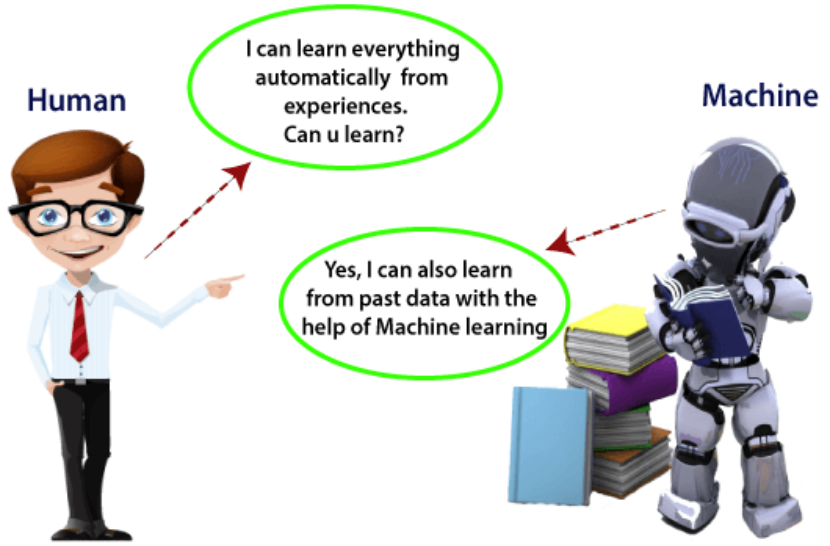
What is Machine Learning

It is subfield of computer science that give "Computer the ability to learn **without being explicitly programmed"**

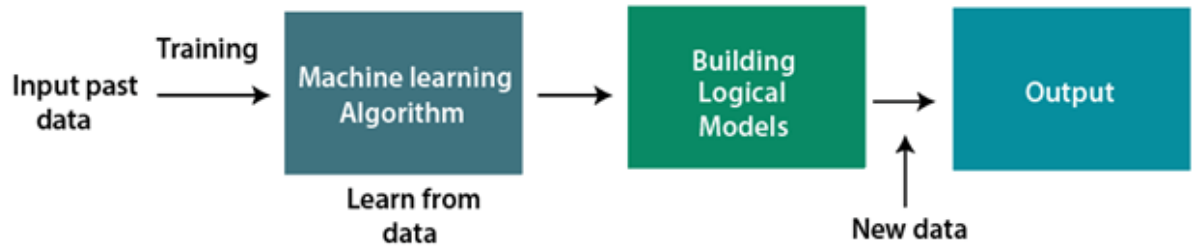


Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**. We can define it in a summarized way as:

How does Machine Learning work

A Machine Learning system **learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it**. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.

Need for Machine Learning

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, Currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyse the user interest and recommend product accordingly.

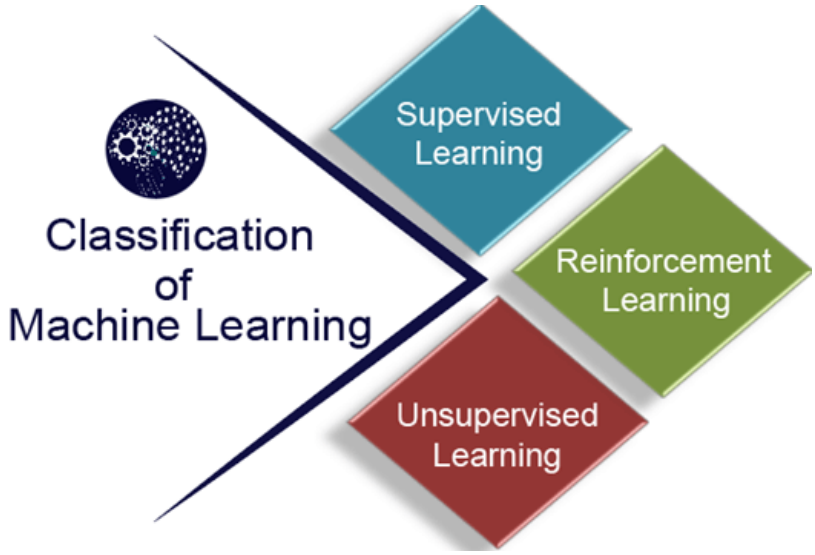
Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**



1) Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labelled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labelled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is **spam filtering**.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification**
- **Regression**

2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labelled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

- **Clustering**
- **Association**

3) Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

Applications of Machine learning

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:

1. Image Recognition:

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:

Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's **face detection** and **recognition algorithm**.

It is based on the Facebook project named "**Deep Face**," which is responsible for face recognition and person identification in the picture.

2. Speech Recognition

While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:

If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- **Real Time location** of the vehicle form Google Map app and sensors
- **Average time has taken** on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:

Machine learning is widely used by various e-commerce and entertainment companies such as **Amazon**, **Netflix**, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

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

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

Supervised Machine Learning

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

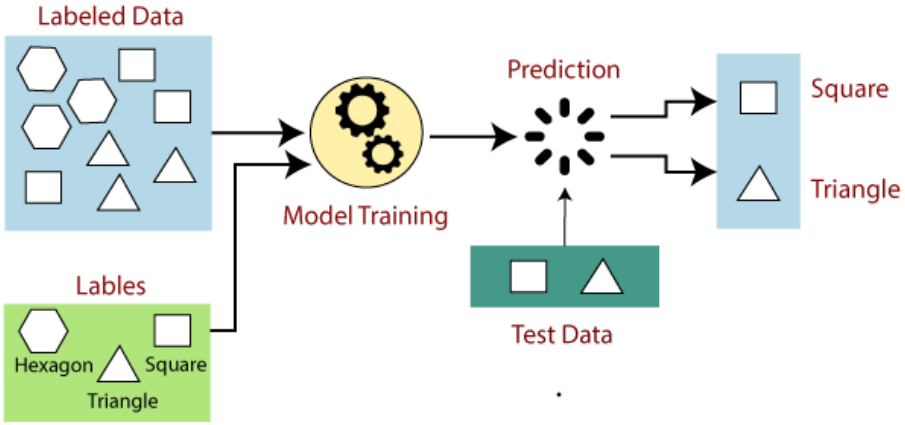
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides, then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

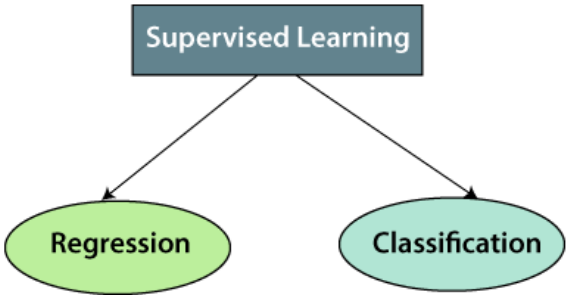
The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:



1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Note: We will discuss these algorithms in detail in later chapters.

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.

Disadvantages of supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.

- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

Unsupervised Machine Learning

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

> *Unsupervised learning is a type of machine learning in which models are trained using unlabelled dataset and are allowed to act on that data without any supervision.*

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

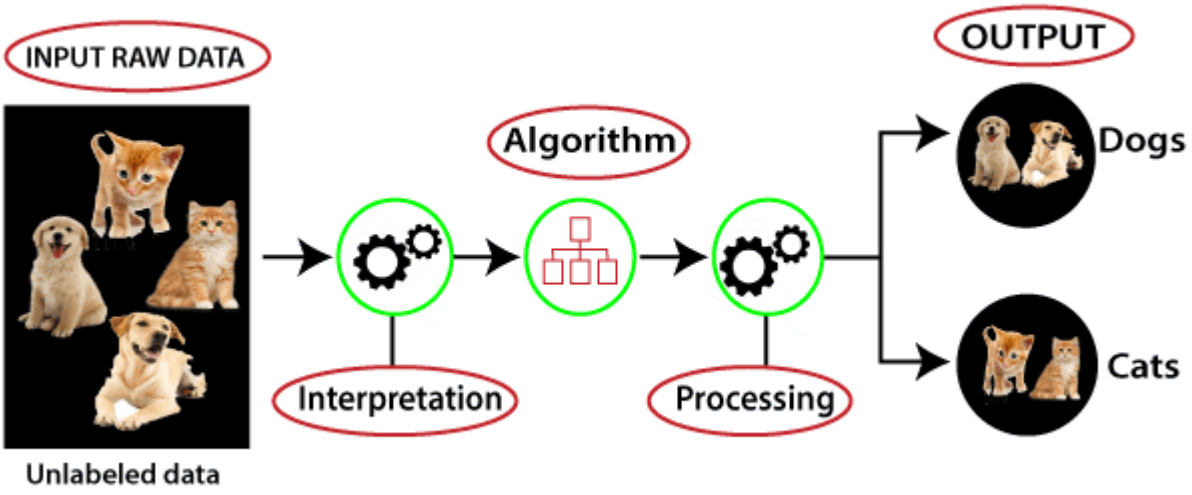
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabelled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

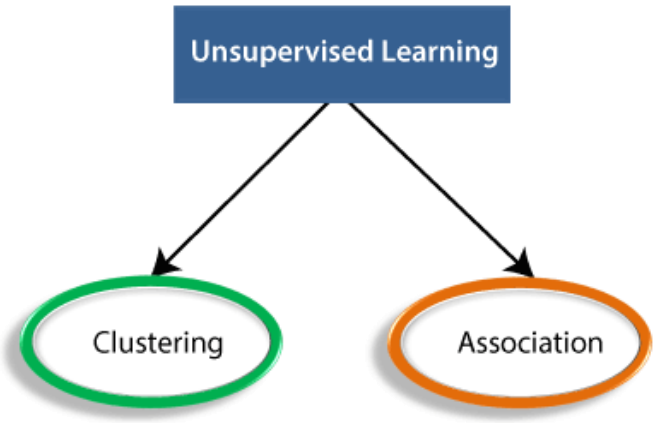


Here, we have taken an unlabelled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabelled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Note: We will learn these algorithms in later chapters.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labelled data.	Unsupervised learning algorithms are trained using unlabelled data.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** prob lems.	Unsupervised Learning can be classified in **Clustering** and **Associations** prob lems.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise).
It has controlled environment.	It has less controlled environment.

Hierarchical Clustering

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabelled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

Agglomerative Hierarchical clustering

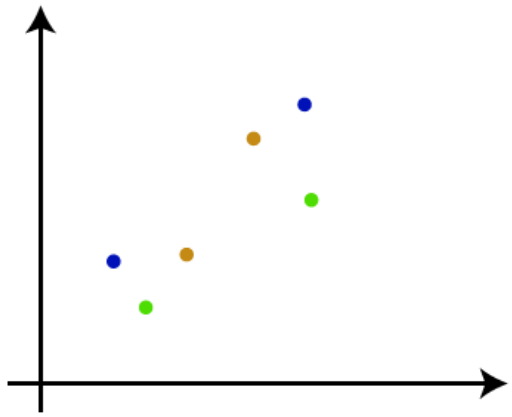
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

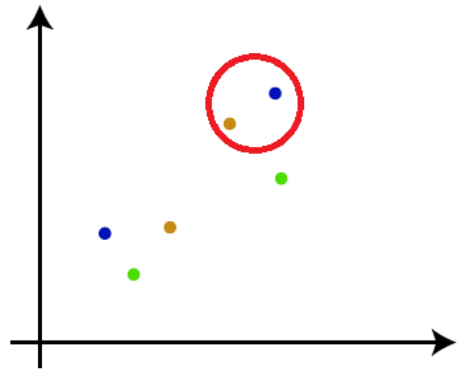
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

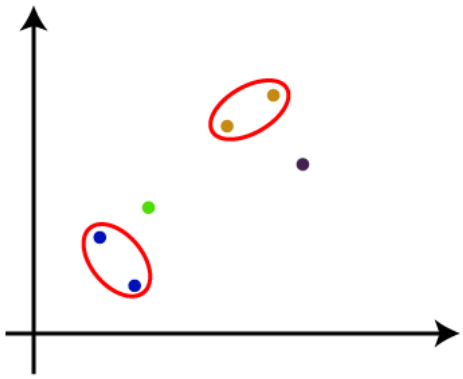
Step-1: Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



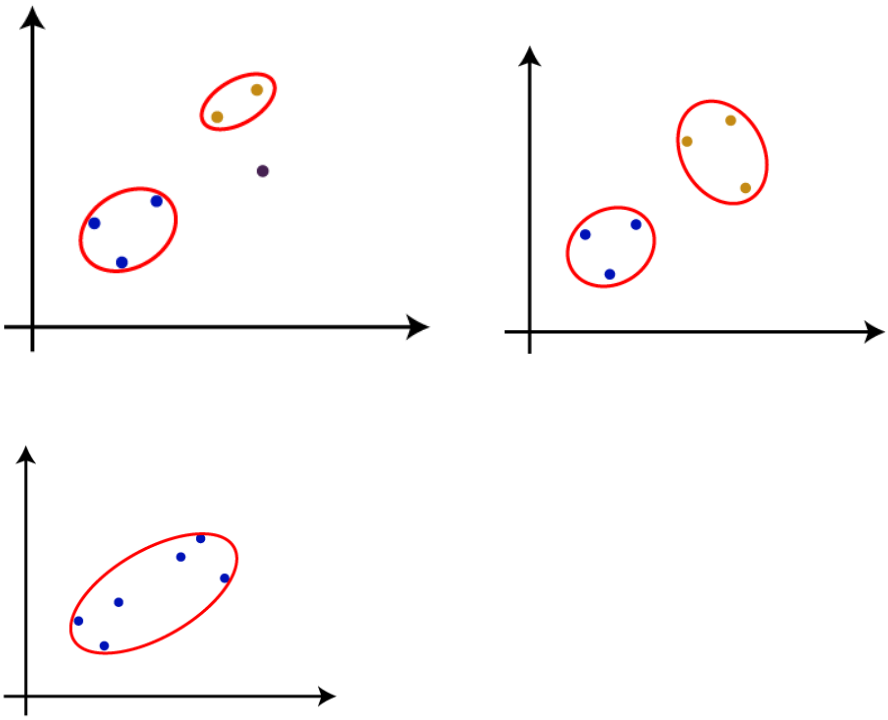
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.



- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:

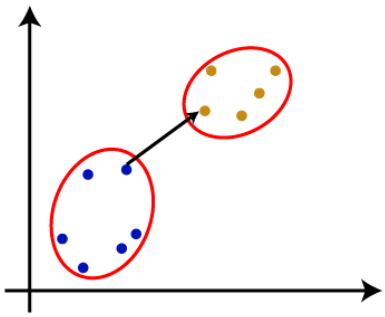


- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

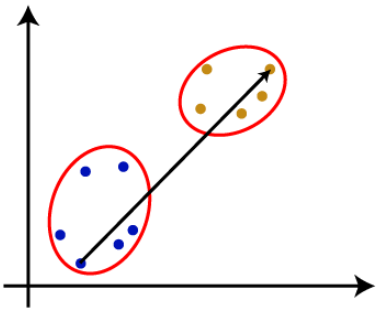
Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

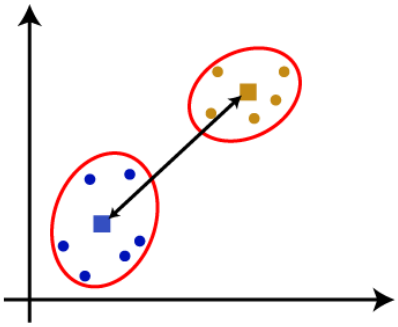
1. **Single Linkage:** It is the Minimum Distance between the closest points of clusters.



2. **Complete Linkage:** It is the Maximum distance between the two points of two different clusters.



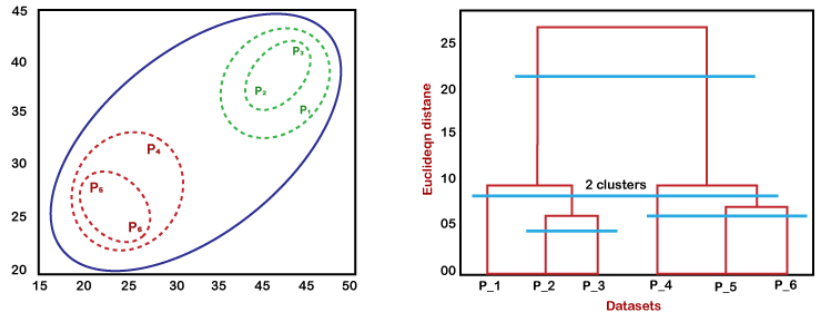
3. **Average Linkage:** It is Average distance between the data points of two different clusters.
4. **Centroid Linkage:** It is the distance between the centroid of the clusters.



Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.

Advantages of Hierarchical Clustering:

- Don’t required number of clusters to be specified.
- Easy to implement.
- Produces dendrogram, which helps with understanding data.

Disadvantages of Hierarchical Clustering:

- Can never undo any previous step thought the algorithm.
- Generally, has long run times.
- Sometimes difficult to identify the number of clusters by the dendrogram.

K-Means Clustering Algorithm

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

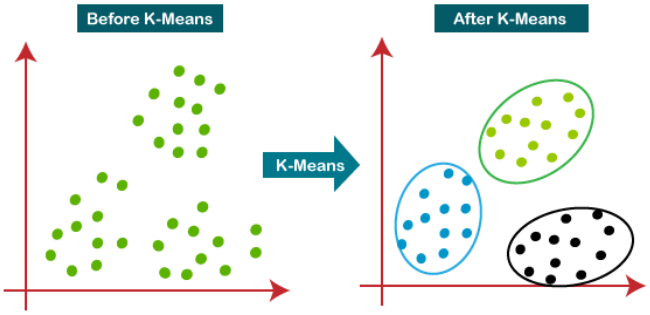
The algorithm takes the unlabelled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K centre points or centroids by an iterative process.
- Assigns each data point to its closest k-centre. Those data points which are near to the particular k-centre, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

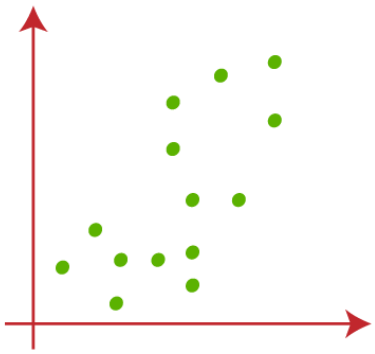
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

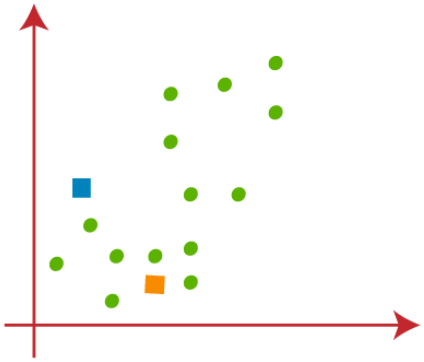
Let's understand the above steps by considering the visual plots:

Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

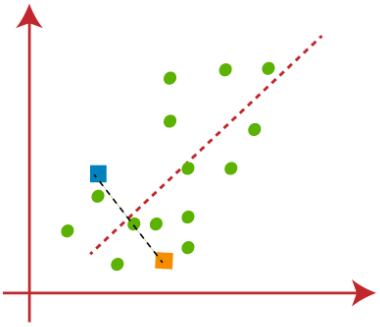


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not

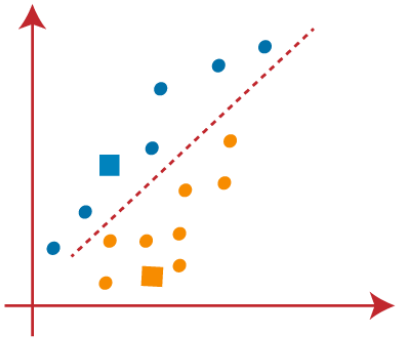
the part of our dataset. Consider the below image:



- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:

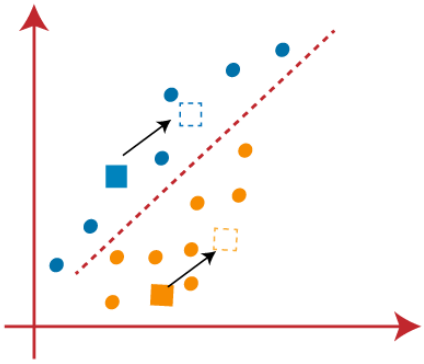


From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's colour them as blue and yellow for clear visualization.

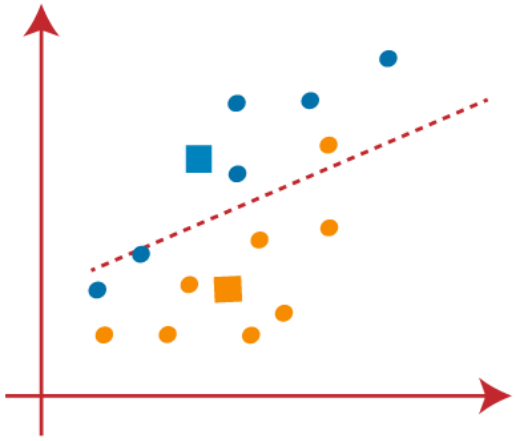


- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute

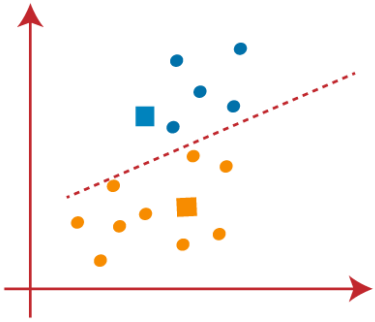
the centre of gravity of these centroids, and will find new centroids as below:



Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:

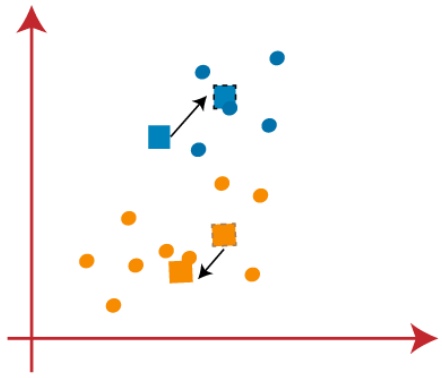


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

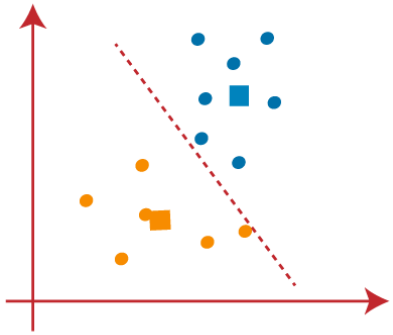


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

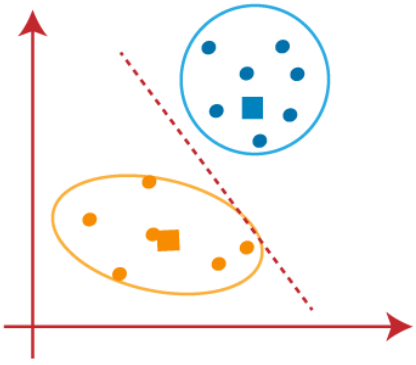
- We will repeat the process by finding the centre of gravity of centroids, so the new centroids will be as shown in the below image:



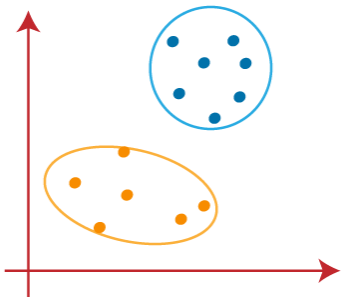
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of TWSS value. **TWSS** stands for **Total Within Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of TWSS (for 3 clusters) is given below:

$$TWSS= \sum_{Pi \text{ in } Cluster1} distance(P_i\ C_1)^2 +\sum_{Pi \text{ in } Cluster2} distance(P_i\ C_2)^2+\sum_{Pi \text{ in } CLuster3} distance(P_i\ C_3)^2$$

In the above formula of TWSS,

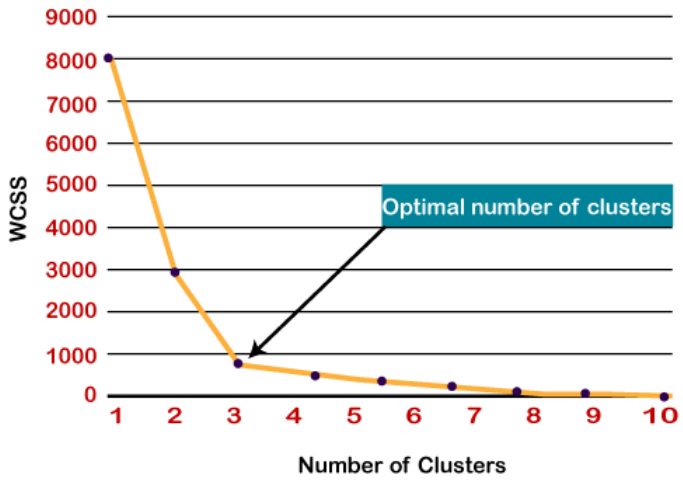
$\sum_{Pi \text{ in } Cluster1} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the TWSS value.
- Plots a curve between calculated TWSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Train and Test datasets in Machine Learning

Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

What is Training Dataset?

The ***training data is the biggest (in -size) subset of the original dataset, which is used to train or fit the machine learning model***. Firstly, the training data is fed to the ML algorithms, which lets them learn how to make predictions for the given task.

For example, for training a sentiment analysis model, the training data could be as below:

Input	Output (Labels)
The New UI is Great	Positive
Update is really Slow	Negative

The training data varies depending on whether we are using Supervised Learning or Unsupervised Learning Algorithms.

For **Unsupervised learning**, the training data contains unlabelled data points, i.e., inputs are not tagged with the corresponding outputs. Models are required to find the patterns from the given training datasets in order to make predictions.

On the other hand, for supervised learning, the training data contains labels in order to train the model and make predictions.

The type of training data that we provide to the model is highly responsible for the model's accuracy and prediction ability. It means that the better the quality of the training data, the better will be the performance of the model. Training data is approximately more than or equal to 60% of the total data for an ML project.

What is Test Dataset?

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. ***The test dataset is another subset of original data, which is independent of the training dataset***. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project.

At this stage, we can also check and compare the testing accuracy with the training accuracy, which means how accurate our model is with the test dataset against the training dataset. If the accuracy of the model on training data is greater than that on testing data, then the model is said to have **overfitting**.

Accuracy = (Training>Testing) -> Overfit Model

Accuracy = (Training<Testing) -> Underfit Model

Accuracy = (Training=Testing) -> Best fit Model

The testing data should:

- Represent or part of the original dataset.
- It should be large enough to give meaningful predictions.

Need of Splitting dataset into Train and Test set

Splitting the dataset into train and test sets is one of the important parts of data pre-processing, as by doing so, we can improve the performance of our model and hence give better predictability.

We can understand it as if we train our model with a training set and then test it with a completely different test dataset, and then our model will not be able to understand the correlations between the features.

Therefore, if we train and test the model with two different datasets, then it will decrease the performance of the model. Hence it is important to split a dataset into two parts, i.e., train and test set.

In this way, we can easily evaluate the performance of our model. Such as, if it performs well with the training data, but does not perform well with the test dataset, then it is estimated that the model may be overfitted.

For splitting the dataset, we can use the **train_test_split** function of **scikit-learn.**

Overfitting and Underfitting issues

Overfitting and underfitting are the most common problems that occur in the Machine Learning model.

*A model can be said as **overfitted** when it performs quite well with the training dataset but does not generalize well with the new or unseen dataset*. The issue of overfitting occurs when the model tries to cover all the data points and hence starts caching noises present in the data. Due to this, it can't generalize well to the new dataset. Because of these issues, the accuracy and efficiency of the model

degrade. Generally, the complex model has a high chance of overfitting. There are various ways by which we can avoid overfitting in the model, such as Using the **Cross-Validation method, early stopping the training, or by regularization**, etc.

On the other hand, the ***model is said to be under-fitted when it is not able to capture the underlying trend of the data***. It means the model shows poor performance even with the training dataset. In most cases, underfitting issues occur when the model is not perfectly suitable for the problem that we are trying to solve. To avoid the overfitting issue, we can either increase the training time of the model or increase the number of features in the dataset.

Training data vs. Testing Data

- The main difference between training data and testing data is that training data is the subset of original data that is used to train the machine learning model, whereas testing data is used to check the accuracy of the model.
- The training dataset is generally larger in size compared to the testing dataset. The general ratios of splitting train and test datasets are **80:20, 70:30, or 90:10.**
- Training data is well known to the model as it is used to train the model, whereas testing data is like unseen/new data to the model.

How do training and testing data work in Machine Learning?

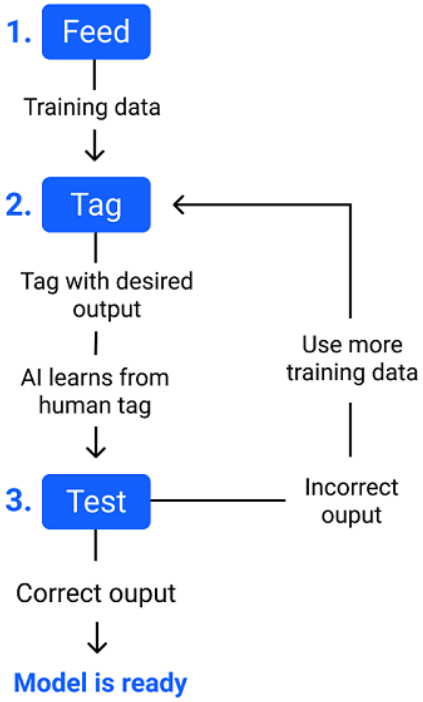
Machine Learning algorithms enable the machines to make predictions and solve problems on the basis of past observations or experiences. These experiences or observations an algorithm can take from the training data, which is fed to it. Further, one of the great things about ML algorithms is that they can learn and improve over time on their own, as they are trained with the relevant training data.

Once the model is trained enough with the relevant training data, it is tested with the test data. We can understand the whole process of training and testing in three steps, which are as follows:

1. **Feed:** Firstly, we need to train the model by feeding it with training input data.
2. **Define:** Now, training data is tagged with the corresponding outputs (in Supervised Learning), and the model transforms the training data into text vectors or a number of data features.

3. **Test:** In the last step, we test the model by feeding it with the test data/unseen dataset. This step ensures that the model is trained efficiently and can generalize well.

The above process is explained using a flowchart given below:

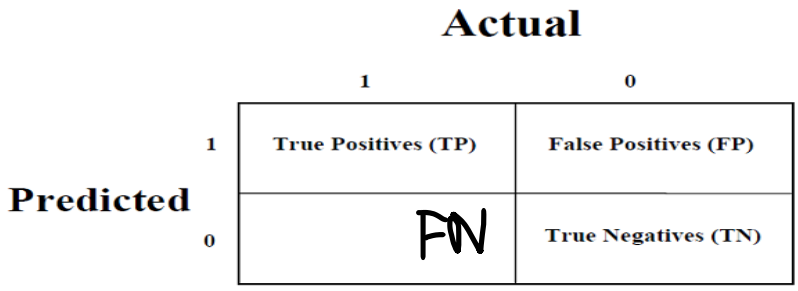


Performance Metrics for Classification Problems

We have discussed classification and its algorithms in the previous chapters. Here, we are going to discuss various performance metrics that can be used to evaluate predictions for classification problems.

Confusion Matrix

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)” as shown below −



Explanation of the terms associated with confusion matrix are as follows −

- **True Positives (TP)** − It is the case when both actual class & predicted class of data point is 1.
- **True Negatives (TN)** − It is the case when both actual class & predicted class of data point is 0.
- **False Positives (FP)** − It is the case when actual class of data point is 0 & predicted class of data point is 1.
- **False Negatives (FN)** − It is the case when actual class of data point is 1 & predicted class of data point is 0.

We can use confusion_matrix function of sklearn.metrics to compute Confusion Matrix of our classification model.

Classification Accuracy

It is most common performance metric for classification algorithms. It may be defined as the number of correct predictions made as a ratio of all predictions made. We can easily calculate it by confusion matrix with the help of following formula −

$$\text{Accuracy} = TP+TN/(TP+TN+FP+FN)$$

We can use accuracy_score function of sklearn.metrics to compute accuracy of our classification model.

Classification Report

This report consists of the scores of Precisions, Recall, F1 and Support. They are explained as follows −

1.Precision

Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

$$\text{Precision} = TP/(TP+FP)$$

2.Recall or Sensitivity

Recall may be defined as the number of positives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

$$\text{Recall} = TP/(TP+FN)$$

3.F1 Score

This score will give us the harmonic mean of precision and recall. Mathematically, F1 score is the weighted average of the precision and recall. The best value of F1 would be 1 and worst would be 0. We can calculate F1 score with the help of following formula −

$$F1 = 2 * (precision * recall) / (precision + recall)$$

F1 score is having equal relative contribution of precision and recall.

We can use classification_report function of sklearn.metrics to get the classification report of our classification model.

4.Jaccard Index

Defined as the size of the intersection divided by the size of the union of two label sets, is used to compare set of predicted labels for a sample to the corresponding set of labels in y_true.

$$JaccardIndex = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$JaccardDistance = 1 - JaccardIndex$$

5.Log Loss

Log loss, aka logistic loss or cross-entropy loss.

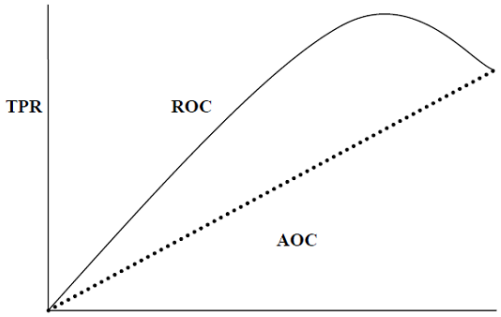
This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of a logistic model that returns `y_pred` probabilities for its training data `y_true`. The log loss is only defined for two or more labels. For a single sample with true label $y \in \{0, 1\}$ and a probability estimate $p = \Pr(y = 1)$, the log loss is:

$$L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

AUC (Area Under ROC curve)

AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric, based on varying threshold values, for classification problems. As name suggests, ROC is a probability curve and AUC measure the separability. In simple words, AUC-ROC metric will tell us about the capability of model in distinguishing the classes. Higher the AUC, better the model.

Mathematically, it can be created by plotting TPR (True Positive Rate) i.e., Sensitivity or recall vs FPR (False Positive Rate) i.e., 1-Specificity, at various threshold values. Following is the graph showing ROC, AUC having TPR at y-axis and FPR at x-axis −



We can use roc_auc_score function of sklearn.metrics to compute AUC-ROC.

Performance Metrics for Regression Problems

Here, we are going to discuss various performance metrics that can be used to evaluate predictions for regression problems.

Mean Absolute Error (MAE)

It is the simplest error metric used in regression problems. It is basically the sum of average of the absolute difference between the predicted and actual values. In simple words, with MAE, we can get an idea of how wrong the predictions were. MAE does not indicate the direction of the model i.e., no indication about underperformance or overperformance of the model. The following is the formula to calculate MAE −

$$MAE=1n\sum|Y-Y^|$$

Here, Y=Actual Output Values

And Y^= Predicted Output Values.

We can use mean_absolute_error function of sklearn.metrics to compute MAE.

Mean Square Error (MSE)

MSE is like the MAE, but the only difference is that the it squares the difference of actual and predicted output values before summing them all instead of using the absolute value. The difference can be noticed in the following equation −

$$MSE=1n\sum(Y-Y^)^2$$

Here, Y=Actual Output Values

And Y^ = Predicted Output Values.

We can use mean_squared_error function of sklearn.metrics to compute MSE.

R Squared (R^2)

R Squared metric is generally used for explanatory purpose and provides an indication of the goodness or fit of a set of predicted output values to the actual output values. The following formula will help us understanding it −

$$R2=1-1n\sum ni=1(Yi-Yi^)2 / \sum ni=1(Yi-Yi)2$$

In the above equation, numerator is MSE and the denominator is the variance in Y values.

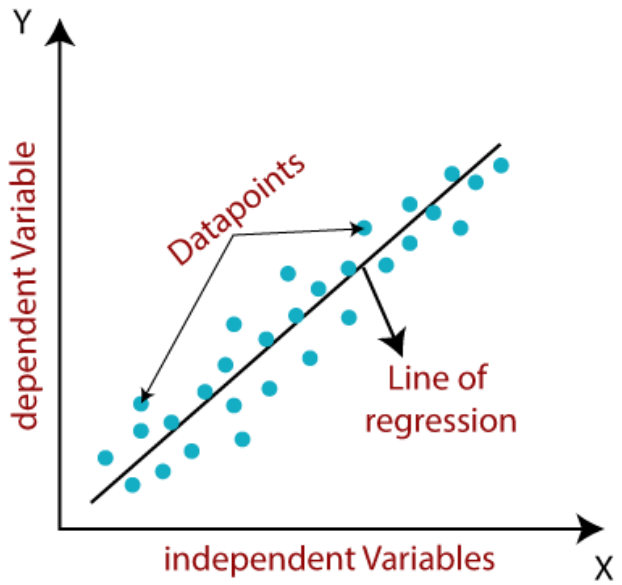
We can use r2_score function of sklearn.metrics to compute R squared value.

Linear Regression in Machine Learning

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x+ \varepsilon$$

Here,

X= Independent Variable (predictor Variable)
Y= Dependent Variable (Target Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

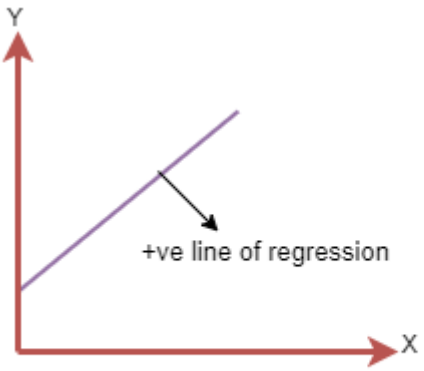
Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**
  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regression:**
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

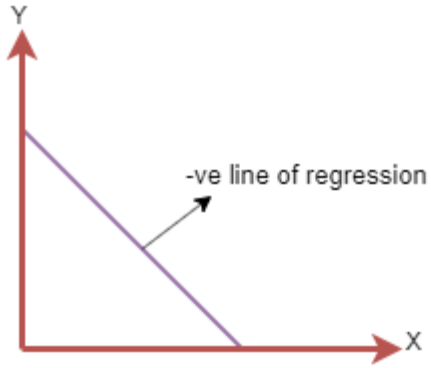
- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent

variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y= -a_0+a_1x$

Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Cost function-

- The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.
- Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.
- We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis function**.

For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

For the above linear equation, MSE can be calculated as:

$$MSE = 1\frac{1}{N}\sum_{i=1}^{n}(y_i - (a_1x_i + a_0))^2$$

Where,

N=Total number of observation
Yi = Actual value
$(a1x_i+a_0)$= Predicted value.

Residuals: The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

Gradient Descent:

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**. It can be achieved by below method:

1. R-squared method:

- R-squared is a statistical method that determines the goodness of fit.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a **coefficient of determination,** or **coefficient of multiple determination** for multiple regression.
- It can be calculated from the below formula:

$$\text{R-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$$

Simple Linear Regression in Machine Learning

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the ***dependent variable must be a continuous/real value***. However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

- **Model the relationship between the two variables.** Such as the relationship between Income and expenditure, experience and Salary, etc.
- **Forecasting new observations.** Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

Simple Linear Regression Model:

The Simple Linear Regression model can be represented using the below equation:

$$y= a_0+a_1x+ \varepsilon$$

Where,

a0= It is the intercept of the Regression line (can be obtained putting x=0)
a1= It is the slope of the regression line, which tells whether the line is increasing or decreasing.
ε = The error term. (For a good model it will be negligible)

Output:

By executing the above line of code, we will get the output as:

In the above plot, there are observations given by the blue colour, and prediction is given by the red regression line. As we can see, most of the observations are close to the regression line, hence we can say our Simple Linear Regression is a good model and able to make good predictions.

Multiple Linear Regression

In the previous topic, we have learned about Simple Linear Regression, where a single Independent/Predictor(X) variable is used to model the response variable (Y). But there may be various cases in which the response variable is affected by more than one predictor variable; for such cases, the Multiple Linear Regression algorithm is used.

Moreover, Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. We can define it as:

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Example:

Prediction of CO_2 emission based on engine size and number of cylinders in a car.

Some key points about MLR:

- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- Each feature variable must model the linear relationship with the dependent variable.
- MLR tries to fit a regression line through a multidimensional space of data-points.

MLR equation:

In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x_1, x_2, x_3, ...,x_n. Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

1. $\mathbf{Y = b_0x_1 + b_1x2 + b_2x_3 + b + ... b_nx_n}$ ..... (a)

Where,

Y= Output/Response variable

b_0, b_1, b_2, b_3 , b_n....= Coefficients of the model.

x_1, x_2, x_3, x_4,...= Various Independent/feature variable

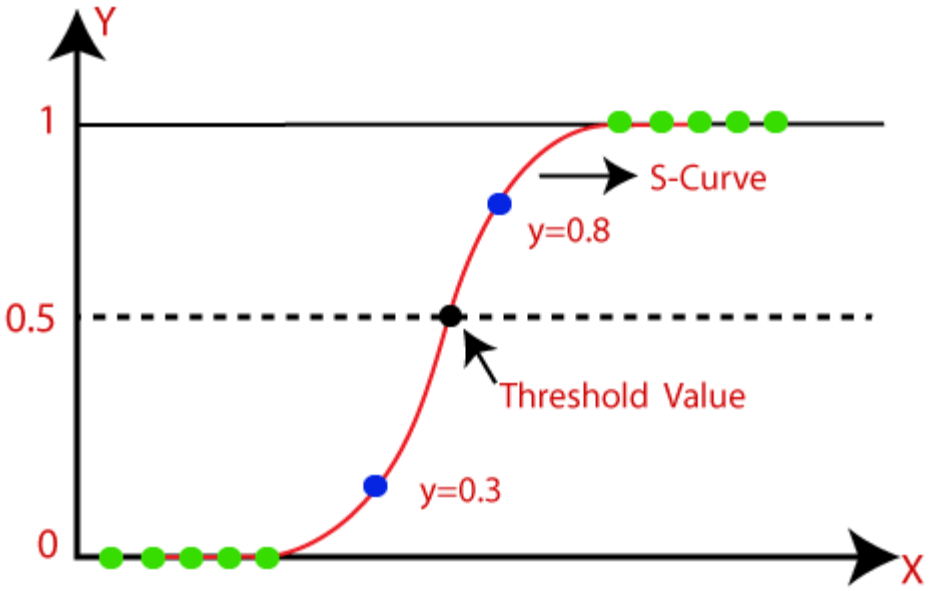
Applications of Multiple Linear Regression:

There are mainly two applications of Multiple Linear Regression:

- Effectiveness of Independent variable on prediction:
- Predicting the impact of changes:

Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- 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**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: ***Logistic regression uses the concept of predictive modelling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.***

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

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

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

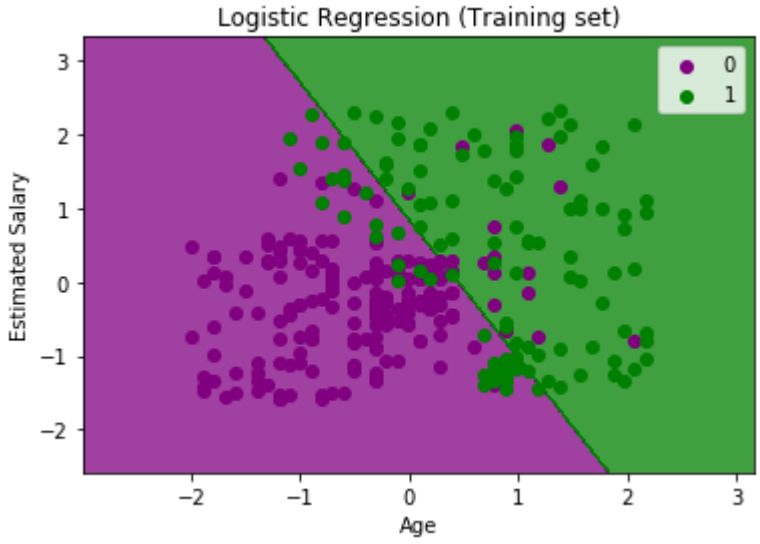
The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Output: By executing the above code, we will get the below output:



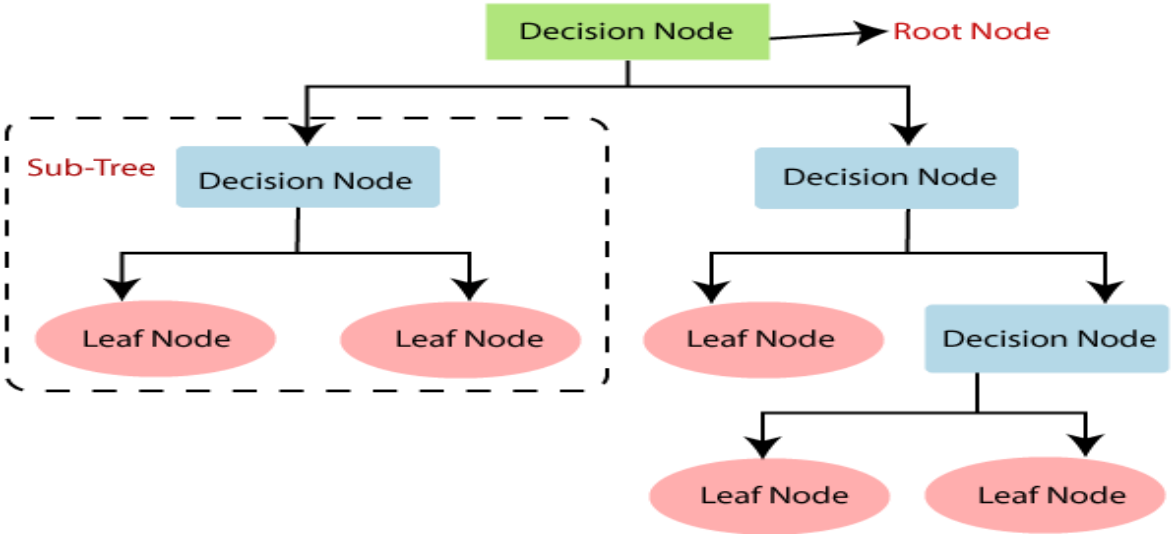
The graph can be explained in the below points:

- In the above graph, we can see that there are some **Green points** within the green region and **Purple points** within the purple region.
- All these data points are the observation points from the training set, which shows the result for purchased variables.
- This graph is made by using two independent variables i.e., **Age on the x-axis** and **Estimated salary on the y-axis**.
- The **purple point observations** are for which purchased (dependent variable) is probably 0, i.e., users who did not purchase the SUV car.
- The **green point observations** are for which purchased (dependent variable) is probably 1 means user who purchased the SUV car.
- We can also estimate from the graph that the users who are younger with low salary, did not purchase the car, whereas older users with high estimated salary purchased the car.
- But there are some purple points in the green region (Buying the car) and some green points in the purple region(Not buying the car). So we can say that younger users with a high estimated salary purchased the car, whereas an older user with a low estimated salary did not purchase the car.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

> ***Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.***



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

 Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

 Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

 Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

 Branch/Sub Tree: A tree formed by splitting the tree.

 Pruning: Pruning is the process of removing the unwanted branches from the tree.

 Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

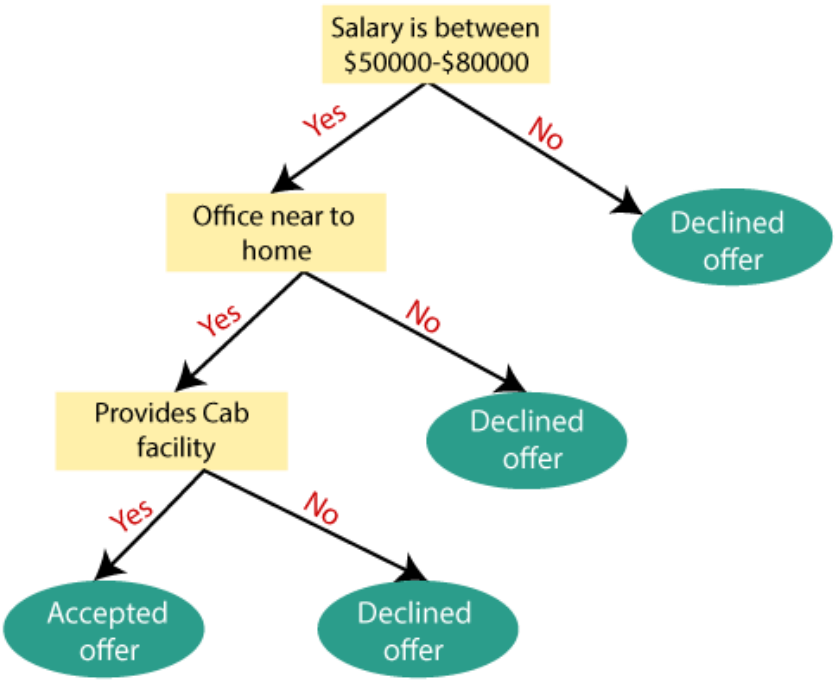
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **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:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

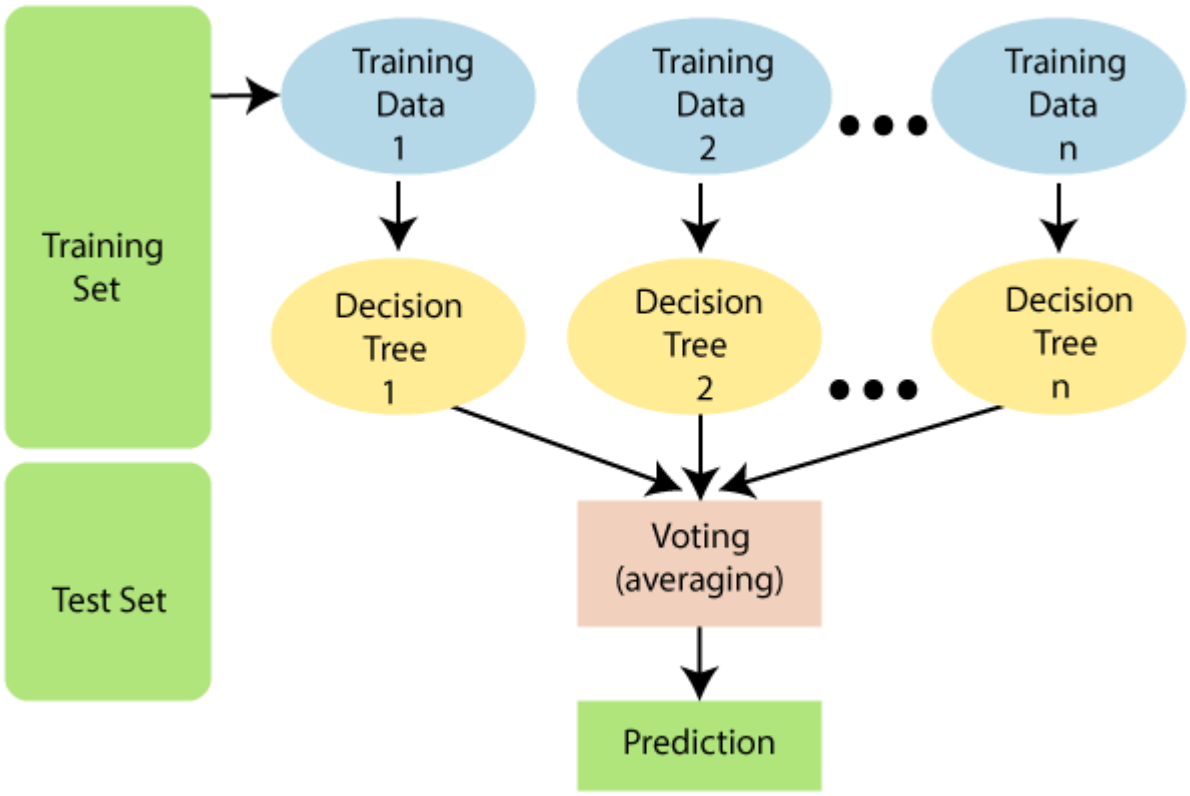
Random Forest Algorithm

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.

Why use Random Forest?

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

<=”” li=””>

- 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).

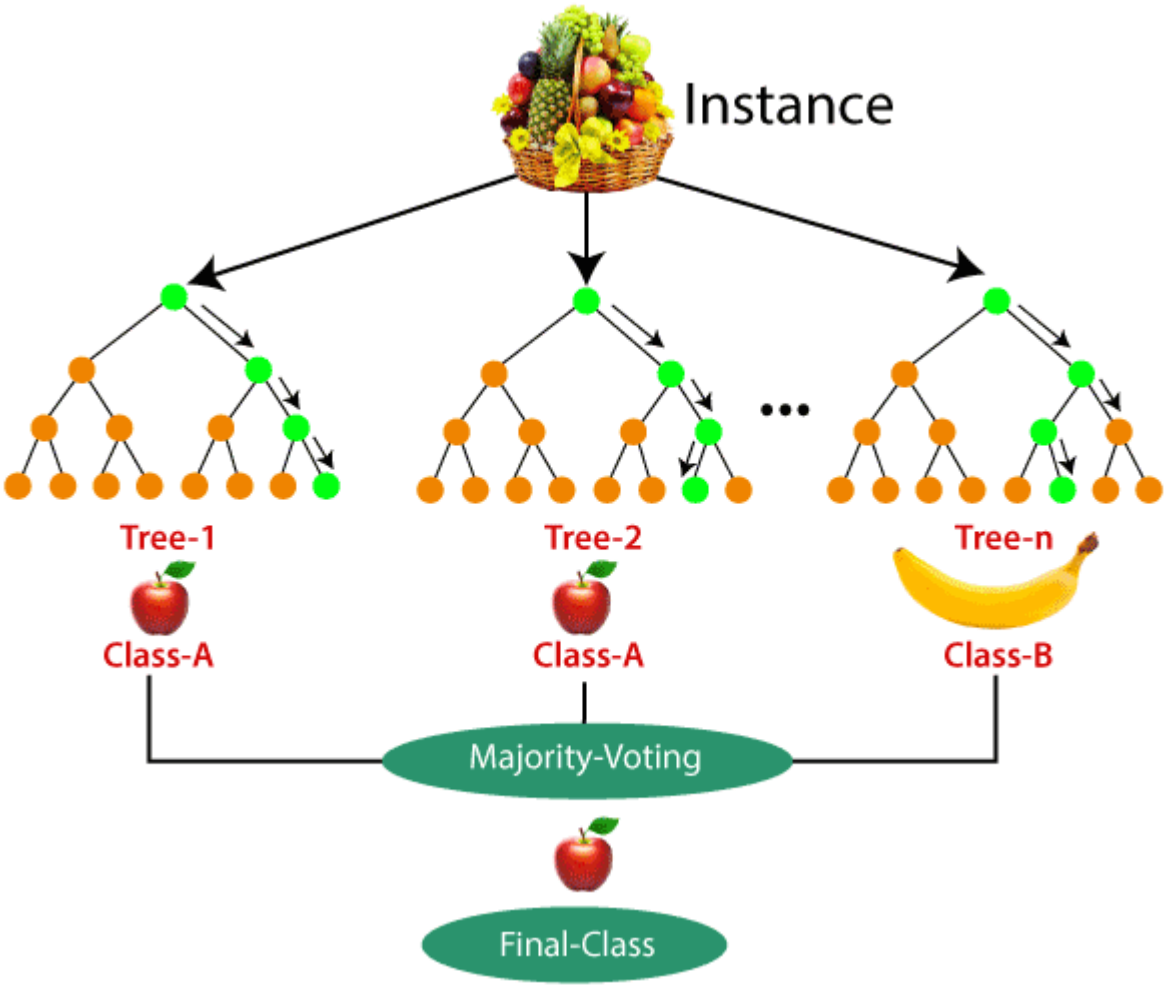
Step-3: Choose the number N for decision trees that you want to build.

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.

Bagging Vs Boosting

We all use the Decision Tree Technique on day to day life to make the decision. Organizations use these supervised machine learning techniques like Decision trees to make a better decision and to generate more surplus and profit.

Ensemble methods combine different decision trees to deliver better predictive results, afterward utilizing a single decision tree. The primary principle behind the ensemble model is that a group of weak learners come together to form an active learner.

There are two techniques given below that are used to perform ensemble decision tree.

Bagging

Bagging is used when our objective is to reduce the variance of a decision tree. Here the concept is to create a few subsets of data from the training sample, which is chosen randomly with replacement. Now each collection of subset data is used to prepare their decision trees thus, we end up with an ensemble of various models. The average of all the assumptions from numerous tress is used, which is more powerful than a single decision tree.

Random Forest is an expansion over bagging. It takes one additional step to predict a random subset of data. It also makes the random selection of features rather than using all features to develop trees. When we have numerous random trees, it is called the Random Forest.

These are the following steps which are taken to implement a Random Forest:

- Let us consider **X** observations **Y** features in the training data set. First, a model from the training data set is taken randomly with substitution.
- The tree is developed to the largest.
- The given steps are repeated, and prediction is given, which is based on the collection of predictions from n number of trees.

Advantages of using Random Forest technique:

- It manages a higher dimension data set very well.
- It manages missing quantities and keeps accuracy for missing data.

Disadvantages of using Random Forest technique:

Since the last prediction depends on the mean predictions from subset trees, it won't give precise value for the regression model.

Boosting:

Boosting is another ensemble procedure to make a collection of predictors. In other words, we fit consecutive trees, usually random samples, and at each step, the objective is to solve net error from the prior trees.

If a given input is misclassified by theory, then its weight is increased so that the upcoming hypothesis is more likely to classify it correctly by consolidating the entire set at last converts weak learners into better performing models.

Gradient Boosting is an expansion of the boosting procedure.

1. Gradient Boosting = Gradient Descent + Boosting

It utilizes a gradient descent algorithm that can optimize any differentiable loss function. An ensemble of trees is constructed individually, and individual trees are summed successively. The next tree tries to restore the loss (It is the difference between actual and predicted values).

Advantages of using Gradient Boosting methods:

- It supports different loss functions.
- It works well with interactions.

Disadvantages of using a Gradient Boosting method:

- It requires cautious tuning of different hyper-parameters.

Difference between Bagging and Boosting:

Bagging	Boosting
Various training data subsets are randomly drawn with replacement from the whole training dataset.	Each new subset contains the components that were misclassified by previous models.
Bagging attempts to tackle the over-fitting issue.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then we need to apply bagging.	If the classifier is steady and straightforward (high bias), then we need to apply boosting.
Every model receives an equal weight.	Models are weighted by their performance.
Objective to decrease variance, not bias.	Objective to decrease bias, not variance.
It is the easiest way of connecting predictions that belong to the same type.	It is a way of connecting predictions that belong to the different types.
Every model is constructed independently.	New models are affected by the performance of the previously developed model.

Extra Tree Classifier for Feature Selection

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

To perform feature selection using the above forest structure, during the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria used in the decision of feature of split (Gini Index if the Gini Index is used in the construction of the forest) is computed. This value is called the Gini Importance of the feature. To perform feature selection, each feature is ordered in descending order according to the Gini Importance of each feature and the user selects the top k features according to his/her choice.

Consider the following data:-

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Let us build a hypothetical Extra Trees Forest for the above data with **five decision trees** and the value of k which decides the number of features in a random sample of features be **two**. Here the decision criteria used will be Information Gain. First, we calculate the entropy of the data. Note the formula for calculating the entropy is:-

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2(p_i)$$

where c is the number of unique class labels and p_i is the proportion of rows with output label is i.

Therefore, for the given data, the **entropy** is:-

$$Entropy(S) = -\frac{9}{14}log(\frac{9}{14}) - \frac{5}{14}log(\frac{5}{14})$$

$$\Rightarrow Entropy(S) = 0.940$$

Let the decision trees be constructed such that:-

- **1st Decision Tree gets data with the features Outlook and Temperature:**

Note that the formula for Information Gain is:-

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

Thus,

$$Gain(S, Outlook) = 0.940 - (\frac{5}{14}(\frac{-2}{5}log_2(\frac{2}{5}) + \frac{-3}{5}log_2(\frac{3}{5})) + \frac{4}{14}(\frac{-4}{4}log_2(\frac{4}{4}) + \frac{-0}{4}log_2(\frac{0}{4})) + \frac{5}{14}(\frac{-3}{5}log_2(\frac{3}{5}) + \frac{-2}{5}log_2(\frac{2}{5})))$$

$$\Rightarrow Gain(S, Outlook) = 0.246$$

Similarly:

$$Gain(S, Temperature) = 0.029$$

- **2nd Decision Tree gets data with the features Temperature and Wind:**

Using the above-given formulas:-

$$Gain(S, Temperature) = 0.029$$

$$Gain(S, Wind) = 0.048$$

c. strong>3rd Decision Tree gets data with the features Outlook and Humidity:

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

d. **4th Decision Tree gets data with the features Temperature and Humidity:**

$$Gain(S, Temperature) = 0.029$$

$$Gain(S, Humidity) = 0.151$$

e. **5th Decision Tree gets data with the features Wind and Humidity:**

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Humidity) = 0.151$$

Computing total Info Gain for each feature:-

Total Info Gain for Outlook = 0.246+0.246 = 0.492

Total Info Gain for Temperature = 0.029+0.029+0.029 = 0.087

Total Info Gain for Humidity = 0.151+0.151+0.151 = 0.453

Total Info Gain for Wind = 0.048+0.048 = 0.096

Thus, the most important variable to determine the output label according to the above constructed Extra Trees Forest is the feature “Outlook”.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.

- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

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.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So, to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes

7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

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

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35

Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes 'theorem:

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

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

P(Sunny)= 0.35

P(Yes)=0.71

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

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

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

P(No)= 0.29

P(Sunny)= 0.35

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

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

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

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.

- It performs well in multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

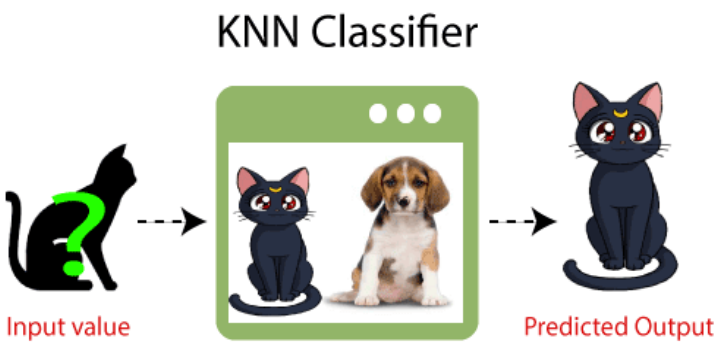
Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

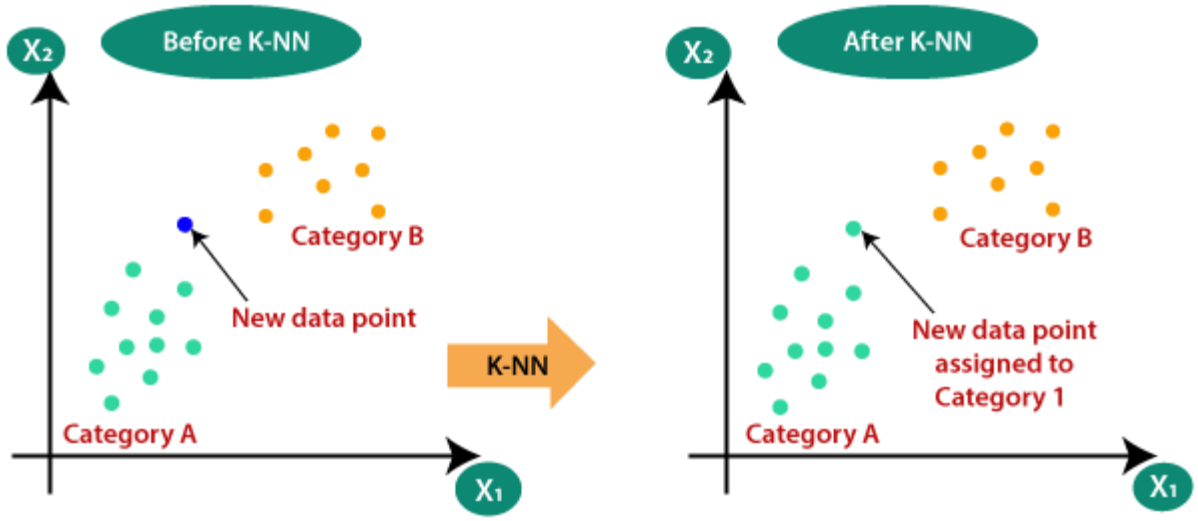
K-Nearest Neighbour(KNN) Algorithm for Machine Learning

- 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:



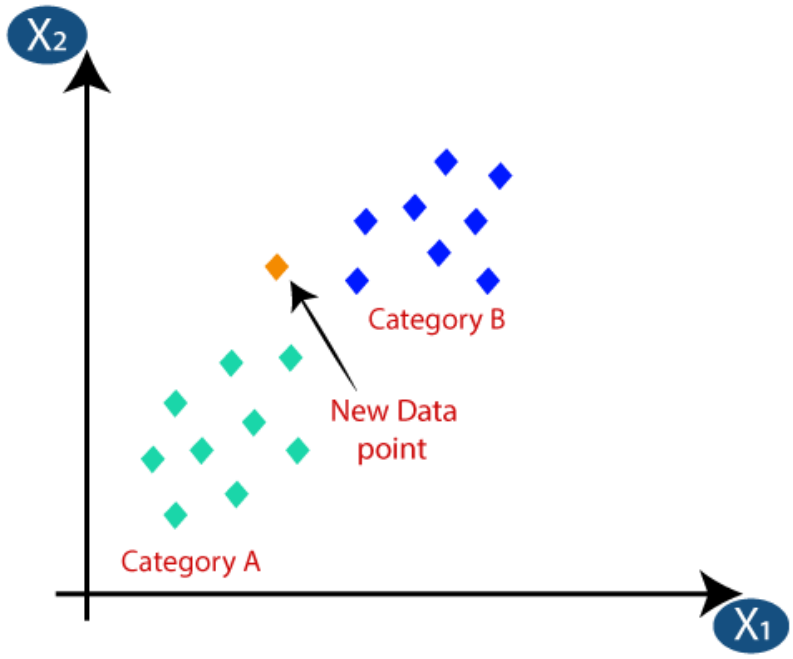
How does K-NN work?

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

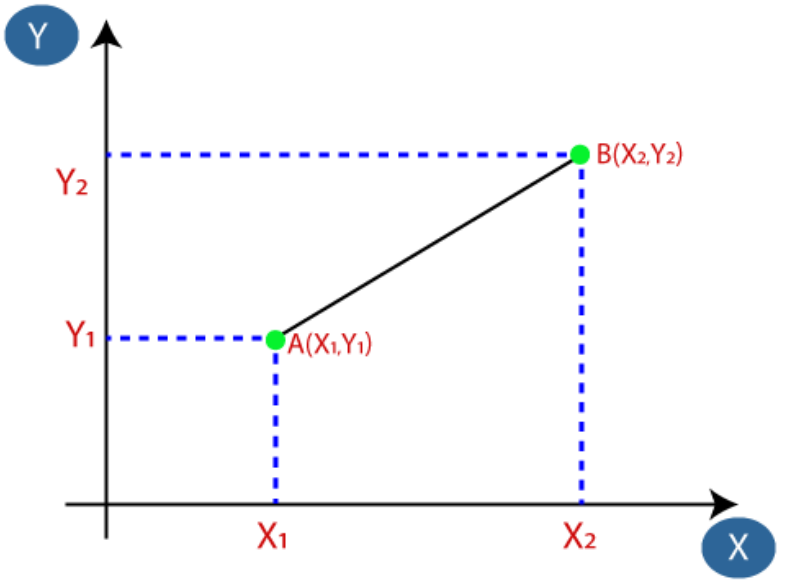
- **Step-1:** Select the number K of the neighbours
- **Step-2:** Calculate the Euclidean distance of **K number of neighbours**
- **Step-3:** Take the K nearest neighbours as per the calculated Euclidean distance.

- **Step-4:** Among these k neighbours, 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 neighbour 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 neighbours, 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 neighbours, as three nearest neighbours in category A and two nearest neighbours in category B. Consider the below image:

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

How to select the value of K in the K-NN Algorithm?

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.

Advantages of KNN Algorithm:

- 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:

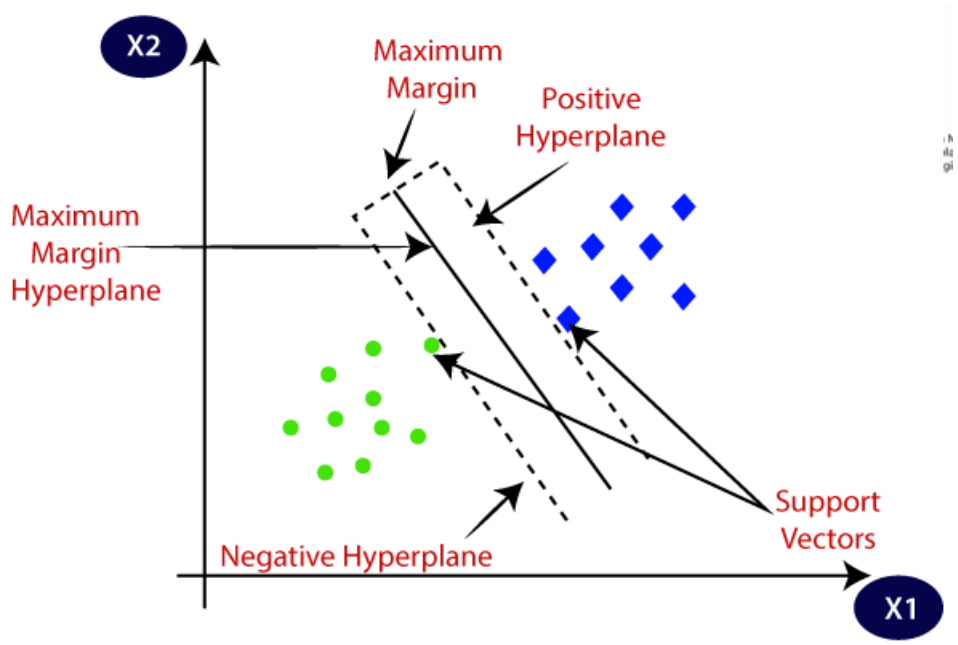
- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Support Vector Machine Algorithm

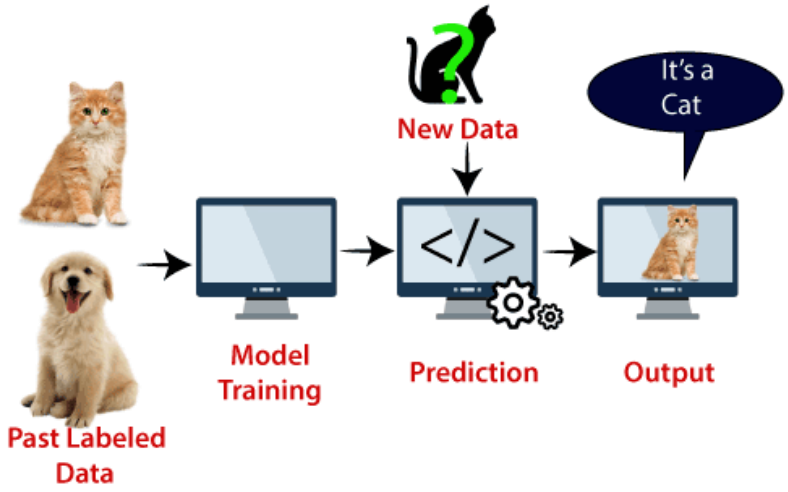
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

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.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

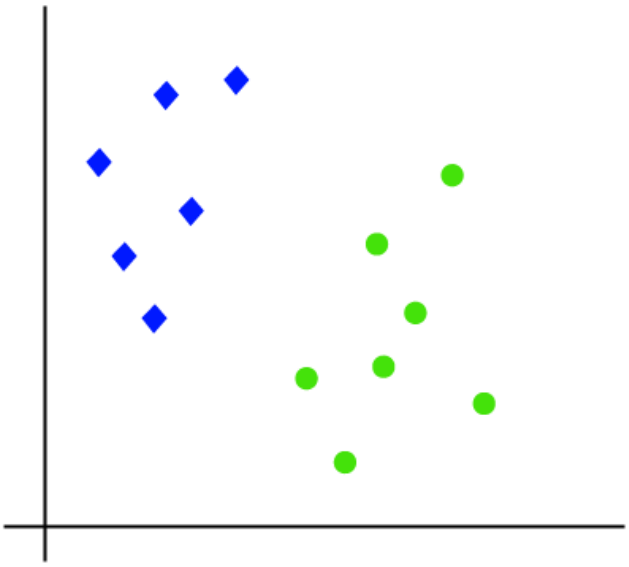
Support Vectors:

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.

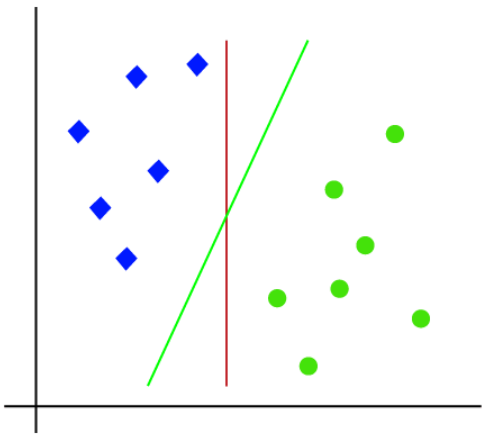
How does SVM works?

Linear SVM:

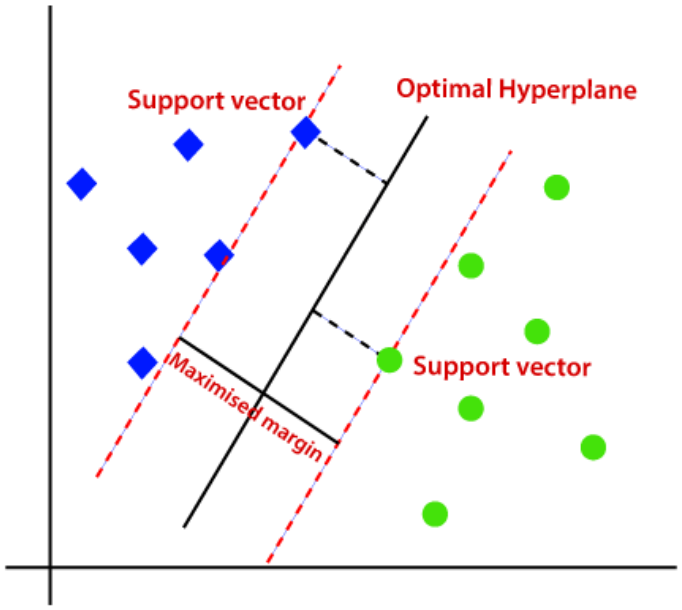
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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:

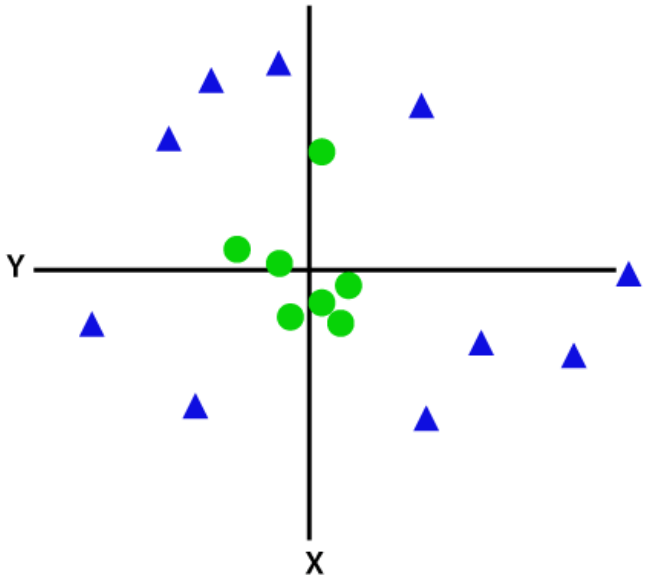


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane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

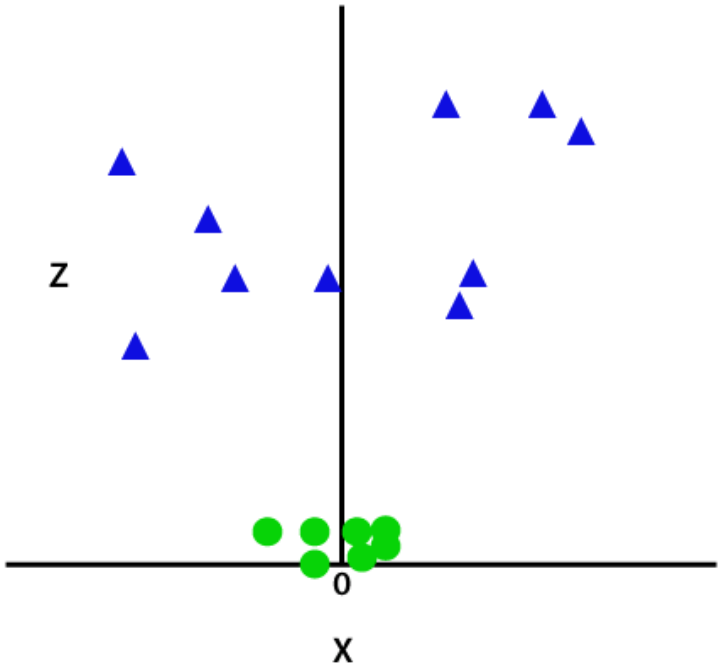
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



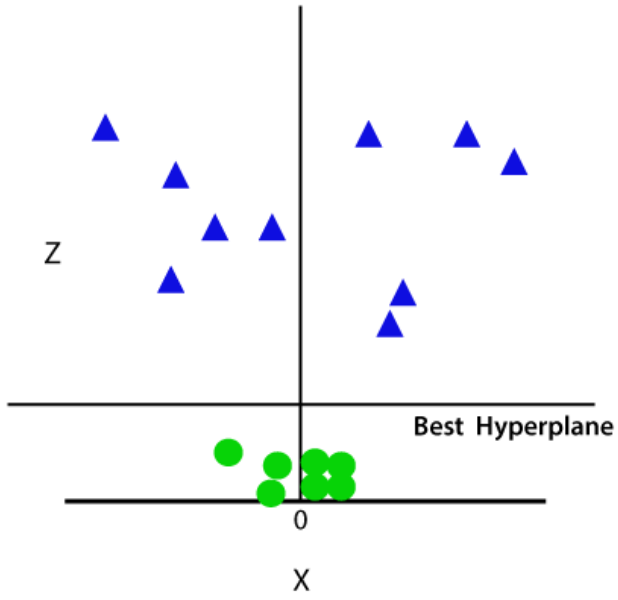
So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as:

$z=x^2 +y^2$

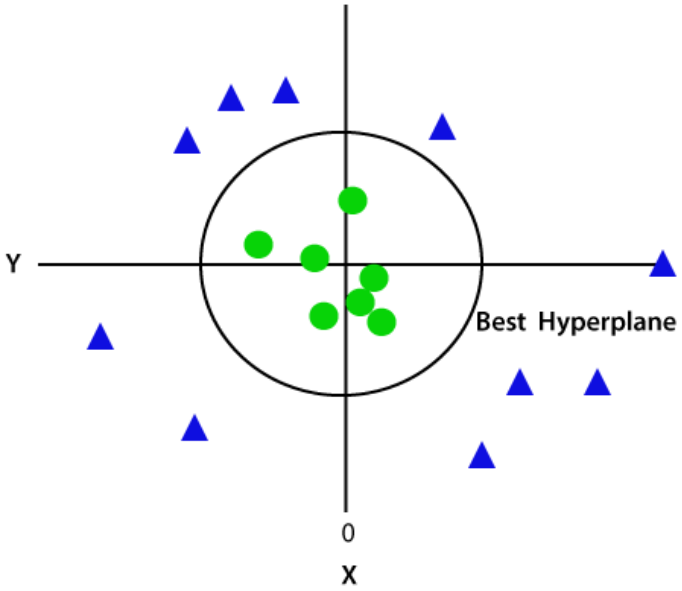
By adding the third dimension, the sample space will become as below image:



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.