SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)

MCA DEPARTMENT

LECTURE NOTES

Subject Name: MACHINE LEARNING

Year / Branch: II MCA – I SEMESTER

Regulation: R24

Prepared By: Mr. M RAJESH,

Assistant Professor

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
MCA DEPARTMENT

II MCA – I SEMESTER			
COURSE CODE:	**24MCA211**	**CREDITS:**	**4**
COURSE TITLE:	**MACHINE LEARNING**	**L-T-P:**	**4-0-0**

PREREQUISITES: *Courses on "***Artificial Intelligence***"*& *"***Data Mining"**

COURSE EDUCATIONAL OBJECTIVES*:*

- ***CEO1*** *To introduce students to the basic concepts and techniques of Machine Learning.*
- ***CEO2*** *To have a thorough understanding of the Supervised and Unsupervised learning techniques*
- ***CEO3*** *To study the various probability based learning techniques.*
- ***CEO4*** *To understand graphical models of machine learning algorithms.*

UNIT I:INTRODUCTION AND DATA PREPROCESSING: **Lecture Hrs:10**

Basic concept of Machine Learning- Types of Machine Learning-– Supervised Learning – Unsupervised Learning- Reinforcement learningApplications and challenges-A Brief Review of ProbabilityTheory-Baye's Theorem.

Data Preprocessing- Data Cleaning, Data Integration, Data Transformation, Data Reduction or Dimensionality Reduction - Principal component Analysis.

UNIT II:MODEL EVALUATION AND FEATURE ENGINEERING: **Lecture Hrs:12**

Selecting a Model, training a Model, Model Representation andInterpretability, Evaluating Performance of a Model, Improving performance of a Model. Basics ofFeature Engineering-Feature Transformation, Feature Subset Selection.

UNIT III: REGRESSION : **Lecture Hrs:12**

Introduction to Supervised Learning and Regression, Linear Regression, Evaluation of ModelEstimators, Regularization-Ridge regression, LASSO regression, Multi Linear Regression,GradientBased Methods. Cost function- Minimizing the Cost Function for a Single-VariableFunction, Minimizing the Cost Function for a Two-Variable Function, Evaluation Metrics (MeanAbsolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Root MeanSquared Log Error (RMSLE), R Squared (R2), Adjusted R Squared).

UNIT IV: Classification : **Lecture Hrs:12**

Introduction to Classification, Logistic Regression-Building Logistic Regression Model (LogitFunction), Maximum Likelihood Estimation. Decision Tree-Steps to Construct a Decision Tree,Classification Using Decision Trees, Issues in Decision Trees, Ensemble Learning-Random Forest.Bayesian Classification-Naive Bayes Classifier, k-Nearest Neighbor (KNN). Neuron-Neural Networks-The PerceptronMultilayerPerceptron(MLP), Support Vector Machines- Linear Support Vector Machines, Optimal Hyperplane, RadialBasis Functions, Evaluation Metrics (Accuracy, Confusion Matrix, Precision, Recall, F1 Score).

UNIT V: Clustering **Lecture Hrs:12**

Introduction to Unsupervised Learning Algorithms, Clustering- Types of Clustering, Partitioning Methods of Clustering, Hierarchical Methods, Density-based Methods and Grid-Based Methods- K-

means Clustering- Choosing number of Clusters.

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, "Machine Learning", Wiley Publisher,2019.
2. SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, "Machine Learning",Pearson,2019.
3. Alpaydin Ethem, "Introduction to Machine Learning", 3rd Edition, PHI learningprivate limited, 2019.

REFERENCE BOOKS:

1. *"Machine Learning: A Probabilistic Perspective", 2012, Kevin P. Murphy, MIT Press.*
2. *"Pattern Recognition and Machine Learning", 2007, Christopher Bishop, Springer.*
3. *"Introduction to Machine Learning", MIT Press, 3/e, 2014, Ethem Alpaydin.*

COURSE OUTCOMES: *On successful completion of this course, students will be able to:*		POs related to COs
CO1	Use knowledge on statistics for machine learning, Understand about Data Preprocessing, Dimensionality reduction	PO1, PO2,PO3
CO2	Apply proper model for the given problem and use feature Engineering Techniques.	PO1, PO2,PO3, PO4, PO8
CO3	Analyze various regression techniques and Neural Networks	PO1, PO2,PO3,P04
CO4	Identify various classification algorithm and their methodologies	PO1, PO2,PO3,P04, PO8
CO5	Understand and apply various clustering algorithms	PO1, PO2,PO3,P04, PO8

CO-PO MAPPING(DETAILED; HIGH:3; MEDIUM:2; LOW:1)

Course	**POs / COs**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**
C301: Machine Learning	**C301.1**	3	3	3	-	-	-	-	-
	C301.2	3	2	3	2	-	-	-	**2**
	C301.3	3	2	2	3	-	-	-	-
	C301.4	3	3	3	3	-	-	-	**3**
	C301.5	3	3	3	2	-	-	-	**3**
	C301	3	2.6	2.8	2.5	-			**2.4**

UNIT I

INTRODUCTION AND DATA PREPROCESSING

“Machine Learning teaches computers to learn, adapt, and improve without being explicitly programmed.”

— Arthur Samuel

Unit Overview

This unit introduces the fundamental concepts of Machine Learning (ML), a branch of Artificial Intelligence that enables computers to learn from data and make predictions or decisions without being explicitly programmed. It covers the basic concepts of Machine Learning, different types of learning techniques such as Supervised Learning, Unsupervised Learning, and Reinforcement Learning, along with their applications and challenges. The unit also provides a brief review of Probability Theory and Bayes' Theorem, which form the mathematical foundation for many machine learning algorithms.

In addition, the unit explains the importance of Data Preprocessing, including Data Cleaning, Data Integration, Data Transformation, and Data Reduction (Dimensionality Reduction). Finally, it introduces Principal Component Analysis (PCA) as an effective technique for reducing the dimensionality of data while preserving essential information, thereby improving model efficiency and performance.

Objectives of the Unit

After studying this unit, the students will be able to:

- Understand the basic concepts and workflow of Machine Learning.
- Differentiate between Supervised, Unsupervised, and Reinforcement Learning.
- Explain the applications and challenges of Machine Learning.
- Understand the concepts of Probability Theory and Bayes' Theorem.
- Describe the importance of Data Preprocessing in Machine Learning.
- Explain various data preprocessing techniques such as Data Cleaning, Data Integration, Data Transformation, and Data Reduction.
- Understand the concept and working of Principal Component Analysis (PCA).
- Apply preprocessing techniques to improve the quality and performance of machine learning models.

Learning Outcomes

After completing this unit, students will be able to:

- Explain the fundamental concepts of Machine Learning.
- Classify machine learning algorithms into Supervised, Unsupervised, and Reinforcement Learning.
- Identify suitable machine learning techniques for different real-world problems.
- Apply Bayes' Theorem to solve probability-based machine learning problems.
- Perform data preprocessing to improve data quality.
- Implement data cleaning, integration, transformation, and reduction techniques.
- Understand the role of Principal Component Analysis in dimensionality reduction.

Importance of Studying this Unit

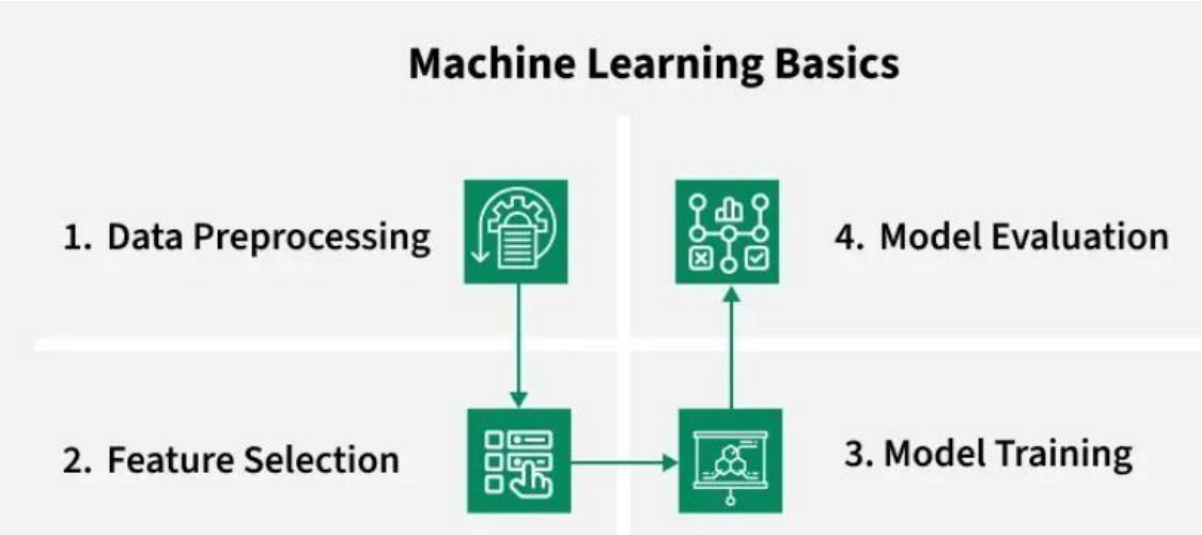
- Provides the foundation for understanding Machine Learning concepts.
- Helps in selecting suitable learning algorithms for different problems.
- Introduces mathematical concepts required for machine learning.
- Improves understanding of data preprocessing techniques.
- Enhances data quality before model training.
- Introduces dimensionality reduction techniques for handling high-dimensional data.
- Develops practical skills required for real-world machine learning applications.
- Forms the basis for advanced machine learning and artificial intelligence topics.

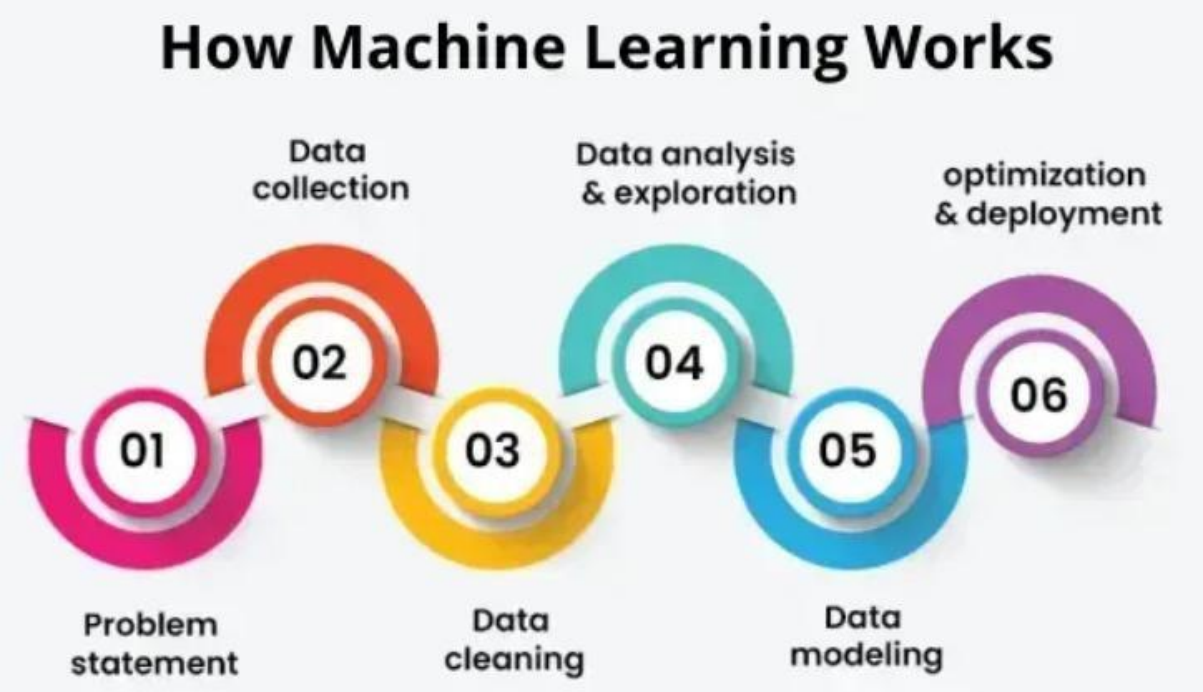
Key Concepts

- **Machine Learning (ML):** A branch of Artificial Intelligence that enables computers to learn from data and make predictions without explicit programming.
- **Artificial Intelligence (AI):** The field of computer science that focuses on creating systems capable of performing tasks that require human intelligence.
- **Supervised Learning:** A learning approach in which the model is trained using labeled data to predict outputs.
- **Unsupervised Learning:** A learning approach in which the model discovers hidden patterns or groups from unlabeled data.
- **Reinforcement Learning:** A learning method in which an agent learns by interacting with an environment and receiving rewards or penalties.
- **Classification:** A supervised learning task that predicts categorical class labels.
- **Regression:** A supervised learning task used to predict continuous numerical values.
- **Bayes' Theorem:** A probability theorem used to calculate the probability of an event based on prior knowledge.
- **Data Preprocessing:** The process of cleaning and transforming raw data into a suitable format for machine learning.
- **Data Reduction (Dimensionality Reduction):** The process of reducing the number of features while preserving important information.
- **Principal Component Analysis (PCA):** A dimensionality reduction technique that transforms high-dimensional data into fewer principal components while retaining maximum variance.

Introduction to Machine learning

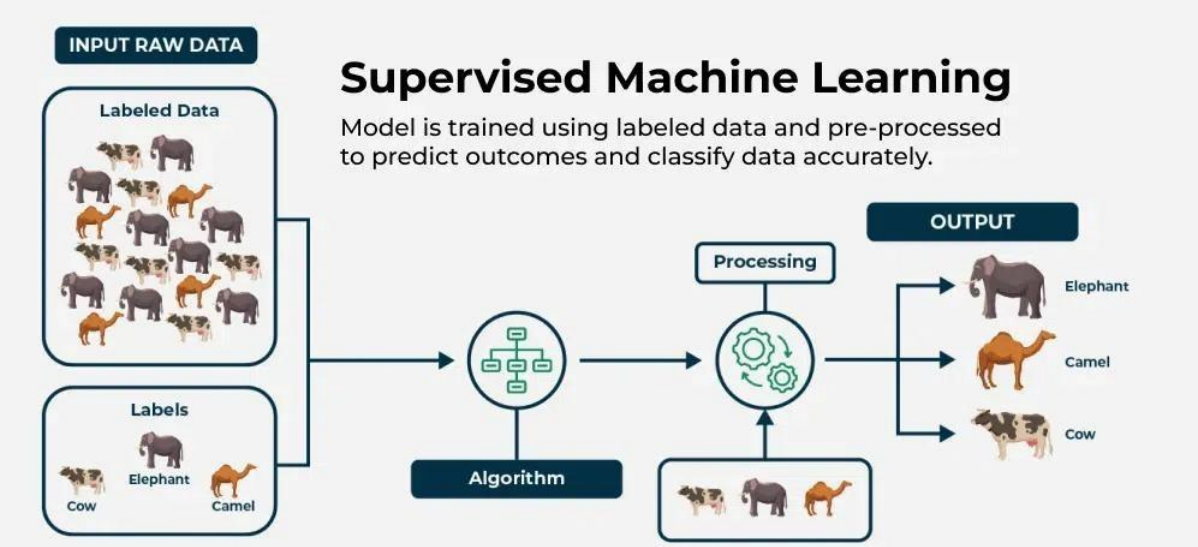
Machine learning is a branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data without being explicitly programmed for every task. In simple words, ML teaches the systems to think and understand like humans by learning from the data.





SUPERVISED MACHINE LEARNING

Supervised machine learning is a fundamental approach for machine learning and artificial intelligence. It involves training a model using labelled data, where each input comes with a corresponding correct output. The process is like a teacher guiding a student—hence the term "supervised" learning. In this article, we'll explore the key components of supervised learning, the different types of supervised machine learning algorithms used, and some practical examples of how it works.



What is Supervised Machine Learning?

As we explained before, **supervised learning** is a type of machine learning where a model is trained on labelled data—meaning each input is paired with the correct output. The model learns by comparing its predictions with the actual answers provided in the training data. Over time, it adjusts itself to minimize errors and improve accuracy. The goal of supervised learning is to make accurate predictions when given new, unseen data. For example, if a model is trained to recognize handwritten digits, it will use what it learned to correctly identify new numbers it hasn't seen before.

Supervised learning can be applied in various forms, including supervised learning classification and supervised learning regression, making it a crucial technique in the field of artificial intelligence and supervised data mining. A fundamental concept in supervised machine learning is learning a class from examples. This involves providing the model with examples where the correct label is known, such as learning to classify images of cats and dogs by being shown labeled examples of both. The model then learns the distinguishing features of each class and applies this knowledge to classify new images.

How Supervised Machine Learning Works?

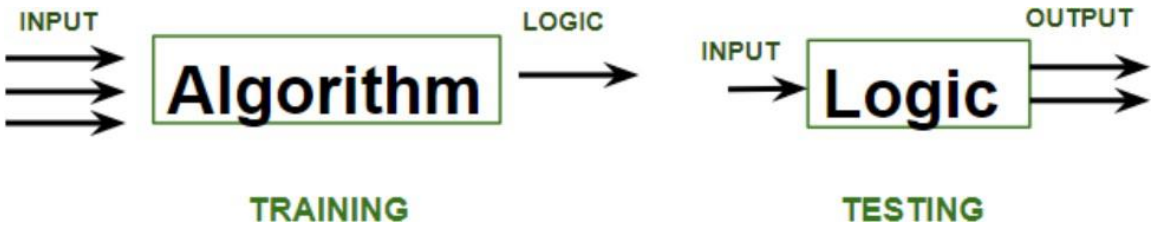
Where **supervised learning algorithm** consists of input features and corresponding output labels. The process works through:

- **Training Data:** The model is provided with a training dataset that includes input data (features) and corresponding output data (labels or target variables).
- **Learning Process:** The algorithm processes the training data, learning the relationships between the input features and the output labels. This is achieved by adjusting the model's parameters to minimize the difference between its predictions and the actual labels.

After training, the model is evaluated using a test dataset to measure its accuracy and performance. Then the model's performance is optimized by adjusting parameters and using techniques like cross-validation to balance bias and variance. This ensures the model generalizes well to new, unseen data.

In summary, supervised machine learning involves training a model on labelled data to learn patterns and relationships, which it then uses to make accurate predictions on new data.

Let's learn how a supervised machine learning model is trained on a dataset to learn a mapping function between input and output, and then with learned function is used to make predictions on new data:



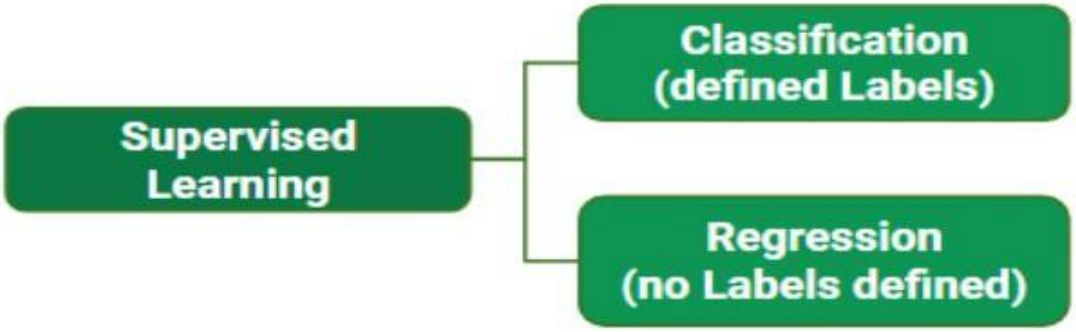
In the image above,

- **Training** phase involves feeding the algorithm labeled data, where each data point is paired with its correct output. The algorithm learns to identify patterns and relationships between the input and output data.
- **Testing** phase involves feeding the algorithm new, unseen data and evaluating its ability to predict the correct output based on the learned patterns.

Types of Supervised Learning in Machine Learning

Now, supervised learning can be applied to two main types of problems:

- Classification: Where the output is a categorical variable (e.g., spam vs. non-spam emails, yes vs. no).
- Regression: Where the output is a continuous variable (e.g., predicting house prices, stock prices



While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and the rest as testing data. In training data, we feed input as well as output for 80% of data. The model learns from training data only.

Let's first understand the classification and regression data through the table below:

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

Both the above figures have labelled data set as follows:

- **Figure A:** It is a dataset of a shopping store that is useful in predicting whether a customer will purchase a particular product under consideration or not based on his/ her gender, age, and salary.
  Input: Gender, Age, Salary
  Output: Purchased i.e. 0 or 1; 1 means yes the customer will purchase and 0 means that the customer won't purchase it.

- **Figure B:** It is a Meteorological dataset that serves the purpose of predicting wind speed based on different parameters.
  Input: Dew Point, Temperature, Pressure, Relative Humidity, Wind Direction
  Output: Wind Speed

Practical Examples of Supervised learning

Few practical examples of supervised machine learning across various industries:

- **Fraud Detection in Banking:** Utilizes supervised learning algorithms on historical transaction data, training models with labelled datasets of legitimate and fraudulent transactions to accurately predict fraud patterns.
- **Parkinson Disease Prediction:** Parkinson's disease is a progressive disorder that affects the nervous system and the parts of the body controlled by the nerves.
- **Customer Churn Prediction:** Uses supervised learning techniques to analyse historical customer data, identifying features associated with churn rates to predict customer retention effectively.
- **Cancer cell classification:** Implements supervised learning for cancer cells based on their features, and identifying them if they are 'malignant' or 'benign.
- **Stock Price Prediction:** Applies supervised learning to predict a signal that indicates whether buying a particular stock will be helpful or not.

Supervised Machine Learning Algorithms

Supervised learning can be further divided into several different types, each with its own unique characteristics and applications. Here are some of the most common types of supervised learning algorithms:

- **Linear Regression**: Linear regression is a type of supervised learning regression algorithm that is used to predict a continuous output value. It is one of the simplest and most widely used algorithms in supervised learning.
- **Logistic Regression**: Logistic regression is a type of supervised learning classification algorithm that is used to predict a binary output variable.
- **Decision Trees**: Decision tree is a tree-like structure that is used to model decisions and their possible

consequences. Each internal node in the tree represents a decision, while each leaf node represents a possible outcome.

- **Random Forests**: Random forests again are made up of multiple decision trees that work together to make predictions. Each tree in the forest is trained on a different subset of the input features and data. The final prediction is made by aggregating the predictions of all the trees in the forest.
- **Support Vector Machine (SVM)**: The SVM algorithm creates a hyper plane to segregate n-dimensional space into classes and identify the correct category of new data points. The extreme cases that help create the hyper plane are called support vectors, hence the name Support Vector Machine.
- **K-Nearest Neighbours (KNN) :** KNN works by finding k training examples closest to a given input and then predicts the class or value based on the majority class or average value of these neighbours. The performance of KNN can be influenced by the choice of k and the distance metric used to measure proximity.
- **Gradient Boosting**: Gradient Boosting combines weak learners, like decision trees, to create a strong model. It iteratively builds new models that correct errors made by previous ones.
- **Naive Bayes Algorithm**: The Naive Bayes algorithm is a supervised machine learning algorithm based on applying Bayes' Theorem with the “naive” assumption that features are independent of each other given the class label.

Algorithm	Regression, Classification	Purpose	Method	Use Cases
Linear Regression	Regression	Predict continuous output values	Linear equation minimizing sum of squares of residuals	Predicting continuous values
Logistic Regression	Classification	Predict binary output variable	Logistic function transforming linear relationship	Binary classification tasks
Decision Trees	Both	Model decisions and outcomes	Tree-like structure with decisions and outcomes	Classification and Regression tasks
Random Forests	Both	Improve classification and regression accuracy	Combining multiple decision trees	Reducing overfitting, improving prediction accuracy
SVM	Both	Create hyperplane for classification or predict continuous values	Maximizing margin between classes or predicting continuous values	Classification and Regression tasks
KNN	Both	Predict class or value based on k closest neighbors	Finding k closest neighbors and predicting based on majority or average	Classification and Regression tasks, sensitive to noisy data
Gradient Boosting	Both	Combine weak learners to create strong model	Iteratively correcting errors with new models	Classification and Regression tasks to improve prediction accuracy
Naive Bayes	Classification	Predict class based on feature independence assumption	Bayes' theorem with feature independence assumption	Text classification, spam filtering, sentiment analysis, medical

Training a Supervised Learning Model: Key Steps

The goal of supervised learning is to generalize well to unseen data. Training a model for supervised learning involves several crucial steps; each designed to prepare the model to make accurate predictions or decisions based on labelled data. Below are the key steps involved in training a model for supervised machine learning:

1. **Data Collection and Pre-processing**: Gather a labelled dataset consisting of input features and target output labels. Clean the data, handle missing values, and scale features as needed to ensure high quality for **supervised learning** algorithms.
2. **Splitting the Data**: Divide the data into **training set** (80%) and the **test set** (20%).
3. **Choosing the Model**: Select appropriate algorithms based on the problem type. This step is crucial for effective **supervised learning** in AI.
4. **Training the Model**: Feed the model input data and output labels, allowing it to learn patterns by adjusting internal parameters.

5. **Evaluating the Model**: Test the trained model on the unseen test set and assess its performance using

various metrics.

6. **Hyper parameter Tuning:** Adjust settings that control the training process (e.g., learning rate) using techniques like grid search and cross-validation.
7. **Final Model Selection and Testing**: Retrain the model on the complete dataset using the best hyper parameters testing its performance on the test set to ensure readiness for deployment.
8. **Model Deployment**: Deploy the validated model to make predictions on new, unseen data.

By following these steps, supervised learning models can be effectively trained to tackle various tasks, from **learning a class from examples** to making predictions in real-world applications.

Advantages of Supervised Learning

The power of **supervised learning** lies in its ability to accurately predict patterns and make data-driven decisions across a variety of applications. Here are some advantages of **supervised learning** listed below:

- **Supervised learning** excels in accurately predicting patterns and making data-driven decisions.
- **Labelled training data** is crucial for enabling **supervised learning models** to learn input-output relationships effectively.
- **Supervised machine learning** encompasses tasks such as **supervised learning classification** and **supervised learning regression**.
- Applications include complex problems like image recognition and natural language processing.
- Established evaluation metrics (accuracy, precision, recall, F1-score) are essential for assessing **supervised learning model** performance.
- Advantages of **supervised learning** include creating complex models for accurate predictions on new data.
- **Supervised learning** requires substantial labelled training data, and its effectiveness hinges on data quality and representativeness.

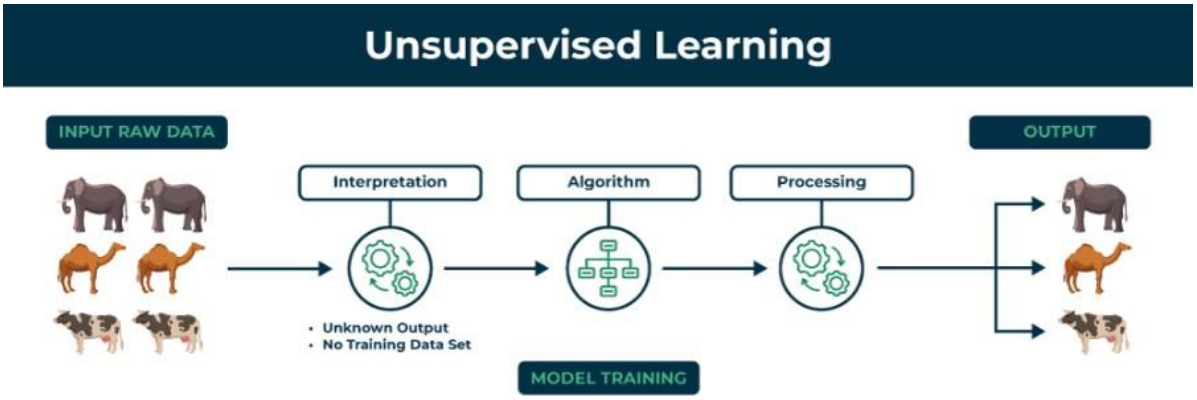
Disadvantages of Supervised Learning

Despite the benefits of **supervised learning methods**, there are notable **disadvantages of supervised learning**:

1. **Over fitting**: Models can over fit training data, leading to poor performance on new data due to capturing noise in **supervised machine learning**.
2. **Feature Engineering**: Extracting relevant features is crucial but can be time-consuming and requires domain expertise in **supervised learning applications**.
3. **Bias in Models:** Bias in the training data may result in unfair predictions in **supervised learning algorithms**.
4. **Dependence on Labelled Data**: **Supervised learning** relies heavily on labelled training data, which can be costly and time-consuming to obtain, posing a challenge for **supervised learning techniques**

UNSUPERVISED LEARNING

Unsupervised learning is a branch of machine learning that deals with unlabelled data. Unlike supervised learning, where the data is labelled with a specific category or outcome, unsupervised learning algorithms **are tasked with finding patterns and relationships within the data without any prior knowledge of the data's meaning**. Unsupervised machine learning algorithms **find hidden patterns and data without any human intervention, i.e., we don't give output to our model. The training model has only input parameter values and discovers the groups or patterns on its own.**



The image shows set of animals: elephants, camels, and cows that represents raw data that the unsupervised learning algorithm will process.

- The "Interpretation" stage signifies that the algorithm doesn't have predefined labels or categories for the data. It needs to figure out how to group or organize the data based on inherent patterns.
- **Algorithm** represents the core of unsupervised learning process using techniques like clustering, dimensionality reduction, or anomaly detection to identify patterns and structures in the data.
- **Processing** stage shows the algorithm working on the data.

The output shows the results of the unsupervised learning process. In this case, the algorithm might have grouped the animals into clusters based on their species (elephants, camels, and cows).

How does unsupervised learning work?

Unsupervised learning works by analysing unlabelled data to identify patterns and relationships. The data is not labelled with any predefined categories or outcomes, so the algorithm must find these patterns and relationships on its own. This can be a challenging task, but it can also be very rewarding, as it can reveal insights into the data that would not be apparent from a labelled dataset.

Data-set in Figure A is Mall data that contains information about its clients that subscribe to them. Once subscribed they are provided a membership card and the mall has complete information about the

customer and his/her every purchase. Now using this data and unsupervised learning techniques, the mall can easily group clients based on the parameters we are feeding in.

CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35

The input to the unsupervised learning models is as follows:

- **Unstructured data**: May contain noisy(meaningless) data, missing values, or unknown data
- **Unlabeled data**: Data only contains a value for input parameters, there is no targeted value(output). It is easy to collect as compared to the labeled one in the Supervised approach.

Unsupervised Learning Algorithms

There are mainly 3 types of Algorithms which are used for unsupervised dataset.

- **Clustering**
- **Association Rule Learning**
- **Dimensionality Reduction**

1. **Clustering Algorithms**

Clustering in unsupervised machine learning is the process of grouping unlabelled data into clusters based on their similarities. The goal of clustering is to identify patterns and relationships in the data without any prior knowledge of the data's meaning.

Broadly this technique is applied to group data based on different patterns, such as similarities or differences, our machine model finds. These algorithms are used to process raw, unclassified data objects into groups. For example, in the above figure, we have not given output parameter values, so this technique will be used to group clients based on the input parameters provided by our data.

Some common clustering algorithms:

- **K-means Clustering:** Groups data into K clusters based on how close the points are to each other.
- **Hierarchical Clustering:** Creates clusters by building a tree step-by-step, either merging or splitting groups.
- **Density-Based Clustering (DBSCAN):** Finds clusters in dense areas and treats scattered

points as noise.

- **Mean-Shift Clustering:** Discovers clusters by moving points toward the most crowded areas.
- **Spectral Clustering:** Groups data by analysing connections between points using graphs.

2. Association Rule Learning

Association rule learning is also known as association rule mining is a common technique used to discover associations in unsupervised machine learning. This technique is a rule-based ML technique that finds out some very useful relations between parameters of a large data set. This technique is basically used for market basket analysis that helps to better understand the relationship between different products.

For e.g. shopping stores use algorithms based on this technique to find out the relationship between the sale of one product w.r.t to another's sales based on customer behavior. **Like if a customer buys milk, then he may also buy bread, eggs, or butter**. Once trained well, such models can be used to increase their sales by planning different offers.

Some common Association Rule Learning algorithms:

- **Apriori Algorithm:** Finds patterns by exploring frequent item combinations step-by-step.
- **FP-Growth Algorithm:** An Efficient Alternative to Apriori. It quickly identifies frequent patterns without generating candidate sets.
- **Eclat Algorithm:** Uses intersections of itemsets to efficiently find frequent patterns.
- **Efficient Tree-based Algorithms:** Scales to handle large datasets by organizing data in tree structures.

3. Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features in a dataset while preserving as much information as possible. This technique is useful for improving the performance of machine learning algorithms and for data visualization.

Here are some popular **Dimensionality Reduction algorithms**:

- **Principal Component Analysis (PCA):** Reduces dimensions by transforming data into uncorrelated principal components.
- **Linear DiscriminantAnalysis (LDA):** Reduces dimensions while maximizing class separability for classification tasks.
- **Non-negative Matrix Factorization (NMF):** Breaks data into non-negative parts to simplify representation.
- **Locally Linear Embedding (LLE):** Reduces dimensions while preserving the relationships between nearby points.
- **Isomap:** Captures global data structure by preserving distances along a manifold.

Imagine a dataset of 100 features about students (height, weight, grades, etc.). To focus on key traits, you

reduce it to just 2 features: height and grades, making it easier to visualize or analyze the data.

Challenges of Unsupervised Learning

Here are the key challenges of unsupervised learning:

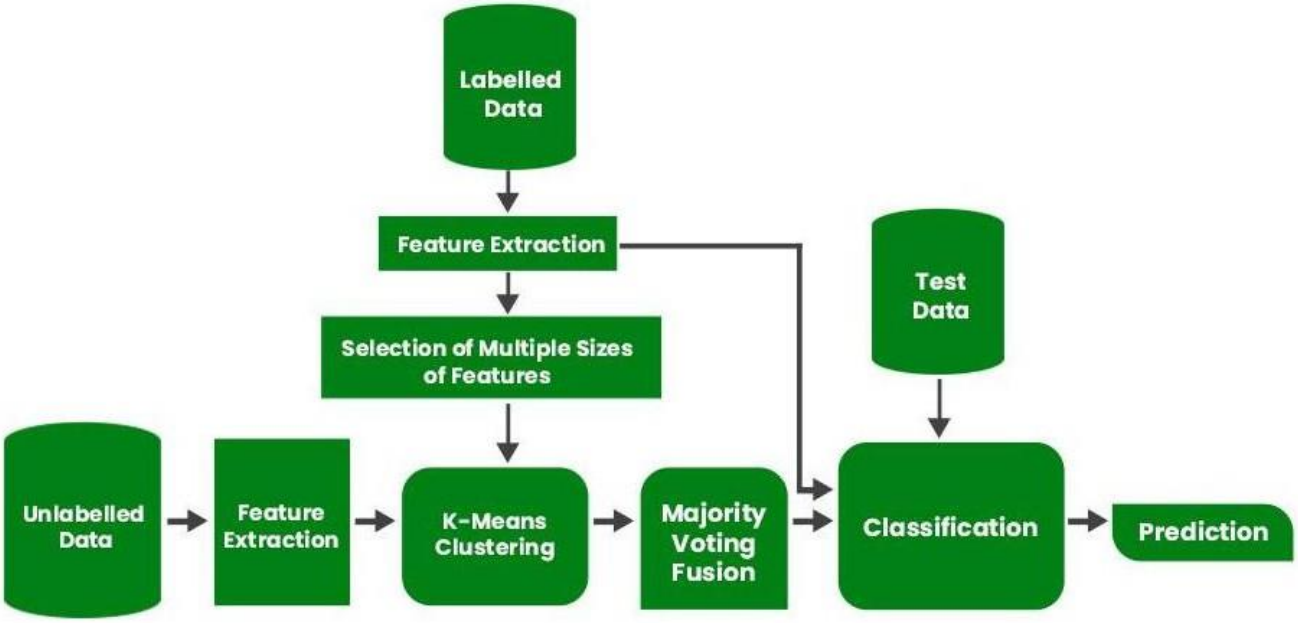
- **Noisy Data**: Outliers and noise can distort patterns and reduce the effectiveness of algorithms.
- **Assumption Dependence**: Algorithms often rely on assumptions (e.g., cluster shapes), which may not match the actual data structure.
- **Overfitting Risk**: Overfitting can occur when models capture noise instead of meaningful patterns in the data.
- **Limited Guidance**: The absence of labels restricts the ability to guide the algorithm toward specific outcomes.
- **Cluster Interpretability**: Results, such as clusters, may lack clear meaning or alignment with real-world categories.
- **Sensitivity to Parameters**: Many algorithms require careful tuning of hyperparameters, such as the number of clusters in k-means.
- **Lack of Ground Truth**: Unsupervised learning lacks labeled data, making it difficult to evaluate the accuracy of results.

Applications of Unsupervised learning

- **Customer Segmentation:** Algorithms cluster customers based on purchasing behaviour or demographics, enabling targeted marketing strategies.
- **Anomaly Detection:** Identifies unusual patterns in data, aiding fraud detection, cyber security, and equipment failure prevention.
- **Recommendation Systems**: Suggests products, movies, or music by analysing user behaviour and preferences.
- **Image and Text Clustering**: Groups similar images or documents for tasks like organization, classification, or content recommendation.
- **Social Network Analysis**: Detects communities or trends in user interactions on social media platforms.
- **Astronomy and Climate Science:** Classifies galaxies or groups weather patterns to support scientific research

SEMI-SUPERVISED LEARNING IN ML

Semi-supervised learning is a type of machine learning that falls in between supervised and unsupervised learning. It is a method that uses a small amount of labelled data and a large amount of unlabelled data to train a model. The goal of semi-supervised learning is to learn a function that can accurately predict the output variable based on the input variables, similar to supervised learning. However, unlike supervised learning, the algorithm is trained on a dataset that contains both labelled and unlabelled data.



Semi-Supervised Learning Flow Chart

Semi-supervised learning is particularly useful when there is a large amount of unlabelled data available, but it's too expensive or difficult to label all of it.

Intuitively, one may imagine the three types of learning algorithms as Supervised learning where a student is under the supervision of a teacher at both home and school, Unsupervised learning where a student has to figure out a concept himself and Semi-Supervised learning where a teacher teaches a few concepts in class and gives questions as homework which are based on similar concepts.

Examples of Semi-Supervised Learning

- **Text classification**: In text classification, the goal is to classify a given text into one or more predefined categories. Semi-supervised learning can be used to train a text classification model using a small amount of labelled data and a large amount of unlabelled text data.

- **Image classification:** In image classification, the goal is to classify a given image into one or more predefined categories. Semi-supervised learning can be used to train an image classification model using a small amount of labelled data and a large amount of unlabelled image data.
- **Anomaly detection: In** anomaly detection, the goal is to detect patterns or observations that are unusual or different from the norm

Assumptions followed by Semi-Supervised Learning

A Semi-Supervised algorithm assumes the following about the data

1. **Continuity Assumption:** The algorithm assumes that the points which are closer to each other are more likely to have the same output label.
2. **Cluster Assumption:** The data can be divided into discrete clusters and points in the same cluster are more likely to share an output label.
3. **Manifold Assumption:** The data lie approximately on a manifold of a much lower dimension than the input space. This assumption allows the use of distances and densities which are defined on a manifold.

Applications of Semi-Supervised Learning

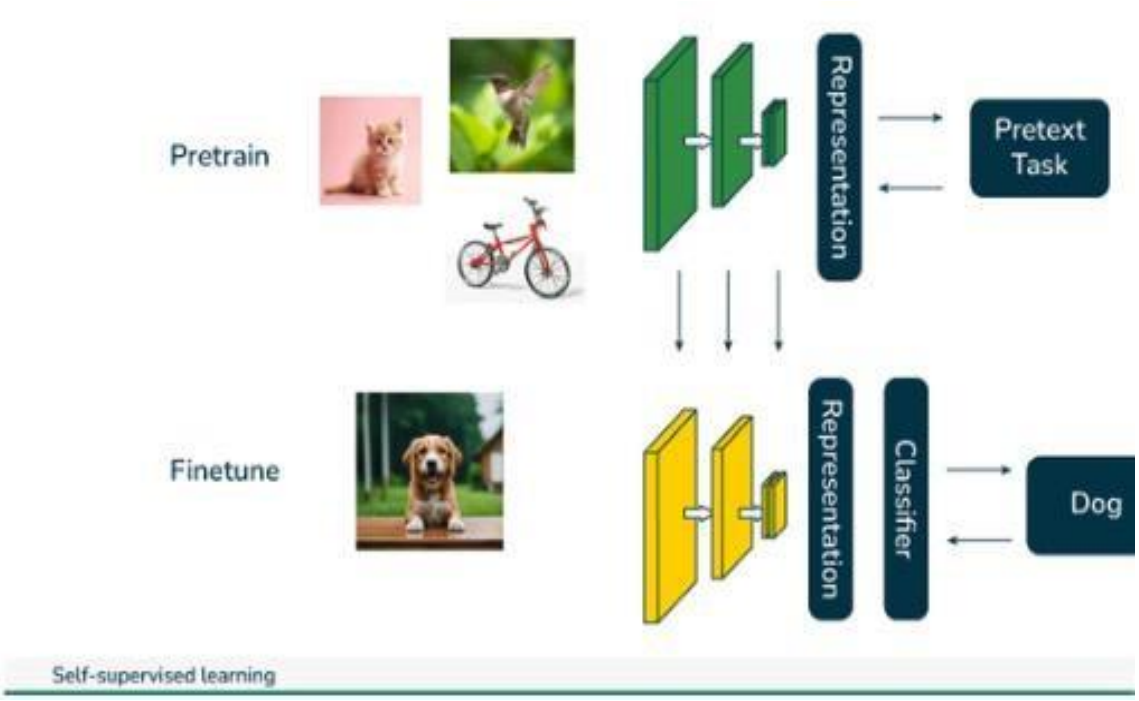
1. **Speech Analysis:** Since labelling audio files is a very intensive task, Semi-Supervised learning is a very natural approach to solve this problem.
2. **Internet Content Classification:** Labelling each webpage is an impractical and unfeasible process and thus uses Semi-Supervised learning algorithms. Even the Google search algorithm uses a variant of Semi-Supervised learning to rank the relevance of a webpage for a given query.
3. **Protein Sequence Classification:** Since DNA strands are typically very large in size, the rise of Semi-Supervised learning has been imminent in this field.

Disadvantages of Semi-Supervised Learning

The most basic disadvantage of any Supervised Learning algorithm is that the dataset has to be hand-labelled either by a Machine Learning Engineer or a Data Scientist. This is a very costly process, especially when dealing with large volumes of data. The most basic disadvantage of any Unsupervised Learning is that its application spectrum is limited.

SELF-SUPERVISED LEARNING (SSL)

Self-supervised learning is a deep learning methodology where a **model is pre-trained using unlabelled data** and the data labels are generated automatically, which are further used in subsequent iterations as ground truths. The fundamental idea for self-supervised learning is to create supervisory signals by making sense of the unlabelled data provided to it in an unsupervised fashion on the first iteration. Then, the model uses the high-confidence data labels among those generated to train the model in subsequent iterations like the supervised learning model via back propagation. The only difference is, the data labels used as ground truths in every iteration are changed.



How to train a Self-Supervised Learning Model in ML

1. **Select a property of the data to predict:** To predict the next word in a sentence, the orientation of an object in an image, or the speaker of an audio clip.
2. **Define a loss function:** The loss function measures the model's performance on the task of predicting the property of the data. It should be designed to encourage the model to learn useful features and representations of the data that are relevant to the task.
3. **Train the model:** The model is trained on a large dataset by minimizing the loss function. This is typically done using an optimization algorithm, such as stochastic gradient descent (SGD) or Adam.
4. **Fine-tune the model:** Once the model has been trained, it can be fine-tuned on a specific task by adding a few labelled examples and fine-tuning the model's weights using supervised learning techniques. This allows the model to learn task-specific features and further improve its performance on the target task.

Application of SSL in Computer Vision

Image and video recognition: Self-supervised learning has been used to improve the performance of image and video recognition tasks, such as object recognition, image classification, and video classification. For example, a self-supervised learning model might be trained to predict the location of an object in an image given the surrounding pixels to classify a video as depicting a particular action.

Application of SSL in Natural Language Processing

- **Language understanding:** Self-supervised learning has been used to improve the performance of natural language processing (NLP) tasks, such as machine translation, language modelling, and text classification. For example, a self-supervised learning model might be trained to predict the nex t word in a sentence given the previous words, or to classify a sentence as positive or negative.
- **Speech recognition:** Self-supervised learning has been used to improve the performance of speech recognition tasks, such as transcribing audio recordings into text. For example, a self-supervised learning model might be trained to predict the speaker of an audio clip based on the characteristics of their voice.

Self-Supervised Learning Techniques

- **Pretext tasks:** Pretext tasks are auxiliary tasks designed to solve using the inherent structure of the data, but are also related to the main task. For example, the model might be trained on a pretext task of predicting the rotation of an image, with the goal of improving performance on the main task of image classification.
- **Contrastive learning:** Contrastive Learning is a self-supervised learning technique that involves training a model to distinguish between a noisy version of the data to a clean version. The model is trained to distinguish between the two, with the goal of learning a robust representation of noise.

Advantages of Self-Supervised Learning

- **Reduced Reliance on Labelled Data:** One of the main benefits of self-supervised learning is that it allows a model to learn useful features and representations of the data without the need for large amounts of labelled data. This can be particularly useful in situations where it is expensive or time-consuming to obtain labelled data, or where there is a limited amount of labelled data available.
- **Improved Generalization:** Self-supervised learning can improve the generalization performance of a model, meaning that it is able to make more accurate predictions on unseen data. This is because self-supervised learning allows a model to learn from the inherent structure of the data, rather than just memorizing specific examples.

- **Scalability:** Self-supervised learning can be more scalable than supervised learning, as it allows a model to learn from a larger dataset without the need for human annotation. This can be particularly useful in situations where the amount of data is too large to be labelled by humans.
- **Transfer Learning:** Self-supervised learning can be useful for transfer learning, which involves using a model trained on one task to improve performance on a related task. By learning useful features and representations of the data through self-supervised learning, a model can be more easily adapted to new tasks and environments.

Limitations of Self-Supervised Learning

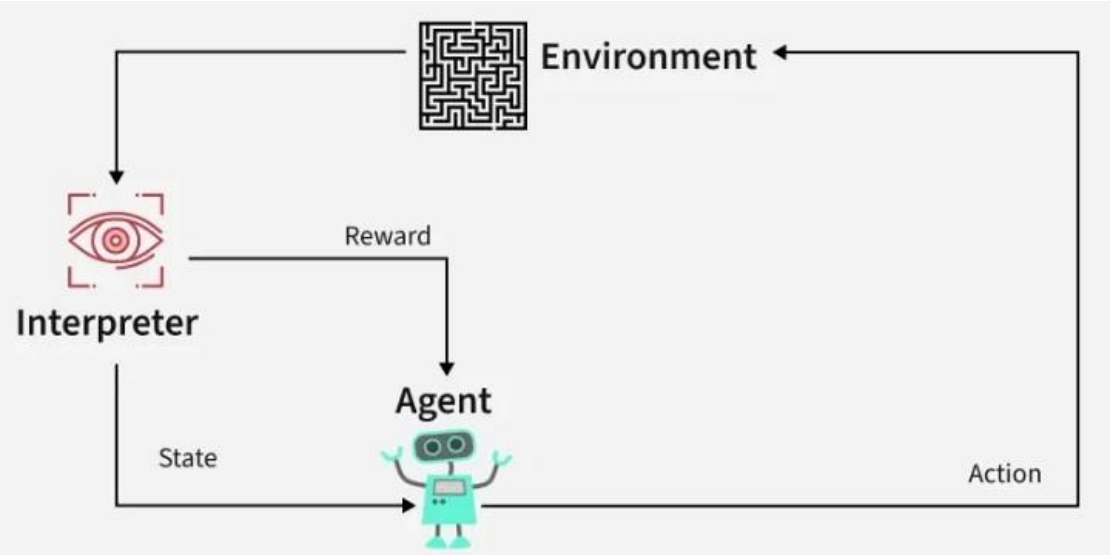
- **Quality of supervision signal:** One of the main limitations of self-supervised learning is that the quality of the supervision signal can be lower than in supervised learning. This is because the supervision signal is derived from the data itself, rather than being explicitly provided by a human annotator. As a result, the supervision signal may be noisy or incomplete, which can lead to lower performance on the task.
- **Limited to certain types of tasks:** Self-supervised learning may not be as effective for tasks where the data is more complex or unstructured.
- **The complexity of training:** Some self-supervised learning techniques can be more complex to implement and train than supervised learning techniques. For example, contrastive learning and unsupervised representation learning can be more challenging to implement and tune than supervised learning methods.

Differences between Supervised, Unsupervised, and Self-Supervised Learning

Supervised	Unsupervised	Self-Supervised
Supervised learning is a type of machine learning where the model is trained on labeled data, meaning that the input data is accompanied by its corresponding correct output.	Unsupervised learning is a type of machine learning where the model is trained on unlabeled data, meaning that the input data does not have a corresponding correct output.	Self-supervised learning is a type of machine learning that falls between supervised and unsupervised learning. It is a form of unsupervised learning where the model is trained on unlabeled data, but the goal is to learn a specific task or representation of the data that can be used in a downstream supervised learning task.
The goal of supervised learning is to learn a mapping from input data to the correct output.	The goal of unsupervised learning is to learn patterns or structures in the input data without the guidance of a labeled output.	In self-supervised learning, the model learns to predict certain properties of the input data, such as a missing piece or its rotation angle. This learned representation can then be used to initialize a supervised learning model, providing a good starting point for fine-tuning on a smaller labeled dataset.
Common examples of supervised learning include image classification, object detection, and Natural Language Processing tasks.	Common examples of unsupervised learning include clustering, dimensionality reduction, and anomaly detection.	A common example of self-supervised learning is the task of image representation learning, sentiment analysis, question answering, and machine translation.

REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a branch of machine learning that focuses on how agents can learn to make decisions through trial and error to maximize cumulative rewards. RL allows machines to learn by interacting with an environment and receiving feedback based on their actions. This feedback comes in the form of **rewards or penalties**.



Reinforcement Learning revolves around the idea that an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent performs actions and receives feedback to optimize its decision-making over time.

- **Agent:** The decision-maker that performs actions.
- **Environment:** The world or system in which the agent operates.
- **State:** The situation or condition the agent is currently in.
- **Action:** The possible moves or decisions the agent can make.
- **Reward:** The feedback or result from the environment based on the agent's action.

How Reinforcement Learning Works?

The RL process involves an agent performing actions in an environment, receiving rewards or penalties based on those actions, and adjusting its behaviour accordingly. This loop helps the agent improve its decision-making over time to maximize the **cumulative reward**.

Here's a breakdown of RL components:

- **Policy:** A strategy that the agent uses to determine the next action based on the current state.

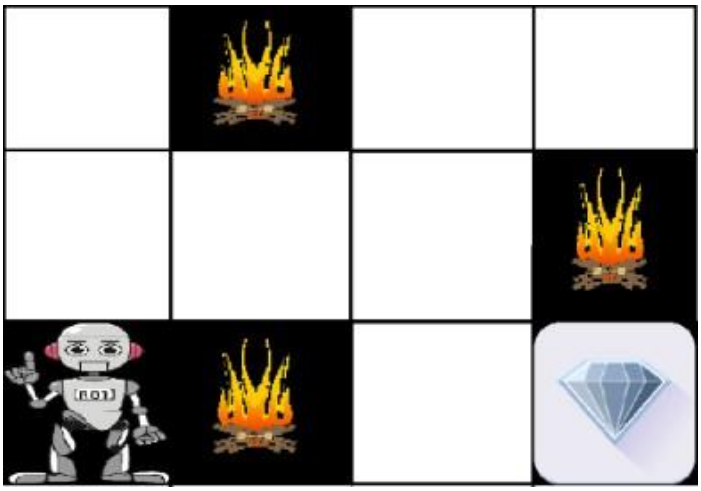
- **Model of the Environment:** A representation of the environment that predicts future states and rewards, aiding in planning.
- **Reward Function:** A function that provides feedback on the actions taken, guiding the agent towards its goal.
- **Value Function:** Estimates the future cumulative rewards the agent will receive from a given state.

Reinforcement Learning Example: Navigating a Maze

Imagine a robot navigating a maze to reach a diamond while avoiding fire hazards. The goal is to find the optimal path with the least number of hazards while maximizing the reward:

- Each time the robot moves correctly, it receives a reward.
- If the robot takes the wrong path, it loses points.

The robot learns by exploring different paths in the maze. By trying various moves, it evaluates the rewards and penalties for each path. Over time, the robot determines the best route by selecting the actions that lead to the highest cumulative reward.



The robot's learning process can be summarized as follows:

- **Exploration:** The robot starts by exploring all possible paths in the maze, taking different actions at each step (e.g., move left, right, up, or down).
- **Feedback:** After each move, the robot receives feedback from the environment:
- A positive reward for moving closer to the diamond.
- A penalty for moving into a fire hazard.
- **Adjusting Behaviour:** Based on this feedback, the robot adjusts its behaviour to maximize the cumulative reward, favouring paths that avoid hazards and bring it closer to the diamond.
- Optimal Path: Eventually, the robot discovers the optimal path with the least number of hazards and the highest reward by selecting the right actions based on past experiences.

TYPES OF REINFORCEMENTS IN RL

Positive Reinforcement

Positive Reinforcement is defined as when an event, occurs due to a particular behaviour, increases the strength and the frequency of the behaviour. In other words, it has a positive effect on behaviour.

Advantages: Maximizes performance, helps sustain change over time.

Disadvantages: Overuse can lead to excess states that may reduce effectiveness.

1. Negative Reinforcement

Negative Reinforcement is defined as strengthening of behavior because a negative condition is stopped or avoided.

Advantages: Increases behaviour frequency ensures a minimum performance standard.

Disadvantages: It may only encourage just enough action to avoid penalties.

Application of Reinforcement Learning

- **Robotics:** RL is used to automate tasks in structured environments such as manufacturing, where robots learn to optimize movements and improve efficiency.
- **Game Playing:** Advanced RL algorithms have been used to develop strategies for complex games like chess, Go, and video games, outperforming human players in many instances.
- **Industrial Control:** RL helps in real-time adjustments and optimization of industrial operations, such as refining processes in the oil and gas industry.
- **Personalized Training Systems:** RL enables the customization of instructional content based on an individual's learning patterns, improving engagement and effectiveness.

Advantages of Reinforcement Learning

- **Solving Complex Problems:** RL is capable of solving highly complex problems that cannot be addressed by conventional techniques.
- **Error Correction:** The model continuously learns from its environment and can correct errors that occur during the training process.
- **Direct Interaction with the Environment:** RL agents learn from real-time interactions with their environment, allowing adaptive learning.
- **Handling Non-Deterministic Environments:** RL is effective in environments where outcomes are uncertain or change over time, making it highly useful for real-world applications.

Disadvantages of Reinforcement Learning

- **Not Suitable for Simple Problems:** RL is often overkill for straightforward tasks where simpler algorithms would be more efficient.
- **High Computational Requirements:** Training RL models requires a significant amount of data and computational power, making it resource-intensive.
- **Dependency on Reward Function:** The effectiveness of RL depends heavily on the design of the reward function. Poorly designed rewards can lead to suboptimal or undesired behaviours.
- **Difficulty in Debugging and Interpretation:** Understanding why an RL agent makes certain decisions can be challenging, making debugging and troubleshooting complex.

Bayes' Theorem :

Theorem Statement

Bayes' Theorem states that:The probability of an event A occurring given that event B has already occurred is equal to the probability of event B occurring given event A, multiplied by the prior probability of A, and divided by the probability of B.

Mathematical Formula

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

Where

- **P(A|B)** = Posterior Probability (Probability of A given B)
- **P(B|A)** = Likelihood (Probability of B given A)
- **P(A)** = Prior Probability (Probability of A before observing B)
- **P(B)** = Evidence (Probability of B)

Proof (Derivation) of Bayes' Theorem

From the definition of conditional probability,

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} (1)$$

Similarly,

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} (2)$$

From Equation (2),

$$P(A \cap B) = P(B \mid A) \times P(A)$$

Substitute this into Equation (1):

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

Hence,

$$\boxed{P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}}$$

This is called **Bayes' Theorem**.

Real world problem:

A hospital uses a medical test to check whether a person has a disease. Only 1% of people actually have the disease. The test correctly gives a positive result to 99% of people who have the disease, but it also gives a positive result to 5% of healthy people by mistake. If a person receives a positive test result, what is the probability that the person actually has the disease? This problem can be solved using Bayes' Theorem.

Step 1: Define A and B

- A = Person has the disease.
- B = Test result is positive.

Step 2: Write the given probabilities

Quantity	**Value**
P(A)	0.01
P(B|A)	0.99
P(B)	?

Step 3: Calculate P(B)

The probability of a positive test is:

P(B) = Positive from diseased + Positive from healthy

Healthy people = 1 − 0.01 = 0.99

So,

P(B) = (0.99 × 0.01) + (0.05 × 0.99)

0.0099 + 0.0495

= 0.0594

Step 4: Apply Bayes' Formula

Substitute the values

P(A|B) = [P(B|A) × P(A)] / P(B)

= (0.99 × 0.01) / 0.0594

0.0099 / 0.0594

= 0.1667

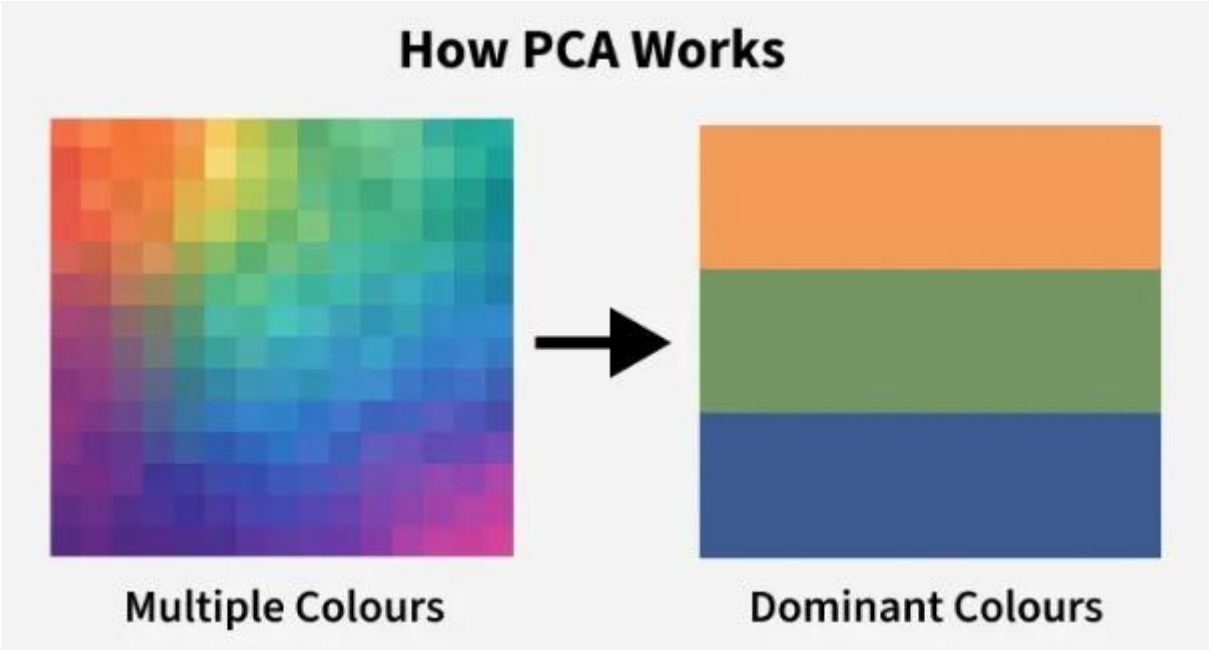
Final Answer

P(A|B) = 0.1667 ≈ 16.7%

The probability that the person actually has the disease after testing positive is 16.7%.

Principal Component Analysis (PCA)

PCA (Principal Component Analysis) is a dimensionality reduction technique and helps us to reduce the number of features in a dataset while keeping the most important information. It changes complex datasets by transforming correlated features into a smaller set of uncorrelated components.



It helps us to remove redundancy, improve computational efficiency and make data easier to visualize and analyze.

How Principal Component Analysis Works

PCA uses linear algebra to transform data into new features called principal components. It finds these by calculating eigenvectors (directions) and eigenvalues (importance) from the covariance matrix. PCA selects the top components with the highest eigenvalues and projects the data onto them simplify the dataset.

Imagine you're looking at a messy cloud of data points like stars in the sky and want to simplify it. PCA helps you find the "most important angles" to view this cloud so you don't miss the big patterns. Here's how it works step by step:

Step 1: Collect the Dataset

Gather the dataset containing multiple features (variables). PCA works on numerical data, so categorical data should be converted into numerical form before applying PCA.

Step 2: Standardize the Data

Standardize all features so that they have a **mean of 0** and a **standard deviation of 1**. This ensures that features with larger values do not dominate those with smaller values.

Formula:

$$Z = \frac{X - \mu}{\sigma}$$

Where:

- X= Original value
- μ= Mean of the feature
- σ= Standard deviation

Step 3: Compute the Covariance Matrix

Calculate the covariance matrix to determine how the features are related to one another. A positive covariance indicates that variables increase together, while a negative covariance indicates that one variable increases as the other decreases.

Formula:

$$C = \frac{1}{n-1} Z^T Z$$

Where:

- C= Covariance matrix
- Z= Standardized data
- n= Number of observations

Step 4: Calculate Eigenvalues and Eigenvectors

Find the **eigenvalues** and **eigenvectors** of the covariance matrix.

Formula:

$$Cv = \lambda v$$

Where:

- C= Covariance matrix
- v= Eigenvector
- λ= Eigenvalue
- **Eigenvalues** indicate the amount of variance explained by each principal component.
- **Eigenvectors** determine the direction of the principal components.

Step 5: Sort the Eigenvalues

Arrange the eigenvalues in **descending order**. The principal components with the highest eigenvalues contain the maximum information (variance).

Example:

Principal Component	Eigenvalue
PC1	5.6
PC2	2.8
PC3	1.1
PC4	0.5

Step 6: Select the Principal Components

Choose the top **k principal components** that explain most of the total variance. These selected components represent the reduced feature space.

Feature Vector Formula:

$$W = [v_1, v_2, \dots, v_k]$$

Where:

- W= Feature vector
- $v_1, v_2, \dots, v_k$= Selected eigenvectors

Step 7: Transform the Original Data

Project the original standardized data onto the selected principal components to obtain the reduced dataset.

Formula:

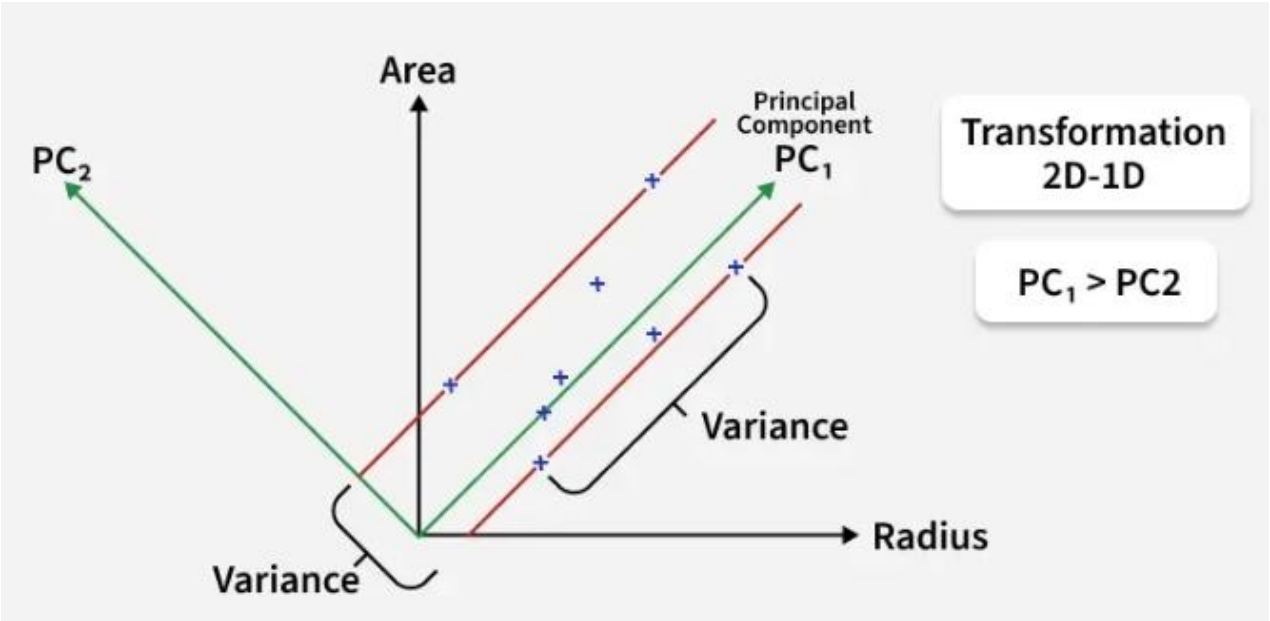
$$Y = ZW$$

Where:

- Y= Reduced-dimensional data
- Z= Standardized data
- W= Feature vector

Step 8: Obtain the Reduced Dataset

The transformed dataset contains fewer features while preserving most of the original information. This reduced dataset is then used for visualization, data analysis, or training machine learning models.



Transform this 2D dataset into a 1D representation while preserving as much variance as possible

In the above image the original dataset has two features "Radius" and "Area" represented by the black axes. PCA identifies two new directions: PC_1 and PC_2 which are the principal components.

- These new axes are rotated versions of the original ones. PC_1 captures the maximum variance in the data meaning it holds the most information while PC_2 captures the remaining variance and is perpendicular to PC_1.
- The spread of data is much wider along PC_1 than along PC_2. This is why PC_1 is chosen for dimensionality reduction. By projecting the data points (blue crosses) onto PC_1 we effectively transform the 2D data into 1D and retain most of the important structure and patterns.

Problem:

Problem:

Given the following data, use PCA to reduce the dimension from 2 to 1.

Feature	E_1	E_2	E_3	E_4
x	4	8	13	7
y	11	4	5	14

Sol: Step 1 : No. of Features $n = 2$

No. of Examples $N = 4$

Step 2 : Find the mean

$$\bar{x} = \frac{4+8+13+7}{4} = \frac{32}{4}$$

$$\boxed{\bar{x} = 8}$$

$$\bar{y} = \frac{11+4+5+14}{4} = \frac{34}{4}$$

$$\boxed{\bar{y} = 8.5}$$

Step 3: Compute the co-variance matric ordered pairs are:

$(x,x), (x,y), (y,x), (y,y)$

(i) Co-variance of all pairs

$$\boxed{Co(x,y) = \frac{1}{N-1}\sum_{K=1}^{N}(x_{iK} - \bar{x})(y_{jK} - \bar{y})}$$

$$Co(x,x) = \frac{1}{N-1}\sum_{K=1}^{N}(x_{iK} - \bar{x})(x_{jK} - \bar{x})$$

$$= \frac{1}{4-1}\left((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right)$$

$$= \frac{1}{3}\left((-4)^2 + (0)^2 + (5)^2 + (-1)^2\right)$$

$$= \frac{1}{3}(16+0+25+1)$$

$$= \frac{1}{3}(16+26)$$

$= \frac{42}{3}$

$\boxed{Co(x,x) = 14}$

$Co(x,y) = \frac{1}{N-1}\sum_{K=1}^{N}(x_{iK}-\bar{x})(y_{jK}-\bar{y})$

$= \frac{1}{3}\left((4-8)(11-8.5)+(8-8)(4-8.5)+(13-8)(5-8.5)+(7-8)(14-8.5)\right)$

$= \frac{1}{3}\left((-4)(2.5)+(0)(-4.5)+(5)(-3.5)+(-1)(5.5)\right)$

$= \frac{1}{3}(-10)+0+(-17.5)+(-5.5)$

$= \frac{1}{3}(-33)$

$= \frac{-33}{3}$

$\boxed{(x,y) = -11}$ similar to (y,x)

$Co(y,y) = \frac{1}{N-1}\sum_{K=1}^{N}(y_{jK}-\bar{y})(y_{jK}-\bar{y})$

$= \frac{1}{4-1}\left[(11-8.5)^2+(4-8.5)^2+(5-8.5)^2+(14-8.5)^2\right]$

$= \frac{1}{3}\left[(2.5)^2+(-4.5)^2+(-3.5)^2+(5.5)^2\right]$

$= \frac{1}{3}(6.25+20.25+12.25+30.25)$

$= \frac{1}{3}(69)$

$= \frac{69}{3}^{23}$

$\boxed{y,y = 23}$

(ii) Co-variance matrix

$$S = \begin{bmatrix} Co(x,x) & Co(x,y) \\ Co(y,x) & Co(y,y) \end{bmatrix}$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 4: Eigen value, Eigen vector, Normalized vector

i) Eigen value:

$(S-\lambda I) = 0$

$$\begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$\begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} = 0$$

Applying $\boxed{AD-BC=0}$

$(14-\lambda)(23-\lambda) - (-11)(-11) = 0$

$(14)(23) + (-14\lambda) - 23\lambda + \lambda^2 = 0$

$\lambda^2 - 37\lambda + 201 = 0$

$a\lambda^2 + b\lambda + c = 0$

Again applying Quadratic equation formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

$\lambda = 30.3649, 6.6151$

$\lambda_1 = 30.3649$ & $\lambda_2 = 6.6151$

$\lambda_1 > \lambda_2$

$\boxed{\lambda_1 = 30.3649}$ $\boxed{\lambda_2 = 6.6151}$

(ii) Eigen vector

$[S - \lambda I][u] = 0$

$$\begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} (14-\lambda)u_1 + (-11)u_2 \\ -11(u_1) + (23-\lambda)u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$(14-\lambda)u_1 - 11u_2 = 0 \rightarrow (1)$

$-11u_1 + (23-\lambda)u_2 = 0 \rightarrow (2)$

From eq (1)

$(14-\lambda)u_1 - 11u_2 = 0$

Assume $\boxed{0 = t}$

$(14-\lambda)u_1 - 11u_2 = t$

Again assume that $\boxed{t = 1}$

$(14-\lambda)u_1 - 11u_2 = 1$

$(14-\lambda)u_1 = 11u_2$

$\frac{u_1}{11} = \frac{u_2}{14-\lambda} = t$

where $t = 1$

$\frac{u_1}{11} = 1$ & $\frac{u_2}{14-\lambda} = 1$

$\boxed{u_1 = 11, \; u_2 = 14 - \lambda}$

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 11 \\ 14+\lambda \end{bmatrix}$$

$$= \begin{bmatrix} 11 \\ 14 - 30.3849 \end{bmatrix}$$

$$u = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix}$$

(iii) Normalize Vector

$$e_1 = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix} \quad \left[\because \begin{bmatrix} a \\ b \end{bmatrix} \right]$$

According to above matrix

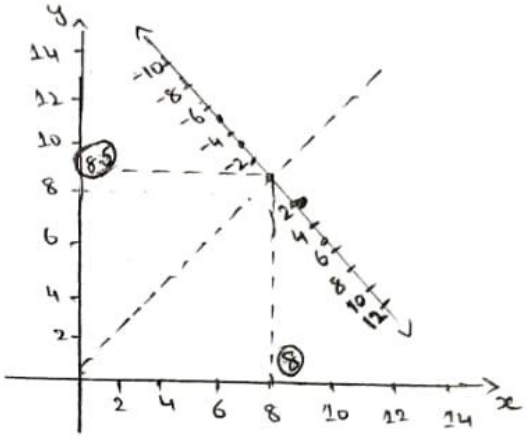
$\begin{bmatrix} a \\ b \end{bmatrix}$ we can use $\frac{a}{\sqrt{a^2+b^2}}$

$$e_1 = \begin{bmatrix} \frac{11}{\sqrt{11^2 + (-16.3849)^2}} \\ \frac{-16.3849}{\sqrt{11^2 + (-16.3849)^2}} \end{bmatrix} \quad \left[\because \begin{bmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \end{bmatrix}\right]$$

$$e_1 = \begin{bmatrix} \frac{11}{19.785} \\ \frac{-16.3849}{19.785} \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$



Step 5: New Data Set

First	E_1	E_2	E_3	E_4
Features	P_{11}	P_{12}	P_{13}	P_{14}

$$P_{11} = e_1^T \begin{bmatrix} 4-8 \\ 11-8.5 \end{bmatrix} \quad \because \begin{bmatrix} x-\bar{x} \\ y-\bar{y} \end{bmatrix}$$

$$= (0.5574 - 0.8303) \begin{bmatrix} -4 \\ 2.5 \end{bmatrix}$$

$$\boxed{P_{11} = -4.3052}$$

$$P_{12} = e_2^T \begin{bmatrix} 8-8 \\ 4-8.5 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 - 0.8303 \end{bmatrix} \begin{bmatrix} 0 \\ -4.5 \end{bmatrix}$$

$$\boxed{P_{12} = 3.7361}$$

$$P_{13} = e_3^T \begin{bmatrix} 13-8 \\ 5-8.5 \end{bmatrix}$$

$$= (0.5574 - 0.8303) \begin{bmatrix} 5 \\ 3.5 \end{bmatrix}$$

$$\boxed{P_{13} = 5.6928}$$

$$P_{14} = e_4^T \begin{bmatrix} 7-8 \\ 14-8.5 \end{bmatrix}$$

$$= (0.5374 - 0.8303) \begin{bmatrix} -1 \\ 5.5 \end{bmatrix}$$

$$\boxed{P_{14} = -5.1238}$$

Advantages

- Multicollinearity Handling: Creates new, uncorrelated variables to address issues when original features are highly correlated.
- Noise Reduction: Eliminates components with low variance enhance data clarity.
- Data Compression: Represents data with fewer components reduce storage needs and speeding up processing.
- Outlier Detection: Identifies unusual data points by showing which ones deviate significantly in the reduced space.

Disadvantages

- Interpretation Challenges: The new components are combinations of original variables which can be hard to explain.
- Data Scaling Sensitivity: Requires proper scaling of data before application or results may be misleading.
- Information Loss: Reducing dimensions may lose some important information if too few components are kept.
- Assumption of Linearity: Works best when relationships between variables are linear and may struggle with non-linear data.
- Computational Complexity: Can be slow and resource-intensive on very large datasets.

Case Study

Diagnosing Crop Disease Using Machine Learning

Diagnosing crop disease using Machine Learning (ML) is the process of using machine learning algorithms to automatically identify and classify diseases affecting crops based on images, sensor data, or environmental information. The machine learning model learns from previously collected data and predicts the type of disease present in a crop, helping farmers take timely preventive and corrective actions.

Introduction

Crop diseases are one of the major causes of reduced agricultural productivity and financial losses for farmers. Traditional disease diagnosis relies on manual inspection by agricultural experts, which is time-consuming and may not always be accurate. Machine Learning provides an intelligent solution by analyzing crop images and other agricultural data to detect diseases quickly and accurately. Early disease detection enables farmers to apply appropriate treatments, reduce crop damage, improve yield, and minimize the excessive use of pesticides.

Working of Diagnosing Crop Disease Using Machine Learning

Step 1: Data Collection

The first step is to collect a large dataset of healthy and diseased crop images. The images can be captured using smartphones, digital cameras, drones, or agricultural monitoring systems. In addition to images, environmental data such as temperature, humidity, rainfall, and soil moisture may also be collected to improve prediction accuracy.

Step 2: Data Preprocessing

The collected data is cleaned and prepared before training the machine learning model. Image preprocessing includes resizing images, removing noise, adjusting brightness, normalizing pixel values, and eliminating duplicate or missing data. Data augmentation techniques such as rotation, flipping, and scaling are also used to increase the diversity of the training dataset.

Step 3: Feature Extraction

After preprocessing, important features are extracted from the crop images. These features include leaf color, texture, shape, size, vein patterns, disease spots, and lesion characteristics. In traditional machine learning, these features are extracted manually, whereas deep learning models such as Convolutional Neural Networks (CNNs) automatically learn the important features from the images.

Step 4: Model Training

The extracted features are used to train a machine learning model. The model learns the relationship between the input features and the corresponding disease labels by analyzing a large number of labeled examples. Common algorithms used for crop disease diagnosis include Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, and Convolutional Neural Networks (CNNs).

Step 5: Model Testing and Evaluation

Once the training process is completed, the model is tested using a separate dataset that was not used during training. The performance of the model is evaluated using various metrics such as Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix. These evaluation measures help determine how effectively the model identifies crop diseases.

Step 6: Disease Prediction

When a new crop image is provided to the trained model, the model analyzes the image and predicts whether the crop is healthy or affected by a disease. If a disease is detected, the model identifies the specific disease, such as Early Blight, Late Blight, Leaf Spot, or Powdery Mildew.

Step 7: Recommendation

After predicting the disease, the system provides recommendations for disease management. These recommendations may include suitable pesticides or fungicides, proper irrigation practices, nutrient management techniques, and preventive measures to control the spread of the disease and improve crop health.

Advantages

- Machine Learning detects crop diseases at an early stage, reducing crop damage.
- It provides faster and more accurate disease diagnosis than manual inspection.

- It helps farmers make informed decisions about disease treatment.
- It reduces unnecessary pesticide usage, lowering production costs.
- It improves agricultural productivity and crop quality.
- It supports precision farming by enabling timely interventions.
- It can be integrated with smartphones, drones, and IoT devices for real-time monitoring.

Limitations

- Machine learning models require a large amount of high-quality labeled data for training.
- The prediction accuracy depends on the quality of the input images.
- Poor lighting conditions or low-resolution images may reduce model performance.
- Similar diseases with overlapping symptoms may be difficult to distinguish.
- Deep learning models require significant computational resources for training.
- The model must be updated regularly to recognize newly emerging crop diseases.

Applications

- Detection of diseases in rice, wheat, maize, cotton, tomato, potato, and other crops.
- Smart farming and precision agriculture.
- Automated crop monitoring using drones and satellites.
- Greenhouse crop health monitoring.
- Mobile applications for farmers to identify crop diseases.
- Agricultural research and decision support systems.

Conclusion

Machine Learning has transformed crop disease diagnosis by providing a fast, accurate, and automated method for identifying plant diseases. By analyzing crop images and environmental data, machine learning models help farmers detect diseases at an early stage, recommend appropriate treatments, and reduce crop losses. This technology improves agricultural productivity, supports sustainable farming practices, and contributes to food security.

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, “Machine Learning”, Wiley Publisher,2019.

2.SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, “Machine Learning”,Pearson,2019.

3. Alpaydin Ethem, “Introduction to Machine Learning”, 3rd Edition, PHI learningprivate limited, 2019.

REFERENCE BOOKS:

1. “Machine Learning: A Probabilistic Perspective”, 2012, Kevin P. Murphy, MIT Press.

2. “Pattern Recognition and Machine Learning”, 2007, Christopher Bishop, Springer.

3. “Introduction to Machine Learning”, MIT Press, 3/e, 2014, Ethem Alpaydin.

Question Bank:

PART-A		
Q.No.	**Questions**	**Blooms Taxonomy Level**
1.	Define Machine Learning	L1
2.	List the types of Machine Learning	L1
3.	State any two applications of Machine Learning.	L1
4.	Mention two challenges of Machine Learning.	L1
5.	Define Probability.	L1
6.	State Bayes' Theorem.	L1
7.	What is Data Preprocessing?	L1
8.	Define Data Cleaning.	L1
9.	What is Data Integration?	L1
10.	Define Principal Component Analysis (PCA).	L2
PART-A		
1.	Explain the basic concepts and workflow of Machine Learning with suitable examples.	L2
2.	Compare Supervised, Unsupervised, and Reinforcement Learning with examples and applications	L3
3.	Discuss the applications and challenges of Machine Learning	L2
4.	xplain the basics of Probability Theory and Bayes' Theorem ith a suitable example.	L3
5.	Describe the steps involved in Data Preprocessing.	L2
6.	Explain Data Cleaning, Data Integration, Data Transformation and Data Reduction	L2
7.	Explain Principal Component Analysis (PCA) and its advantages in dimensionality reduction.	L3
8.	Analyze the importance of Data Preprocessing in Machine Learning	L4

L4 – Analyze

UNIT II
MODEL EVALUATION AND FEATURE ENGINEERING

"Most of the gains in machine learning come from better features, not better algorithms." — Pedro Domingos

Unit Overview

Model evaluation and feature engineering are essential stages in the machine learning lifecycle. After preprocessing the data, selecting an appropriate model and evaluating its performance are necessary to ensure accurate and reliable predictions. This unit introduces techniques for selecting, training, and evaluating machine learning models. It also explains how feature engineering improves model performance through feature transformation and feature subset selection. Understanding these concepts enables learners to build efficient, interpretable, and high-performing machine learning models.

Objectives of the Unit

After studying this unit, students will be able to:

- Understand the process of selecting an appropriate machine learning model.
- Learn how to train machine learning models using training data.
- Understand model representation and interpretability.
- Evaluate the performance of machine learning models using various metrics.
- Apply techniques to improve model performance.
- Understand the fundamentals of feature engineering.
- Learn different feature transformation techniques.
- Understand feature subset selection methods for improving model efficiency.
- Build machine learning models that are accurate, efficient, and interpretable.

Learning Outcomes

At the end of this unit, students will be able to:

- Explain the process of selecting a suitable machine learning model.
- Train machine learning models using appropriate datasets.
- Interpret machine learning models and explain their predictions.
- Evaluate model performance using suitable evaluation metrics.
- Improve model performance through optimization techniques.
- Apply feature engineering techniques to enhance prediction accuracy.
- Perform feature transformation and feature subset selection.
- Develop efficient machine learning models with reduced complexity.

Importance of Studying this Unit

Studying this unit is important because:

- It helps in choosing the most appropriate machine learning algorithm for a given problem.
- Proper model evaluation ensures reliable and accurate predictions.
- Feature engineering significantly improves model performance.

- Selecting relevant features reduces computational cost and training time.
- Model interpretability helps users understand how predictions are made.
- Performance improvement techniques help reduce errors and increase accuracy.
- These concepts are widely used in healthcare, finance, marketing, agriculture, cybersecurity, and recommendation systems.

Key Concepts

- **Selecting a Model:** Choosing the most suitable machine learning algorithm based on the problem, data, and performance requirements.
- **Training a Model:** Training is the process of enabling a machine learning model to learn patterns from training data.
- **Model Representation:** Model representation defines how a machine learning model stores the learned knowledge internally.
- **Model Interpretability:** Model interpretability is the ability to understand and explain how a machine learning model makes predictions.
- **Evaluating Performance of a Model:** Model evaluation measures how accurately a trained model performs on unseen data.
- **Improving Performance of a Model:** Model performance is improved using techniques such as hyperparameter tuning, cross-validation, and feature engineering.
- **Feature Engineering:** Feature engineering is the process of creating, modifying, or selecting useful features to improve model performance.
- **Feature Transformation:** Feature transformation converts data into a suitable format for effective machine learning.
- **Feature Subset Selection:** Feature subset selection identifies and retains only the most relevant features for building an efficient model.

Introduction to Model Evaluation and Feature Engineering:

Model evaluation and feature engineering are essential steps in the machine learning process. After training a model, its performance must be evaluated to ensure it makes accurate predictions on new data. Feature engineering improves the quality of input data by creating, transforming, and selecting relevant features, leading to better model accuracy, reduced complexity, and improved overall performance. Understanding these concepts helps in developing efficient, reliable, and interpretable machine learning models.

Model selection:

Model selection in machine learning is the process of choosing the most appropriate machine learning model (ML model) for the selected task. The selected model is usually the one that generalizes best to unseen data while most successfully meeting relevant model performance metrics.

Steps in Model Selection:

Step 1: Define the Problem

The first step is to clearly identify the type of machine learning problem. The problem may be classification, regression, or clustering.

Step 2: Prepare the Dataset

The collected data should be cleaned and preprocessed. Missing values, duplicate records, and noisy data should be removed. The dataset is then divided into training and testing datasets.

Step 3: Select Candidate Models

Several suitable machine learning algorithms are selected based on the problem type. These candidate models will later be compared to determine the best-performing model.

Step 4: Train the Models

Each selected algorithm is trained using the training dataset. During training, the model learns patterns and relationships from the available data.

Step 5: Evaluate the Models

After training, each model is evaluated using the testing dataset. Appropriate evaluation metrics are used to measure the model's performance.

Step 6: Compare Model Performance

The performance of all candidate models is compared using evaluation metrics such as accuracy, precision, recall, F1-score, MAE, RMSE, and R^2 score. The model with the best overall performance is selected.

Step 7: Select the Best Model

Finally, the model that provides the highest accuracy, good generalization ability, low computational cost, and better interpretability is selected for deployment.

Advantages of Model Selection

- It helps in selecting the most suitable algorithm for a given problem.
- It improves the prediction accuracy of the machine learning model.
- It reduces computational cost and training time.
- It minimizes overfitting and underfitting.
- It improves the reliability and efficiency of the machine learning system.
- It helps in developing models that perform well on unseen data.

Limitations of Model Selection

- Selecting the wrong model may reduce prediction accuracy.
- Comparing multiple models can be time-consuming.
- Complex models require high computational resources.
- Model performance depends heavily on the quality of the dataset.

Applications of Model Selection

Model selection is widely used in many real-world applications, including:

- Disease prediction in healthcare.
- Fraud detection in banking and finance.
- Email spam detection.
- Customer segmentation in marketing.

- House price prediction.
- Weather forecasting.
- Image recognition and speech recognition.

Real-World Example:

Suppose a bank wants to predict whether a customer will repay a loan.

- If the task is to predict **"Yes" or "No"**, a **Decision Tree** or **Logistic Regression** model can be selected because the problem is a **classification** task.
- If the task is to predict the **loan amount**, **Linear Regression** can be selected because the problem is a **regression** task.
- If the bank wants to group customers with similar spending habits, **K-Means Clustering** can be selected because the problem is a **clustering** task.

Training a Model:

Training a model is the process of teaching a machine learning algorithm to learn patterns, relationships, and trends from historical data by adjusting its parameters, so that it can accurately predict the output for new or unseen data.

Need for Training a Model:

Training a model is necessary because machine learning algorithms do not possess prior knowledge about the data. They must learn from examples provided in the training dataset.

A properly trained model can recognize patterns, reduce prediction errors, improve decision-making, and provide accurate predictions for future data. Without training, a machine learning model cannot perform meaningful predictions.

Process of Training a Model

The process of training a machine learning model consists of several steps.

Step 1: Collect the Dataset

The first step is to collect a relevant dataset that contains sufficient examples for the machine learning problem. The quality and quantity of data significantly influence the model's performance.

Example: Customer records for loan prediction or patient records for disease prediction.

Step 2: Data Preprocessing

The collected data is usually incomplete or inconsistent. Therefore, preprocessing is performed to improve data quality.

Data preprocessing includes:

- Handling missing values
- Removing duplicate records
- Removing noisy data
- Detecting and treating outliers
- Encoding categorical variables
- Feature scaling and normalization

A clean dataset improves the learning ability of the model.

Step 3: Split the Dataset

The dataset is divided into different subsets.

Training Set: The training dataset is used to teach the model by allowing it to learn patterns from the data.

Validation Set: The validation dataset is used to tune model parameters and prevent overfitting.

Testing Set: The testing dataset is used only after training to evaluate the model's performance on unseen data.

A commonly used split is:

- Training Data – 80%
- Testing Data – 20%

or

- Training Data – 70%

- Validation Data – 15%
- Testing Data – 15%

Step 4: Select a Machine Learning Algorithm

The appropriate algorithm is selected according to the problem.

Examples include:

- Linear Regression
- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Naïve Bayes

Step 5: Train the Model

The selected algorithm is trained using the training dataset.

During training:

- The model receives input variables (X).
- It predicts the output (Y').
- The predicted output is compared with the actual output (Y).
- The prediction error is calculated.
- The model adjusts its parameters to reduce the error.
- This process is repeated many times until the error becomes minimum.

This iterative learning process is called **training**.

Step 6: Validate the Model

After training, the model is validated using the validation dataset.

Validation helps to:

- Tune hyperparameters.
- Detect overfitting.
- Improve generalization.

If the validation accuracy is poor, the model is retrained with different parameter settings.

Step 7: Evaluate the Model

The trained model is evaluated using the testing dataset.

Classification Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix
- ROC Curve
- AUC

Regression Evaluation Metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

These metrics indicate how well the trained model performs on unseen data.

Step 8: Improve the Model

If the model performance is not satisfactory, several improvement techniques can be applied.

These include:

- Hyperparameter tuning
- Cross-validation
- Feature engineering

- Feature selection
- Regularization
- Ensemble learning
- Increasing training data
- Removing noisy data

These techniques improve prediction accuracy and reduce overfitting.

Advantages of Training a Model:

Training a machine learning model provides several benefits that improve its performance and reliability. The major advantages are:

1. Learns from Historical Data: Training enables the model to learn patterns, trends, and relationships from historical data, allowing it to make informed predictions.

2. Improves Prediction Accuracy: A well-trained model produces more accurate predictions by minimizing the difference between actual and predicted values.

3. Discovers Hidden Patterns: Training helps identify hidden relationships and complex patterns in the dataset that may not be easily recognized by humans.

4. Reduces Prediction Errors: During training, the model continuously adjusts its parameters to reduce errors, resulting in improved performance.

Limitations of Training a Model:

1. Requires Large Amounts of High-Quality Data: A model requires sufficient, accurate, and representative data to achieve good performance.

2. Time-Consuming Process: Training complex machine learning and deep learning models may require several hours or even days.

3. High Computational Cost: Advanced models often require powerful CPUs, GPUs, large memory, and significant storage capacity.

4. Sensitive to Data Quality: Poor-quality data, missing values, noisy data, and outliers can reduce model accuracy and reliability.

5. Risk of Overfitting: The model may memorize the training data instead of learning general patterns, resulting in poor performance on unseen data.

6. Risk of Underfitting: If the model is too simple or insufficiently trained, it may fail to capture important pattern

Applications of Model Training:

1. Healthcare: Used for disease diagnosis, medical image analysis, patient risk prediction, drug discovery, and personalized treatment recommendations.

2. Banking and Finance: Used for fraud detection, credit scoring, loan approval, stock market prediction, financial forecasting, and risk management.

3. Real Estate: Used to predict house prices, estimate property values, and analyze real estate market trends.

4. Weather Forecasting: Used to predict weather conditions, rainfall, temperature, storms, and climate changes.

5. Agriculture: Used for crop yield prediction, disease detection, soil quality analysis, irrigation management, and precision farming.

6. Image Recognition: Used in facial recognition, object detection, medical imaging, autonomous vehicles, and surveillance systems.

7. Speech Recognition: Used in virtual assistants, voice-controlled devices, automatic transcription, and language translation systems.

8. Natural Language Processing (NLP): Used in chatbots, sentiment analysis, machine translation, text summarization, question answering, and document classification.

Model Representation and Interpretability:

Introduction

In machine learning, a model is trained to learn the relationship between input features and the target output. The way in which this relationship is expressed is called model representation. The ability to understand and explain how the model makes predictions is called model interpretability. These concepts are important because a good model should not only produce accurate predictions but should also generalize well to new data and be understandable to users.

Model Representation

- Model representation refers to the mathematical, logical, or structural form used by a machine learning algorithm to represent the relationship between input variables and output variables. It defines how the model stores the knowledge learned from the training data.
- For example, in linear regression, the relationship between input and output is represented using a linear equation.

$$Y = wX + b$$

- In this equation, Y represents the predicted output, X represents the input feature, w represents the weight, and b represents the bias or intercept.

Example of model representation

Suppose we want to predict the salary of an employee based on years of experience. A linear regression model may represent this relationship as follows:

$$Salary = 5000 \times Experience + 20000$$

This means that for every additional year of experience, the salary increases by 5000 units, and the base salary is 20000 units. This equation is the representation of the model

Types of model representation:

- **Linear models:** These models represent the relationship using linear equations. For example, linear regression predicts house price based on area.
- **Tree-based models:** These models represent decisions using a tree structure. For example, a decision

tree may classify whether a student will pass or fail based on attendance and marks.

- **Probabilistic models:** These models represent relationships using probabilities. For example, Naïve Bayes predicts whether an email is spam.
- **Neural networks:** These models represent knowledge using interconnected neurons and weighted connections. For example, a neural network can recognize handwritten digits.

Underfitting:

- Underfitting occurs when a machine learning model is too simple to learn the underlying patterns in the training data. As a result, the model performs poorly on both training data and testing data.
- If the target function is kept too simple, it may not be able to capture the essential nuances and represent the underlying data well.
- A typical case of underfitting may occur when trying to represent a non-linear data with a linear model as demonstrated by both cases of underfitting shown in figure 3.5. Many times underfitting happens due to unavailability of sufficient training data.
- Underfitting results in both poor performance with training data as well as poor generalization to test data. Underfitting can be avoided by
  1. using more training data
  2. reducing features by effective feature selection

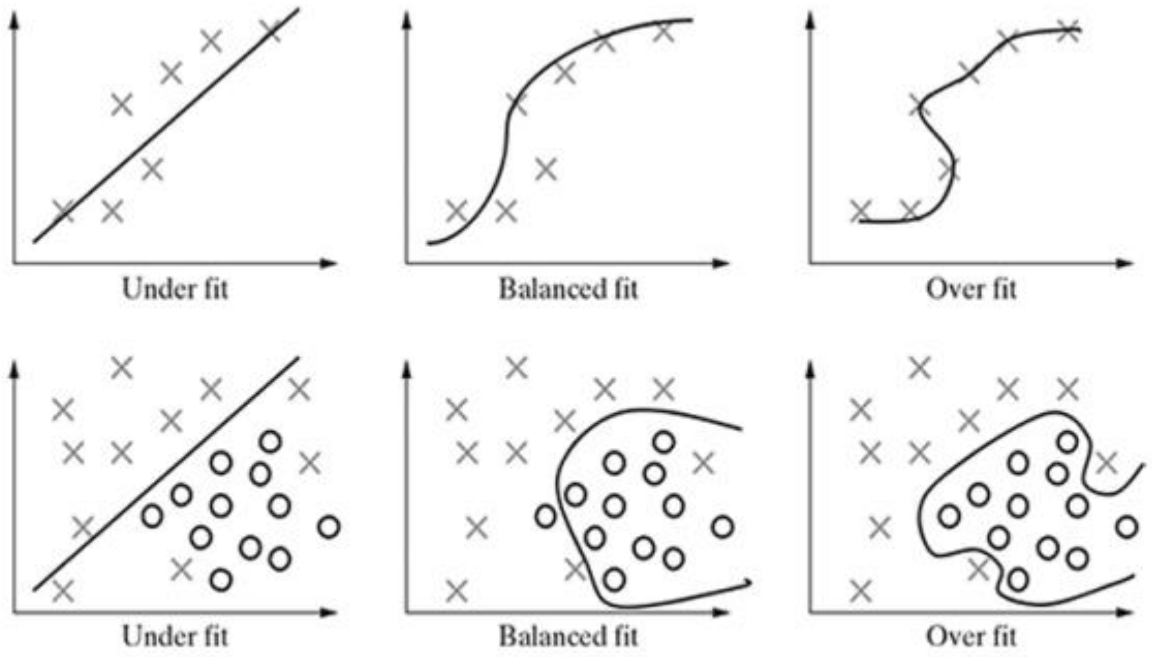


FIG. 3.5 Underfitting and Overfitting of models

Example of underfitting

Suppose we want to predict student marks based on study hours. If we use a very simple straight-line model for data that actually follows a curved pattern, the model will not capture the relationship correctly. Therefore, the predictions will be inaccurate.

Overfitting

Overfitting occurs when a machine learning model learns the training data too closely, including noise and outliers. As a result, the model performs very well on training data but poorly on unseen testing data.

- **Overfitting** is a situation where the machine learning model is designed in such a way that it learns the training data too closely.
- In an overfitted model, specific deviations in the training data, such as **noise** and **outliers**, become part of the learned model.
- As a result, the model memorizes the training data instead of learning the actual underlying patterns.
- Overfitting negatively affects the model's performance on unseen or test data.
- It usually occurs when an **excessively complex model** is used to closely fit the training dataset.
- The target function attempts to classify or fit **every training data point correctly** by creating a highly

complex decision boundary.

- However, the same complex decision boundary does not represent the patterns in new or unknown test data.
- Consequently, the model makes incorrect predictions or classifications on the test dataset.
- An overfitted model achieves **high accuracy on the training dataset** but **low accuracy on the testing dataset**.
- Therefore, overfitting results in **poor generalization**, meaning the model cannot perform well on new, unseen data.

Example of overfitting

Suppose we have data about house prices. If the model memorizes every training example, including unusual or noisy data points, it may predict training data accurately. However, when a new house is given as input, the prediction may be incorrect because the model has not learned the general pattern.

Difference between underfitting and overfitting

Aspect	Underfitting	Overfitting
Model complexity	Too simple	Too complex
Training error	High	Low
Testing error	High	High
Generalization	Poor	Poor
Bias	High	Low
Variance	Low	High

Bias–Variance Trade-off

Machine learning model should balance bias and variance. If the model is too simple, it will underfit the data. If the model is too complex, it will overfit the data. The ideal model captures the important patterns while ignoring noise.

Errors due to Bias:

- Errors due to bias arise from simplifying assumptions made by the model to make the target function less complex or easier to learn. In short, it is due to underfitting of the model.

- Parametric models generally have high bias making them easier to understand/interpret and faster to learn. These algorithms have a poor performance on data sets, which are complex in nature and do not align with the simplifying assumptions made by the algorithm. Underfitting results in high bias.

Errors due to Variance:

- Errors due to variance occur from difference in training data sets used to train the model. Different training data sets (randomly sampled from the input data set) are used to train the model.
- Ideally the difference in the data sets should not be significant and the model trained using different training data sets should not be too different. However, in case of overfitting, since the model closely matches the training data, even a small difference in training data gets magnified in the model.

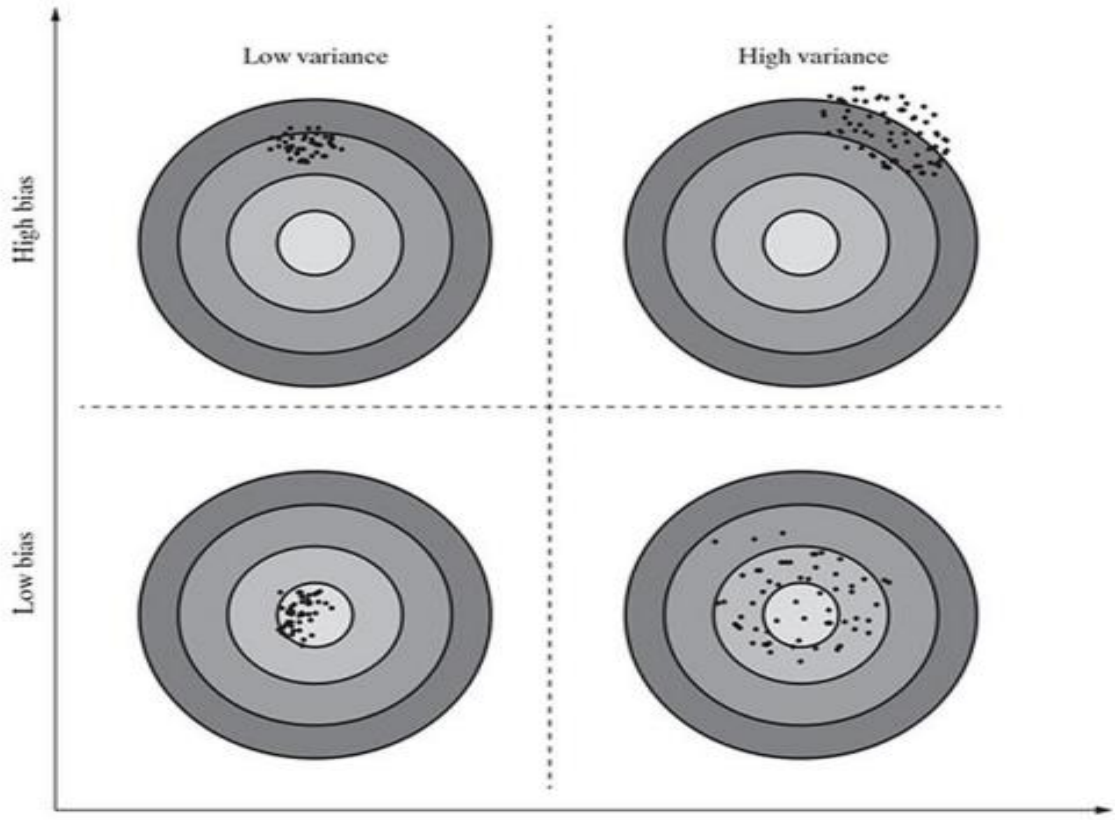


FIG. 3.6 Bias-variance trade-off

So, the problems in training a model can either happen because either (a) the model is too simple and hence fails to interpret the data grossly or (b) the model is extremely complex and magnifies even small differences in the training data. As is quite understandable:

- Increasing the bias will decrease the variance, and

- Increasing the variance will decrease the bias On one hand, parametric algorithms are generally seen to demonstrate high bias but low variance.

On the other hand, non-parametric algorithms demonstrate low bias and high variance. As can be observed in Figure 3.6 , the best solution is to have a model with low bias as well as low variance. However, that may not be possible in reality. Hence, the goal of supervised machine learning is to achieve a balance between bias and variance. The learning algorithm chosen and the user parameters which can be configured helps in striking a trade off between bias and variance.

For example, in a popular supervised algorithm k-Nearest Neighbors or KNN, the user configurable parameter ‘k’ can be used to do a trade-off between bias and variance. In one hand, when the value of ‘k’ is decreased, the model becomes simpler to fit and bias increases. On the other hand, when the value of ‘k’ is increased, the variance increases.

Model Interpretability:

Model Interpretability is the ability to understand, explain, and justify how a machine learning model makes predictions or decisions. An interpretable model enables humans to understand the relationship between the input features and the predicted output. It helps users know why a model produced a particular result instead of treating the model as a "black box."

Need for Model Interpretability

Model interpretability is essential because it increases trust and confidence in machine learning models. It allows developers, domain experts, and end users to understand the reasoning behind the model's predictions. Interpretability also helps identify errors, detect bias, improve fairness, and ensure that the model behaves as expected.

In critical fields such as healthcare, finance, law, and autonomous systems, interpretability is necessary because decisions made by machine learning models can have significant consequences.

Types of Model Interpretability

1. Global Interpretability

Global interpretability explains the overall behavior of a machine learning model. It helps us understand how the model makes predictions across the entire dataset and identifies which features have the greatest influence on the model.

Example:

In a house price prediction model, global interpretability may show that location, size of the house, and number of bedrooms are the most important factors affecting the predicted price.

2. Local Interpretability

Local interpretability explains the prediction made for a single data instance. It provides the specific reasons why the model produced a particular output for one input.

Example:

If a bank rejects a customer's loan application, the model may explain that the rejection was due to a low credit score, high debt, and insufficient monthly income.

Interpretable Models

Some machine learning models are naturally easy to understand because their decision-making process is simple and transparent.

Examples include:

- Linear Regression
- Logistic Regression
- Decision Tree
- Naïve Bayes

Example:

In a decision tree, the prediction can be explained by tracing the path from the root node to the leaf node based on the input conditions.

Less Interpretable (Black-Box) Models

Some models are highly complex and difficult to explain because they involve many parameters and hidden computations. These models are often called black-box models.

Examples include:

- Random Forest
- Support Vector Machine (Complex Kernel)
- Artificial Neural Networks
- Deep Learning Models

Example:

A deep learning model may accurately identify cancer from medical images, but it is difficult to explain exactly how the millions of learned parameters contributed to the prediction.

Techniques to Improve Model Interpretability

Several techniques are used to explain the predictions of complex machine learning models.

- **Feature Importance:** Identifies the features that have the greatest impact on the prediction.
- **LIME (Local Interpretable Model-Agnostic Explanations):** Explains the prediction for an individual data instance.
- **SHAP (SHapley Additive Explanations):** Calculates the contribution of each feature to the prediction.
- **Partial Dependence Plot (PDP):** Shows how changes in a feature affect the predicted output.

Evaluating Performance of a Model

Model evaluation is the process of assessing the performance of a machine learning model by comparing its predicted outputs with the actual outputs using various evaluation metrics. The main objective is to determine whether the model makes accurate and reliable predictions.

Steps in Model Evaluation

Evaluating the performance of a machine learning model involves a series of systematic steps to determine how accurately the model predicts unseen data. The following are the major steps involved in model evaluation.

Step 1: Split the Dataset

The first step in model evaluation is to divide the available dataset into two separate subsets: the **training dataset** and the **testing dataset**. The training dataset is used to train the machine learning model, while the testing dataset is used to evaluate its performance on unseen data. A commonly used data split is **80% for training and 20% for testing**, although other ratios such as 70:30 or 75:25 may also be used depending on the size of the dataset.

Step 2: Train the Model

In this step, the selected machine learning algorithm is trained using the training dataset. During the training process, the model learns the underlying patterns and relationships present in the data by adjusting its internal parameters. The objective is to build a model that can accurately predict outputs for new data.

Step 3: Make Predictions

Once the training process is complete, the trained model is applied to the testing dataset to make predictions. Since the testing data has not been used during training, it provides an unbiased evaluation of the model's predictive capability.

Step 4: Compare Predicted Values with Actual Values

The predicted outputs generated by the model are compared with the actual target values in the testing dataset. This comparison helps determine how accurately the model has performed. Any differences between the predicted and actual values indicate prediction errors, which are used to assess the model's effectiveness.

Performance Metrics for Classification Models

1. Accuracy

Accuracy is a fundamental metric used for evaluating the performance of a classification model. It tells us the proportion of correct predictions made by the model out of all predictions.

$$\textbf{Accuracy} = \frac{\textbf{NumberofCorrectPredictions}}{\textbf{TotalNumberofPredictions}}$$

While accuracy provides a quick snapshot, it can be misleading in cases of imbalanced datasets. For example, in a dataset with 90% class A and 10% class B, a model predicting only class A will still achieve 90% accuracy but it will fail to identify any class B instances.

Accuracy is good but it gives a False Positive sense of achieving high accuracy. The problem arises due to the possibility of misclassification of minor class samples being very high.

2. Precision

It measures how many of the positive predictions made by the model are actually correct. It's useful when the cost of false positives is high such as in medical diagnoses where predicting a disease when it’s not present can have serious consequences.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Where:

- TP = True Positives
- FP = False Positives

Precision helps ensure that when the model predicts a positive outcome, it’s likely to be correct.

3. Recall

Recall or Sensitivity measures how many of the actual positive cases were correctly identified by the model. It is important when missing a positive case (false negative) is more costly than false positives.

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

Where:

- FN = False Negatives

In scenarios where catching all positive cases is important (like disease detection), recall is a key metric.

4. F1 Score

The F1 Score is the harmonic mean of precision and recall. It is useful when we need a balance between precision and recall as it combines both into a single number. A high F1 score means the model performs well on both metrics. Its range is [0,1].

Lower recall and higher precision gives us great accuracy but then it misses a large number of instances. More the F1 score better will be performance. It can be expressed mathematically in this way:

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

5. Logarithmic Loss (Log Loss)

Log loss measures the uncertainty of the model's predictions. It is calculated by penalizing the model for assigning low probabilities to the correct classes. This metric is used in multi-class classification and is helpful when we want to assess a model's confidence in its predictions. If there are N samples belonging to the M class, then we calculate the Log loss in this way:

$$\textbf{Logarithmic Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot \log(p_{ij})$$

Where:

- y_{ij}=Actual class (0 or 1) for sample i and class j
- p_{ij} =Predicted probability for sample i and class j

The goal is to minimize Log Loss, as a lower Log Loss shows higher prediction accuracy.

6. Area Under Curve (AUC) and ROC Curve

It is useful for binary classification tasks. The AUC value represents the probability that the model will rank a randomly chosen positive example higher than a randomly chosen negative example. AUC ranges from 0 to 1 with higher values showing better model performance.

1. True Positive Rate(TPR)

Also known as sensitivity or recall, the True Positive Rate measures how many actual positive instances were correctly identified by the model. It answers the question: "Out of all the actual positive cases, how many did the model correctly identify?"

Formula:

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}}$$

Where:

- TP = True Positives (correctly predicted positive cases)
- FN = False Negatives (actual positive cases incorrectly predicted as negative)

2. True Negative Rate(TNR)

Also called specificity, the True Negative Rate measures how many actual negative instances were correctly identified by the model. It answers the question: "Out of all the actual negative cases, how many did the model correctly identify as negative?"

Formula:

$$\mathbf{TNR} = \frac{\mathbf{TN}}{\mathbf{TN} + \mathbf{FP}}$$

Where:

- TN = True Negatives (correctly predicted negative cases)
- FP = False Positives (actual negative cases incorrectly predicted as positive)

3. False Positive Rate(FPR)

It measures how many actual negative instances were incorrectly classified as positive. It's a key metric when the cost of false positives is high such as in fraud detection.

Formula:

$$\mathrm{F}\mathbf{PR} = \frac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}}$$

Where:

- FP = False Positives (incorrectly predicted positive cases)
- TN = True Negatives (correctly predicted negative cases)

4. False Negative Rate(FNR)

It measures how many actual positive instances were incorrectly classified as negative. It answers: "Out of all the actual positive cases, how many were misclassified as negative?"

Formula:

$$\mathbf{FNR} = \frac{\mathbf{FN}}{\mathbf{FN} + \mathbf{TP}}$$

Where:

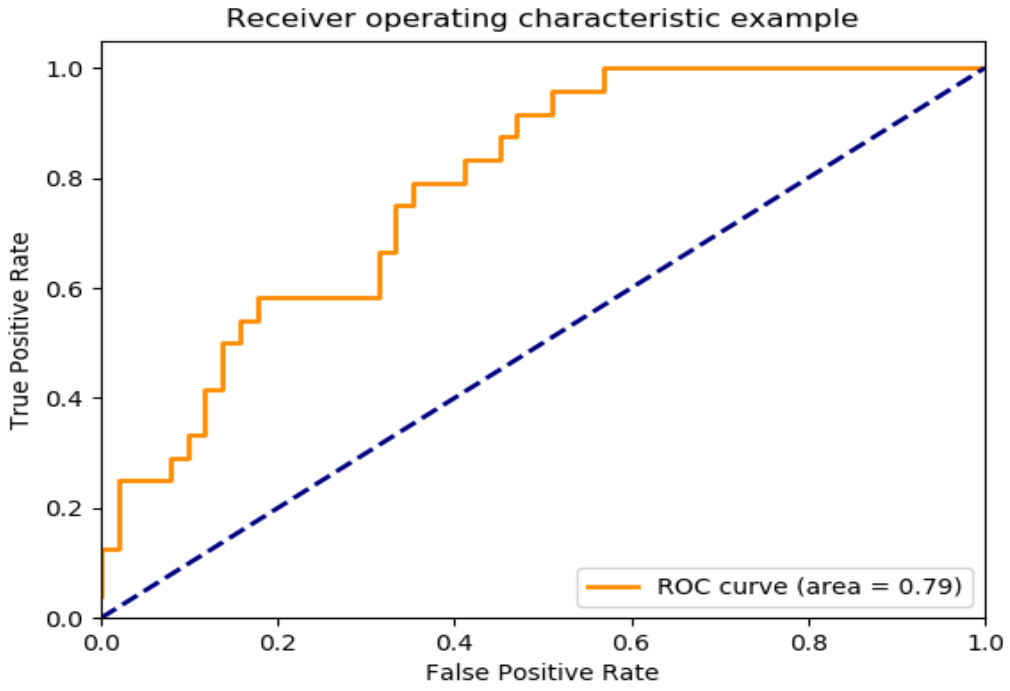
- FN = False Negatives (incorrectly predicted negative cases)
- TP = True Positives (correctly predicted positive cases)

ROC Curve

It is a graphical representation of the True Positive Rate (TPR) vs the False Positive Rate (FPR) at different

classification thresholds. The curve helps us visualize the trade-offs between sensitivity (TPR) and specificity (1 - FPR) across various thresholds. Area Under Curve (AUC) quantifies the overall ability of the model to distinguish between positive and negative classes.

- AUC = 1: Perfect model (always correctly classifies positives and negatives).
- AUC = 0.5: Model performs no better than random guessing.
- AUC < 0.5: Model performs worse than random guessing (showing that the model is inverted).



ROC Curve for Evaluation of Classification Models

7. Confusion Matrix

Confusion matrix creates a N X N matrix, where N is the number of classes or categories that are to be predicted. Here we have N = 2, so we get a 2 X 2 matrix. Suppose there is a problem with our practice which is a binary classification.

Samples of that classification belong to either Yes or No. So, we build our classifier which will predict the class for the new input sample. After that, we tested our model with 165 samples and we get the following result.

n=165	Predicted No	Predicted Yes
Actual No	50	10
Actual Yes	5	100

There are 4 terms we should keep in mind:

1. **True Positives:** It is the case where we predicted Yes and the real output was also Yes.
2. **True Negatives:** It is the case where we predicted No and the real output was also No.
3. **False Positives:** It is the case where we predicted Yes but it was actually No.
4. **False Negatives:** It is the case where we predicted No but it was actually Yes.

Regression Metrics

In the regression task, we are supposed to predict the target variable which is in the form of continuous values. To evaluate the performance of such a model below metrics are used:

1. Mean Absolute Error (MAE)

MAE calculates the average of the absolute differences between the predicted and actual values. It gives a clear view of the model's prediction accuracy but it doesn't shows whether the errors are due to over- or under-prediction. It is simple to calculate and interpret helps in making it a good starting point for model evaluation.

$$\mathbf{MAE} = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{j=1}}^{\mathbf{N}} \mid \mathbf{y_j} - \mathbf{\hat{y}_j} \mid$$

Where:

- y_j = Actual value
- $\hat{y}_j$ = Predicted value

2. Mean Squared Error (MSE)

MSE calculates the average of the squared differences between the predicted and actual values. Squaring the differences ensures that larger errors are penalized more heavily helps in making it sensitive to outliers. This is useful when large errors are undesirable but it can be problematic when outliers are not relevant to the model's purpose.

Formula:

$$\mathbf{MSE} = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{j=1}}^{\mathbf{N}} \left(\mathbf{y_j} - \mathbf{\hat{y}_j}\right)^{\mathbf{2}}$$

Where:

- y_j = Actual value

- $\hat{y}_j$ = Predicted value

3. Root Mean Squared Error (RMSE)

RMSE is the square root of MSE, bringing the metric back to the original scale of the data. Like MSE, it heavily penalizes larger errors but is easier to interpret as it's in the same units as the target variable. It's useful when we want to know how much our predictions deviate from the actual values in terms of the same scale.

Formula:

$$\mathbf{RMSE} = \sqrt{\frac{\sum_{\mathbf{j=1}}^{\mathbf{N}} \left(\mathbf{y_j} - \hat{\mathbf{y}}_\mathbf{j}\right)^2}{\mathbf{N}}}$$

Where:

- y_j = Actual value
- $\hat{y}_j$ = Predicted value

4. Root Mean Squared Logarithmic Error (RMSLE)

RMSLE is useful when the target variable spans a wide range of values. Unlike RMSE, it penalizes underestimations more than overestimations helps in making it ideal for situations where the model is predicting quantities that vary greatly in scale like predicting prices or population.

Formula:

$$\mathbf{RMSLE} = \sqrt{\frac{\sum_{\mathbf{j=1}}^{\mathbf{N}} \left(\mathbf{log}(\mathbf{y_j} + \mathbf{1}) - \mathbf{log}(\hat{\mathbf{y}}_\mathbf{j} + \mathbf{1})\right)^2}{\mathbf{N}}}$$

Where:

- y_j = Actual value
- $\hat{y}_j$ = Predicted value

5. R^2 (R-squared)

R2 score represents the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 value close to 1 shows a model that explains most of the variance while a value close to 0 shows that the model does not explain much of the variability in the data. R^2 is used to assess the goodness-of-fit of regression models.

Formula:

$$R^2 = \mathbf{1} - \frac{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} (\boldsymbol{y_j} - \hat{\boldsymbol{y}}_\boldsymbol{j})^2}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} (\boldsymbol{y_j} - \bar{\boldsymbol{y}})^2}$$

Where:

- y_j = Actual value

- $\hat{y}_j$ = Predicted value
- $\bar{y}$= Mean of the actual values

Improving the performance of a model

Improving the performance of a model is the process of enhancing the accuracy, reliability, and generalization ability of a machine learning model by applying various optimization techniques during data preparation, model training, and evaluation.

Techniques for Improving the Performance of a Model

1. Improve Data Quality

The quality of the training data has a significant impact on model performance. The dataset should be cleaned by removing duplicate records, correcting inconsistencies, handling missing values, and eliminating noisy data. High-quality data enables the model to learn meaningful patterns and produce more accurate predictions.

Example: Replacing missing values with the mean or median improves the quality of the dataset and enhances model performance.

2. Collect More Training Data

A larger and more diverse training dataset helps the model learn different patterns and reduces the chances of overfitting. More training examples improve the model's ability to generalize to unseen data.

Example: Increasing the number of customer records in a sales prediction model leads to better prediction accuracy.

3. Perform Feature Selection

Feature selection involves choosing only the most relevant input features while removing unnecessary or redundant features. This reduces model complexity, decreases training time, and improves prediction accuracy.

Example: In a student performance prediction model, attendance, assignment marks, and internal marks may be selected, while irrelevant features such as student ID are removed.

4. Apply Feature Engineering

Feature engineering is the process of creating new and meaningful features from existing data. Well-designed features help the model identify hidden relationships and improve prediction performance.

Example: Creating a new feature called **Body Mass Index (BMI)** from height and weight provides more useful information than using height and weight separately.

5. Normalize or Standardize the Data

Feature scaling ensures that all input features are on a similar scale. This helps many machine learning algorithms converge faster and improves model accuracy.

- **Normalization** scales feature values between 0 and 1.
- **Standardization** transforms data to have a mean of 0 and a standard deviation of 1.

Example: Standardizing age and salary values before training a logistic regression model improves learning efficiency.

6. Select an Appropriate Machine Learning Algorithm

Different machine learning algorithms perform differently depending on the nature of the dataset. Selecting the most suitable algorithm improves prediction performance.

Example: Decision Trees work well for categorical data, while Linear Regression is suitable for continuous numerical prediction.

7. Tune Hyperparameters

Hyperparameters are settings that control the learning process of the model. Optimizing hyperparameters improves model performance and reduces prediction errors.

Examples of hyperparameters include:

- Learning rate
- Number of neighbors (K in KNN)
- Maximum tree depth
- Number of hidden layers

Example: Choosing the optimal value of **K** in the K-Nearest Neighbors algorithm improves classification accuracy.

8. Prevent Underfitting and Overfitting

The model should maintain a balance between simplicity and complexity.

- **Underfitting** occurs when the model is too simple to capture the underlying patterns.
- **Overfitting** occurs when the model memorizes the training data instead of learning general patterns.

Techniques such as regularization, pruning, early stopping, and cross-validation help overcome these problems.

9. Use Cross-Validation

Cross-validation evaluates the model on different subsets of the dataset. It provides a more reliable estimate of model performance and reduces the risk of selecting a poorly generalized model.

Example: In **5-fold cross-validation**, the dataset is divided into five equal parts, and the model is trained and tested five times using different subsets.

10. Use Ensemble Learning

Ensemble learning combines the predictions of multiple machine learning models to improve overall performance. Ensemble methods often achieve higher accuracy than individual models.

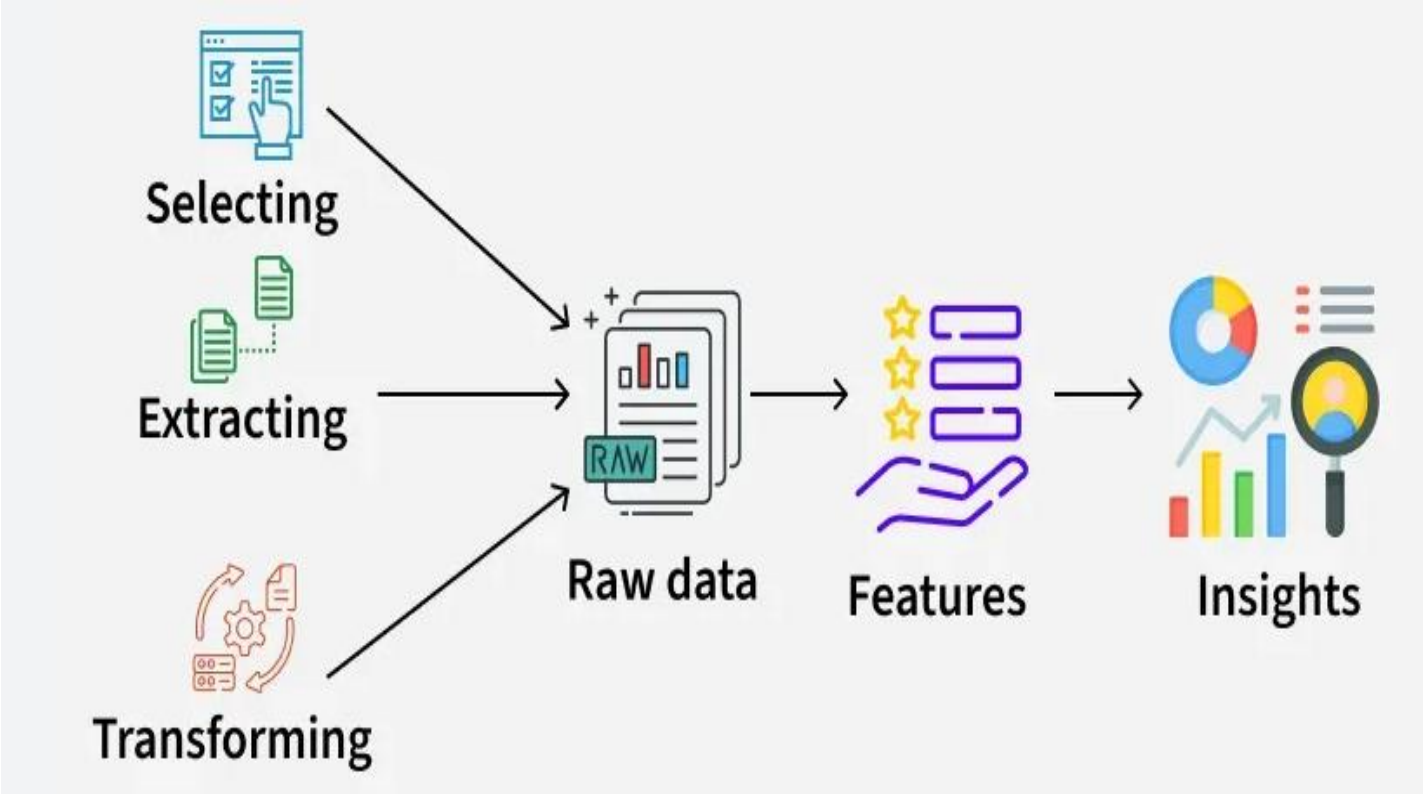
Examples include:

- Random Forest
- Bagging
- Boosting
- AdaBoost
- Gradient Boosting

Example: A Random Forest combines multiple decision trees to produce more accurate and stable predictions.

Basics of Feature Engineering

Feature Engineering is the process of selecting, creating, transforming, and modifying input features from raw data to improve the performance of a machine learning model. It aims to provide the model with the most relevant and informative features for making accurate predictions.



This step may include handling missing values, encoding categories, scaling numbers, creating new features or combining existing ones. It helps turn messy real-world data into a form that models can understand and use for better predictions.

Processes Involved in Feature Engineering:

- **Feature Creation**
- **Feature Transformation**
- **Feature Extraction**
- **Feature Selection**
- **Feature Scaling**

1.Feature Creation: Feature creation involves generating new features from domain knowledge or by observing patterns in the data. It can be:

- **Domain-specific**: Created based on industry knowledge like business rules.
- **Data-driven**: Derived by recognizing patterns in data.
- **Synthetic**: Formed by combining existing features.

2. Feature Transformation: Transformation adjusts features to improve model learning:

- **Normalization & Scaling**: Adjust the range of features for consistency.
- **Encoding**: Converts categorical data to numerical form i.e one-hot encoding.
- **Mathematical transformations**: Like logarithmic transformations for skewed data.

3. Feature Extraction: Transform existing features into a lower-dimensional or more informative representation (e.g., PCA).

- **Dimensionality reduction**: Techniques like PCA reduce features while preserving important information.
- **Aggregation & Combination**: Summing or averaging features to simplify the model.

4. Feature Selection: Feature selection involves choosing a subset of relevant features to use:

- **Filter methods**: Based on statistical measures like correlation.
- **Wrapper methods**: Select based on model performance.
- **Embedded methods**: Feature selection integrated within model training.

5. Feature Scaling: Scaling ensures that all features contribute equally to the model:

- **Min-Max scaling**: Rescales values to a fixed range like 0 to 1.
- **Standard scaling**: Standardizes features to have mean 0 and variance 1

Steps in Feature Engineering

Feature engineering can vary depending on the specific problem but the general steps are:

1. **Data Cleaning:** Identify and correct errors or inconsistencies in the dataset to ensure data quality and reliability.
2. **Data Transformation:** Transform raw data into a format suitable for modeling including scaling, normalization and encoding.
3. **Feature Extraction:** Create new features by combining or deriving information from existing ones to provide more meaningful input to the model.
4. **Feature Selection:** Choose the most relevant features for the model using techniques like correlation analysis, mutual information and stepwise regression.
5. **Feature Iteration:** Continuously refine features based on model performance by adding, removing or modifying features for improvement.

Common Techniques in Feature Engineering

1. One-Hot Encoding: One-Hot Encoding converts categorical variables into binary indicators, allowing them

to be used by machine learning models.

```
import pandas as pd

data = {'Color': ['Red', 'Blue', 'Green', 'Blue']}
df = pd.DataFrame(data)

df_encoded = pd.get_dummies(df, columns=['Color'], prefix='Color')

print(df_encoded)
```

Output:

```
   Color_Blue  Color_Green  Color_Red
0       False        False       True
1        True        False      False
2       False         True      False
3        True        False      False
```

2. Binning: Binning transforms continuous variables into discrete bins, making them categorical for easier analysis.

```
import pandas as pd

data = {'Age': [23, 45, 18, 34, 67, 50, 21]}
df = pd.DataFrame(data)

bins = [0, 20, 40, 60, 100]
labels = ['0-20', '21-40', '41-60', '61+']

df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

print(df)
```

Output:

```
   Age Age_Group
0   23     21-40
1   45     41-60
2   18      0-20
3   34     21-40
4   67       61+
5   50     41-60
6   21     21-40
```

3. Text Data Preprocessing: Involves removing stop-words, stemming and vectorizing text data to prepare it for machine learning models.

5. **Feature Splitting**: Divides a single feature into multiple sub-features, uncovering valuable insights and improving model performance.

Importance of Feature Engineering:

1. Improves Model Accuracy

Feature engineering improves the quality of the input data by selecting relevant features and removing unnecessary ones. Better-quality features help the model learn important patterns, resulting in more accurate predictions.

2. Reduces Model Complexity

Removing irrelevant and redundant features simplifies the machine learning model. A simpler model is easier to train, requires less memory, and is less likely to overfit the training data.

3. Enhances Model Generalization

Feature engineering helps the model focus on meaningful patterns instead of memorizing the training data. This improves the model's ability to make accurate predictions on new and unseen data.

4. Reduces Training Time

Using only important features decreases the amount of data processed during training. This reduces computational cost and speeds up the learning process.

5. Handles Missing and Noisy Data

Feature engineering includes data cleaning techniques such as handling missing values, removing duplicate records, and eliminating noisy data. This improves the quality of the dataset and increases model reliability.

6. Improves Data Representation

Feature engineering transforms raw data into a format that machine learning algorithms can understand more effectively. Well-represented data enables the model to identify meaningful relationships between variables.

7. Creates New Informative Features

New features can be created by combining existing features to provide additional information that improves prediction accuracy.

8. Improves Performance of Machine Learning Algorithms

Many machine learning algorithms perform better when the input features are properly scaled and transformed. Feature engineering ensures that the data is suitable for efficient model learning.

9. Reduces Overfitting

Removing irrelevant features prevents the model from learning unnecessary details and noise in the training data. This reduces overfitting and improves the model's ability to generalize.

10. Supports Better Decision-Making

Feature engineering provides meaningful information that enables machine learning models to make reliable predictions. This helps organizations make informed decisions in various applications.

Feature Transformation

Feature Transformation is the process of converting or modifying the original features into a new format or scale without changing the underlying information. The transformed features help machine learning algorithms learn patterns more efficiently and improve prediction performance.

Types of Feature Transformation

1. Normalization

Normalization transforms numerical values into a common scale, usually between **0 and 1**. It is useful when the features have different ranges.

Formula

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

Example

Suppose the ages of students range from **18 to 30 years**. Normalization converts these values into a range between **0 and 1**, making them suitable for machine learning algorithms.

2. Standardization

Standardization transforms the data so that it has a **mean of 0** and a **standard deviation of 1**. It is widely used in algorithms such as Logistic Regression, Support Vector Machines, and Principal Component Analysis (PCA).

Formula

$$Z = \frac{X - \mu}{\sigma}$$

where:

- X= Original value
- μ= Mean of the feature
- σ= Standard deviation

Example

Salary values ranging from ₹20,000 to ₹2,00,000 are standardized to have a mean of 0 and a standard deviation of 1.

3. Log Transformation

Log transformation is used to reduce the effect of highly skewed data. It compresses large values while preserving the relationships between data points.

Example

Annual incomes such as ₹30,000, ₹50,000, and ₹10,00,000 can be transformed using logarithms to reduce skewness.

4. Encoding Categorical Variables

Machine learning algorithms cannot directly process text-based categorical data. Therefore, categorical values must be converted into numerical values.

Label Encoding

Each category is assigned a unique integer.

Example:

Gender	Encoded Value
Male	0
Female	1

One-Hot Encoding

Each category is represented by a separate binary column.

Example:

Color	Red	Green	Blue
Red	1	0	0
Green	0	1	0
Blue	0	0	1

5. Binning (Discretization)

Binning converts continuous numerical values into discrete intervals or groups. It simplifies the data and reduces the effect of small variations.

Example

Age values can be grouped as:

- Child (0–12)
- Teenager (13–19)
- Adult (20–59)
- Senior Citizen (60 and above)

Advantages of Feature Transformation:

- **Improves Prediction Accuracy:** Feature transformation converts raw data into a suitable format, enabling machine learning models to learn patterns more effectively and produce more accurate predictions.
- **Converts Data into a Suitable Format:** It transforms numerical and categorical data into formats that can be efficiently processed by machine learning algorithms.
- **Handles Different Feature Scales:** Feature transformation brings features with different ranges to a common scale, preventing features with larger values from dominating the learning process.
- **Reduces Skewness in Data:** Techniques such as log transformation reduce the effect of highly skewed data, resulting in a more balanced data distribution.
- **Improves Convergence of Learning Algorithms:** Properly transformed features help optimization algorithms converge faster, reducing the time required for model training.
- **Enhances Model Efficiency:** By providing well-structured and standardized input data, feature transformation improves the overall efficiency and performance of machine learning models.
- **Reduces Computational Complexity:** Transforming and scaling features simplify data processing, reduce unnecessary computations, and make model training faster and more efficient

Limitations of Feature Transformation:

- **Selecting an Inappropriate Transformation May Reduce Model Performance:** Choosing an unsuitable transformation technique can distort the original data, leading to lower prediction accuracy and poor model performance.
- **Increases Preprocessing Time:** Some feature transformation techniques require additional preprocessing steps, which increase the time and computational effort before model training.
- **Requires Domain Knowledge:** Selecting the most appropriate transformation method often requires a

good understanding of the dataset and the problem domain. Without sufficient domain knowledge, inappropriate transformations may be applied.

- **May Reduce Interpretability:** After transformation, the new feature values may become difficult to understand and interpret, making it harder to explain the model's predictions.
- **May Cause Information Loss:** Some transformation techniques, such as binning or dimensionality reduction, may remove useful information from the original data, affecting the model's performance.
- **Not Suitable for All Algorithms:** Certain machine learning algorithms do not require feature transformation. Applying unnecessary transformations may not improve performance and can sometimes reduce model efficiency.
- **Requires Additional Storage and Processing:** Creating transformed features may increase memory usage and computational requirements, especially for large datasets

Applications of Feature Transformation:

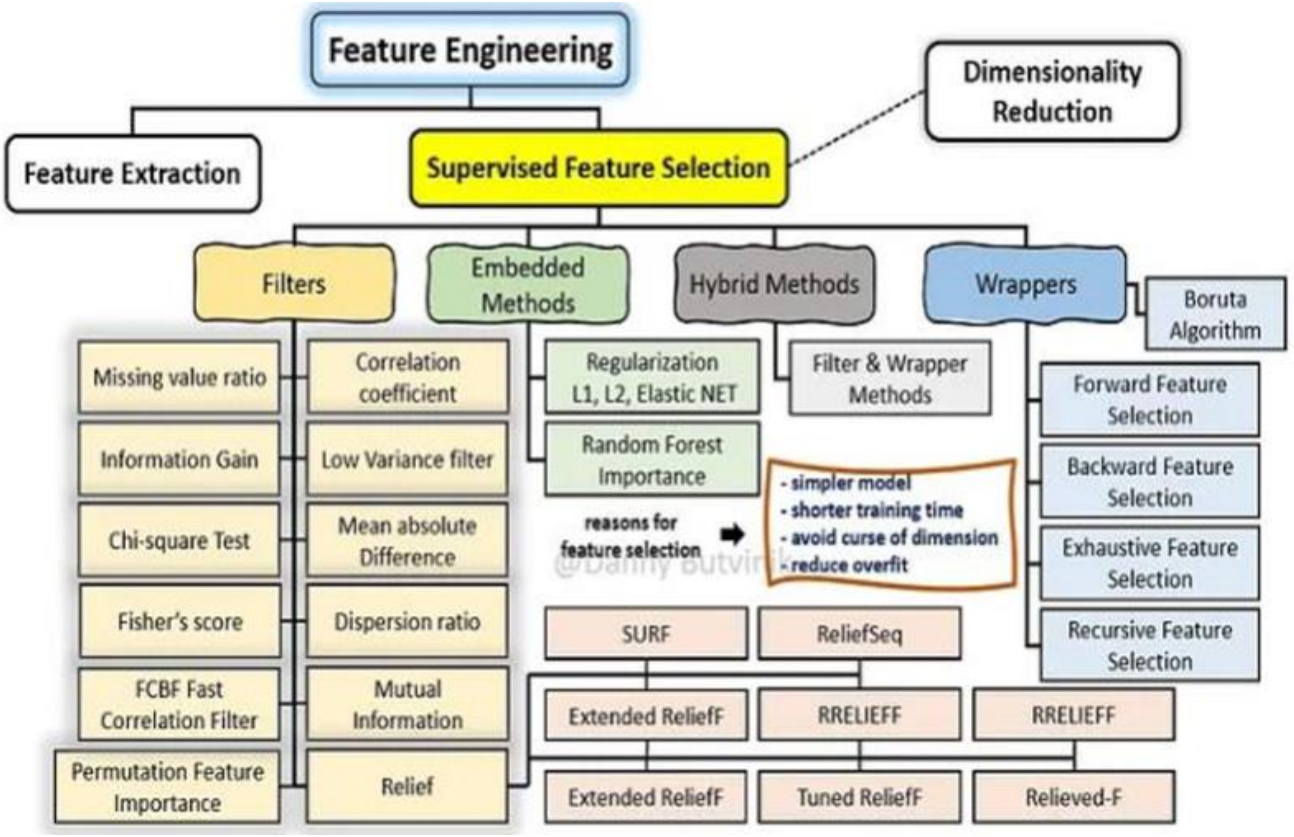
- **Healthcare:** Feature transformation is used to standardize medical data such as age, blood pressure, and cholesterol levels, improving the accuracy of disease prediction models.
- **Finance and Banking:** It is used to transform financial data, such as customer income, credit scores, and transaction amounts, to improve loan approval, credit risk assessment, and fraud detection.
- **House Price Prediction:** Feature transformation normalizes features such as house area, number of rooms, and property age, enabling accurate prediction of house prices.
- **Spam Email Detection:** Text data is transformed into numerical features using techniques such as encoding and vectorization, allowing machine learning models to classify emails as spam or non-spam.
- **Image Recognition:** Feature transformation converts raw pixel values into meaningful features such as edges, textures, and shapes, improving image classification accuracy.
- **Fraud Detection:** Financial institutions transform transaction data into meaningful features to identify unusual patterns and detect fraudulent activities.
- **Weather Forecasting:** Meteorological data such as temperature, humidity, and wind speed are transformed into suitable formats to improve weather prediction models.

Feature Subset Selection

Feature Subset Selection is the process of selecting a subset of the most relevant and useful features from

the original dataset while removing irrelevant, redundant, or less important features. The main objective is to improve the performance of the machine learning model by reducing the dimensionality of the data.

- Helps remove irrelevant and redundant features
- Improves accuracy and reduces overfitting
- Speeds up model training
- Makes models simpler and easier to interpret



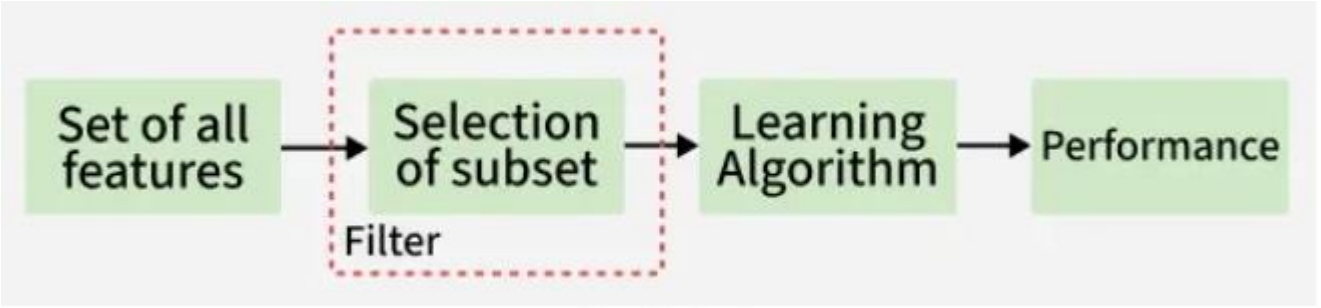
Types of Feature Selection Methods:

There are various algorithms used for feature selection and are grouped into three main categories and each one has its own strengths and trade-offs depending on the use case

Filter Methods:

- Filter methods are feature selection techniques that evaluate each feature independently with respect to the target variable.
- They use statistical measures to determine the importance or relevance of each feature.
- Features that have high relevance to the target variable are selected, while irrelevant or redundant features are removed.
- These methods do not depend on any machine learning algorithm for feature selection.
- Filter methods are fast, simple, and computationally efficient, making them suitable for large datasets.

- They are commonly applied during the data preprocessing stage, before training the machine learning model.
- Filter methods help reduce the dimensionality of the dataset by selecting only the most useful features.



Filter Method

- They improve model performance by removing unnecessary features and reducing computational complexity.
- Common statistical techniques used in filter methods include Correlation Coefficient, Chi-Square Test, Information Gain, Mutual Information, and Variance Threshold.
- Filter methods are widely used as an initial step in feature selection because they are easy to implement and independent of the learning algorithm.

Common Filter Techniques

1. Information Gain

- **Information Gain** measures the reduction in entropy (uncertainty) when a feature is used to split the data.
- Features with higher information gain provide more useful information for predicting the target variable.
- It is commonly used in **Decision Tree** algorithms.

2. Chi-Square Test

- The **Chi-Square Test** measures the relationship between categorical input features and the target variable.
- Features that have a strong association with the target variable receive higher Chi-Square values and are selected.
- It is mainly used for **categorical data**.

3. Fisher's Score

- **Fisher's Score** ranks features based on their ability to separate different classes.
- Features with higher Fisher's Score provide better discrimination between classes.
- It is commonly used in classification problems.

4. Pearson's Correlation Coefficient

- **Pearson's Correlation Coefficient** measures the strength and direction of the linear relationship between two continuous variables.
- The correlation value ranges from **–1 to +1**.
- Features with a high positive or negative correlation with the target variable are considered more important.

5. Variance Threshold

- **Variance Threshold** removes features that have very low variance because they provide little or no useful information for prediction.
- Features with higher variance are generally more informative.

6. Mean Absolute Difference (MAD)

- **Mean Absolute Difference** measures the average absolute difference between feature values and their mean.
- Features with larger mean absolute differences usually contain more useful information for classification or prediction.

7. Dispersion Ratio

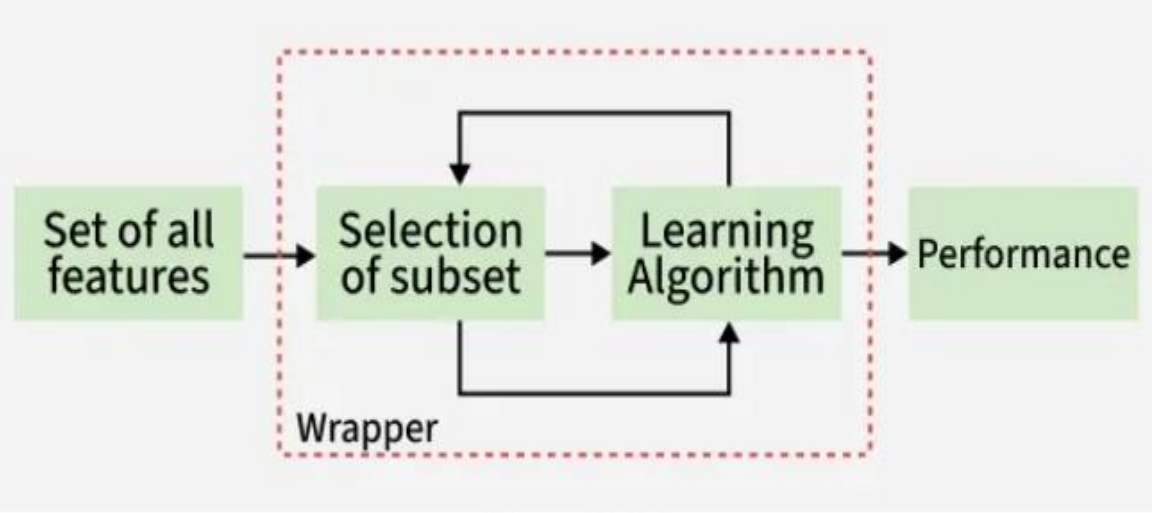
- **Dispersion Ratio** is the ratio of the **arithmetic mean** to the **geometric mean** of a feature.
- Features with higher dispersion ratios are considered more useful because they show greater variability and discrimination among data points.

Wrapper methods:

Wrapper methods are feature selection techniques that evaluate different combinations of features by measuring their impact on model performance. They use search strategies to add or remove features and select the optimal subset based on predefined stopping criteria.

- Evaluates feature subsets using a machine learning model
- Uses greedy or non-greedy search strategies
- Measures the relationship between feature subsets and the target variable
- Adds or removes features based on model performance

- Stops when performance decreases or the desired number of features is reached



Wrapper Method

Common Wrapper Techniques:

1. Forward Selection

- **Forward Selection** is a wrapper feature selection technique that starts with **no features** in the model.
- Features are added **one at a time** based on their contribution to improving the model's performance.
- At each step, the feature that provides the greatest improvement is selected.
- The process continues until adding more features does not significantly improve the model.

2. Backward Elimination

- **Backward Elimination** starts with **all available features** in the dataset.
- At each step, the **least important feature** is removed based on its effect on model performance.
- The process continues until only the most significant features remain.
- This method is useful when the dataset contains many features.

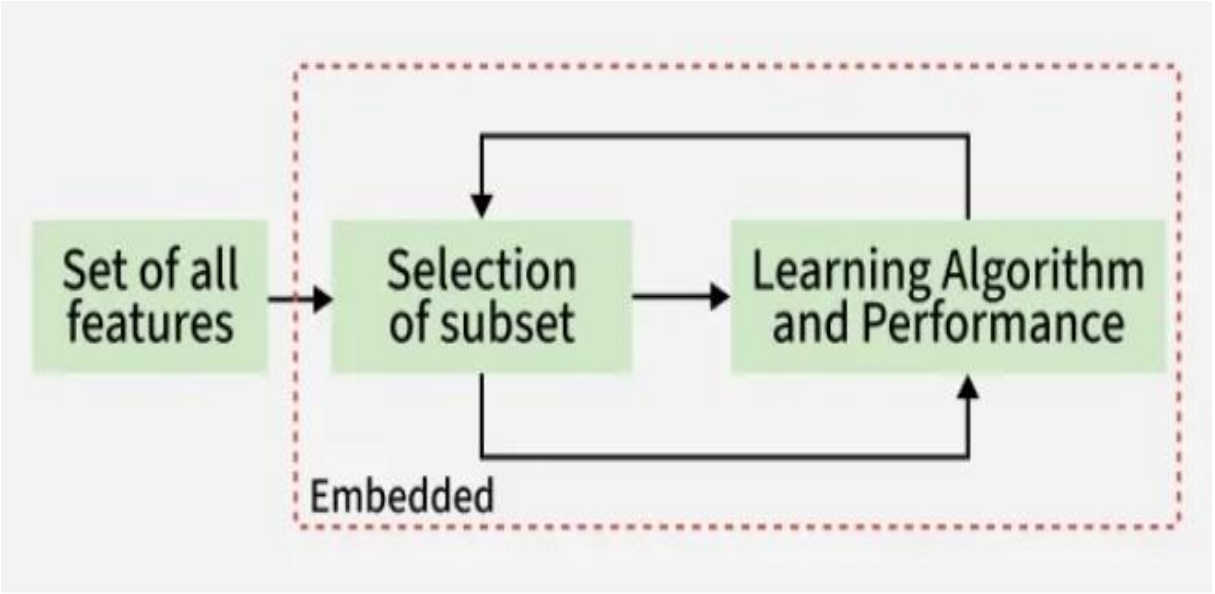
3. Recursive Feature Elimination (RFE)

- **Recursive Feature Elimination (RFE)** is a wrapper method that repeatedly trains the model and removes the **least important feature** in each iteration.
- The process continues until the desired number of features is selected.
- RFE ranks features according to their importance and identifies the best subset for the model.
- It generally provides better feature selection but requires more computational time.

Embedded Methods

Embedded methods perform feature selection during the model training process. They combine the benefits of both filter and wrapper methods. Feature selection is integrated into the model training allowing the model to select the most relevant features based on the training process dynamically.

- Embedded methods are feature selection techniques in which the selection of important features is performed automatically during the training of the machine learning model.
- These methods combine feature selection and model training into a single process.
- They select the most relevant features while the model is learning from the data.
- Embedded methods reduce computational cost compared to wrapper methods because feature selection is integrated with model building.



Embedded Method

Common Embedded Techniques

1. LASSO (Least Absolute Shrinkage and Selection Operator)

- Uses **L1 regularization** to perform feature selection.
- Reduces the coefficients of less important features to **zero**, effectively removing them from the model.
- Helps reduce overfitting and improves model interpretability.

2. Ridge Regression

- Uses **L2 regularization** to reduce the coefficients of less important features.
- Reduces feature importance but **does not eliminate** any feature completely.
- Improves model stability and reduces overfitting.

3. Decision Tree Feature Importance

- Automatically calculates the importance of each feature during model training.
- Features that contribute more to decision making receive higher importance scores.
- Less important features receive lower scores and may be ignored.

4. Random Forest Feature Importance

- Determines feature importance by combining the results of multiple decision trees.
- Assigns higher importance scores to features that contribute most to prediction accuracy.
- Suitable for identifying important features in large datasets.

Advantages of Feature Subset Selection:

- **Feature subset selection improves the prediction accuracy** of a machine learning model by selecting only the most relevant features for training.
- **It reduces overfitting** by removing irrelevant and redundant features, enabling the model to generalize better to unseen data.
- **It decreases the training time** because the model is trained using a smaller number of important features.
- **It reduces computational complexity** by minimizing the amount of data that needs to be processed during model training.
- **It improves the interpretability of the model** by making it simpler and easier to understand and explain.

Limitations of Feature Subset Selection:

- **An inappropriate feature selection method may remove important features**, which can reduce the prediction accuracy of the model.
- **Some feature selection techniques, especially wrapper methods, are computationally expensive** and require more processing time.
- **Feature subset selection often requires domain knowledge** to identify the most relevant features for a particular problem.

- **Different feature selection methods may produce different subsets of features**, leading to variations in model performance.
- **Some feature selection methods evaluate features independently** and may ignore the interactions between multiple features.

Applications of Feature Subset Selection:

- **Feature subset selection is widely used in healthcare** to identify the most important medical features for disease diagnosis and prediction.
- **It is used in banking and finance** to select relevant features for credit scoring, loan approval, and fraud detection.
- **It is applied in customer churn prediction** to identify the factors that influence whether a customer is likely to leave a service.
- **It is used in spam email detection** to select the most important words and phrases for classifying emails as spam or non-spam.
- **It is used in house price prediction** to select relevant features such as location, area, and the number of bedrooms for accurate price estimation.

Case Study

Real-World Example of Model Selection

Example: Predicting Student Performance

A university wants to predict whether students will achieve high academic performance based on factors such as **attendance, internal assessment marks, study hours, assignment scores, and previous semester GPA**. The university plans to use the prediction to identify students who may need additional academic support.

Step 1: Define the Problem

The objective is to predict the **final examination marks** of students.

- **Problem Type:** Regression
- **Target Variable:** Final Examination Marks
- **Independent Variables:** Attendance, Study Hours, Assignment Scores, Internal Marks, Previous GPA

Step 2: Collect and Prepare the Data

The university collects historical student records and preprocesses the data by:

- Removing duplicate records.
- Handling missing values.
- Normalizing numerical features.
- Splitting the dataset into training and testing sets.

Step 3: Select Candidate Models

The university considers the following regression models:

- Linear Regression
- Ridge Regression
- LASSO Regression
- Decision Tree Regression
- Random Forest Regression

Step 4: Train the Models

Each model is trained using **80% of the dataset**, while the remaining **20%** is reserved for testing.

Step 5: Evaluate the Models

The performance of each model is evaluated using **MAE, RMSE, and R^2**.

Model	MAE	RMSE	R^2
Linear Regression	5.2	6.4	0.84
Ridge Regression	4.8	5.9	0.87
LASSO Regression	4.9	6.0	0.86
Decision Tree Regression	5.5	6.8	0.82
Random Forest Regression	**3.1**	**4.2**	**0.94**

Step 6: Select the Best Model

The university compares the performance of all models. **Random Forest Regression** is selected because it has:

- The **lowest MAE**, indicating smaller prediction errors.
- The **lowest RMSE**, showing more accurate predictions.

- The **highest R^2 value (0.94)**, indicating that the model explains **94% of the variation** in students' final examination marks.

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, “Machine Learning”, Wiley Publisher,2019.
2. SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, “Machine Learning”,Pearson,2019.
3. Alpaydin Ethem, “Introduction to Machine Learning”, 3rd Edition, PHI learningprivate limited, 2019.

REFERENCE BOOKS:

1. “Machine Learning: A Probabilistic Perspective”, 2012, Kevin P. Murphy, MIT Press.
2. “Pattern Recognition and Machine Learning”, 2007, Christopher Bishop, Springer.
3. “Introduction to Machine Learning”, MIT Press, 3/e, 2014, Ethem Alpaydin

Question Bank:

PART-A		
Q.No.	**Questions**	**Blooms Taxonomy Level**
1.	What is model selection?	L1
2.	Define model training.	L1
3.	What is model interpretability?	L2
4.	Define model evaluation.	L1
5.	What is overfitting?	L2
6.	What is underfitting?	L2
7.	Define Feature Engineering.	L1
8.	What is Feature Transformation?	L2
9.	Define Feature Subset Selection.	L1
10.	Mention two techniques to improve model performance.	L2
PART-A		
1.	Explain the process of selecting and training a Machine Learning model.	L2
2.	Describe Model Representation and Model Interpretability with examples.	L2
3.	Explain different methods used to evaluate the performance of a Machine Learning model.	L2
4.	Discuss techniques used to improve the performance of a Machine Learning model.	L3
5.	Explain the concept and importance of Feature Engineering.	L2
6.	Describe Feature Transformation techniques with suitable examples.	L3
7.	Explain Feature Subset Selection methods and their advantages.	L3
8.	Analyze the impact of Feature Engineering on model performance.	L4

UNIT III

REGRESSION

"The goal is to turn data into information, and information into insight." — Carly Fiorina

Unit Overview

Regression is one of the most important supervised learning techniques used to predict continuous numerical values based on historical data. It helps in understanding the relationship between dependent and independent variables and is widely used in forecasting, trend analysis, finance, healthcare, agriculture, and business analytics. This unit introduces the concepts of supervised learning and regression, different regression techniques such as Linear Regression, Multiple Linear Regression, Ridge Regression, and LASSO Regression, along with gradient-based optimization methods. It also explains cost functions, optimization techniques, and evaluation metrics used to measure the performance of regression models.

Objectives of the Unit

After studying this unit, students will be able to:

- Understand the concept of supervised learning and regression.
- Explain the working of Linear Regression and Multiple Linear Regression.
- Analyze the importance of model evaluation in regression.
- Understand the concept of regularization and its role in preventing overfitting.
- Differentiate between Ridge Regression and LASSO Regression.
- Explain gradient-based optimization methods used in regression.
- Understand cost functions and their minimization techniques.
- Evaluate regression models using various performance metrics.
- Apply regression algorithms to solve real-world prediction problems.
- Develop efficient regression models with improved prediction accuracy.

Learning Outcomes

Upon successful completion of this unit, students will be able to:

- Explain the principles of supervised learning and regression.
- Build and interpret Linear Regression and Multiple Linear Regression models.
- Apply Ridge and LASSO regression techniques for regularization.
- Use gradient descent methods to optimize regression models.
- Calculate and interpret regression cost functions.
- Minimize cost functions for single-variable and multi-variable regression.
- Evaluate regression models using MAE, MSE, RMSE, RMSLE, R^2, and Adjusted R^2.
- Compare different regression models based on their performance.
- Select appropriate regression techniques for different datasets.
- Apply regression concepts in real-world machine learning applications.

Importance of Studying this Unit

- Regression is one of the most widely used predictive techniques in Machine Learning.
- It helps predict continuous values such as house prices, stock prices, temperature, and sales.
- It forms the foundation for advanced predictive analytics.
- It enables organizations to make data-driven decisions.
- Regularization techniques improve model generalization and reduce overfitting.
- Gradient-based optimization is fundamental to training many Machine Learning models.
- Cost functions help measure prediction errors during model training.
- Evaluation metrics provide quantitative measures of model accuracy.
- Regression techniques are extensively used in research, industry, healthcare, finance, agriculture, and engineering.
- Understanding regression is essential before learning advanced Deep Learning algorithms.

Key Concepts

- **Supervised Learning:** Supervised Learning is a Machine Learning technique that learns from labeled data to make predictions.
- **Regression:** Regression is a supervised learning method used to predict continuous numerical values.
- **Dependent Variable (Target Variable):** The dependent variable is the output value that the model predicts.
- **Independent Variable (Feature):** An independent variable is an input feature used to predict the target variable.
- **Linear Regression:** Linear Regression predicts a target value using a straight-line relationship between variables.
- **Multiple Linear Regression:** Multiple Linear Regression uses two or more independent variables to predict a dependent variable.
- **Cost Function:** A cost function measures the prediction error of a regression model.
- **Gradient Descent:** Gradient Descent is an optimization algorithm used to minimize the cost function.
- **Learning Rate:** The learning rate determines the step size used to update model parameters during training.
- **Ridge Regression:** Ridge Regression uses L2 regularization to reduce overfitting by shrinking coefficient values.
- **LASSO Regression:** LASSO Regression uses L1 regularization to reduce overfitting and perform

feature selection.

- **Regularization:** Regularization is a technique used to improve model performance by reducing overfitting.
- **Overfitting:** Overfitting occurs when a model performs well on training data but poorly on new data.
- **Mean Squared Error (MSE):** MSE measures the average squared difference between actual and predicted values.
- **R Squared (R^2):** R^2 measures how well a regression model explains the variation in the target variable.

Introduction to Supervised Learning and Regression

Introduction to Supervised Learning

Supervised Learning is one of the most widely used types of Machine Learning in which a model is trained using **labeled data**. In this approach, each input (feature) is associated with a correct output (label). The objective is to enable the model to learn the relationship between the input and output so that it can accurately predict the output for new, unseen data.

The learning process involves providing the algorithm with a training dataset, allowing it to identify patterns, and then evaluating its performance using a testing dataset. Supervised learning is mainly used for **classification** (predicting categories) and **regression** (predicting numerical values).

Examples of Supervised Learning

- Predicting house prices based on area and location.
- Predicting student marks based on study hours.
- Email spam detection.
- Disease prediction using patient health records.
- Sales forecasting.

Introduction to Regression

Regression is a supervised learning technique used to predict continuous numerical values by establishing a relationship between one or more independent variables (features) and a dependent variable (target).

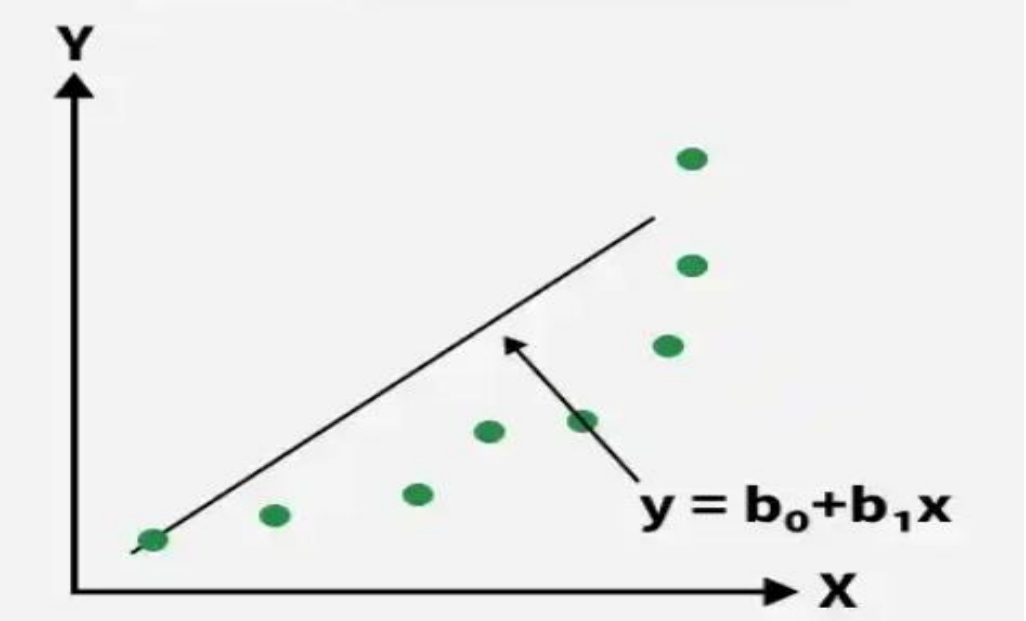
The main objective of regression is to estimate how changes in the input variables affect the output variable. It helps in forecasting, trend analysis, and decision-making by predicting future values from historical data.

The simplest regression algorithm is **Linear Regression**, which models the relationship between variables using a straight-line equation. More advanced regression techniques, such as **Multiple Linear**

Regression, Ridge Regression, and **LASSO Regression**, are used when datasets contain multiple features or require regularization.

Definition:

Regression is a supervised learning technique used to predict continuous numerical values by learning relationships between input variables (features) and an output variable (target). It helps understand how changes in one or more factors influence a measurable outcome and is widely used in forecasting, risk analysis, decision-making and trend estimation.



- Works with real valued output variables
- Helps to identify strengths and the type of relationships
- Supports both simple and complex predictive models.
- Used for tasks like price prediction, trend forecasting and risk scoring.

Types of Regression

Regression can be classified into different types based on the number of predictor variables and the nature of the relationship between variables:

1. Simple Linear Regression

Simple Linear Regression models the relationship between one independent variable and a continuous dependent variable by fitting a straight line that minimizes the sum of squared errors. It assumes a constant rate of change, meaning the output varies proportionally with the input.

- **Application:** Estimating house price from only its size

- **Advantage:** Highly interpretable due to its simple mathematical structure
- **Disadvantage:** Cannot capture curved or complex data patterns

2. Multiple Linear Regression

Multiple Linear Regression extends simple linear regression by incorporating multiple independent variables to predict a continuous outcome. Each predictor is assigned a coefficient that reflects its individual impact while holding other variables constant.

- **Application:** Predicting house prices using multiple factors like size, location, age and number of rooms
- **Advantage:** Captures the combined influence of many factors simultaneously
- **Disadvantage:** Performance drops in the presence of multicollinearity (features highly correlated with each other)

3. Polynomial Regression

Polynomial Regression models non-linear relationships by transforming input features into higher-degree polynomial terms (e.g x^2, x^3). Although it models non-linear relationships in input features, it is linear in coefficients (parameters), which is why it is still considered a linear model.

- **Application:** Modelling curved growth trends like population increase or temperature variation
- **Advantage:** Effectively captures non-linear relationships without switching to non-linear algorithms
- **Disadvantage:** Higher-degree polynomials may lead to overfitting and unstable predictions

4. Ridge and Lasso Regression

Ridge and Lasso are regularized linear regression techniques that add penalty terms to limit large coefficients and reduce overfitting. Ridge (L2) shrinks coefficients smoothly, while Lasso (L1) can reduce some coefficients to zero, enabling feature selection.

- **Application:** Used in high-dimensional datasets like marketing attribution or gene expression data
- **Advantage:** Controls overfitting and improves generalization, especially with many predictors
- **Disadvantage:** Penalty terms make model interpretation less straightforward

5. Support Vector Regression (SVR)

Support Vector Regression applies the principles of Support Vector Machines to regression tasks. It fits a function within a defined margin (epsilon-tube) and penalizes errors only when predictions fall outside this boundary. Kernel functions allow SVR to model non-linear relationships.

- **Application:** Predicting continuous outcomes such as stock values or energy consumption
- **Advantage:** Works well with high-dimensional, complex datasets and non-linear patterns
- **Disadvantage:** Computationally intensive and requires careful tuning of kernels and parameters

6. Decision Tree Regression

Decision Tree Regression splits the data into hierarchical branches based on feature thresholds. Each internal node represents a decision question and leaf nodes represent predicted continuous values. It learns patterns by recursively partitioning the data to minimize prediction errors.

- **Application:** Predicting customer spending behavior based on demographic and financial features
- **Advantage:** Easy to visualize and understand decision logic
- **Disadvantage:** Easily overfits, especially when the tree becomes deep and complex

7. Random Forest Regression

Random Forest Regression is an ensemble method that builds multiple decision trees using different data samples and averages their predictions. This reduces the overfitting tendency of single trees and improves accuracy through diversity (bagging). Each tree captures a slightly different aspect of the data.

- **Application:** Sales forecasting, demand planning, churn prediction
- **Advantage:** High accuracy and robust performance even on noisy datasets
- **Disadvantage:** Acts as a black-box model, making interpretation difficult due to many trees

Regression Evaluation Metrics

Evaluation in machine learning measures the performance of a model. Here are some popular evaluation metrics for regression:

- **Mean Absolute Error (MAE):** The average absolute difference between the predicted and actual values of the target variable.
- **Mean Squared Error (MSE):** The average squared difference between the predicted and actual values of the target variable.
- **Root Mean Squared Error (RMSE):** Square root of the mean squared error.
- **Huber Loss:** A hybrid loss function that transitions from MAE to MSE for larger errors, providing balance between robustness and MSE's sensitivity to outliers.
- **R2 – Score:** Higher values indicate better fit ranging from 0 to 1.

Linear Regression

Linear Regression is a fundamental supervised learning algorithm used to model the relationship between a dependent variable and one or more independent variables. It predicts continuous values by fitting a straight line that best represents the data.

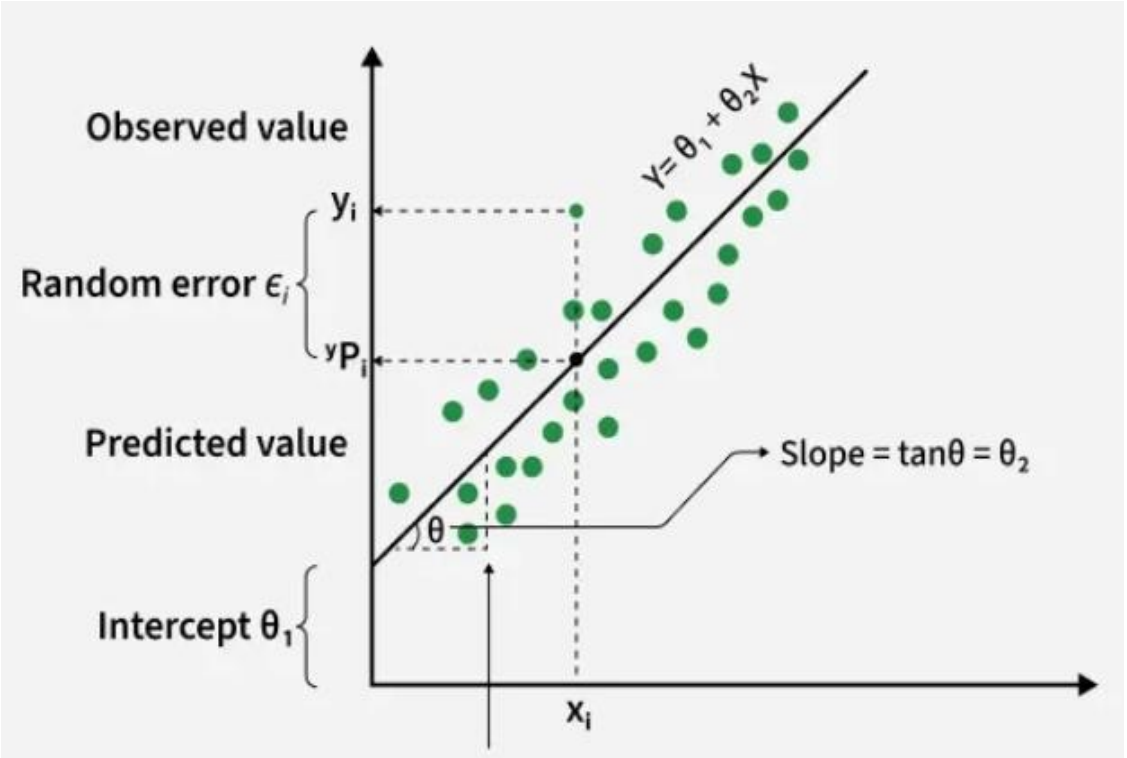
- It assumes that there is a linear relationship between the input and output
- Uses a best-fit line to make predictions
- Commonly used in forecasting, trend analysis, and predictive modelling

Best Fit Line

In linear regression, the best-fit line is the straight line that best represents the relationship between the independent variable (input) and the dependent variable (output). The goal is to minimize the difference between the actual data points and the predicted values generated by the model.

1. Goal of the Best-Fit Line

The goal of linear regression is to find a straight line that minimizes the error (the difference) between the observed data points and the predicted values. This line helps us predict the dependent variable for new, unseen data.



Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y.

- θ_1 represents the intercept, which is the value of Y when X = 0
- θ_2 represents the slope, which shows how much Y changes for a unit change in X

There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

2. Equation of the Best-Fit Line

For simple linear regression (with one independent variable), the best-fit line is represented by the equation

$$y = mx + b$$

Where:

- **y** is the predicted value (dependent variable)

- **x** is the input (independent variable)
- **m** is the slope of the line (how much y changes when x changes)
- **b** is the intercept (the value of y when x = 0)

The best-fit line will be the one that optimizes the values of m (slope) and b (intercept) so that the predicted y values are as close as possible to the actual data points.

3. Minimizing the Error using the Least Squares Method

To determine the best-fit line, linear regression uses the Least Squares Method, which minimizes the difference between actual and predicted values. These differences are called residuals. These differences are called residuals.

The formula for residuals is:

$Residual = y_i - \hat{y}_i$

Where:

- y_i is the actual observed value
- $\hat{y}_i$ is the predicted value from the line for that x_i

The least squares method minimizes the sum of the squared residuals:

$$\Sigma(y_i - \hat{y}_i)^2$$

This method ensures that the line best represents the data where the sum of the squared differences between the predicted values and actual values is as small as possible.

4. Interpretation of the Best-Fit Line

- **Slope (m):** The slope indicates how much the dependent variable changes for every one-unit increase in the independent variable. For example, if the slope is 5, then y increases by 5 units for every 1-unit increase in x.
- **Intercept (b):** The intercept represents the predicted value of y when x = 0. It's the point where the line crosses the y-axis.

In linear regression some hypothesis are made to ensure reliability of the model's results.

Limitations:

- ***Assumes Linearity:*** *The method assumes the relationship between the variables is linear. If the relationship is non-linear, linear regression might not work well.*
- ***Sensitivity to Outliers:*** *Outliers can significantly affect the slope and intercept, skewing the best-fit line.*

Hypothesis Function

In linear regression, the hypothesis function is the equation used to make predictions about the dependent variable based on the independent variables. It represents the relationship between the input features and the target output.

For a simple case with one independent variable, the hypothesis function is:

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

Where:

- $h(x)$ or $(\hat{y})$ is the predicted value of the dependent variable (y).
- $x$is the independent variable.
- β_0 is the intercept, representing the value of y when x is 0.
- β_1 is the slope, indicating how much y changes for each unit change in x.

For multiple linear regression (with more than one independent variable), the hypothesis function expands to:

$$h(x_1, x_2, \ldots, x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Where:

- $x_1, x_2, \ldots, x_k$ are the independent variables.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients, representing the influence of each respective independent variable on the predicted output.

Cost Function

In Linear Regression, the cost function measures how far the predicted values $\hat{Y}$ are from the actual values (Y). It helps identify and reduce errors to find the best-fit line. The most common cost function used is Mean Squared Error (MSE), which calculates the average of squared differences between actual and predicted values.

$$\text{Cost function}(J) = \frac{1}{n}\sum_{n}^{i} (\hat{y_i} - y_i)^2$$

Here:

- $\hat{y_i} = \theta_1 + \theta_2 x_i$: It is used to minimize this cost, we use Gradient Descent, which iteratively updates θ_1 and θ_2 until the MSE reaches its lowest value. This ensures the line fits the data as accurately as possible.

Gradient Descent

Gradient descent is an optimization technique used to train a linear regression model by minimizing the prediction error. It works by starting with random model parameters and repeatedly adjusting them to reduce the difference between predicted and actual values.

Types of Linear Regression

When there is only one independent feature it is known as Simple Linear Regression or Univariate Linear Regression and when there are more than one feature it is known as Multiple Linear Regression or Multivariate Regression.

1. Simple Linear Regression

Simple linear regression is used when we want to predict a target value (dependent variable) using only

one input feature (independent variable). It assumes a straight-line relationship between the two.

Formula:

$$\hat{y} = \theta_0 + \theta_1 x$$

Where:

- $\hat{y}$ is the predicted value
- x is the input (independent variable)
- θ_0 is the intercept (value of $\hat{y}$ when x=0)
- θ_1 is the slope or coefficient (how much $\hat{y}$ changes with one unit of x)

Example: Predicting a person's salary (y) based on their years of experience (x).

2. Multiple Linear Regression

Multiple linear regression involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Where:

- $\hat{y}$ is the predicted value
- $x_1, x_2, \ldots, x_n$ are the independent variables
- $\theta_1, \theta_2, \ldots, \theta_n$ are the coefficients (weights) corresponding to each predictor
- θ_0 is the intercept

The goal is to find the best-fit line that predicts Y accurately for given inputs X.

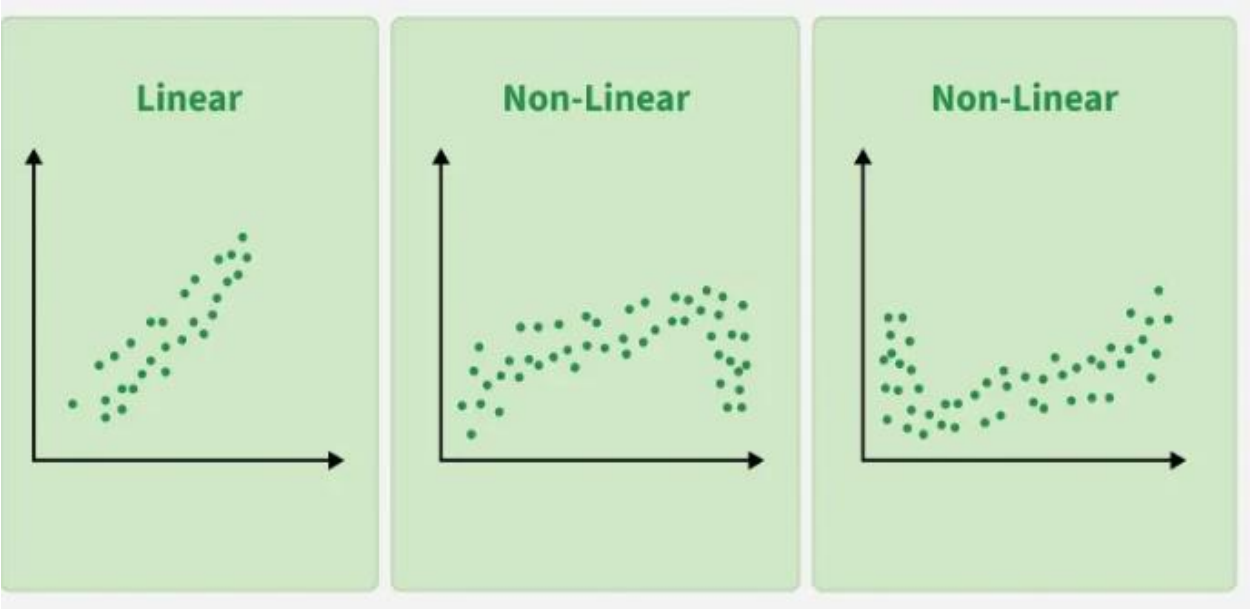
Use Cases:

- **Real Estate:** Predict property prices using location, size and other factors.
- **Finance:** Forecast stock prices using interest rates and inflation data.
- **Agriculture:** Estimate crop yield from rainfall, temperature and soil quality.
- **E-commerce:** Analyze how price, promotions and seasons affect sales.

Assumptions of the Linear Regression

1. Linearity: The relationship between inputs (X) and the output (Y) is a straight line.

2. Independence of Errors: The errors in predictions should not affect each other.



3. Constant Variance (Homoscedasticity): The errors should have equal spread across all values of the input. If the spread changes (like fans out or shrinks), it's called heteroscedasticity and it's a problem for the model.

4. Normality of Errors: The errors should follow a normal (bell-shaped) distribution.

5. No Multicollinearity (for multiple regression): Input variables shouldn't be too closely related to each other.

6. No Autocorrelation: Errors shouldn't show repeating patterns, especially in time-based data.

7. Additivity: The total effect on Y is just the sum of effects from each X, no mixing or interaction between them.

Regularization Techniques

- **Lasso Regression**: Regularizes a linear regression model, it adds a penalty term to the linear regression objective function to prevent overfitting.
- **Ridge regression**: Adds a regularization term to the standard linear objective to prevent overfitting by penalizing large coefficient in linear regression equation. It useful when the dataset has multicollinearity where predictor variables are highly correlated.
- **Elastic Net Regression**: Hybrid regularization technique that combines the power of both L1 and L2 regularization in linear regression objective.

Evaluation of Model Estimators

Model evaluation is the process of assessing how well a machine learning model performs on unseen data. In regression, a **model estimator** is evaluated by measuring how accurately it predicts continuous target values. Proper evaluation helps determine whether the model has learned meaningful patterns from the training data and whether it can generalize well to new data.

The performance of a regression model is commonly evaluated using statistical measures such as **Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Root Mean Squared Log Error (RMSLE), R Squared (R^2),** and **Adjusted R Squared**.

Model Estimator:

A **model estimator** is a machine learning algorithm that learns the relationship between input features (independent variables) and the target variable (dependent variable) from the training data. After learning, it predicts the output for new, unseen data.

1. Cross-Validation

Cross-validation is a statistical technique used to evaluate the performance of a Machine Learning model by dividing the dataset into multiple subsets. It repeatedly trains and tests the model on different portions of the data to estimate how well it will perform on unseen data. Cross-validation helps reduce overfitting and provides a more reliable estimate of the model's prediction accuracy.

Definition

Cross-validation is a model evaluation technique in which the dataset is divided into multiple subsets. The model is trained on one subset and tested on another subset repeatedly. The final performance is obtained by averaging the results from all testing rounds.

Types of Cross-Validation

(a) Holdout Method

The Holdout Method is the simplest and most commonly used cross-validation technique. In this method, the dataset is divided into two separate parts: one for training the model and another for testing it.

Definition

The Holdout Method is a validation technique in which the dataset is divided into a **training set** and a **testing set**. The model learns from the training data and its performance is evaluated using the testing data.

Working Procedure

1. Divide the dataset into training and testing sets.
2. Train the model using the training dataset.
3. Test the model using the testing dataset.
4. Compare the predicted values with the actual values.
5. Calculate the evaluation metrics.

Common Dataset Split

- Training Data – 80%
- Testing Data – 20%

(b) K-Fold Cross-Validation

K-Fold Cross-Validation divides the dataset into **K equal-sized folds**. The model is trained **K times**, where each fold is used once as the testing dataset while the remaining folds are used for training. The final model performance is calculated as the average of all K evaluation results.

Working Procedure

1. Divide the dataset into K equal folds.
2. Select one fold as the testing dataset.
3. Train the model using the remaining K−1 folds.
4. Evaluate the model.
5. Repeat the process until every fold has been used once for testing.
6. Calculate the average performance of all folds.

Example

If **K = 5**, the dataset is divided into five equal parts. The model is trained five times, and each fold serves as the testing dataset once.

Difference Between Holdout Method and K-Fold Cross-Validation

Feature	Holdout Method	K-Fold Cross-Validation
Dataset Division	One train-test split	K equal folds
Number of Training Cycles	One	K
Accuracy	Less reliable	More reliable
Data Utilization	Lower	Better
Computational Cost	Low	High

---

2. Evaluation Metrics for Classification Tasks

Classification models predict **categorical outputs**, such as Yes/No, Spam/Not Spam, or Disease/No Disease. To evaluate the effectiveness of these models, various classification evaluation metrics are used. These metrics compare the predicted class labels with the actual class labels and indicate how accurately the model performs.

Common Evaluation Metrics

1. Accuracy

Accuracy is the proportion of correctly classified instances to the total number of instances.

Formula

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision

Precision is the proportion of correctly predicted positive instances among all predicted positive instances.

Formula

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. Recall (Sensitivity)

Recall is the proportion of actual positive instances that are correctly identified by the model.

Formula

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. F1-Score

The F1-Score is the harmonic mean of Precision and Recall and provides a balanced measure of classification performance.

Formula

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

5. Confusion Matrix

A Confusion Matrix is a table that summarizes the prediction results of a classification model by displaying:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

3. Evaluation Metrics for Regression Tasks

Regression models predict **continuous numerical values**, such as house prices, salaries, or temperatures. Their performance is evaluated using regression metrics that measure the difference between the predicted values and the actual values. Lower prediction errors generally indicate better model performance.

Common Evaluation Metrics

1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is the average of the absolute differences between the actual values and the predicted values. It measures the average magnitude of prediction errors without considering their direction.

Formula

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$

Where:

- y_i= Actual value
- $\hat{y}_i$= Predicted value
- n= Total number of observations

2. Mean Squared Error (MSE)

Mean Squared Error (MSE) is the average of the squared differences between the actual values and the predicted values. Squaring the errors gives greater importance to larger prediction errors.

Formula

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

Where:

- y_i= Actual value
- $\hat{y}_i$= Predicted value
- n= Total number of observations

3. Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is the square root of the Mean Squared Error and measures the prediction error in the same unit as the target variable.

Formula

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

Where:

- y_i= Actual value

- $\hat{y}_i$= Predicted value
- n= Total number of observations

4. Root Mean Squared Log Error (RMSLE)

Root Mean Squared Log Error (RMSLE) measures the logarithmic difference between the actual values and predicted values. It is useful for datasets with large value ranges or exponential growth.

Formula

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[\log(y_i + 1) - \log(\hat{y}_i + 1)]^2}$$

Where:

- y_i= Actual value
- $\hat{y}_i$= Predicted value
- n= Total number of observations

5. R Squared (R^2)

R Squared (R^2), also known as the **Coefficient of Determination**, measures the proportion of variation in the dependent variable explained by the regression model. Its value ranges from **0 to 1**, where values closer to **1** indicate better model performance.

Formula

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

Where:

- y_i= Actual value
- $\hat{y}_i$= Predicted value
- $\bar{y}$= Mean of the actual values
- n= Total number of observations

6. Adjusted R Squared

Adjusted R^2 is an improved version of R^2 that considers the number of independent variables in the regression model. It provides a more accurate evaluation for Multiple Linear Regression by penalizing unnecessary predictor variables.

Formula

$$\text{Adjusted } R^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-p-1}\right)$$

Where:

- R^2= R Squared value
- n= Total number of observations
- p= Number of independent variables (predictors)

Regularization

Introduction

In Machine Learning, a model may become too complex and memorize the training data instead of learning the underlying patterns. This results in overfitting, where the model performs well on training data but poorly on unseen data. Regularization is a technique used to overcome this problem by adding a penalty term to the model's cost function. It simplifies the model by reducing the magnitude of the coefficients and improves its ability to generalize to new data.

Definition

Regularization is a machine learning technique used to prevent overfitting by adding a penalty term to the cost function. It controls the complexity of the model by reducing the values of the regression coefficients, thereby improving prediction performance on unseen data.

Types of Regularization:

1. L1 Regularization (LASSO Regression)

L1 Regularization adds the **absolute values** of the regression coefficients to the cost function. It can reduce some coefficients exactly to zero, thereby performing **feature selection**.

Formula

$$J(\beta) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

2. L2 Regularization (Ridge Regression)

L2 Regularization adds the **squares of the regression coefficients** to the cost function. It reduces the coefficient values but does not make them exactly zero.

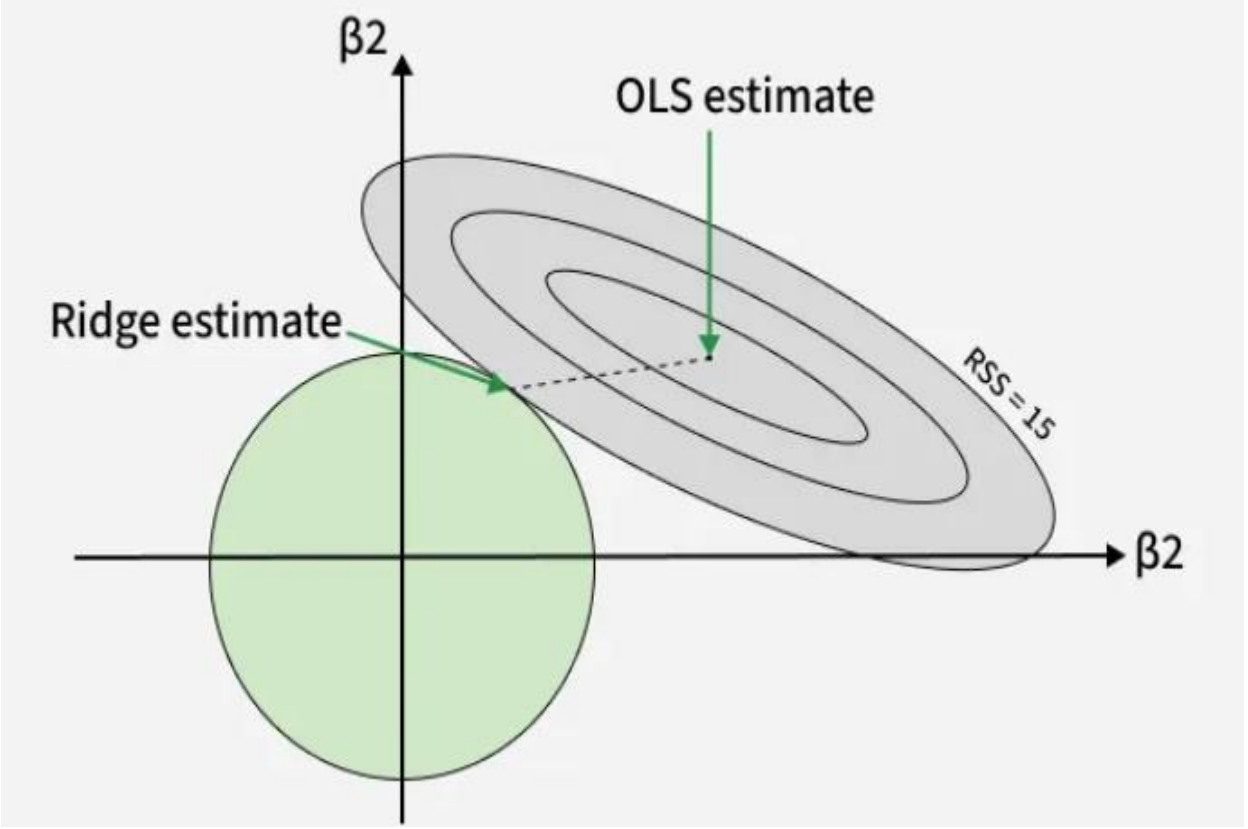
Formula

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

L2 Regularization (Ridge Regression)

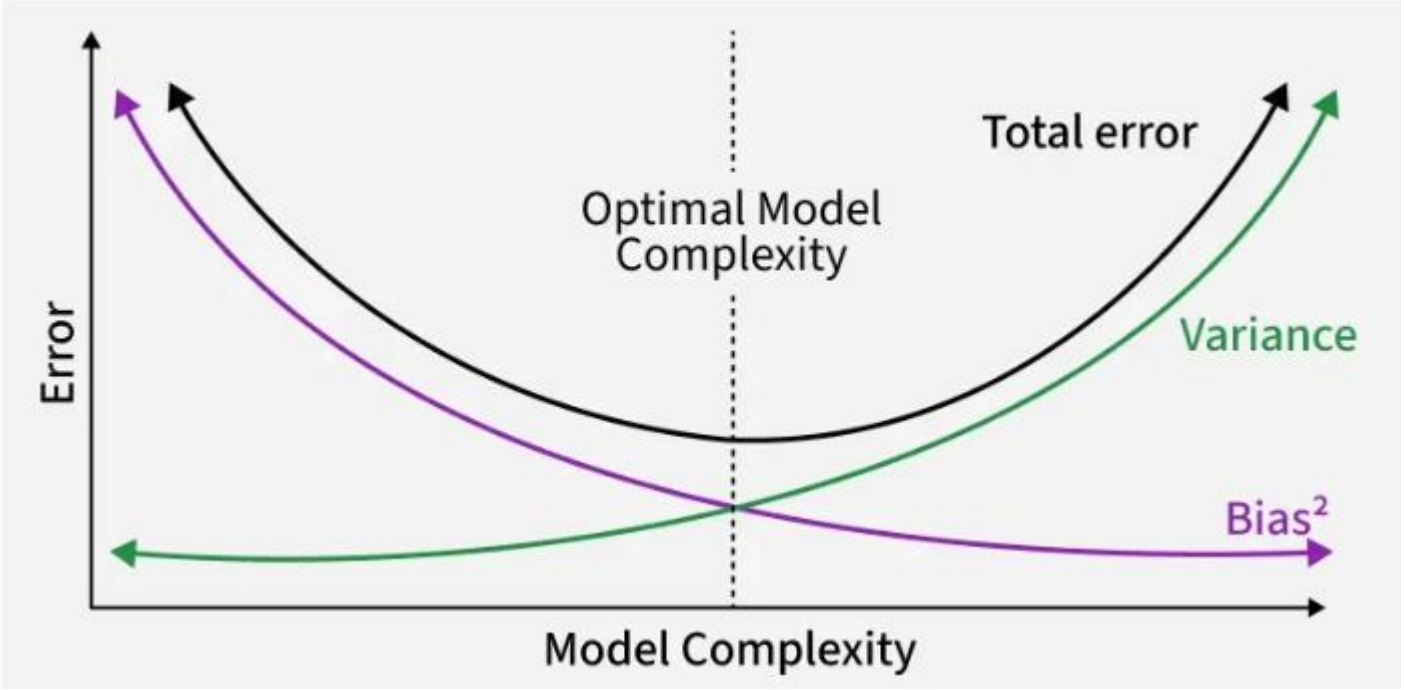
Ridge Regression is a version of linear regression that adds an L2 penalty to control large coefficient values. While Linear Regression only minimizes prediction error, it can become unstable when features are highly correlated. Ridge solves this by shrinking coefficients making the model more stable and reducing overfitting. It helps in:

- **L2 Regularization:** Adds an L2 penalty to model weights
- **Bias-Variance Tradeoff:** Controls how large coefficients can grow
- **Multicollinearity:** Improves stability when features overlap
- **Generalization**: Helps the model generalize better on new data



Bias-Variance Trade-off in Ridge Regression

One of the central ideas behind ridge regression is the bias-variance trade-off:



- **Variance**: In standard linear regression, especially when features are correlated or many, coefficient estimates can vary a lot depending on the specific training data, meaning predictions on new data can be very unstable.
- **Bias**: Ridge regression deliberately introduces some bias by shrinking coefficient magnitudes. This

means the fit to the training data might be slightly worse.

- **Trade-off & Why It Helps**: Ridge shrinks large coefficients hence reducing variance. Even with a small increase in bias, the overall MSE drops, giving better performance than plain linear regression on new data.

Thus, ridge regression accepts a small increase in bias to gain a larger reduction in variance and this tradeoff is often useful when generalization is important.

Selection of the Ridge Parameter:

Choosing the right ridge parameter k is essential because it directly affects the model's bias-variance balance and overall predictive accuracy. Several systematic approaches exist for determining the optimal value of k, each offering unique strengths and considerations. The major methods are:

1. Cross-Validation

Cross-validation selects the ridge parameter by repeatedly training and testing the model on different subsets of data and identifying the value of k that minimizes validation error.

- **K-Fold Cross-Validation:** The dataset is divided into K folds. The model trains on K–1 folds and validates on the remaining fold. This process repeats for all folds and the average error determines the best k.
- **Leave-One-Out Cross-Validation (LOOCV):** A special form of cross-validation where each observation acts once as the validation point. Though computationally expensive, it provides an almost unbiased estimate of prediction error.

2. Generalized Cross-Validation (GCV)

It is an efficient alternative to LOOCV that avoids explicitly splitting the data. It estimates the optimal k by minimizing a function that approximates the LOOCV error.

- Requires fewer computations.
- Often produces results similar to traditional cross-validation.

3. Information Criteria

Model selection metrics like AIC and BIC can also guide the choice of k.

- They balance model fit with complexity.
- Higher penalties discourage overly complex or over-regularized models.

4. Empirical Bayes Methods

These methods treat k as a Bayesian hyperparameter and use observed data to estimate its value.

- **Empirical Bayes Estimation:** A prior distribution is assigned to k and the data are used to update it into a posterior distribution. The posterior mean or mode is then selected as the optimal k.

5. Stability Selection

Stability selection enhances robustness by repeatedly fitting the model on subsampled datasets.

- The ridge parameter that appears most consistently across subsamples is chosen.
- Helps avoid unstable or overly sensitive parameter choices.

LASSO regression

LASSO Regression (Least Absolute Shrinkage and Selection Operator) is a linear regression technique that uses L1 regularization to reduce overfitting by adding the absolute values of regression coefficients to the cost function. It automatically selects important features by reducing less important coefficients to zero.

LASSO Regression Cost Function

LASSO modifies the Linear Regression cost function by adding an **L1 penalty**.

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Where:

- $J(\beta)$= Cost function
- y_i= Actual value
- $\hat{y}_i$= Predicted value
- β_j= Regression coefficient
- n= Number of observations
- p= Number of independent variables
- λ= Regularization parameter

L1 Regularization

L1 Regularization adds the sum of the absolute values of the regression coefficients to the cost function. This penalty encourages sparsity by forcing some regression coefficients to become exactly zero. As a result, less important features are automatically removed from the model, making the regression model simpler and easier to interpret.

Working of LASSO Regression

Step 1: Build the Linear Regression Model

LASSO Regression begins by constructing a standard Linear Regression model that predicts the target variable

using all available independent variables.

Step 2: Add the L1 Regularization Penalty

The algorithm adds an L1 regularization term to the cost function. This penalty increases as the regression coefficients become larger.

Step 3: Minimize the Cost Function

The optimization algorithm minimizes the modified cost function by adjusting the regression coefficients while balancing prediction accuracy and model complexity.

Step 4: Shrink the Regression Coefficients

As the regularization parameter increases, the regression coefficients gradually decrease in magnitude. Less important coefficients shrink more rapidly than important coefficients.

Step 5: Perform Automatic Feature Selection

Some regression coefficients become exactly zero during optimization. The corresponding independent variables are automatically removed from the regression model.

Step 6: Generate Predictions

The final regression model uses only the selected features to predict the target values for new data.

Regularization Parameter (λ)

The regularization parameter **λ (lambda)** determines the strength of the L1 penalty applied to the regression coefficients.

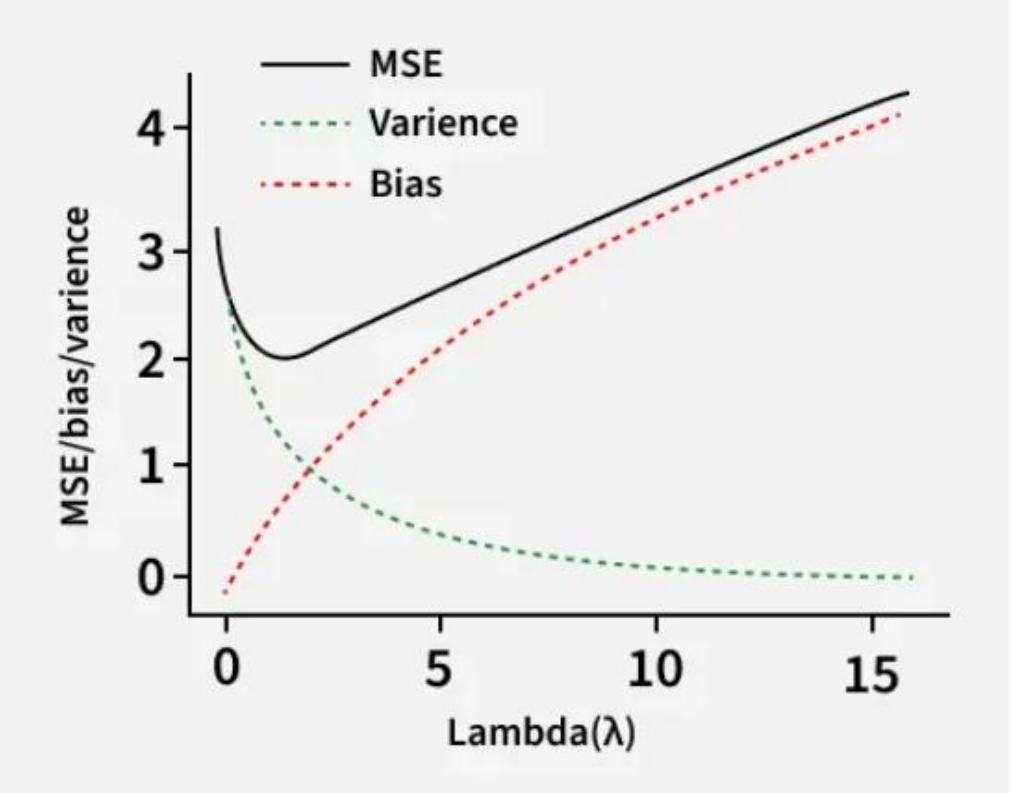
- When **λ = 0**, LASSO Regression behaves exactly like ordinary Linear Regression.
- When **λ is small**, the regression coefficients are only slightly reduced, and most features remain in the model.
- When **λ is large**, many regression coefficients become zero, resulting in a simpler model with fewer features.

Bias–Variance Tradeoff

The **Bias–Variance Tradeoff** describes the balance between model complexity and prediction accuracy.

Low λ (Weak Regularization)

When the value of λ is small, the regression model closely fits the training data. This results in low bias, high variance, and an increased risk of overfitting.



High λ (Strong Regularization)

When the value of λ is large, the regression coefficients are heavily reduced or removed. This increases the bias, reduces the variance, and improves the model's ability to generalize to unseen data.

How LASSO Balances the Bias–Variance Tradeoff

LASSO Regression reduces variance by simplifying the regression model through feature selection. Although this process may slightly increase bias, it generally produces a lower prediction error on unseen data and improves overall model performance

Multi Linear Regression

Multiple Linear Regression is a supervised machine learning algorithm that models the relationship between one dependent variable and two or more independent variables by fitting a linear equation to the observed data. It predicts the value of the dependent variable based on the combined effect of multiple predictor variables.

Steps for Multiple Linear Regression

Steps to perform multiple linear regression are similar to that of simple linear Regression but difference comes in the evaluation process. We can use it to find out which factor has the highest influence on the predicted output and how different variables are related to each other. Equation for multiple linear regression is:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where:

- $y$is the dependent variable
- $X_1, X_2, \cdots X_n$ are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \cdots \beta_n$ are the slopes

The goal of the algorithm is to find the best fit line equation that can predict the values based on the independent variables. A regression model learns from the dataset with known X and y values and uses it to predict y values for unknown X.

Handling Categorical Data with Dummy Variables

In multiple regression model we may encounter categorical data such as gender (male/female), location (urban/rural), etc. Since regression models require numerical inputs then categorical data must be transformed into a usable form. This is where Dummy Variables used. These are binary variables (0 or 1) that represent the presence or absence of each category. For example:

- **Male**: 1 if male, 0 otherwise
- **Female**: 1 if female, 0 otherwise

GENDER	MALE	FEMALE
Male	1	0
Male	1	0
Female	0	1
Female	0	1
Male	1	0
Female	0	1
Male	1	0

In the case of multiple categories we create a dummy variable for each category excluding one to avoid multicollinearity. This process is called one-hot encoding which converts categorical variables into a numerical format suitable for regression models.

Multicollinearity in Multiple Linear Regression

Multicollinearity arises when two or more independent variables are highly correlated with each other. This can make it difficult to find the individual contribution of each variable to the dependent variable.

To detect multi collinearity we can use:

1. **Correlation Matrix:** A correlation matrix helps to find relationships between independent variables. High correlations (close to 1 or -1) suggest multicollinearity.
2. **VIF (Variance Inflation Factor):** VIF quantifies how much the variance of a regression coefficient increases if predictors are correlated. A high VIF typically above 10 indicates multicollinearity.

Assumptions of Multiple Regression Model:

Similar to simple linear regression we have some assumptions in multiple linear regression which are as follows:

1. **Linearity**: Relationship between dependent and independent variables should be linear.
2. **Homoscedasticity**: Variance of errors should remain constant across all levels of independent variables.
3. **Multivariate Normality**: Residuals should follow a normal distribution.
4. **No Multicollinearity**: Independent variables should not be highly correlated.

Gradient-Based Methods

Gradient-Based Methods are optimization algorithms that minimize the cost function by iteratively updating the model parameters in the direction opposite to the gradient of the cost function. The objective is to find the optimal values of the parameters that produce the minimum prediction error.

Gradient Descent Update Rule

The model parameters are updated using the following equation:

$$\theta = \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Where:

- **θ** = Model parameter (weight or coefficient)
- **α** = Learning rate
- **J(θ)** = Cost function
- **∂J(θ)/∂θ** = Gradient of the cost function

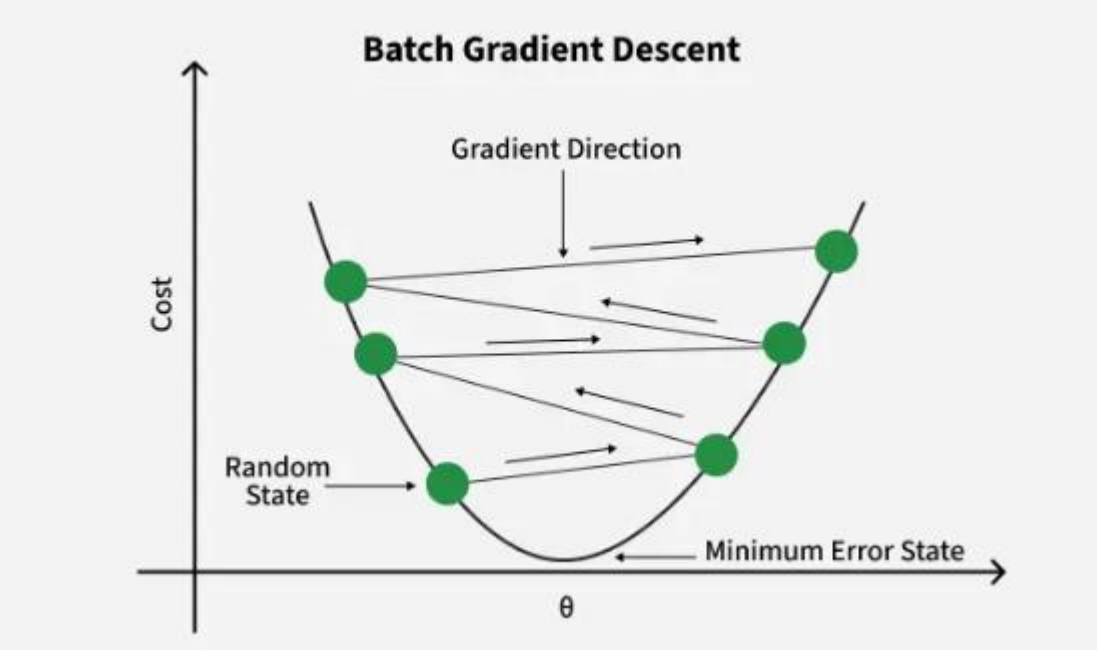
Types of Gradient-Based Methods

Gradient-Based Methods are optimization techniques that minimize the cost function by updating the model parameters in the direction opposite to the gradient. Depending on how the training data is used to compute the gradient, Gradient Descent is classified into three main types:

1. Batch Gradient Descent
2. Stochastic Gradient Descent (SGD)
3. Mini-Batch Gradient Descent

1. Batch Gradient Descent

Batch Gradient Descent is an optimization algorithm in which the gradient of the cost function is calculated using the **entire training dataset** before updating the model parameters. During each iteration, all training samples are processed, and the parameters are updated only once after calculating the average gradient.



The update rule for batch gradient descent is:

where:

- θ represents the parameters of the model
- η is the learning rate
- $\nabla J(\theta)$ is the gradient of the loss function $J(\theta))$ with respect to θ.

Advantages

- **Stable Convergence:** Since the gradient is averaged over all training examples the updates are less noisy and more stable.
- **Global View:** It considers the entire dataset for each update providing a global perspective of the loss landscape.

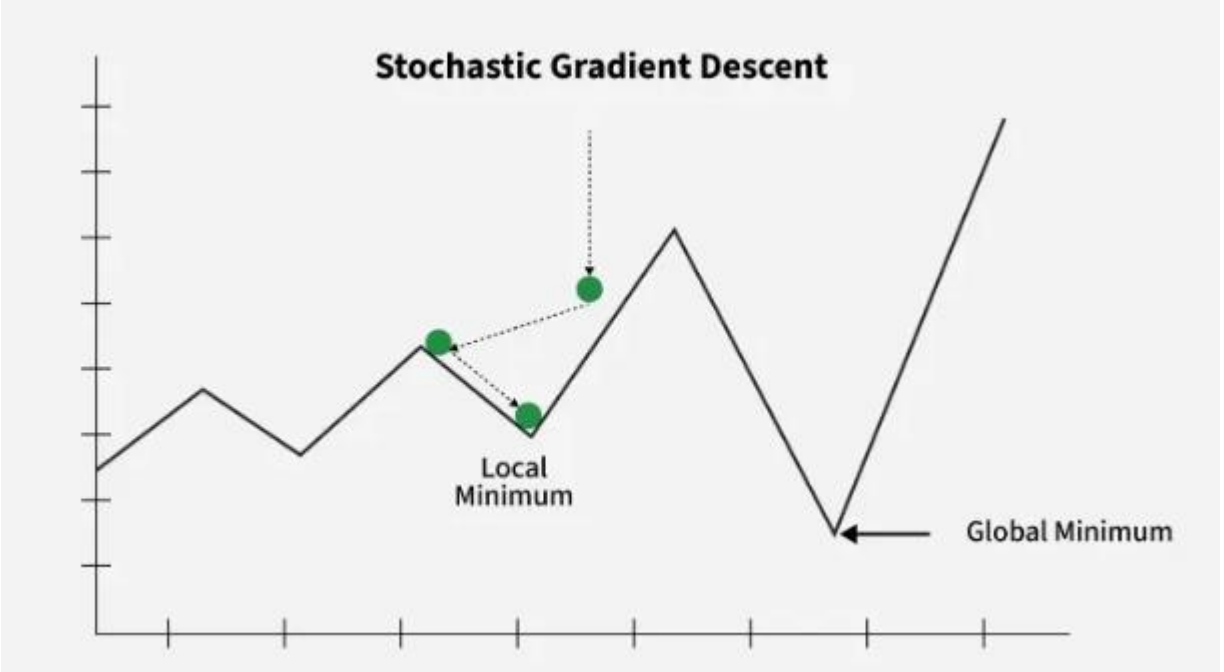
Disadvantages

- **Computationally Expensive:** It Processing the entire dataset in each iteration can be slow and resource-intensive especially for large datasets.

- **Memory Intensive:** This requires storing and processing the entire dataset in memory which can be impractical for very large datasets.

2. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a variant of the gradient descent algorithm where the model parameters are updated using the gradient of the loss function with respect to a single training example at each iteration. Unlike batch gradient descent which uses the entire dataset SGD updates the parameters more frequently, leading to faster convergence.



The update rule for SGD is:

$$\theta = \theta - \eta \nabla J(\theta; x^{(i)}, y^{(i)})$$

where:

- θ represents the parameters of the model,
- $\eta$is the learning rate,
- $\nabla J(\theta; x^{(i)}, y^{(i)})$ is the gradient of the loss function$J(\theta)$ with respect to θ for the i^{th} training example $\left(x^{(i)}, y^{(i)}\right)$.

Advantages

- **Faster Convergence:** Frequent updates can lead to faster convergence, especially in large datasets.
- **Less Memory Intensive:** Since it processes one training example at a time, it requires less memory compared to batch gradient descent.
- **Better for Online Learning:** Suitable for scenarios where data comes in a stream, allowing the model

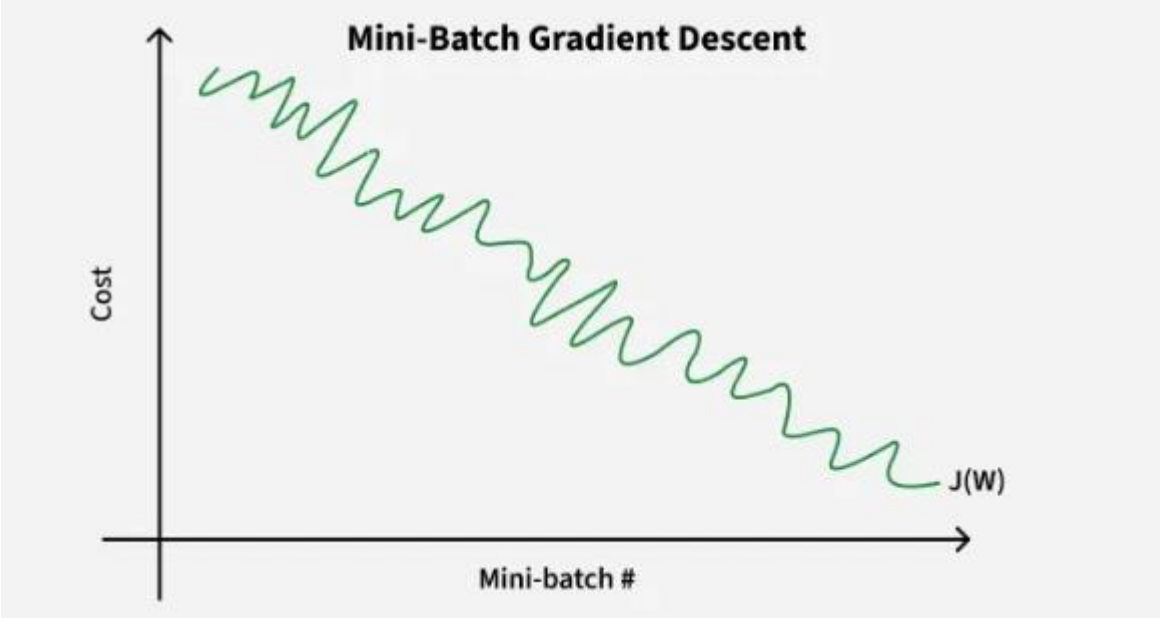
to be updated continuously.

Disadvantages

- **Noisy Updates:** Updates can be noisy, leading to a more erratic convergence path.
- **Potential for Overshooting:** The frequent updates can cause the algorithm to overshoot the minimum, especially with a high learning rate.
- **Hyperparameter Sensitivity:** Requires careful tuning of the learning rate to ensure stable and efficient convergence.

3. Mini-Batch Gradient Descent

Mini-Batch Gradient Descent is a compromise between Batch Gradient Descent and Stochastic Gradient Descent. Instead of using the entire dataset or a single training example Mini-Batch Gradient Descent updates the model parameters using a small, random subset of the training data called a mini-batch.



Update rule for Mini-Batch Gradient Descent is:

$$\theta = \theta - \eta \nabla J(\theta; \{x^{(i)}, y^{(i)}\}_{i=1}^{m})$$

where:

- θ represents the parameters of the model,
- $\eta$is the learning rate,
- $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$represents a mini-batch of mmm training examples,
- $\nabla J(\theta; \{x^{(i)}, y^{(i)}\}_{i=1}^{m})$ is the gradient of the loss function $J(\theta)$ with respect to θ for the mini-batch.

Advantages

- **Faster Convergence:** By using mini-batches, it achieves a balance between the noisy updates of SGD

and the stable updates of Batch Gradient Descent, often leading to faster convergence.

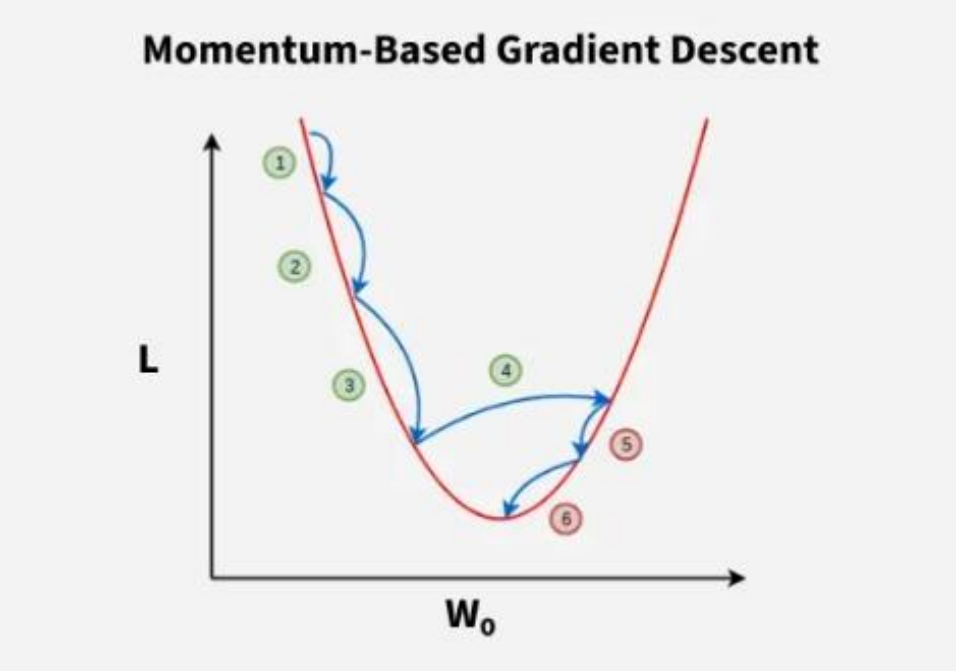
- **Reduced Memory Usage:** Requires less memory than Batch Gradient Descent as it only needs to store a mini-batch at a time.
- **Efficient Computation:** Allows for efficient use of hardware optimizations and parallel processing, making it suitable for large datasets.

Disadvantages

- **Complexity in Tuning:** Requires careful tuning of the mini-batch size and learning rate to ensure optimal performance.
- **Less Stable than Batch GD:** While more stable than SGD, it can still be less stable than Batch Gradient Descent, especially if the mini-batch size is too small.
- **Potential for Suboptimal Mini-Batch Sizes:** Selecting an inappropriate mini-batch size can lead to suboptimal performance and convergence issues.

Momentum-Based Gradient Descent

Momentum-Based Gradient Descent is an enhancement of standard gradient descent algorithm that aims to accelerate convergence particularly in the presence of high curvature, small but consistent gradients or noisy gradients. It introduces a velocity term that accumulates the gradient of the loss function over time thereby smoothing the path taken by the parameters.



The update rule for Momentum-Based Gradient Descent is:

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta_t)$$

$$\theta_{t+1} = \theta_t - v_t$$

where:

- v_t is the velocity at iteration t,
- γ is the momentum term (typically between 0 and 1),
- $\eta$is the learning rate,
- $\nabla J(\theta_t)$ is the gradient of the loss function $J(\theta)$ with respect to θ at iteration t.

Advantages

- **Accelerated Convergence:** Helps in faster convergence especially in scenarios with small but consistent gradients.
- **Smoother Updates:** Reduces the oscillations in the gradient updates, leading to a smoother and more stable convergence path.
- **Effective in Ravines:** Particularly effective in dealing with ravines or regions of steep curvature and is common in deep learning loss landscapes.

Disadvantages

- **Additional Hyperparameter:** Introduces an additional hyperparameter (momentum term) that needs to be tuned.
- **Complex Implementation:** Slightly more complex to implement compared to standard gradient descent.
- **Potential Overcorrection:** If not properly tuned the momentum can lead to overcorrection and instability in the updates.

Cost function

A Cost Function is a mathematical function that measures the difference between the actual values and the predicted values of a machine learning model. It calculates the overall prediction error for the entire training dataset and provides a numerical value that indicates how well the model is performing.

Cost Function refers to the relationship between input costs and output. In simple terms, the functional relationship between cost and output is referred to as the cost function. It is written as:

C = f(q)

Where;

C = Cost of Production;

q = Quantity of Production; and

f = Functional Relationship

Mathematical Formula

For Linear Regression, the most commonly used cost function is the **Mean Squared Error (MSE) Cost Function**.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Where:

- **J(θ)** = Cost function
- **m** = Total number of training examples
- **hθ(x_i)** = Predicted value
- **y_i** = Actual value
- **θ** = Model parameters (weights)

Components of the Cost Function

1. Actual Value (y)

The actual value is the true output available in the training dataset.

2. Predicted Value (hθ(x))

The predicted value is the output generated by the machine learning model.

3. Prediction Error

The prediction error is the difference between the actual value and the predicted value.

$$\text{Error} = h_\theta(x) - y$$

4. Squared Error

The prediction error is squared so that both positive and negative errors become positive values.

$$(h_\theta(x) - y)^2$$

5. Average Error

The squared errors of all training examples are averaged to obtain the overall cost.

Minimizing the Cost Function for a Single Variable Function

Minimizing the Cost Function for a Single-Variable Function is the process of finding the optimal values of the intercept (**θ_0**) and slope (**θ_1**) that minimize the prediction error of a Linear Regression model with one independent variable.

Single-Variable Linear Regression Equation

The equation of Single-Variable Linear Regression is

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Where:

- $h_\theta(x)$= Predicted value
- x= Independent variable
- θ_0= Intercept (Bias)
- θ_1= Slope (Weight)

Cost Function

The cost function for Single-Variable Linear Regression is

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Where:

- $J(\theta_0, \theta_1)$= Cost function
- m= Number of training examples
- $h_\theta(x_i)$= Predicted value
- y_i= Actual value

The objective is to minimize the value of the cost function so that the prediction error becomes as small as possible.

Gradient Descent Update Rule

The model parameters are updated using the following equations:

Updating the Intercept

$$\theta_0 = \theta_0 - \alpha\frac{\partial J}{\partial \theta_0}$$

Updating the Slope

$$\theta_1 = \theta_1 - \alpha\frac{\partial J}{\partial \theta_1}$$

Where:

- θ_0= Intercept

- θ_1= Slope
- α= Learning rate
- $\frac{\partial J}{\partial \theta}$= Gradient of the cost function

Steps for Minimizing the Cost Function

Step 1: Initialize the Parameters

Initially, the values of the intercept (θ_0) and slope (θ_1) are assigned small values or zero.

Step 2: Calculate the Predicted Values

The predicted values are calculated using the Linear Regression equation:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Step 3: Compute the Cost Function

The difference between the actual values and the predicted values is calculated, and the overall prediction error is measured using the cost function.

Step 4: Calculate the Gradient

The partial derivatives of the cost function with respect to θ_0 and θ_1 are computed to determine the direction in which the parameters should be updated.

Step 5: Update the Parameters

The values of θ_0 and θ_1 are updated using the Gradient Descent equations to reduce the cost function.

Step 6: Repeat the Process

The prediction, cost calculation, gradient computation, and parameter update steps are repeated until the cost function reaches its minimum value or the model converges.

Minimizing the Cost Function for a Two-Variable Function

Minimizing the Cost Function for a Two-Variable Function is the process of finding the optimal values of the regression parameters (θ_0, θ_1, **and** θ_2) that minimize the overall prediction error in a Multiple Linear Regression model containing two independent variables.

Two-Variable Linear Regression Equation

The mathematical equation of a two-variable Linear Regression model is

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2}$$

Where:

- $h_\theta(x)$= Predicted value

- x_1= First independent variable
- x_2= Second independent variable
- θ_0= Intercept (Bias)
- θ_1= Regression coefficient of the first variable
- θ_2= Regression coefficient of the second variable

Algorithm for Minimizing the Cost Function

Step 1: Initialize the Model Parameters

Initially, the values of **θ_0, θ_1, and θ_2** are assigned small random values or zeros.

Step 2: Compute the Predicted Values

The predicted value for each training example is calculated using the regression equation.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Step 3: Calculate the Prediction Error

The difference between the actual value and the predicted value is computed.

$$\text{Error} = h_\theta(x) - y$$

Step 4: Compute the Cost Function

The prediction errors are squared, summed for all training examples, and averaged to calculate the overall cost.

$$J(\theta_0, \theta_1, \theta_2) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Step 5: Compute the Gradients

The partial derivatives of the cost function with respect to **θ_0, θ_1, and θ_2** are calculated. These gradients indicate the direction in which each parameter should be updated to reduce the prediction error.

Step 6: Update the Parameters

The values of **θ_0, θ_1, and θ_2** are updated simultaneously using the Gradient Descent equations. Each update moves the parameters in the direction opposite to the gradient, thereby reducing the cost function.

Step 7: Repeat the Process

The prediction, cost calculation, gradient computation, and parameter update steps are repeated until the cost function reaches its minimum value or the algorithm converges.

Graphical Representation

- The cost function for a two-variable regression model forms a **three-dimensional convex surface**.
- The vertical axis represents the value of the cost function, while the horizontal axes represent the regression coefficients.
- The lowest point on the surface is called the **Global Minimum**.
- Gradient Descent moves toward the global minimum by updating the regression parameters in small steps.
- At the global minimum, the regression model produces the minimum prediction error and the highest prediction accuracy.

Evaluation Metrics in Regression

Introduction

Evaluation metrics are statistical measures used to evaluate the performance of a regression model. They compare the predicted values generated by the model with the actual values in the dataset. These metrics help determine how accurately a model predicts continuous values and assist in comparing different regression models. A good evaluation metric provides meaningful information about the prediction errors and the overall performance of the model.

1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is the average of the absolute differences between the actual values and the predicted values. It measures the average magnitude of prediction errors without considering their direction.

Formula

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$

Where:

- y_i= Actual value
- $\hat{y}_i$= Predicted value
- n= Number of observations

Interpretation

- MAE measures the average prediction error.
- A lower MAE indicates better model performance.

- MAE is expressed in the same unit as the target variable.

2. Mean Squared Error (MSE)

Mean Squared Error (MSE) is the average of the squared differences between the actual values and the predicted values. Squaring the errors gives greater importance to larger prediction errors.

Formula

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

Interpretation

- MSE measures the average squared prediction error.
- A lower MSE indicates higher prediction accuracy.
- Larger errors have a greater impact because they are squared.

3. Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is the square root of the Mean Squared Error. It measures the average prediction error in the same unit as the target variable.

Formula

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

Interpretation

- RMSE indicates the average prediction error.
- A lower RMSE represents a better regression model.
- RMSE is expressed in the original unit of the output variable.

4. Root Mean Squared Log Error (RMSLE)

Root Mean Squared Log Error (RMSLE) measures the difference between the logarithms of the actual and predicted values. It is useful when the target values have a large range or exponential growth.

Formula

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$

Interpretation

- RMSLE measures the relative prediction error.
- A lower RMSLE indicates better prediction performance.
- It reduces the influence of very large values.

5. R Squared (R^2)

R Squared (R^2), also known as the Coefficient of Determination, measures the proportion of variation in the dependent variable that is explained by the regression model. Its value ranges from **0 to 1**, where values closer to **1** indicate a better fit.

Formula

$$\boxed{R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}}$$

Where:

- $\bar{y}$= Mean of the actual values

Interpretation

- **$R^2 = 1$** indicates a perfect fit.
- **$R^2 = 0$** indicates that the model explains none of the variation.
- A higher R^2 value indicates better model performance.

6. Adjusted R Squared

Adjusted R Squared is a modified version of R^2 that considers the number of independent variables used in the regression model. It provides a more accurate measure of model performance, especially in Multiple Linear Regression.

Formula

$$\boxed{\text{Adjusted } R^2 = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1}\right)}$$

Where:

- n= Number of observations
- p= Number of independent variables

Interpretation

- Adjusted R^2 increases only when new variables improve the model significantly.
- It decreases if unnecessary variables are added.

- A higher Adjusted R^2 indicates a better regression model.

Case Study

Evaluation Metrics for House Price Prediction

Problem Statement

A real estate company developed a **Multiple Linear Regression** model to predict house prices based on features such as house area, number of bedrooms, and house age. To evaluate the performance of the model, the company uses **Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Root Mean Squared Log Error (RMSLE), R-Squared (R^2), and Adjusted R-Squared**.

The actual and predicted house prices are given below.

House	Actual Price (Y) (₹ Lakhs)	Predicted Price ($\hat{Y}$) (₹ Lakhs)
H1	50	48
H2	60	63
H3	70	68
H4	80	82
H5	90	89

Step 1: Calculate Errors

House	Actual (Y)	Predicted ($\hat{Y}$)	Error ($Y-\hat{Y}$)	|Error|	Error²
H1	50	48	2	2	4
H2	60	63	-3	3	9
H3	70	68	2	2	4
H4	80	82	-2	2	4
H5	90	89	1	1	1
Total				**10**	**22**

Number of observations,

$$n = 5$$

1. Mean Absolute Error (MAE)

Formula

$$MAE = \frac{\sum | Y - \hat{Y} |}{n}$$

Calculation

$$MAE = \frac{10}{5} = 2$$

MAE = 2 Lakhs

2. Mean Squared Error (MSE)

Formula

$$\boxed{MSE = \frac{\sum(Y - \hat{Y})^2}{n}}$$

Calculation

$$MSE = \frac{22}{5} = 4.4$$

MSE = 4.4 Lakhs²

3. Root Mean Squared Error (RMSE)

Formula

$$\boxed{RMSE = \sqrt{MSE}}$$

Calculation

$$RMSE = \sqrt{4.4} = 2.10$$

RMSE = 2.10 Lakhs

4. Root Mean Squared Log Error (RMSLE)

Formula

$$\boxed{RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\log(Y_i + 1) - \log(\hat{Y}_i + 1)\right]^2}}$$

Calculation

Using the given data,

$$RMSLE \approx 0.034$$

RMSLE = 0.034

5. R-Squared (R²)

Formula

$$\boxed{R^2 = 1 - \frac{\sum(Y - \hat{Y})^2}{\sum(Y - \bar{Y})^2}}$$

Where,

$$\bar{Y} = \frac{50 + 60 + 70 + 80 + 90}{5} = 70$$

$$SS_{Total} = (50 - 70)^2 + (60 - 70)^2 + (70 - 70)^2 + (80 - 70)^2 + (90 - 70)^2$$

$$= 400 + 100 + 0 + 100 + 400 = 1000$$

$$SS_{Residual} = 22$$

Calculation

$$R^2 = 1 - \frac{22}{1000}$$

$$R^2 = 0.978$$

R^2 = 0.978 (97.8%)

6. Adjusted R-Squared

Assume the model contains **2 independent variables**.

Formula

$$\boxed{Adjusted\ R^2 = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1}\right)}$$

Where,

- $n = 5$
- $p = 2$
- $R^2 = 0.978$

Calculation

$$Adjusted\ R^2 = 1 - \left(\frac{(1 - 0.978)(5 - 1)}{5 - 2 - 1}\right)$$

$$= 1 - \left(\frac{0.022 \times 4}{2}\right)$$

$$= 1 - 0.044$$

$$Adjusted\ R^2 = 0.956$$

Adjusted R^2 = 0.956 (95.6%)

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, “Machine Learning”, Wiley Publisher,2019.
2. SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, “Machine Learning”,Pearson,2019.
3. Alpaydin Ethem, “Introduction to Machine Learning”, 3rd Edition, PHI learningprivate limited, 2019.

REFERENCE BOOKS:

1. “Machine Learning: A Probabilistic Perspective”, 2012, Kevin P. Murphy, MIT Press.
2. “Pattern Recognition and Machine Learning”, 2007, Christopher Bishop, Springer.
3. “Introduction to Machine Learning”, MIT Press, 3/e, 2014, Ethem Alpaydin.

Question Bank:

PART –A		
Q.No.	**Questions**	**Blooms Taxonomy Level**
1.	Define Supervised Learning.	L1
2.	What is Regression in Machine Learning?	L1
3.	Define Linear Regression.	L1
4.	State the equation of Simple Linear Regression.	L1
5.	What is a Model Estimator?	L2
6.	Define Regularization.	L1
7.	What is Ridge Regression?	L1
8.	What is LASSO Regression?	L1
9.	Define Multiple Linear Regression.	L1
10.	What is Gradient Descent?	L2
11.	Define Cost Function.	L2
12.	What is meant by minimizing the Cost Function?	L2
13.	Define Mean Absolute Error (MAE).	L1
14.	Define Mean Squared Error (MSE).	L1
15	What is Root Mean Squared Error (RMSE)?	L1
16	Define Root Mean Squared Log Error (RMSLE).	L1
17	What is R-Squared (R^2)?	L2
18	Define Adjusted R-Squared.	L2
19	List any two evaluation metrics used for regression models.	L1

20	Mention two applications of Linear Regression.	L2
	PART –B	
1.	Explain the concept of Supervised Learning and Regression with suitable examples.	L2
2.	Explain Linear Regression with its mathematical model, assumptions, advantages, and limitations.	L2
3.	Describe the evaluation of model estimators and explain the characteristics of a good estimator.	L2
4.	Explain Regularization and compare Ridge Regression and LASSO Regression with suitable examples.	L4
5.	Explain Multiple Linear Regression with mathematical formulation and real-world applications.	L3
6.	Explain Gradient-Based Methods and describe the Gradient Descent algorithm with suitable examples.	L3
7.	Explain the Cost Function in Linear Regression and describe the process of minimizing the Cost Function for a single-variable function.	L3
8.	Explain the minimization of the Cost Function for a two-variable function using Gradient Descent.	L3
9.	Explain the regression evaluation metrics MAE, MSE, RMSE, RMSLE, R^2, and Adjusted R^2 with suitable examples.	L2
10.	Compare MAE, MSE, RMSE, RMSLE, R^2, and Adjusted R^2 and discuss their significance in evaluating regression models.	L4
11.	Analyze the advantages and limitations of Ridge Regression and LASSO Regression in handling overfitting.	L4
12.	Evaluate the performance of different regression models using appropriate evaluation metrics.	L5
13.	Compare Simple Linear Regression and Multiple Linear Regression with suitable examples.	L4
14.	Discuss the importance of Cost Functions and Gradient-Based Optimization techniques in Machine Learning.	L3
15.	Evaluate the effectiveness of different regression evaluation metrics for various real-world applications.	L5

UNIT IV

Classification

"The field of Machine Learning is concerned with the question of how to construct computer programs that automatically improve with experience."

— Tom M. Mitchell

Unit Overview

Classification is one of the most important supervised learning techniques in Machine Learning. It is used to predict categorical class labels based on input data. This unit introduces fundamental classification algorithms such as Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, K-Nearest Neighbor (KNN), Neural Networks (Perceptron and Multilayer Perceptron), and Support Vector Machines (SVMs). It also covers model evaluation metrics such as Accuracy, Confusion Matrix, Precision, Recall, and F1-Score, which help assess the performance of classification models. These techniques are widely applied in spam detection, disease diagnosis, fraud detection, sentiment analysis, image recognition, and many other real-world applications.

Objectives of the Unit

After studying this unit, students will be able to:

- Understand the concept of classification and its significance in Machine Learning.
- Explain the working of Logistic Regression using the Logit Function and Maximum Likelihood Estimation.
- Construct and interpret Decision Trees for classification problems.
- Understand the principles of Ensemble Learning and Random Forest.
- Explain Bayesian Classification using the Naïve Bayes algorithm.
- Understand the working of the K-Nearest Neighbor (KNN) classifier.
- Describe the architecture and learning process of Perceptron and Multilayer Perceptron (MLP).
- Explain the concepts of Linear Support Vector Machines, Optimal Hyperplane, and Radial Basis Function (RBF).
- Evaluate classification models using performance metrics such as Accuracy, Confusion Matrix, Precision, Recall, and F1-Score.

Learning Outcomes

At the end of this unit, students will be able to:

- Explain the fundamentals of classification in supervised learning.
- Build Logistic Regression models for binary classification.

- Construct and analyze Decision Tree models.
- Apply Random Forest for improving classification accuracy.
- Implement Naïve Bayes and KNN algorithms for classification tasks.
- Understand the working principles of Artificial Neural Networks.
- Apply Support Vector Machines to solve classification problems.
- Interpret evaluation metrics to measure classifier performance.
- Compare various classification algorithms based on accuracy, complexity, and applications.
- Develop suitable classification models for real-world datasets.

Importance of Studying this Unit

Studying classification techniques is essential because they enable computers to make intelligent decisions by assigning data to predefined categories. Classification algorithms are among the most widely used Machine Learning methods in industry and research.

This unit helps students:

- Understand how intelligent systems perform automated decision-making.
- Learn algorithms used in healthcare, banking, cybersecurity, marketing, and social media.
- Develop predictive models for real-world classification problems.
- Improve analytical and problem-solving skills using Machine Learning techniques.
- Gain practical knowledge required for Artificial Intelligence and Data Science applications.
- Build a strong foundation for advanced Machine Learning and Deep Learning concepts.
- Understand how different classifiers behave with different types of datasets.
- Evaluate and improve model performance using standard evaluation metrics.

Key Concepts:

- Classification is a supervised machine learning technique used to predict categorical class labels from input data.
- Supervised learning is a machine learning approach in which models are trained using labeled datasets.
- Logistic Regression is a classification algorithm used to predict the probability of categorical outcomes.
- A Decision Tree is a tree-based model that classifies data using a sequence of decision rules.
- Information Gain is a measure used to select the best attribute for splitting data in a decision tree.

- Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve prediction accuracy.
- The Naïve Bayes Classifier is a probabilistic algorithm based on Bayes' Theorem that assumes feature independence.
- K-Nearest Neighbor (KNN) is a classification algorithm that assigns a class based on the majority class of its nearest neighbors.
- An Artificial Neuron is the basic computational unit of a neural network that processes input signals.
- A Multilayer Perceptron (MLP) is a neural network with multiple hidden layers used to solve complex classification problems.

- A Support Vector Machine (SVM) is a supervised learning algorithm that separates classes using an optimal decision boundary.
- The Optimal Hyperplane is the decision boundary that maximizes the margin between different classes in an SVM.
- A Confusion Matrix is a table that compares actual and predicted class labels to evaluate classifier performance.
- Accuracy is the percentage of correctly classified instances among all predictions.
- Precision, Recall, and F1-Score are performance metrics used to evaluate the effectiveness of a classification model, especially for imbalanced datasets.
- Precision, Recall, and F1-Score are performance metrics used to evaluate the effectiveness of a classification model, especially for imbalanced datasets

Introduction to Classification:

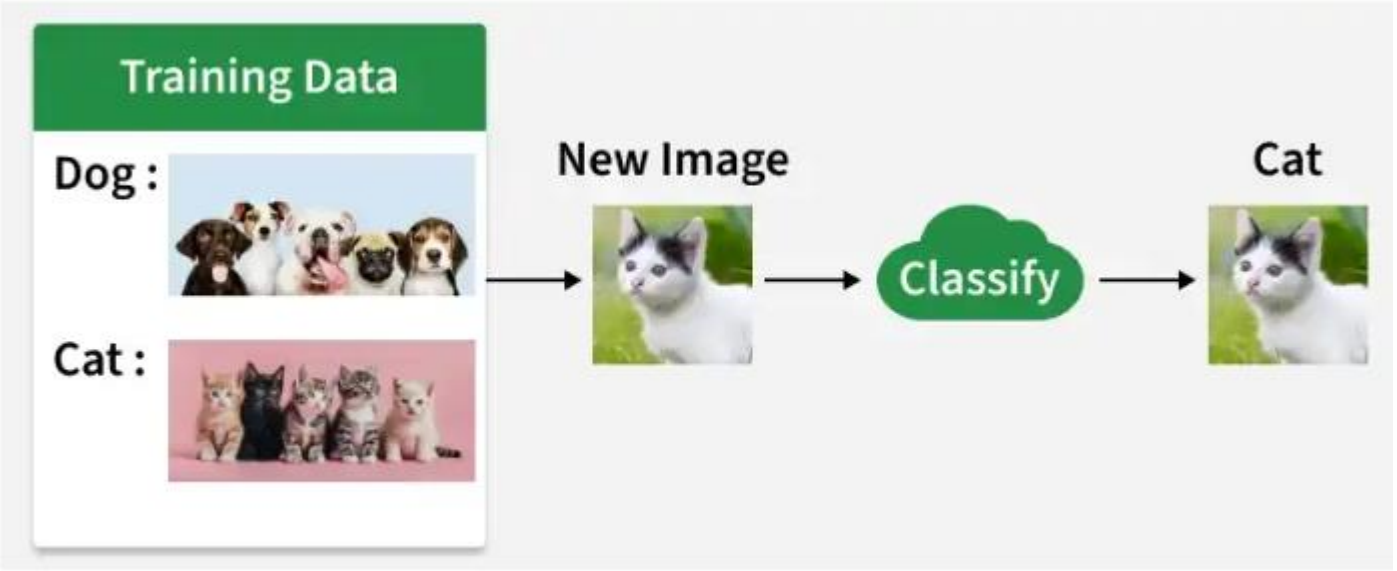
Classification is a supervised machine learning technique used to predict labels or categories from input data. It assigns each data point to a predefined class based on learned patterns.

- **Predict categories**: Determines the class of new data points.
- **Uses labeled data:** Trained on datasets where the correct class is known.
- **Common examples**: Spam vs non spam emails, diseased vs. healthy patients.

Classification is a supervised machine learning technique used to assign input data to one of the predefined class labels based on patterns learned from labeled training data. The output of a classification model is always

categorical, such as Yes/No, Spam/Not Spam, or Pass/Fail.

For example: A classification model might be trained on dataset of images labeled as either dogs or cats and it can be used to predict the class of new and unseen images as dogs or cats based on their features such as colour, texture or shape.



Cat-Dog Classification

Types of Classification:

Classification in machine learning involves sorting data into categories based on their features or characteristics. The type of classification problem depends on how many classes exist and how the categories are structured.

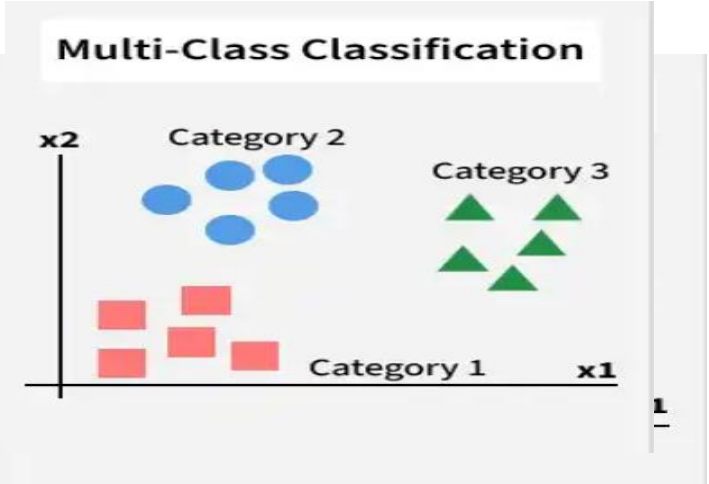
1. Binary Classification

Binary classification is the simplest type of classification where data is divided into two possible categories. The model analyzes input features and decides which of the two classes the data belongs to. Some important aspects of binary classification are:

- **Two classes only:** Each data point is assigned to one of two categories.
- **Common examples:** Spam vs not spam emails, diseased vs healthy patients.
- **Decision based on features:** The model uses input features to determine the correct class.

2. Multiclass Classification

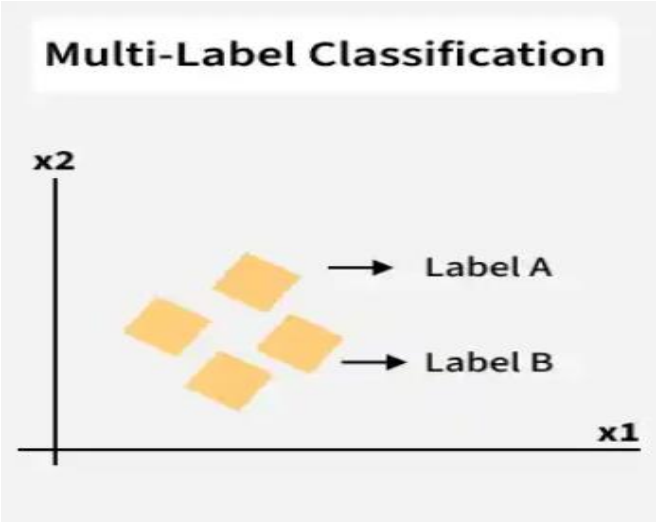
Multiclass classification is used when data needs to be divided into more than two categories. The model analyzes the input features and selects the class that best matches the data. Some important aspects of multiclass classification are:



- **Multiple classes:** Each data point is assigned to one of several possible categories.
- **Single final prediction:** The model selects only one class for each input.
- **Common examples:** Image classification such as identifying animals like cat, dog or bird.

3. Multi Label Classification

Multi label classification allows a single piece of data to belong to multiple categories at the same time. Unlike multiclass classification, where each data point is assigned only one class, this approach allows the model to assign multiple labels to the same input. Key aspects include:



- **Multiple labels per data point:** One input can belong to more than one category.

- **Labels can overlap:** Classes are not mutually exclusive.
- **Common example:** A movie recommendation system may tag a movie as both action and comedy based on features like plot, actors or genre tags.

Logistic Regression

Introduction

Logistic Regression is a supervised machine learning algorithm used for classification problems. It predicts the probability that an input belongs to a particular class. Unlike Linear Regression, which predicts continuous values, Logistic Regression predicts categorical outcomes such as Yes/No, True/False, Pass/Fail, or Spam/Not Spam. It uses the Sigmoid (Logistic) Function to convert the output into a probability value between 0 and 1.

Definition

Logistic Regression is a supervised machine learning algorithm used to classify data into one or more categories by estimating the probability of an input belonging to a particular class using the Sigmoid (Logistic) Function.

Types

1. **Binomial Logistic Regression:** Used when the dependent variable has only two possible categories, such as Yes/No, Pass/Fail, or 0/1. It is the most common type and is used for binary classification tasks.
2. **Multinomial Logistic Regression:** Used when the dependent variable has three or more unordered categories. For example, classifying animals as cat, dog, or sheep. It extends logistic regression to handle multiple classes.
3. **Ordinal Logistic Regression:** Used when the dependent variable has three or more categories with a natural order, such as Low, Medium, and High. It considers the ranking of categories during prediction.

Assumptions

1. **Independent Observations:** Each data point should be independent of the others, meaning there should be no dependence between observations.
2. **Binary Dependent Variable:** The target variable is typically binary and can take only two values, such as 0/1 or Yes/No. For multiclass problems, extensions such as Softmax-based logistic regression are used.
3. **Linearity of Log-Odds:** The independent variables should have a linear relationship with the log-odds of the dependent variable.
4. **No Extreme Outliers:** Extreme outliers can distort coefficient estimates and negatively affect model

performance.

5. **Large Sample Size:** A sufficiently large dataset helps produce more reliable and stable predictions.

Sigmoid Function:

1. The sigmoid function is a key component of Logistic Regression that converts the model's raw output into a probability value between 0 and 1.

2. This function takes any real number and maps it into the range 0 to 1 forming an "S" shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.

3. In logistic regression, we use a threshold value usually 0.5 to decide the class label.

- If the sigmoid output is same or above the threshold, the input is classified as Class 1.
- If it is below the threshold, the input is classified as Class 0.

This approach helps to transform continuous input values into meaningful class predictions.

Working

Logistic regression computes a linear combination of input features (z = w·X + b) and passes it through a sigmoid function to produce a probability between 0 and 1. This probability is then used to assign the input to a class.

Suppose we have input features represented as a matrix:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Yhaving only binary value i.e 0 or 1.

$$Y = \begin{cases} 0 & \text{if } Class1 \\ 1 & \text{if } Class2 \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = \left(\sum_{i=1}^{n} w_i\, x_i\right) + b$$

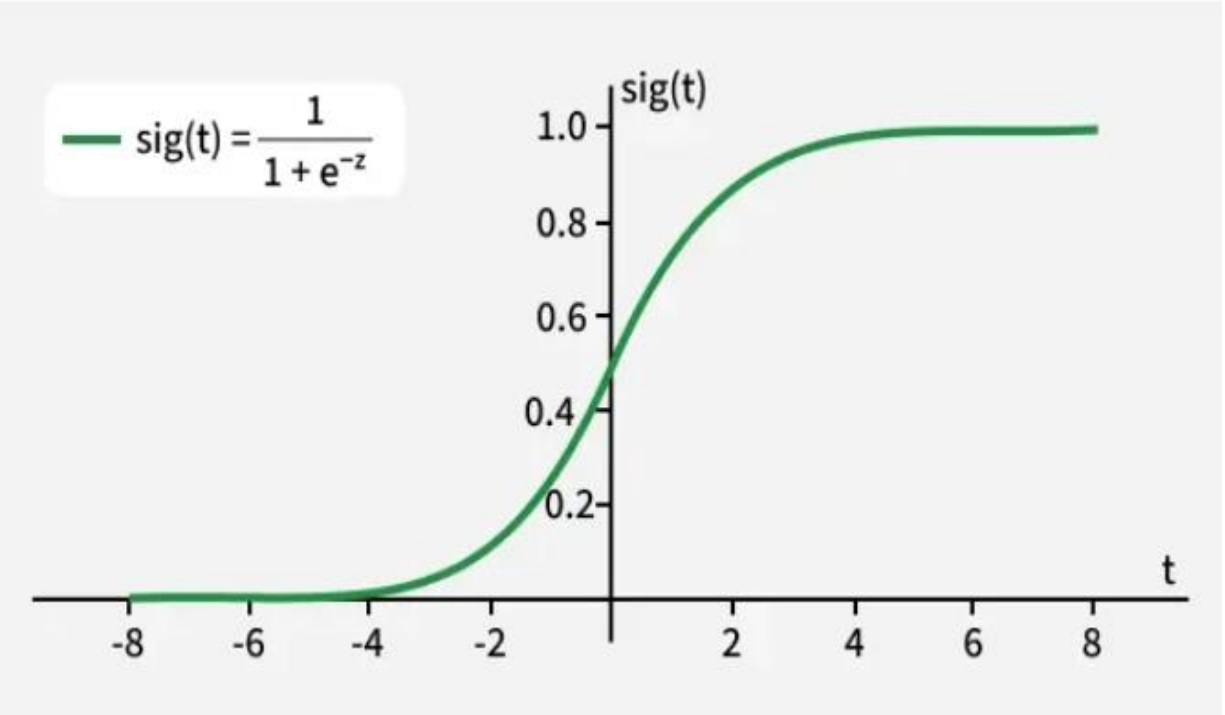
Here x_i is the ith observation of X, $w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient and $b$is the bias term also known as intercept. Simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

At this stage, $z$is a continuous value from the linear regression. Logistic regression then applies the sigmoid function to $z$to convert it into a probability between 0 and 1 which can be used to predict the class.

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Sigmoid function

As shown above the sigmoid function converts the continuous variable data into the probability i.e between 0 and 1.

- $\sigma(z)$ tends towards 1 as $z \to \infty$
- $\sigma(z)$ tends towards 0 as $z \to -\infty$
- $\sigma(z)$ is always bounded between 0 and 1

where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z) P(y = 0) = 1 - \sigma(z)$$

Logistic Regression Model (Logit Function)

The **Logit Function** is the natural logarithm of the odds of an event occurring. It transforms the probability of an event into a continuous value ranging from $-\infty$ **to** $+\infty$, enabling Logistic Regression to model binary classification problems effectively.

Logit Function Equation

The Logistic Regression Model is represented by the following equation:

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where:

- **P** = Probability that the event occurs.
- **1 − P** = Probability that the event does not occur.
- **P/(1 − P)** = Odds.
- **log(P/(1 − P))** = Log-Odds (Logit).
- **β_0** = Intercept (Bias).
- **$\beta_1, \beta_2, \ldots, \beta_n$** = Regression coefficients.
- **$X_1, X_2, \ldots, X_n$** = Input features.

Sigmoid Function

The Logit value is converted into a probability using the **Sigmoid Function**.

$$P = \frac{1}{1 + e^{-z}}$$

where,

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

The Sigmoid Function converts any real-valued number into a probability between **0 and 1**.

Logistic Regression Equation and Odds

Logistic Regression Equation

The Logistic Regression model first calculates a linear combination of the input features.

Linear Equation

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Where:

- **z** = Linear combination of the input features.
- **$x_1, x_2, \ldots, x_n$** = Input features.
- **$w_1, w_2, \ldots, w_n$** = Weights assigned to the features.
- **b** = Bias (Intercept).

Since the output of the linear equation can be any real number, it is converted into a probability using the **Sigmoid (Logistic) Function**.

Logistic Regression Equation (Sigmoid Function)

$$P(Y = 1) = \frac{1}{1 + e^{-z}}$$

or

$$P(Y=1) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b)}}$$

Where:

- **P(Y = 1)** = Probability that the input belongs to the positive class.
- **e** = Euler's constant (approximately 2.718).
- **z** = Linear combination of the input features.

The Sigmoid function converts any real-valued number into a probability between **0 and 1**.

Concept of Odds

Odds represent the ratio of the probability that an event occurs to the probability that it does not occur.

Odds Formula

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

Where:

- **P** = Probability that the event occurs.
- **1 − P** = Probability that the event does not occur.

Example of Odds

Suppose the probability that a customer will repay a loan is **0.80**.

Then,

$$\text{Odds} = \frac{0.80}{1 - 0.80} = \frac{0.80}{0.20} = 4$$

Interpretation:

The odds are **4 : 1**, which means the customer is **four times more likely** to repay the loan than to default.

Log-Odds (Logit Function)

Logistic Regression uses the **logarithm of the odds**, called the **Logit Function**.

Log-Odds Formula

$$\log\left(\frac{P}{1 - P}\right) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Where:

- $\frac{P}{1-P}$= Odds.
- $\log\left(\frac{P}{1-P}\right)$= Log-Odds (Logit).
- $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$= Linear equation.

The logit function converts probabilities into values ranging from $-\infty$ **to** $+\infty$, making it easier to model the relationship between input features and the target variable

Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a statistical technique used to estimate the values of the model parameters by maximizing the likelihood function, which represents the probability of observing the given training data under a particular set of parameter values.

Likelihood Function:

The **Likelihood Function** measures how likely the observed data is for a given set of model parameters.

For Logistic Regression, the likelihood function is:

$$L(\theta) = \prod_{i=1}^{n} P(y_i \mid x_i; \theta)$$

Where:

- **L(θ)** = Likelihood function.
- **θ** = Model parameters (weights and bias).
- $\mathbf{x_i}$ = Input feature vector.
- $\mathbf{y_i}$ = Observed class label.
- **n** = Number of training samples.

The objective of MLE is to find the parameter values that maximize **L(θ)**.

Log-Likelihood Function:

Instead of maximizing the likelihood directly, the **log-likelihood function** is used because it simplifies mathematical calculations.

$$\log L(\theta) = \sum_{i=1}^{n} \log P(y_i \mid x_i; \theta)$$

The logarithm converts multiplication into addition, making optimization easier and computationally efficient.

This is known as the log-likelihood function.

Maximum Likelihood Estimation (MLE) in Logistic Regression

The objective of Maximum Likelihood Estimation (MLE) is to find the values of the **weights (w)** and **bias (b)** that maximize the likelihood of observing the given training data.

Step 1: Probability of Prediction

For each training sample:

- If the actual class is $y = 1$,

$$P(Y = 1 \mid X) = p(x)$$

- If the actual class is $y = 0$,

$$P(Y = 0 \mid X) = 1 - p(x)$$

where,

$$p(x) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

Step 2: Likelihood Function

The likelihood function for all **n** training samples is

$$L(b, w) = \prod_{i=1}^{n} [p(x_i)]^{y_i} [1 - p(x_i)]^{1 - y_i}$$

where

- $y_i = 1$for the positive class.
- $y_i = 0$for the negative class.
- $p(x_i)$is the predicted probability.

Step 3: Log-Likelihood Function

Taking the natural logarithm on both sides,

$$\log L(b, w) = \sum_{i=1}^{n} [y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))]$$

The logarithm converts multiplication into addition, making the calculations simpler.

Step 4: Substitute the Sigmoid Function

Since

$$p(x_i) = \frac{1}{1 + e^{-(w \cdot x_i + b)}}$$

the log-likelihood becomes

$$\log L(b, w) = \sum_{i=1}^{n} [y_i (w \cdot x_i + b) - \log(1 + e^{w \cdot x_i + b})]$$

This is the **final log-likelihood equation** used in Logistic Regression.

Terminologies Used

1. **Independent Variables:** These are the input features or predictor variables used to make predictions about the dependent variable.
2. **Dependent Variable**: This is the target variable that we aim to predict. In logistic regression, the dependent variable is categorical.
3. **Logistic Function**: This function transforms the independent variables into a probability between 0 and 1 which represents the likelihood that the dependent variable is either 0 or 1.
4. **Odds**: This is the ratio of the probability of an event happening to the probability of it not happening. It differs from probability because probability is the ratio of occurrences to total possibilities.
5. **Log-Odds (Logit)**: The natural logarithm of the odds. In logistic regression, the log-odds are modeled as a linear combination of the independent variables and the intercept.
6. **Coefficient**: These are the parameters estimated by the logistic regression model which shows how strongly the independent variables affect the dependent variable.
7. **Intercept**: The constant term in the logistic regression model which represents the log-odds when all independent variables are equal to zero.
8. **Maximum Likelihood Estimation (MLE)**: This method is used to estimate the coefficients of the logistic regression model by maximizing the likelihood of observing the given data.

Decision Tree

Introduction

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It represents decisions and their possible outcomes in the form of a tree-like structure. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents the final class label or prediction. Decision Trees are easy to understand, interpret, and implement.

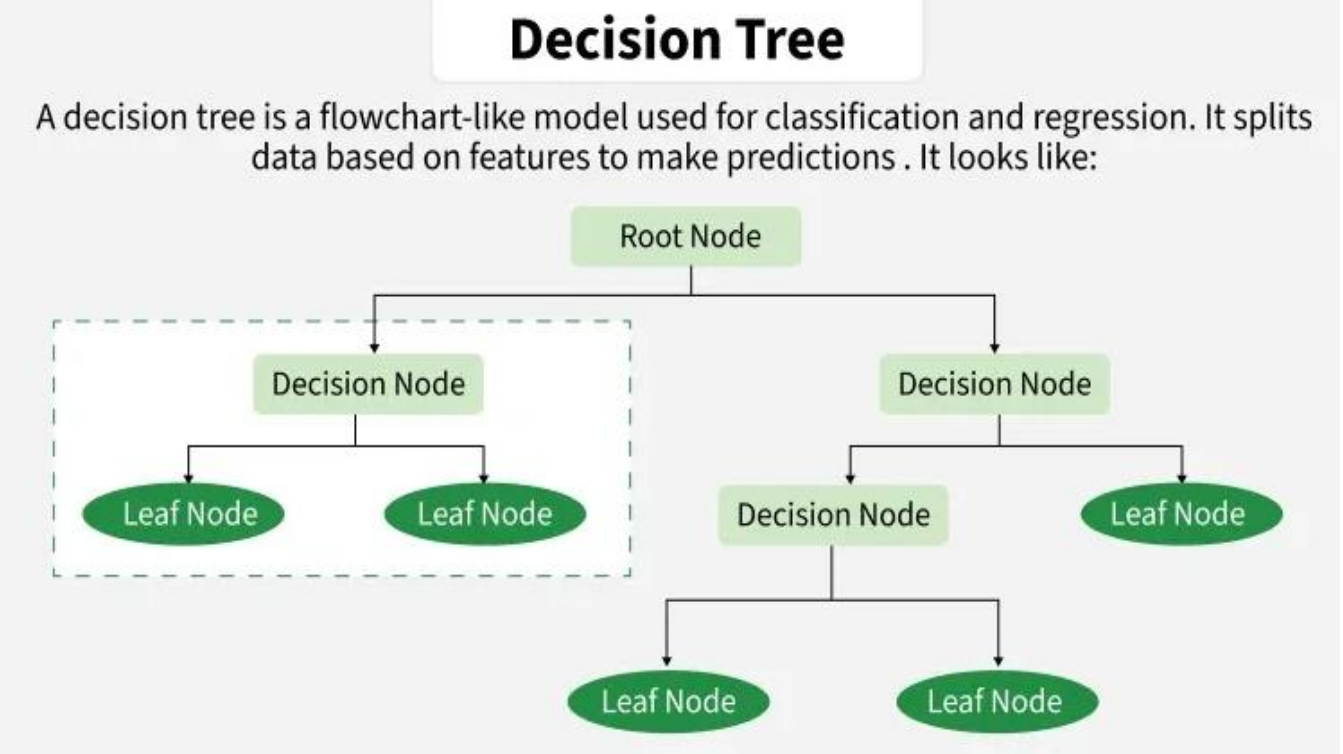
Definition

A Decision Tree is a tree-structured machine learning model that classifies data by recursively splitting the dataset based on the values of input features until a final decision or prediction is reached.

(or)

A decision tree is a supervised learning algorithm used for both classification and regression tasks. It has a hierarchical tree structure which consists of a root node, branches, internal nodes and leaf nodes. It works like a flowchart that helps in making step-by-step decisions, where:

- Internal nodes represent attribute tests
- Branches represent attribute values
- Leaf nodes represent final decisions or predictions.



Components of a Decision Tree

- **Root Node:** The topmost node that represents the entire dataset and performs the first split.
- **Internal Node or Decision node:** Represents a test or decision based on an attribute.
- **Branch:** Represents the outcome of a decision or test.
- **Leaf Node:** Represents the final class label or prediction.

Types of Decision Trees

Decision Trees are classified into **Classification Trees** and **Regression Trees** based on the type of output they predict.

1. Classification Tree

A Classification Tree is a type of Decision Tree used when the target variable is categorical. It classifies data into predefined classes such as Yes/No, True/False, Pass/Fail, or Spam/Not Spam. The objective of a Classification Tree is to assign the correct class label to a new data instance.

Characteristics

- It is used for **classification problems**.
- The output is a **categorical or discrete class label**.
- It divides the dataset into different classes based on the values of input features.
- It uses splitting criteria such as **Information Gain, Gain Ratio,** or **Gini Index**.
- Each leaf node represents the final predicted class.

Example

Suppose a bank wants to decide whether a customer is eligible for a loan.

Input Features:

- Credit Score
- Monthly Income
- Employment Status

Output:

- **Yes** → Loan Approved
- **No** → Loan Rejected

The Classification Tree predicts whether the loan should be **approved** or **rejected**, making it a classification problem.

2. Regression Tree

A Regression Tree is a type of Decision Tree used when the target variable is continuous or numerical. Instead of predicting a class label, it predicts a numerical value. Regression Trees are mainly used for estimating quantities such as prices, salaries, sales, and temperatures.

Characteristics

- It is used for **regression problems**.
- The output is a **continuous numerical value**.
- It divides the dataset into smaller subsets to minimize prediction error.
- It commonly uses **Mean Squared Error (MSE)** or **Variance Reduction** as the splitting criterion.
- Each leaf node contains a predicted numerical value.

Example

Suppose a real estate company wants to predict the selling price of a house.

Input Features:

- Area of the House
- Number of Bedrooms
- Location
- Age of the House

Output:

- ₹45,00,000
- ₹62,50,000
- ₹80,00,000

The Regression Tree predicts the **selling price of the house**, which is a continuous numerical value.

Classification Using Decision Trees

Introduction

Classification using Decision Trees is a supervised machine learning technique used to classify data into predefined classes. A Decision Tree classifies an input instance by following a sequence of decision rules from the root node to a leaf node. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents the final class label. Decision Trees are widely used because they are simple, easy to understand, and provide accurate classification results.

Definition

Classification using a Decision Tree is the process of predicting the class label of an unknown instance by traversing the decision tree from the root node to a leaf node based on the values of its input features.

Working of Classification Using Decision Trees:

Step 1: Start at the Root Node: The classification process begins at the **root node**, which represents the most important attribute selected during the construction of the decision tree.

Step 2: Evaluate the Attribute: The value of the input feature is compared with the decision condition at the current node. The result determines which branch should be followed.

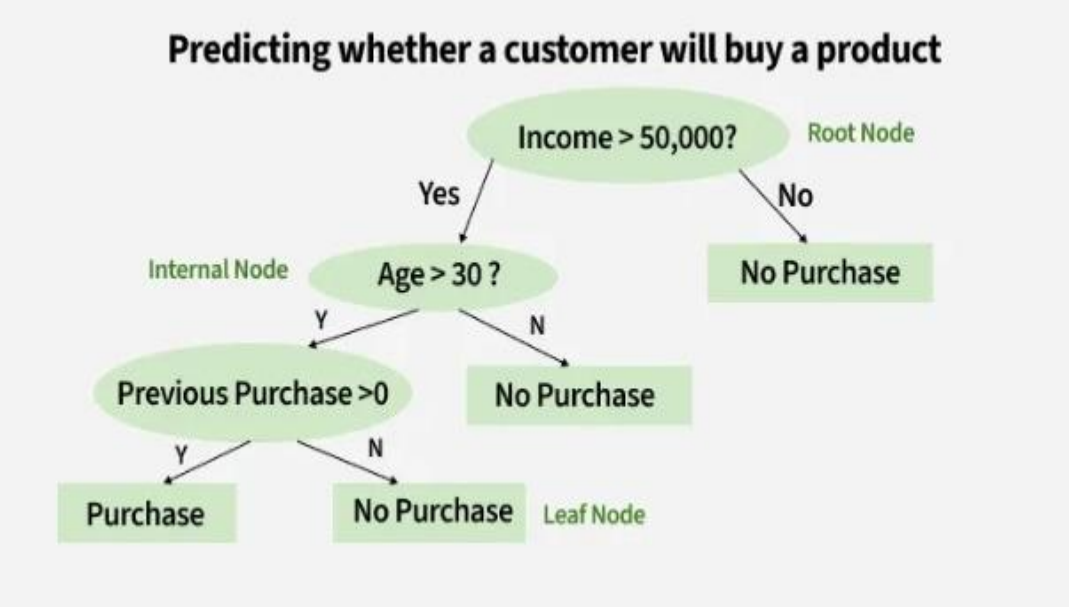
Step 3: Move to the Next Node: Based on the outcome of the test, the algorithm follows the appropriate branch to the next internal node.

Step 4: Repeat the Process: The process of evaluating attributes and selecting branches continues until a **leaf**

node is reached.

Step 5: Predict the Class Label: The class label stored in the leaf node is assigned as the predicted output for the input instance.

Example:



1. Root Node (Income)

First Question: **"Is the person's income greater than $50,000?"**

- If Yes, proceed to the next question.
- If No, predict "No Purchase" (leaf node).

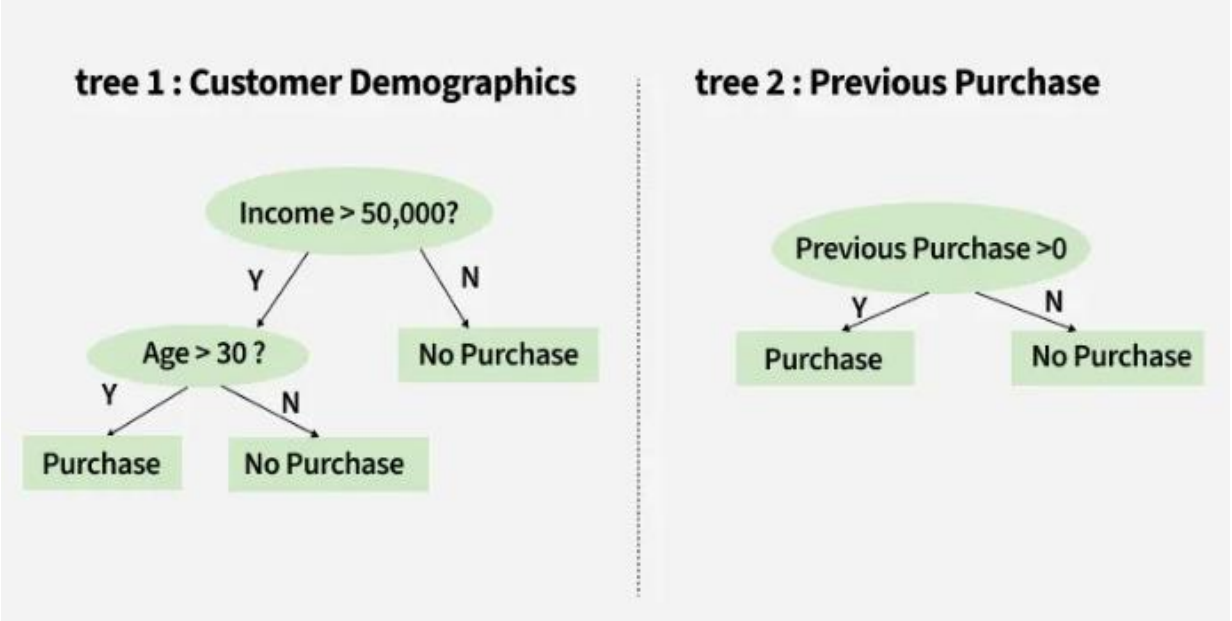
2. Internal Node (Age):

If the person's income is greater than $50,000, ask: **"Is the person's age above 30?"**

- If Yes, proceed to the next question.
- If No, predict "No Purchase" (leaf node).

3. Internal Node (Previous Purchases):

- If the person is above 30 and has made previous purchases, predict "Purchase" (leaf node).
- If the person is above 30 and has not made previous purchases, predict "No Purchase" (leaf node).



Example: A decision tree makes predictions using a single tree structure by following decision paths from root to leaf.

Tree 1: Customer Demographics

First tree asks two questions:

1. "Income > $50,000?"
   - If Yes, Proceed to the next question.
   - If No, "No Purchase"
2. "Age > 30?"
   - Yes: "Purchase"
   - No: "No Purchase"

Tree 2: Previous Purchases

"Previous Purchases > 0?"

- Yes: "Purchase"
- No: "No Purchase"

Once we have predictions from both trees, we can combine the results to make a final prediction. If Tree 1 predicts "Purchase" and Tree 2 predicts "No Purchase", the final prediction might be "Purchase" or "No Purchase" depending on the weight or confidence assigned to each tree. This can be decided based on the problem context.

Advantages of Classification Using Decision Trees

- Classification using Decision Trees is simple and easy to understand because the decision-making process is represented in a tree-like structure.
- It provides clear and interpretable decision rules, making it easier to explain the model's predictions.

- It can handle both numerical and categorical data, making it suitable for a wide variety of machine learning applications.
- It requires very little data preprocessing, as it can work with raw data without requiring feature scaling or normalization.
- It provides fast and accurate classification by following a sequence of decision rules from the root node to the leaf node.
- It automatically performs feature selection during tree construction by selecting the most important attributes for splitting the data.

Limitations of Classification Using Decision Trees

- Classification using Decision Trees is prone to overfitting when the tree becomes very large and learns the training data too closely.
- Small changes in the training data may produce a completely different decision tree, making the model unstable.
- The decision tree may become complex and difficult to interpret when it is constructed using large datasets.
- It may be biased toward attributes with many distinct values, which can affect the accuracy of the model.
- It may not perform well for highly complex relationships that cannot be represented by simple decision rules.

Applications of Classification Using Decision Trees

- Classification using Decision Trees is widely used in healthcare for diagnosing diseases based on a patient's symptoms and medical records.
- It is used in banking and finance to determine whether a customer is eligible for a loan or credit.
- It is used in email systems to classify emails as spam or non-spam.
- It is used in fraud detection to identify suspicious or fraudulent financial transactions.
- It is used in customer churn prediction to identify customers who are likely to stop using a company's products or services.
- It is used in education to predict student performance based on academic and personal factors.
- It is used in credit risk assessment to evaluate the risk associated with lending money to customers.

Issues in Decision Trees

Decision Trees are simple and effective machine learning models, but they have several limitations that

can affect their performance. The major issues in Decision Trees are explained below.

1. Overfitting

A Decision Tree may become too complex by creating many branches to fit the training data perfectly. This causes the model to memorize the training data rather than learn general patterns, resulting in poor performance on new or unseen data.

2. Underfitting

If the tree is too small or shallow, it may fail to capture the underlying patterns in the data. Such a model produces low accuracy on both training and testing datasets.

3. High Variance

Decision Trees are highly sensitive to changes in the training data. Even a small variation in the dataset may produce a completely different tree structure, making the model unstable.

4. Bias Toward Attributes with Many Values

Decision Trees often prefer attributes that have a large number of unique values because they provide higher information gain. This may lead to incorrect feature selection and reduced model accuracy.

5. Difficulty in Handling Continuous Data

Continuous numerical attributes must be converted into intervals by selecting suitable threshold values. Finding the optimal split increases computational complexity.

6. Computational Complexity

Building an optimal Decision Tree requires evaluating many possible splits at every node. For large datasets with many features, the training process becomes time-consuming and computationally expensive.

7. Class Imbalance Problem

When one class has significantly more instances than others, the Decision Tree tends to favor the majority class, resulting in poor classification of minority class instances.

8. Missing Values

Missing attribute values make it difficult to determine the correct splitting criteria. Additional preprocessing or imputation techniques are required before training the model.

9. Greedy Splitting Strategy

Decision Trees use a greedy approach by selecting the best split at each node without considering future splits. This may not produce the globally optimal tree.

10. Large and Complex Trees

For datasets with many features and instances, the generated tree may become very large and difficult to understand, reducing its interpretability.

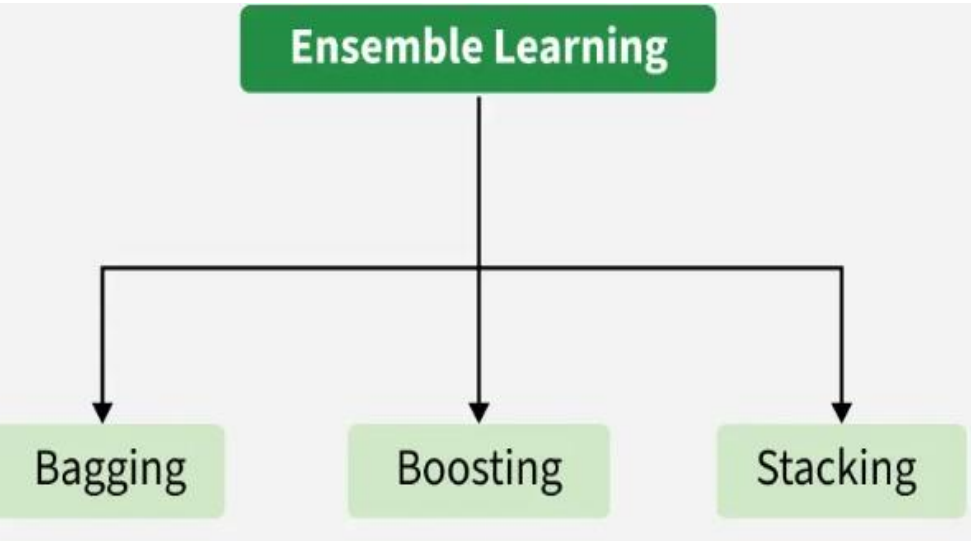
Ensemble Learning

Ensemble learning is a method where multiple models are combined instead of using just one. Even if individual models are weak, combining their results gives more accurate and reliable predictions.

- Uses multiple models together to improve overall accuracy.
- Reduces errors by balancing mistakes across models.
- Works on a simple idea similar to combining opinions from a group.

Types of Ensemble Learning

There are three main types of ensemble methods:

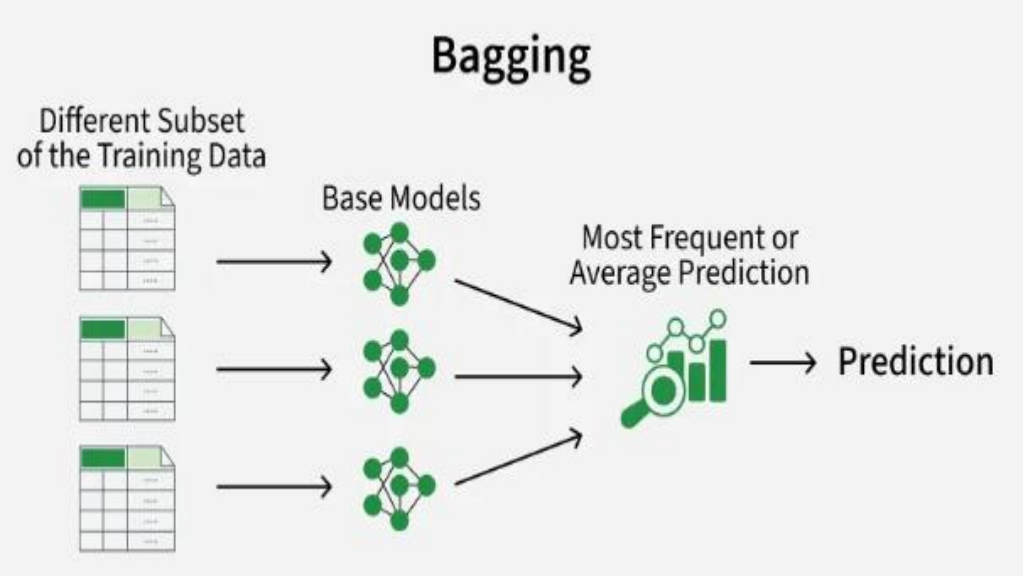


Bagging (Bootstrap Aggregating):

Models are trained independently on different random subsets of the training data. Their results are then combined—usually by averaging (for regression) or voting (for classification). This helps reduce variance and prevents overfitting.

Bagging Algorithm:

Bagging classifier can be used for both regression and classification tasks. Here is an overview of Bagging classifier algorithm**:**



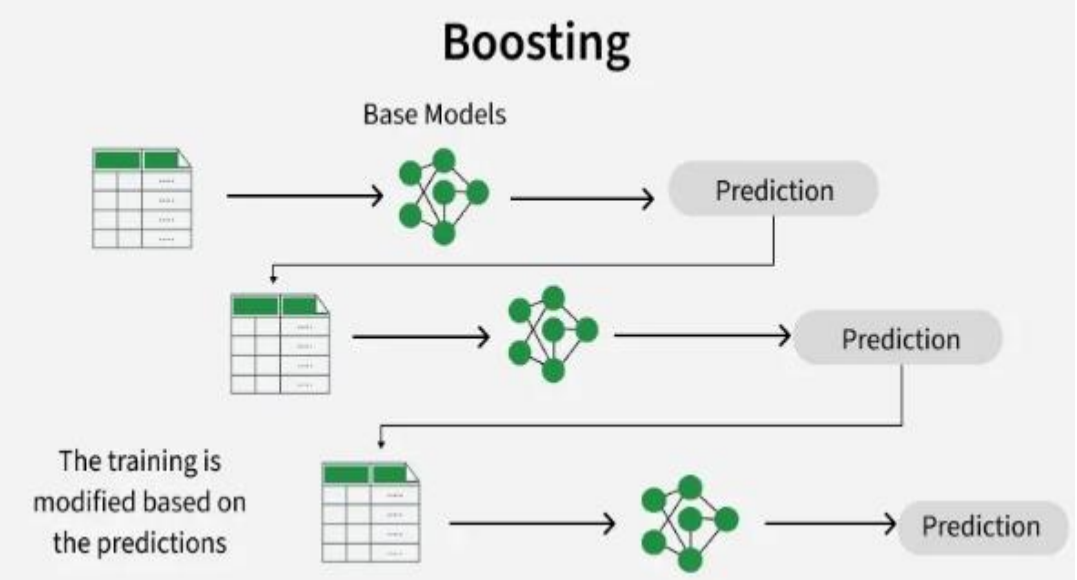
- **Bootstrap Sampling :** The dataset is divided into multiple subsets by sampling with replacement, creating diverse training data
- **Base Model Training** : A separate model is trained on each subset independently, often in parallel for efficiency
- **Prediction Aggregation** : Predictions from all models are combined using majority voting (classification) or averaging (regression)
- **OOB Evaluation** : Samples not included in a subset are used to evaluate model performance without cross-validation
- **Final Prediction** : The combined output of all models gives a more reliable and accurate result

Boosting:

Models are trained one after another. Each new model focuses on fixing the errors made by the previous ones. The final prediction is a weighted combination of all models, which helps reduce bias and improve accuracy.

Boosting Algorithm

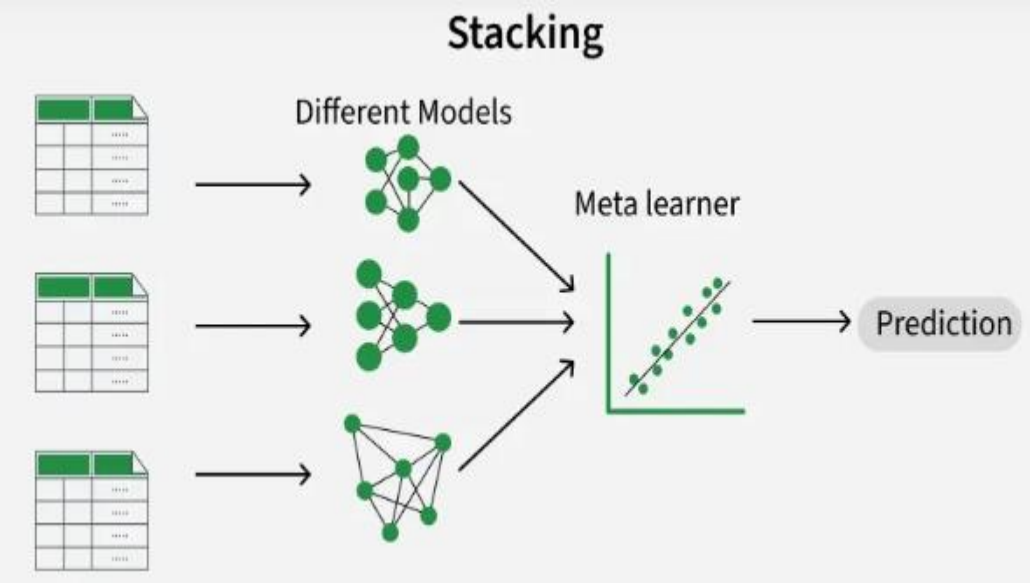
Boosting is an ensemble technique where multiple weak models are trained one after another, and each new model focuses on correcting the errors of the previous one to build a strong model. The process works as follows:



- **Initialize Weights** : Start with equal weights for all training data
- **Train Weak Learner** : Train a simple model on the dataset
- **Sequential Learning** : Each new model learns from previous errors
- **Weight Adjustment** : Misclassified samples get higher weights so future models focus more on them

3.Stacking (Stacked Generalization):

Multiple different models (often of different types) are trained and their predictions are used as inputs to a final model, called a meta-model. The meta-model learns how to best combine the predictions of the base models, aiming for better performance than any individual model.



Working of Stacking

1. The training dataset is provided to multiple base learning algorithms.
2. Each base model is trained independently on the same dataset.

3. Each base model makes predictions on the validation or training data.
4. These predictions become the input features for a new dataset.
5. A meta-learner is trained using this new dataset.
6. During testing, the base models first make predictions, and the meta-learner combines them to produce the final prediction.

Components of Stacking

- **Base Learners (Level-0 Models):** Multiple machine learning algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), Logistic Regression, or K-Nearest Neighbors (KNN).
- **Meta-Learner (Level-1 Model):** A model that learns from the predictions of the base learners and produces the final output. Logistic Regression is commonly used as the meta-learner.

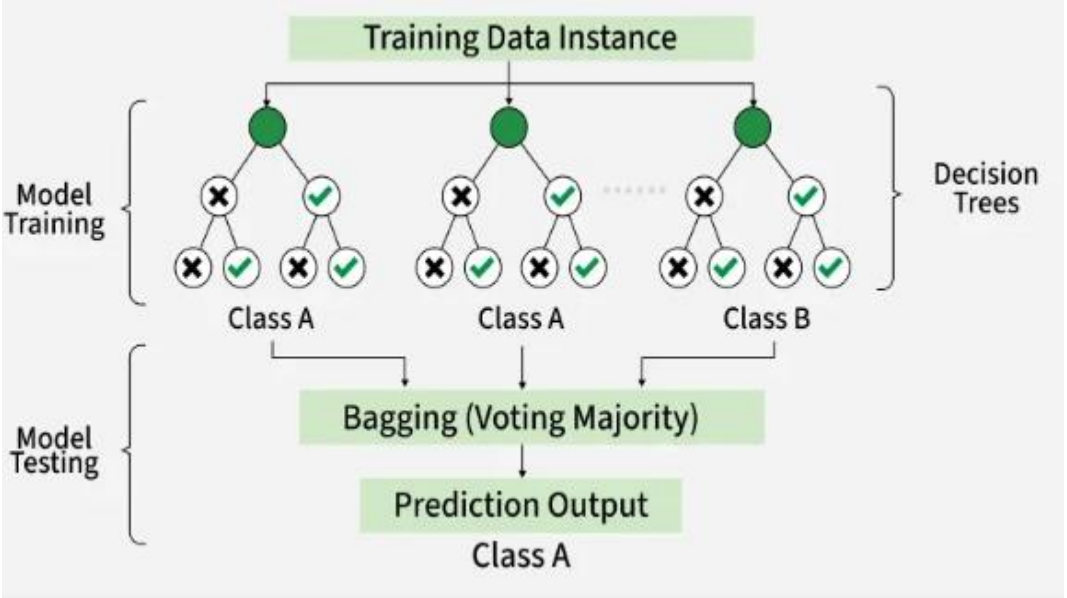
Advantages of Stacking

- Improves prediction accuracy by combining multiple models.
- Reduces errors made by individual models.
- Takes advantage of the strengths of different algorithms.
- Provides better generalization on unseen data.
- Works for both classification and regression problems.
- Handles complex datasets effectively.
- Reduces the impact of model bias.
- Can combine both homogeneous and heterogeneous models.

Random Forest:

Random Forest is a supervised ensemble learning algorithm used for both classification and regression problems. It creates multiple Decision Trees using randomly selected subsets of the training data and features, and combines their predictions to produce a more accurate, stable, and reliable result.

- Classification: Final prediction is based on Majority Voting.
- Regression: Final prediction is based on the Average of all predictions.



Working of Random Forest Algorithm

1. **Create Multiple Decision Trees:**

   The Random Forest algorithm creates a large number of Decision Trees by selecting different random subsets of the training data using **bootstrap sampling**. Since each tree is trained on a different subset of data, every tree is unique.

2. **Select Random Features:**

   While constructing each Decision Tree, the algorithm randomly selects a subset of features at every split instead of considering all available features. This increases diversity among the trees and reduces correlation between them.

3. **Train Each Decision Tree:**

   Each Decision Tree is trained independently using its corresponding bootstrap sample and randomly selected features. Every tree learns different patterns from the data.

4. **Make Individual Predictions:**

   When a new input is provided, each Decision Tree independently predicts the output based on the knowledge it has learned during training.

5. **Combine the Predictions:**

   The predictions of all Decision Trees are combined to obtain the final result.

   - **Classification:** The class receiving the highest number of votes (**majority voting**) is selected as the final prediction.
   - **Regression:** The final prediction is the **average** of the outputs produced by all Decision Trees.

6. **Improved Accuracy and Generalization:**

   Since the model combines the predictions of multiple diverse Decision Trees, it minimizes overfitting, improves prediction accuracy, and provides better generalization for unseen data.

Applications of Random Forest:

- **Medical Diagnosis:** Random Forest is used to predict diseases such as diabetes, heart disease, and cancer by analyzing patient medical records and clinical data.
- **Fraud Detection:** It is widely used in banking and financial institutions to identify fraudulent transactions and detect suspicious activities.
- **Customer Churn Prediction:** Random Forest helps businesses predict whether customers are likely to stop using their products or services, enabling them to improve customer retention.
- **Spam Email Detection:** It is used to classify emails as spam or non-spam by analyzing the content and characteristics of email messages.
- **Weather Forecasting:** Random Forest is used to predict weather conditions such as rainfall, temperature, and humidity by analyzing historical weather data.

Advantages of Random Forest :

- **High Prediction Accuracy:** Random Forest provides high prediction accuracy by combining the predictions of multiple Decision Trees.
- **Reduces Overfitting:** It reduces the risk of overfitting because the final prediction is based on the combined results of many Decision Trees rather than a single tree.
- **Handles Different Types of Problems:** Random Forest can effectively solve both classification and regression problems.
- **Works Well with Large Datasets:** It performs efficiently on large datasets and can handle a large number of input features.
- **Robust to Noise and Missing Values:** Random Forest can handle noisy data, outliers, and missing values without significantly affecting its performance.

Limitations of Random Forest:

- **High Memory Requirement:** Random Forest requires a large amount of memory because it stores multiple Decision Trees.
- **Long Training Time:** Training a Random Forest model takes more time than training a single Decision Tree due to the construction of many trees.
- **Less Interpretable:** The model is more difficult to interpret because it consists of multiple Decision Trees instead of a single tree.
- **Slower Prediction:** The prediction process can be slower since the algorithm must collect and combine predictions from all Decision Trees.
- **High Computational Cost:** Random Forest requires more computational resources, especially when

dealing with very large datasets and a large number of Decision Trees.

Bayesian Classification

Bayesian classification is a probabilistic machine learning approach that predicts the likelihood that a given data point belongs to a particular class. It relies on Bayes' Theorem to calculate the posterior probability of a class based on prior knowledge and new evidence, assigning the data point to the class with the highest probability

The fundamental foundation of these techniques is Bayes' Theorem:

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

Where:

- P(C|X) is the Posterior Probability: The probability that the class C is true given the observed data features X.
- P(X|C) is the Likelihood: The probability of generating the features X given that the class C is true.
- P(C) is the Prior Probability: The initial probability of class C before considering the data X.
- P(X) is the Evidence: The overall probability of observing the data features X.

Common Types of Bayesian Classifiers

1. Naive Bayes Classifier

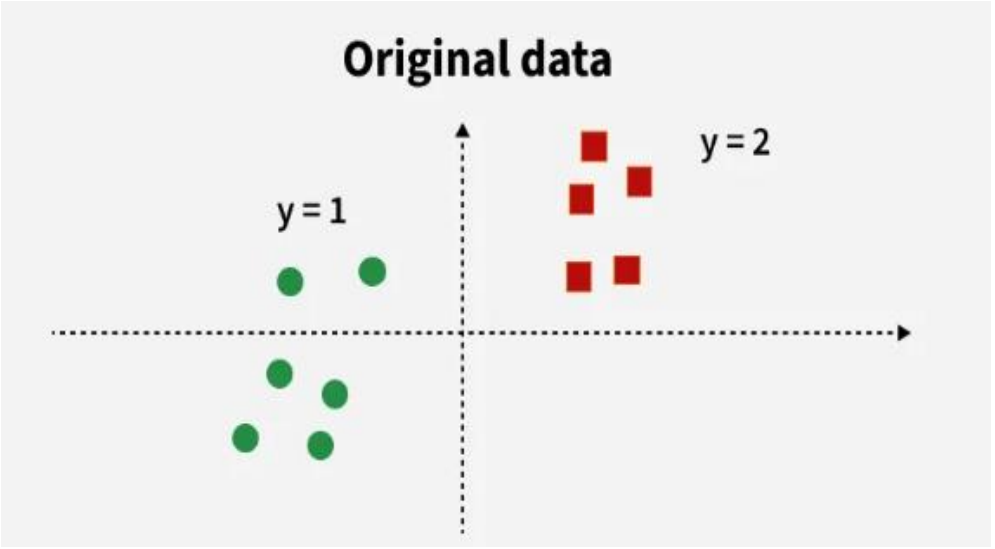
This is the most widely used variant. It is "naive" because it makes the simplifying assumption that all features in a dataset are completely **independent** of one another, given the class variable. Despite this simplifying assumption, it is highly scalable, fast, and often provides excellent accuracy, particularly in text classification (such as spam filtering and sentiment analysis).

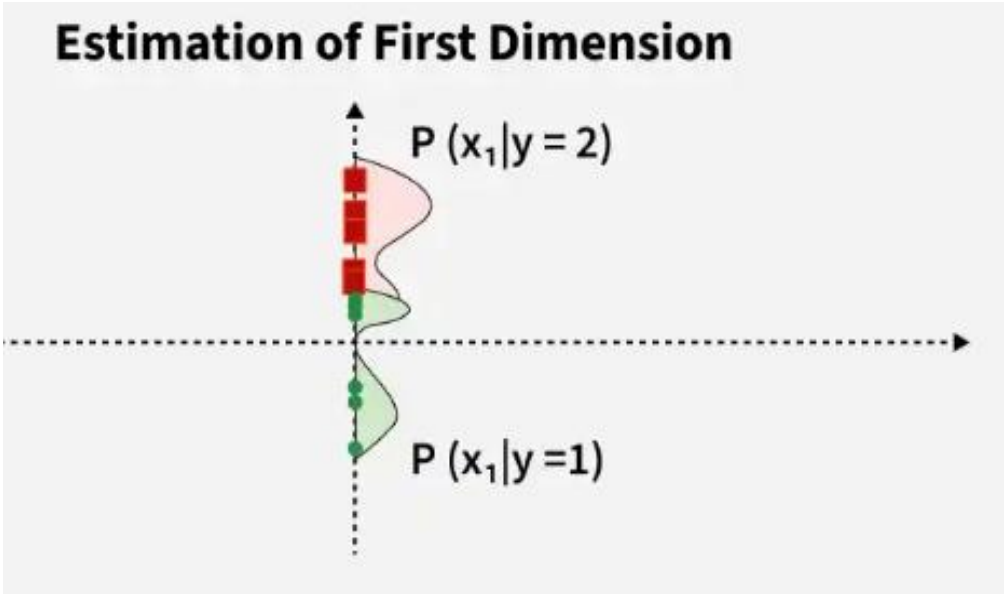
2. Bayesian Belief Networks (BBNs)

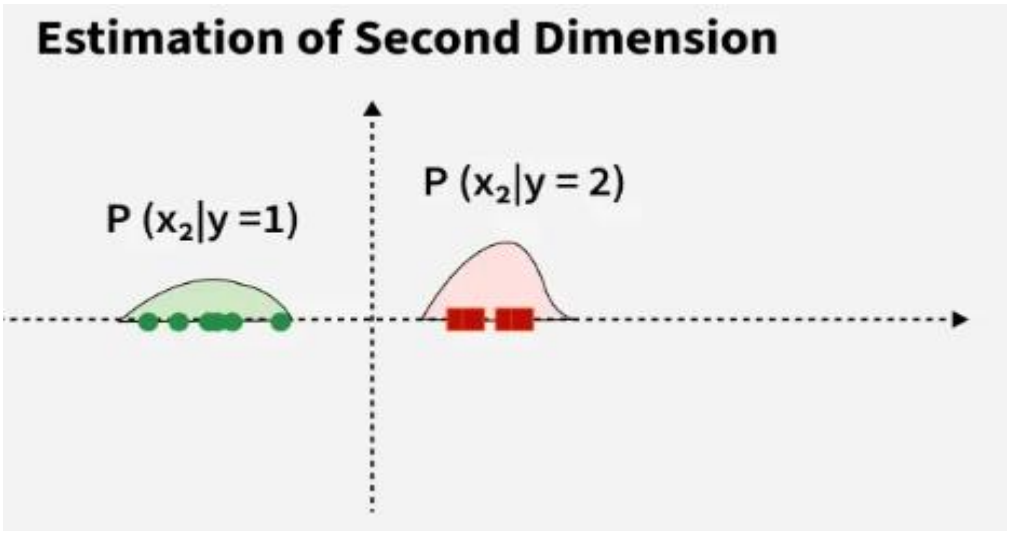
Unlike Naive Bayes, BBNs do not assume class-conditional independence. Instead, they use a Directed Acyclic Graph (DAG) to model dependencies and causal relationships between subsets of variables. BBNs compute joint conditional probabilities, allowing them to capture real-world complexities while still outputting probabilistic predictions.

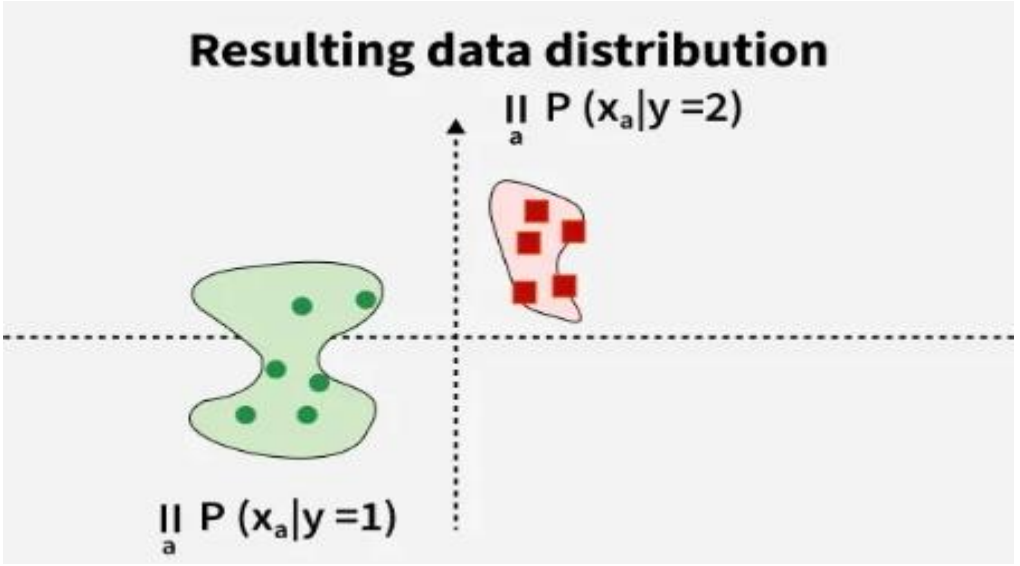
Naive Bayes Classifiers

Naïve Bayes Classifier is a supervised machine learning classification algorithm based on Bayes' Theorem. It is called "naïve" because it assumes that all input features are independent of each other, even though this assumption may not hold in real-world data. It is widely used for text classification, spam filtering, sentiment analysis, and medical diagnosis.









Here:

- Original data has two classes: green circles (y = 1) and red squares (y = 2).
- Estimate probability distribution along the first dimension i.e $P(x_1 \mid y = 1), P(x_1 \mid y = 2)$
- Estimate probability distribution along the second dimension i.e $P(x_2 \mid y = 1), P(x_2 \mid y = 2)$
- Combine both dimensions using conditional independence i.e $P(x \mid y) = \prod_{\alpha} P(x_\alpha \mid y)$

Key Features of Naive Bayes Classifiers

The main idea behind the Naive Bayes classifier is to use Bayes' Theorem to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification

- The Naive Bayes Classifier is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.
- It is a probabilistic classifier because it assumes that one feature in the model is independent of existence

of another feature. In other words, each feature contributes to the predictions with no relation between each other.

- Naive Bayes Algorithm is used in spam filtration, Sentimental analysis, classifying articles and many more.

Problem:

A hospital wants to predict whether a patient has **Flu** based on the symptom **Fever**.

The training data is given below.

Patient	Fever	Disease
P1	Yes	Flu
P2	Yes	Flu
P3	Yes	Flu
P4	No	No Flu
P5	No	No Flu

A new patient has **Fever = Yes**. Use the **Naïve Bayes Classifier** to predict whether the patient has **Flu**.

Step 1: Calculate Prior Probabilities

Total number of patients = **5**

Number of Flu patients = **3**

Number of No Flu patients = **2**

$$P(\text{Flu}) = \frac{3}{5} = 0.6$$

$$P(\text{No Flu}) = \frac{2}{5} = 0.4$$

Step 2: Calculate Likelihood Probabilities

For Flu

Among the 3 Flu patients:

- Fever = Yes → 3

$$P(\text{Fever=Yes} \mid \text{Flu}) = \frac{3}{3} = 1$$

For No Flu

Among the 2 No Flu patients:

- Fever = Yes → 0

$$P(\text{Fever=Yes} \mid \text{No Flu}) = \frac{0}{2} = 0$$

Step 3: Apply Bayes' Theorem

For Flu

$$P(\text{Flu} \mid \text{Fever=Yes}) \propto P(\text{Fever=Yes} \mid \text{Flu}) \times P(\text{Flu})$$
$$= 1 \times 0.6 = 0.6$$

For No Flu

$$P(\text{No Flu} \mid \text{Fever=Yes}) \propto P(\text{Fever=Yes} \mid \text{No Flu}) \times P(\text{No Flu})$$
$$= 0 \times 0.4 = 0$$

Step 4: Compare the Probabilities

Disease	Posterior Probability
Flu	**0.6**
No Flu	**0**

Since

$$0.6 > 0$$

the patient is classified as **Flu**.

The new patient is predicted to have Flu because the posterior probability of Flu (0.6) is greater than the posterior probability of No Flu (0).

Advantages:

- Naïve Bayes is simple and easy to understand and implement.
- It requires less training data to build an effective classification model.
- It provides fast training and prediction, making it suitable for large datasets.
- It performs well for high-dimensional data such as text documents.
- It works effectively for multi-class classification problems.

Limitations:

1. Naïve Bayes assumes that all features are independent, which is often unrealistic in real-world datasets.
2. Its performance decreases when the input features are highly correlated.
3. If a feature value is not present in the training data, the predicted probability may become zero unless smoothing techniques are used.

4. It cannot capture complex relationships among features.
5. Its prediction accuracy may be lower than more advanced machine learning algorithms for complex datasets.

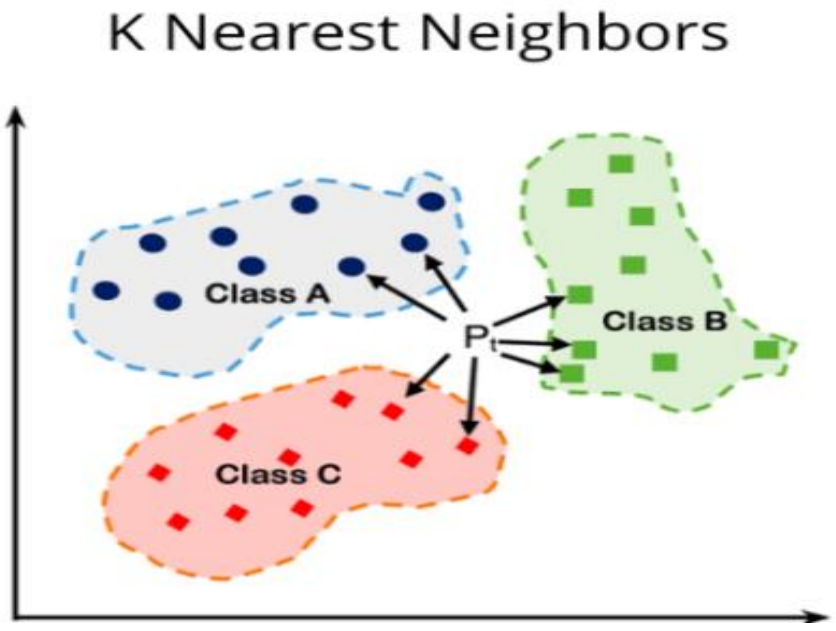
Applications:

1. Naïve Bayes is widely used for spam email detection by classifying emails as spam or non-spam.
2. It is used in sentiment analysis to determine whether a review or comment expresses a positive or negative opinion.
3. It is applied in document and news article classification.
4. It is used in medical diagnosis to predict diseases based on patient symptoms and medical records.
5. It is used in recommendation systems to classify user preferences and suggest relevant products or services.

k-Nearest Neighbour (KNN)

K-Nearest Neighbor (KNN) is a simple and widely used machine learning technique for classification and regression tasks. It works by identifying the *K* closest data points to a given input and making predictions based on the majority class or average value of those neighbors.

- Classifies data based on similarity with nearby data points
- Uses distance metrics like Euclidean distance to find nearest neighbors
- Since KNN makes no assumptions about the underlying data distribution, it makes it a non-parametric and instance-based learning method.



KNN is also called as a lazy learner algorithm because it does not learn from the training set immediately instead

it stores the entire dataset and performs computations only at the time of classification.

For example, consider two features i.e Category 1 and Category 2:

- KNN assigns the category based on the majority of nearby points. The image shows how KNN predicts the category of a new data point based on its closest neighbours.
- The green points represent Category 1 and the red points represent Category 2.
- The new data point checks its closest neighbors (circled points).
- Since the majority of its closest neighbors are red points (Category 2) KNN predicts the new data point belongs to Category 2.

KNN works by using proximity and majority voting to make predictions.

K' in K Nearest Neighbour

In the KNN algorithm k is just a number that tells the algorithm how many nearby points or neighbors to look at when it makes a decision.

Example: Imagine you're deciding which fruit it is based on its shape and size. You compare it to fruits you already know.

- If k = 3, the algorithm looks at the 3 closest fruits to the new one.
- If 2 of those 3 fruits are apples and 1 is a banana, the algorithm says the new fruit is an apple because most of its neighbors are apples.

How to choose the value of k for KNN Algorithm?

- The value of k in KNN decides how many neighbors the algorithm looks at when making a prediction.
- Choosing the right k is important for good results.
- If the data has lots of noise or outliers, using a larger k can make the predictions more stable.
- But if k is too large the model may become too simple and miss important patterns and this is called underfitting.
- So k should be picked carefully based on the data.

Statistical Methods for Selecting k

- **Cross-Validation**: Cross-Validation is a good way to find the best value of k is by using k-fold cross-validation. This means dividing the dataset into k parts. The model is trained on some of these parts and tested on the remaining ones. This process is repeated for each part.
- **Elbow Method**: In Elbow Method we draw a graph showing the error rate or accuracy for different k values. As k increases the error usually drops at first. But after a certain point error stops decreasing quickly. The point where the curve changes direction and looks like an "elbow" is usually the best choice for k.
- **Odd Values for k**: It’s a good idea to use an odd number for k especially in classification problems.

This helps avoid ties when deciding which class is the most common among the neighbors.

Distance Metrics Used in KNN Algorithm

KNN uses distance metrics to identify nearest neighbor, these neighbors are used for classification and regression task. To identify nearest neighbor we use below distance metrics:

1. Euclidean Distance

Euclidean distance is defined as the straight-line distance between two points in a plane or space. You can think of it like the shortest path you would walk if you were to go directly from one point to another.

$$d(x, X_i) = \sqrt{\sum_{j=1}^{n} \left(x_j - X_{ij}\right)^2}$$

2. Manhattan Distance

This is the total distance you would travel if you could only move along horizontal and vertical lines like a grid or city streets. It's also called "taxicab distance" because a taxi can only drive along the grid-like streets of a city.

$$d(x, y) = \sum_{i=1}^{n} | x_i - y_i |$$

3. Minkowski Distance

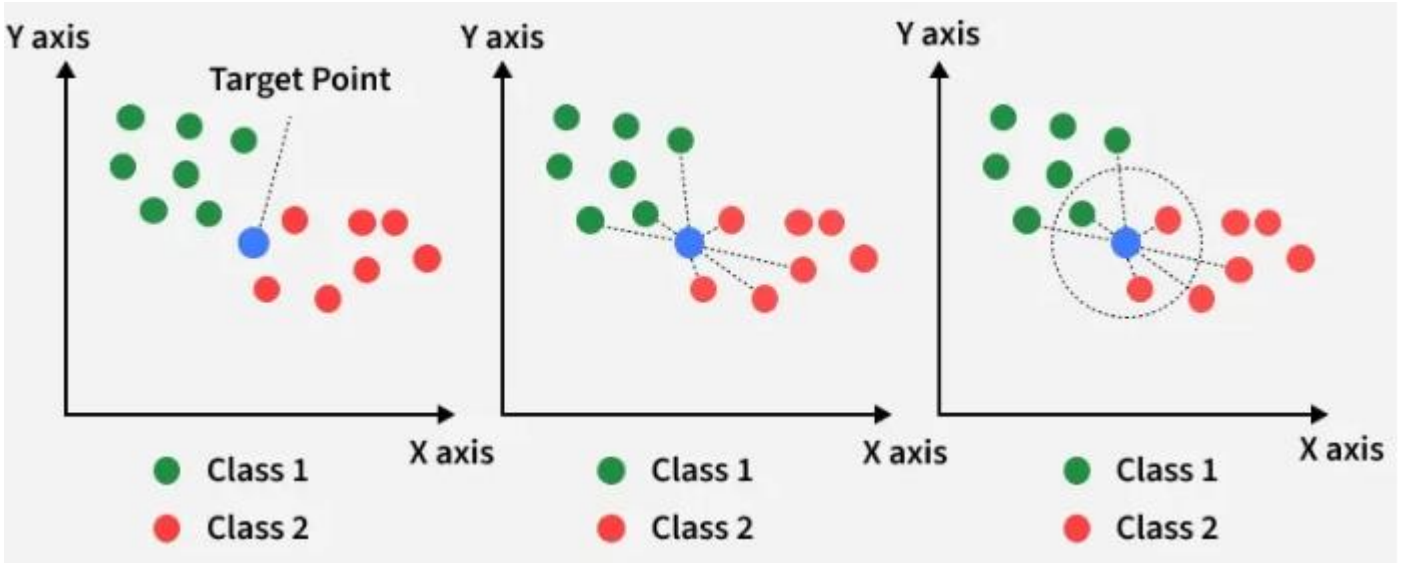
Minkowski distance is like a family of distances, which includes both Euclidean and Manhattan distances as special cases.

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}$$

From the formula above, when p=2, it becomes the same as the Euclidean distance formula and when p=1, it turns into the Manhattan distance formula. Minkowski distance is essentially a flexible formula that can represent either Euclidean or Manhattan distance depending on the value of p.

Working of KNN algorithm

The KNN algorithm operates on the principle of similarity where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbours in the training dataset.



step 1: Selecting the optimal value of K

- K is the number of nearest neighbours' considered for prediction.

Step 2: Calculating distance

- To measure the similarity between target and training data points Euclidean distance is widely used. Distance is calculated between data points in the dataset and target point.

Step 3: Finding Nearest Neighbours

- The k data points with the smallest distances to the target point are nearest neighbors.

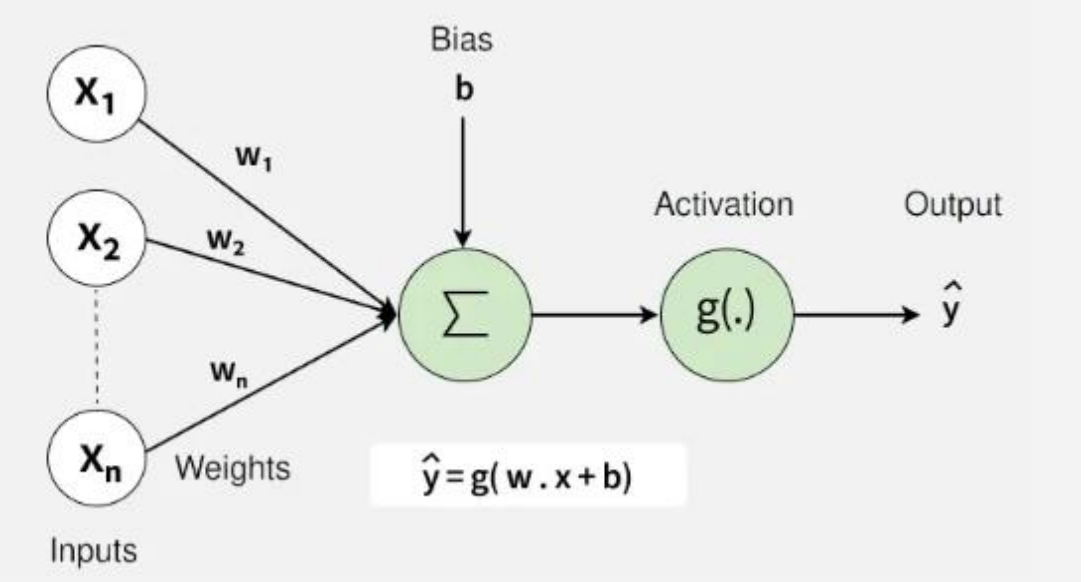
Step 4: Voting for Classification or Taking Average for Regression

- When you want to classify a data point into a category like spam or not spam, the KNN algorithm looks at the K closest points in the dataset. These closest points are called neighbors. The algorithm then looks at which category the neighbors belong to and picks the one that appears the most. This is called majority voting.
- In regression, the algorithm still looks for the K closest points. But instead of voting for a class in classification, it takes the average of the values of those K neighbors. This average is the predicted value for the new point for the algorithm.

It shows how a test point is classified based on its nearest neighbors. As the test point moves the algorithm identifies the closest 'k' data points i.e. 5 in this case and assigns test point the majority class label that is grey label class here.

Neural Networks

Neural networks are machine learning models that mimic the complex functions of the human brain. These models consist of interconnected nodes or neurons that process data, learn patterns directly from data without relying on pre-defined rules, enabling them to perform tasks such as pattern recognition and decision-making.

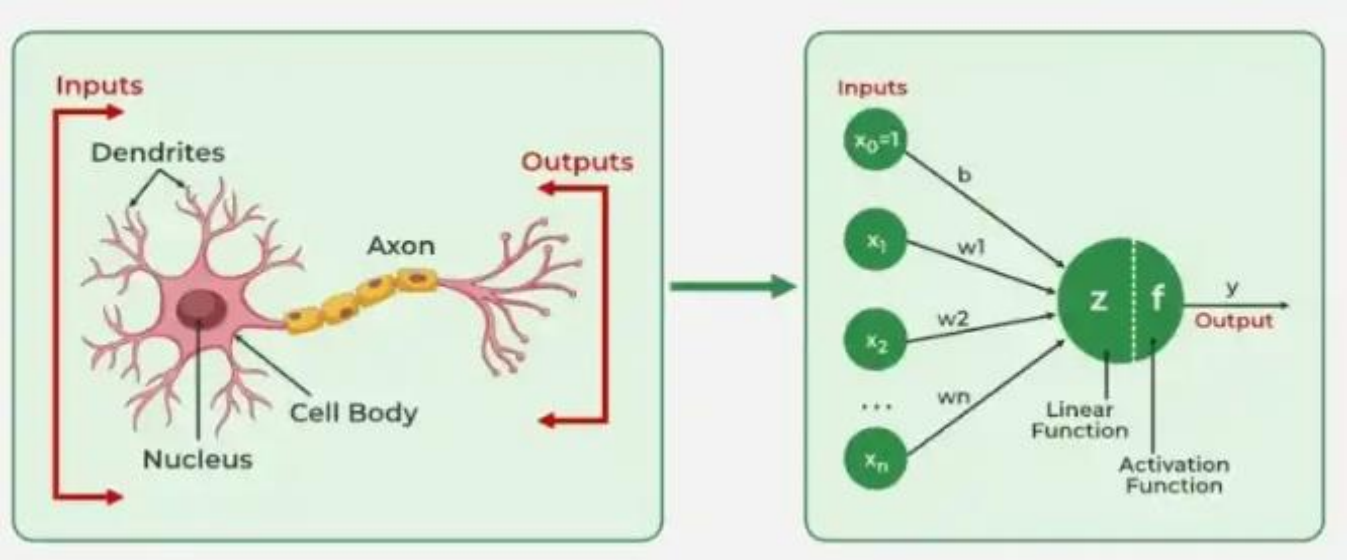


Neural Networks are built from several key components:

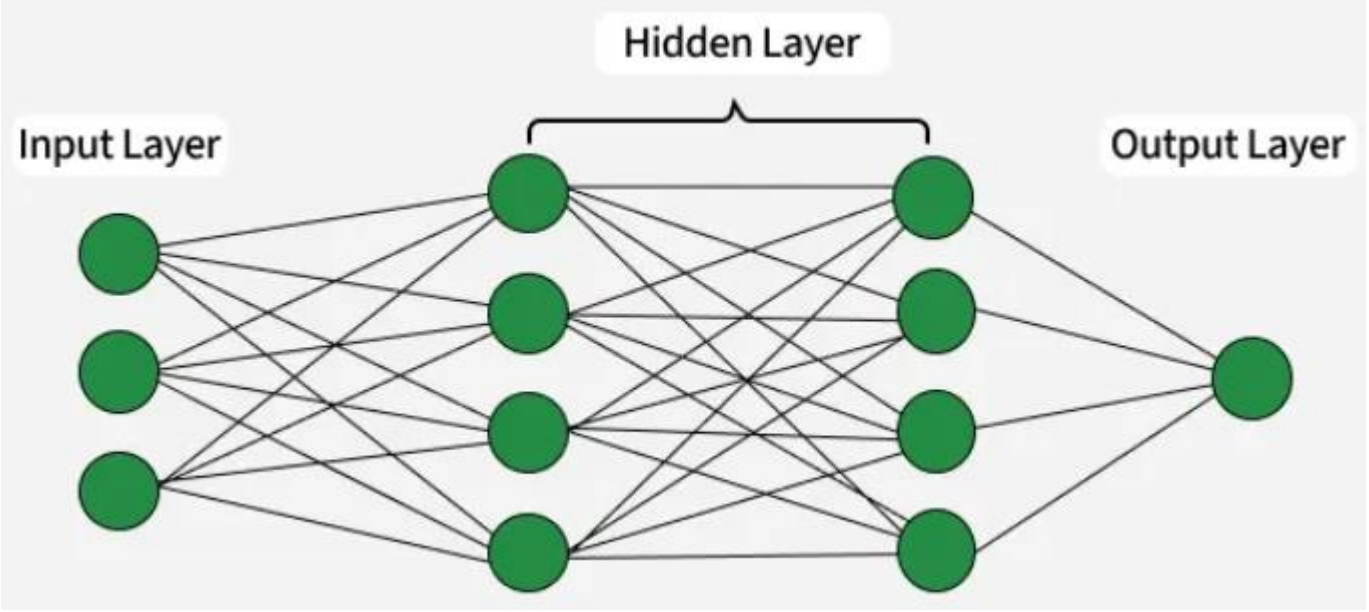
- **Neurons**: The basic units that receive inputs, each neuron is governed by a threshold and an activation function.
- **Connections**: Links between neurons that carry information, regulated by weights and biases.
- **Weights and Biases**: These parameters determine the strength and influence of connections.
- **Propagation Functions**: Mechanisms that help process and transfer data across layers of neurons.
- **Learning Rule**: The method that adjusts weights and biases over time to improve accuracy.

Learning in neural networks follows a structured, three-stage process:

1. **Input Computation**: Data is fed into the network.
2. **Output Generation**: Based on the current parameters, the network generates an output.
3. **Iterative Refinement**: The network refines its output by adjusting weights and biases, gradually improving its performance on diverse tasks.



Layers in Neural Network Architecture



1. **Input Layer:** This is where the network receives its input data. Each input neuron in the layer corresponds to a feature in the input data.
2. **Hidden Layers:** These layers perform most of the computational heavy lifting. A neural network can have one or multiple hidden layers. Each layer consists of units (neurons) that transform the inputs into something that the output layer can use.
3. **Output Layer:** The final layer produces the output of the model. The format of these outputs varies depending on the specific task like classification, regression.

Working of Neural Networks

1. Forward Propagation

When data is input into the network, it passes through the network in the forward direction, from the input layer through the hidden layers to the output layer. This process is known as forward propagation. Here's what happens during this phase:

1. Linear Transformation: Each neuron in a layer receives inputs which are multiplied by the weights associated with the connections. These products are summed together and a bias is added to the sum. This can be represented mathematically as:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

where

- w represents the weights
- x represents the inputs
- b is the bias

2. Activation: The result of the linear transformation (denoted as z) is then passed through an activation function. The activation function is crucial because it introduces non-linearity into the system, enabling the network to learn more complex patterns. Popular activation functions include ReLU, sigmoid and tanh.

2. Backpropagation

After forward propagation, the network evaluates its performance using a loss function which measures the difference between the actual output and the predicted output. The goal of training is to minimize this loss. This is where backpropagation comes into play:

- **Loss Calculation:** The network calculates the loss which provides a measure of error in the predictions. The loss function could vary; common choices are mean squared error for regression tasks or cross-entropy loss for classification.
- **Gradient Calculation:** The network computes the gradients of the loss function with respect to each weight and bias in the network. This involves applying the chain rule of calculus to find out how much each part of the output error can be attributed to each weight and bias.
- **Weight Update:** Once the gradients are calculated, the weights and biases are updated using an optimization algorithm like stochastic gradient descent (SGD). The weights are adjusted in the opposite direction of the gradient to minimize the loss. The size of the step taken in each update is determined by the learning rate.

3. Iteration

This process of forward propagation, loss calculation, backpropagation and weight update is repeated for many iterations over the dataset. Over time, this iterative process reduces the loss and the network's predictions become more accurate.

Through these steps, neural networks can adapt their parameters to better approximate the relationships in the data, thereby improving their performance on tasks such as classification, regression or any other predictive modeling.

Example of Email Classification

Let's consider a record of an email dataset:

Email ID	Email Content	Sender	Subject Line	Label
1	"Get free gift cards now!"	spam@example.com	"Exclusive Offer"	1

To classify this email, we will create a feature vector based on the analysis of keywords such as "free" "win" and "offer"

The resulting feature vector is:

- **free:** Present (1)
- **win:** Absent (0)
- **offer:** Present (1)

Feature Vector:

[1, 0, 1]

The feature vector is passed through the neural network, where each neuron computes a weighted sum of the inputs and applies an activation function to produce its output.

Hidden Layer Calculation:

Assume the hidden layer contains two neurons with the following weights

- Neuron H1: [0.5, −0.2, 0.3]
- Neuron H2: [0.4, 0.1, −0.5]

For the input vector [1, 0, 1], the weighted sums are:

- **For H1: (**1 × 0.5) +(0 ×− 0.2) + (1 × 0.3) = 0.5 + 0+ 0.3 = 0.8
- **For H2:** (1 × 0.4) +(0 × 0.1) + (1 × −0.5) = 0.4 + 0 −0.5 =−0.1

Applying the ReLU activation function:

- **H1 Output:** ReLU(0.8) = 0.8
- **H2 Output:** ReLu(-0.1) = 0

Output Layer: The outputs from the hidden layer are then passed to the output neuron.

- **Output Weights:** [0.7, 0.2]
- **Input from Hidden Layer:** [0.8, 0]
- **Weighted Sum:** (0.8×0.7)+(0×0.2)=0.56+0=0.56
- **Activation (Sigmoid):** $\sigma(0.56) = \frac{1}{1+e^{-0.56}} \approx 0.636$

Final Classification:

- The output value of approximately **0.636** indicates the probability of the email being spam.
- Since this value is greater than 0.5, the neural network classifies the email as spam (1).

Types of Neural Networks

There are several types of neural networks, including:

- **Feedforward Networks:** It is a simple artificial neural network architecture in which data moves from input to output in a single direction.
- **Single layer Perceptron:** It has one layer and it applies weights, sums inputs and uses activation to produce output.
- **Multilayer Perceptron (MLP):** It is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers and an output layer.
- **Convolutional Neural Network (CNN):** It is designed for image processing. It uses convolutional layers to automatically learn features from input images, enabling effective image recognition and classification.
- **Recurrent Neural Network (RNN):** Handles sequential data using feedback loops to retain context over time.
- **Long Short-Term Memory (LSTM):** A type of RNN with memory cells and gates to handle long-term dependencies and avoid vanishing gradients.

Importance of Neural Networks

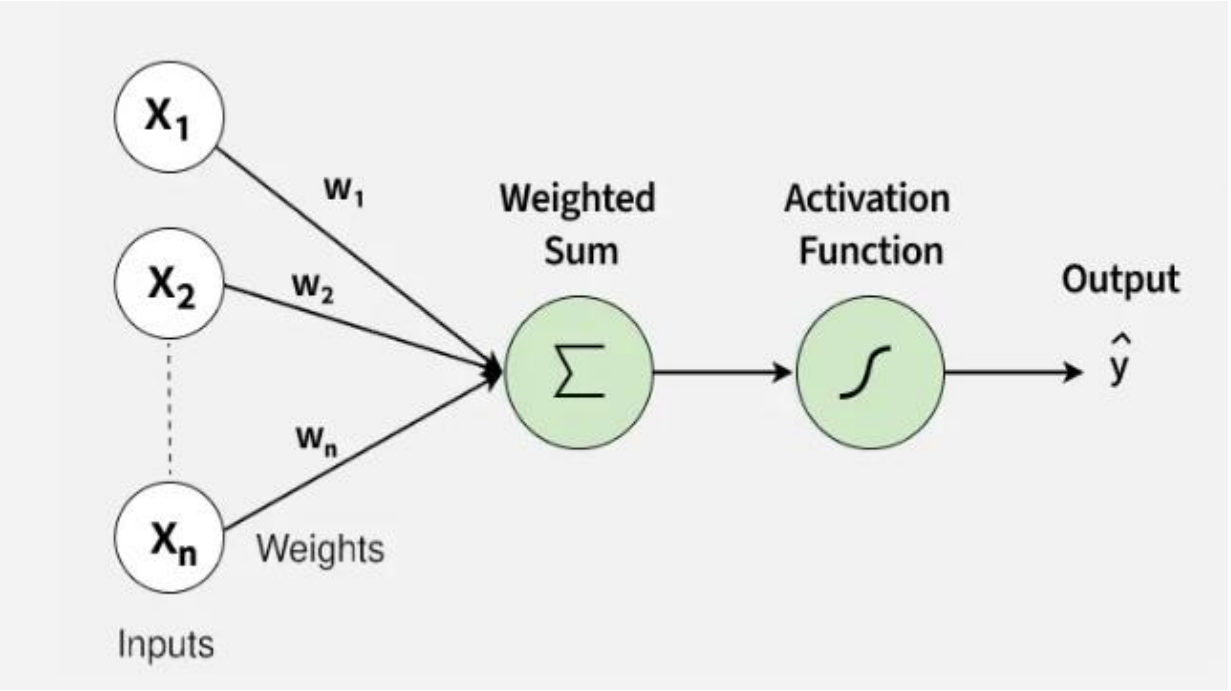
- **Identify Complex Patterns:** Recognize intricate structures and relationships in data; adapt to dynamic and changing environments.
- **Learn from Data:** Handle vast datasets efficiently; improve performance with experience and retraining.
- **Drive Key Technologies:** Power natural language processing (NLP); enable self-driving vehicles; support automated decision-making systems.
- **Boost Efficiency:** Streamline workflows and processes; enhance productivity across industries.
- **Backbone of AI:** Serve as the core driver of artificial intelligence progress; continue shaping the future of technology and innovation.

The Perceptron, Multilayer Perceptron(MLP)

Perceptron:

The Perceptron is a single-layer neural network that receives multiple input values, multiplies each input

by its corresponding weight, adds a bias, and passes the result through a step activation function to produce a binary output. It is used to classify linearly separable data into two classes.



1. Inputs $(x_1, x_2, \ldots, x_n)$

These are the features or measurable attributes of a data point that the perceptron uses to make a decision. Each input provides a signal that contributes to the final output.

- **Example**: For an OR gate, the inputs are binary: $(x_1, x_2) \in \{0,1\}^2$
- Inputs themselves have no inherent influence unless multiplied by weights.

2. Weights $(w_1, w_2, \ldots, w_n)$

Weights determine how strongly each input contributes to the prediction. A larger weight means the corresponding input has a higher impact.

- Weights are learned during training, adjusting based on errors.
- They act like importance scores for each feature.

3. Bias (b)

The bias is a constant value added to the weighted sum to shift the decision boundary.

- It allows the perceptron to classify correctly even when all input features are zero.
- Bias ensures the model is not forced to pass the decision boundary through the origin.

Difference Between Weights and Bias

- Weights control how much each input influences the output.

- Bias controls when the perceptron activates, independent of any input.
- Mathematically, Weights tilt the line and Bias shifts the line up/down or left/right.

4. Net Input (Weighted Sum)

This is the combined effect of all inputs and their weights:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

- Represents the activation strength before passing through the activation function.
- If z is high or low enough, it determines the final class.

5. Activation Function (Step Function)

The activation function converts the numerical input into a binary output:

$$\hat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- It introduces non-linearity in the decision-making, although the decision boundary remains linear.
- Output is always 0 or 1 making perceptrons suitable for binary classification.

Types of Perceptron Models: Based on the layers, Perceptron models are divided into two types. These are as follows:

Single-layer Perceptron Model

Multi-layer Perceptron model

Single Layer Perceptron model

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

Fundamentals of Neural Network

A neural network extends the perceptron by connecting many neurons across multiple layers.

1. Input layer: The input layer provides the network with the raw feature vector:

$$x = (x_1, x_2, \ldots, x_n)$$

- No computation happens here.
- It simply passes the input values to the next layer.

2. Hidden layers: Hidden layers contain multiple perceptrons (neurons) that learn intermediate representations of the data.

Hidden Layer Computation:

$$z^{(1)} = W^{(1)}\mathbf{x} + b^{(1)}$$

$$a^{(1)} = \sigma(z^{(1)})$$

where:

- $W^{(1)}$: weight matrix for hidden layer
- $b^{(1)}$: bias vector
- σ: non-linear activation function (ReLU, Sigmoid, Tanh, etc.)
- Hidden layers identify complex patterns not visible from raw input alone.
- Adding more hidden layers improves model expressiveness.

3. Output layer: The output layer produces the final prediction, which may be binary, multi-class or a continuous value.

Output Layer Computation:

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$

$$\hat{y} = \sigma(z^{(2)})$$

Output activation depends on the task:

- **Sigmoid:** binary classification
- **Softmax:** multi-class classification
- **Linear:** regression

Because of multiple layers and non-linear activations, neural networks can model complex, non-linear decision boundaries, while a single perceptron can only model a straight line.

Perceptron vs. Multi-Layer Perceptron (MLP)

Aspect	Perceptron	Multi-Layer Perceptron (MLP)
Model Depth	Single layer with no hidden neurons.	Multiple layers with one or more hidden layers.
Type of Patterns Learned	Learns only linear relationships; straight-line separation.	Learns complex, non-linear patterns and curved boundaries.

Aspect	Perceptron	Multi-Layer Perceptron (MLP)
Problem-Solving Ability	Cannot solve XOR or non-linearly separable problems.	Easily solves XOR and other complex classification tasks.
Activation Functions	Uses a simple step function (hard 0/1 output).	Uses advanced activations like ReLU, Sigmoid, Tanh for richer learning.
Learning Method	Trained with a simple perceptron update rule.	Trained using backpropagation and gradient descent.
Real-World Use	Limited to simple demonstrations and basic classification.	Used in real-world AI systems like vision, NLP and deep learning applications.

Applications

- **Binary Classification**: Used for simple yes/no decision problems such as spam detection or basic quality checks.
- **Logic Gate Modelling**: Implements linearly separable gates like AND or NAND with perfect accuracy.
- **Pattern Recognition Basics**: Helps identify simple patterns where classes can be separated with a straight line.
- **Feature Importance Insight**: Weight values indicate which features influence the output more strongly.
- **Educational Tool**: Commonly used to teach foundations of machine learning and neural networks.

Advantages

- **Simple Architecture**: Easy to understand, implement and visualize as a basic neural model.
- **Fast Training**: Lightweight computations make learning very quick even on small devices.
- **Works on Linearly Separable Data**: Achieves perfect performance when a straight-line boundary exists.
- **Low Resource Requirement**: Requires minimal memory and computation, suitable for tiny datasets.
- **Foundation for Deep Learning**: Forms the conceptual basis for multilayer perceptrons and complex networks.

Limitations

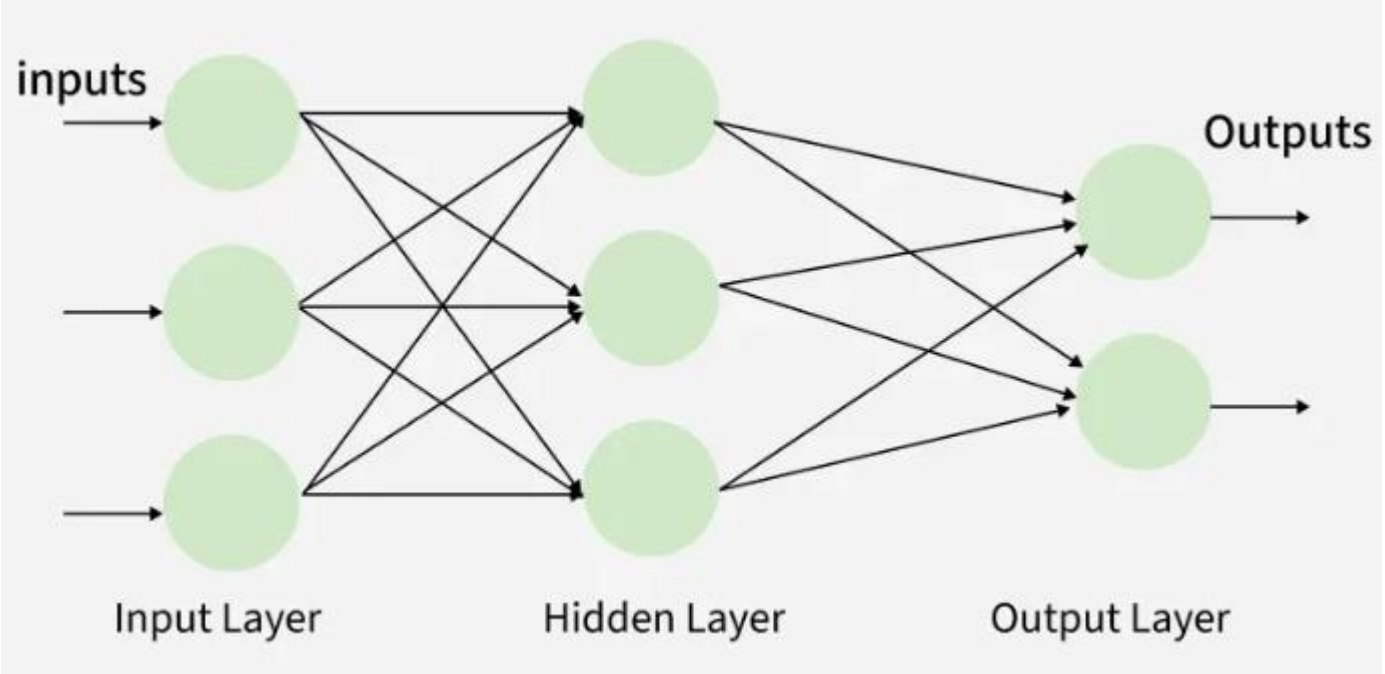
- **No Non-Linear Learning**: Fails on problems like XOR where a curved or complex boundary is needed.

- **Binary Output Only**: Cannot handle multi-class or probabilistic outcomes without extensions.
- **No Hidden Layers**: Lacks depth, making it unable to learn hierarchical or abstract patterns.
- **Sensitive to Data Scale**: Performance drops if features are not normalized or scaled properly.
- **Not Suitable for Real-World ML**: Too simplistic for modern tasks like vision, NLP or sequence modeling.

Multi-Layer Perceptron Model:

Multi-Layer Perceptron (MLP) consists of fully connected dense layers that transform input data from one dimension to another. It is called multi-layer because it contains an input layer, one or more hidden layers and an output layer. The purpose of an MLP is to model complex relationships between inputs and outputs.

Components of MLP



- **Input Layer**: Each neuron or node in this layer corresponds to an input feature. For instance, if you have three input features the input layer will have three neurons.
- **Hidden Layers**: MLP can have any number of hidden layers with each layer containing any number of nodes. These layers process the information received from the input layer.
- **Output Layer**: The output layer generates the final prediction or result. If there are multiple outputs, the output layer will have a corresponding number of neurons.

Working of Multi-Layer Perceptron

Let's see working of the multi-layer perceptron. The key mechanisms such as forward propagation, loss function, backpropagation and optimization.

1. Forward Propagation

In forward propagation the data flows from the input layer to the output layer, passing through any hidden layers. Each neuron in the hidden layers processes the input as follows:

Weighted Sum: The neuron computes the weighted sum of the inputs:

$$z = \sum_i w_i x_i + b$$

Where:

- x_i is the input feature.
- w_i is the corresponding weight.
- b is the bias term.

Activation Function: The weighted sum z is passed through an activation function to introduce non-linearity. Common activation functions include:

- **Sigmoid**: $\sigma(z) = \frac{1}{1+e^{-z}}$
- **ReLU (Rectified Linear Unit)**: $f(z) = \max(0, z)$
- **Tanh (Hyperbolic Tangent)**: $\tanh(z) = \frac{2}{1+e^{-2z}} - 1$

2. Loss Function

Once the network generates an output the next step is to calculate the loss using a loss function. In supervised learning this compares the predicted output to the actual label.

For a classification problem the commonly used binary cross-entropy loss function is:

$$L = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Where:

- y_i is the actual label.
- $\hat{y}_i$ is the predicted label.
- N is the number of samples.

For regression problems the mean squared error (MSE) is often used:

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

3. Backpropagation

The goal of training an MLP is to minimize the loss function by adjusting the network's weights and biases. This is achieved through backpropagation:

1. **Gradient Calculation**: The gradients of the loss function with respect to each weight and bias are calculated using the chain rule of calculus.
2. **Error Propagation**: The error is propagated back through the network, layer by layer.
3. **Gradient Descent**: The network updates the weights and biases by moving in the opposite direction of the gradient to reduce the loss: $w = w - \eta \cdot \frac{\partial L}{\partial w}$

Where:

- w is the weight.
- η is the learning rate.
- $\frac{\partial L}{\partial w}$ is the gradient of the loss function with respect to the weight.

4. Optimization

MLPs rely on optimization algorithms to iteratively refine the weights and biases during training. Popular optimization methods include:

- **Stochastic Gradient Descent (SGD)**: Updates the weights based on a single sample or a small batch of data: $w = w - \eta \cdot \frac{\partial L}{\partial w}$
- **Adam Optimizer**: An extension of SGD that incorporates momentum and adaptive learning rates for more efficient training:
- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t$
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2$

Here g_t represents the gradient at time t and β_1, β_2 are decay rates.

Now that we are done with the theory part of multi-layer perception, let's go ahead and implement code in python using the TensorFlow library.

Support Vector Machines

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It classifies data by finding the optimal hyperplane that separates different classes with the maximum margin. The data points closest to the hyperplane are called support vectors, and they play a crucial role in determining the position of the decision boundary.

Basic Concepts of SVM

1. Hyperplane

A hyperplane is the decision boundary that separates data points belonging to different classes.

2. Support Vectors

Support vectors are the data points that are closest to the hyperplane. They determine the position and orientation of the hyperplane.

3. Margin

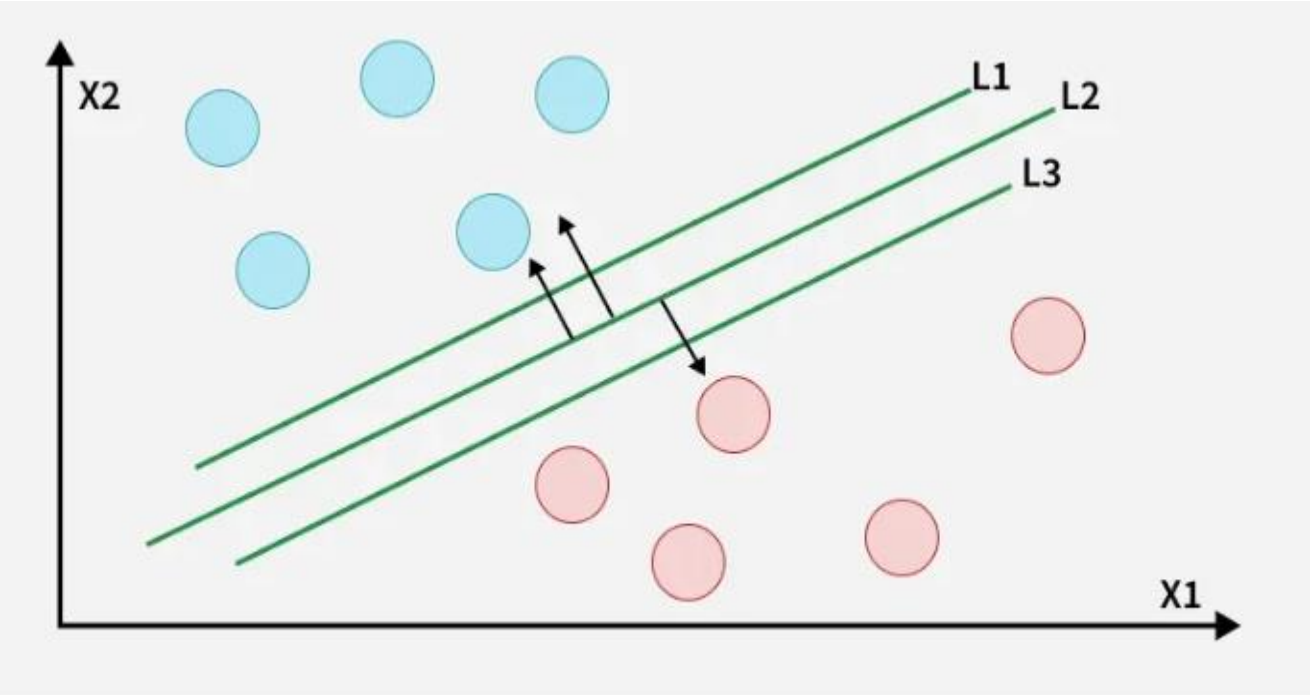
The margin is the distance between the hyperplane and the nearest support vectors. SVM aims to maximize this margin to improve classification accuracy.

Key Concepts

- **Hyperplane**: A decision boundary separating different classes in feature space and is represented by the equation wx + b = 0 in linear classification.
- **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
- **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
- **Kernel**: A function that maps data to a higher-dimensional space enabling SVM to handle non-linearly separable data.
- **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
- **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
- **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value forces stricter penalty for misclassifications.
- **Hinge Loss**: A loss function penalizing misclassified points or margin violations and is combined with regularization in SVM.
- **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

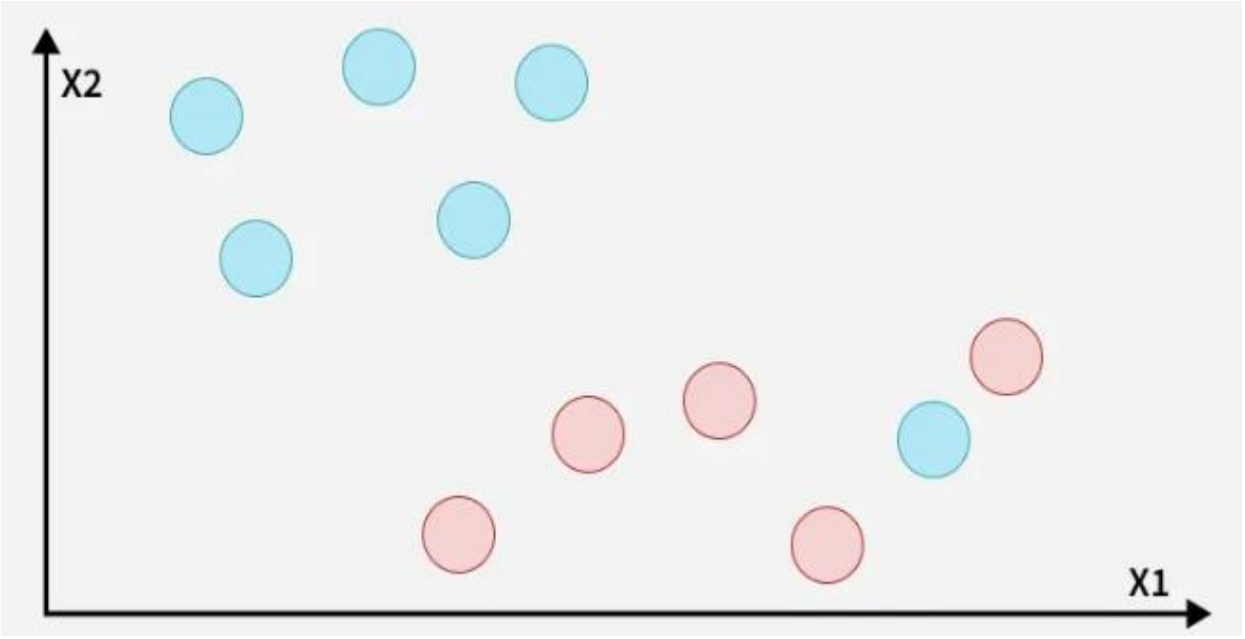
Working of Support Vector Machine Algorithm

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



Multiple hyperplanes separate the data from two classes

The best hyperplane also known as the hard margin is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So from the above figure, we choose L2 as hard margin. Let's consider a scenario like shown below:



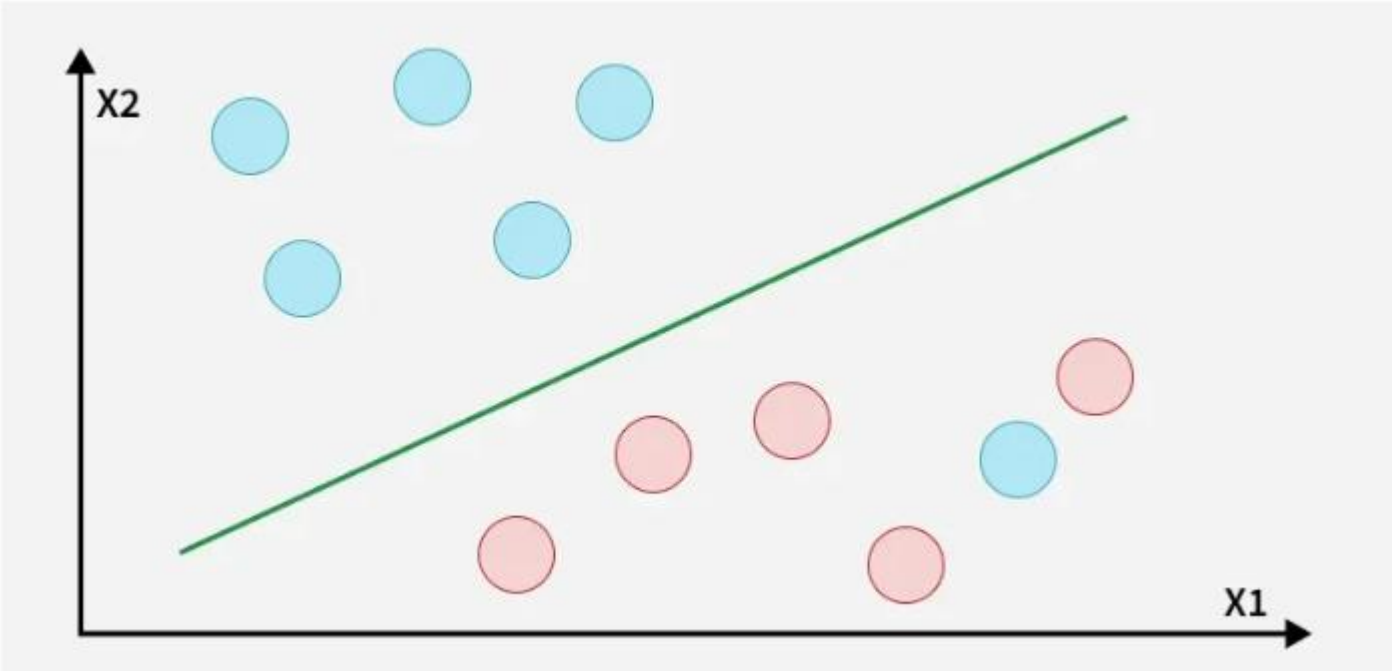
Types of Support Vector Machine

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

Linear SVM

- Linear SVMs use a linear decision boundary to separate the data points of different classes.
- When the data can be precisely linearly separated, linear SVMs are very suitable.
- This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes.
- A hyperplane that maximizes the margin between the classes is the decision boundary.

The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM can be sensitive to outliers, especially in the case of a hard margin, while soft margin SVM helps reduce their impact by allowing some misclassifications.



A soft margin allows for some misclassifications or violations of the margin to improve generalization. The SVM optimizes the following equation to balance margin maximization and penalty minimization:

$$\text{Objective Function} = \left(\frac{1}{\text{margin}}\right) + \lambda \sum \text{penalty}$$

The penalty used for violations is often hinge loss which has the following behavior:

- If a data point is correctly classified and lies outside the margin, there is no penalty (loss = 0). If it is correctly classified but lies within the margin or is misclassified, the hinge loss is greater than 0.
- If a point is incorrectly classified or violates the margin the hinge loss increases proportionally to the distance of the violation.

Till now we were talking about linearly separable data that separates group of blue balls and red balls by a straight line/linear line.

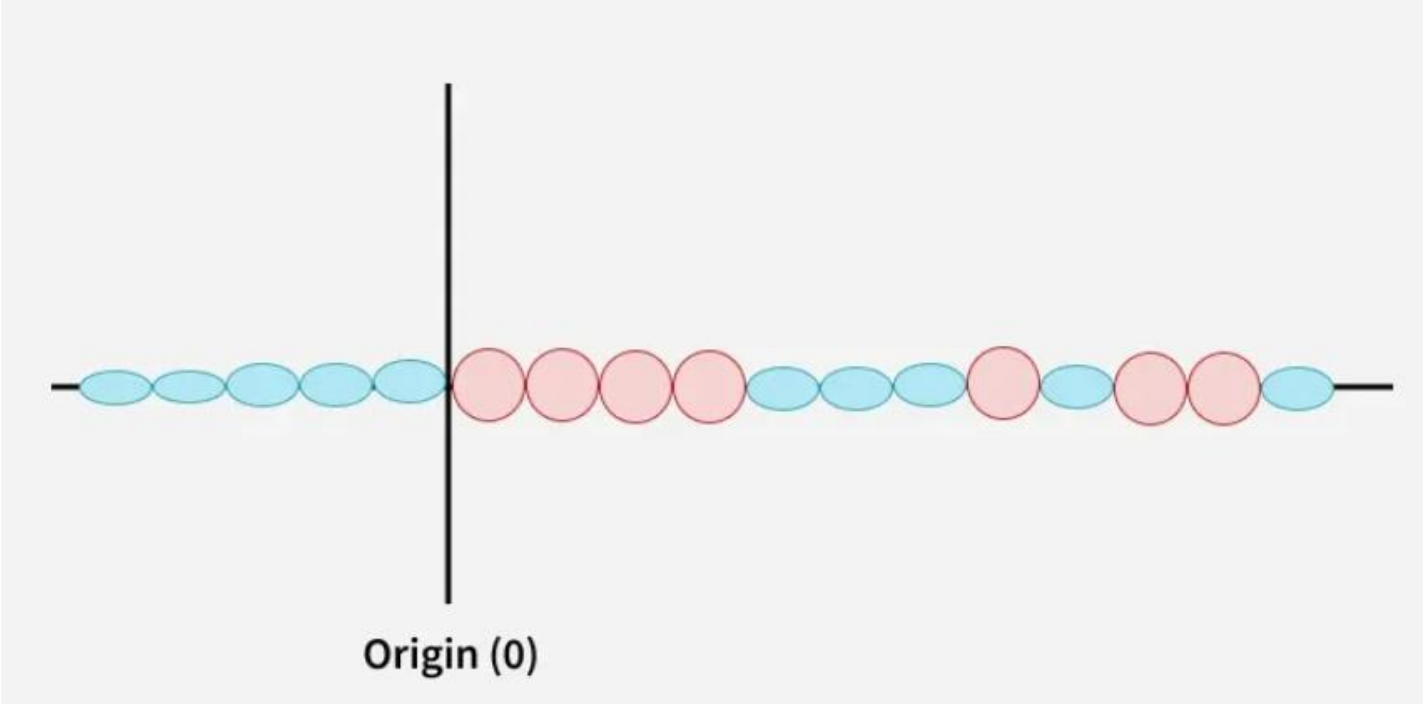
Non-Linear SVM

- Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight

line (in the case of 2D).

- By using kernel functions, nonlinear SVMs can handle nonlinearly separable data.
- The original input data is transformed by these kernel functions into a higher-dimensional feature space where the data points can be linearly separated.
- A linear SVM is used to locate a nonlinear decision boundary in this modified space.

hen data is not linearly separable i.e it can't be divided by a straight line, SVM uses a technique called kernels to map the data into a higher-dimensional space where it becomes separable. This transformation helps SVM find a decision boundary even for non-linear data.



Original 1D dataset for classification

A kernel is a function that maps data points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping. For example consider data points 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.

- **Linear Kernel**: For linear separability.
- **Polynomial Kernel**: Maps data into a polynomial space.
- **Radial Basis Function (RBF) Kernel**: Transforms data into a space based on distances between data points.

Mathematical Computation of SVM

Consider a binary classification problem with two classes, labeled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y. The equation for the linear

hyperplane can be written as:

$$w^T x + b = 0$$

Where:

- w is the normal vector to the hyperplane (the direction perpendicular to it).
- b is the offset or bias term representing the distance of the hyperplane from the origin along the normal vector w.

Distance from a Data Point to the Hyperplane

The distance between a data point $x_i$and the decision boundary can be calculated as:

$$d_i = \frac{w^T x_i + b}{|| w ||}$$

where ||w|| represents the Euclidean norm of the weight vector w.

Linear SVM Classifier

Distance from a Data Point to the Hyperplane:

$$\hat{y} = \begin{cases} 1 & : w^T x + b \geq 0 \\ -1 & : w^T x + b < 0 \end{cases}$$

Where $\hat{y}$ is the predicted label of a data point.

Optimization Problem for SVM

For a linearly separable dataset the goal is to find the hyperplane that maximizes the margin between the two classes while ensuring that all data points are correctly classified. This leads to the following optimization problem:

$$\text{minimize}_{w,b} \frac{1}{2} \| w \|^2$$

Subject to the constraint:

$$y_i(w^T x_i + b) \geq 1 for i = 1,2,3,\cdots,m$$

Where:

- y_i is the class label (+1 or -1) for each training instance.
- x_i is the feature vector for the i-th training instance.
- m is the total number of training instances.

The condition $y_i(w^T x_i + b) \geq 1$ ensures that each data point is correctly classified and lies outside the margin.

Soft Margin in Linear SVM Classifier

In the presence of outliers or non-separable data the SVM allows some misclassification by introducing slack variables ζ_i. The optimization problem is modified as:

$$\text{minimize w, b}\frac{1}{2}\parallel w \parallel^2 + C\sum_{i=1}^{m}\zeta_i$$

Subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \zeta_i \text{and} \zeta_i \geq 0 \text{for } i = 1,2,\dots,m$$

Where:

- C is a regularization parameter that controls the trade-off between margin maximization and penalty for misclassifications.
- ζ_i are slack variables that represent the degree of violation of the margin by each data point.

Dual Problem for SVM

The dual problem involves maximizing the Lagrange multipliers associated with the support vectors. This transformation allows solving the SVM optimization using kernel functions for non-linear classification.

The dual objective function is given by:

$$\max_\alpha \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\,\alpha_j t_i t_j K(x_i, x_j) - \sum_{i=1}^{m}\alpha_i$$

Where:

- α_i are the Lagrange multipliers associated with the i^{th} training sample.
- t_i is the class label for the i^{th}-th training sample.
- $K(x_i, x_j)$ is the kernel function that computes the similarity between data points x_i and x_j. The kernel allows SVM to handle non-linear classification problems by mapping data into a higher-dimensional space.

The dual formulation optimizes the Lagrange multipliers α_i and the support vectors are those training samples where $\alpha_i > 0$.

SVM Decision Boundary

Once the dual problem is solved, the decision boundary is given by:

$$w = \sum_{i=1}^{m}\alpha_i\, t_i K(x_i, x) + b$$

Where w is the weight vector, x is the test data point and b is the bias term. Finally the bias term b is determined by the support vectors which satisfy:

$$t_i(w^T x_i - b) = 1 \Rightarrow b = w^T x_i - t_i$$

Where x_i is any support vector.

This completes the mathematical framework of the Support Vector Machine algorithm which allows for both linear and non-linear classification using the dual problem and kernel trick.

Radial basis function

Radial Basis Function (RBF) Neural Networks are used for function approximation tasks. They are a special category of feed-forward neural networks comprising of three layers. Due to this distinct three-layer architecture and universal approximation capabilities they offer faster learning speeds and efficient performance in classification and regression problems.

How Do RBF Networks Work?

RBF Networks are conceptually similar to K-Nearest Neighbor (k-NN) models though their implementation is distinct. The fundamental idea is that nearby items with similar predictor variable values influence an item's predicted target value. Here's how RBF Networks operate:

1. **Input Vector**: The network receives an n-dimensional input vector that needs classification or regression.
2. **RBF Neurons**: Each neuron in the hidden layer represents a prototype vector from the training set. The network computes the Euclidean distance between the input vector and each neuron's center.
3. **Activation Function**: The Euclidean distance is transformed using a Radial Basis Function (typically a Gaussian function) to compute the neuron's activation value. This value decreases exponentially as the distance increases.
4. **Output Nodes**: Each output node calculates a score based on a weighted sum of the activation values from all RBF neurons. For classification the category with the highest score is chosen.

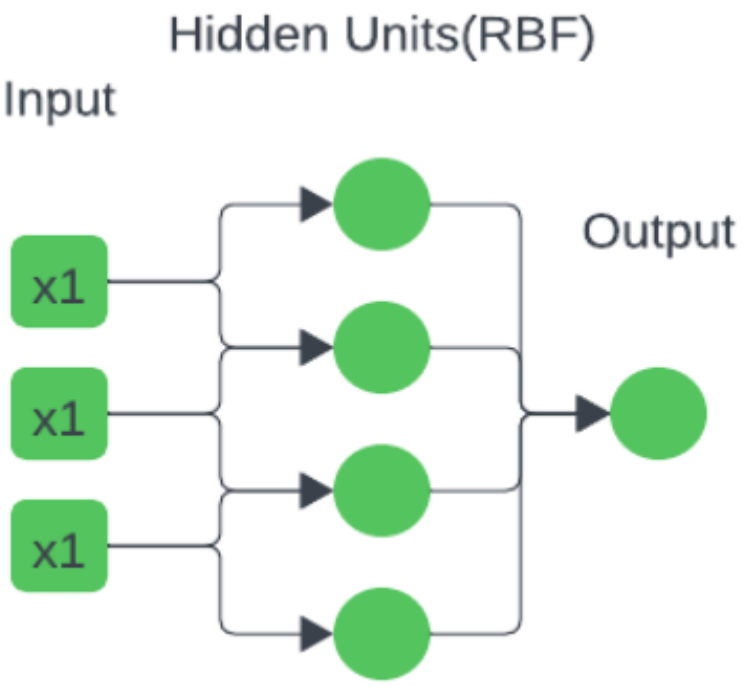
For example, consider a dataset with two-dimensional data points from two classes. An RBF Network trained with 20 neurons will have each neuron representing a prototype in the input space. The network computes category scores which can be visualized using 3-D mesh or contour plots. We assign positive weights to neurons in the same category and negative weights to neurons in different categories. The decision boundary can be plotted by evaluating scores over a grid.

Key Characteristics of RBFs

- **Radial Basis Functions**: These are real-valued functions dependent solely on the distance from a central point. The Gaussian function is the most commonly used type.
- **Dimensionality**: The network's dimensions correspond to the number of predictor variables.
- **Center and Radius**: Each RBF neuron has a center and a radius (spread). The radius affects how broadly each neuron influences the input space.

Architecture of RBF Networks

The architecture of an RBF Network typically consists of three layers:



Input Layer

- **Function:** After receiving the input features the input layer sends them straight to the hidden layer.
- **Components:** It is made up of the same number of neurons as the characteristics in the input data. One feature of the input vector corresponds to each neuron in the input layer.

Hidden Layer

- **Function:** This layer uses radial basis functions (RBFs) to conduct the non-linear transformation of the input data.
- **Components:** Neurons in the buried layer apply the RBF to the incoming data. The Gaussian function is the RBF that is most frequently utilized.
- **RBF Neurons:** Every neuron in the hidden layer has a spread parameter (σ) and a center which are also referred to as prototype vectors. The spread parameter modulates the distance between the center of an RBF neuron and the input vector which in turn determines the neuron's output.

Output Layer

- **Function:** The output layer uses weighted sums to integrate the hidden layer neurons outputs to create the network's final output.
- **Components:** It is made up of neurons that combine the outputs of the hidden layer in a linear fashion. To reduce the error between the network's predictions and the actual target values, the weights of these combinations are changed during training.

Evaluation Metrics

Evaluation metrics are quantitative measures used to evaluate the performance of a machine learning model. They compare the predicted outputs with the actual outputs and determine how accurately the model

performs classification. These metrics help in selecting the best model and improving its performance.

1. Confusion Matrix

A **Confusion Matrix** is a table that summarizes the performance of a classification model by comparing the actual class labels with the predicted class labels. It provides the number of correct and incorrect predictions made by the classifier.

Confusion Matrix Diagram

	Predicted Class	
	Positive	Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Components of Confusion Matrix

True Positive (TP)

True Positive represents the number of positive instances that are correctly classified as positive by the model.

True Negative (TN)

True Negative represents the number of negative instances that are correctly classified as negative by the model.

False Positive (FP)

False Positive represents the number of negative instances that are incorrectly classified as positive. It is also known as a **Type I Error**.

False Negative (FN)

False Negative represents the number of positive instances that are incorrectly classified as negative. It is also known as a **Type II Error**.

Example

Consider a disease prediction system that produces the following results:

	Predicted Positive	**Predicted Negative**
Actual Positive	45	5
Actual Negative	10	40

From the confusion matrix:

- **True Positive (TP) = 45**
- **True Negative (TN) = 40**
- **False Positive (FP) = 10**
- **False Negative (FN) = 5**

2. Accuracy

Accuracy is the ratio of correctly predicted instances to the total number of instances in the dataset. It measures the overall correctness of the classification model.

Formula

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Calculation

$$= \frac{45 + 40}{45 + 40 + 10 + 5} = \frac{85}{100} = 0.85$$

Accuracy = 85%

Advantages of Accuracy

- It is simple and easy to calculate.
- It provides an overall measure of model performance.
- It is suitable when the dataset is balanced.

Limitation

Accuracy can be misleading when the dataset contains imbalanced classes.

3. Precision

Precision is the ratio of correctly predicted positive instances to the total number of instances predicted as positive. It indicates how reliable the positive predictions are.

Formula

$$\text{Precision} = \frac{TP}{TP + FP}$$

Calculation

$$= \frac{45}{45 + 10} = \frac{45}{55} = 0.818$$

Precision = 81.8%

Advantages of Precision

- It reduces false positive predictions.
- It is useful when the cost of false positives is high.
- It improves the reliability of positive predictions.

Applications

- Spam email detection.
- Fraud detection.

- Medical testing where false alarms should be minimized.

4. Recall (Sensitivity)

Recall is the ratio of correctly predicted positive instances to the total number of actual positive instances. It measures the ability of the classifier to identify all positive samples.

Formula

$$\text{Recall} = \frac{TP}{TP + FN}$$

Calculation

$$= \frac{45}{45 + 5} = \frac{45}{50} = 0.90$$

Recall = 90%

Advantages of Recall

- It minimizes false negative predictions.
- It is useful when missing positive cases is costly.
- It measures the completeness of positive predictions.

Applications

- Disease diagnosis.
- Cancer detection.
- Security threat detection.

5. F1-Score

F1-Score is the harmonic mean of Precision and Recall. It provides a balanced evaluation of a classification model, especially when the dataset is imbalanced.

Formula

$$F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Calculation

$$= \frac{2 \times (0.818 \times 0.90)}{0.818 + 0.90}$$

$$= \frac{1.4724}{1.718} = 0.857$$

F1-Score = 85.7%

Advantages of F1-Score

- It balances Precision and Recall.
- It is suitable for imbalanced datasets.
- It provides a single performance measure.

Comparison of Evaluation Metrics

Metric	Purpose	Best Used When
Accuracy	Measures overall correctness	Dataset is balanced
Precision	Measures correctness of positive predictions	False positives are costly
Recall	Measures ability to identify positive cases	False negatives are costly
F1-Score	Balances Precision and Recall	Dataset is imbalanced

Applications of Evaluation Metrics

1. Evaluation metrics are used to assess the performance of machine learning classification models.
2. They are used in disease diagnosis to evaluate prediction accuracy.
3. They are used in spam email filtering systems.
4. They are used in fraud detection applications.
5. They are used in sentiment analysis and text classification.
6. They are used in image recognition and object detection.

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, “Machine Learning”, Wiley Publisher,2019.
2. SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, “Machine Learning”,Pearson,2019.
3. Alpaydin Ethem, “Introduction to Machine Learning”, 3rd Edition, PHI learningprivate limited, 2019

REFERENCE BOOKS:

1. “Machine Learning: A Probabilistic Perspective”, 2012, Kevin P. Murphy, MIT Press.
2. “Pattern Recognition and Machine Learning”, 2007, Christopher Bishop, Springer.
3. “Introduction to Machine Learning”, MIT Press, 3/e, 2014, Ethem Alpaydin.

Question Bank:

PART –A		
Q.No.	**Questions**	**Blooms Taxonomy Level**
1.	Define Classification in Machine Learning.	L1
2.	What is Logistic Regression?	L1
3.	Define the Logit Function.	L1
4.	What is Maximum Likelihood Estimation (MLE)?	L2
5.	Define Decision Tree.	L1
6.	List the steps involved in constructing a Decision Tree.	L2
7.	What is Information Gain?	L1
8.	Define Entropy in Decision Trees.	L1
9.	Mention any two issues in Decision Trees.	L2
10.	What is Ensemble Learning?	L1
11.	Define Random Forest.	L1
12.	What is Bayesian Classification?	L1
13.	Define Naïve Bayes Classifier.	L1
14.	What is the assumption of the Naïve Bayes algorithm?	L2
15.	Define K-Nearest Neighbor (KNN).	L1
16.	What is the role of the value of K in the KNN algorithm?	L2
17.	Define a Neuron in an Artificial Neural Network.	L1
18.	What is a Perceptron?	L1
19.	Define Multilayer Perceptron (MLP).	L1
20.	What is a Support Vector Machine (SVM)?	L1
21.	Define Optimal Hyperplane.	L2
22.	What is a Radial Basis Function (RBF)?	L1
23.	Define Accuracy.	L1
24.	What is a Confusion Matrix?	L2
PART –B		
1.	Explain the concept of Classification in Machine Learning with suitable examples.	L2
2.	Explain Logistic Regression and describe the steps involved in building a Logistic Regression model using the Logit Function.	L3
3.	Explain the concept of Maximum Likelihood Estimation (MLE) and its role in	L3

	Logistic Regression.	
4.	Explain the steps involved in constructing a Decision Tree with a suitable example.	L3
5.	Explain the process of Classification using Decision Trees and discuss the issues associated with Decision Trees.	L3
6.	Discuss Ensemble Learning and explain the Random Forest algorithm with its advantages and limitations.	L2
7.	Explain Bayesian Classification and describe the Naïve Bayes Classifier with suitable examples.	L2
8.	Explain the K-Nearest Neighbor (KNN) algorithm with suitable examples and applications.	L3
9.	Explain the architecture and working of Artificial Neural Networks, including the Perceptron and Multilayer Perceptron (MLP).	L2
10.	Explain Support Vector Machines (SVM), Linear SVM, Optimal Hyperplane, and Radial Basis Function (RBF) with suitable diagrams.	L3
11.	Compare Logistic Regression, Decision Tree, Naïve Bayes, KNN, and SVM based on their working principles, advantages, and limitations. L4 – Analyze	L4
12.	Explain the evaluation metrics Accuracy, Confusion Matrix, Precision, Recall, and F1 Score with suitable examples.	L2
13.	Compare Precision, Recall, and F1 Score, and discuss their importance in evaluating classification models.	L4
14.	Analyze the advantages and limitations of Decision Trees and Random Forest in classification problems.	L4
15.	Evaluate the performance of different classification algorithms using appropriate evaluation metrics.	L4

UNIT V

CLUSTERING

"Data is a precious thing and will last longer than the systems themselves." — Tim Berners-Lee

Unit Overview

This unit introduces Unsupervised Learning, where machine learning algorithms analyze unlabeled data to discover hidden patterns. It focuses on Clustering, including Partitioning, Hierarchical, Density-Based, and Grid-Based methods, along with K-Means Clustering and techniques for choosing the optimal number of clusters. These methods are widely used in data mining, healthcare, marketing, banking, and image processing.

Learning Outcomes

After completing this unit, students will be able to:

- Explain the concept of Unsupervised Learning.
- Describe the principles of clustering.
- Differentiate between various clustering methods.
- Apply the K-Means clustering algorithm.
- Choose the optimal number of clusters using appropriate techniques.
- Identify real-world applications of clustering.

Importance of Studying this Unit

- Understand hidden patterns in unlabeled data.
- Learn essential clustering techniques used in AI and Data Science.
- Build skills for customer segmentation, fraud detection, and recommendation systems.
- Develop a foundation for advanced machine learning concepts.

Key Concepts

Unsupervised Learning, Clustering, Partitioning Clustering, Hierarchical Clustering, Density-Based Clustering, Grid-Based Clustering, K-Means Clustering, Elbow Method, Silhouette Method, Cluster Evaluation

Introduction to Unsupervised Learning Algorithms

- Unsupervised learning is a branch of machine learning where the model is given data without any explicit labels or targets.
- Unlike supervised learning (where the model learns to map inputs to known outputs, like predicting housing prices or classifying emails as spam), unsupervised learning is all about discovery.
- The algorithm looks at the data, finds hidden structures, and groups or simplifies the information entirely on its own.

Types of unsupervised learning algorithms

Clustering Algorithms

Clustering is the process of grouping data points so that objects in the same group (a cluster) are more similar to each other than to those in other groups.

- **K-Means Clustering:** This algorithm partitions data into *K* distinct, non-overlapping clusters. It works by placing *K* centroids (center points) in the data space, assigning each data point to its nearest centroid, and then updating the centroids based on the average of the points assigned to them. This repeats until the centers stop moving.
- **Hierarchical Clustering**: Instead of picking a fixed number of clusters upfront, this method builds a tree-like hierarchy of clusters (a dendrogram). It can be *agglomerative* (starting with individual points and merging them into larger clusters) or *divisive* (starting with one massive cluster and splitting it down)
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** Unlike K-Means, which assumes clusters are roughly spherical, DBSCAN groups points based on how closely packed they are. It defines clusters as continuous regions of high density and is excellent at identifying outliners as "noise."

Dimensionality Reduction

- When dealing with datasets that have dozens or hundreds of features (high-dimensional data), models can become slow, inefficient, and prone to overfitting.
- Dimensionality reduction simplifies the data by compressing it into fewer variables while retaining as much critical information as possible.
- **Principal Component Analysis (PCA):** A linear transformation technique that identifies the axes (principal components) along which the variance of the data is maximized. By projecting the data onto these primary components, you can drop the dimensions that contribute very little information.

- **t-SNE (t-Distributed Stochastic Neighbor Embedding):** A non-linear technique primary used for visualizing high-dimensional data. It maps complex, multi-dimensional structures into a 2D or 3D space, keeping similar objects close together and dissimilar objects apart.

Association Rule Learning

This approach is used to discover interesting relationships or hidden rules between variables in large datasets. It looks for "if/then" patterns based on how frequently items appear together.

Apriori Algorithm: It operates on the principle that if an itemset is frequent, all of its subsets must also be frequent. It uses a bottom-up approach to identify relationships

Clustering-

Clustering is an unsupervised learning technique that groups similar data objects into clusters. Objects within the same cluster are more similar to each other than to objects in other clusters.

Clustering is widely used in **data mining, artificial intelligence, business analytics, healthcare, image processing, social network analysis, and recommendation systems**.

Types of Clustering

Clustering algorithms are classified into several categories

1. Partitioning Clustering

- Partitioning clustering divides the dataset into a predefined number (**K**) of non-overlapping clusters.
- Each data point belongs to exactly one cluster. The objective is to minimize the distance between data points within the same cluster while maximizing the distance between different clusters.
- Partitioning methods work well when the number of clusters is known in advance and the clusters have simple shapes.

Partitioning Clustering Algorithms

The two most popular partitioning algorithms are:

1.K-Means Clustering

2. K-Medoids

1.K-Means Clustering

- K-Means is the most widely used partitioning algorithm. It represents each cluster by its centroid (mean value).
- The algorithm minimizes the Within-Cluster Sum of Squares (WCSS), which measures how close the data points are to their cluster centroid.

Algorithm Steps

1. Select the number of clusters (K).
2. Randomly choose K centroids.
3. Calculate the distance between each data point and every centroid.
4. Assign each point to the nearest centroid.
5. Recalculate the centroid of each cluster.
6. Repeat until centroids no longer change.

Example

Suppose the ages of customers are:

22, 23, 25, 28, 60, 62, 65 If K = 2

Cluster 1[22, 23, 25, 28]

Centroid [(22+23+25+28)/4] = **24.5**

Cluster 2[60, 62, 65]

Centroid [(60+62+65)/3] = **62.3**

2. K-Medoids

K-Medoids is similar to K-Means, but instead of using the mean as the cluster center, it selects an **actual data object (medoid)** as the center.Because it uses real data points, it is more robust to outliers.

Algorithm Steps

1. Select K medoids.
2. Assign each object to the nearest medoid.
3. Replace medoids if a better representative point is found.
4. Repeat until no further improvement occurs.

Example

Customer ages:

20, 22, 24, 60, 62, 65

Possible medoids: 22 and 62

These actual data points serve as cluster centers.

2. Hierarchical Clustering

Hierarchical clustering creates a **tree-like hierarchy of clusters**, known as a **dendrogram**. Instead of producing only one partition, it provides multiple levels of clustering, allowing users to choose the desired level.

There are two approaches:

- Agglomerative (Bottom-Up)
- Divisive (Top-Down)

Agglomerative Hierarchical Clustering (Bottom-Up)

- Agglomerative clustering is the most commonly used hierarchical clustering method. It starts with each data point as an individual cluster.
- The algorithm repeatedly merges the two closest clusters until only one cluster remains or the desired number of clusters is obtained.
- This approach is called Bottom-Up because it starts from individual objects and gradually builds larger clusters.

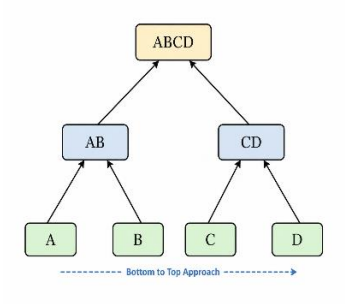
Algorithm Steps

- Treat every data point as a separate cluster.
- Calculate the distance between every pair of clusters. Distance can be calculated using:
    - Euclidean Distance
    - Manhattan Distance
    - Cosine Distance
- Merge the two closest clusters.
- Recalculate distances between the new cluster and remaining clusters.
- Continue merging until only one cluster remains.

Example: A B C D

Initially, {A} {B} {C} {D}

AB C D



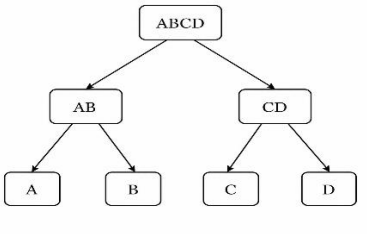
Divisive Hierarchical Clustering (Top-Down)

- Divisive clustering follows the Top-Down approach. It starts with all data points in one cluster and repeatedly divides them into smaller clusters until each object forms its own cluster or the desired number of clusters is reached.
- Although less commonly used than Agglomerative clustering, it often produces high-quality clusters.

Algorithm Steps

- Start with one cluster containing all objects.
- Split the cluster into two groups.
- Continue dividing the clusters.

Example: ABCD



3. Density-Based Clustering

- **Density-Based Clustering** is a clustering technique that forms clusters based on the density of data points in a region.
- Data points in high-density regions form clusters, while points in low-density regions are treated as noise or outliers.

Density-based clustering mainly depends on two parameters.

1. Epsilon (ε)
2. Minimum Points (MinPts)

Epsilon (ε)

Epsilon (ε) is the radius that defines the neighborhood around a data point.

- Small ε → Creates many small clusters.
- Large ε → Creates fewer, larger clusters.

Minimum Points (MinPts)

- **MinPts** specifies the **minimum number of neighboring points** required within ε distance to form a dense region.
- A point must have at least **5 neighboring points within radius 2** to become a core point.

Types of Data Points

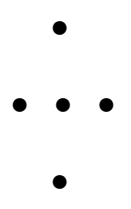
Density-based algorithms classify data points into three categories.

1. Core Point
2. Border Point
3. Noise Point (Outlier)

1. Core Point

A Core Point has at least MinPts neighboring points within ε distance.

Example:



The center point has enough neighboring points to form a cluster.

2. Border Point

A **Border Point** has fewer than MinPts neighbors but lies within the ε neighborhood of a Core Point. Border points belong to a cluster but do not expand it.

Core Points

● ● ● ●

○

Border Point

3. Noise Point (Outlier)

A **Noise Point** has very few or no neighboring points and does not belong to any cluster. Noise points are ignored during clustering.

×

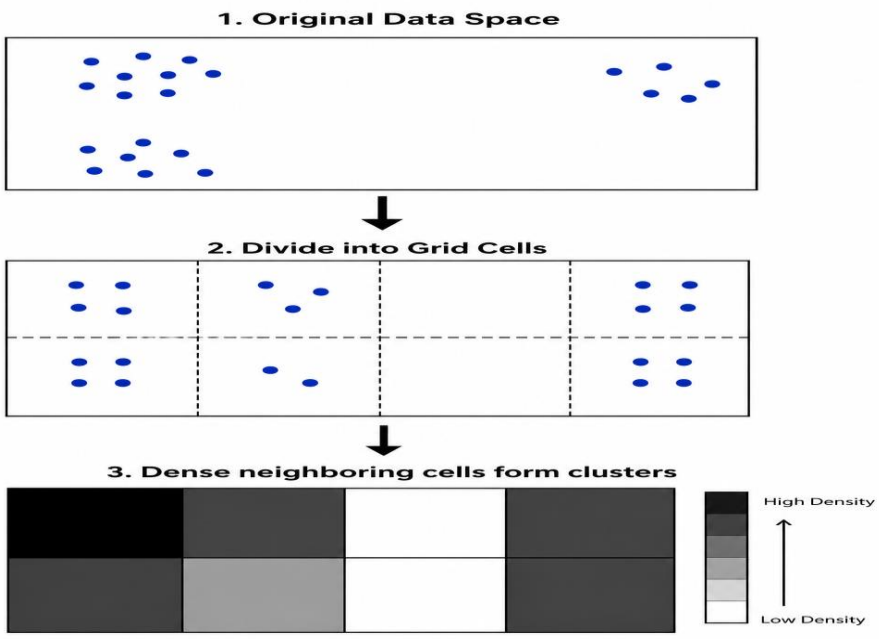
Noise Point

4. Grid-Based Clustering

Grid-based clustering divides the data space into a fixed number of cells called grids. Instead of processing individual data points, the algorithm processes these grid cells, making it much faster for large datasets.

The data space is divided into small rectangular cells (grids). Each grid cell contains a certain number of data points.

- Dense cells → Become part of clusters.
- Sparse cells → Considered empty or ignored.
- Neighboring dense cells are merged to form clusters.



Working Principle of Grid-Based Clustering

- Divide the data space into a fixed number of grid cells.
- Count the number of data points present in each grid cell.
- Calculate the density of each cell.
  - High-density cell → Dense Cell
  - Low-density cell → Sparse Cell
- Merge neighboring dense cells to form clusters.
- Ignore sparse or empty cells.
- Output the final clusters.

The most commonly used grid-based algorithms are:

1. STING
2. CLIQUE
3. WaveCluster

STING (Statistical Information Grid)

STING (Statistical Information Grid) is one of the earliest and most popular grid-based clustering algorithms. It divides the data space into a hierarchical grid structure where each grid cell stores statistical information about the data points it contains.

Working Steps

1. Divide the data space into rectangular grid cells.
2. Store statistical information for each cell.
3. Identify dense cells.
4. Merge neighboring dense cells.
5. Generate clusters.

CLIQUE (Clustering In QUEst)

CLIQUE is a grid-based clustering algorithm specially designed for high-dimensional datasets. Instead of considering the entire feature space, CLIQUE searches for dense regions in different subspaces of the data.

It is widely used in:

- Bioinformatics
- Text Mining
- Data Warehousing
- Scientific Data Analysis

Working Steps

1. Divide each dimension into equal-sized intervals.
2. Form multidimensional grid cells.
3. Identify dense cells.
4. Merge connected dense cells.
5. Generate clusters.

WaveCluster

WaveCluster is a grid-based clustering algorithm that applies Wavelet Transformation to the data space before clustering. Wavelet transformation smooths noisy data and highlights dense regions, making it easier to detect clusters.

It is capable of detecting:

- Irregular-shaped clusters

- Clusters of different sizes
- Noisy data

Working Steps

1. Divide the data into grid cells.
2. Apply wavelet transformation.
3. Identify dense regions.
4. Merge neighboring dense regions.
5. Generate clusters.

K-means Clustering- Choosing number of Clusters.

- One of the most important decisions in the K-Means Clustering algorithm is selecting the appropriate number of clusters (K).
- The value of K determines how the data points are grouped. If the chosen value of K is too small, different groups may be merged into one cluster.
- If K is too large, a single natural group may be divided into many smaller clusters.

What is K in K-Means

In **K-Means Clustering**, **K** represents the **number of clusters** into which the dataset is divided.

For example:

- **K = 2** → Dataset is divided into 2 clusters.
- **K = 3** → Dataset is divided into 3 clusters.
- **K = 5** → Dataset is divided into 5 clusters.

The user must specify the value of **K** before running the K-Means algorithm.

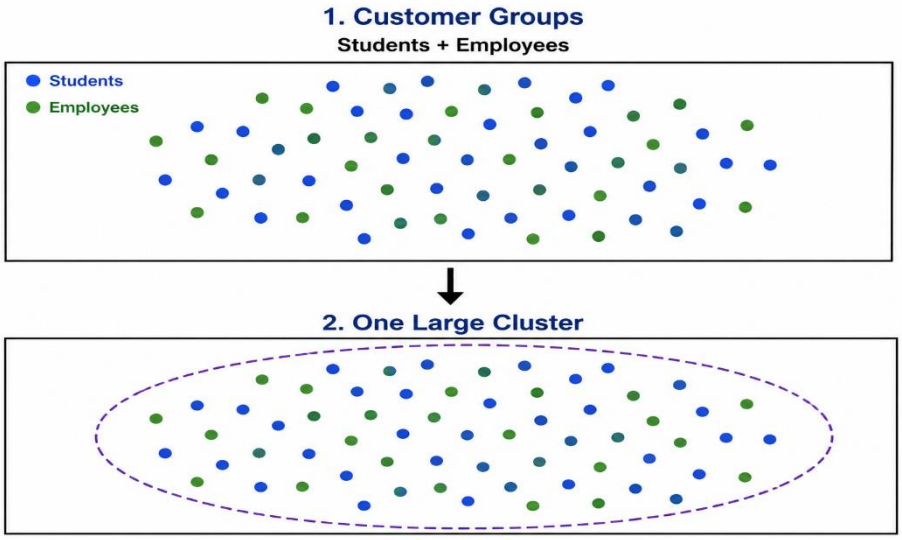
Choosing the Correct Number of Clusters

Selecting the correct value of K is essential because it directly affects clustering performance.

If K is Too Small

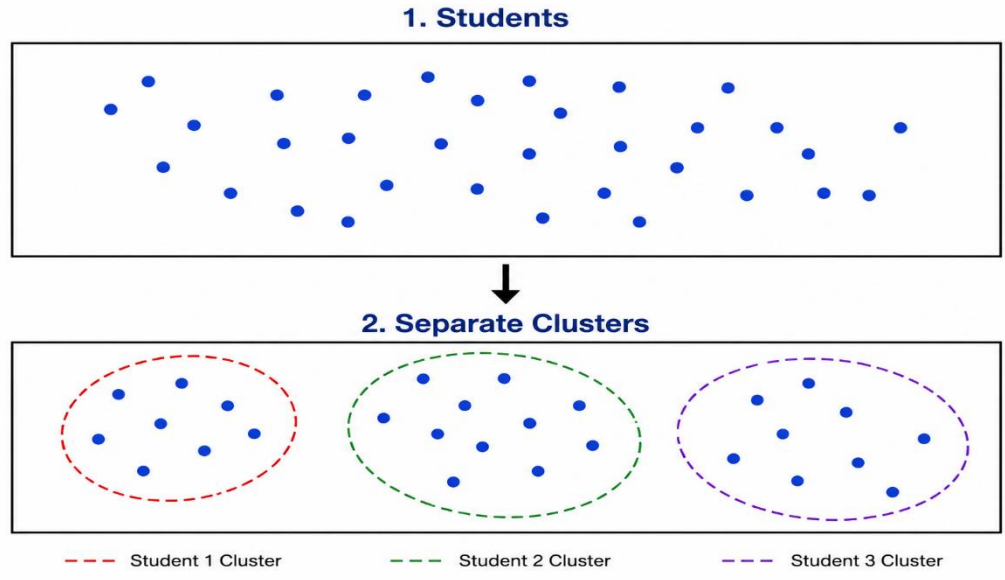
- Different groups are merged into one cluster.
- Important patterns may be lost.

- Clusters become too broad.



If K is Too Large

- A natural group is divided into many small clusters.
- Results become difficult to interpret.
- Overfitting may occur.



Methods for Choosing the Number of Clusters

Several techniques are available to determine the optimal value of K.

1. Elbow Method
2. Silhouette Method

3. Gap Statistic

Elbow Method

The Elbow Method is the most commonly used technique for selecting the optimal number of clusters in K-Means.

It calculates the Within-Cluster Sum of Squares (WCSS) for different values of K.

WCSS measures how closely the data points are grouped around their cluster centroid.

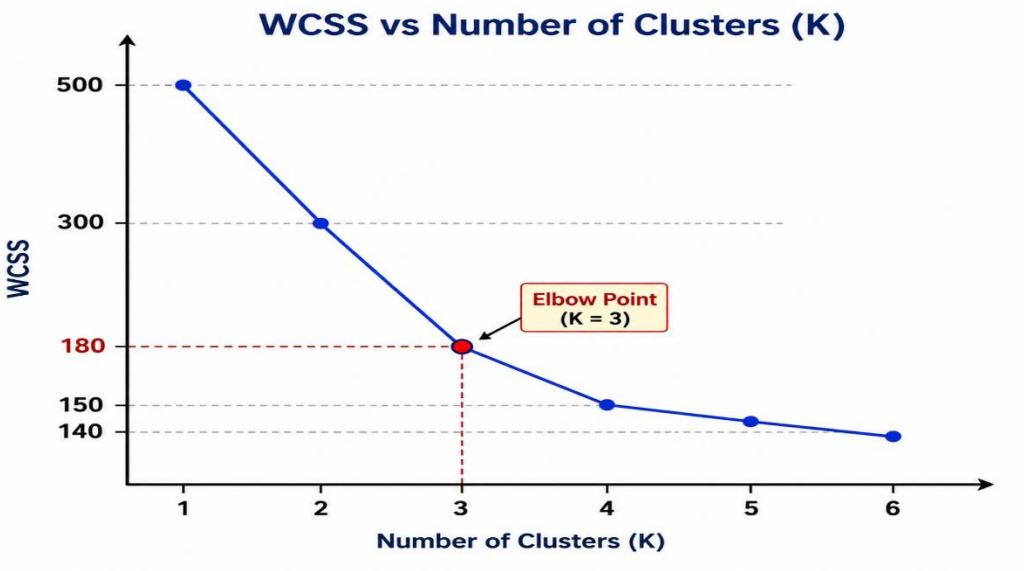
Working Steps

1. Run K-Means for different values of K.
2. Calculate the WCSS value for each K.
3. Plot K on the X-axis and WCSS on the Y-axis.
4. Identify the "Elbow Point", where the decrease in WCSS starts slowing down significantly

Example:

K = 1, 2, 3, 4, 5, 6, 7

K	WCSS
1	**500**
2	**300**
3	**180**
4	**150**
5	**140**
6	**135**



Silhouette Method

The Silhouette Method evaluates how well each data point fits within its own cluster compared to other clusters.

It measures:

- Cluster cohesion (how close points are within a cluster)
- Cluster separation (how distinct clusters are from one another)

The **Silhouette Score** ranges from **-1 to +1**.

Score	**Meaning**
+1	Excellent clustering
0	Overlapping clusters
-1	Incorrect clustering

Gap Statistic Method

The Gap Statistic compares the clustering result with randomly generated reference datasets. The optimal value of K is where the gap between the observed clustering and random clustering is the largest.

Unit Highlights

- Introduction to Unsupervised Learning

- Types of Clustering
- Partitioning Methods (K-Means, K-Medoids)
- Hierarchical Clustering
- Density-Based Methods (DBSCAN)
- Grid-Based Methods (STING, CLIQUE, WaveCluster)
- Choosing the Number of Clusters
- Real-world Applications of Clustering

TEXT BOOKS:

1. Anuradha Srinivasaraghavan and Vincy Joseph, “Machine Learning”, Wiley Publisher,2019.

2. SaikatDutt, Subramanian Chandramouli and Amit Kumar Das, “Machine Learning”,Pearson,2019.

3. Alpaydin Ethem, “Introduction to Machine Learning”, 3rd Edition, PHI learningprivate limited, 2019.

REFERENCE BOOKS:

1. “Machine Learning: A Probabilistic Perspective”, 2012, Kevin P. Murphy, MIT Press.

2. “Pattern Recognition and Machine Learning”, 2007, Christopher Bishop, Springer.

3. “Introduction to Machine Learning”, MIT Press, 3/e, 2014, Ethem Alpaydin.

Question Bank:

PART –A		
Q.No.	**Questions**	**Blooms Taxonomy Level**
1.	Define Unsupervised Learning.	L1
2.	What is Clustering?	L1
3.	List the different types of clustering.	L1
4.	Define Partitioning Clustering.	L1

5.	What is Hierarchical Clustering?	L1
6.	Define Density-Based Clustering.	L1
7.	What is DBSCAN?	L2
8.	Define Grid-Based Clustering.	L1
9.	What is K-Means Clustering?	L1
10.	What is the objective of the K-Means algorithm?	L2
11.	What is the Elbow Method?	L2
12.	Mention any two applications of Clustering.	L2

PART –B

1.	Explain the concept of Unsupervised Learning and discuss the importance of clustering in Machine Learning.	L2
2.	Explain the different types of clustering with suitable examples.	L2
3.	Describe the Partitioning Method of Clustering and explain the K-Means Clustering algorithm with a suitable example.	L3
4.	Explain Hierarchical Clustering and compare Agglomerative and Divisive clustering methods.	L4
5.	Explain Density-Based Clustering and discuss the DBSCAN algorithm with its advantages and limitations.	L3
6.	Describe Grid-Based Clustering methods and explain their advantages and applications.	L2
7.	Explain different methods used for choosing the optimal number of clusters, including the Elbow Method and Silhouette Score.	L3
8.	Compare Partitioning, Hierarchical, Density-Based, and Grid-Based clustering methods based on their characteristics, advantages, and limitations.	L4
9.	Analyze the advantages, limitations, and real-world applications of K-Means Clustering.	L4
10.	Evaluate the performance of clustering algorithms and justify the selection of an appropriate clustering method for a given dataset.	L5