

UNIT – 1



Machine Learning

❖ Introduction of ML:

Machine learning is a subfield of artificial intelligence that involves the development of algorithms and statistical models that enable computers to improve their performance in tasks through experience. These algorithms and models are designed to learn from data and make predictions or decisions without explicit instructions.

There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning involves training a model on labeled data, while unsupervised learning involves training a model on unlabeled data.

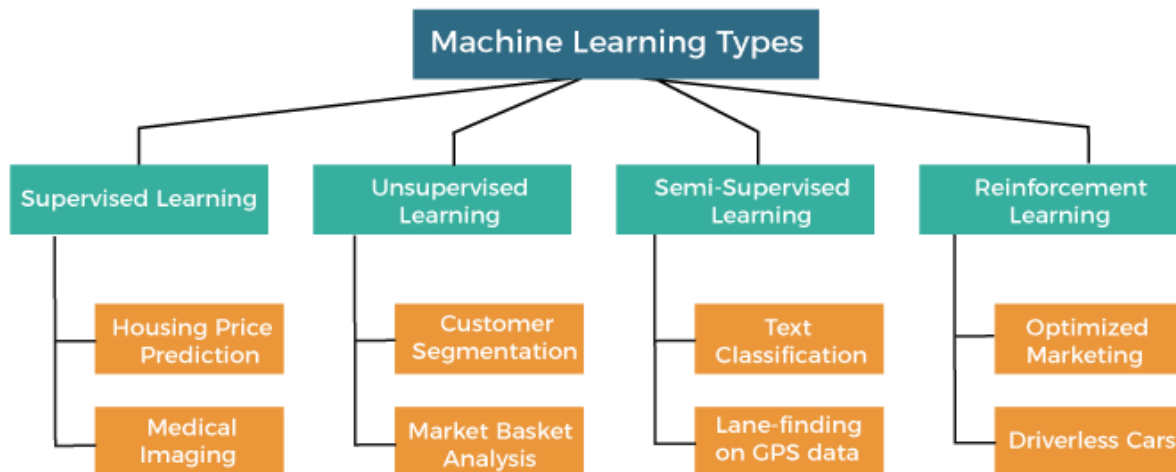
Reinforcement learning involves training a model through trial and error.

Machine learning is used in a wide variety of applications, including image and speech recognition, natural language processing, and recommender systems.

❖ Applications: -

1. Image Recognition.
2. Speech Recognition.
3. Traffic Prediction.
4. Product recommendations.
5. Self-driving cars.
6. Email spam and Malware filtering.
7. Virtual personal assistant.
8. Online fraud detection.
9. Stock market trading.
10. Medical diagnosis.
11. Automatic Language Translation.

Types of Learning



1. Supervised Machine Learning:

As its name suggests, Supervised machine learning is based on supervision.

It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output.

Here, the labelled data specifies that some of the inputs are already mapped to the output.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y).

❖ Types of Supervised Learning:

1) Classification:

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as **"Yes" or No, Male or Female, Red or Blue, etc.**

The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

1. Random Forest Algorithm
2. Decision Tree Algorithm
3. Logistic Regression Algorithm
4. Support Vector Machine Algorithm

2) Regression:

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables.

These are used to predict continuous output variables, such as market trends, weather prediction, etc. Some popular Regression algorithms are given below:

1. Simple Linear Regression Algorithm
2. Multivariate Regression Algorithm
3. Decision Tree Algorithm
4. Lasso Regression

❖ **Advantages:**

- 1) Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.
- 2) These algorithms are helpful in predicting the output based on prior experience.

❖ **Disadvantages:**

1. These algorithms are not able to solve complex tasks.
2. It may predict the wrong output if the test data is different from the training data.
3. It requires lots of computational time to train the algorithm.

❖ **Applications of Supervised Learning:**

1. **Image Segmentation** - Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
2. **Medical Diagnosis** - Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
3. **Fraud Detection** - Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
4. **Spam detection** - In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
5. **Speech Recognition** - Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

2. Unsupervised Machine Learning:

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision.

It means, in unsupervised machine learning, the machine is trained using the unlabelled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences.

Machines are instructed to find the hidden patterns from the input dataset.

❖ **Types of Unsupervised machine learning:**

1) **Clustering:**

The clustering technique is used when we want to find the inherent groups from the data.

It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups.

Some of the popular clustering algorithms are given below:

1. K-Means Clustering algorithm
2. Mean-shift algorithm
3. DBSCAN Algorithm
4. Principal Component Analysis
5. Independent Component Analysis

2) **Association:**

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset.

The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit.

This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm**.

❖ **Advantages:**

1. These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabelled dataset.
2. Unsupervised algorithms are preferable for various tasks as getting the unlabelled dataset is easier as compared to the labelled dataset.

❖ **Disadvantages:**

1. The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
2. Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

❖ **Applications of Unsupervised machine learning:**

1. **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
2. **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
3. **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
4. **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

3. **Semi-Supervised Learning:**

The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning.

Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data.

It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

❖ **Advantages:**

1. It is simple and easy to understand the algorithm.
2. It is highly efficient.
3. It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

❖ **Disadvantages:**

1. Iterations results may not be stable.
2. We cannot apply these algorithms to network-level data.
3. Accuracy is low.

4. **Reinforcement Learning:**

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

Due to its way of working, reinforcement learning is employed in different fields such as **Game theory, Operation Research, Information theory, multi-agent systems.**

A reinforcement learning problem can be formalized using **Markov Decision Process (MDP).**

❖ **Types of Reinforcement Learning:**

- a. **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something.
It enhances the strength of the behaviour of the agent and positively impacts it.
- b. **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL.
It increases the tendency that the specific behaviour would occur again by avoiding the negative condition.

❖ **Applications of Reinforcement Learning:**

1. **Video Games:**
RL algorithms are much popular in gaming applications. It is used to gain super-human performance. Some popular games that use RL algorithms are **AlphaGO** and **AlphaGO Zero**.
2. **Resource Management:**
The "Resource Management with Deep Reinforcement Learning" paper showed that how to use RL in computer to automatically learn and schedule resources to wait for different jobs in order to minimize average job slowdown.
3. **Robotics:**
RL is widely being used in Robotics applications. Robots are used in the industrial and manufacturing area, and these robots are made more powerful with reinforcement learning. There are different industries that have their vision of building intelligent robots using AI and Machine learning technology.
4. **Text Mining**
Text-mining, one of the great applications of NLP, is now being implemented with the help of Reinforcement Learning by Salesforce company.

❖ **Advantages:**

1. It helps in solving complex real-world problems which are difficult to be solved by general techniques.
2. The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
3. Helps in achieving long term results.

❖ **Disadvantage:**

1. RL algorithms are not preferred for simple problems.
2. RL algorithms require huge data and computations.
3. Too much reinforcement learning can lead to an overload of states which can weaken the results.

Linear Regression Model

Linear regression is one of the easiest and most popular Machine Learning algorithms.

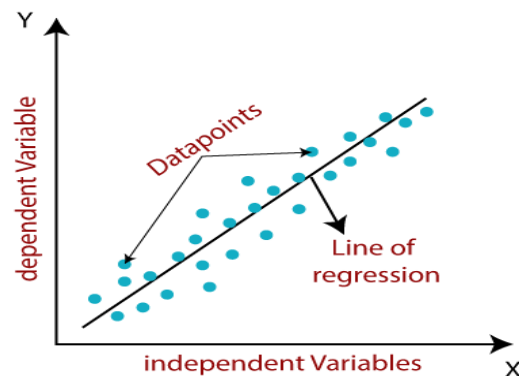
It is a statistical method that is used for predictive analysis.

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 (x) 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.



Mathematically, we can represent a linear regression as:

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

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a_0 = intercept of the line (Gives an additional degree of freedom)

a_1 = 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:

1. 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.

2. Multiple Linear regression:

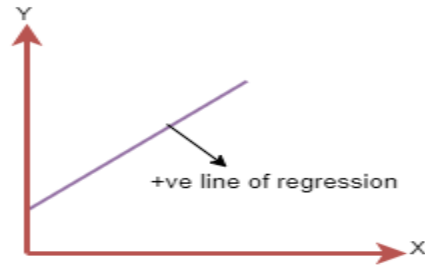
If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

❖ 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:

1. **Positive Linear Relationship:**

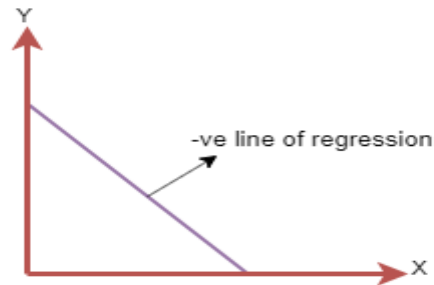
If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



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

2. **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: $Y = -a_0 + a_1x$

❖ **Applications: -**

1. Economic Growth
2. Product price
3. Housing sales
4. Score prediction

Naïve Bayes Classifier

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 based on the probability of an object.

Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles.**

❖ 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.

❖ Advantages of Naïve Bayes Classifier:

1. Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
2. It can be used for Binary as well as Multi-class Classifications.
3. It performs well in multi-class predictions as compared to the other Algorithms.
4. It is the most popular choice for **text classification problems.**
5. Very simple and easy to implement.
6. Needs less training data.
7. Handles both continuous and discrete data.
8. Highly scalable with number of predictors and data points.
9. As it fast, it can be used in real time predictions.
10. Non sensitive to irrelevant features.

❖ Disadvantages of Naïve Bayes Classifier:

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

❖ Applications of Naïve Bayes Classifier:

1. It is used for Credit Scoring.
2. It is used in medical data classification.
3. It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
4. It is used in Text classification such as Spam filtering and Sentiment analysis.
5. Face Recognition.
6. Weather Prediction.
7. Medical Diagnosis.
8. News Classification.

Decision Tree

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**.

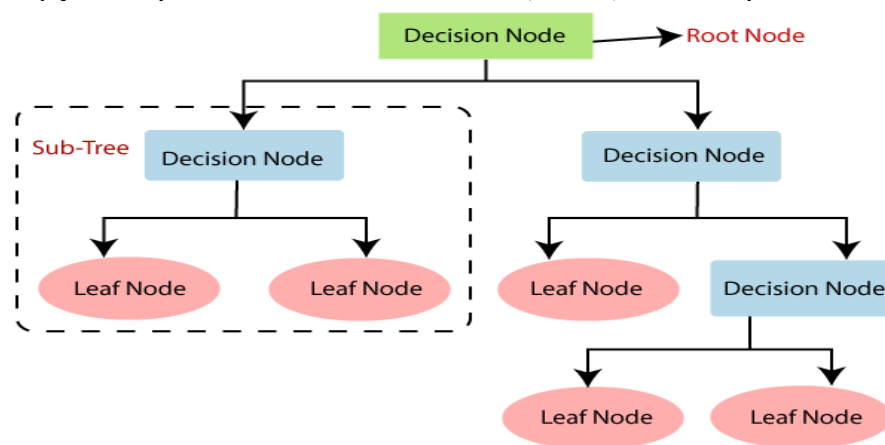
Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given data set.

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.



❖ Decision Tree Terminologies:

1. **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.
2. **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3. **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4. **Branch/Sub Tree:** A tree formed by splitting the tree.
5. **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
6. **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.

❖ **Algorithm:**

1. **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
2. **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.
3. **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
4. **Step-4:** Generate the decision tree node, which contains the best attribute.
5. **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.

❖ **Advantage of decision tree: -**

1. Simple to understand, interpret and visualize.
2. Little effort required for data preparation.
3. Can handle both numerical and categorical data.
4. Non-linear parameters do not affect its performance.

❖ **Disadvantage of decision tree: -**

1. Overfitting occurs when the algorithm captures noise in the data.
2. The model can get unstable due to small variation in data.
3. A highly complicated Decision tree tends to have a low bias which makes it difficult for the model to work with new data.

❖ **Important Terms: -**

1. Entropy: - Entropy is the measure of randomness or unpredictability in the dataset.
2. Information Gain: - It is the measure of decrease in entropy after the dataset s split.
3. Leaf Node: - Leaf node carries the classification or the decision.
4. Root Node: - The top most decision node is known as the root node.

K-Nearest Neighbor (KNN)

K-Nearest Neighbor 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.



❖ How does K-NN work?

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

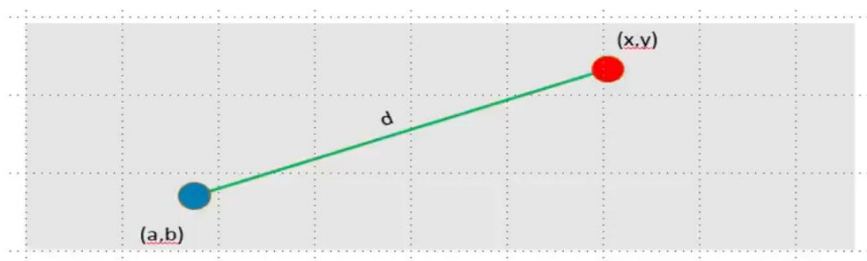
❖ How to select the value of K in the K-NN Algorithm?

1. 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.
2. 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.
3. Large values for K are good, but it may find some difficulties.
4. $\text{Sqrt}(n)$, where n is the total number of data points.
5. Odd value of k is selected to avoid confusion between two classes of data.

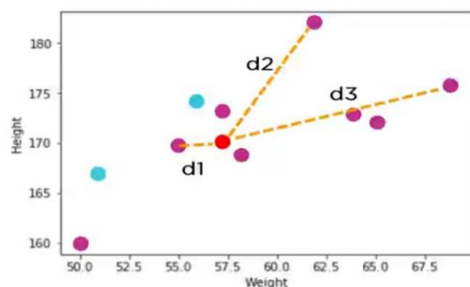
❖ How to calculate Euclidean distance?

According to the **Euclidean distance** formula, the **distance** between two points in the plane with coordinates (x, y) and (a, b) is given by:

$$\text{dist}(d) = \sqrt{(x - a)^2 + (y - b)^2}$$



Let's calculate it to understand clearly:



$$\text{dist}(d1) = \sqrt{(170-167)^2 + (57-51)^2} \approx 6.7$$

$$\text{dist}(d2) = \sqrt{(170-182)^2 + (57-62)^2} \approx 13$$

$$\text{dist}(d3) = \sqrt{(170-176)^2 + (57-69)^2} \approx 13.4$$

Similarly, we will calculate Euclidean distance of unknown data point from all the points in the dataset

● Unknown data point

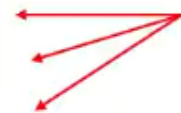
Hence, we have calculated the Euclidean distance of unknown data point from all the points as shown:

Where (x1, y1) = (57, 170) whose class we have to classify

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

Now, let's calculate the nearest neighbor at $k=3$

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

 $k = 3$

57 kg	170 cm	?
-------	--------	---

❖ **Advantages of KNN Algorithm:**

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

❖ **Disadvantages of KNN Algorithm:**

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

Logistic Regression

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.

Used to solve classification problems.

The response variable is categorical in nature.

It helps calculate the possibility of a particular event taking place.

An S-curve. (S = sigmoid).

A dataset with one or more independent variables is used to determine binary output of the dependent variable.

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 like 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).

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.

❖ Assumptions for Logistic Regression:

1. The dependent variable must be categorical in nature.
2. The independent variable should not have multi-collinearity.

❖ Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation.

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

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

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

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

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

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

❖ **Type of Logistic Regression:**

1. **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.
2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
3. **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

❖ **Application of logistic regression: -**

1. Helps determine the kind of weather that can be expected.
2. Identifies the different components that are present in the image, and helps categorize them.
3. Determine the possibility of patient survival, taking age, ISS and RTS into consideration.

Support Vector Machine

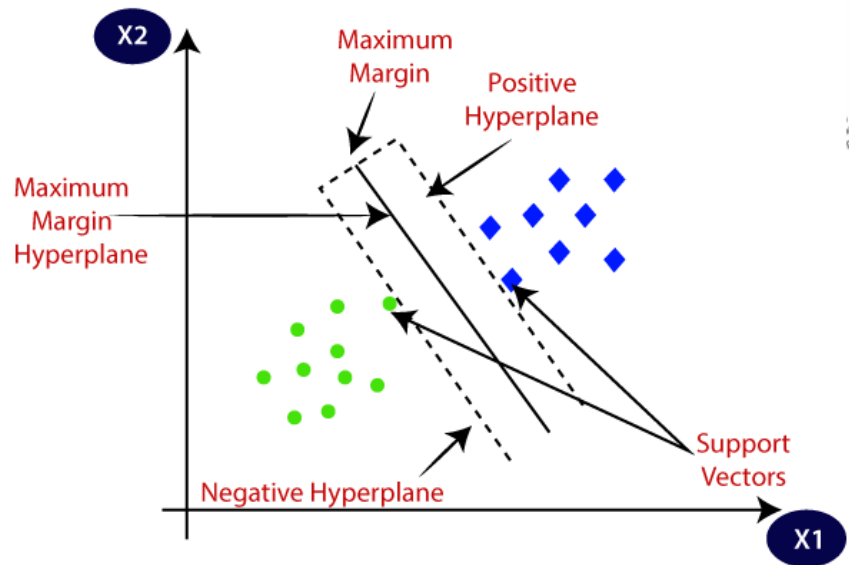
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

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.



❖ Types of SVM:

1. **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.
2. **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.

➤ 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.

Random Forest Algorithm

❖ Why Random Forest?

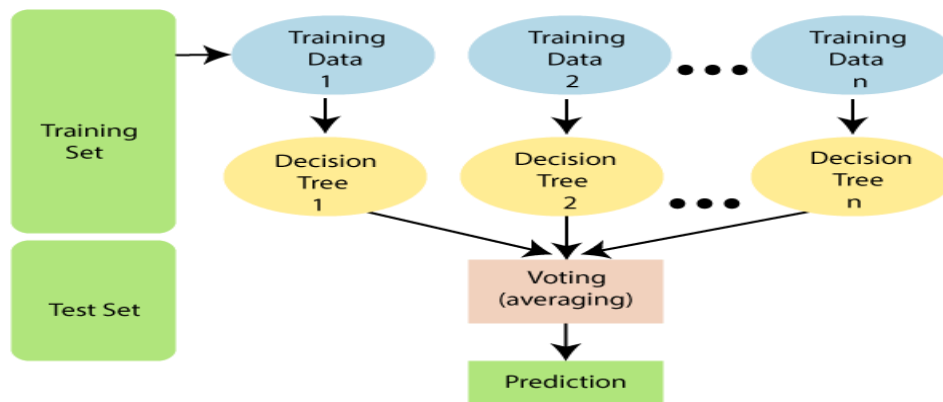
1. No overfitting: Use of multiple trees reduce the risk of overfitting. Training time is less.
2. High Accuracy: Runs efficiently on large database. For large data, it produces highly accurate predictions.
3. Estimates missing data: Random Forest can maintain accuracy when a large proportion of data is missing.

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.

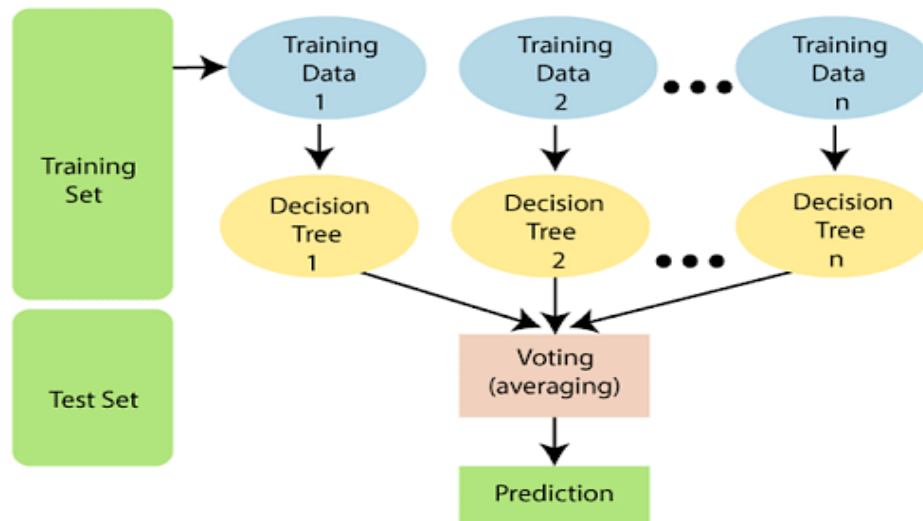
It is based on the concept of **ensemble learning**, which is a process of **combining multiple classifiers to solve a complex problem and to improve the performance of the model**.

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.*"

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



❖ Working of Random Forest Algorithm:

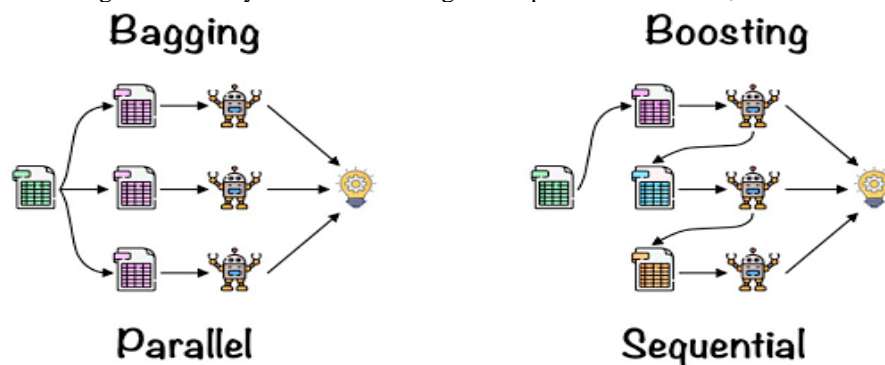


➤ The following steps explain the working Random Forest Algorithm:

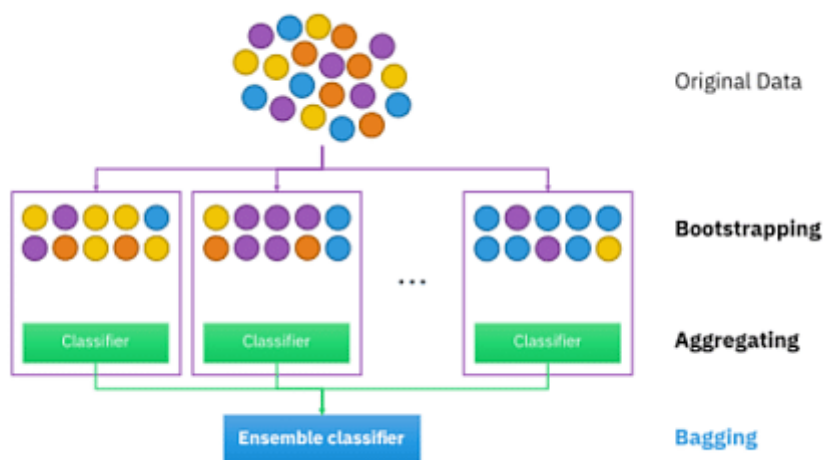
- Step 1: Select random samples from a given data or training set.
- Step 2: This algorithm will construct a decision tree for every training data.
- Step 3: Voting will take place by averaging the decision tree.
- Step 4: Finally, select the most voted prediction result as the final prediction result.

❖ **This combination of multiple models is called Ensemble. Ensemble uses two methods:**

1. **Bagging:** Creating a different training subset from sample training data with replacement is called Bagging. The final output is based on majority voting.
2. **Boosting:** Combining weak learners into strong learners by creating sequential models such that the final model has the highest accuracy is called Boosting. Example: ADA BOOST, XG BOOST.



Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organised into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.



❖ **Assumptions for Random Forest:**

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

❖ 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.

1. **Step-1:** Select random K data points from the training set.
2. **Step-2:** Build the decision trees associated with the selected data points (Subsets).
3. **Step-3:** Choose the number N for decision trees that you want to build.
4. **Step-4:** Repeat Step 1 & 2.
5. **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.

❖ 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.
5. **Remote Sensing:** Used in ETM devices to acquire images of the earth's surface. Accuracy is higher and training time is less.
6. **Object Detection:** Multiclass object detection is done using Random Forest algorithms. Provides better detection in complicated environments.
7. **Kinect:** Random Forest is used in a game console called Kinect. Tracks body movements and recreates it in the game.

❖ Advantages of Random Forest:

1. Random Forest can perform both Classification and Regression tasks.
2. It is capable of handling large datasets with high dimensionality.
3. It enhances the accuracy of the model and prevents the overfitting issue.

❖ Disadvantages of Random Forest:

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

UNIT – 2



Unsupervised Learning

In unsupervised learning, you train the machine using data which is “unlabelled” and models itself find the hidden patterns and insights from the given data.
The goal of Unsupervised learning is to group unlabelled data according to the similarities, patterns and differences without any prior training of data.

❖ Types of Unsupervised learning: -

1. Clustering.
2. Association.

❖ Unsupervised Learning Algorithm: -

1. K-Means Clustering.
2. KNN (K-Nearest Neighbour).
3. Hierarchical Clustering.
4. Neural Networks/Deep Learning.
5. Single Value Decomposition.
6. Distribution Models.
7. Principal Component Analysis.
8. Apriori Algorithm.

❖ Advantages: -

1. Used for more complex tasks.
2. It's helpful in finding patterns in data.
3. Saves lot of manual work and expense.

❖ Disadvantages: -

1. Less Accuracy.
2. Time Consuming.
3. More the features, More the complexity.

K-Means Clustering

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

❖ What is K-Means 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 unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

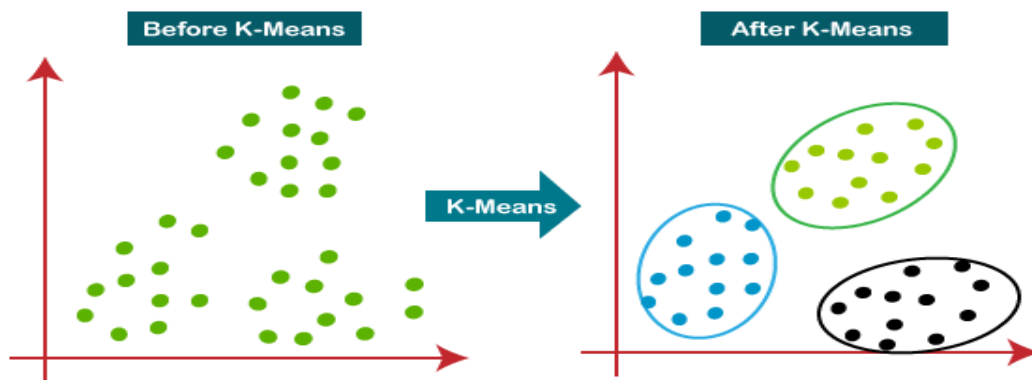
It is a centroid-based algorithm, where each cluster is associated with a centroid.

The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

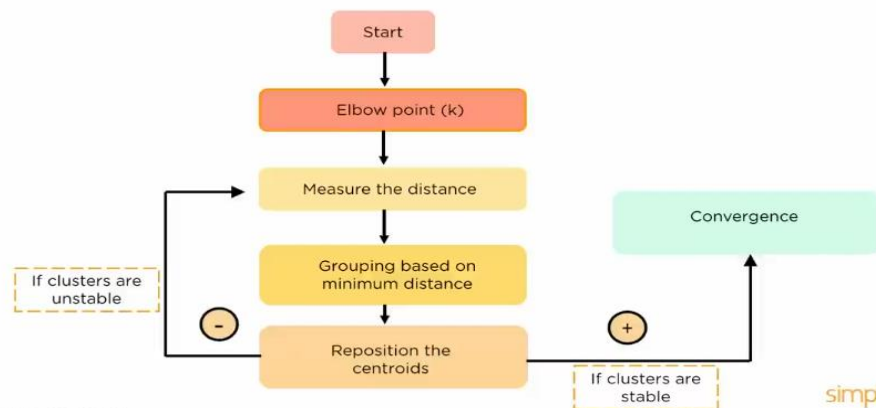
The algorithm takes the unlabelled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

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



How does K-Means clustering work?



©Simplilearn. All rights reserved.

simplilearn

❖ **How does the K-Means Algorithm Work?**

1. **Step-1:** Select the number K to decide the number of clusters.
2. **Step-2:** Select random K points or centroids. (It can be other from the input dataset).
3. **Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
4. **Step-4:** Calculate the variance and place a new centroid of each cluster.
5. **Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
6. **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
7. **Step-7:** The model is ready.

❖ **Applications: -**

1. Academic Performance.
2. Diagnostic Systems.
3. Search Engines.
4. Wireless Sensor Network's.

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**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

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**.

❖ Why hierarchical clustering?

As we already have other clustering algorithms such as **K-Means Clustering**, then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

❖ Agglomerative Hierarchical clustering:

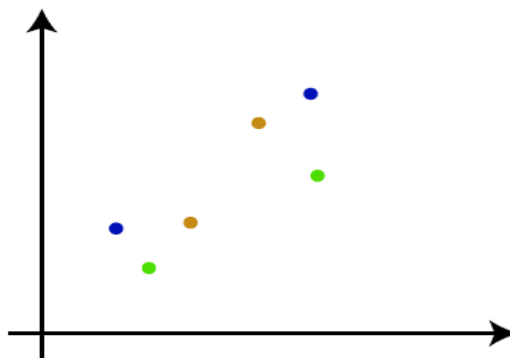
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together.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

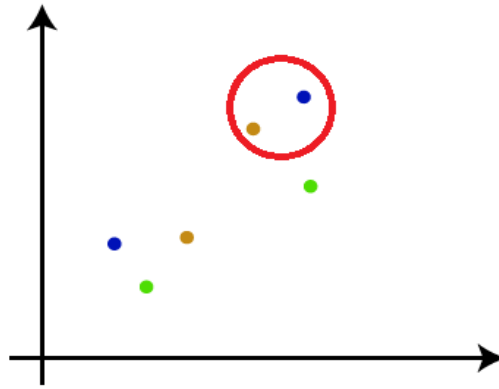
❖ How the Agglomerative Hierarchical clustering Work?

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

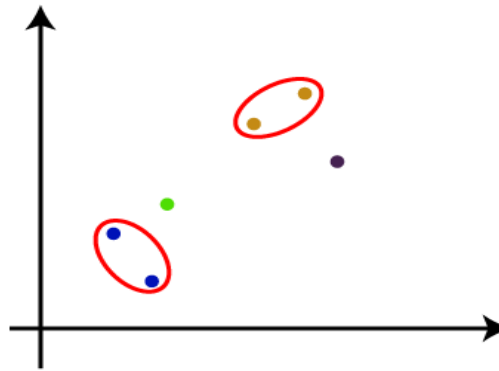
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



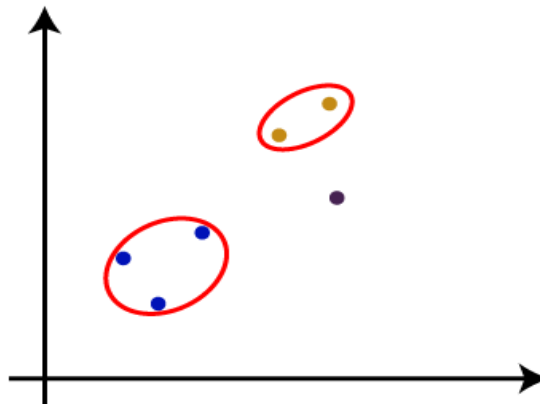
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be $N-1$ clusters.

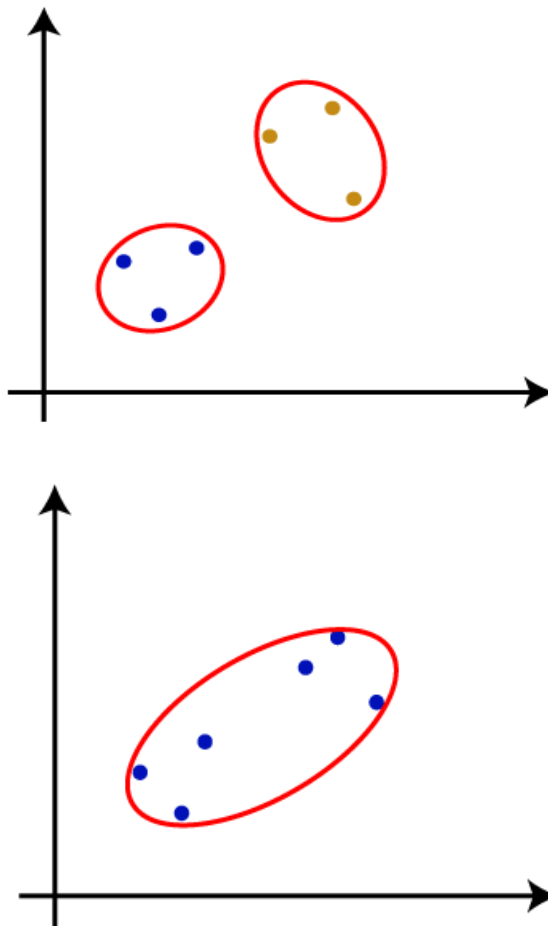


- **Step-3:** Again, take the two closest clusters and merge them together to form one cluster. There will be $N-2$ clusters.



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



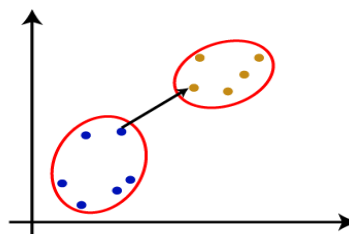


- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

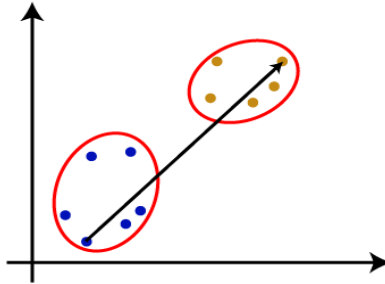
❖ Measure for the distance between two clusters:

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:

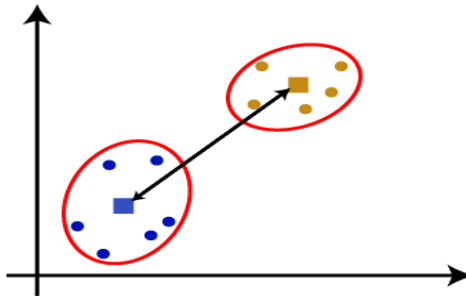
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:

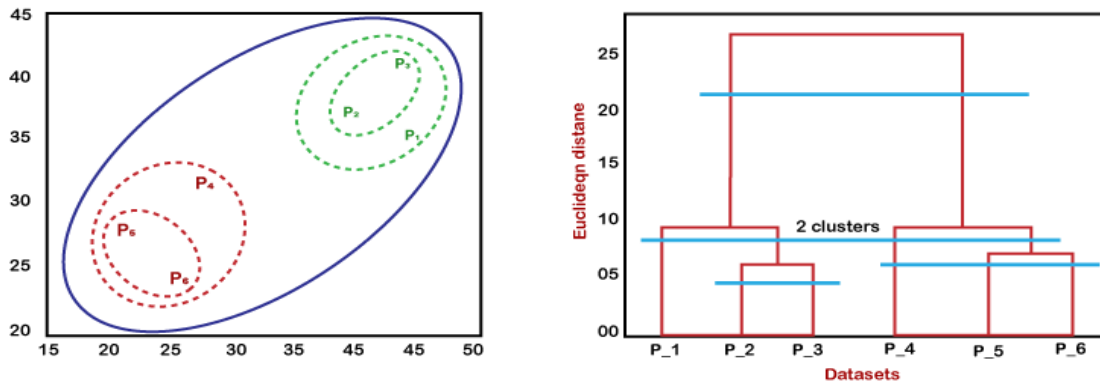


From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

❖ Working 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 height 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.

We can cut the dendrogram tree structure at any level as per our requirement.

❖ Python Implementation of Agglomerative Hierarchical Clustering:

Now we will see the practical implementation of the agglomerative hierarchical clustering algorithm using Python. To implement this, we will use the same dataset problem that we have used in the previous topic of K-means clustering so that we can compare both concepts easily.

The dataset is containing the information of customers that have visited a mall for shopping. So, the mall owner wants to find some patterns or some behaviour of his customers using the dataset information.

❖ Steps for implementation of AHC using Python:

The steps for implementation will be the same as the k-means clustering, except for some changes such as the method to find the number of clusters. Below are the steps:

1. **Data Pre-processing.**
2. **Finding the optimal number of clusters using the Dendrogram.**
3. **Training the hierarchical clustering model.**
4. **Visualizing the clusters.**

❖ Applications of clustering: -

1. Customer Segmentation.
2. Social Network Analysis.

Association Rule Mining

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable.

It tries to find some interesting relations or associations among the variables of dataset.

The association rule learning is one of the very important concepts of machine learning, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.**

Association rule learning can be divided into three types of algorithms:

1. Apriori.
2. F-P Growth Algorithm.
3. Gaussian Mixture Model.

Apriori Algorithm

The Apriori algorithm uses frequent item sets to generate association rules, and it is designed to work on the databases that contain transactions.

With the help of these association rule, it determines how strongly or how weakly two objects are connected.

This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently.

It is the iterative process for finding the frequent item sets from the large dataset.

❖ Steps for Apriori Algorithm:

1. **Step-1:** Determine the support of itemsets in the transactional database, and select the minimum support and confidence.
2. **Step-2:** Take all supports in the transaction with higher support value than the minimum or selected support value.
3. **Step-3:** Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.
4. **Step-4:** Sort the rules as the decreasing order of lift.

❖ Advantages of Apriori Algorithm:

1. This is easy to understand algorithm.
2. The join and prune steps of the algorithm can be easily implemented on large datasets.

❖ Disadvantages of Apriori Algorithm:

1. The apriori algorithm works slow compared to other algorithms.
2. The overall performance can be reduced as it scans the database for multiple times.
3. The time complexity and space complexity of the apriori algorithm is $O(2^D)$, which is very high. Here D represents the horizontal width present in the database.

F-P Growth Algorithm

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance.

For so much, it uses a divide-and-conquer strategy.

The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

Using this strategy, the FP-Growth reduces the search costs by recursively looking for short patterns and then concatenating them into the long frequent patterns.

❖ This algorithm works as follows:

1. First, it compresses the input database creating an FP-tree instance to represent frequent items.
2. After this first step, it divides the compressed database into a set of conditional databases, each associated with one frequent pattern.
3. Finally, each such database is mined separately.

❖ FP-Tree:

The frequent-pattern tree (FP-tree) is a compact data structure that stores quantitative information about frequent patterns in a database.

Each transaction is read and then mapped onto a path in the FP-tree.

This is done until all transactions have been read.

Different transactions with common subsets allow the tree to remain compact because their paths overlap.

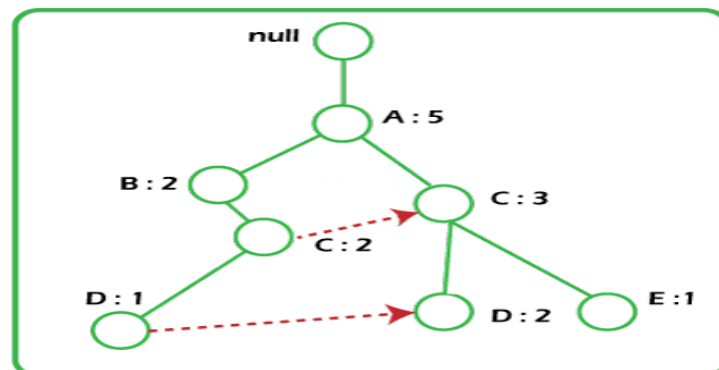
A frequent Pattern Tree is made with the initial item sets of the database. Each node of the FP tree represents an item of the item set.

The root node represents null, while the lower nodes represent the item sets.

➤ FP-tree as the tree structure given below:

1. One root is labelled as "null" with a set of item-prefix subtrees as children and a frequent-item-header table.
2. Each node in the item-prefix subtree consists of three fields:
 - Item-name: registers which item is represented by the node;
 - Count: the number of transactions represented by the portion of the path reaching the node;
 - Node-link: links to the next node in the FP-tree carrying the same item name or null if there is none.
3. Each entry in the frequent-item-header table consists of two fields:
 - Item-name: as the same to the node;
 - Head of node-link: a pointer to the first node in the FP-tree carrying the item name.

Additionally, the frequent-item-header table can have the count support for an item.



The worst-case scenario occurs when every transaction has a unique item set.

So the space needed to store the tree is greater than the space used to store the original data set because the FP-tree requires additional space to store pointers between nodes and the counters for each item.

❖ **FP-Growth Algorithm:**

After constructing the FP-Tree, it's possible to mine it to find the complete set of frequent patterns.

Input: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold?

Output: The complete set of frequent patterns.

Method: Call FP-growth (FP-tree, null).

#1) The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.

#2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

#3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.

#4) The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.

This means that the common itemset is linked to the new node of another itemset in this transaction.

#5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.

#6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base.

Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

#7) Construct a Conditional FP Tree, which is formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.

#8) Frequent Patterns are generated from the Conditional FP Tree.

❖ **Advantages of FP Growth Algorithm:**

1. This algorithm needs to scan the database twice when compared to Apriori, which scans the transactions for each iteration.
2. The pairing of items is not done in this algorithm, making it faster.
3. The database is stored in a compact version in memory.
4. It is efficient and scalable for mining both long and short frequent patterns.

❖ **Disadvantages of FP-Growth Algorithm:**

1. FP Tree is more cumbersome and difficult to build than Apriori.
2. It may be expensive.
3. The algorithm may not fit in the shared memory when the database is large.

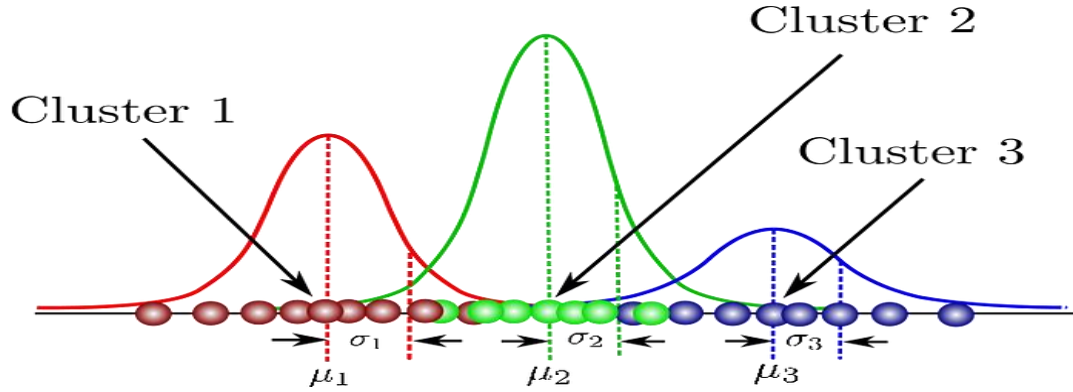
Gaussian Mixture Model

It is a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to allowing the model to learn subpopulations automatically.

Since subpopulation assignment is not known, this constitutes a form of unsupervised learning.

A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1, \dots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.



Here, we can see that there are three Gaussian functions, hence $K = 3$.

Each Gaussian explains the data contained in each of the three clusters available.

The mixing coefficients are themselves probabilities and must meet this condition:

$$\sum_{k=1}^K \pi_k = 1 \quad (1)$$

In general, the Gaussian density function is given by:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

Where \mathbf{x} represents our data points, D is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively. If we have a dataset comprised of $N = 1000$ three-dimensional points ($D = 3$), then \mathbf{x} will be a 1000×3 matrix. μ will be a 1×3 vector, and Σ will be a 3×3 matrix. For later purposes, we will also find it useful to take the log of this equation, which is given by:

$$\ln \mathcal{N}(\mathbf{x}|\mu, \Sigma) = -\frac{D}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma| - \frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu) \quad (2)$$

If we differentiate this equation with respect to the mean and covariance and then equate it to zero, then we will be able to find the optimal values for these parameters, and the solutions will correspond to the Maximum Likelihood Estimates (MLE) for this setting. However, because we are dealing with not just one, but many Gaussians, things will get a bit complicated when time comes for us to find the parameters for the whole mixture. In this regard, we will need to introduce some additional aspects that we discuss in the next section.

UNIT – 3

Feature extraction – Principal component analysis

PCA is a dimensionality reduction technique that has four main parts: feature covariance, eigen decomposition, principal component transformation, and choosing components in terms of explained variance.

❖ A Quick Review of Dimensionality Reduction:

➤ Curse of Dimensionality:

“As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.”

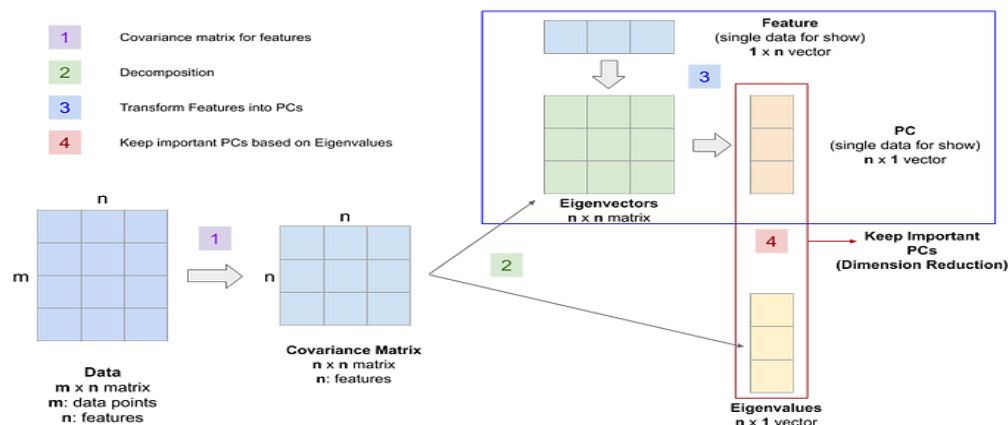
There are two options to reduce dimensionality:

1. Feature elimination: we remove some features directly.
2. Feature extraction: we keep the important fraction of all the features. We apply PCA to achieve this. Note that PCA is not the only method that does the feature extraction.

➤ PCA:

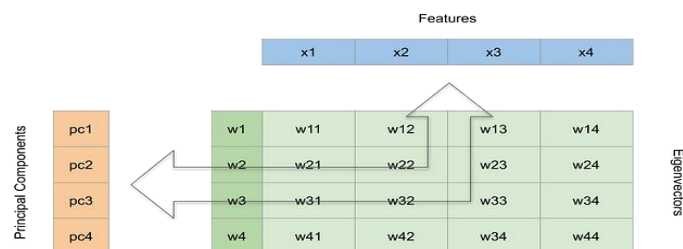
PCA is a dimensionality reduction that **identifies important relationships** in our data, **transforms the existing data** based on these relationships, and then **quantifies the importance** of these relationships so we can keep the most important relationships and drop the others. To remember this definition, we can break it down into four steps:

1. We identify the relationship among features through a **Covariance Matrix**.
2. Through the linear transformation or **eigen decomposition** of the Covariance Matrix, we get **eigenvectors** and **eigenvalues**.
3. Then we transform our data using Eigenvectors into principal components.
4. Lastly, we quantify the importance of these relationships using Eigenvalues and keep the important principal components.



❖ A Simplified Visual Demo:

The following demo presents the linear transformation between features and principal components using eigenvectors for a single data point from the Iris database. I describe the calculations without using any linear algebra terms. However, it would be helpful if you understand the dot product between two vectors (since we demonstrate the transformation for a single data point) and matrix multiplication (when we transform all data points).



1. **Features:** represented by the blue horizontal on the top. Note that x_1 , x_2 , x_3 , and x_4 represents the four features of a single iris (i.e., sepal length, sepal width, petal length, and petal width), not four different irises.
2. **Eigenvectors:** represented by the green matrix.
3. **Principal components:** represented by the orange vertical bar to the left.

		Features							
		x_1	x_2	x_3	x_4				
		-0.90	1.02	-1.34	-1.32				
Principal Components	pc_1	-2.26	w_1	0.52	-0.27	0.58	0.56	Eigenvectors	
	pc_2	0.48	w_2	0.38	0.92	0.02	0.07		
	pc_3	-0.13	w_3	-0.72	0.24	0.14	0.63		
	pc_4	-0.02	w_4	-0.26	0.12	0.80	-0.52		

❖ Pros:

- PCA reduces the dimensionality without losing information from any features.
- Reduce storage space needed to store data.
- Speed up the learning algorithm (with lower dimension).
- Address the multicollinearity issue (all principal components are orthogonal to each other).
- Help visualize data with high dimensionality (after reducing the dimension to 2 or 3).

❖ Cons:

- Using PCA prevents interpretation of the original features, as well as their impact because eigenvectors are not meaningful.

❖ Misuse of PCA (not an exhaustive list):

- We should not reduce dimensionality using PCA to prevent overfitting.
- We should not apply PCA blindly before running the machine learning model with original data. We should consider PCA as an alternative if using original data does not work well.

Singular value decomposition

The **singular value decomposition** helps reduce datasets containing a large number of values. Furthermore, this method is also helpful to generate significant solutions for fewer values. However, these fewer values also comprise immense variability available in the original data.

A **Singular Value Decomposition** analysis supports and yields results for a more compact demonstration of these correlations. By using multivariate datasets, you can produce insights into temporal and spatial variations. These variations exhibit data after the analysis.

Even though there are fewer limitations to the technique, you should understand these before computing the **Singular Value Decomposition** of the datasets. First, there should be anomalies in the data that the first structure will capture. If you are analysing the data to find spatial correlations independent of trends, you should de-trend the data before applying it to the analysis.

❖ Singular vectors & Singular Values:

The matrix AA^T and A^TA in linear algebra are very special. By multiplying the A^T with the matrix after considering them $\times n$ matrix A , we can form AA^T and A^TA individually. The matrices include:

- Square
- Symmetrical
- Same matrices with both positive eigenvalues
- Positive semidefinite, and
- Same r as A with both rank

A major property of symmetric matrices is that they are symmetric, and we choose eigenvectors to be orthonormal. We use these covariance matrices in machine learning a lot.

❖ Example of Singular Value Decomposition:

To understand the concept, let's suppose the matrix $m \times n$, A , collects the training data set. These sets of data will take the row for each training vector. Here, N indicates that the dimension of each vector will be very large. By feeding the A in a clustering algorithm, you will generate a fixed number of cluster centers as the output. Since ' n ' is quite large, the algorithm will be unstable or take too long. So, we will utilize **singular value decomposition** to reduce the number of variables. We will use a transparent method for computation, considering that we are still solving the problem with un-transformed coordinates.

❖ What is Feature Selection?

A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model. So, we can define feature Selection as, "It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

➤ Need for Feature Selection:

Before implementing any technique, it is really important to understand, need for the technique and so for the Feature Selection. As we know, in machine learning, it is necessary to provide a pre-processed and good input dataset in order to get better outcomes. We collect a huge amount of data to train our model and help it to learn better. Generally, the dataset consists of noisy data, irrelevant data, and some part of useful data. Moreover, the huge amount of data also slows down the training process of the model, and with noise and irrelevant data, the model may not predict and perform well. So, it is very necessary to remove such noises and less-important data from the dataset and to do this, and Feature selection techniques are used. Selecting the best features helps the model to perform well.

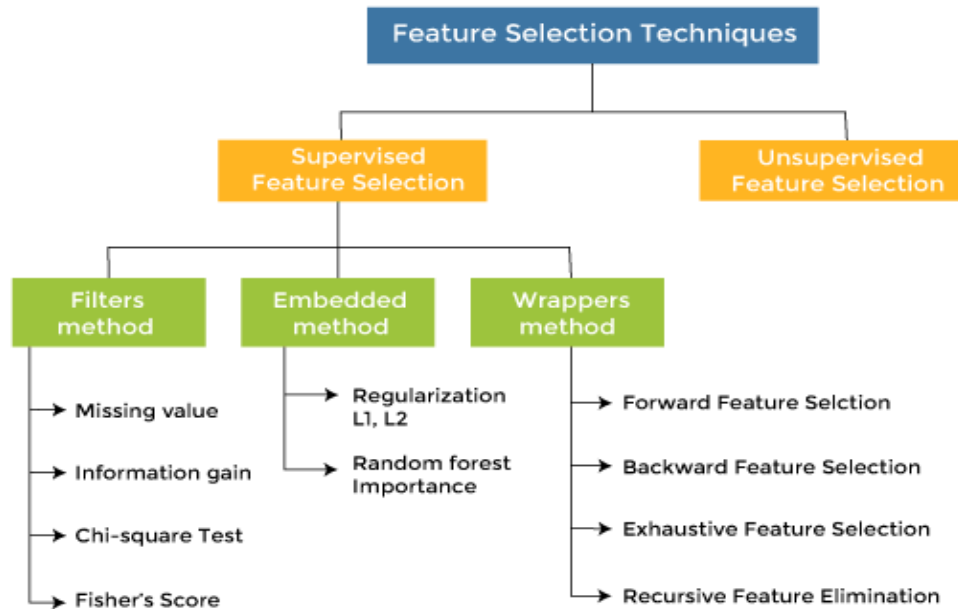
❖ Some benefits of using feature selection in machine learning:

- It helps in avoiding the curse of dimensionality.
- It helps in the simplification of the model so that it can be easily interpreted by the researchers.
- It reduces the training time.
- It reduces overfitting hence enhance the generalization.

❖ Feature Selection Techniques:

There are mainly two types of Feature Selection techniques, which are:

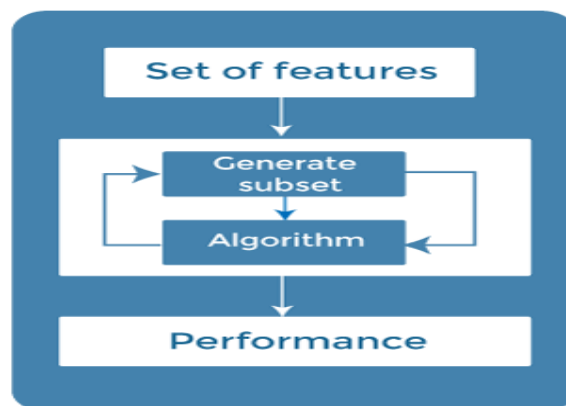
- **Supervised Feature Selection technique**
Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection technique**
Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.



➤ There are mainly three techniques under supervised feature Selection:

1. Wrapper Methods:

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

➤ **Some techniques of wrapper methods are:**

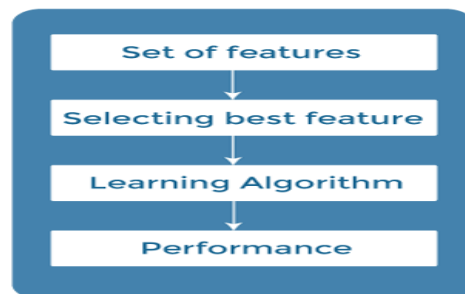
- **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new variable/feature does not improve the performance of the model.
- **Backward elimination** - Backward elimination is also an iterative approach, but it is the opposite of forward selection. This technique begins the process by considering all the features and removes the least significant feature. This elimination process continues until removing the features does not improve the performance of the model.
- **Exhaustive Feature Selection**- Exhaustive feature selection is one of the best feature selection methods, which evaluates each feature set as brute-force. It means this method tries & make each possible combination of features and return the best performing feature set.
- **Recursive Feature Elimination**- Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features. Now, an estimator is trained with each set of features, and the importance of each feature is determined using `coef_attribute` or through a `feature_importances_attribute`.

2. Filter Methods:

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.

The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.

The advantage of using filter methods is that it needs low computational time and does not overfit the data.



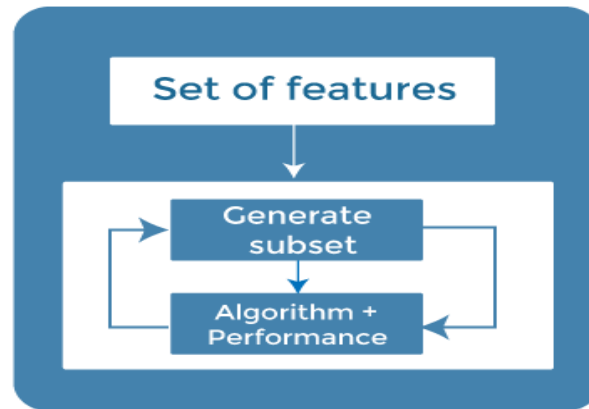
➤ **Some common techniques of Filter methods are as follows:**

- **Information Gain:** Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.
- **Chi-square Test:** Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.
- **Fisher's Score:** Fisher's score is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.
- **Missing Value Ratio:** The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.

$$\text{Missing Value Ratio} = \frac{\text{Number of Missing values} \times 100}{\text{Total number of observations}}$$

3. Embedded Methods:

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.



- These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:
 - **Regularization**- Regularization adds a penalty term to different parameters of the machine learning model for avoiding overfitting in the model. This penalty term is added to the coefficients; hence it shrinks some coefficients to zero. Those features with zero coefficients can be removed from the dataset. The types of regularization techniques are L1 Regularization (Lasso Regularization) or Elastic Nets (L1 and L2 regularization).
 - **Random Forest Importance** - Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, which is a type of bagging algorithm that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the impurity (Gini impurity) over all the trees. Nodes are arranged as per the impurity values, and thus it allows to pruning of trees below a specific node. The remaining nodes create a subset of the most important features.

Evaluating algorithm and Model selection

Model Selection and Evaluation is a hugely important procedure in the machine learning workflow. This is the section of our workflow in which we will analyse our model. We look at more insightful statistics of its performance and decide what actions to take in order to improve this model. This step is usually the difference between a model that performs well and a model that performs very well. When we evaluate our model, we gain a greater insight into what it predicts well and what it does not and this helps us turn it from a model that predicts our dataset with a 65% accuracy level to closer to 80% or 90%.

❖ Metrics and Scoring:

Let's say we have two hypotheses for a task, $h(x)$ and $h'(x)$. How would we know which one is better. Well from a high-level perspective, we might take the following steps:

1. Measure the accuracy of both hypotheses
2. Determine whether there is any statistical significance between the two results. If there is, select the better performing hypothesis. If not, we cannot say with any statistical certainty that either $h(x)$ or $h'(x)$ is better.

When we have a classification task, we will consider the accuracy of our model by its ability to assign an instance to its correct class. Consider this on a binary level. We have two classes, 1 and 0. We would classify a correct prediction therefore as being when the model classifies a class 1 instance as class 1, or a class 0 instance as class 0. Assuming our 1 class as being the 'Positive class' and the 0 class being the 'Negative class', we can build a table that outlines all the possibilities our model might produce.

		Predicted Class	
		1	0
Actual Class	1	TP	FN
	0	FP	TN

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

❖ Overfitting:

Overfitting is a key consideration when looking at the quality of a model. Overfitting occurs when we have a hypothesis that performs significantly better on its training data examples than it does on the test data examples. This is an extremely important concept in machine learning, because overfitting is one of the main features we want to avoid. For a machine learning model to be robust and effective in the 'real world', it needs to be able to predict unseen data well, it needs to be able to generalise. Overfitting essentially prevents generalisation and presents us with model that initially looks great because it fits to our training data very well, but what we ultimately find is that the model will have fit to the training data too well. The model has essentially not identified general relationships in the data, but instead has focused on figuring out exactly how to predict this one sample set.

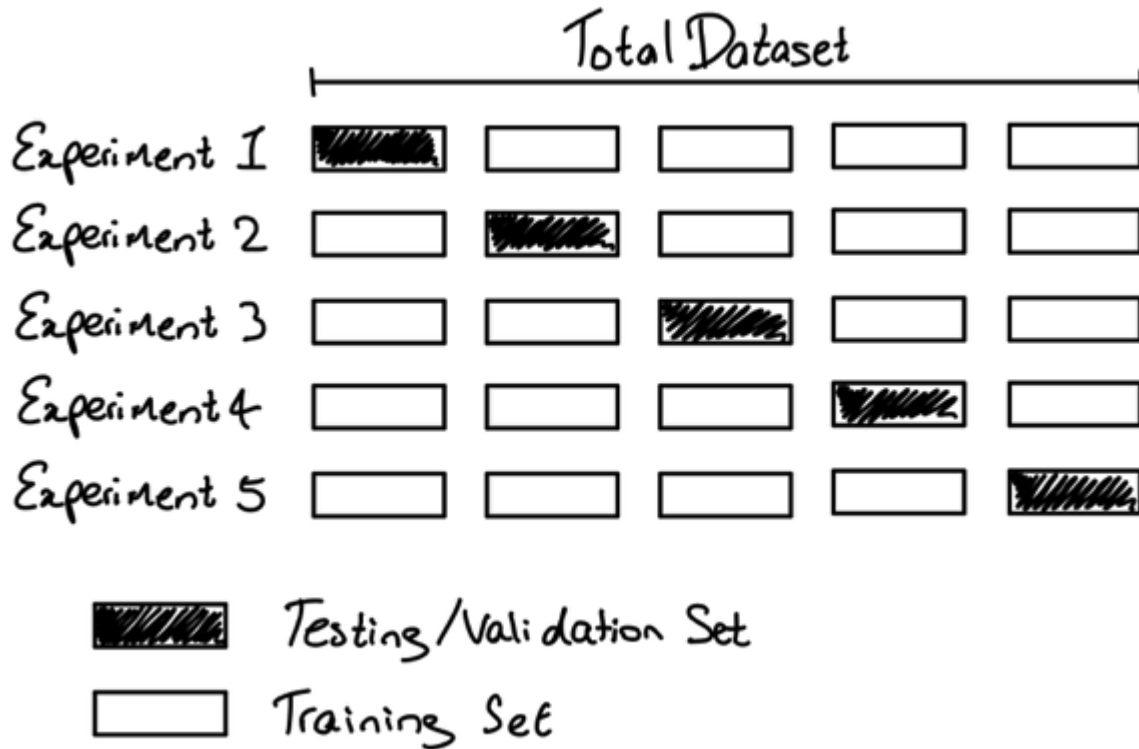
➤ This can happen for several reasons:

- The learning process is allowed to continue for too long
- The examples in the training set are not an accurate representation of the test set and therefore also of the wider picture
- Some features in the dataset are uninformative and so the model is distracted, assigning weights to these values that don't realise any positive value

❖ Cross Validation:

Cross Validation can be considered under the model improvement section. It is a particularly useful method for smaller datasets. Splitting our data into training and test data will reduce the number of samples that our model receives to train on. When we have a relatively small number of instances, this can have a large impact on the performance of our model because it does not have enough data points to study relationships and build reliable and robust coefficients. In addition to this, we make an assumption here that our training data has a similar relationship between the variables that our testing data does, because only in this situation will we actually be able to generate a reasonable model that can predict with a good level of accuracy on unseen data.

The solution that we can turn to here is a method called cross validation. In cross validation, we run our modelling process on different subsets of the data to get multiple measures of model quality. Consider the following dataset, split into 5 sections, called folds.



UNIT – 4

Reinforcement Learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behaviour in an environment to obtain maximum reward. In RL, the data is accumulated from machine learning systems that use a trial-and-error method. Data is not part of the input that we would find in supervised or unsupervised machine learning.

Reinforcement learning uses algorithms that learn from outcomes and decide which action to take next. After each action, the algorithm receives feedback that helps it determine whether the choice it made was correct, neutral, or incorrect. It is a good technique to use for automated systems that must make a lot of small decisions without human guidance.

Reinforcement learning is an autonomous, self-teaching system that essentially learns by trial and error. It performs actions with the aim of maximizing rewards, or in other words, it is learning by doing in order to achieve the best outcomes.

❖ Main points in Reinforcement learning: –

- Input: The input should be an initial state from which the model will start
- Output: There are many possible outputs as there are a variety of solutions to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

❖ Types of Reinforcement:

There are two types of Reinforcement:

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

Advantages of reinforcement learning are:

- Maximizes Performance
- Sustain Change for a long period of time
- Too much Reinforcement can lead to an overload of states which can diminish the results

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

Advantages of reinforcement learning:

- Increases Behavior
- Provide defiance to a minimum standard of performance
- It Only provides enough to meet up the minimum behavior

❖ Elements of Reinforcement Learning:

Reinforcement learning elements are as follows:

1. **Policy:** Policy defines the learning agent behavior for given time period. It is a mapping from perceived states of the environment to actions to be taken when in those states.
2. **Reward function:** Reward function is used to define a goal in a reinforcement learning problem. A reward function is a function that provides a numerical score based on the state of the environment
3. **Value function:** Value functions specify what is good in the long run. The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.
4. **Model of the environment:** Models are used for planning.

❖ Application of Reinforcement Learnings: -

1. Robotics: Robots with pre-programmed behaviour are useful in structured environments, such as the assembly line of an automobile manufacturing plant, where the task is repetitive in nature.
2. A master chess player makes a move. The choice is informed both by planning, anticipating possible replies and counter replies.
3. An adaptive controller adjusts parameters of a petroleum refinery's operation in real time.

RL can be used in large environments in the following situations:

- A model of the environment is known, but an analytic solution is not available;
- Only a simulation model of the environment is given (the subject of simulation-based optimization)
- The only way to collect information about the environment is to interact with it.

❖ Advantages of Reinforcement learning: -

1. Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques.
2. The model can correct the errors that occurred during the training process.
3. In RL, training data is obtained via the direct interaction of the agent with the environment
4. Reinforcement learning can handle environments that are non-deterministic, meaning that the outcomes of actions are not always predictable. This is useful in real-world applications where the environment may change over time or is uncertain.
5. Reinforcement learning can be used to solve a wide range of problems, including those that involve decision making, control, and optimization.
6. Reinforcement learning is a flexible approach that can be combined with other machine learning techniques, such as deep learning, to improve performance.

❖ Disadvantages of Reinforcement learning: -

1. Reinforcement learning is not preferable to use for solving simple problems.
2. Reinforcement learning needs a lot of data and a lot of computation
3. Reinforcement learning is highly dependent on the quality of the reward function. If the reward function is poorly designed, the agent may not learn the desired behaviour.
4. Reinforcement learning can be difficult to debug and interpret. It is not always clear why the agent is behaving in a certain way, which can make it difficult to diagnose and fix problems.

Markov Decision Process

Reinforcement Learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problem is known as a **Markov Decision Process**.

❖ A **Markov Decision Process (MDP)** model contains:

- A set of possible world states S .
- A set of Models.
- A set of possible actions A .
- A real-valued reward function $R(s,a)$.
- A policy the solution of **Markov Decision Process**.

States:	S
Model:	$T(S, a, S') \sim P(S' S, a)$
Actions:	$A(S), A$
Reward:	$R(S), R(S, a), R(S, a, S')$
Policy:	$\Pi(S) \rightarrow a$ Π^*

Markov Decision Process

A **State** is a set of tokens that represent every state that the agent can be in.

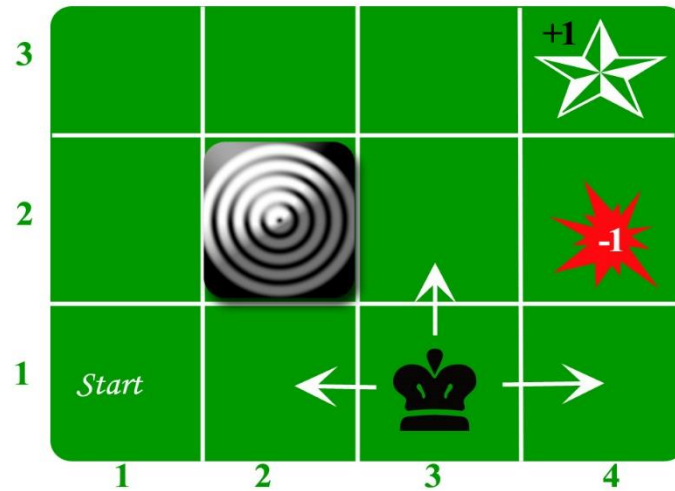
A **Model** (sometimes called Transition Model) gives an action's effect in a state. In particular, $T(S, a, S')$ defines a transition T where being in state S and taking an action 'a' takes us to state S' (S and S' may be the same). For stochastic actions (noisy, non-deterministic) we also define a probability $P(S'|S,a)$ which represents the probability of reaching a state S' if action 'a' is taken in state S . Note Markov property states that the effects of an action taken in a state depend only on that state and not on the prior history.

An **Action** A is a set of all possible actions. $A(s)$ defines the set of actions that can be taken being in state S .

A **Reward** is a real-valued reward function. $R(s)$ indicates the reward for simply being in the state S . $R(S,a)$ indicates the reward for being in a state S and taking an action 'a'. $R(S,a,S')$ indicates the reward for being in a state S , taking an action 'a' and ending up in a state S' .

A **Policy** is a solution to the Markov Decision Process. A policy is a mapping from S to a . It indicates the action 'a' to be taken while in state S .

Let us take the example of a grid world:



An agent lives in the grid. The above example is a 3*4 grid. The grid has a START state(grid no 1,1). The purpose of the agent is to wander around the grid to finally reach the Blue Diamond (grid no 4,3). Under all circumstances, the agent should avoid the Fire grid (orange color, grid no 4,2). Also the grid no 2,2 is a blocked grid, it acts as a wall hence the agent cannot enter it.

The agent can take any one of these actions: **UP, DOWN, LEFT, RIGHT**

Walls block the agent path, i.e., if there is a wall in the direction the agent would have taken, the agent stays in the same place. So for example, if the agent says LEFT in the START grid he would stay put in the START grid.

First Aim: To find the shortest sequence getting from START to the Diamond. Two such sequences can be found:

- **RIGHT RIGHT UP UP RIGHT**
- **UP UP RIGHT RIGHT RIGHT**

Let us take the second one (UP UP RIGHT RIGHT RIGHT) for the subsequent discussion.

The move is now noisy. 80% of the time the intended action works correctly. 20% of the time the action agent takes causes it to move at right angles. For example, if the agent says UP the probability of going UP is 0.8 whereas the probability of going LEFT is 0.1, and the probability of going RIGHT is 0.1 (since LEFT and RIGHT are right angles to UP).

The agent receives rewards each time step:-

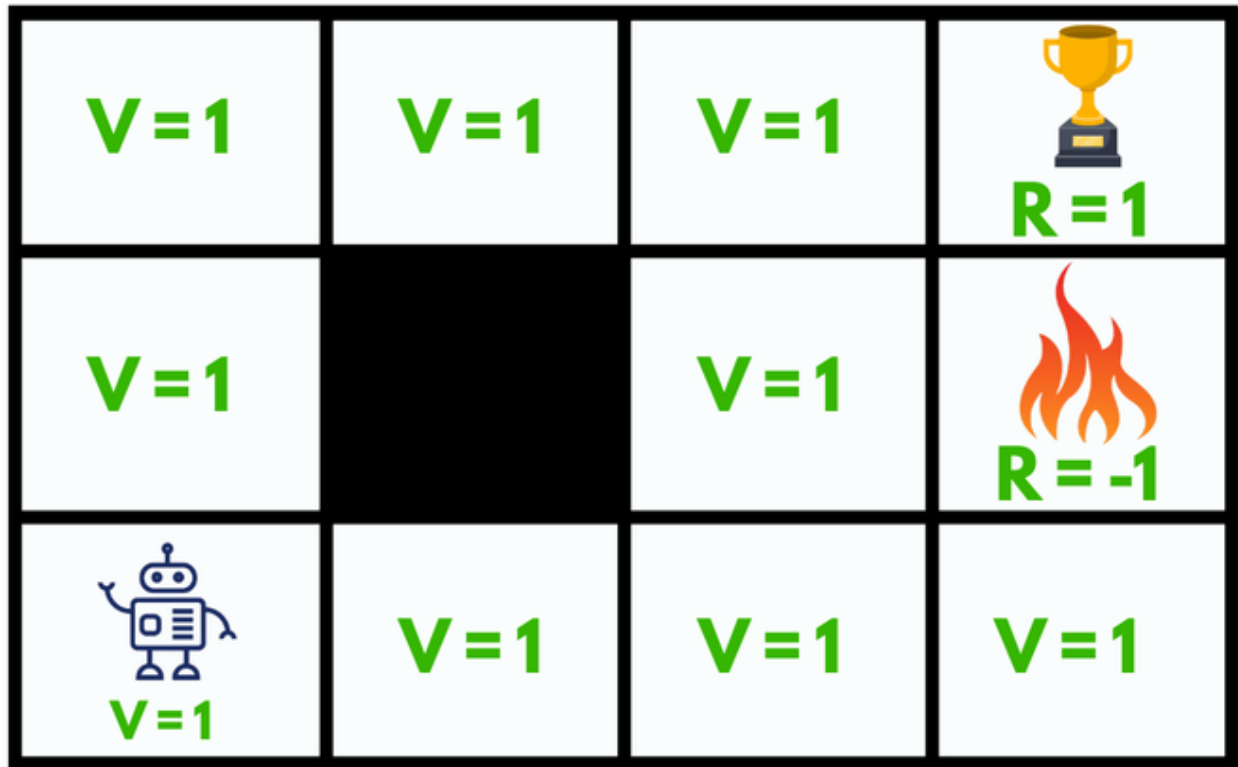
- Small reward each step (can be negative when can also be term as punishment, in the above example entering the Fire can have a reward of -1).
- Big rewards come at the end (good or bad).
- The goal is to Maximize the sum of rewards.

Bellman Equation

According to the **Bellman Equation**, long-term- reward in a **given action** is **equal to the reward from the current action combined with the expected reward from the future actions** taken at the following time.

Let's take an example:

Here we have a **maze** which is our environment and the sole **goal** of our agent is to reach the **trophy state** (**R = 1**) or to get **Good reward** and to **avoid the fire state** because it will be a **failure** (**R = -1**) or will get **Bad reward**.



What happens without Bellman Equation?

Initially, we will give our agent some time to explore the environment and let it figure out a path to the goal. As soon as it reaches its goal, it will **back trace its steps** back to its starting position and **mark values of all the states** which eventually leads towards the goal as **V = 1**.

The agent will face **no problem until** we **change its starting position**, as it will **not be able** to find a path towards the trophy state since the value of all the states is **equal to 1**. So, to solve this problem we should use **Bellman Equation**:

$$V(s) = \max_a (R(s,a) + \gamma V(s'))$$

State(s): current state where the agent is in the environment

Next State(s'): After taking action(a) at state(s) the agent reaches s'

Value(V): Numeric representation of a state which helps the agent to find its path. **V(s)** here means the value of the state s.

Reward(R): treat which the agent gets after performing an action(a).




- **R(s)**: reward for being in the state s
- **R(s,a)**: reward for being in the state and performing an action a
- **R(s,a,s')**: reward for being in a state s, taking an action a and ending up in s'

e.g. **Good reward** can be +1, **Bad reward** can be -1, **No reward** can be 0.

Action(a): set of possible actions that can be taken by the agent in the state(s). e.g.

(LEFT, RIGHT, UP, DOWN)

Discount factor(γ): determines how much the agent cares about rewards in the distant future relative to those in the immediate future. It has a value **between 0 and 1**. **Lower value** encourages **short-term** rewards while **higher value** promises **long-term** reward

$V = 0.81$ $(V = 0 + (0.9)(0.9))$	$V = 0.9$ $(V = 0 + 0.9)$	$V = 1$ $(V = 1 + 0)$	 $R = 1$
$V = 0.73$ $(V = 0 + (0.9)(0.81))$		$V = 0.9$	 $R = -1$
 $V = 0.4$	$V = 0.73$	$V = 0.81$	$V = 0.73$

The **max** denotes the most **optimum** action among all the actions that the agent can take in a particular state which can lead to the reward after **repeating this process every consecutive step**.

For example:

- The state left to the fire state ($V = 0.9$) can go **UP**, **DOWN**, **RIGHT** but **NOT LEFT** because it's a wall(not accessible). Among all these actions available the **maximum value** for that state is the **UP** action.
- The current starting state of our agent can choose any **random** action **UP** or **RIGHT** since both lead towards the reward with the **same number of steps**.

By using the Bellman equation our agent will calculate the value of every step **except for the trophy and the fire state ($V = 0$)**, they cannot have values since they are the **end of the maze**.

So, after making such a plan our agent can easily accomplish its goal by just following the **increasing values**.

Policy evaluation using Monte Carlo

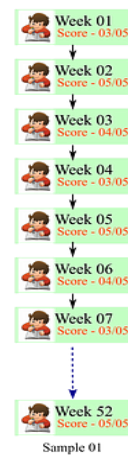
Monte Carlo method on the other hand is a very simple concept where agent learn about the states and reward when it interacts with the environment. In this method agent generate experienced samples and then based on average return, value is calculated for a state or state-action. Below are key characteristics of Monte Carlo (MC) method:

1. There is no model (agent does not know state MDP transitions)
2. agent **learns** from **sampld** experience
3. learn state value $v\pi(s)$ under policy π by experiencing **average** return from all sampled episodes (value = average return)
4. only after a **complete episode**, values are updated (because of this algorithm convergence is slow and update happens after a episode is Complete)
5. There is no bootstrapping
6. Only can be used in **episodic problems**

Consider a real-life analogy; Monte Carlo learning is like annual examination where student completes its episode at the end of the year. Here, the result of the annual exam is like the return obtained by the student. Now if the goal of the problem is to find how students score during a calendar year (which is a episode here) for a class, we can take sample result of some student and then calculate mean result to find score for a class (don't take the analogy point by point but on a holistic level I think you can get the essence of MC learning). Similarly, we have TD learning or temporal difference learning (TD learning is like updating value in every time step and does not require wait till end of episode to update the values) that we will cover in future blog, can be thought like a weekly or monthly examination (student can adjust their performance based on this score (reward received) after every small interval and final score is accumulation of the all weekly tests (total rewards)).



Monte Carlo

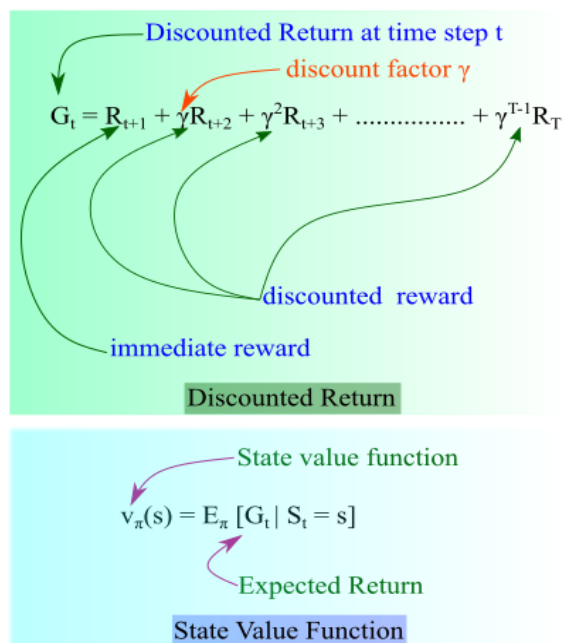


Temporal Difference : TD(0)

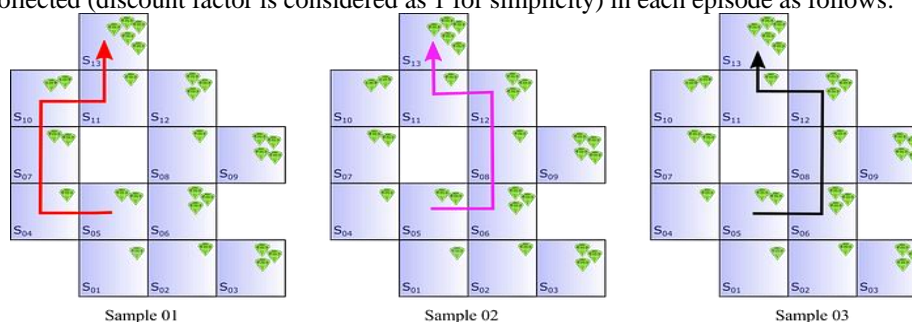
Value function = Expected **Return**

Expected return is equal to discounted sum of all rewards.

In Monte Carlo Method instead of expected return we use empirical return that agent has sampled based following the policy.



If we go back to our very first example of gem collection, agent follows policy and complete an episode, along the way in each step it collects rewards in the form of gem. To get state value agent sum-up all the gems collected after each episode starting from that state. Refer to below diagram where 3 samples collected starting from State S_{05} . Total reward collected (discount factor is considered as 1 for simplicity) in each episode as follows:



3 Samples starting from State S_{05}

Return (Sample 01) = $2 + 1 + 2 + 2 + 1 + 5 = 13$ gems

Return (Sample 02) = $2 + 3 + 1 + 3 + 1 + 5 = 15$ gems

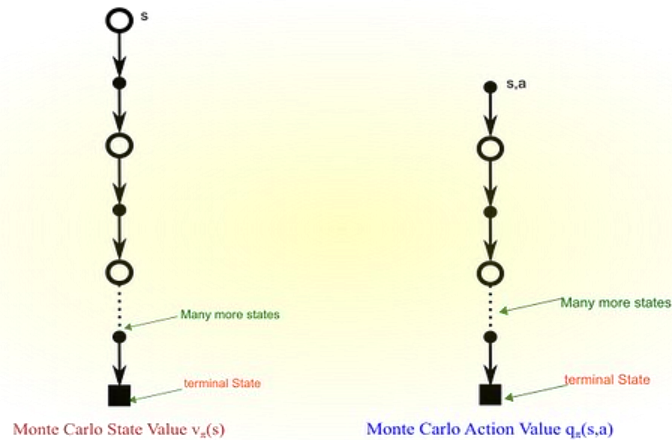
Return (Sample 03) = $2 + 3 + 1 + 3 + 1 + 5 = 15$ gems

Observed mean return (based on 3 samples) = $(13 + 15 + 15)/3 = 14.33$ gems

Thus state value as per Monte Carlo Method, $v_{\pi}(S_{05})$ is 14.33 gems based on 3 samples following policy π .

❖ Monte Carlo Backup diagram:

Monte Carlo Backup diagram would look like below (refer to [3rd blog](#) post for more on backup diagram).



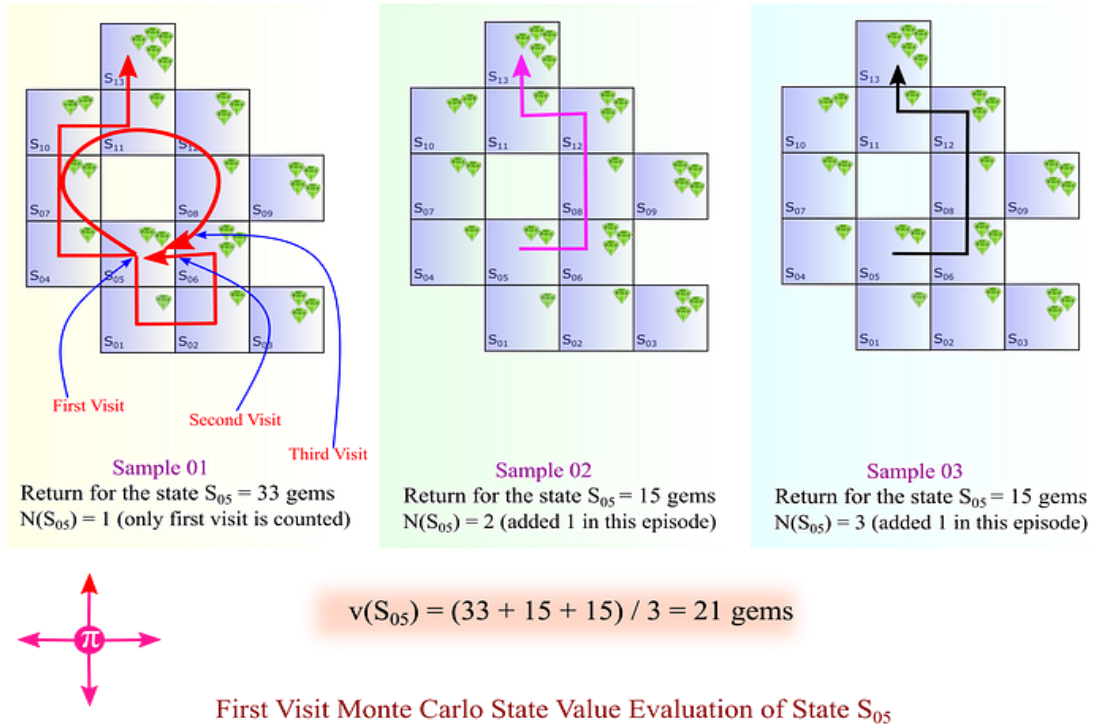
❖ There are two types of MC learning policy evaluation (prediction) methods:

1. First Visit Monte Carlo Method:

In this case in an episode first visit of the state is counted (even if agent comes-back to the same state multiple time in the episode, only first visit will be counted). Detailed step as below:

1. To evaluate state s , first we set number of visit, $N(s) = 0$, Total return $TR(s) = 0$ (these values are updated across episodes)
2. The **first** time-step t that state s is visited in an episode, increment counter $N(s) = N(s) + 1$
3. Increment total return $TR(s) = TR(s) + G_t$
4. Value is estimated by mean return $V(s) = TR(s)/N(s)$
5. By law of large numbers, $V(s) \rightarrow v_\pi(s)$ (this is called true value under policy π) as $N(s)$ approaches infinity

Refer to below diagram for better understanding of counter increment.

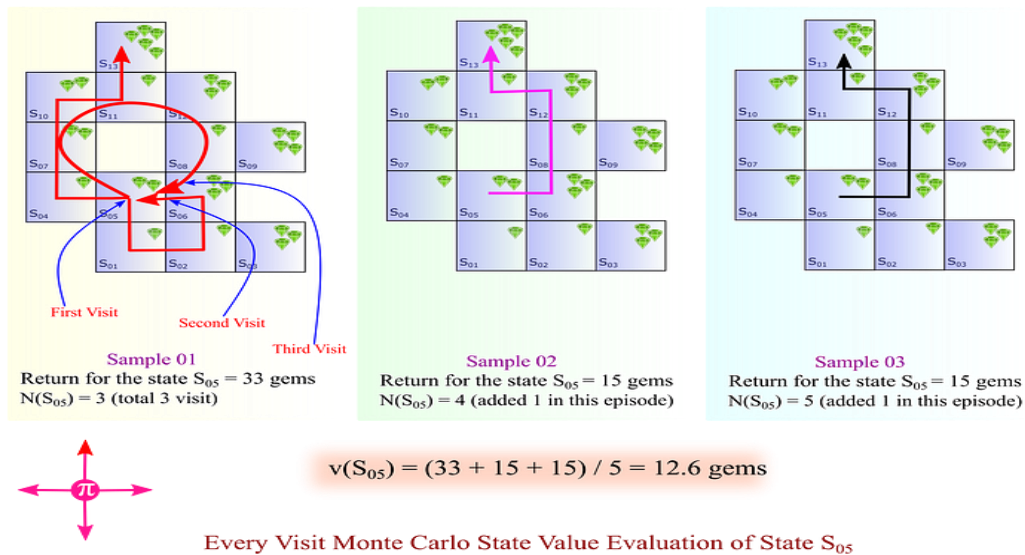


2. Every Visit Monte Carlo Method:

In this case in an episode every visit of the state is counted. Detailed step as below:

1. To evaluate state s , first we set number of visit, $N(s) = 0$, Total return $TR(s) = 0$ (these values are updated across episodes)
2. **every** time-step t that state s is visited in an episode, increment counter $N(s) = N(s) + 1$
3. Increment total return $TR(s) = TR(s) + G_t$
4. Value is estimated by mean return $V(s) = TR(s)/N(s)$
5. By law of large numbers, $V(s) \rightarrow v_{\pi}(s)$ (this is called true value under policy π) as $N(s)$ approaches infinity

Refer to below diagram for better understanding of counter increment.



Usually MC is updated incrementally after every episode (no need to store old episode values, it could be a running mean value for the state updated after every episode).

Update $V(s)$ incrementally after episode $S_1, A_2, R_3, \dots, S_T$ For each state S_t with return G_t

Usually in place of $1/N(S, t)$ a constant learning rate (α) is used and above equation becomes :
For Policy improvement, Generalized Policy Improvement concept is used to update policy using action value function of Monte Carlo Method.

❖ **Advantages:**

- zero bias
- Good convergence properties (even with function approximation)
- Not very sensitive to initial value
- Very simple to understand and use

❖ **Limitations:**

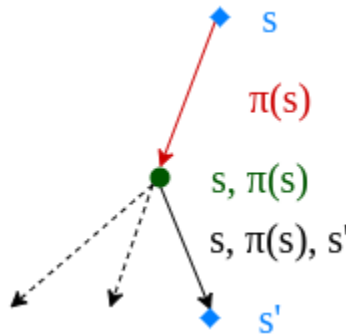
- MC must wait until end of episode before return is known
- MC has high variance
- MC can only learn from complete sequences
- MC only works for episodic (terminating) environments

Policy iteration and value iteration

A policy is a mapping from states to actions. Finding an optimal policy leads to generating the maximum reward. Given an MDP environment, we can use dynamic programming algorithms to compute optimal policies, which lead to the highest possible sum of future rewards at each state.

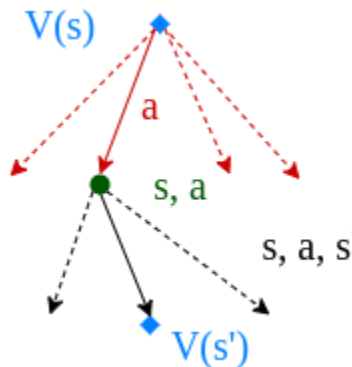
❖ Policy iteration: -

In policy iteration, we start by choosing an arbitrary policy. Then, we iteratively evaluate and improve the policy until convergence:



❖ Value Iteration: -

In value iteration, we compute the optimal state value function by iteratively updating the estimate :



The update step is very similar to the update step in the policy iteration algorithm. The only difference is that we take the maximum over all possible actions in the value iteration algorithm. Instead of evaluating and then improving, the value iteration algorithm updates the state value function in a single step. This is possible by calculating all possible rewards by looking ahead. The value iteration algorithm is guaranteed to converge to the optimal values.

❖ Policy Iteration vs Value Iteration: -

Policy iteration and value iteration are both dynamic programming algorithms that find an optimal policy in a reinforcement learning environment. They both employ variations of Bellman updates and exploit one-step look-ahead:

In policy iteration, we start with a fixed policy. Conversely, in value iteration, we begin by selecting the value function. Then, in both algorithms, we iteratively improve until we reach convergence.

The policy iteration algorithm updates the policy. The value iteration algorithm iterates over the value function instead. Still, both algorithms implicitly update the policy and state value function in each iteration.

In each iteration, the policy iteration function goes through two phases. One phase evaluates the policy, and the other one improves it. The value iteration function covers these two phases by taking a maximum over the utility function for all possible actions.

The value iteration algorithm is straightforward. It combines two phases of the policy iteration into a single update operation. However, the value iteration function runs through all possible actions at once to find the maximum action value. Subsequently, the value iteration algorithm is computationally heavier.

Both algorithms are guaranteed to converge to an optimal policy in the end. Yet, the policy iteration algorithm converges within fewer iterations. As a result, the policy iteration is reported to conclude faster than the value iteration algorithm.

Q-Learning

Q-Learning is a Reinforcement learning policy which will find the next best action, given a current state. It chooses this action at random and aims to maximize the reward.



Using Q-learning, we can make an Ad recommendation system which will suggest related products to our previous purchase.
The reward will be if user clicks on the suggested product.

Reinforcement Learning briefly is a paradigm of Learning Process in which a learning agent learns, overtime, to behave optimally in a certain environment by interacting continuously in the environment. The agent during its course of learning experience various different situations in the environment it is in. These are called *states*. The agent while being in that state may choose from a set of allowable actions which may fetch different *rewards* (or penalties). The learning agent overtime learns to maximize these rewards to behave optimally at any given state it is in. **Q-Learning** is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behaviour of the learning agent.

1. **Q-Values or Action-Values:** Q-values are defined for states and actions. This estimation of will be iteratively computed using the **TD- Update rule**.
2. **Rewards and Episodes:** An agent over the course of its lifetime starts from a start state, makes several transitions from its current state to a next state based on its choice of action and also the environment the agent is interacting in. At every step of transition, the agent from a state takes an action, observes a reward from the environment, and then transits to another state. If at any point of time the agent ends up in one of the terminating states that means there are no further transition possible. This is said to be the completion of an episode.
3. **Temporal Difference or TD-Update:** The Temporal Difference or TD-Update rule can be represented as follows :

$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

This update rule to estimate the value of Q is applied at every time step of the agent's interaction with the environment.

4. **Choosing the Action to take using -greedy policy:** greedy policy is a very simple policy of choosing actions using the current Q-value estimations.

❖ Pros:

- Long-term outcomes, which are exceedingly challenging to accomplish, are best achieved with this strategy.
- This learning paradigm closely resembles how people learn. Consequently, it is almost ideal.
- The model has the ability to fix mistakes made during training.
- Once a model has fixed a mistake, there is virtually little probability that it will happen again.
- It can produce the ideal model to address a certain issue.

❖ Cons:

- drawback of using actual samples. Think about the situation of robot learning, for instance. The hardware for robots is typically quite expensive, subject to deterioration, and in need of meticulous upkeep. The expense of fixing a robot system is high.
- Instead of abandoning reinforcement learning altogether, we can combine it with other techniques to alleviate many of its difficulties. Deep learning and reinforcement learning are one common combo.

SARSA algorithm is a slight variation of the popular Q-Learning algorithm. For a learning agent in any Reinforcement Learning algorithm, its policy can be of two types: -

1. **On Policy:** In this, the learning agent learns the value function according to the current action derived from the policy currently being used.
2. **Off Policy:** In this, the learning agent learns the value function according to the action derived from another policy.

Q-Learning technique is an **Off Policy** technique and uses the greedy approach to learn the Q-value. SARSA technique, on the other hand, is an **On Policy** and uses the action performed by the current policy to learn the Q-value.

SARSA (State-Action-Reward-State-Action) is a reinforcement learning algorithm that is used to learn a policy for an agent interacting with an environment. It is a type of on-policy algorithm, which means that it learns the value function and the policy based on the actions that are actually taken by the agent. The SARSA algorithm works by maintaining a table of action-value estimates $Q(s, a)$, where s is the state and a is the action taken by the agent in that state. The table is initialized to some arbitrary values, and the agent uses an epsilon-greedy policy to select actions.

❖ SARSA algorithm:

Initialize the action-value estimates $Q(s, a)$ to some arbitrary values.

1. Set the initial state s .
2. Choose the initial action a using an epsilon-greedy policy based on the current Q values.
3. Take the action a and observe the reward r and the next state s' .
4. Choose the next action a' using an epsilon-greedy policy based on the updated Q values.
5. Update the action-value estimate for the current state-action pair using the SARSA update rule:
$$Q(s, a) = Q(s, a) + \alpha * (r + \gamma * Q(s', a') - Q(s, a))$$

where α is the learning rate, γ is the discount factor, and $r + \gamma * Q(s', a')$ is the estimated return for the next state-action pair.

Set the current state s to the next state s' , and the current action a to the next action a' .

Repeat steps 4-7 until the episode ends.

The SARSA algorithm learns a policy that balances exploration and exploitation, and can be used in a variety of applications, including robotics, game playing, and decision making. However, it is important to note that the convergence of the SARSA algorithm can be slow, especially in large state spaces, and there are other reinforcement learning algorithms that may be more effective in certain situations.

Model-based reinforcement learning

Reinforcement learning systems can make decisions in one of two ways.

model-based approach, a system uses a predictive model of the world to ask questions of the form “what will happen if I do x ?” to choose the best x . In the alternative *model-free* approach, the modelling step is bypassed altogether in favour of learning a control policy directly. Although in practice the line between these two techniques can become blurred, as a coarse guide it is useful for dividing up the space of algorithmic possibilities.

Predictive models can be used to ask “what if?” questions to guide future decisions.

The natural question to ask after making this distinction is whether to use such a predictive model. The field has grappled with this question for quite a while, and is unlikely to reach a consensus any time soon. However, we have learned enough about designing model-based algorithms that it is possible to draw some general conclusions about best practices and common pitfalls.

❖ Model-based techniques: -

➤ Model-based algorithms are grouped into four categories:

1. **Analytic gradient computation:**
Assumptions about the form of the dynamics and cost function are convenient because they can yield closed-form solutions for locally optimal control, as in the LQR framework. Even when these assumptions are not valid, receding-horizon control can account for small errors introduced by approximated dynamics. Similarly, dynamics models parametrized as Gaussian processes have analytic gradients that can be used for policy improvement. Controllers derived via these simple parametrizations can also be used to provide guiding samples for training more complex nonlinear policies.
2. **Sampling-based planning:**
In the fully general case of nonlinear dynamics models, we lose guarantees of local optimality and must resort to sampling action sequences. The simplest version of this approach, random shooting, entails sampling candidate actions from a fixed distribution, evaluating them under a model, and choosing the action that is deemed the most promising. More sophisticated variants iteratively adjust the sampling distribution, as in the cross-entropy method (CEM; used in PlaNet, PETS, and visual foresight) or path integral optimal control (used in recent model-based dexterous manipulation work).
3. **Model-based data generation:**
An important detail in many machine learning success stories is a means of artificially increasing the size of a training set. It is difficult to define a manual data augmentation procedure for policy optimization, but we can view a predictive model analogously as a learned method of generating synthetic data. The original proposal of such a combination comes from the Dyna algorithm by Sutton, which alternates between model learning, data generation under a model, and policy learning using the model data. This strategy has been combined with iLQG, model ensembles, and meta-learning; has been scaled to image observations; and is amenable to theoretical analysis. A close cousin to model-based data generation is the use of a model to improve target value estimates for temporal difference learning.
4. **Value-equivalence prediction:**
A final technique, which does not fit neatly into model-based versus model-free categorization, is to incorporate computation that resembles model-based planning without supervising the model’s predictions to resemble actual states. Instead, plans under the model are constrained to match trajectories in the real environment only in their predicted cumulative reward. These value-equivalent models have shown to be effective in high-dimensional observation spaces where conventional model-based planning has proven difficult.

UNIT – 5

Collaborative filtering

❖ What is Collaborative Filtering?

Collaborative filtering is used by most recommendation systems to find similar patterns or information of the users, this technique can filter out items that users like on the basis of the ratings or reactions by similar users. An example of collaborative filtering can be to predict the rating of a particular user based on user ratings for other movies and others' ratings for all movies. This concept is widely used in recommending movies, news, applications, and so many other items.

Let's take one example and understand more about what is **Collaborative Filtering**,

let's assume I have user U1, who likes movies m1,m2,m4. user U2 who likes movies m1,m3,m4, and user U3 who likes movie m1.

So our job is to recommend which are the new movie to watch for the user U3 next.

So here we can see users U1, U2, U3 watch/likes movies m1, so three have the same taste. now in user U1, U2 has like/watch movies m4, so user U3 could like movie m3 so I recommend movie m4, this is the flow of logic.

The key idea of CF is Users who agreed in the past tend to also agree in the future.

Users	Movie 1	Movie 2	Movie 3	Movie 4
User 1	5	4		5
User 2	4		3	
User 3		1		2
User 4	1	2		

❖ Types of Filtering:

➤ There are two types of Collaborative Filtering available:

1. User-User-Based Collaborative Filtering:

user-user collaborative filtering is one kind of recommendation method which looks for similar users based on the items users have already liked or positively interacted with. Let's take a one eg to understand user-user collaborative filtering.

Let's assume given matrix A which contains user id and item id and rating or movies.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

Compute a User User similarity follow these steps, so find a similarity between two users we can use cosine similarity.

so cosine similarity means the similarity between two vectors of inner product space, It is measured by the cosine of the angle between two vectors.

$$\text{similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

2. Item-Item Based Collaborative Filtering:

This is also very simple and very similar in idea with USER-USER Similarity. Let's dive deep into it. This item-item similarity is solving a problem that occurs in a user user-based similarity. Here we find a similarity matrix of items/movies, here we find a similarity between the two movies. to find a similarity we use a cosine distance between the two movies.

ITEM-ITEM SIMILARITY MATRIX							
	I_1	I_2		I_j		I_{m-1}	I_m
I_1	1	Sim_{12}	...	Sim_{1j}	...		
I_2		1		

I_i		Sim_{i2}	...	Sim_{ij}	...		

I_{m-1}			1	
I_m				1

Sim_{ij} = similarity (item_i , item_j)

So how to recommend an item to the user?

Let's suppose we have to recommend new items to user10, and we know a user10 already likes/watch item7,8,1. Now we go to the item-item similarity matrix, we take the most similar item to items7,8,1 based on the similarity values.

let's suppose the most similar item for item7 is {item9, item4, item10}, the Most similar item to item8 is {item19, item 4, item10} and the Most similar item to item 1 is {item9, item14, item10}

Now we take a very common item from every set of items and the common items are {item9, item4, item10, item 19, item 14} and we recommend these all items to user10.

The most popular filtering is item item-based filtering because over time item is not changed like the user user-based similarity.

Content-based filtering

A Content-Based Recommender works by the data that we take from the user, either explicitly (rating) or implicitly (clicking on a link). By the data we create a user profile, which is then used to suggest to the user, as the user provides more input or take more actions on the recommendation, the engine becomes more accurate.

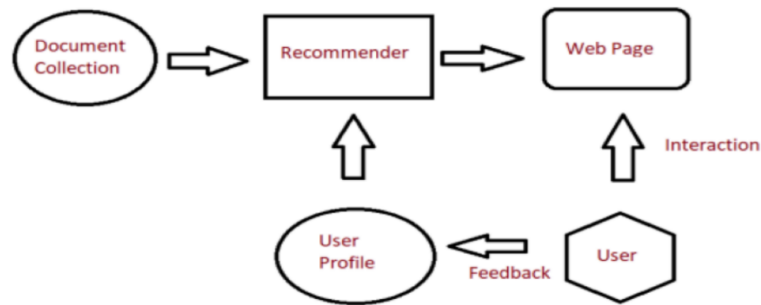


Fig: Recommender System

User Profile:

In the User Profile, we create vectors that describe the user's preference. In the creation of a user profile, we use the utility matrix which describes the relationship between user and item. With this information, the best estimate we can make regarding which item user likes, is some aggregation of the profiles of those items.

Item Profile:

In Content-Based Recommender, we must build a profile for each item, which will represent the important characteristics of that item.

For example, if we make a movie as an item then its actors, director, release year and genre are the most significant features of the movie. We can also add its rating from the IMDB (Internet Movie Database) in the Item Profile.

Utility Matrix:

Utility Matrix signifies the user's preference with certain items. In the data gathered from the user, we have to find some relation between the items which are liked by the user and those which are disliked, for this purpose we use the utility matrix. In it we assign a particular value to each user-item pair, this value is known as the degree of preference. Then we draw a matrix of a user with the respective items to identify their preference relationship.

Users	Movie 1	Movie 2	Movie 3
User 1	3		1
User 2	2	4	

Fig: Utility Matrix of Movie Recommendation System

Some of the columns are blank in the matrix that is because we don't get the whole input from the user every time, and the goal of a recommendation system is not to fill all the columns but to recommend a movie to the user which he/she will prefer. Through this table, our recommender system won't suggest Movie 3 to User 2, because in Movie 1 they have given approximately the same ratings, and in Movie 3 User 1 has given the low rating, so it is highly possible that User 2 also won't like it.

❖ Recommending Items to User Based on Content:

• Method 1:

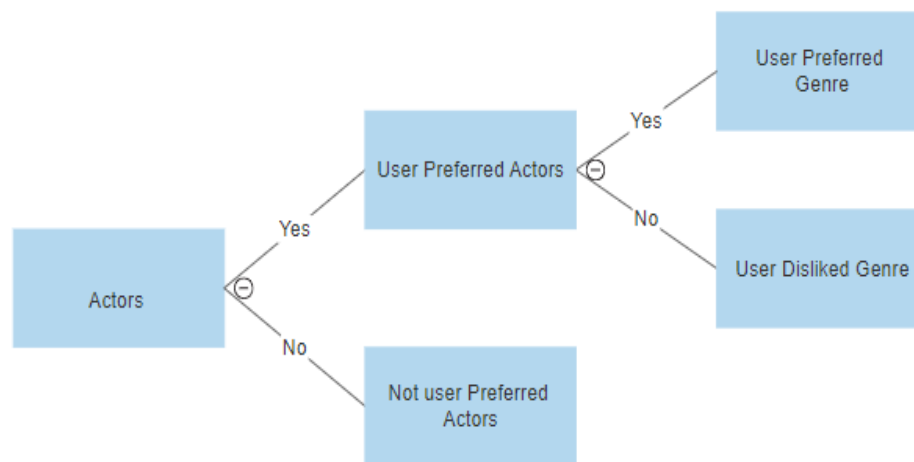
We can use the cosine distance between the vectors of the item and the user to determine its preference to the user. For explaining this, let us consider an example:

We observe that the vector for a user will have a positive number for actors that tend to appear in movies the user likes and negative numbers for actors user doesn't like. Consider a movie with actors which user likes and only a few actors which user doesn't like, then the cosine angle between the user's and movie's vectors will be a large positive fraction. Thus, the angle will be close to 0, therefore a small cosine distance between the vectors.

It represents that the user tends to like the movie, if the cosine distance is large, then we tend to avoid the item from the recommendation.

• Method 2:

We can use a classification approach in the recommendation systems too, like we can use the Decision Tree for finding out whether a user wants to watch a movie or not, like at each level we can apply a certain condition to refine our recommendation. For example:



❖ Pros: -

1. Easily scalable due to low amounts of data.
2. Unlike other models, this does not need to compare with other user's data.

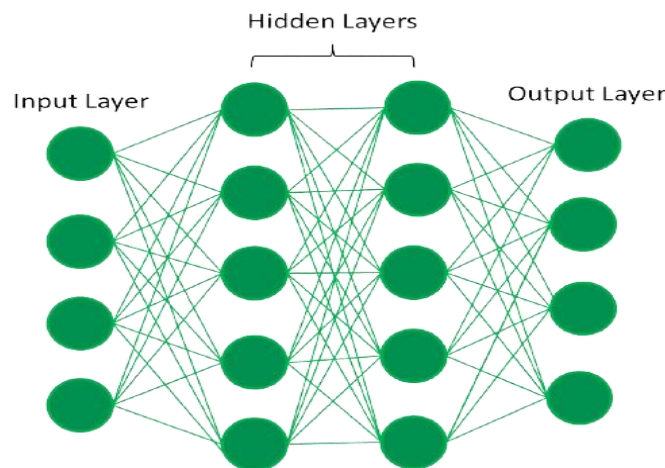
❖ Cons: -

1. Model requires a fair amount of domain knowledge.
2. Content-based filtering depends greatly on previously known user interests.

Artificial Neural Network

Artificial Neural Networks contain artificial neurons which are called **units**. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset. Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

In the majority of neural networks, units are interconnected from one layer to another. Each of these connections has weights that determine the influence of one unit on another unit. As the data transfers from one unit to another, the neural network learns more and more about the data which eventually results in an output from the output layer.



The structures and operations of human neurons serve as the basis for artificial neural networks. It is also known as neural networks or neural nets. The input layer of an artificial neural network is the first layer, and it receives input from external sources and releases it to the hidden layer, which is the second layer. In the hidden layer, each neuron receives input from the previous layer neurons, computes the weighted sum, and sends it to the neurons in the next layer. These connections are weighted means effects of the inputs from the previous layer are optimized more or less by assigning different-different weights to each input and it is adjusted during the training process by optimizing these weights for improved model performance.

❖ Artificial neurons vs Biological neurons

The concept of artificial neural networks comes from biological neurons found in animal brains So they share a lot of similarities in structure and function wise.

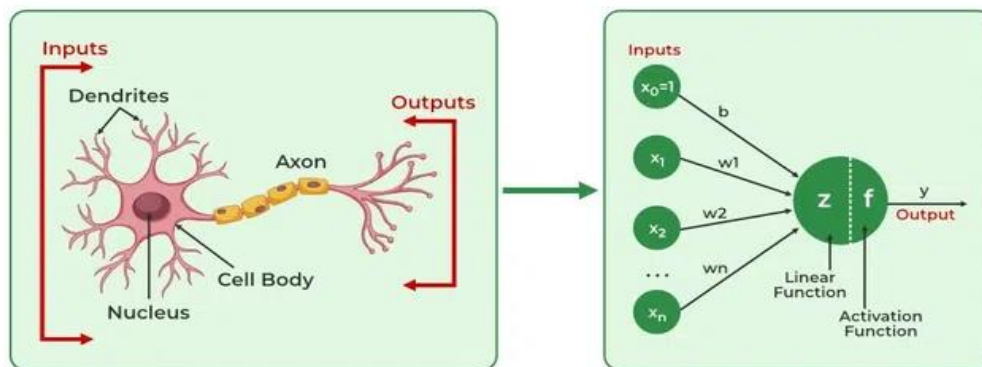
- **Structure:** The structure of artificial neural networks is inspired by biological neurons. A biological neuron has a cell body or soma to process the impulses, dendrites to receive them, and an axon that transfers them to other neurons. The input nodes of artificial neural networks receive input signals, the hidden layer nodes compute these input signals, and the output layer nodes compute the final output by processing the hidden layer's results using activation functions.

Biological Neuron	Artificial Neuron
Dendrite	Inputs
Cell nucleus or Soma	Nodes
Synapses	Weights
Axon	Output

- **Synapses:** Synapses are the links between biological neurons that enable the transmission of impulses from dendrites to the cell body. Synapses are the weights that join the one-layer nodes to the next-layer nodes in artificial neurons. The strength of the links is determined by the weight value.
- **Learning:** In biological neurons, learning happens in the cell body nucleus or soma, which has a nucleus that helps to process the impulses. An action potential is produced and travels through the axons if the impulses are powerful enough to reach the threshold. This becomes possible by synaptic plasticity, which represents the ability of synapses to become stronger or weaker over time in reaction to changes in their activity. In artificial neural networks, backpropagation is a technique used for learning, which adjusts the weights between nodes according to the error or differences between predicted and actual outcomes.

Biological Neuron	Artificial Neuron
Synaptic plasticity	Backpropagations

- **Activation:** In biological neurons, activation is the firing rate of the neuron which happens when the impulses are strong enough to reach the threshold. In artificial neural networks, A mathematical function known as an activation function maps the input to the output, and executes activations.



❖ How do Artificial Neural Networks learn?

Artificial neural networks are trained using a training set. For example, suppose you want to teach an ANN to recognize a cat. Then it is shown thousands of different images of cats so that the network can learn to identify a cat. Once the neural network has been trained enough using images of cats, then you need to check if it can identify cat images correctly. This is done by making the ANN classify the images it is provided by deciding whether they are cat images or not. The output obtained by the ANN is corroborated by a human-provided description of whether the image is a cat image or not. If the ANN identifies incorrectly then **backpropagation** is used to adjust whatever it has learned during training. **Backpropagation** is done by fine-tuning the weights of the connections in ANN units based on the error rate obtained. This process continues until the artificial neural network can correctly recognize a cat in an image with minimal possible error rates.

❖ What are the types of Artificial Neural Networks?

1. **Feedforward Neural Network:** The feedforward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided travels in a single direction. It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist. So the feedforward neural network has a front-propagated wave only and usually does not have backpropagation.
2. **Convolutional Neural Network:** A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layer that uses a convolution operation on the input and then passes the result obtained in the form of output to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.
3. **Modular Neural Network:** A Modular Neural Network contains a collection of different neural networks that work independently towards obtaining the output with no interaction between them. Each of the different neural networks performs a different sub-task by obtaining unique inputs compared to other networks. The advantage of this modular neural network is that it breaks down a large and complex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.
4. **Radial basis function Neural Network:** Radial basis functions are those functions that consider the distance of a point concerning the center. RBF functions have two layers. In the first layer, the input is mapped into all the Radial basis functions in the hidden layer and then the output layer computes the output in the next step. Radial basis function nets are normally used to model the data that represents any underlying trend or function.
5. **Recurrent Neural Network:** The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer. The first layer in the RNN is quite similar to the feed-forward neural network and the recurrent neural network starts once the output of the first layer is computed. After this layer, each unit will remember some information from the previous step so that it can act as a memory cell in performing computations.

❖ Applications of Artificial Neural Networks:

1. **Social Media:** Artificial Neural Networks are used heavily in Social Media. For example, let's take the 'People you may know' feature on Facebook that suggests people that you might know in real life so that you can send them friend requests. Well, this magical effect is achieved by using Artificial Neural Networks that analyze your profile, your interests, your current friends, and also their friends and various other factors to calculate the people you might potentially know. Another common application of **Machine Learning** in social media is **facial recognition**. This is done by finding around 100 reference points on the person's face and then matching them with those already available in the database using convolutional neural networks.
2. **Marketing and Sales:** When you log onto E-commerce sites like Amazon and Flipkart, they will recommend your products to buy based on your previous browsing history. Similarly, suppose you love Pasta, then Zomato, Swiggy, etc. will show you restaurant recommendations based on your tastes and previous order history. This is true across all new-age marketing segments like Book sites, Movie services, Hospitality sites, etc. and it is done by implementing **personalized marketing**. This uses Artificial Neural Networks to identify the customer likes, dislikes, previous shopping history, etc., and then tailor the marketing campaigns accordingly.
3. **Healthcare:** Artificial Neural Networks are used in Oncology to train algorithms that can identify cancerous tissue at the microscopic level at the same accuracy as trained physicians. Various rare diseases may manifest in physical characteristics and can be identified in their premature stages by using **Facial Analysis** on the patient photos. So the full-scale implementation of Artificial Neural Networks in the healthcare environment can only enhance the diagnostic abilities of medical experts and ultimately lead to the overall improvement in the quality of medical care all over the world.
4. **Personal Assistants:** I am sure you all have heard of Siri, Alexa, Cortana, etc., and also heard them based on the phones you have!!! These are personal assistants and an example of speech recognition that uses **Natural Language Processing** to interact with the users and formulate a response accordingly. Natural Language Processing uses artificial neural networks that are made to handle many tasks of these personal assistants such as managing the language syntax, semantics, correct speech, the conversation that is going on, etc.

Perceptron

Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**

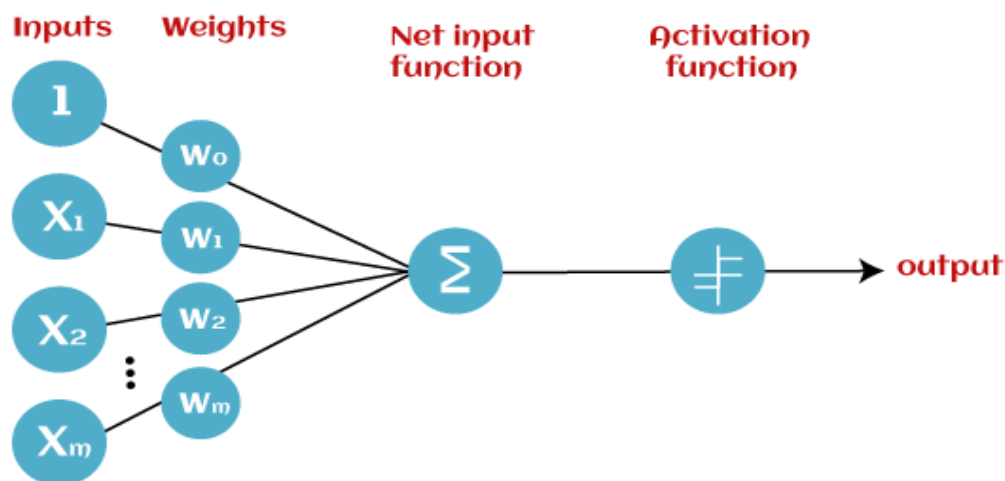
❖ What is Binary classifier in Machine Learning?

In Machine Learning, binary classifiers are defined as the function that helps in deciding whether input data can be represented as vectors of numbers and belongs to some specific class.

Binary classifiers can be considered as linear classifiers. In simple words, we can understand it as a classification algorithm that can predict linear predictor function in terms of weight and feature vectors.

❖ Basic Components of Perceptron:

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



Input Nodes or Input Layer:

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

Wight and Bias:

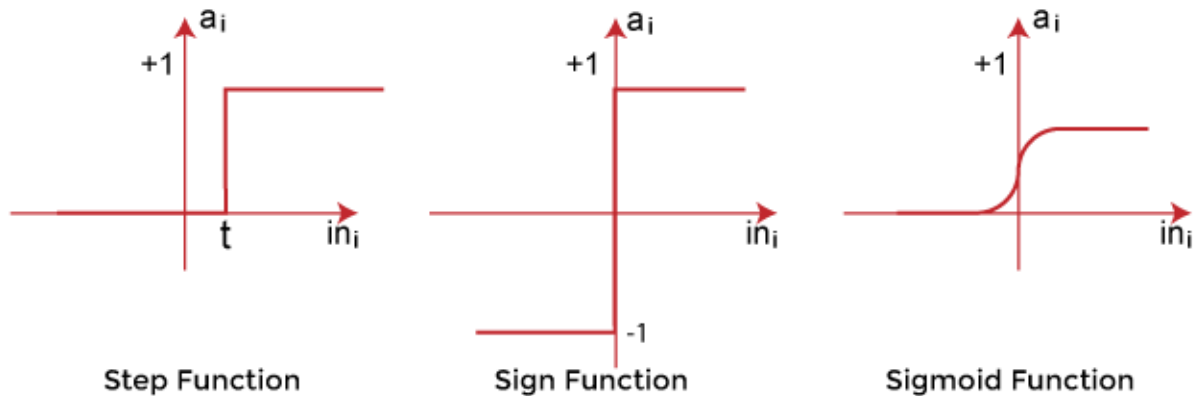
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation Function:

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

➤ Types of Activation functions:

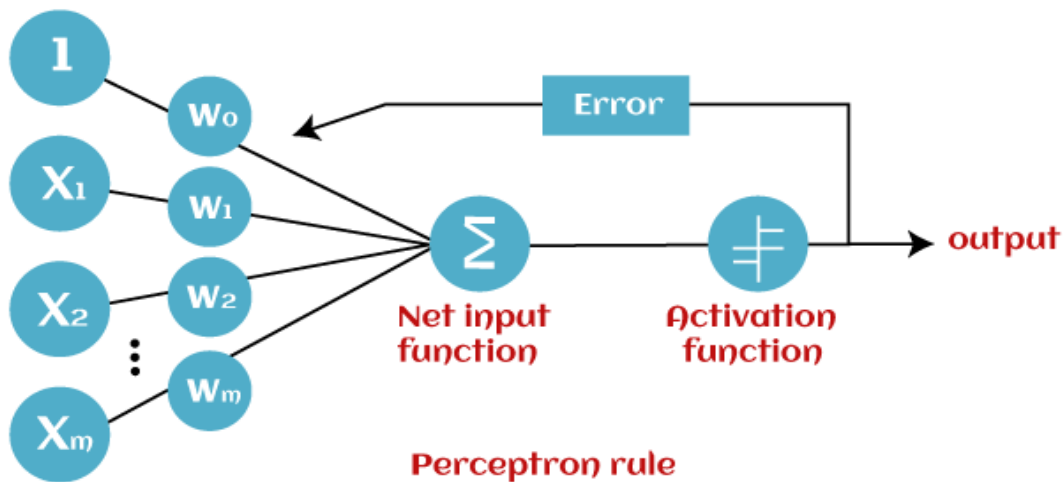
1. Sign function
2. Step function, and
3. Sigmoid function



The data scientist uses the activation function to take a subjective decision based on various problem statements and forms the desired outputs. Activation function may differ (e.g., Sign, Step, and Sigmoid) in perceptron models by checking whether the learning process is slow or has vanishing or exploding gradients.

❖ How does Perceptron work?

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the **step function** and is represented by 'f'.



This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

❖ Perceptron model works in two important steps as follows:

Step-1

In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

$$\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \dots w_n * x_n$$

Add a special term called **bias 'b'** to this weighted sum to improve the model's performance.

$$\sum w_i * x_i + b$$

Step-2

In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$$Y = f(\sum w_i * x_i + b)$$

❖ Types of Perceptron Models:

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

1. Single Layer Perceptron Model:

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

"Single-layer perceptron can learn only linearly separable patterns."

2. Multi-Layered Perceptron Model:

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

❖ **Advantages of Multi-Layer Perceptron:**

1. A multi-layered perceptron model can be used to solve complex non-linear problems.
2. It works well with both small and large input data.
3. It helps us to obtain quick predictions after the training.
4. It helps to obtain the same accuracy ratio with large as well as small data.

❖ **Disadvantages of Multi-Layer Perceptron:**

1. In Multi-layer perceptron, computations are difficult and time-consuming.
2. In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
3. The model functioning depends on the quality of the training.

❖ **Perceptron Function:**

Perceptron function " $f(x)$ " can be achieved as output by multiplying the input ' x ' with the learned weight coefficient ' w '.

Mathematically, we can express it as follows:

$f(x)=1$; if $w \cdot x + b > 0$

otherwise, $f(x)=0$

- ' w ' represents real-valued weights vector
- ' b ' represents the bias
- ' x ' represents a vector of input x values.

❖ **Characteristics of Perceptron:**

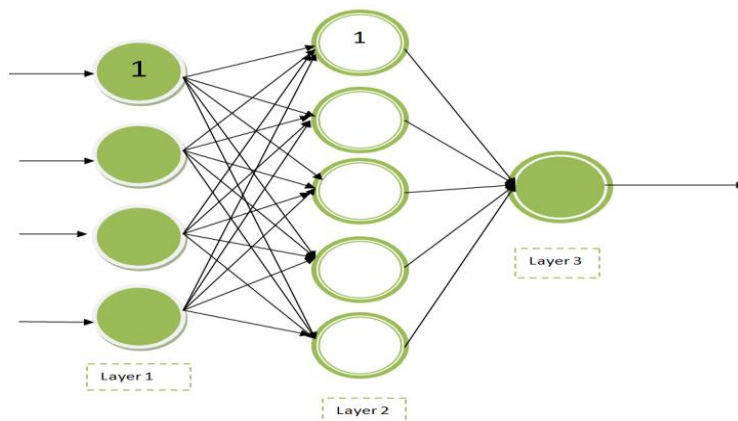
1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.
2. In Perceptron, the weight coefficient is automatically learned.
3. Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.
4. The activation function applies a step rule to check whether the weight function is greater than zero.
5. The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.
6. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

❖ **Limitations of Perceptron Model:**

1. The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.
2. Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

Multilayer Network

To be accurate a fully connected Multi-Layered Neural Network is known as Multi-Layer Perceptron. A Multi-Layered Neural Network consists of multiple layers of artificial neurons or nodes. Unlike Single-Layer Neural networks, in recent times most networks have Multi-Layered Neural Network. The following diagram is a visualization of a multi-layer neural network.



Explanation: Here the nodes marked as “1” are known as **bias units**. The leftmost layer or Layer 1 is the **input layer**, the middle layer or Layer 2 is the **hidden layer** and the rightmost layer or Layer 3 is the **output layer**. It can say that the above diagram has **3 input units** (leaving the bias unit), **1 output unit**, and **4 hidden units** (1 bias unit is not included).

A Multi-layered Neural Network is a typical example of the **Feed Forward Neural Network**. The number of neurons and the number of layers consists of the hyperparameters of Neural Networks which need tuning. In order to find ideal values for the hyperparameters, one must use some cross-validation techniques. Using the Back-Propagation technique, weight adjustment training is carried out.

Formula for Multi-Layered Neural Network

Suppose we have x_n inputs (x_1, x_2, \dots, x_n) and a bias unit. Let the weight applied to be w_1, w_2, \dots, w_n . Then find the summation and bias unit on performing dot product among inputs and weights as:

$$r = \sum_{i=1}^m w_i x_i + \text{bias}$$

On feeding the r into activation function $F(r)$ we find the output for the hidden layers. For the first hidden layer h^1 , the neuron can be calculated as:

$$h_1^1 = F(r)$$

For all the other hidden layers repeat the same procedure. Keep repeating the process until reach the last weight set.

Backpropagation

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

❖ Features of Backpropagation:

1. it is the gradient descent method as used in the case of simple perceptron network with the differentiable unit.
2. it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
3. training is done in the three stages:
 - a. the feed-forward of input training pattern
 - b. the calculation and backpropagation of the error
 - c. updation of the weight

❖ Working of Backpropagation:

Neural networks use supervised learning to generate output vectors from input vectors that the network operates on. It compares generated output to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get your desired output.

❖ Backpropagation Algorithm:

Step 1: Inputs X , arrive through the preconnected path.

Step 2: The input is modeled using true weights W . Weights are usually chosen randomly.

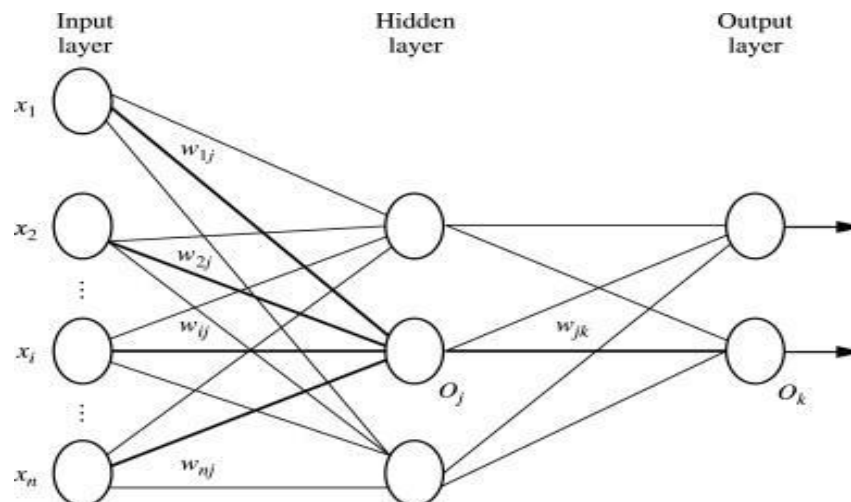
Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Step 4: Calculate the error in the outputs

Backpropagation Error = Actual Output – Desired Output

Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6: Repeat the process until the desired output is achieved.



Parameters:

- x = inputs training vector $x=(x_1, x_2, \dots, x_n)$.
- t = target vector $t=(t_1, t_2, \dots, t_n)$.
- δ_k = error at output unit.
- δ_j = error at hidden layer.
- α = learning rate.
- V_{0j} = bias of hidden unit j .

❖ Need for Backpropagation:

Backpropagation is “backpropagation of errors” and is very useful for training neural networks. It’s fast, easy to implement, and simple. Backpropagation does not require any parameters to be set, except the number of inputs. Backpropagation is a flexible method because no prior knowledge of the network is required.

❖ Types of Backpropagation:

1. **Static backpropagation:** Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
2. **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

❖ Advantages:

1. It is simple, fast, and easy to program.
2. Only numbers of the input are tuned, not any other parameter.
3. It is Flexible and efficient.
4. No need for users to learn any special functions.

❖ Disadvantages:

1. It is sensitive to noisy data and irregularities. Noisy data can lead to inaccurate results.
2. Performance is highly dependent on input data.
3. Spending too much time training.
4. The matrix-based approach is preferred over a mini-batch.

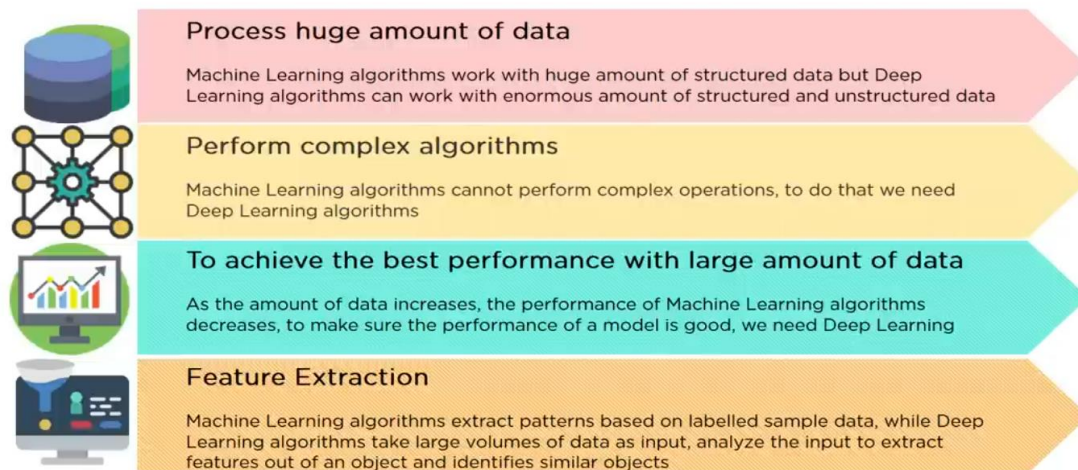
Deep Learning

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs).

These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

1. Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data.
2. The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering.
3. Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs).
4. Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the development of specialized hardware, such as Graphics Processing Units (GPUs), has made it easier to train deep neural networks.

Why do we need Deep Learning?



simplilearn

❖ Applications: -

1. Cancer Detection.
2. Robot Navigation.
3. Autonomous Driving Cars.
4. Machine Translation.
5. Music Composition.
6. Colorization of images.