (Under 30)	5	15500	7500	25000
Middle-Aged (30-59)	4	27500	12000	50000
Senior (60 and above)	1	30000	30000	30000

From the data of Table 1.5 it can be concluded that the spending is lesser among young customers as compared to middle-aged customers, while senior customers have a single high order.

Check Your Progress-1

a) The primary goal of clustering is to conceal the knowledge underlying large data sets without predefined labels. (True/False)
b) Proper partitioning of the dataset can improve the efficiency of query execution. (True/False)
c) Re-organization of data can help businesses improve their service strategies. (True/False)
d) The goal of data compression is to increase the size of the dataset while still preserving its key information. (True/False)
e) Data summarization allows us to create a small but better representation of a large dataset. (True/False)

1.3 CLUSTERING APPROACHES

Clustering as mentioned earlier is used to group similar data points together. There are two primary approaches to clustering based on the way data points are assigned to clusters: Hard Clustering and Soft Clustering.

Hard Clustering

The clustering approach in which each data point is assigned to exactly one and only one cluster is known as hard clustering. In this technique a data point either belongs to a cluster or it does not. Assume that we have five data points { (x_1,y_1), (x_2,y_2), ….,(x_5,y_5) }. We want to segregate them into three clusters, say C1, C2 and C3. Here if we apply a hard clustering approach then the data points based on some similarity may be assigned to clusters as shown in Table 1.6.

The clustering techniques that use a hard clustering approach are easy to understand and interpret. It provides a clear separation of clusters such that we get well defined, disjoint clusters that are easy to evaluate. Generally such techniques are faster and computationally less expensive. At times though as each point belongs to only one cluster it may not reflect the true nature of the data, especially in cases of clusters where data might be overlapping.

Table 1.6: Hard Clustering of data points

Cluster	Data Points
C1	(x_1,y_1)
C2	(x_2,y_2)
C3	(x_3,y_3)
C1	(x_4,y_4)
C2	(x_5,y_5)

Soft Clustering

The clustering approach in which each data point can belong to more than one cluster, but with different degrees of membership is known as soft clustering. It is also known as fuzzy clustering due to the property that the assignment of data points to clusters is probabilistic or fuzzy. It means that the data point has a certain likelihood of belonging to one or another cluster. If we consider the data of Table 1.6 and apply a soft clustering approach then we need to evaluate the probability of data points belonging to cluster C1, C2 and C3. The said probability needs to be calculated for every data point in the data set. A sample probability calculation is shown in Table 1.7 as an example.

Table 1.7: Soft Clustering of data points

Data Points	Probability of C1	Probability of C2	Probability of C3
(x_1,y_1)	0.90	0.06	0.04
(x_2,y_2)	0.20	0.70	0.10
(x_3,y_3)	0	0	1
(x_4,y_4)	0.85	0.10	0.05
(x_5,y_5)	0.05	0.95	0

Soft clustering is a good option where data points belong to multiple clusters or are near the boundary between multiple clusters. It provides more clear insights, such as the degree of membership of a point in each cluster. At times though results of soft clustering approach are harder to interpret as each data point has a degree of membership for multiple clusters. The technique often requires more computations and can be slower when working with large datasets.

1.4 CLUSTERING TECHNIQUES

Machine learning datasets can have millions of data points. Clustering is an unsupervised machine learning technique that is used to group data points together into clusters or subsets. The data points within a cluster are more similar to each other than to those in other clusters. There are several types of clustering techniques each with its own approach and application. The most commonly used clustering methods are: Partition based clustering, Hierarchical based clustering, Density based clustering and Distribution based clustering

Partition Based Clustering

The Partition-based clustering technique divides the dataset into a set of clusters where each data point belongs to one particular cluster. The technique aims to minimize a predefined cost function such as the sum of squared distances between data points and the cluster centre. The partition based clustering organizes the data into non-hierarchical clusters. These techniques are efficient but sensitive to initial conditions and outliers. This technique is also known as centroid based clustering because the identification and the assignment of data points to a cluster is driven by the centroid of a central point. These central point's represent the cluster, and the technique works by iteratively adjusting the centres to best fit the data. Figure 1.2 shows an example of partition-based clustering.

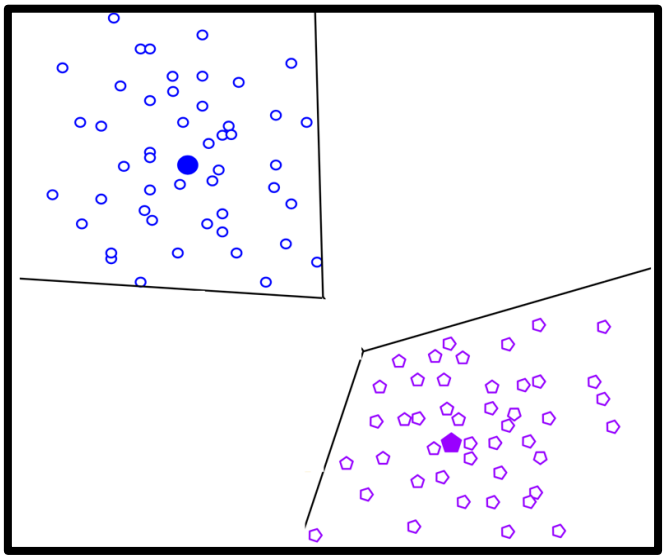


Figure 1.2: Example of partition based clustering

Advantages

- The technique is simple and easy to implement.
- The technique is efficient for large datasets.

Disadvantages

- Knowledge of the number of clusters ‘K’ is required in advance.
- The technique is sensitive to initial centroid or central point selection.
- The technique struggles if the clusters are non-spherical in shape or clusters vary in size and density.

Hierarchical Based Clustering

Hierarchical based clustering technique builds a tree like structure of the data set. The clusters at times here are nested inside other clusters. This method does not require the number of clusters to be predefined. It creates a dendrogram, a tree diagram that shows the arrangement of the clusters. Any number of clusters can be chosen by cutting the tree at the right level. There are two categories of hierarchical clustering, Agglomerative also known as bottom-up and Divisive also known as top-down.

Agglomerative hierarchical clustering

The agglomerative method starts with considering an individual data point as a cluster and then iteratively tries and merges the closest clusters until all points belong to a single cluster.

Divisive hierarchical clustering

The divisive method on the other hand starts with the consideration that the entire dataset is a single cluster. It then recursively tries and splits the single cluster into smaller clusters. Figure 1.3 shows an example of hierarchical based clustering.

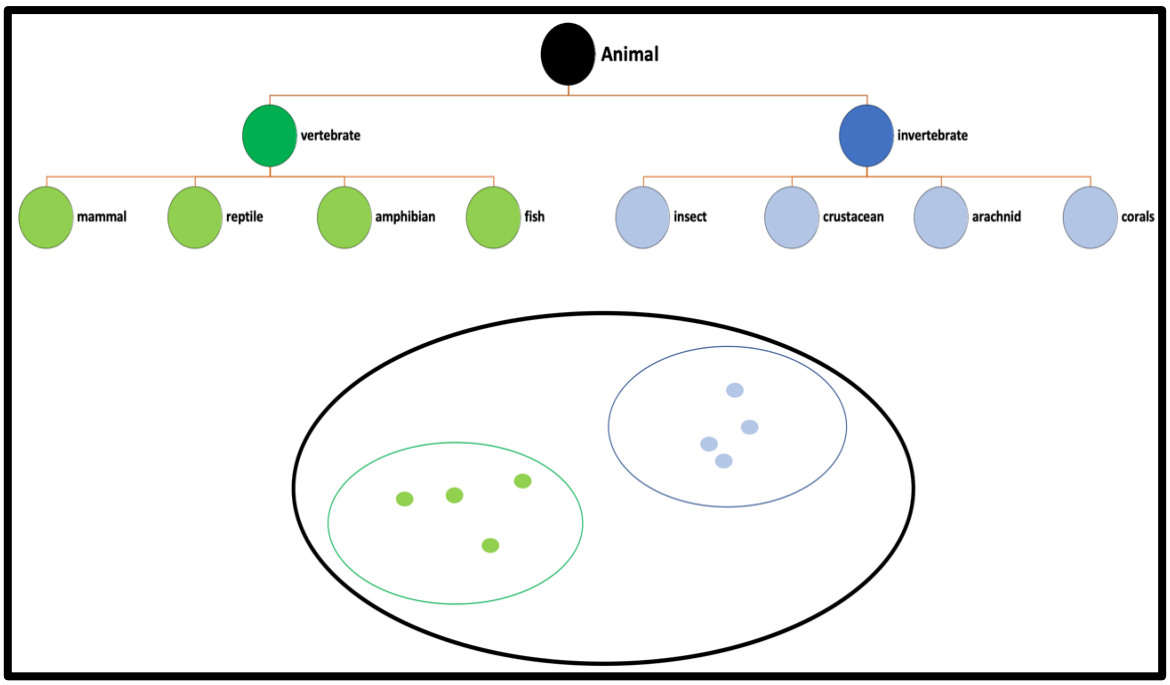


Figure 1.3: Example of hierarchical based clustering

Advantages

- Knowledge of the number of clusters 'K' is not required in advance.
- The use of a dendrogram makes visualisation very easy.
- The technique is more flexible and better for smaller datasets.

Disadvantages

- The technique is computationally expensive if the dataset is large.
- The technique can suffer from scalability issues at times.
- The results are sensitive to the linkage criteria (single, complete, or average linkage).

Density Based Clustering

Density based clustering techniques group together the data points that are closely packed. It marks all the data points that lie in low density regions as outliers. The outliers are not assigned to any clusters. This technique allows the discovery of any number of clusters irrespective of its shape. Figure 1.4 shows an example of density based clustering.

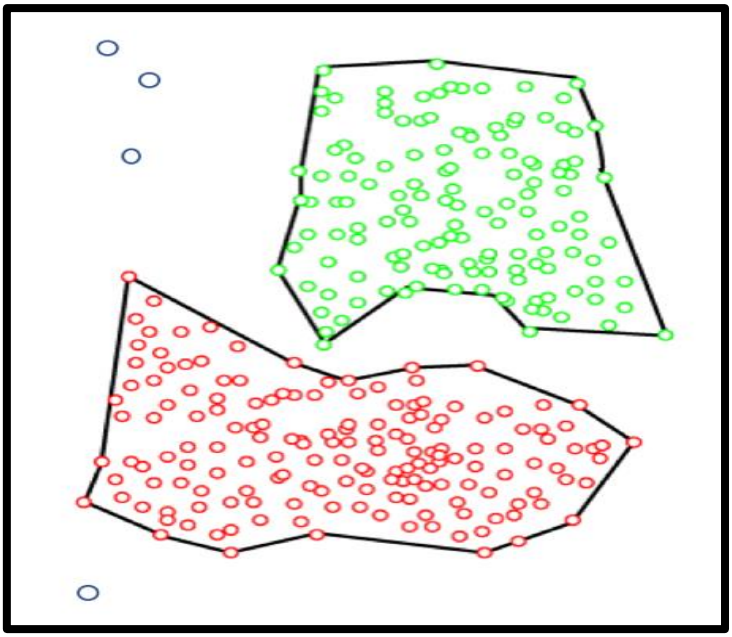


Figure 1.4: Example of density based clustering

Advantages

- This technique can be used to find clusters of arbitrary shapes.
- In this technique it is possible to identify outliers points or noise present in the dataset.
- Knowledge of the number of clusters 'K' is not required in advance.

Disadvantages

- The technique depends on two parameters; the radius of the neighbourhood known as epsilon (ε) and the minimum number of points to form a cluster.
- If the clusters have varying density then the technique struggles to generate proper clusters.

Distribution Based Clustering

The distribution model based clustering method divides the data based on the probability of how a dataset belongs to a particular distribution. The distribution is done by assuming some distributions (commonly Gaussian Distribution). Figure 1.5 shows an example of distribution based clustering.

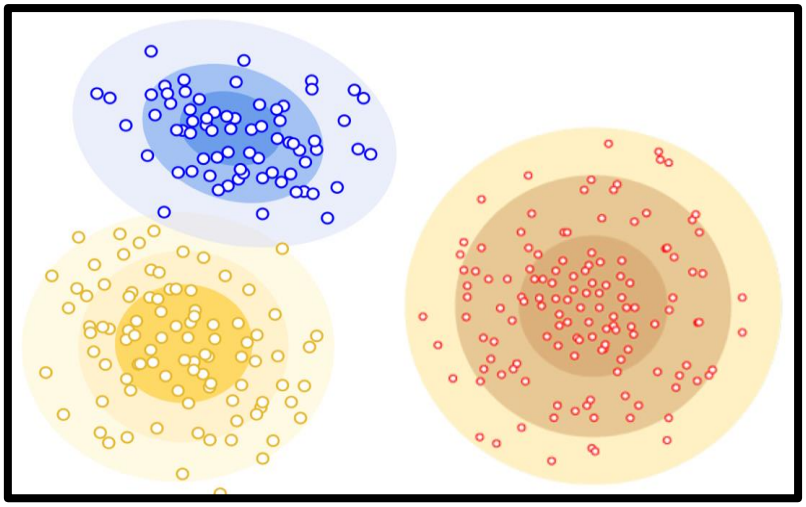


Figure 1.5: Example of distribution based clustering

Advantages

- The technique can be used to model clusters with elliptical shapes.
- A point can belong to more than one cluster with different probabilities.
- The technique can be used to handle overlapping clusters better.

Disadvantages

- The technique is computationally expensive.
- The technique assumes that the data follows a Gaussian distribution, which may not always be the case.
- The technique is sensitive to initialization.

Check Your Progress-2

a) In a hard clustering approach each data point can belong to more than one cluster. (True/False)

b) In partition based clustering the knowledge of the number of clusters 'K' is required in advance. (True/False)

c) Partition based clustering creates a dendrogram. (True/False)

d) The agglomerative method starts with consideration that the entire dataset is a single cluster. (True/False)

e) In density based clustering techniques all the data points that lie in low density regions are marked as outliers. (True/False)

f) The distribution based clustering technique can be used to model clusters with elliptical shapes. (True/False)

1.5 LET US SUM UP

In this unit we have discussed the concept of clustering and its approaches. Clustering is a process of dividing a data set into groups. As learnt we can apply two approaches for clustering hard and soft. You also got a detailed understanding of different techniques used for clustering like partition based clustering, hierarchical based clustering, density based clustering and distribution based clustering.

1.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a False
1-b True
1-c True
1-d False
1-e True
2-a False
2-b True
2-c False
2-d False
2-e True
2-f True

1.7 ASSIGNMENTS

- What is the advantage of using clustering?
- Explain the concept of data summarization with examples.
- Differentiate between hard and soft clustering approaches.
- Explain different types of hierarchical clustering approaches.
- Differentiate between density-based clustering and distribution-based clustering.

Unit-2: Clustering Algorithms 2

Unit Structure

2.0. Learning Objectives

2.1. Introduction

2.2. K-Means Clustering Technique

2.3. K-Medoids Clustering Technique

2.4. DBSCAN Clustering Technique

2.5. Let us sum up

2.6. Check your Progress: Possible Answers

2.7. Assignments

2.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand how to perform clustering in machine learning.
- Understand the working of K-Means clustering technique.
- Understand the working of K-Medoids clustering technique.
- Understand the working of DBSCAN clustering technique.
- Implement the K-Means, K-Medoids and DBSCAN algorithms.

2.1 INTRODUCTION

In the previous chapter we learnt that clustering is a technique used to arrange similar or dissimilar data points into distinct groups known as clusters based on some specific criteria. We also looked at different techniques that can be applied for clustering.

In this chapter we will learn about different clustering algorithms that are used under various techniques. We will see how the clustering is performed using these algorithms and will also look at the implementation of these algorithms.

2.2 K-MEANS CLUSTERING TECHNIQUE

K-Means clustering technique is a popular unsupervised machine learning algorithm used for clustering. It follows the partitioning approach for creating clusters. The goal of the K-Means clustering algorithm is to group similar data points together. The algorithm partitions a set of 'n' data points into 'K' clusters. Here 'K' defines the number of predefined clusters that need to be created. If K=2, then there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabelled dataset into k different clusters. Each data point belongs to the cluster with the nearest mean. It is also known as a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of the algorithm is to maximize the homogeneity within the clusters and thus to maximize the differences between the clusters. The homogeneity and differences are measured in terms of the distance between the objects or points in the data set. Thus the K-Means clustering algorithm helps users to organize data into groups where the

points in each group are more similar to each other than to those in other groups. It's often used in tasks like customer segmentation, document clustering, image compression and many more. The generic K-Means algorithm is as mentioned:

Generic K-Means algorithm

Step 1: Select 'k' points from the data points and mark them as initial centroids.

Step 2: Assign each data point to the nearest centroid to form 'k' clusters by measuring the distance of each point in the cluster from the centroid.

The distance between each data point and the centroids is typically calculated using the Euclidean distance formula:

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Here (x_1,y_1) is a data point and (x_2,y_2) is centroid.

Step 3: Identify the new centroid of each cluster on the basis of distance between points. The value is calculated by taking the mean of all the data points assigned within the cluster.

Step 4: Repeat Steps 2 and 3 to refine the clusters till centroids do not change.

Example:

Let us have a look at an example that uses the K-Means algorithm to form clusters. Assume that we have details of 10 customers along with features like Age and Order Total (The dataset has been kept small only for the purpose of demonstrating the working of the example). Our goal is to group these customers into 2 groups i.e. k=2 based on the two features. The data pertaining to the features for each customer is as shown in Table 2.1.

Table 2.1 Sample Order Data

Order ID	Age (x)	Order Total (y)
1	22	12000
2	29	25000
3	35	7500
4	42	18000
5	52	50000
6	60	30000

Order ID	Age (x)	Order Total (y)
7	30	12000
8	40	22000
9	25	15000
10	50	35000

Figure 2.1 shows the X-Y scatter plot of the data points for visualization purposes.

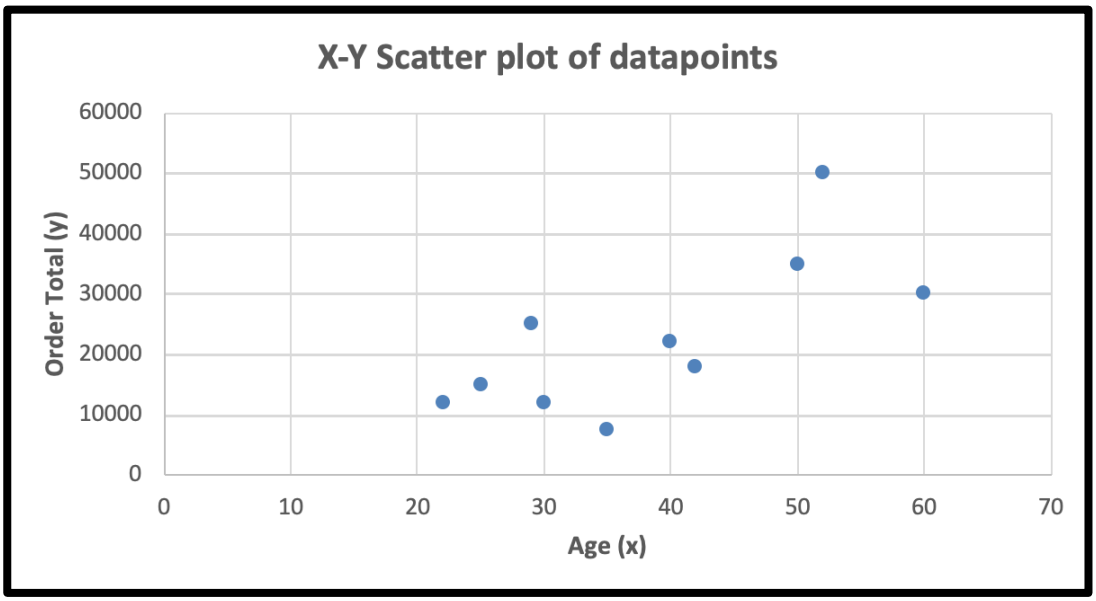


Figure 2.1: X-Y scatter plot of the data points in Table 2.1

Step-1: Initialize centroids:

The first step in the K-means clustering algorithm is to initialize the centroids. In the problem definition we have mentioned that k = 2, thus we need to randomly select two points as initial centroids. Assume that the centroids are as mentioned:

Centroid 1 (C1) is **(22, 12000)** and **Centroid 2 (C2)** is **(52, 50000)**

These centroids will be the starting point for our algorithm.

Step 2: Assign each data point to the nearest centroid by calculating the distance:

Now we will assign each data point to the nearest centroid by calculating the Euclidean distance between the data point and the centroid. Table 2.2 shows the distance calculation and cluster allocation.

Table 2.2 Distance calculation

Data Point (29, 25000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(29-22)^2 + (25000-12000)^2}$ $= \sqrt{(7)^2 + (13000)^2}$ $\cong 12987$	$= \sqrt{(29-52)^2 + (25000-50000)^2}$ $= \sqrt{(-23)^2 + (-25000)^2}$ $\cong 25001$	**C1**

Data Point (35, 7500)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(35-22)^2 + (7500-12000)^2}$ $= \sqrt{(13)^2 + (-4500)^2}$ $\cong 7124$	$= \sqrt{(35-52)^2 + (7500-50000)^2}$ $= \sqrt{(-17)^2 + (-42500)^2}$ $\cong 43315$	**C1**

Data Point (42, 18000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(42-22)^2 + (18000-12000)^2}$ $= \sqrt{(20)^2 + (6000)^2}$ $\cong 6000$	$= \sqrt{(42-52)^2 + (18000-50000)^2}$ $= \sqrt{(-10)^2 + (-32000)^2}$ $\cong 32344$	**C1**

Data Point (60, 30000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(60-22)^2 + (30000-12000)^2}$ $= \sqrt{(38)^2 + (18000)^2}$ $\cong 18000$	$= \sqrt{(60-52)^2 + (30000-50000)^2}$ $= \sqrt{(8)^2 + (-20000)^2}$	**C1**

Data Point (60, 30000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
	$\cong 20000$	

Data Point (30, 12000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(30-22)^2 + (12000-12000)^2}$ $= \sqrt{(8)^2 + (0)^2}$ $= 8$	$= \sqrt{(30-52)^2 + (12000-50000)^2}$ $= \sqrt{(-22)^2 + (-38000)^2}$ $\cong 38000$	**C1**

Data Point (40, 22000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(40-22)^2 + (22000-12000)^2}$ $= \sqrt{(18)^2 + (10000)^2}$ $\cong 10000$	$= \sqrt{(40-52)^2 + (22000-50000)^2}$ $= \sqrt{(-12)^2 + (-28000)^2}$ $\cong 28000$	**C1**

Data Point (25, 15000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(25-22)^2 + (15000-12000)^2}$ $= \sqrt{(3)^2 + (3000)^2}$ $\cong 3000$	$= \sqrt{(25-52)^2 + (15000-50000)^2}$ $= \sqrt{(-27)^2 + (-35000)^2}$ $\cong 35000$	**C1**

Data Point (50, 35000)		
Distance to C1 (22, 12000)	**Distance to C2 (52, 50000)**	**Closer to Cluster**
$= \sqrt{(50-22)^2 + (35000-12000)^2}$ $= \sqrt{(28)^2 + (23000)^2}$ $\cong 23000$	$= \sqrt{(50-52)^2 + (35000-50000)^2}$ $= \sqrt{(-2)^2 + (-15000)^2}$ $\cong 15000$	**C2**

The two clusters thus have following data points as per the distance calculation:
Cluster 1 = { (22, 12000), (29,25000), (35,7500), (42,18000), (60,30000), (30,12000), (40,22000), (25,15000) }
Cluster 2 = { (50,35000), (52,50000) }

Step 3: Update the Centroids:

We will now compute the new centroids based on the mean of the points assigned to each centroid in the above step. The centroid C1 is assigned data points (29,25000), (35,7500), (42,18000), (60,30000), (30,12000), (40,22000) and (25,15000), thus new centroid will be calculated as follows:

$$New\ x = \frac{22 + 29 + 35 + 42 + 60 + 30 + 40 + 25}{8} = \frac{213}{8} = 26.62 \cong 27$$

Similarly

$$New\ y = \frac{12000 + 25000 + 7500 + 18000 + 30000 + 12000 + 22000 + 15000}{8}$$
$$= \frac{141500}{8} \cong 17688$$

Thus the value of the new centroid C1 is **(27, 17688)**.

Similarly new centroid C2 will be calculated as follows:

$$New\ x = \frac{50 + 52}{2} = \frac{102}{2} = 51$$

And

$$New\ y = \frac{35000 + 50000}{2} = \frac{85000}{2} \cong 42500$$

Thus the value of the new centroid C2 is **(52, 42500)**.

Step 4: Repeat Step 2 – Iteration 2

Table 2.3 shows the distance calculation with new centroids and cluster allocation. The detailed calculation and intermediate steps have been omitted. The reader can evaluate it further for the purpose of better understanding.

Table 2.3 Distance calculation

Data Point No.	Age (x)	Order Total (y)	Distance to C1 (27, 17688)	Distance to C2 (52, 42500)	Closer to Cluster
1	22	12000	5688	30500	**C1**
2	29	25000	7312	17500	**C1**
3	35	7500	10188	35000	**C1**
4	42	18000	312	24500	**C1**
5	52	50000	32312	7500	**C2**
6	60	30000	12312	12500	**C1**
7	30	12000	5688	30500	**C1**
8	40	22000	4312	20500	**C1**
9	25	15000	2688	27500	**C1**
10	50	35000	17312	7500	**C2**

The two clusters thus have following data points as per the distance calculation:
Cluster 1 = { (22, 12000), (29,25000), (35,7500), (42,18000), (60,30000), (30,12000), (40,22000), (25,15000) }
Cluster 2 = { (50,35000), (52,50000) }

As can be seen from the output of Table 2.2 and 2.3 the data points in both clusters have remained unchanged between the two iterations. Thus the final cluster assignment would be done as mentioned:

Cluster 1 = { (22, 12000), (29,25000), (35,7500), (42,18000), (60,30000), (30,12000), (40,22000), (25,15000) }
Cluster 2 = { (50,35000), (52,50000) }

Figure 2.2 shows the X-Y scatter plot along with the distribution of the data points in two clusters.

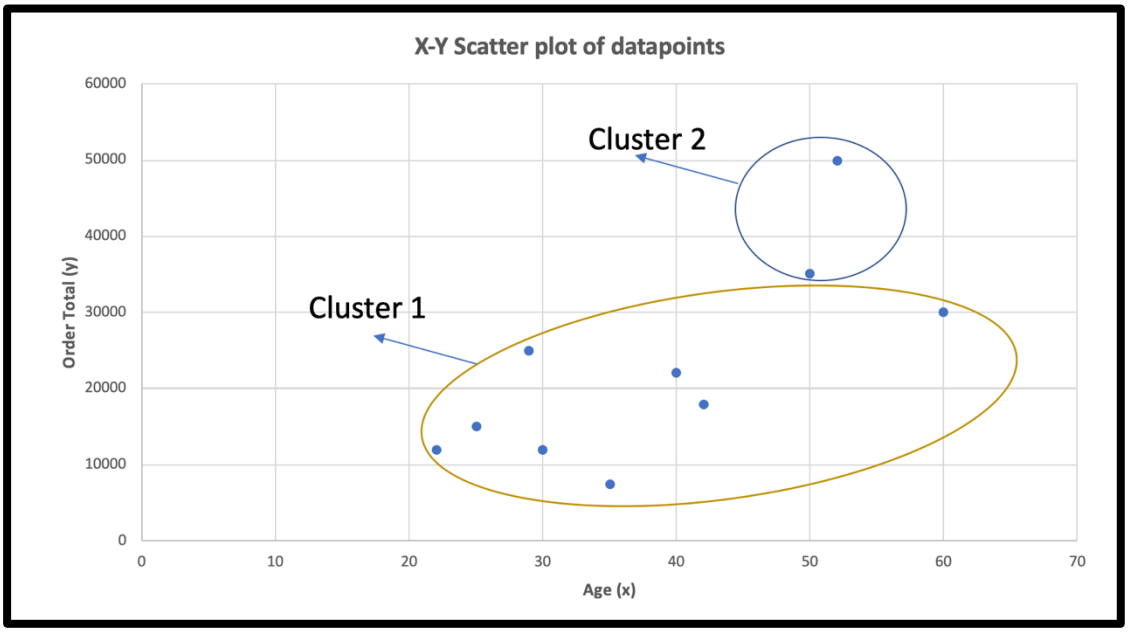


Figure 2.2: X-Y scatter plot of the data points in two clusters

Check Your Progress-1

a) K-means is an unsupervised machine learning algorithm. (True/False)
b) The value of K in the K-Means algorithm is by default 2. (True/False)
c) In the K-Means algorithm each data point belongs to the cluster with the nearest mean. (True/False)
d) Each cluster in K-Means algorithm is associated with a medoid. (True/False)
e) There are multiple ways to calculate distance between centroid and data points. (True/False)

2.3 K-MEDIODS CLUSTERING TECHNIQUE

K-Medoids clustering technique is another popular unsupervised machine learning algorithm used for clustering. The goal of K-Medoids clustering algorithm is to group similar data points together such that a single data point can belong to only one cluster and each cluster has a minimum one data point. The method is also known as partitioning around medoids (PAM). Similar to the K-Means clustering technique here also 'K' defines the number of predefined clusters that need to be created.

Here the term medoid refers to a data point in the cluster within a dataset from which the sum of distances to other data points is minimal. It is the data point in a cluster characterized by the lowest dissimilarity with other data points.. The generic K-Medoids algorithm is as mentioned:

Generic K-Medoids algorithm

Step 1: Select 'k' random points from the data points and mark them as initial medoids.

Step 2: Calculate the distance between the initial medoids and other data points (non-medoid) and assign the non-medoid points to the cluster to which its distance to the medoid point is minimum.

The distance between the data point and the medoids is typically calculated using metrics like Euclidean distance, or Manhattan distance or any other distance metric. Here we are will use the Manhattan distance, the formula of the same is:

$$d = |x_2 - x_1| + |y_2 - y_1|$$

Here (x_1,y_1) is a data point and (x_2,y_2) is medoid.

Step 3: Calculate the total cost as the sum of distances from other data points to the medoid point within a cluster.

Step 4: Select a new non-medoid data point (q) and swap it with the initial medoids (p).

Step 5: Repeat the steps from 2 to 4 if $cost_q < cost_p$. If $cost_q >= cost_p$ then revert to previous medoids and finalize the cluster.

Example:

Let us have a look at an example that uses the K-Medoids algorithm to form clusters. Assume that we have details of 10 students along with features like Height and Weight. Our goal is to group these students into 2 groups i.e. K = 2 based on the given two features. The data pertaining to the features for each person is as shown in Table 2.4.

Table 2.4 Sample Data of students

Sr. No.	Height in foot (x)	Weight in KG (y)
1	5.4	85
2	4.7	70
3	6	75
4	5.7	65
5	4.3	50
6	3	35
7	5.2	70
8	3.5	40
9	4.5	45
10	3.7	50

Figure 2.3 shows the X-Y scatter plot of the data points for visualization purposes.

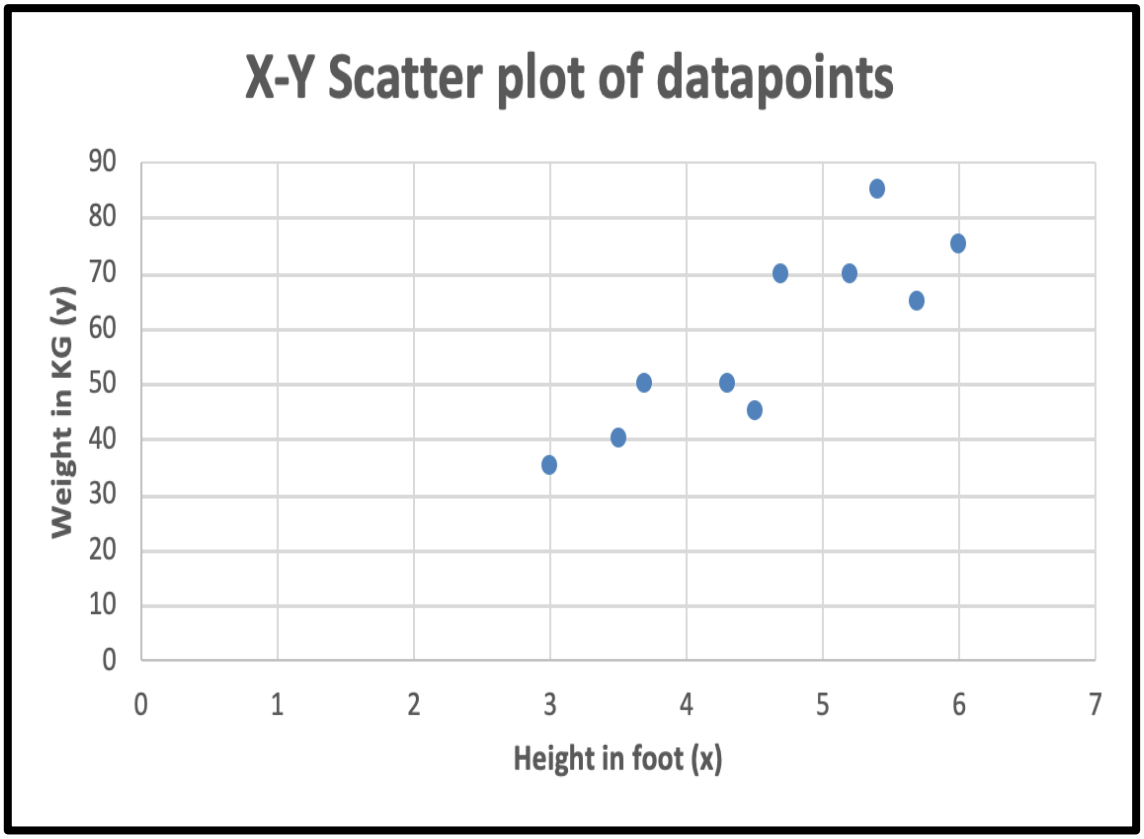


Figure 2.3: X-Y scatter plot of the data points in Table 2.4

Step-1: Initialize medoids

The first step in K-medoids clustering algorithm is to initialize random medoids. In the problem definition we have mentioned that k = 2, thus we need to randomly select two data points as initial medoids. Assume that the medoids are as mentioned:

Medoid 1 (M1) is **(5.4, 85)** and **Medoid 2 (M2)** is **(4.3, 50)**

These medoids will be the starting point for our algorithm.

Step 2: Calculate the distance between the initial medoids and other data points: Now we will assign each data point to the nearest medoid by calculating the Manhattan distance between the data point and the medoid. Table 2.5 shows the distance calculation.

Table 2.5 Distance calculation

Data Point No.	Height in foot (x)	Weight in KG (y)	Distance to M1 (5.4, 85)	Distance to M2 (4.3, 50)
1	5.4	85	-	-
2	4.7	70	= |5.4 – 4.7| + |85 – 70| = |0.7| + |15| = 15.7	= |4.3 – 4.7| + |50 – 70| = |-0.4| + |-20| = 20.4
3	6	75	= |5.4 – 6| + |85 – 75| = |-0.6| + |10| = 10.6	= |4.3 – 6| + |50 – 75| = |-1.7| + |-25| = 26.7
4	5.7	65	= |5.4 – 5.7| + |85 – 65| = |-0.3| + |20| = 20.3	= |4.3 – 5.7| + |50 – 65| = |-1.4| + |-15| = 16.4
5	4.3	50	-	-

Data Point No.	Height in foot (x)	Weight in KG (y)	Distance to M1 (5.4, 85)	Distance to M2 (4.3, 50)
6	3	35	= |5.4 – 3| + |85 – 35| = |2.4| + |50| = 52.4	= |4.3 – 3| + |50 – 35| = |1.3| + |15| = 16.3
7	5.2	70	= |5.4 – 5.2| + |85 – 70| = |0.2| + |15| = 15.2	= |4.3 – 5.2| + |50 – 70| = |-0.9| + |-20| = 20.9
8	3.5	40	= |5.4 – 3.5| + |85 – 40| = |1.9| + |45| = 46.9	= |4.3 – 3.5| + |50 – 40| = |0.8| + |10| = 10.8
9	4.5	45	= |5.4 – 4.5| + |85 – 45| = |0.9| + |40| = 40.9	= |4.3 – 4.5| + |50 – 45| = |-0.2| + |15| = 5.2
10	3.7	50	= |5.4 – 3.7| + |85 – 50| = |1.7| + |35| = 36.7	= |4.3 – 3.7| + |50 – 50| = |0.6| + |0| = 0.6

Thus as per the distance calculation data points 2, 3 and 7 are assigned to cluster C1 with medoid (5.4, 85), while data points 4, 6, 8, 9 and 10 are assigned to cluster C2 with medoid (4.3, 50)

Step 3: Calculate the total cost

The total cost Co1 after the first iteration is as mentioned:

Co1 = (15.7 + 10.6 + 15.2) + (16.5 + 16.3 + 10.8 + 5.2 + 0.6) = 90.8

Step 4: Select a new non-medoid data points (q)

Let us now randomly select another initial medoids and swap it with the current ones. Assume that the new medoids are as mentioned:

Medoid 1 (M1) is **(6, 75)** and **Medoid 2 (M2)** is **(3.5, 40)**

Step 5: Repeat Step 2 - Iteration 2

Table 2.6 shows the distance calculation with new medoids. The detailed calculation and intermediate steps have been omitted. The reader can evaluate it further for the purpose of better understanding.

Table 2.6 Distance calculation

Data Point No.	Height in foot (x)	Weight in KG (y)	Distance to M1 (6, 75)	Distance to M2 (3.5, 40)
1	5.4	85	10.6	46.9
2	4.7	70	6.3	31.2
3	6	75	-	-
4	5.7	65	10.3	27.2
5	4.3	50	26.7	10.8
6	3	35	43	5.5
7	5.2	70	5.8	31.7
8	3.5	40	-	-
9	4.5	45	31.5	6
10	3.7	50	27.3	10.2

Thus as per the distance calculation data points 1, 2, 4 and 7 are assigned to cluster C1 with medoid (6, 75), while data points 5, 6, 9 and 10 are assigned to cluster C2 with medoid (3.5, 40)

Step 5: Repeat Step 3 - Iteration 2

The total cost Co2 after the second iteration is as mentioned:

Co2 = (10.6 + 6.3 + 10.3 + 5.8) + (10.8 + 5.5 + 6 + 10.2) = 65.5

As the total cost in iteration 2 is less than the total cost of iteration 1, we will randomly select the new medoids.

Step 5: Repeat Step 4 - Iteration 2

Let us now randomly select another initial medoids and swap it with the current ones. Assume that the new medoids are as mentioned:

Medoid 1 (M1) is **(5.7, 65)** and **Medoid 2 (M2)** is **(3.7, 50)**

Step 6: Repeat Step 2 - Iteration 3

The Table 2.7 shows the distance calculation with new medoids. The detailed calculation and intermediate steps have been omitted in the calculation made

Table 2.7 Distance calculation

Data Point No.	Height in foot (x)	Weight in KG (y)	Distance to M1 (5.7, 65)	Distance to M2 (3.7, 50)
1	5.4	85	20.3	36.7
2	4.7	70	6	21
3	6	75	10.3	27.3
4	5.7	65	-	-
5	4.3	50	16.4	0.6
6	3	35	32.7	15.7
7	5.2	70	5.5	21.5
8	3.5	40	27.2	10.2
9	4.5	45	21.2	5.8
10	3.7	50	-	-

Thus as per the distance calculation data points 1, 2, 3 and 7 are assigned to cluster C1 with medoid (5.7, 65), while data points 5, 6, 8 and 9 are assigned to cluster C2 with medoid (3.7, 50)

Step 6: Repeat Step 3 - Iteration 3

The total cost Co3 after the third iteration is as mentioned:

Co3 = (20.3 + 6 + 10.3 + 5.5) + (0.6 + 15.7 + 10.2 + 5.8) = 74.4

As the total cost in iteration 3 is greater than the total cost of iteration 2 the process converges. The final clusters thus will be formed with medoids (6, 75) and (3.5, 40).

Thus the final cluster assignment would be done as mentioned:
Cluster 1 = { (5.4,85), (4.7,70), (6,75), (5.7,65), (5.2,70) } and Cluster C2 = { (4.3,50), (3,35), (3.5,40), (4.5,45), (3.7,50) }.

Figure 2.4 shows the X-Y scatter plot along with the distribution of the data points in two clusters.

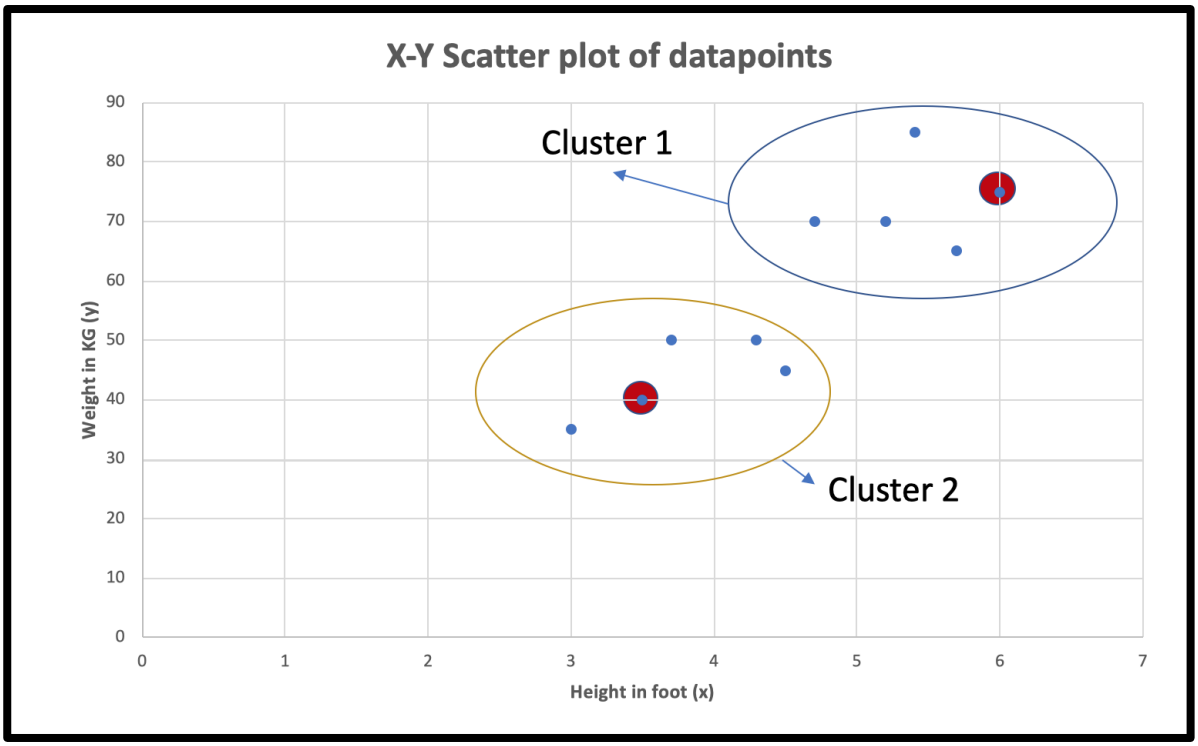


Figure 2.4: X-Y scatter plot of the data points in two clusters

Check Your Progress-2

a) In K-Medoids algorithm a data point of a cluster can belong to two clusters. (True/False)
b) The value of K in the K-Medoids algorithm is to be predefined by the user. (True/False)
c) Each cluster formed by the K-Medoid algorithm is associated with a medoid. (True/False)
d) The K-Medoids algorithm is not good at identifying outliers as compared to K-Means algorithm because it uses the mean of the data points in a cluster as the center. (True/False)
e) K-Medoids algorithm may not converge to a global optimum due to the initial choice of medoids. (True/False)

2.4 DBSCAN CLUSTERING TECHNIQUE

Density-Based Spatial Clustering of Applications with Noise usually abbreviated as DBSCAN as the name suggests is a density based clustering algorithm. The DBSCAN algorithm works on the principle of density of data points in a region. The algorithm thus groups data points that are closely packed together in data space. It works by defining clusters as dense regions separated by regions of lower density. Using this approach the DBSCAN algorithm can be used to discover clusters of arbitrary shapes. It is also an excellent algorithm when it comes to identifying outliers as noise in a data set. There is no need to predefine the number of clusters when using the DBSCAN algorithm.

The two main parameters used in the DBSCAN algorithm are ε (epsilon) and MinPts. The term ε (epsilon) refers to the maximum distance between two data points for them to be considered as neighbours. MinPts refers to the minimum number of points that are required to form a dense region or a cluster.

The data points when using the DBSCAN algorithm are classified as core, border or noise points. The core points are data points that have at least a minimum number of

other points (MinPts) within a specified distance (ε or epsilon). The border points are points that are within the ε distance of a core point but don't have MinPts neighbours themselves. The noise points are points that are neither core points nor border points. These data points are not close enough to any of the clusters that are formed and hence cannot be included in them. Figure 2.5 gives the idea of the concepts mentioned.

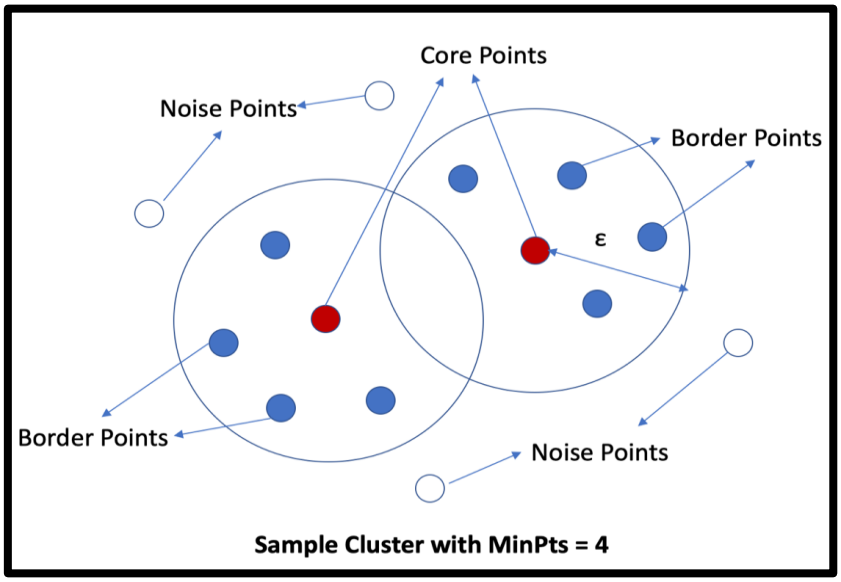


Figure 2.5: Basic Concepts of DBSCAN

Generic DBSCAN algorithm

Step 1: Select parameters ε (epsilon) and MinPts.

Step 2: Select random data point as start point.

Step 3: Examine the neighbourhood by calculating the distance matrix of all data points.

- Retrieve all points within the ε distance of the starting point.
- If the number of neighbouring points is less than MinPts, the point is labelled as noise (for now).
- If there are at least MinPts points within ε distance, the data point is marked as a core point, and a new cluster is formed.

Step 4: Expand the Cluster, all the neighbours of the core point are added to the cluster. For each of these neighbours:

- If it's a core point, its neighbours are added to the cluster recursively.
- If it's not a core point, it's marked as a border point, and the expansion stops.

Step 5: Repeat steps 3 and 4 until all data points have been visited.

Step 6: Finalize the clusters, if all data points have been visited. Points initially labelled as noise might now be border points if they're within ε distance of a core point.

Step 7: Handle Noise, any data points if it does not belong to a cluster is classified as noise.

Example:

Assume that we have a data set that consists of 10 points { (1, 2), (2, 2), (2, 3), (4,7), (5,6), (5,8), (6,7), (8, 7), (8, 8), (25, 20) }. We need to partition them into multiple clusters using the DBSCAN algorithm. Let the value of ε (epsilon) be 2 and MinPts be 3. Figure 2.6 shows the scatter plot of the same.

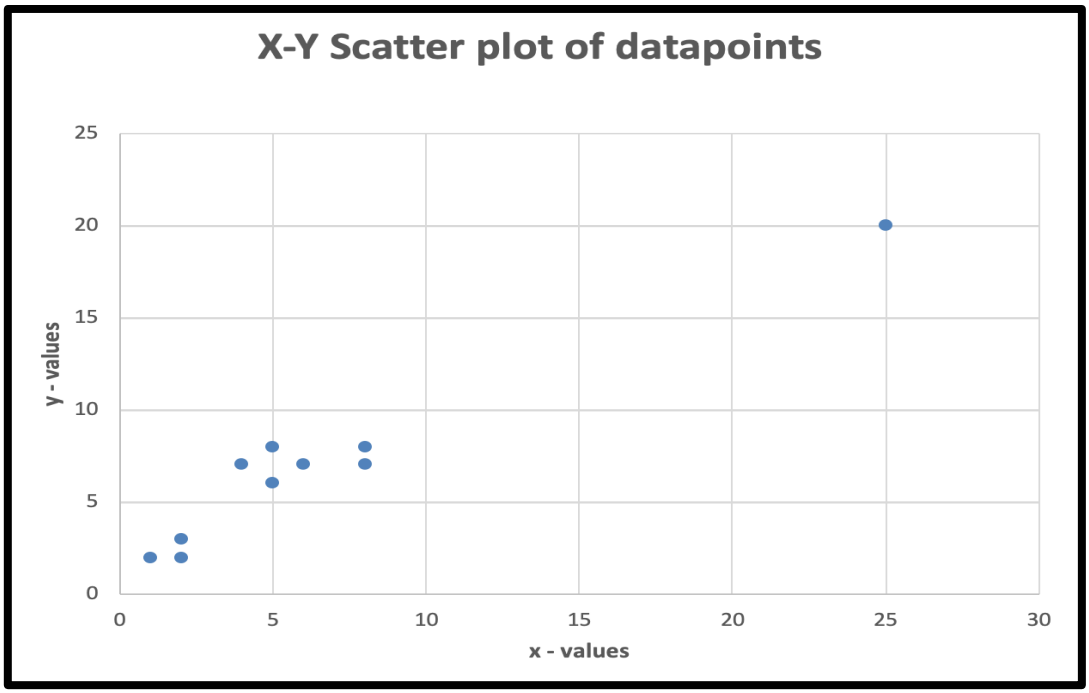


Figure 2.6: X-Y Scatter plot of data

Step 1: Given ε = 2 and MinPts = 3

Step 2: Let us start with the data point (1,2)

Step 3: Calculate pairwise distance

We will use the Manhattan distance formula to calculate the distance between any two points i.e. $d = |x_2 – x_1| + |y_2 – y_1|$

The calculation of pairwise distance from the first data point (1,2) is shown for the reference in Table 2.8.

Table 2.8 Distance calculation

Data Point	Distance to (1, 2)	Less than ε = 2
(2, 2)	= |1 – 2| + |2 – 2| = |-1| + | 0 | = 1	Y
(2, 3)	= |1 – 2| + |2 – 3| = |-1| + | -1 | = 2	Y
(4,7)	= |1 – 4| + |2 – 7| = |-3| + | -5 | = 8	N
(5,6)	= |1 – 5| + |2 – 6| = |-4| + | -4 | = 8	N
(5,8)	= |1 – 5| + |2 – 8| = |-4| + | -6 | = 10	N
(6,7)	= |1 – 6| + |2 – 7| = |-5| + | -5 | = 10	N
(8, 7)	= |1 – 8| + |2 – 7| = |-7| + |-5| = 12	N
(8, 8)	= |1 – 8| + |2 – 8| = |-7| + |-6| = 13	N
(25, 20)	= |1 – 25| + |2 – 20| = |-24| + |-18| = 42	N

Table 2.9 shows the distance matrix for all the data points calculated using Manhattan distance formula.

Table 2.9 Distance Matrix of all data points

	1,2	2,2	2,3	4,7	5,6	5,8	6,7	8,7	8,8	25,20
1,2	0	1	2	8	8	10	10	12	13	42
2,2	1	0	1	7	7	9	9	11	12	41
2,3	2	1	0	6	6	8	8	10	11	40
4,7	8	7	6	0	2	2	2	4	5	34
5,6	8	7	6	2	0	2	2	4	5	34
5,8	10	9	8	2	2	0	2	4	3	32
6,7	10	9	8	2	2	2	0	2	3	32
8,7	12	11	10	4	4	4	2	0	1	30
8,8	13	12	11	5	5	3	3	1	0	29
25,20	42	41	40	34	34	32	32	30	29	0

The observation of the distance calculation in Table 2.9 leads us to the outcome given in Table 2.10.

Table 2.10 Deciding Core Points

Data Point	Data Points having ε = 2	No. of Neighbours	Core Point
(1, 2)	(2,2) (2,3)	2	N
(2, 2)	(1,2) (2,3)	2	N
(2, 3)	(1,2) (2,2)	2	N
(4,7)	(5,6) (5,8) (6,7)	3	Y
(5,6)	(4,7) (5,8) (6,7)	3	Y
(5,8)	(4,7) (5,6) (6,7)	3	Y
(6,7)	(4,7) (5,6) (5,8) (8,7)	4	Y
(8, 7)	(6,7) (8,8)	2	N
(8, 8)	(8,7)	1	N
(25, 20)	-	0	N

From Table 2.10 we can say that only four data points (4,7), (5,6), (5,8) and (6,7) have a number of neighbours greater than or equal to Minpts, thus these four data points become our core points.

As the core points have been identified we can start forming clusters by checking the neighborhood of core points. The clusters are expanded by including the data points that can be directly reached i.e. data points within ε distance. The non-core points that are within ε distance of any core point are added to the cluster, but non-core points that are not reachable are marked as noise.

Let the formation of Cluster 1 start with data point (4,7), we now add core points (5,6), (5,8) and (6,7) to Cluster 1. Thus data points that represent Cluster 1 are {(4, 7), (5, 6), (5, 8), (6, 7)}. As data points (1, 2), (2, 2), (2, 3), (8, 7), (8, 8), and (25, 20) are not core points and are not reachable by any core points within ε = 2 they are considered as noise.

As all data points have been taken into consideration we get the final result as mentioned:

Final Clustering Result:

Cluster 1: {(4, 7), (5, 6), (5, 8), (6, 7)}

Noise: {(1, 2), (2, 2), (2, 3), (8, 7), (8, 8), (25, 20)}

Figure 2.7 shows the outcome of the DBSCAN algorithm.

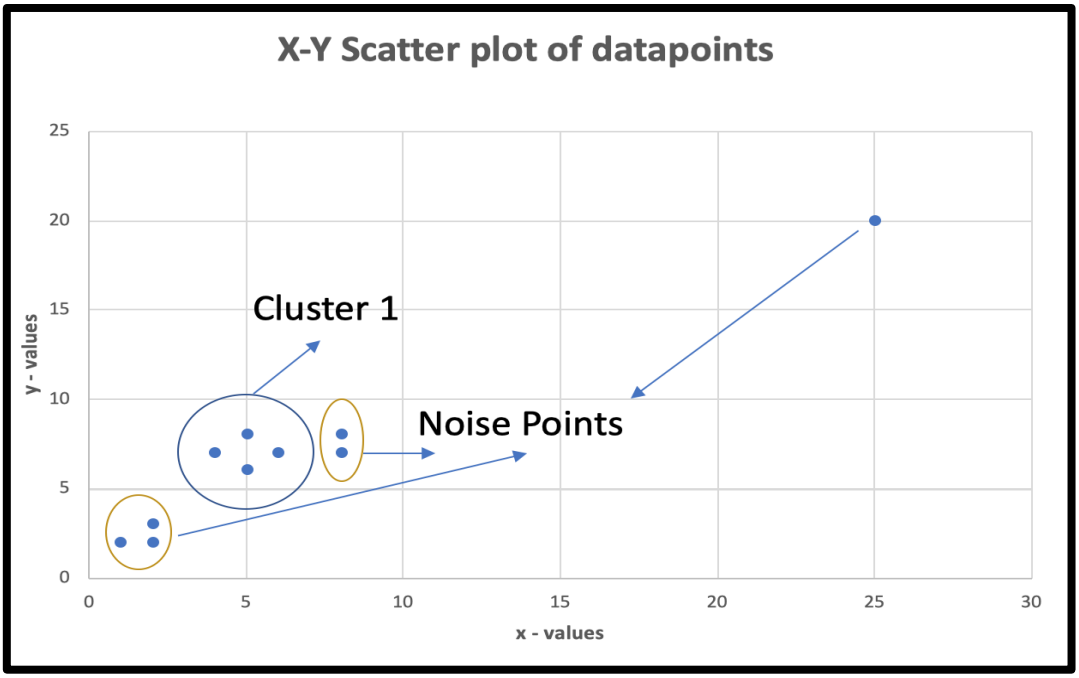


Figure 2.7: Outcome of the DBSCAN algorithm

Check Your Progress-3

a) There is no need to specify the number of clusters in advance when using the DBSCAN algorithm. (True/False)

b) The DBSCAN algorithm cannot be used to identify clusters of arbitrary shapes. (True/False)

c) The DBSCAN algorithm is sensitive to the choice of the ε that defines the maximum distance between two points to be considered neighbours. (True/False)

d) The data points classified as noise in the DBSCAN algorithm are assigned to its nearest cluster. (True/False)

e) The DBSCAN algorithm is computationally expensive when working with large datasets and high-dimensional data. (True/False)

2.5 LET US SUM UP

In this unit we have discussed the three algorithms used for clustering. We understood the generic algorithms of K-Means, K-Medoid and DBSCAN clustering algorithms. We also looked at an example and performed mathematical calculation of each of these methods.

2.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a True
1-b False
1-c True
1-d False
1-e True
2-a False
2-b True
2-c True
2-d False
2-e True
3-a True
3-b False
3-c True
3-d False
3-e True

2.7 ASSIGNMENTS

- Explain the general steps involved in the K-Means algorithm.
- What is the primary difference between K-Means and K-Medoids algorithm?
- What is the significance of the ε (epsilon) and MinPts parameters in the DBSCAN algorithm?
- Explain the concept of core points, border points, and noise.

Unit-3: Introduction to Association Rules

3

Unit Structure

3.0. Learning Objectives

3.1. Introduction

3.2. Key Components of Association Rules

3.3. Rule Evaluation Metrics

3.4. How Does Association Rule Learning Work?

3.5. Types of Association Rule Learning

3.6. Applications of Association Rules

3.7. Example of Association Rules

3.8. Let us sum up

3.9. Check your Progress: Possible Answers

3.10. Assignments

3.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the concept of Association Rule Learning.
- Understand the purpose of pattern search.
- Explore real-world applications of Association Rule Learning.

3.1 INTRODUCTION

Association rule mining or learning is a method to identify patterns in large data sets, which determines the correlations between variables in the data and uses those correlations to predict or make a decision. Association rule mining identifies patterns that interpret the relationships among different pieces of data collection.

For instance, an associated data set containing transactions based on grocery stores may be used to identify relationships between those products which are often bought together through association rule mining. An association rule that could be extracted from this data set is "if a customer buys bread, they are also likely to buy milk." We may use these rules to aid our decisions on marketing strategies, store design, and product placement.

3.2 KEY COMPONENTS OF ASSOCIATION RULES

Association rules focus on identifying strong associations between different items or variables in the data. It presents these associations in the form of if-then rules. An association rule consists of an antecedent (if part) and a consequent (then part). The dataset contains an antecedent, and we derive a consequent by using the antecedent.

Antecedent: The “if” part of the rule, representing the condition.

Example: a customer buys bread and butter

Consequent: The “then” part of the rule, representing the outcome.

Example: The customer also buy milk

As can be seen a typical association rule in a market basket analysis might state that if a customer buys bread and butter (X), they are likely to also buy milk (Y).

3.3 RULE EVALUATION METRICS

Association rules are evaluated using key metrics that determine their relevance, strength, and reliability. These metrics include support, confidence, and lift, which quantify the frequency and strength of relationships between data items.

Support: It refers to the frequency of a data set (one or more items) that appears in all the considered transactions. Generally it looks for how often the given data or combination appears in the given data set. Mathematically, support of an item 'X' is defined as:

$$Support(X) = \frac{Number\ of\ Transactions\ containing\ X}{Total\ number\ of\ Transactions}$$

If we obtain a high value for support it indicates that an item or itemset is common in the dataset, while low support value indicates that it is rare.

Confidence: Confidence is defined as the likelihood of obtaining item 'y' along with an item 'x'. Mathematically, it is defined as the ratio of frequency of transactions containing items 'x' and 'y' to the frequency of transactions that contained item x.

$$Confidence(X \rightarrow Y) = \frac{Number\ of\ Transactions\ containing\ X\ and\ Y}{Number\ of\ Transactions\ containing\ X}$$

In terms of support, confidence can be described as:

$$Confidence(X \rightarrow Y) = \frac{support(\ X\ \cup\ Y\ )}{support\ \ X}$$

If the value of confidence is high, it indicates that the presence of the first item is a strong predictor of the presence of the second item.

Lift: Lift is a measure of the strength of the association between two items, taking into account the frequency of both items in the dataset.

It is calculated as the confidence of the association divided by the support of the second item. Lift is used to compare the strength of the association between two items to the expected strength of the association if the items were independent.

$$Lift(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{Number\ of\ Transactions\ containing\ X}$$

A lift value greater than 1 indicates that the association between two items is stronger than expected based on the frequency of the individual items. This suggests that the association may be meaningful and worth further investigation. A lift value less than 1 indicates that the association is weaker than expected and may be less reliable or less significant.

Check Your Progress-1

a) An itemset whose support is greater than or equal to a minimum support threshold is _____.

b) Support (A) means Total number of transactions containing A. (True/False)

c) Frequency of occurrence of an itemset is called as _____.

d) An association rule consists of an _____(if part) and a _____(then part).

e) High confidence indicates that the presence of the first item is a _____of the presence of the second item.

f) A lift value less than 1 indicates that the association is weaker than expected and may be less reliable or less significant. (True/False)

3.4 HOW DOES ASSOCIATION RULE LEARNING WORK?

Association rule learning is a multi-step process designed to identify meaningful patterns and relationships in large datasets. It involves the steps as shown in figure 3.1:

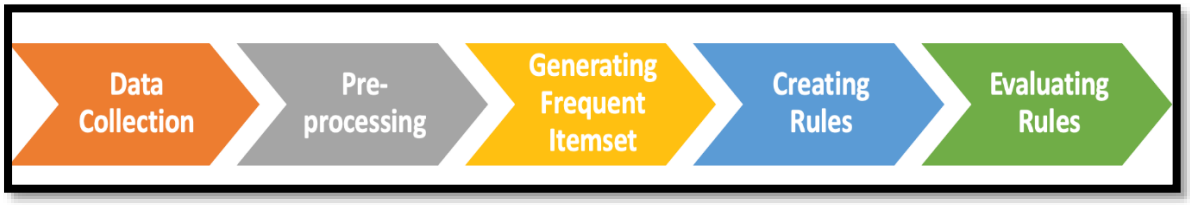


Figure 3.1: Steps to learn association rule

Let us have a look at each of these steps in brief.

Data Collection: The process starts with gathering a dataset that contains a list of transactions. Each transaction includes a set of items.

Pre-processing: Data is cleaned and transformed into a suitable format for analysis. This may involve removing missing values, duplicates, or irrelevant data.

Generating Frequent Itemset: The next step is to find combinations of items that frequently occur together. This is done using algorithms like Apriori, FP-Growth, or Eclat.

Creating Rules: Once frequent itemset are identified, rules are generated. Each rule is in the form of "If A, then B" (A → B), indicating that if item A appears, item B is likely to appear as well.

Evaluating Rules: The generated rules are evaluated based on support, confidence, and lift to find the most meaningful associations.

3.5 TYPES OF ASSOCIATION RULE LEARNING

Association rule learning can be classified into different types based on the nature of the rules and the data being analysed.

Multi-relational Association Rules

Multi-relational association rules involve finding patterns across multiple related datasets or tables. Unlike traditional association rules that work with a single dataset, these rules integrate data from different sources to uncover more complex relationships.

Example:

In a university setting, a rule might link students' academic performance (grades) with extracurricular involvement (clubs) and demographic data (age group), revealing patterns across different types of information.

Generalised Association Rules

Generalised association rules aim to identify patterns at different levels of a data hierarchy. This type of rule takes into account relationships not just at a specific item level but also across broader categories or groups.

Example:

In retail, a generalised rule might be "If a customer buys dairy products, then they also tend to buy baked goods." This rule covers broader categories (dairy and baked goods) rather than specific products (milk and bread).

Quantitative Association Rules

Quantitative association rules focus on numerical attributes and analyse patterns based on the quantity or range of values, rather than just the presence or absence of items. These rules can capture more detailed relationships in the data.

Example:

A rule such as “If a customer buys more than 5 items, then they are likely to spend over Rs. 500” is a quantitative association. This helps in identifying trends based on quantities rather than item pairs.

Interval Information Association Rules

Interval information association rules take into account the data that falls within specific ranges. These rules help identify patterns where the relationships depend on certain intervals or thresholds.

Example:

In healthcare, an interval rule might state, “If a patient’s blood pressure is between 120 and 140, then there is a higher likelihood of prescribing medication A.” The rule works with ranges rather than specific values, making it suitable for continuous data.

These different types of association rule learning expand the scope of pattern discovery, enabling more nuanced and actionable insights from data.

3.6 APPLICATIONS OF ASSOCIATION RULES

Association rule mining is commonly used in a variety of applications, some common application areas are as mentioned:

Market Basket Analysis

One of the most well-known applications of association rule mining is in market basket analysis. This involves analyzing the items customers purchase together to understand their purchasing habits and preferences.

Example:

A supermarket discovers that customers who buy bread often purchase butter and jam, leading to strategic placement of these items together.

Recommendation systems

Association rule mining can be used to suggest items that a customer might be interested in based on their past purchases or browsing history.

Example:

A music streaming service might use association rule mining to recommend new artists or albums to a user based on their listening history. Another example is if a user watches several sci-fi movies, the system may recommend other sci-fi titles.

Fraud Detection

Association rule mining can be used to detect fraudulent activity in the finance sector.

Example:

A credit card company might use association rule mining to identify patterns of fraudulent transactions, such as multiple purchases from the same merchant within a short period of time.

Healthcare

In healthcare, association rules help discover co-occurrence patterns in symptoms, aiding in diagnostic processes and treatment plans.

Example:

Identifying that patients with high blood pressure often have a higher risk of developing diabetes can guide preventative care strategies.

Social network analysis

Various companies use association rule mining to identify patterns in social media data that can inform the analysis of social networks.

Example:

an analysis of social media data like Twitter might reveal that users who tweet about a particular topic are also likely to tweet about other related topics, which could inform the identification of groups or communities within the network.

Inventory Management

Association rule mining improves inventory control by predicting which products are often purchased together. It helps in managing stock levels and reducing overstock or shortages.

Example:

If customers frequently buy batteries for electronic gadgets, an extra stock of batteries can be maintained.

3.7 EXAMPLE OF ASSOCIATION RULES

Consider a small transaction dataset of a grocery store where customers purchase items like bread, butter, and milk as shown in table 3.1.

Transaction ID	Items Purchased
1	Bread, Butter
2	Bread, Milk
3	Bread, Butter, Milk
4	Milk
5	Bread, Butter

Table 3.1: Example Dataset of grocery store

The rule discovery process can be then be shown as mentioned:

Rule Discovery Process:

Rule Example: "If bread is purchased, then butter is likely to be purchased."

1. **Support Calculation:**

$$Support(Bread \rightarrow Butter) = \frac{Transaction\ containing\ both\ bread\ and\ butter}{Total\ transactions}$$

Observe that in table 3.1 we have total 5 transaction. Also note the 3 of these transaction contains both bread and butter. Hence the calculation of support is:

Support (Bread → Butter) = 3/5 = 0.6 (60%)

2. **Confidence Calculation:**

$$confidence(Bread \rightarrow Butter) = \frac{support(\ bread\ \cup\ butter)}{support(bread)}$$

Confidence = 3/4 = 0.75 (75%)

3. **Lift Calculation:**

$$Lift(Bread \rightarrow Butter) = \frac{Confidence}{support(butter)}$$

Confidence = 0.75 / 0.6 = 1.25

A lift value is greater than 1, it indicates a positive association between bread and butter.

This example demonstrates how association rules are derived and evaluated, providing actionable insights from transactional data.

Check Your Progress-2

a) A lift value greater than 1 indicates a _____ association between X and Y.

b) Quantitative association rules focus on numerical attributes and analyse patterns based on the quantity or range of values. (True/False)

c) _____ association rules involve finding patterns across multiple related datasets or tables.

d) A music streaming service might use association rule mining to recommend new artists or albums to a user based on their _____ history.

e) _____ analysis involves looking for the items customers purchase together to understand their purchasing habits and preferences.

3.8 LET US SUM UP

In this unit we have discussed the basics of association rule mining. Association Rules are one of the most important tools in data mining. As learnt we can use association rule mining in multiple contexts to extract insights and understand the underlying structure of data. Applications of Association Rules can be found in retail, healthcare, finance, and other industries that can drive smarter decision-making processes.

3.9 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a Frequent Itemset

1-b False

1-c Support Count

1-d antecedent, consequent

1-e strong predictor

1-f True

2-a positive

2-b True

2-c Multi-relational

2-d listening

2-e Market Basket

3.10 ASSIGNMENTS

- Explain the concept of Association Rule Mining and describe its significance in data analysis.
- What are the key components of an association rule?
- Explain various types of association rule mining.
- Define support, confidence, and lift with suitable examples.
- Identify applications areas other than the ones mentioned in the chapter where association rule mining can be applied.

Unit-4: Association Rule Algorithms

4

Unit Structure

4.0. Learning Objectives

4.1. Introduction

4.2. Apriori Algorithm

4.3. FP-Growth Algorithm

4.4. Comparison of FP Growth and Apriori

4.5. Let us sum up

4.6. Check your Progress: Possible Answers

4.7. Assignments

4.0 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the fundamental concepts of Association Rule Learning Algorithms.
- Analyse and Explore working of Apriori and FP Growth Algorithms.
- Compare the Apriori and FP-Growth algorithm.

4.1 INTRODUCTION

The topic of association rule learning is wide and varied, with several algorithms developed to manage the complexities of massive amounts of data and identify important patterns and correlations. Examining the various types of association rule learning algorithms helps to clarify their unique characteristics and guides the selection process for certain data mining projects.

Algorithms like Apriori, FP-Growth, and ECLAT use a different techniques to extract relevant rules from massive amounts of data. Apriori uses an iterative approach to generate frequent itemsets, it includes only those itemsets that satisfied the minimal support requirement. To increase efficiency, FP-Growth simply builds a tree structure to prevent the creation of candidates. ECLAT uses a depth-first search approach, which has been shown to produce better rule extraction outcomes, particularly for dense data sets.

Each algorithm's capabilities allow it to be specifically tailored to particular dataset properties, such as transaction volume, dimensionality, or sparsity. Support, confidence, lift, and other metrics that assess the importance of the rules extracted determine how well association rule learning algorithms perform.

These regulations are then used in recommendation systems, cyber security, retail, and healthcare decision support systems. Comprehending the subtleties of various algorithms guarantees the choice of the best method for certain data mining tasks, maximizing knowledge extraction and pattern recognition.

4.2 APRIORI ALGORITHM

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

Frequent Itemsets Generation:

Consider an itemset I = {P,Q,R,S} with four items. The total number of potential combinations is 2^n-1 = 2^4 – 1 = 16 – 1 = 15. Thus the possible subsets of I are:

{{P}, {Q}, {R},{ S}, {P,Q}, {P,R}, {P,S}, {Q,R}, {Q,S}, {R,S}, {P,Q,R}, {P,Q,S}, {P,R,S}, {Q,R,S}, {P,Q,R,S}}

A lattice diagram shown in figure 4.1 illustrates the relationship among these itemsets, where each level represents itemsets of increasing size:

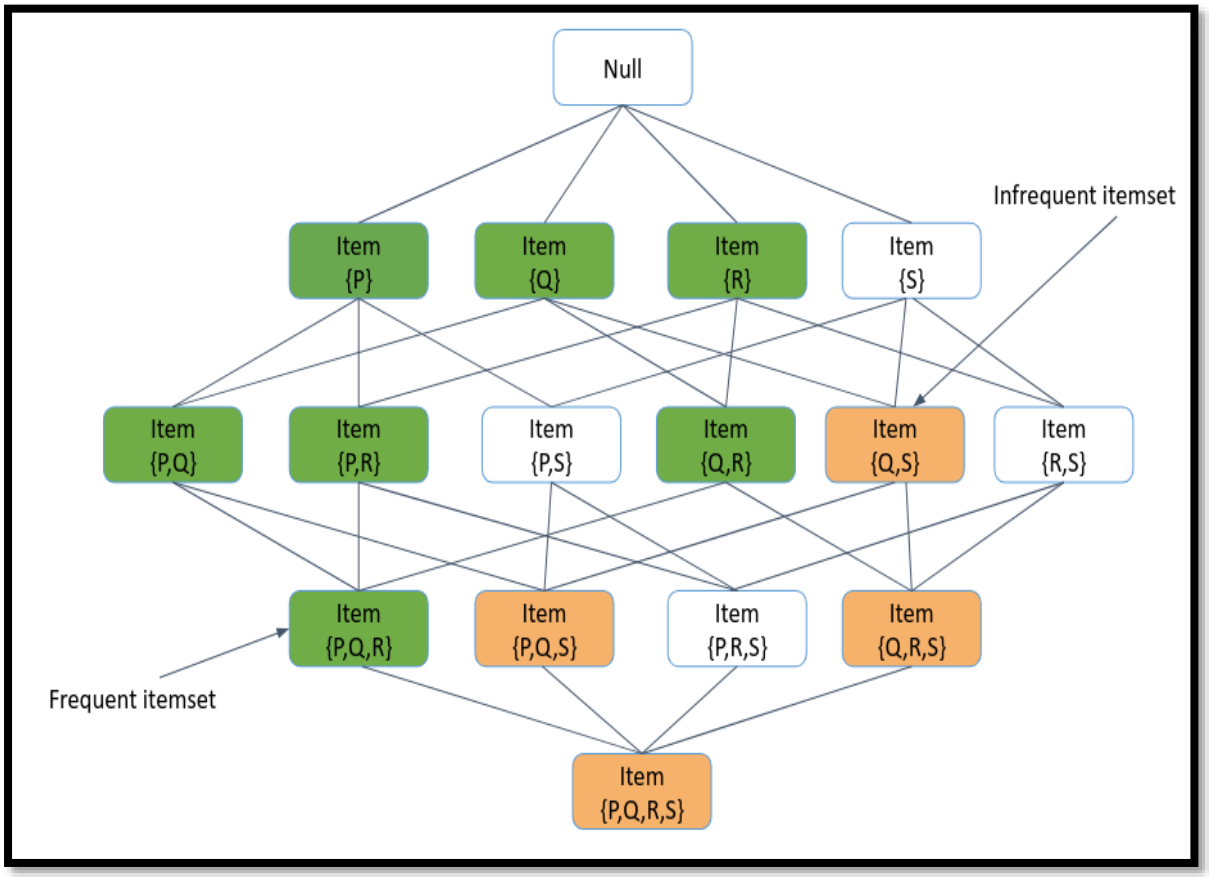


Figure 4.1: A lattice diagram to illustrate the relationships among itemsets

The itemsets at each level are shown as mentioned:

Level 1: Single-item sets → {P}, {Q}, {R}, {S}

Level 2: Two-item sets → {P, Q}, {P, R}, {P, S}, {Q, R}, {Q, S}, {R, S}

Level 3: Three-item sets → {P, Q, R}, {P, Q, S}, {P, R, S}, {Q, R, S}

Level 4: Full itemset → {P, Q, R, S}

Apriori states following two properties:

1. **Apriori Property:** If an itemset is frequent, then all of its subsets must be frequent.

   Example: Since {P, Q, R} is a frequent itemset, the subsets {P, Q}, {P, R}, {Q, R}, {P}, {Q}, and {R} formed by the combination of elements of {P, Q, R} will also be termed frequent itemsets. This is because if a transaction contains the itemset {P, Q, R} it will also contain all its subsets.

   Conversely, an itemset like {Q, S} is not a frequent itemset. Therefore, all supersets that contain them, i.e., {P, Q, S}, {Q, R, S}, and {P, Q, R, S}, will also not be frequent itemsets. This holds because if an itemset X is thought of as non-frequent, any other itemset that contains X shall also be deemed non-frequent. Restating the definition, an itemset will only be termed frequent if its support is greater than or equal to minimum support threshold.

2. **Antimonotone Property:** If an itemset set has value less than minimum support then all of its supersets will also fall below minimum support, and thus can be ignored. This property is called the Antimonotone property. This property further helps in reducing the search space.

The steps followed in Apriori Algorithm of data mining include:

- **Join Step:** In this step, (K+1) itemsets are generated by joining K-itemsets by itself.
- **Prune Step:** In this step, the counts of the candidate items are scanned from the database. If an item does not meet minimum support, it is supposed to be infrequent and hence removed. This is done with the intent to reduce the size of candidate itemsets.

Let us have a look at the steps of an Apriori Algorithm:

1. **Define minimum support threshold -** This is the minimum number of times an item set must appear in the dataset to be considered as frequent.
2. **Generate a list of frequent 1-itemsets -** Count support for each item and eliminate those with lower support than the prescribed minimal support.
3. **Generate candidate itemsets** - a list of candidate itemsets of length k+1 will be generated from all frequent k-itemsets identified in the previous step.
4. **Count the support of each candidate itemset** - Scan the dataset and count the number of appearances of each candidate itemset in the dataset.
5. **Prune the candidate itemsets** - Remove the item sets that do not meet the minimum support threshold.
6. Repeat steps 3-5 until no more frequent item sets can be generated.
7. **Generate association rules -** Once the frequent item sets have been identified, the algorithm generates association rules from them. Association rules are rules of form AB, where A and B are item sets. The rule indicates that if a transaction contains A, it is also likely to contain B.
8. **Evaluate the association rules** - Finally, the association rules are evaluated based on metrics such as confidence and lift**.**

The pseudo-code for Apriori algorithm is given in table 4.1:

```
Input: T: a set of transactions; msup: minimum support threshold
Output: Lk: set of k-frequent itemsets (result)
  L1 ← {large 1 - itemsets}
  for   k ← 2 and  Lk−1 is not empty do
        Pk← Apriori_gen( Lk−1, k)
        for transactions t in T do
                Dt ← {c in Pk : c ⊆ t}
        for candidates c in Dt do
                    count[c] ++
           Lk ← {c in Pk: count[c] ≥ msup}
           k ← k + 1
         return Union( Lk)
```

```
Apriori_gen(L, k)
for all X ⊆ L, Y ⊆ L where X1 = Y1, X2 = Y2, ..., Xk-2 = Yk-2 and  Xk-1,
                c = X ∪ {Yk-1}
if u ⊆ c for all u in C
    result ←append (result, c)
return result
```

Table 4.1: Pseudo-code for Apriori algorithm

Having seen the algorithm and the pseudocode of Apriori algorithm, let us now look at an example of an Apriori Algorithm. Let's consider the transaction dataset of a retail store as shown in the table 4.2. Assume minimum support threshold to be 2.

TID	**ITEMS**
T1	Milk, Sugar, Bread
T2	Milk, Bread, Cookies, Butter
T3	Milk, Sugar, Bread, Butter
T4	Sugar, Bread, Butter

Table 4.2: dataset of a retail store

Step 1: Create a table as shown in table 4.3 which has a support count of all the items present in the transaction database.

ITEMS	**Support Count**
Bread	4
Butter	3
Cookies	1
Milk	3
Sugar	3

Table 4.3: Count of items

Step 2: Prune the items whose support score is less than the minimum support threshold 2 as sown in table 4.4.

ITEMS	Support Count
Bread	4
Butter	3
Milk	3
Sugar	3

Table 4.4: Pruned dataset

Step 3: Find all the superset with 2 items of all the items present in the last step as shown in table 4.5.

ITEMS	Support Count
Bread, Butter	3
Bread, Milk	3
Bread, Sugar	3
Butter, Milk	2
Butter, Sugar	2
Milk, Sugar	2

Table 4.5: Superset with 2 items

As the support score of each itemset is at least 2, hence, none of the itemset is pruned.

Step 4: Find superset with 3 items in each set present in the last transaction dataset as shown table 4.6.

ITEMS	Support Count
Bread, Butter, Milk	2
Bread, Butter, Sugar	2
Bread,Milk, Sugar	2
Butter, Milk, Sugar	1

Table 4.6: Superset with 3 items

Step 5: Prune the items whose support score is less than the minimum support threshold as shown in table 4.7.

ITEMS	Support Count
Bread, Butter, Milk	2
Bread, Butter, Sugar	2
Bread, Milk, Sugar	2

Table 4.7: Pruned dataset with 3 items

Step 6: Find a superset with 4 items in each set present in the last transaction dataset as shown in table 4.8.

ITEMS	Support Count
Milk, Sugar, Bread, Butter	1

Table 4.8: Superset with 4 items

As the only itemset in table has support count 1, so it is pruned and F_K =Ø. As we have discovered all the frequent itemset. We will generate strong association rules.

Step 7: Generating Association Rules using Frequent Itemset.

Once frequent itemsets F(k) are obtained from the transaction set T, the next step involves generating strong association rules from them. The strong association rules must satisfy both minimum support and minimum confidence.

$$Confidence(X \rightarrow Y) = \frac{Number\ of\ Transactions\ containing\ X\ and\ Y}{Number\ of\ Transactions\ containing\ X}$$

One of the frequent itemset is: {Bread, Butter, Milk} hence all the possible association rules can be:

{{Bread}, {Butter}, {Milk}, {Bread, Butter}, {Bread, Milk}, {Butter, Milk}}

The generated association rules with their confidence scores are stated in table 4.9:

Rule	Confidence
{Bread, Butter} → {Milk}	$\frac{Support(Bread, Butter, Milk)}{Support(Bread, Butter)} = \frac{2}{3}$ $= 66.67\%$
{Bread, Milk} → {Butter}	$\frac{Support(Bread, Butter, Milk)}{Support(Bread, Milk)} = \frac{2}{3}$ $= 66.67\%$
{Butter, Milk} → {Bread}	$\frac{Support(Bread, Butter, Milk)}{Support(Butter, Milk)} = \frac{2}{2}$ $= 100\%$
{Bread} → {Butter, Milk}	$\frac{Support(Bread, Butter, Milk)}{Support(Bread)} = \frac{2}{4}$ $= 50\%$
{Butter} → {Bread, Milk}	$\frac{Support(Bread, Butter, Milk)}{Support(\ Butter)} = \frac{2}{3}$ $= 66.67\%$
{Milk} → {Bread, Butter}	$\frac{Support(Bread, Butter, Milk)}{Support(Milk)} = \frac{2}{3}$ $= 66.67\%$

Table 4.9: Rules and associated confidence score calculation

Depending on the minimum confidence scores, the obtained association rules are preserved, and rest are pruned.

Check Your Progress-1

a) What is the relation between a candidate and frequent itemsets?
   1. A candidate itemset is always a frequent itemset
   2. A frequent itemset must be a candidate itemset
   3. No relation between these two
   4. Strong relation with transactions

b) The _____ is not suitable for handling large datasets because it generates a large number of candidates.

c) Steps in Apriori algorithm are _____ and _____.

d) In the Apriori algorithm, if its support is greater than or equal to the minimum support threshold an itemset is called _____.

4.3 FREQUENT PATTERN GROWTH ALGORITHM

In spite of a considerable shrinking of the set of candidate itemsets, the Apriori algorithm can be slow due to the scanning of the entire transaction set in every iteration. That is why Frequent Pattern growth, also called FP growth method, adopts a divide and conquer scheme to discover frequent itemsets. The FP-growth (Frequent Pattern Growth) algorithm is currently one of the fastest approaches to frequent itemset mining.

Working of FP Growth:

FP-Growth algorithm mines frequent itemsets using divide-and-conquer strategy. The working process can be summarized as follows:

1. **FP-tree construction:**
    - Make a single scan of the transactional database to collect frequent items and count their support.
    - Sort the frequent items in the descending order of their support.
    - Build the initial FP-tree by inserting transactions into the tree according to item order and their supports.
2. **Generation of Conditional FP-trees:**
    - For each frequent item in order of decreasing support, construct a Conditional Pattern Base by pulling the suffixes from the FP-tree corresponding to that item.
    - Construct a conditional FP-tree from the Conditional Pattern Base by compressing the suffixes.
3. **Mining Frequent Itemsets:**
   Recursively mine the conditional FP-trees for frequent itemsets until no more frequent itemsets are detectable.

Let us have a look at an example of FP Growth Algorithm**.** Let's consider the transaction dataset of a retail store used for Apriori algorithm as shown in the table.

TID	ITEMS
T1	Milk, Sugar, Bread
T2	Milk, Bread, Cookies, Butter
T3	Milk, Sugar, Bread, Butter
T4	Sugar, Bread, Butter

Step 1: Create a table which computes the frequency of each item present in the transaction database as shown in table 4.10.

Items	Frequency
Bread	4
Butter	3
Cookies	1
Milk	3
Sugar	3

Table 4.10: Frequency table

Step 2: Provided minimum support as 2. After removing all the items below minimum support in the table 4.10, we would remain with the items as shown in table 4.11.

Items	Frequency
Bread	4
Butter	3
Milk	3
Sugar	3

Table 4.11: Items above minimum support

Step 3: Let's re-order the transaction database based on the items above minimum support. In this step, in each transaction, we will remove infrequent items and re-order them in the descending order of their frequency, as shown in the table 4.12.

Tid	Items	Ordered Itemset
T1	Milk, Sugar, Bread	Bread, Milk, Sugar
T2	Milk, Bread, Cookies, Butter	Bread, Butter ,Milk
T3	Milk, Sugar, Bread, Butter	Bread, Butter ,Milk ,Sugar
T4	Sugar, Bread, Butter	Bread, Butter ,Sugar

Table 4.12: Reordered transaction dataset

Now we will use the ordered itemset in each transaction to build the FP tree. Each transaction will be inserted individually to build the FP tree, as shown below -

Step 4: First Transaction T1: {Bread, Milk, Sugar}:

In this transaction, all items are simply linked, and their support count is initialized as 1 as can be seen in figure 4.2.

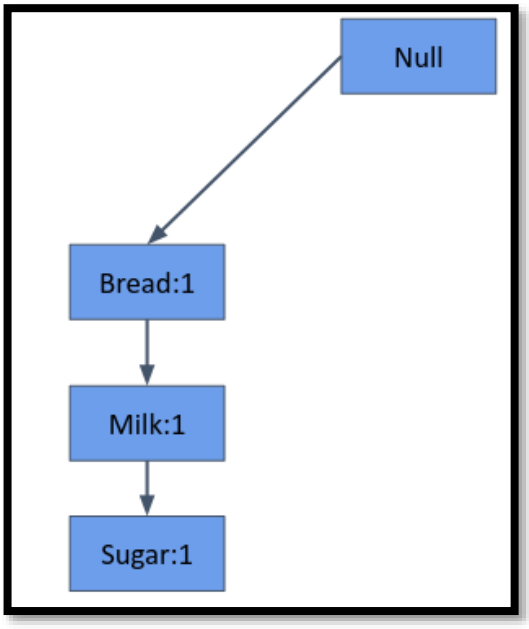


Figure 4.2: FP tree for Transaction T1

Step 5: Second Transaction T2: {Bread, Butter, Milk}:

In this transaction, we will increase the support count of bread in the tree to 2. As no direct link is available from Butter to Milk, we will insert a new path for Butter and Milk and initialize their support count as 1 as shown in figure 4.3.

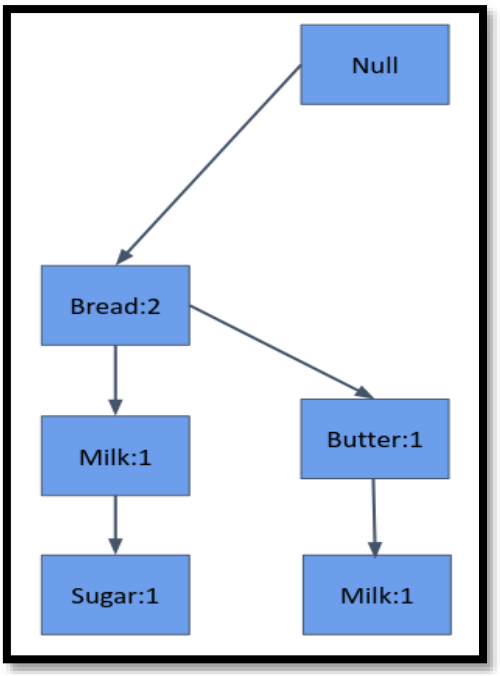


Figure 4.3: FP tree for Transaction T2

Step 6: Now examine the transaction T3 and T4, and further expand the FP tree as shown in figure 4.4.

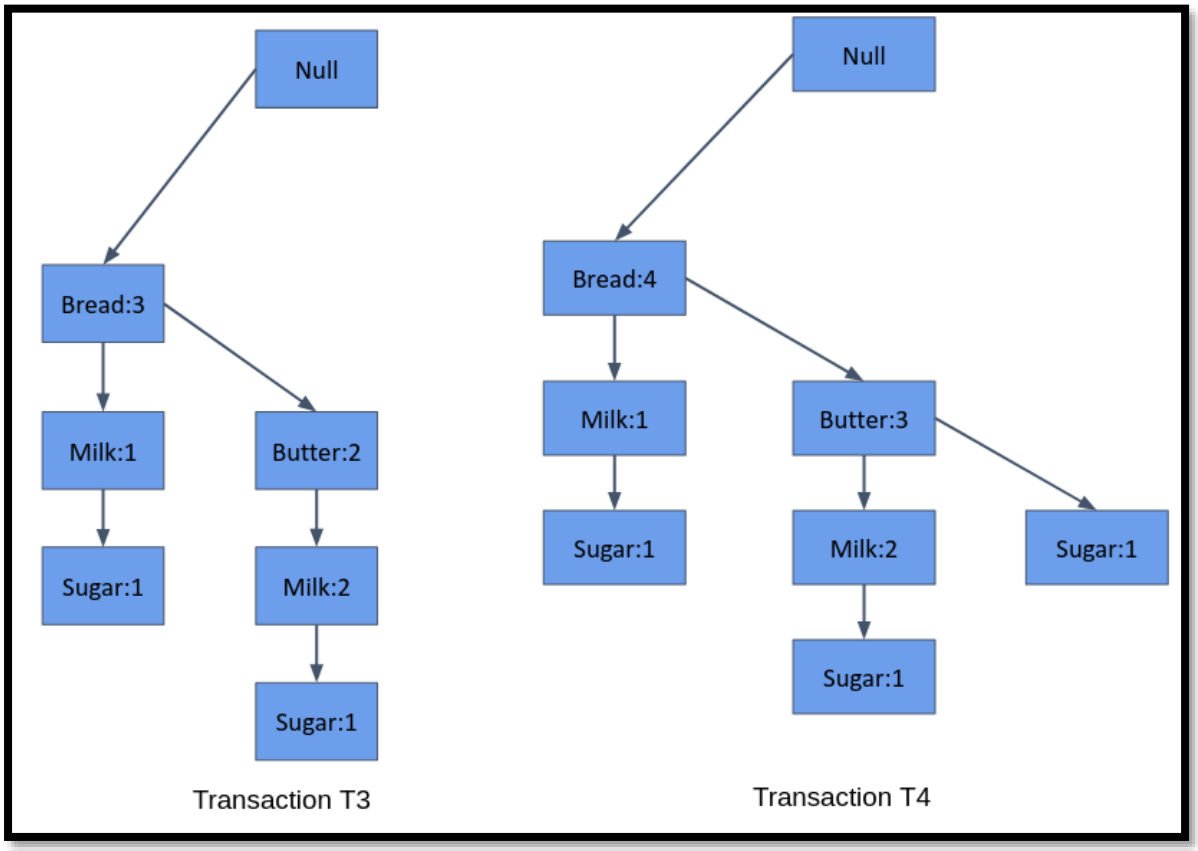


Figure 4.3: FP tree for Transaction T3 and T4

Step 7: Now, obtain the conditional FP tree for all the items by scanning the path from the lowest leaf node as shown in table 4.13.

Items	Conditional Pattern Base
Sugar	{Bread, Milk}:1, {Bread, Butter ,Milk}:1, {Bread, Butter}:1
Milk	{Bread}:1, {Bread, Butter}:3
Butter	{Bread}:1
Bread	

Table 4.13: Conditional FP Tree

Step 8: Now for each item, we will build a conditional frequent pattern tree. It is computed by identifying the set of elements common in all the paths in the conditional pattern base of a given frequent item and computing its support count by summing the support counts of all the paths in the conditional pattern base. The conditional frequent pattern tree will look like the one shown in table 4.14.

Items	Conditional Pattern Base	Conditional FP Tree
Sugar	{Bread, Milk}:1, {Bread, Butter ,Milk}:1, {Bread, Butter}:1	{Bread}:3
Milk	{Bread}:1, {Bread, Butter}:3	{Bread}:4
Butter	{Bread}:1	{Bread}:1
Bread		

Table 4.13: Conditional Pattern Base

Step 9: From the above conditional FP tree, we will generate the frequent itemsets as shown in table 4.14.

Items	Frequent Patterns
Sugar	{Bread, Sugar}:3
Milk	{Bread, Milk}:4
Butter	{Bread, Butter}:1

Table 4.14: Frequent itemsets

4.4 FP GROWTH VS. APRIORI

Apriori and FP Growth are the most common algorithms for mining frequent itemsets. Each algorithm will carry out its own design of discovering or identifying the frequent patterns. Table 4.15 provides a comaprison between the two algorithms based upon various factors like working process, number of scans, use of memory, speed, and scalability.

Factor	FP Growth	Apriori
Working Process	The FP-growth algorithm mines frequent itemsets through the use of the FP-tree.	Apriori algorithm mines frequent items in an iterative manner - 1-itemsets, 2-itemsets, 3-itemsets, etc.
Candidate Generation	Frequent itemsets are generated from FP-tree construction with recursive generation of the conditional pattern bases.	Candidates are generated through join-and-prune.
Database Scanning	The database is scanned only twice to construct the FP-Tree and generate conditional pattern bases.	The database is scanned several times in frequent itemset mining.
Memory	Takes up less memory than Apriori to run because it constructs the FP-Tree, which compresses the database.	Requires considerable memory to store candidate itemsets.
Speed	Faster due to effective data compression and generation of frequent itemsets.	Slower because of candidate generation and multiple database scans.
Scalability	Performs well on large datasets owing to efficient data compression and generation of frequent itemsets.	Performs poorly on large datasets due to excessive candidate itemsets.

Table 4.15: Comparison between the FP Growth and the Apriori

Check Your Progress-2

a) FP growth algorithm mines all frequent patterns by constructing a FP tree. (True/False)

b) FP growth algorithms expand the original database to build FP trees. (True/False)

c) The FP-tree (Frequent Pattern tree) is a data structure used in the FP Growth algorithm that stores the _____ and _____.

d) Unlike Apriori, the FP-Growth algorithm avoids generating _____, making it more efficient.

4.5 LET US SUM UP

In this unit we have discussed two classical algorithms: Apriori and FP-Growth, with their working. In Apriori, it has been found that for all candidate sets of items; at each level, one should discover all subsets; hence the lengths of the frequent sets are greater and so is the number of candidate generations. To rectify this problem, the FP-Growth algorithm was used here, which performs efficiently while being memory consuming due to the tree method.

4.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1-a (1) A frequent itemset must be a candidate itemset

1-b Apriori algorithm

1-c Join, Prune

1-d Frequent

2-a True

2-b False

2-c Frequent item sets and their support counts

2-d Candidate itemsets

4.7 ASSIGNMENTS

- Explain the basic requirements of Aprioiri algorithm.
- What is the FP Growth algorithm in machine learning used for?
- Differentiate Apriori algorithm and the FP Growth algorithm.
- Find the frequent itemset using FP Tree, from the given set of Transaction Dataset with minimum support score as 3.

Transaction ID	**Items**
T1	{M, N, O, E, K, Y}
T2	{D, O, E, N, Y, K}
T3	{K, A, M, E}
T4	{M, C, U, Y, K}
T5	{C, O, K, O, E, I}

BAOU
Education for All

યુનિવર્સિટી ગીત

સ્વાધ્યાયઃ પરમં તપઃ
સ્વાધ્યાયઃ પરમં તપઃ
સ્વાધ્યાયઃ પરમં તપઃ

શિક્ષણ, સંસ્કૃતિ, સદ્‌ભાવ, દિવ્યબોધનું ધામ
ડૉ. બાબાસાહેબ આંબેડકર ઓપન યુનિવર્સિટી નામ;
સૌને સૌની પાંખ મળે, ને સૌને સૌનું આભ,
દશે દિશામાં સ્મિત વહે હો દશે દિશે શુભ-લાભ.

અભણ રહી અજ્ઞાનના શાને, અંધકારને પીવો ?
કહે બુદ્ધ આંબેડકર કહે, તું થા તારો દીવો;
શારદીય અજવાળા પહોંચ્યાં ગુર્જર ગામે ગામ
ધ્રુવ તારકની જેમ ઝળહળે એકલવ્યની શાન.

સરસ્વતીના મયૂર તમારે ફળિયે આવી ગહેકે
અંધકારને હડસેલીને ઉજાસના ફૂલ મહેંકે;
બંધન નહીં કો સ્થાન સમયના જવું ન ઘરથી દૂર
ઘર આવી મા હરે શારદા દૈન્ય તિમિરના પૂર.

સંસ્કારોની સુગંધ મહેંકે, મન મંદિરને ધામે
સુખની ટપાલ પહોંચે સૌને પોતાને સરનામે;
સમાજ કેરે દરિયે હાંકી શિક્ષણ કેરું વહાણ,
આવો કરીયે આપણ સૌ
ભવ્ય રાષ્ટ્ર નિર્માણ...
દિવ્ય રાષ્ટ્ર નિર્માણ...
ભવ્ય રાષ્ટ્ર નિર્માણ

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
(Established by Government of Gujarat)
'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481
Website : www.baou.edu.in