UNIT II

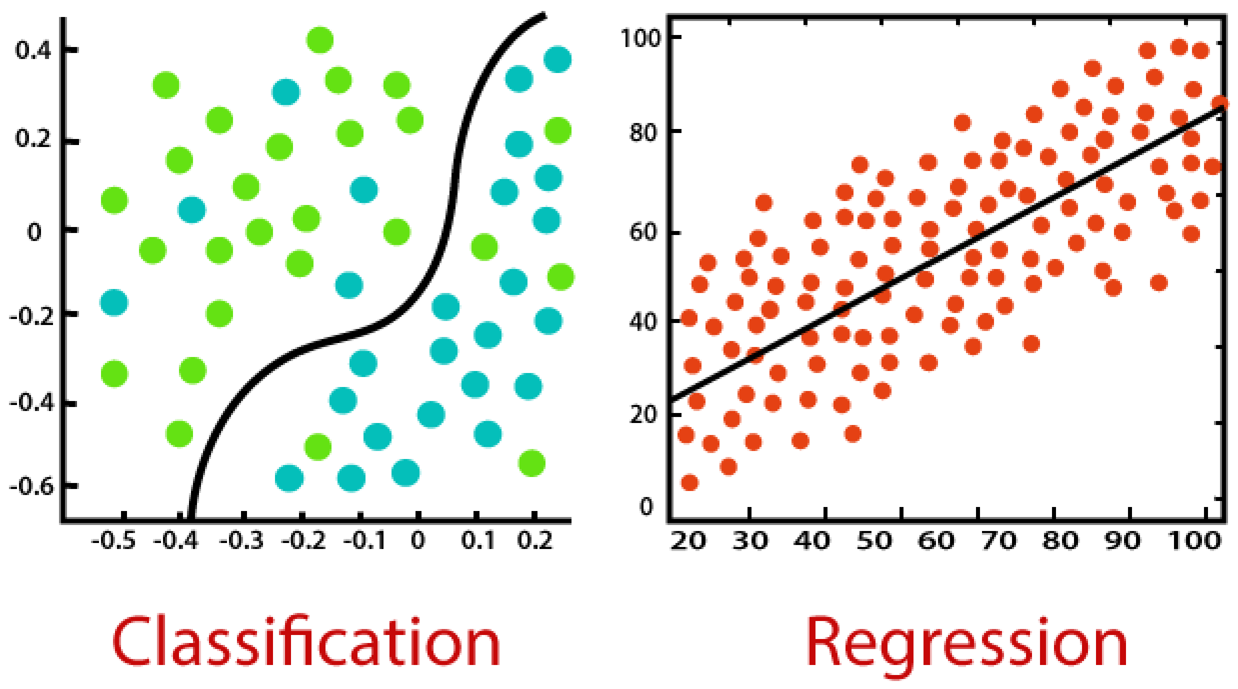
SUPERVISED LEARNING

2.1 Regression vs. Classification in Machine Learning

Regression and Classification algorithms are Supervised Learning algorithms. Both the algorithms are used for prediction in Machine learning and work with the labeled datasets. But the difference between both is how they are used for different machine learning problems.

The main difference between Regression and Classification algorithms that Regression algorithms are used to **predict the continuous** values such as price, salary, age, etc. and Classification algorithms are used to **predict/Classify the discrete values** such as Male or Female, True or False, Spam or Not Spam, etc.

Consider the below diagram:



Classification:

Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.

The task of the classification algorithm is to find the mapping function to map the input(x) to the discrete output(y).

Example: The best example to understand the Classification problem is Email Spam Detection. The model is trained on the basis of millions of emails on different parameters, and whenever it receives a new email, it identifies whether the email is spam or not. If the email is spam, then it is moved to the Spam folder.

Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the following types:

- Logistic Regression
- K-Nearest Neighbours
- Support Vector Machines
- Kernel SVM
- Naive Bayes
- Decision Tree Classification
- Random Forest Classification

Regression:

Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of **Market Trends**, prediction of House prices, etc.

The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).

Example: Suppose we want to do weather forecasting, so for this, we will use the Regression algorithm. In weather prediction, the model is trained on the past data, and once the training is completed, it can easily predict the weather for future days.

Types of Regression Algorithm:

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression

Difference between Regression and Classification

Regression Algorithm	**Classification Algorithm**
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input

	value(x) with the discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.
The regression Algorithm can be further divided into Linear and Non-linear Regression.	The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.

2.2 DISTANCE BASED METHODS

Distance-based methods in machine learning are algorithms that rely on measuring the distance between data points to make predictions or classify data. These methods are particularly useful for problems where the notion of similarity or proximity is meaningful. Common distance-based methods include:

1. K-Nearest Neighbors (KNN)
2. K-Means Clustering
3. Hierarchical Clustering
4. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
5. Support Vector Machines (SVM)

What is Euclidean Distance?

Euclidean distance is a measure of the straight-line distance between two points in Euclidean space. It is the most common and familiar distance metric, often referred to as the "ordinary" distance.
Euclidean Distance gives the distance between any two points in an n-dimensional plane. Euclidean distance between two points in the Euclidean space is defined as the length of the line segment joining the two points.

Euclidean distance is like **measuring the straightest and shortest path between two points**. Imagine you have a string and you stretch it tight between two points on a map; the length of that string is the Euclidean distance. It tells you how far apart the two points are without any turns or bends, just like a bird would fly directly from one spot to another. This **metric is based on the Pythagorean theorem** and is widely utilized in various fields such as machine learning, data analysis, computer vision, and more.

Euclidean Distance Formula

Consider two points (x1, y1) and (x2, y2) in a 2-dimensional space; the Euclidean Distance between them is given by using the formula:

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

Where,

- d is Euclidean Distance
- (x1, y1) is Coordinate of the first point
- (x2, y2) is Coordinate of the second point

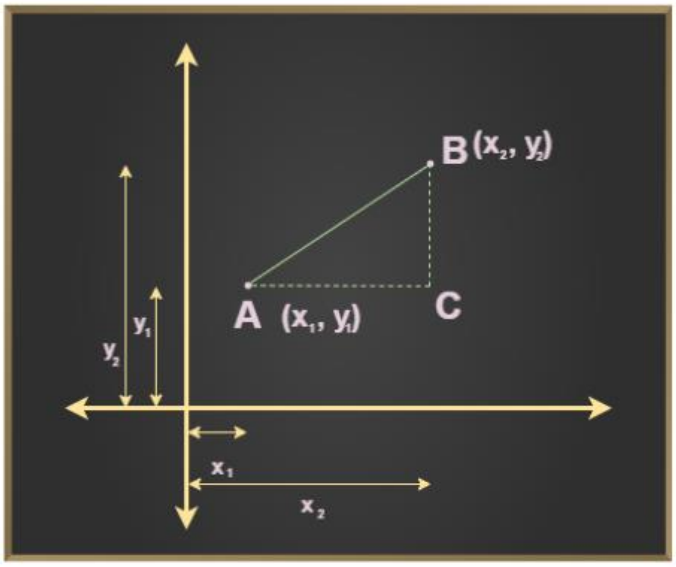
Euclidean Distance Formula Derivation

Euclidean Distance Formula is derived by following the steps added below:

Step 1: Let us consider two points, A (x1, y1) and B (x2, y2), and d is the distance between the two points.

Step 2: Join the points using a straight line (AB).

Step 3: Now, let us construct a right-angled triangle whose hypotenuse is AB, as shown in the figure below.



Step4: Now, using **Pythagoras theorem** we know that,

(Hypotenuse)2 = (Base)2 + (Perpendicular)2

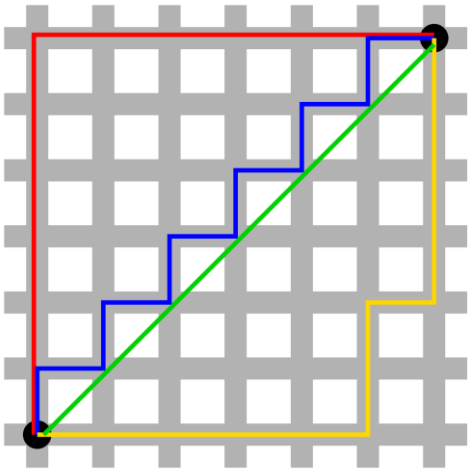
$$\Rightarrow d2 = (x2 - x1)^2 + (y2 - y1)^2$$

Now, take the square root on both sides of the equation, we get

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

2. Manhattan Distance:

This determines the absolute difference among the pair of the coordinates. Suppose we have two points P and Q to determine the distance between these points we simply have to calculate the perpendicular distance of the points from X-Axis and Y-Axis. In a plane with P at coordinate (x1, y1) and Q at (x2, y2). Manhattan distance between P and Q = |x1 – x2| + |y1 – y2|



Here the total distance of the **Red** line gives the Manhattan distance between both the points.

4. Minkowski distance:

It is the **generalized** form of the Euclidean and Manhattan Distance Measure. In an **N-dimensional space**, a point is represented as,

$(x_1, x_2, ..., x_N)$

Consider two points P1 and P2:

P1: $(X_1, X_2, ..., X_N)$
P2: $(Y_1, Y_2, ..., Y_N)$

Then, the Minkowski distance between P1 and P2 is given as:

$$\sqrt[p]{(x1-y1)^p + (x2-y2)^p + \ldots + (xN-yN)^p}$$

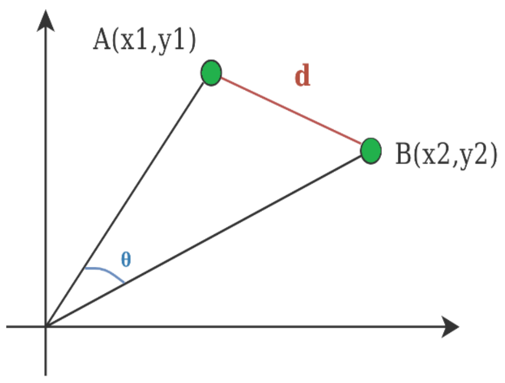
- When **p = 2**, Minkowski distance is same as the **Euclidean** distance.
- When **p = 1**, Minkowski distance is same as the **Manhattan** distance.

5. Cosine Index:

Cosine distance measure for clustering determines the **cosine** of the angle between two vectors given by the following formula.

$$\text{sim}(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

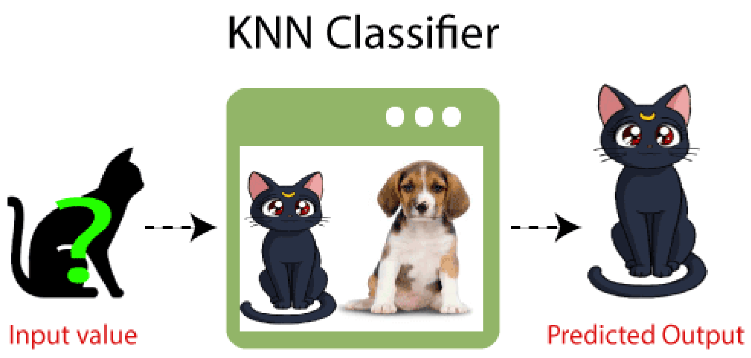
Here (**theta**) gives the angle between two vectors and A, B are n-dimensional vectors.



2.3 K-Nearest Neighbor(KNN) Algorithm

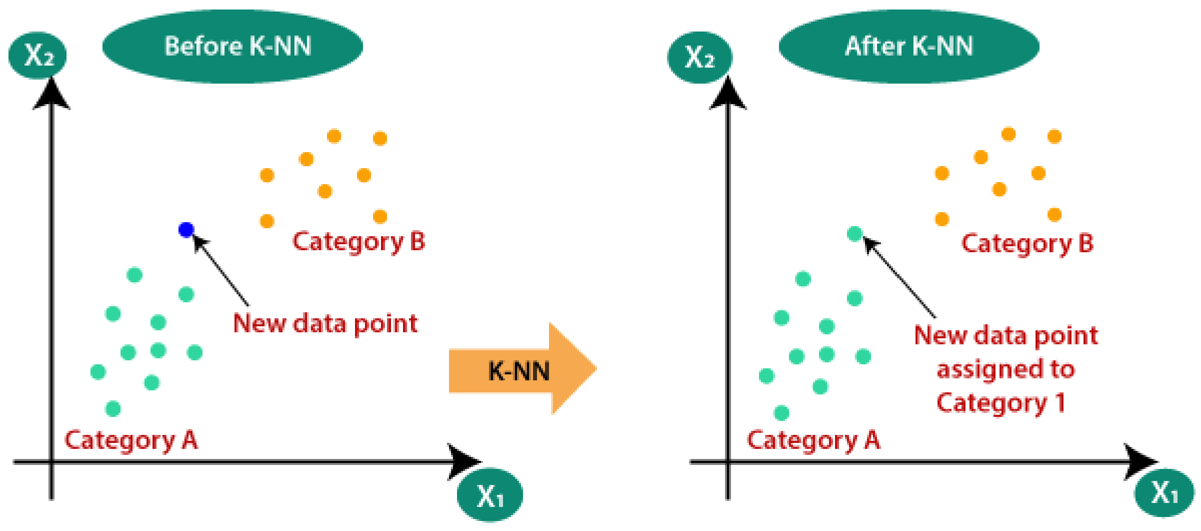
K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the **simplest Machine Learning algorithms based on Supervised Learning technique.**
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for **Regression as well as for Classification** but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

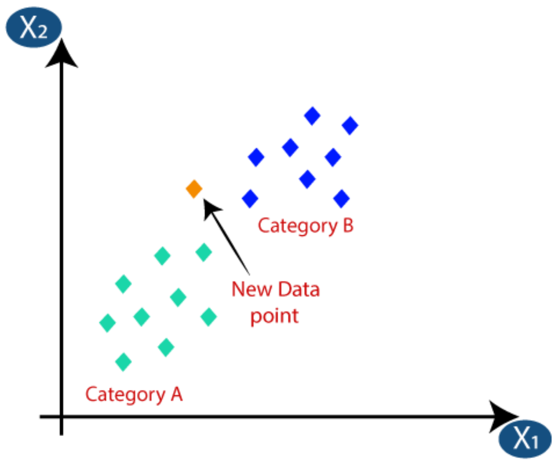


How does K-NN work?

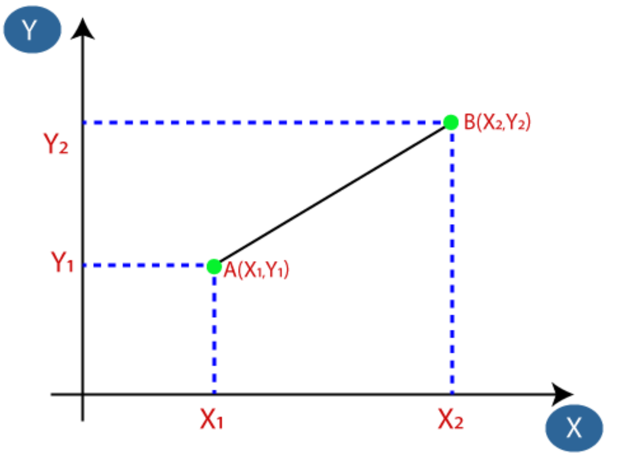
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

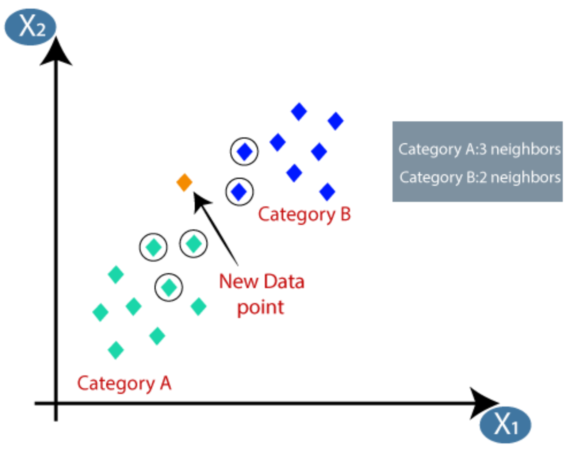


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

<u>How to select the value of K in the K-NN Algorithm?</u>

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.
- Rule of thumb is K < sqrt(n), n is number of examples.

<u>Advantages of KNN Algorithm:</u>

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

2.4 Decision Tree in Machine Learning

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction. The decision tree algorithm falls under the category of supervised learning. They can be used to solve both **regression** and **classification problems**.

Decision Tree Terminologies

There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure. :

- **Root Node:** A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- **Internal Nodes (Decision Nodes)**: Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
- **Leaf Nodes (Terminal Nodes)**: The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- **Branches (Edges)**: Links between nodes that show how decisions are made in response to particular circumstances.

- **Splitting**: The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- **Parent Node**: A node that is split into child nodes. The original node from which a split originates.
- **Child Node**: Nodes created as a result of a split from a parent node.
- **Decision Criterion**: The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- **Pruning**: The process of removing branches or nodes from a decision tree to improve its generalisation and prevent overfitting.

How Decision Tree is formed?

The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes. The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as information gain or Gini impurity. This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node.

Why Decision Tree?

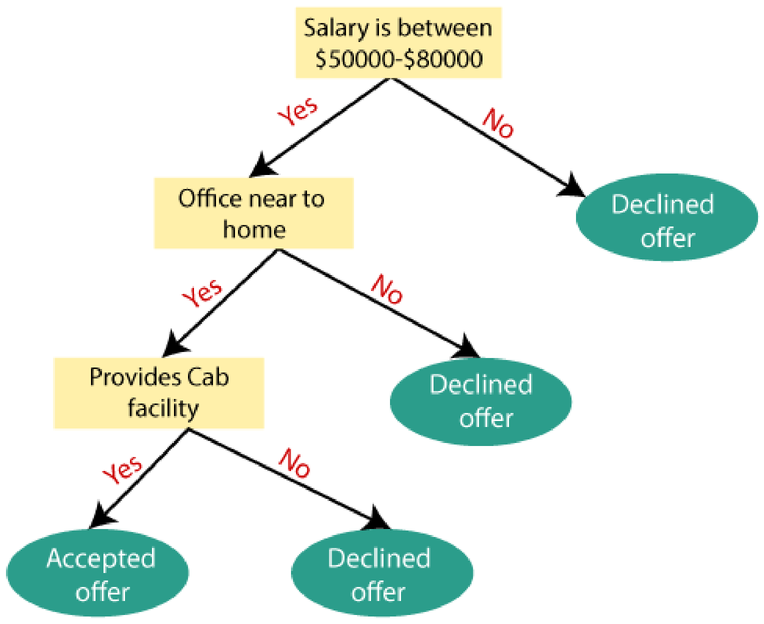
Decision trees are widely used in machine learning for a number of reasons:

- Decision trees are so versatile in simulating intricate decision-making processes, because of their interpretability and versatility.
- Their portrayal of complex choice scenarios that take into account a variety of causes and outcomes is made possible by their hierarchical structure.
- They provide comprehensible insights into the decision logic, decision trees are especially helpful for tasks involving categorisation and regression.
- They are proficient with both numerical and categorical data, and they can easily adapt to a variety of datasets thanks to their autonomous feature selection capability.
- Decision trees also provide simple visualization, which helps to comprehend and elucidate the underlying decision processes in a model.

Decision Tree Approach

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:
  - Suppose S is a set of instances,
  - A is an attribute
  - S_v is the subset of S
  - *v* represents an individual value that the attribute *A* can take and Values (A) is the set of all possible values of A, then

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

Entropy: is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|} . Entropy(S_v)$$

2. Gini Index:

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with a lower Gini index should be preferred.

- Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.
- The Formula for the calculation of the Gini Index is given below.

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

The Gini Index is a measure of the inequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithms. It ranges from 0 to 0.5, where 0 indicates a pure set (all instances belong to the same class), and 0.5 indicates a maximally impure set (instances are evenly distributed across classes).

2.4 NAÏVE BAYES CLASSIFIERS

Naive Bayes classifiers are supervised machine learning algorithms used for classification tasks, based on Bayes’ Theorem to find probabilities.

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.
- Naïve Bayes Algorithm is used in spam filtration, Sentimental analysis, classifying articles and many more.

Why it is Called Naive Bayes?

It is named as “Naive” because it assumes the presence of one feature does not affect other features. The “Bayes” part of the name refers to for the basis in Bayes’ Theorem.
Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where A and B are events and P(B) ≠ 0

Where,

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

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.$X = (x_1, x_2, x_3, \ldots.., x_n)$

2.5 LINEAR REGRESSION

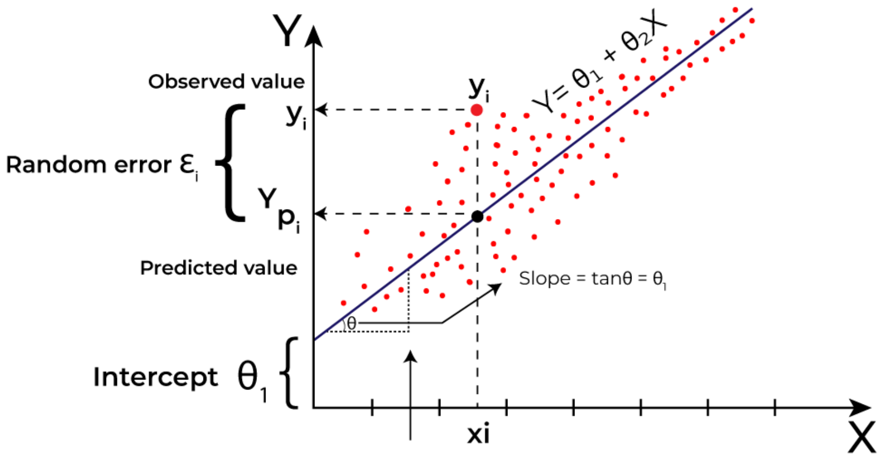
Linear regression is a statistical method used to **model the relationship between a dependent variable and one or more independent variables.** It provides valuable insights for prediction and data analysis.

Linear regression is also a type of supervised machine-learning algorithm that **learns from the labelled datasets and maps the data points with most optimized linear functions** which can be used for prediction on new datasets. It computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation with observed data. It predicts the continuous output variables based on the independent input variable.

For example if we want to predict house price we consider various factor such as house age, distance from the main road, location, area and number of room, linear regression uses all these parameter to predict house price as it consider a linear relation between all these features and price of house.

What is the best Fit Line?

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.
The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).


Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
y_i
$Y = \theta_1 + \theta_2 X$
Slope = tanθ = θ_1
θ
Intercept θ_1
xi
X

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

Simple Linear Regression

Simple linear regression is the simplest form of linear regression and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:

$$y=\beta 0+\beta 1X$$

where:

- Y is the dependent variable
- X is the independent variable
- β0 is the intercept
- β1 is the slope

Multiple Linear Regression

Multiple linear regression involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:
y=β0+β1X1+β2X2+………βnXn

where:
- Y is the dependent variable
- X1, X2, …, Xn are the independent variables
- β0 is the intercept
- β1, β2, …, βn 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.

Advantages and Disadvantages of Linear Regression

Advantages of Linear Regression

- Linear regression is a relatively simple algorithm, making it easy to understand and implement. The coefficients of the linear regression model can be interpreted as the change in the dependent variable for a one-unit change in the independent variable, providing insights into the relationships between variables.
- Linear regression is computationally efficient and can handle large datasets effectively. It can be trained quickly on large datasets, making it suitable for real-time applications.
- Linear regression is relatively robust to outliers compared to other machine learning algorithms. Outliers may have a smaller impact on the overall model performance.
- Linear regression often serves as a good baseline model for comparison with more complex machine learning algorithms.
- Linear regression is a well-established algorithm with a rich history and is widely available in various machine learning libraries and software packages.

Disadvantages of Linear Regression

- Linear regression assumes a linear relationship between the dependent and independent variables. If the relationship is not linear, the model may not perform well.
- Linear regression is sensitive to multicollinearity, which occurs when there is a high correlation between independent variables. Multicollinearity can inflate the variance of the coefficients and lead to unstable model predictions.
- Linear regression assumes that the features are already in a suitable form for the model. Feature engineering may be required to transform features into a format that can be effectively used by the model.
- Linear regression is susceptible to both overfitting and underfitting. Overfitting occurs when the model learns the training data too well and fails to generalize to unseen data. Underfitting occurs when the model is too simple to capture the underlying relationships in the data.

2.6 LOGISTIC REGRESSION

- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

How it works?

The logistic regression model transforms the linear regression function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Coefficient, and b is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

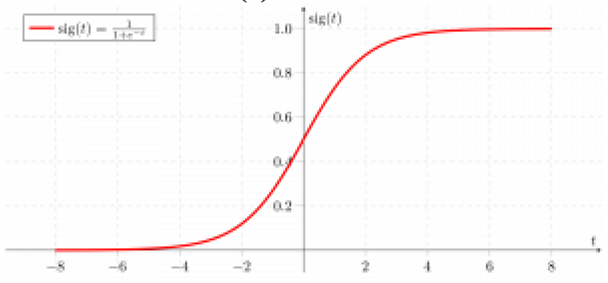
whatever we discussed above is the linear regression.

Sigmoid Function

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.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

σ(z)=1+e−z1



Sigmoid function

As shown above, the figure sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

- σ(z) tends towards 1 as z→∞
- σ(z) tends towards 0 as z→−∞
- σ(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 Equation

The odd is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur. so odd will be:

$$\frac{p(x)}{1-p(x)} = e^z$$

Applying natural log on odd. then log odd will be:

$$\begin{aligned} \log\left[\frac{p(x)}{1-p(x)}\right] &= z \\ \log\left[\frac{p(x)}{1-p(x)}\right] &= w \cdot X + b \\ \frac{p(x)}{1-p(x)} &= e^{w \cdot X+b} \quad \cdots \text{Exponentiate both sides} \\ p(x) &= e^{w \cdot X+b} \cdot (1-p(x)) \\ p(x) &= e^{w \cdot X+b} - e^{w \cdot X+b} \cdot p(x)) \\ p(x) + e^{w \cdot X+b} \cdot p(x)) &= e^{w \cdot X+b} \\ p(x)(1+e^{w \cdot X+b}) &= e^{w \cdot X+b} \\ p(x) &= \frac{e^{w \cdot X+b}}{1+e^{w \cdot X+b}} \end{aligned}$$

then the final logistic regression equation will be:

$$p(X; b, w) = \frac{e^{w \cdot X+b}}{1+e^{w \cdot X+b}} = \frac{1}{1+e^{-w \cdot X+b}}$$

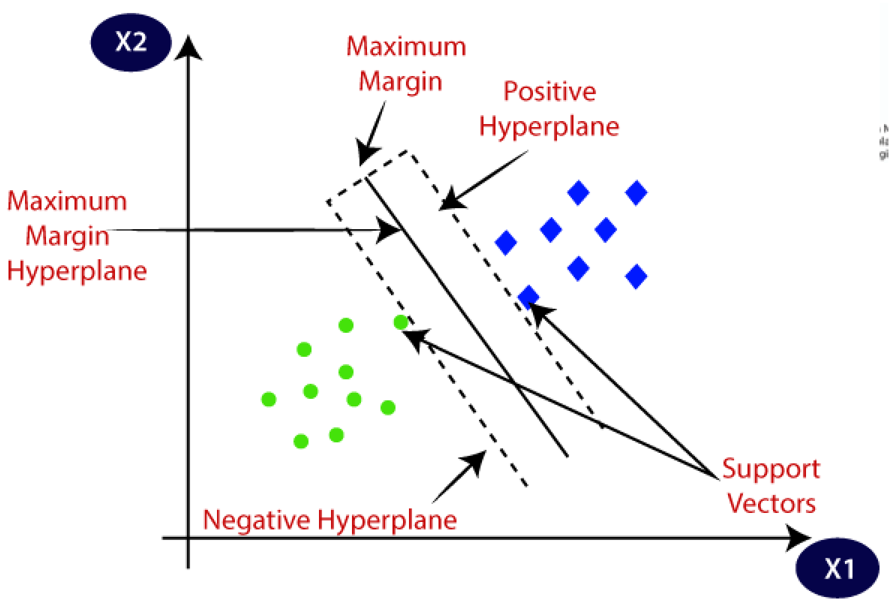
Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear regression is used for solving regression problem.	It is used for solving classification problems.
In this we predict the value of continuous variables	In this we predict values of categorical variables
In this we find best fit line.	In this we find S-Curve.
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for Estimation of accuracy.
The output must be continuous value, such as price, age, etc.	Output must be categorical value such as 0 or 1, Yes or no, etc.
It required linear relationship between dependent and independent variables.	It not required linear relationship.
There may be collinearity between the independent variables.	There should be little to no collinearity between independent variables.

2.7 Support Vector Machines

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

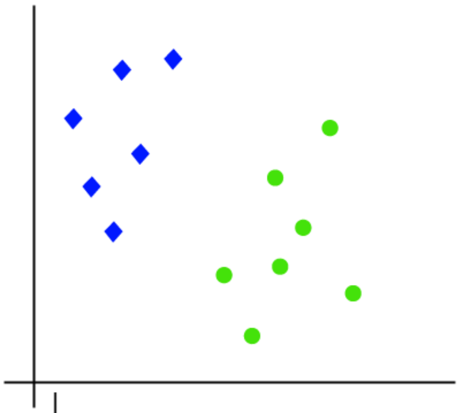
Support vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

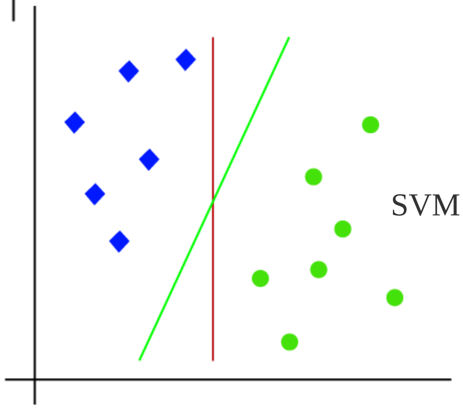
How does SVM works?

Linear SVM:

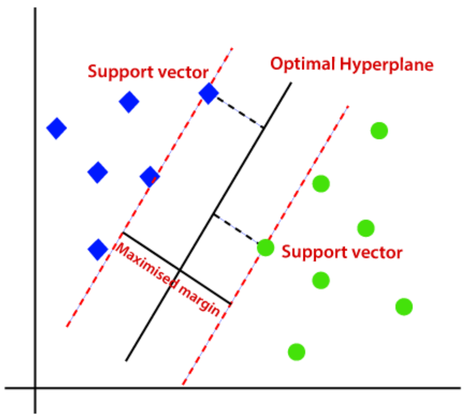
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

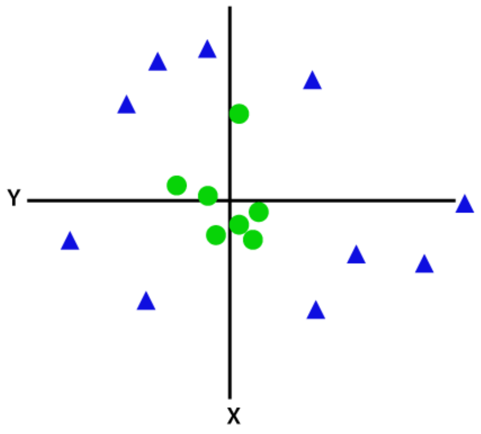


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

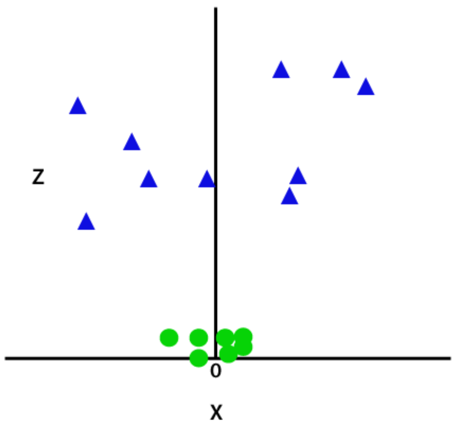
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



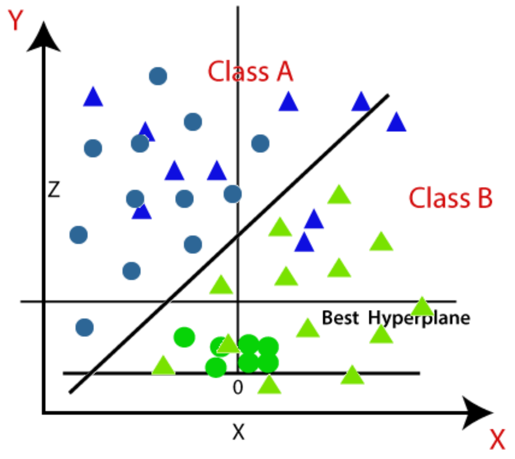
To separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

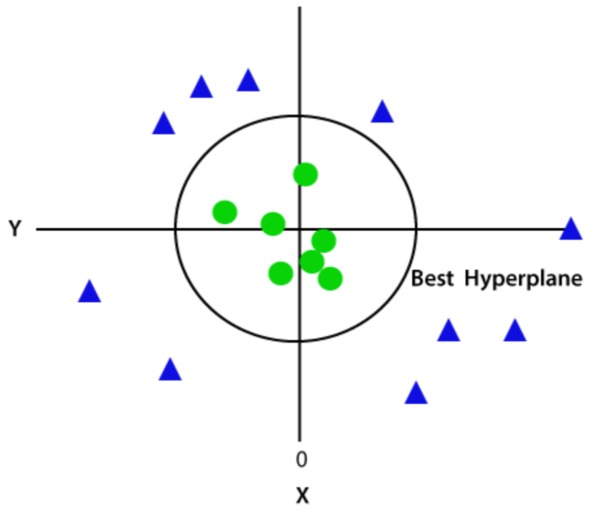
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

2.8 BINARY CLASSIFICATION(Refer PPT)

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

In classification algorithm, a discrete output function(y) is mapped to input variable(x).

1. y=f(x), where y = categorical output

The best example of an ML classification algorithm is **Email Spam Detector**.

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.

The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
  Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
  Example: Classifications of types of crops, Classification of types of music.

It is a type of supervised learning, a method of machine learning where the categories are predefined, and is used to categorize new probabilistic observations into said categories

2.9 MULTICLASS CLASSIFICATION (Refer PPT)

Multiclass classification is a machine learning challenge focused on categorizing data into more than two classes. While binary classification involves distinguishing between only two classes, multiclass classification expands this scope to involve distinguishing between multiple classes. In essence, the goal is to train a model that can effectively sort instances into various predefined categories, providing a nuanced solution for scenarios where items can belong to more than two exclusive groups. This approach is commonly employed in tasks such as handwriting recognition, email categorization, and image classification involving more than two distinct categories.

Binary classification are those tasks where examples are assigned exactly one of two classes. Multi-class classification is those tasks where examples are assigned exactly one of more than two classes.

- **Binary Classification**: Classification tasks with two classes.
  Multi-class Classification: Classification tasks with more than two classes.

Two examples of these heuristic methods include:

- One-vs-Rest (OvR)
- One-vs-One (OvO)

One-Vs-Rest for Multi-Class Classification

One-vs-rest (OvR for short, also referred to as One-vs-All or OvA) is a heuristic method for using binary classification algorithms for multi-class classification.

The obvious approach is to use a one-versus-the-rest approach (also called one-vs-all), in which we train C binary classifiers, fc(x), where the data from class c is treated as positive, and the data from all the other classes is treated as negative.

It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

For example, given a multi-class classification problem with examples for each class '*red*,' '*blue*,' and '*green*'. This could be divided into three binary classification datasets as follows:

- **Binary Classification Problem 1**: red vs [blue, green]
- **Binary Classification Problem 2**: blue vs [red, green]
- **Binary Classification Problem 3**: green vs [red, blue]
-

One-Vs-One for Multi-Class Classification

One-vs-One (OvO for short) is another heuristic method for using binary classification algorithms for multi-class classification.

Like one-vs-rest, one-vs-one splits a multi-class classification dataset into binary classification problems. Unlike one-vs-rest that splits it into one binary dataset for each class, the one-vs-one approach splits the dataset into one dataset for each class versus every other class.

For example, consider a multi-class classification problem with four classes: '*red*,' '*blue*,' and '*green*,' '*yellow*.' This could be divided into six binary classification datasets as follows:

- **Binary Classification Problem 1**: red vs. blue
- **Binary Classification Problem 2**: red vs. green
- **Binary Classification Problem 3**: red vs. yellow
- **Binary Classification Problem 4**: blue vs. green
- **Binary Classification Problem 5**: blue vs. yellow
- **Binary Classification Problem 6**: green vs. yellow

2.10 MNIST

2.11 Ranking

The process of placing objects, entities, or pieces in a certain order to represent their relative significance, importance, or worth in a particular context is known as ranking. Ranking in the context of computer learning and knowledge retrieval is giving objects scores and putting them in either ascending or descending order according to these ratings.

Important Elements of Ranking

- **Items:** The things or things that need to be rated. These might include resumes, job seekers, items, movies, webpages, and papers.
- **Criteria:** The qualities or characteristics that serve as the foundation for comparing and evaluating the objects. These could include user ratings, popularity, and relevancy to a query.
- **Scoring Function:** A mathematical framework or method known as the "scoring function" is used to give scores to things according to the criteria. Each item's relative relevance or significance is indicated by its score.
- **Order:** The organisation of the elements according to their scores; in most cases, this is done in descending order of significance or relevance.

Machine Learning Ranking

In machine learning, the term "ranking" commonly refers back to the manner of extracting a scoring approach from statistics using algorithms. To do this, a model have to be prepared to determine a element's pertinence or importance in a given placing. New gadgets can then be located in view of the found out model.

Different Ranking Methodologies

- **Algorithms for Pointwise Ranking**

Definition: Pointwise ranking methods method the ranking trouble as a set of wonderful classification or regression jobs. Every object is given a unique score based totally on its features.

Examples encompass logistic regression, which treats ranking as a binary class problem, gradient-boosting machine (GBM) such as XGBoost and LightGBM, and regression using linear models, which is often used to are expecting continuum relevance rankings.

Benefits: These algorithms are simple to use and employ attempted-and-authentic category and regression techniques.
Cons: One principal disadvantage is that those rankings won't be most fulfilling as they do not account for the items' relative positions.

- **Methods for Pairwise Ranking**

Pairwise techniques for ranking are defined as following: they compare two items to determine their relative rankings. The goal is to determine which element in each combination is more crucial or vital.

Notable examples are support vector machine rankings (SVMrank), which uses neural network models for pairwise comparisons, and LambdaRank, a form of RankNet improved with gradient boosting.

Rank Ordering Algorithm Applications

- **Online search engines:** Because they arrange search results according to how relevant they are to user queries, ranking algorithms are essential to search engines. To efficiently rank web pages, these algorithms take into account a number of variables, including user engagement metrics, website authority, and keyword relevancy. For example, Google's PageRank algorithm determines a page's authority and relevancy based on the amount and quality of links connecting to it, which affects the page's ranking in search results.
- **Systems for Recommenders:** Personalised suggestions are given to users by recommender systems using ranking algorithms, which are based on their behaviour and preferences. Through the examination of user behaviour, including previous purchases and ratings, these algorithms pinpoint products that are likely to catch the attention of certain users. Ranking algorithms are used by sites including Amazon and Netflix to make personalised product and movie recommendations based on customer preferences, increasing user happiness and engagement.
- **Online shopping:** Ranking algorithms are used by e-commerce platforms to arrange product listings in suggestion widgets and search results. These algorithms decide the sequence in which goods are shown to customers based on things like product popularity, relevancy, reviews from users, and past purchases. Ranking algorithms help internet businesses enhance sales and conversion rates by displaying products that are likely appealing to customers.
- **Internet Promotion:** Online advertising systems rely on ranking algorithms to decide where and when to display adverts. In order to prioritise adverts in search engine results and display networks, ad ranking algorithms consider many parameters, including ad relevancy, bid amount, clicking

through rate, and ad quality. Advanced ranking algorithms are used by Facebook advertisements and Google's AdWords to display advertisements that are likely to result in clicks and conversions, maximising ad income.

- **Social Networking:** Social media companies employ ranking algorithms to sort and order information in users' news feeds according to engagement and relevancy criteria. These algorithms decide the order in which articles are presented by analysing many criteria like the recency of the post, interactions between users (likes, comments, shares), among user preferences. Ranking algorithms improve user retention and involvement on social media sites like Twitter, Instagram, and Facebook by presenting material that is customised to each individual's interests.
- **Information Extraction:** For effective information retrieval in a variety of contexts, such as corporate search, storage of documents, and scholarly research, ranking algorithms are crucial. These algorithms rank search results according to several relevance signals, document quality, and user query relevancy. Ranking algorithms are used by platforms such as Google Scholar and business search engines to assist users in finding pertinent articles, documents, and other resources fast.

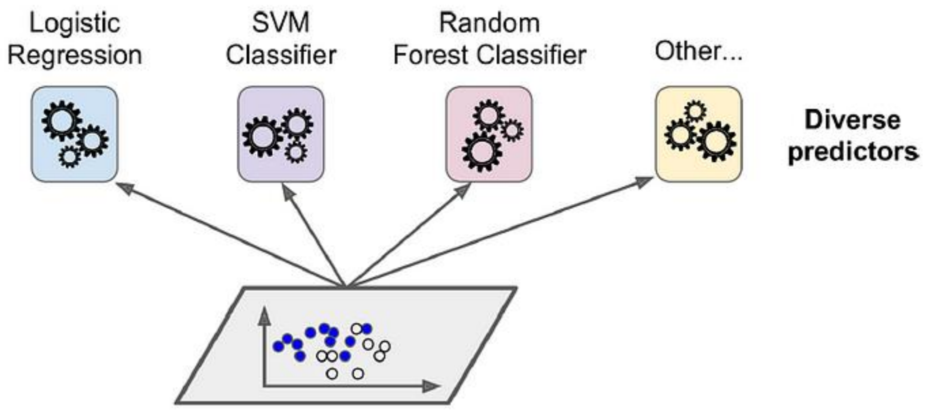
UNIT 3

Ensemble Learning

3.1 ENSEMBLE LEARNING

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

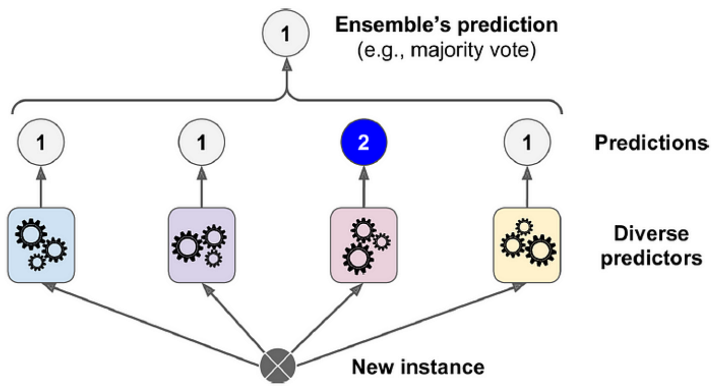
Ensemble learning combines the predictions of multiple models (called "weak learners" or "base models") to make a stronger, more reliable prediction. The goal is to reduce errors and improve performance. The grouping of individual predictors (weak learners) having low accuracy results in **ensemble** (strong learners) achieving high accuracy with reduced variance provided that there are sufficient number of weak learners and they are sufficiently diverse(less correlated).



The most popular Ensemble methods are:

1. Bagging (Bootstrap Aggregation)
2. Boosting Technique
3. Stacking

Hard Voting Classifiers: *It is a majority vote classifier to predict the class labels*. The prediction of each output will be aggregated and the class which got most votes will be the final predicted output. For image shown below, majority of classifiers are predicting class 1 and one classifier is predicting class 2 so according to majority vote classifier class 1 will be the final predicted output.



Soft Voting Classifiers: It measures average predicted probabilities (soft vote). It will predict the output class with highest class probability which is averaged over all the individual classifiers. For below example the individual models are predicting probabilities for class 1 and class 2 and the most of the models have highest probability of class 1 and hence the average of all highest probabilities will be 0.65 for class 1. Sometimes it assigns weighs to each classifier by weights parameter. Weights will be directly proportional to accuracy of individual classifiers.

Model	Class 1 Probability	Class 2 Probability
Model 1	0.6	0.4
Model 2	0.65	0.35
Model 3	0.8	0.2
Model 4	0.7	0.3
Model 5	0.5	0.5
Avg. Output	**0.65**	**0.35**

Types of Ensemble Learning in Machine Learning

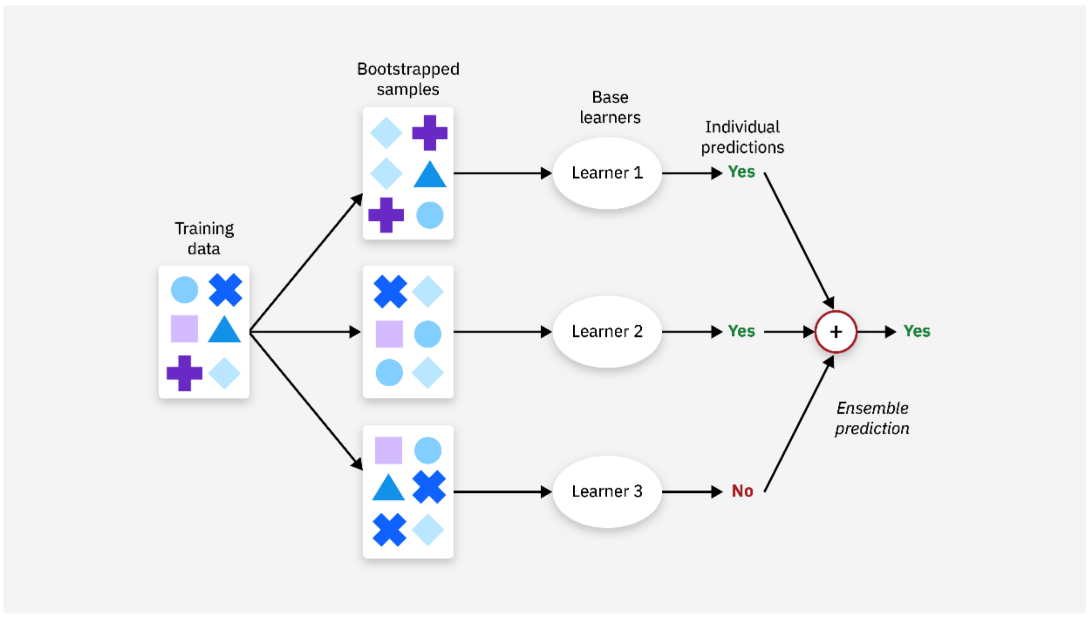
There are two main types of ensemble methods:

1. **Bagging (Bootstrap Aggregating):** Models are trained independently on different subsets of the data, and their results are averaged or voted on.
2. **Boosting:** Models are trained sequentially, with each one learning from the mistakes of the previous model.

3.2. BAGGING ALGORITHM

Bagging is a homogenous parallel method sometimes called *bootstrap aggregating*. It uses modified replicates of a given training data set to train multiple base learners with the same training algorithm. Bagging classifier can be used for both regression and classification tasks.

More specifically, bagging uses a technique called bootstrap resampling to derive multiple new datasets from one initial training dataset in order to train multiple base learners. A training dataset contains *n* training examples. Bootstrap resampling copies *n* data instances from that set into a new subsample dataset, with some initial instances appearing more than once and others excluded entirely. These are bootstrap samples. Repeating this process *x* times produces *x* iterations of the original dataset, each containing *n* samples from the initial set. Each iteration of the initial set is then used to train a separate base learner with the same learning algorithm.



Random forest is an extension of bagging that specifically denotes the use of bagging to construct ensembles of randomized decision trees. This differs from standard decision trees in that the latter samples every feature to identify the best for splitting. By contrast, random forests iteratively sample random subsets of features to create a decision node

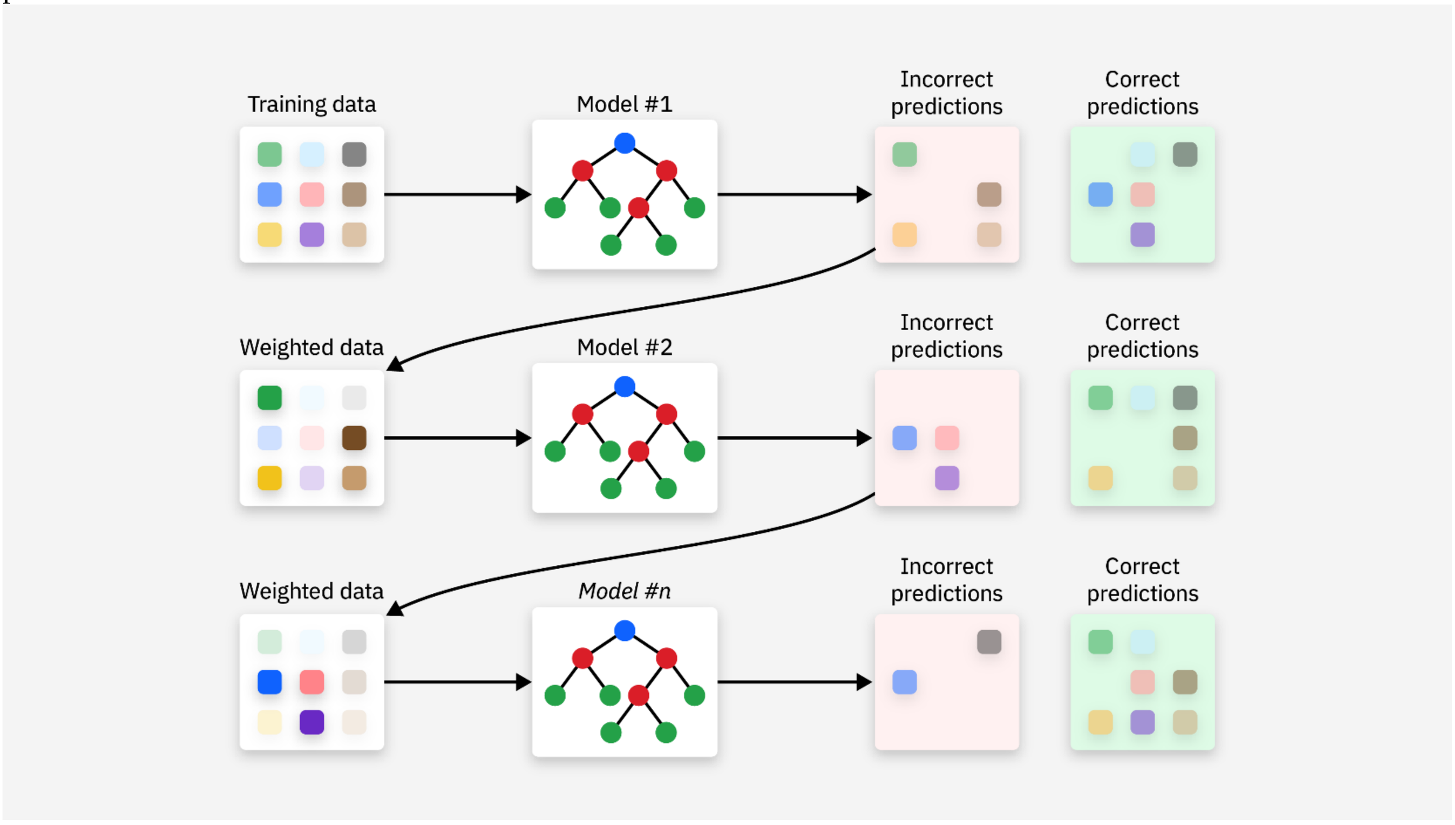
Bagging classifier algorithm:

- **Bootstrap Sampling:** Divides the original training data into 'N' subsets and randomly selects a subset with replacement in some rows from other subsets. This step ensures that the base models are trained on diverse subsets of the data and there is no class imbalance.
- **Base Model Training**: For each bootstrapped sample we train a base model independently on that subset of data. These weak models are trained in parallel to increase computational efficiency and reduce time consumption. **We can use different base learners i.e different ML models as base learners to bring variety and robustness.**
- **Prediction Aggregation:** To make a prediction on testing data combine the predictions of all base models. For classification tasks it can include majority voting or weighted majority while for regression it involves averaging the predictions.
- **Out-of-Bag (OOB) Evaluation**: Some samples are excluded from the training subset of particular base models during the bootstrapping method. These "out-of-bag" samples can be used to estimate the model's performance without the need for cross-validation.
- **Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

3.3. BOOSTING ALGORITHM

Boosting is an ensemble technique that combines multiple weak learners to create a strong learner. Weak models are trained in series such that each next model tries to correct errors of the previous model until the entire training dataset is predicted correctly. One of the most well-known boosting algorithms is AdaBoost (Adaptive Boosting).

Boosting algorithms are a sequential ensemble method. Boosting has many variations, but they all follow the same general procedure. Boosting trains a learner on some initial dataset, *d*. The resultant learner is typically weak, misclassifying many samples in the dataset. Much like bagging, boosting then samples instances from the initial dataset to create a new dataset (d_2). Unlike bagging, however, boosting prioritizes misclassified data instances from the first model or learner. A new learner is trained on this new dataset d_2. Then a third dataset (d_3) is then compiled from d_1 and d_2, prioritizes the second learner's misclassified samples and instances in which d_1 and d_2 disagree. The process repeats *n* times to produce *n* learners. Boosting then combines and weights the all the learners together to produce final predictions.



Boosting algorithms largely differ in how they prioritize erroneously predicted data instances when creating a new dataset. Two of the most prominent boosting methods may illustrate this:

- **Adaptive boosting** (AdaBoost) weights model errors. That is, when creating a new iteration of a dataset for training the next learner, AdaBoost adds weights to the previous learner's misclassified samples, causing the next learner to prioritize those misclassified samples.

- **Gradient boosting** uses residual errors when training new learners. Rather than weight misclassified samples, gradient boosting uses residual errors from a previous model to set target predictions for the next model. In this way, it attempts to close the gap of error left by one model

Here is an overview of Boosting algorithm:

- **Initialize Model Weights**: Begin with a single weak learner and assign equal weights to all training examples.
- **Train Weak Learner**: Train weak learners on these dataset.
- **Sequential Learning**: Boosting works by training models sequentially where each model focuses on correcting the errors of its predecessor. Boosting typically uses a single type of weak learner like decision trees.
- **Weight Adjustment**: Boosting assigns weights to training datapoints. Misclassified examples receive higher weights in the next iteration so that next models pay more attention to them.

3.4 STACKING:

tacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future.

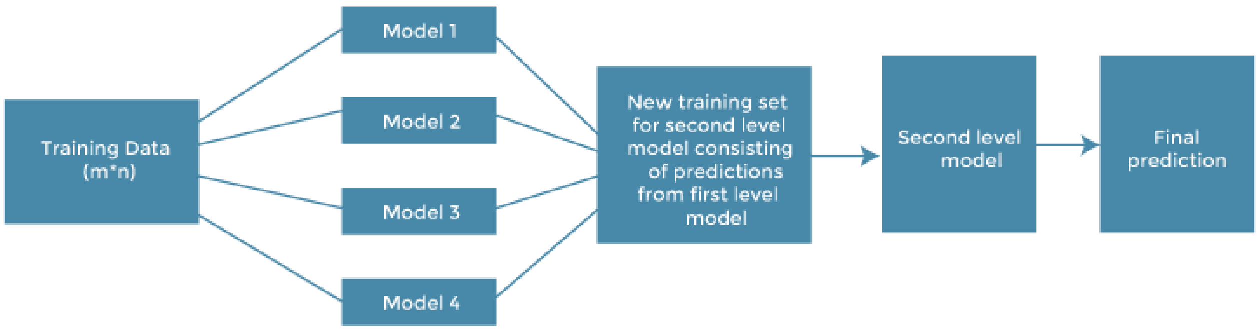
This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

Stacking is also known as **a stacked generalization** and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

Architecture of Stacking

The architecture of the stacking model is designed in such as way that it consists of two or more base/learner's models and a meta-model that combines the predictions of the base models. These base models are called level 0 models, and the meta-model is known as the level 1 model. So, the Stacking ensemble method includes **original (training) data, primary level models, primary level prediction, secondary level model, and final prediction**. The basic architecture of stacking can be represented as shown below the image.


Training Data (m*n)
Model 1
Model 2
Model 3
Model 4
New training set for second level model consisting of predictions from first level model
Second level model
Final prediction

- **Original data:** This data is divided into n-folds and is also considered test data or training data.

- **Base models:** These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- **Level-0 Predictions:** Each base model is triggered on some training data and provides different predictions, which are known as **level-0 predictions.**
- **Meta Model:** The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the **level-1 model**.
- **Level-1 Prediction:** The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

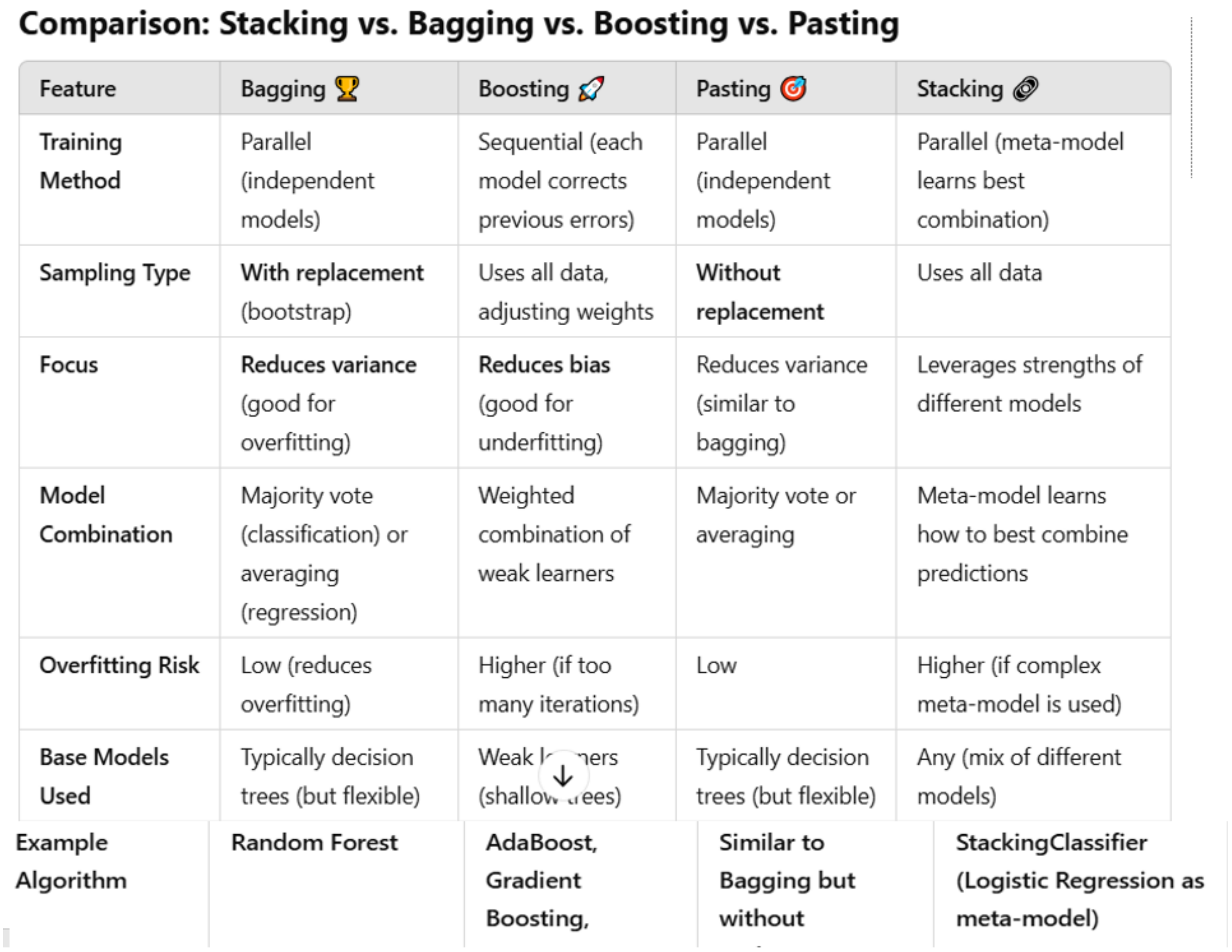
Steps to implement Stacking models:

There are some important steps to implementing stacking models in machine learning. These are as follows:

- Split training data sets into n-folds using the **Repeated Stratified K Fold** as this is the most common approach to preparing training datasets for meta-models.
- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.
- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model.
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

3.5 PASTING in ENSEMBLE LEARNING

- Sampling Without Replacement: Each base learner is trained on a different random subset of the training data, but each sample appears only once.
- Less Variance Reduction: Since no data points are repeated, the diversity among models is slightly lower than in bagging.
- More Efficient for Large Datasets: Since each data point is used only once across models, pasting can be more efficient when dealing with massive datasets.
- 🔹 Example: If you have 1000 samples, each base model gets a unique subset of (say) 800 samples, and no sample appears more than once.



Comparison: Stacking vs. Bagging vs. Boosting vs. Pasting

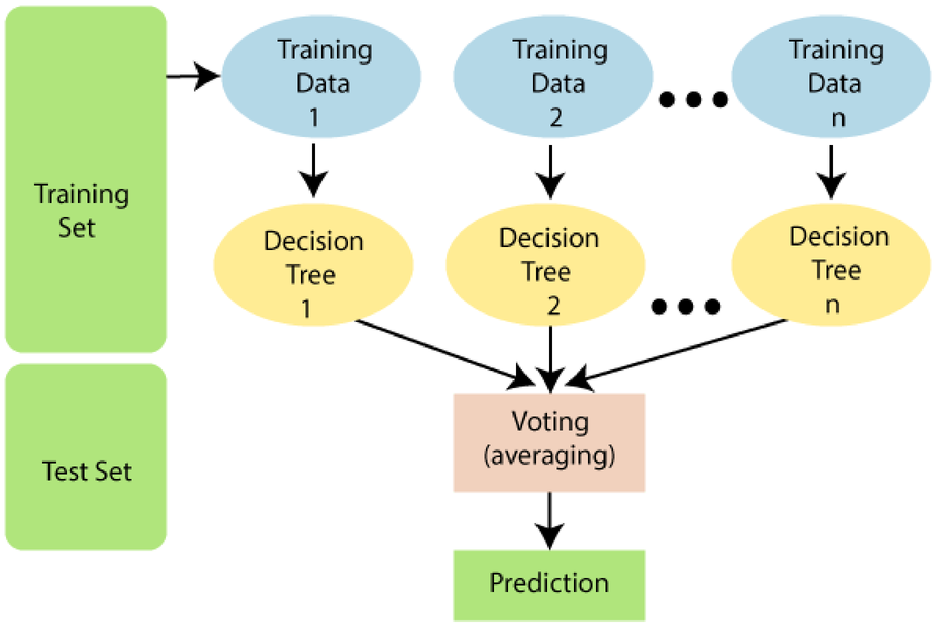
Feature	Bagging 🏆	Boosting 🚀	Pasting 🎯	Stacking 🔗
Training Method	Parallel (independent models)	Sequential (each model corrects previous errors)	Parallel (independent models)	Parallel (meta-model learns best combination)
Sampling Type	**With replacement** (bootstrap)	Uses all data, adjusting weights	**Without replacement**	Uses all data
Focus	**Reduces variance** (good for overfitting)	**Reduces bias** (good for underfitting)	Reduces variance (similar to bagging)	Leverages strengths of different models
Model Combination	Majority vote (classification) or averaging (regression)	Weighted combination of weak learners	Majority vote or averaging	Meta-model learns how to best combine predictions
Overfitting Risk	Low (reduces overfitting)	Higher (if too many iterations)	Low	Higher (if complex meta-model is used)
Base Models Used	Typically decision trees (but flexible)	Weak l[illegible]ners (shallo[illegible]ees)	Typically decision trees (but flexible)	Any (mix of different models)
Example Algorithm	**Random Forest**	**AdaBoost, Gradient Boosting,**	**Similar to Bagging but without**	**StackingClassifier (Logistic Regression as meta-model)**

3.6 RANDOM FORESTS ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. The below diagram explains the working of the Random Forest algorithm:



Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<="" li="" style="box-sizing: border-box;">

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

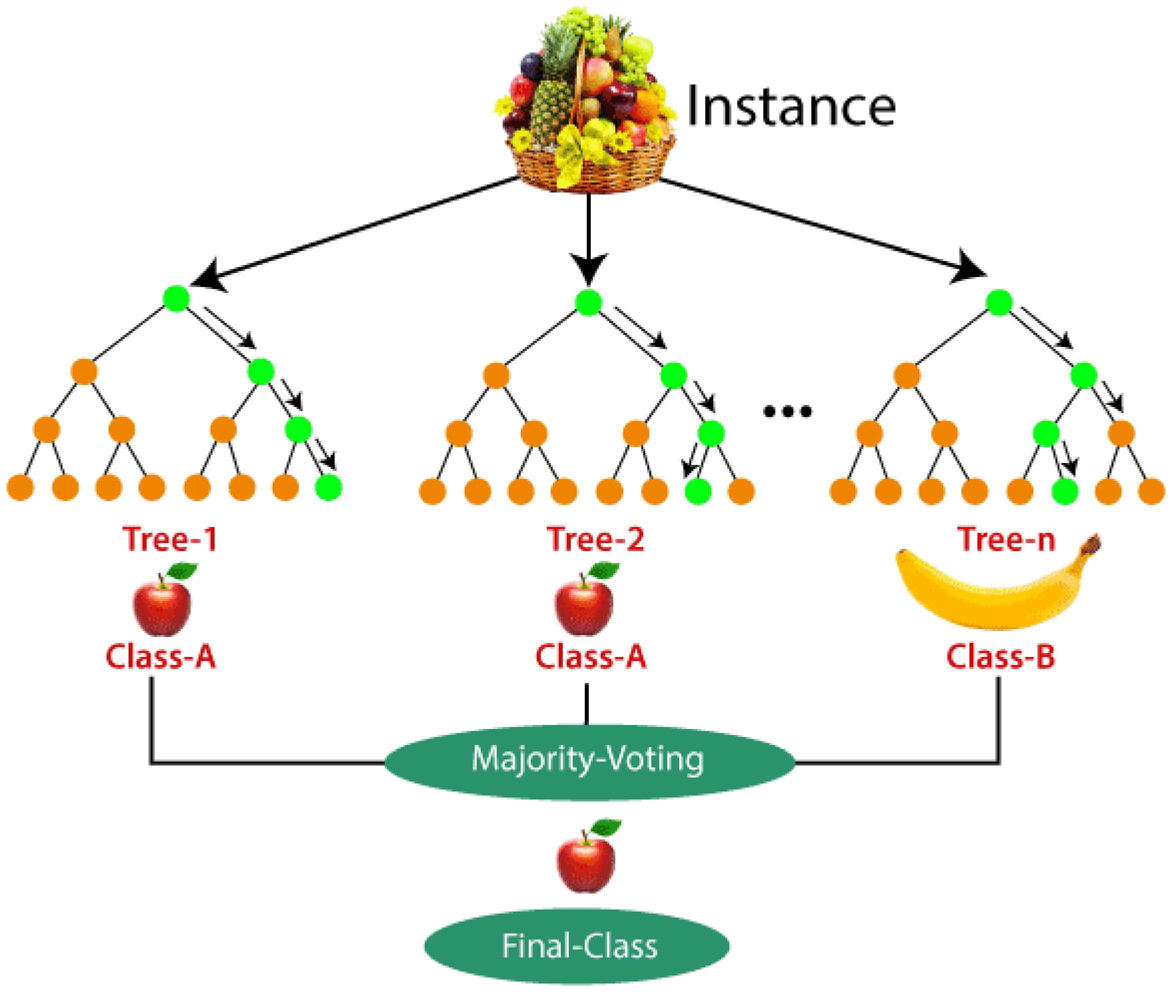
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.

2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.