MACHINE LEARNING FINAL

1. **What is machine learning? Explain types of machine learning.**

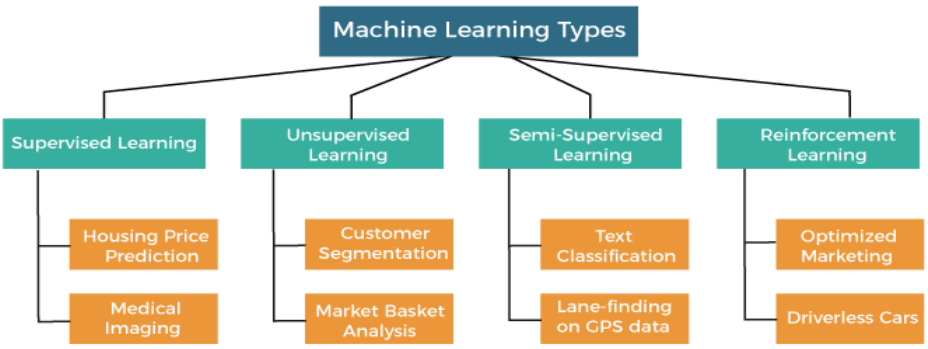
Types of Machine Learning

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



In this topic, we will provide a detailed description of the types of Machine Learning along with their respective algorithms:

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. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc.** After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears,

tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are **Risk Assessment, Fraud Detection, Spam filtering,** etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

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

a) 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:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

b) 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:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

Advantages and Disadvantages of Supervised Learning

Advantages:

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

Disadvantages:

- These algorithms are not able to solve complex tasks.

- It may predict the wrong output if the test data is different from the training data.
- It requires lots of computational time to train the algorithm.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

- **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.
- **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.
- **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.
- **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.
- **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 unlabeled 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.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

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

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. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

- **K-Means Clustering algorithm**
- **Mean-shift algorithm**
- **DBSCAN Algorithm**
- **Principal Component Analysis**
- **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 and Disadvantages of Unsupervised Learning Algorithm

Advantages:

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

Disadvantages:

- 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.
- Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

Applications of Unsupervised Learning

- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **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.

- **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

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. 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.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self-analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

Advantages and disadvantages of Semi-supervised Learning

Advantages:

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

Disadvantages:

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

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded

for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

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

The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

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).** In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **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.
- **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.

Real-world Use cases of Reinforcement Learning

- **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**.
- **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.
- **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.
- **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 and Disadvantages of Reinforcement Learning

Advantages

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

Disadvantage

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

The curse of dimensionality limits reinforcement learning for real physical systems.

Supervised Machine Learning

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

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

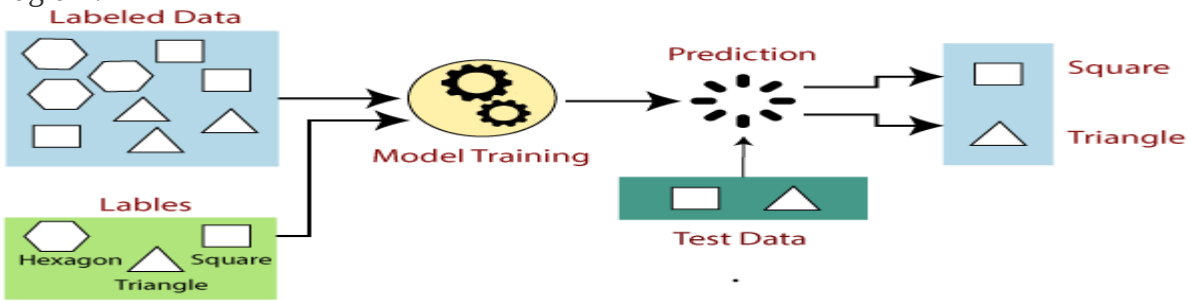
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

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

How Supervised Learning Works?

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

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



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

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

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

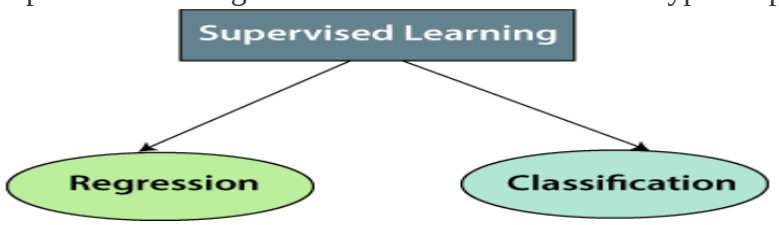
The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

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

Types of supervised Machine learning Algorithms:

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



1. Regression

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

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

2. Classification

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

Spam Filtering,

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

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

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of supervised learning:

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

Unsupervised Machine Learning

In the previous topic, we learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

What is Unsupervised Learning?

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

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

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

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

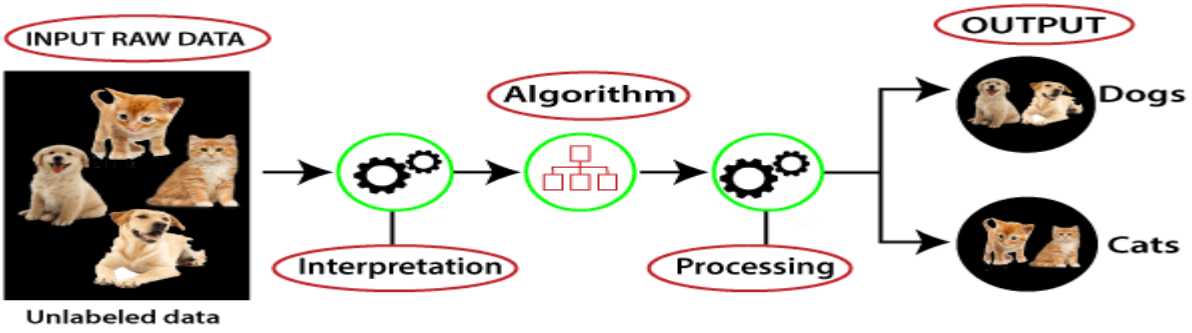
Why use Unsupervised Learning?

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

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

Working of Unsupervised Learning

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

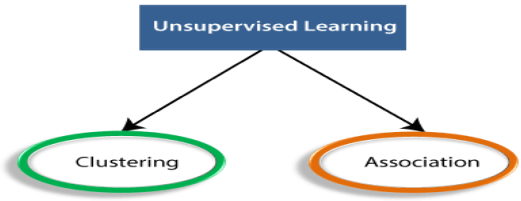


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and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

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

Types of Unsupervised Learning Algorithm:

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



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

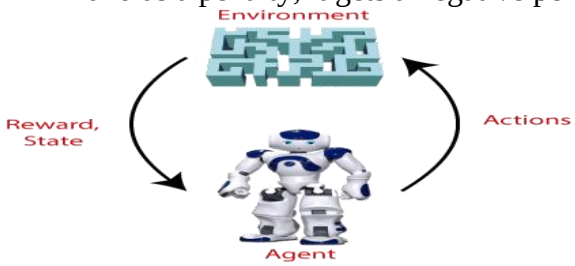
---

4. Explain reinforcement learning.

What is Reinforcement Learning?

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.

- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Key Features of Reinforcement Learning

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.
- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

1. **Value-based:**
   The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.

2. **Policy-based:**
   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy:
   - **Deterministic:** The same action is produced by the policy (π) at any state.
   - **Stochastic:** In this policy, probability determines the produced action.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

1) Policy: A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

For deterministic policy: a = π(s)
For stochastic policy: π(a | s) = P[At =a | St = s]

2) Reward Signal: The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

3) Value Function: The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.

4) Model: The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

---

5. **Explain machine learning problem categories.**
   Supervised Learning: In supervised learning, the algorithm learns from labeled data, meaning the input data is paired with the correct output. The goal is to learn a mapping function from the input variables to the output variable. Supervised learning problems can further be categorized into:
   Classification: This involves predicting a categorical label, such as classifying emails as spam or not spam, or identifying whether an image contains a cat or a dog.
   Regression: In regression, the algorithm predicts a continuous value, such as predicting house prices based on features like size, location, and number of bedrooms.

   Unsupervised Learning: Unsupervised learning involves training algorithms on data without labeled responses. The algorithm tries to learn the underlying structure or distribution in the data. Unsupervised learning problems include:
   Clustering: Clustering involves grouping similar data points together, where the algorithm identifies natural groupings in the data. For example, clustering customers based on their purchasing behavior.
   Dimensionality Reduction: This involves reducing the number of features in the data while preserving its important structure. Techniques like Principal Component Analysis (PCA) fall into this category.
   Reinforcement Learning: Reinforcement learning involves training agents to make sequential decisions in an environment to achieve a certain goal. The agent learns through trial and error, receiving feedback in the form of rewards or penalties. Reinforcement learning is commonly used in scenarios such as game playing (e.g., AlphaGo) and autonomous vehicle control.
   Understanding these categories and the types of problems they encompass is crucial for selecting the appropriate machine learning algorithm and approach for a given task.

---

6. **Explain supervised learning problem categories.**

Supervised learning uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized.

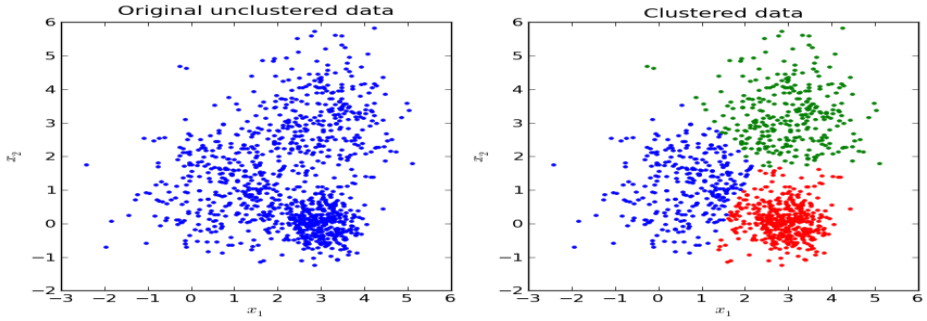
Supervised learning can be separated into two types of problems when data mining—classification and regression:

- Classification uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbor, and random forest, which are described in more detail below.
- Regression is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as for sales revenue for a

given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms.

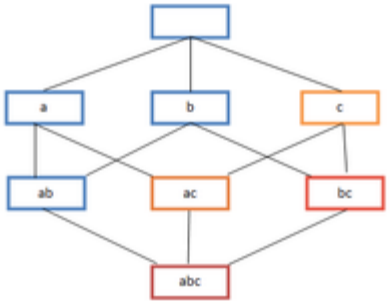
7. Explain unsupervised learning problem categories

Two types of unsupervised learning problems



We can think of unsupervised learning problems as being divided into two categories: **clustering and association rules.**

Clustering is an unsupervised learning technique, which groups unlabeled data points based on their similarity and differences. Hence, points are grouped into clusters in such a way that those in a same group have the most similarity with each other, while points in different groups have little to no similarities. To do so, cluster analysis finds features or characteristics among the data points and groups them based on the absence or presence of these features. Clustering methods include k-means clustering, hierarchical clustering, or even probabilistic clustering.



On the other hand, association rules are another form of unsupervised learning, which find relationships between points in a dataset. In other words, these algorithms find the points that occur together in a database. These methods are often used for market basket analysis, which allows companies to understand the relationship between the purchase of different products. Indeed, it would allow to establish relationships of the form : “People who buy product X also tend to buy product Y”. Association algorithms include the Apriori algorithm, the Eclat algorithm and the FP-growth algorithm.

There are various unsupervised learning algorithms, which we will dive into in the following article. Please take a look at it if you wish to learn more about these methods.

8. Draw and explain machine learning architecture.

Architecting the Machine Learning Process

Fig:- Block diagram of decision flow architecture for **Machine learning systems**,

Let us now try to understand the layers represented in the image above.

1. Data Acquisition

As machine learning is based on available data for the system to make a decision hence the first step defined in the architecture is data acquisition. This involves data collection,

preparing and segregating the case scenarios based on certain features involved with the decision making cycle and forwarding the data to the processing unit for carrying out further categorization. This stage is sometimes called the data preprocessing stage. The data model expects reliable, fast and elastic data which may be discrete or continuous in nature. The data is then passed into stream processing systems (for continuous data) and stored in batch **data warehouses** (for discrete data) before being passed on to data modeling or processing stages.

2. Data Processing

The received data in the data acquisition layer is then sent forward to the data processing layer where it is subjected to advanced integration and processing and involves normalization of the data, data cleaning, transformation, and encoding. The **data processing** is also dependent on the type of learning being used. For e.g., if supervised learning is being used the data shall be needed to be segregated into multiple steps of sample data required for training of the system and the data thus created is called training sample data or simply training data. Also, the data processing is dependent upon the kind of processing required and may involve choices ranging from action upon continuous data which will involve the use of specific function-based architecture, for example, lambda architecture, Also it might involve action upon discrete data which may require memory-bound processing. The data processing layer defines if the memory processing shall be done to data in transit or in rest.

3. Data Modeling

This layer of the architecture involves the selection of different algorithms that might adapt the system to address the problem for which the learning is being devised, These algorithms are being evolved or being inherited from a set of libraries. The algorithms are used to model the data accordingly, this makes the system ready for the execution step.

4. Execution

This stage in machine learning is where the experimentation is done, testing is involved and tunings are performed. The general goal behind being to optimize the algorithm in order to extract the required machine outcome and maximize the system performance, The output of the step is a refined solution capable of providing the required data for the machine to make decisions.

5. Deployment

Like any other software output, ML outputs need to be operationalized or be forwarded for further exploratory processing. The output can be considered as a non-deterministic query which needs to be further deployed into the decision-making system.

It is advised to seamlessly move the ML output directly to production where it will enable the machine to directly make decisions based on the output and reduce the dependency on the further exploratory steps.

9. **Draw and explain machine learning lifecycle**.

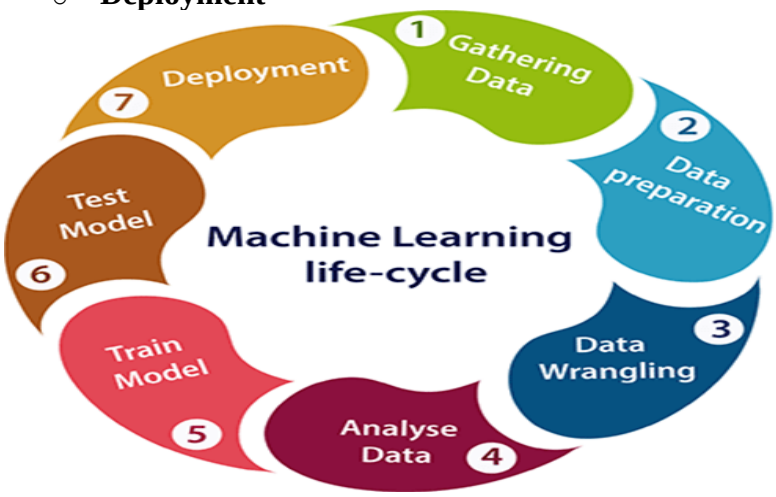
Machine learning Life cycle

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:

- **Gathering Data**

- **Data preparation**
- **Data Wrangling**
- **Analyse Data**
- **Train the model**
- **Test the model**
- **Deployment**



The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

---

1. Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**

By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

---

2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

- **Data exploration:**
  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.
  A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- **Data pre-processing:**
  Now the next step is preprocessing of data for its analysis.

---

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

---

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
- **Building models**
- **Review the result**

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the

problems, where we select the machine learning techniques such as **Classification**, **Regression**, **Cluster analysis**, **Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

---

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

---

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

---

7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Explain performance measures for machine learning

Performance Metrics in Machine Learning

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. ***To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.*** These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.

In machine learning, each task or problem is divided into **classification** and **Regression**. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. In this topic, we will discuss metrics used for classification and regression tasks.

1. Performance Metrics for Classification

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as *Yes or No, 0 or 1, Spam or Not Spam,* etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- **Accuracy**
- **Confusion Matrix**
- **Precision**
- **Recall**
- **F-Score**
- **AUC(Area Under the Curve)-ROC**

I. Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

It can be formulated as:

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

To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this.

Firstly, we need to import the *accuracy_score* function of the scikit-learn library as follows:

```
1. from sklearn.metrics import accuracy_score
2.
3. Here, metrics is a class of sklearn.
4.
5. Then we need to pass the ground truth and predicted values in the function to calculat
   e the accuracy.
```

6.
7. print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')

Although it is simple to use and implement, it is suitable only for cases where an equal number of samples belong to each class.

When to Use Accuracy?

It is good to use the Accuracy metric when the target variable classes in data are approximately balanced. For example, if 60% of classes in a fruit image dataset are of Apple, 40% are Mango. In this case, if the model is asked to predict whether the image is of Apple or Mango, it will give a prediction with 97% of accuracy.

When not to use Accuracy?

It is recommended not to use the Accuracy measure when the target variable majorly belongs to one class. For example, Suppose there is a model for a disease prediction in which, out of 100 people, only five people have a disease, and 95 people don't have one. In this case, if our model predicts every person with no disease (which means a bad prediction), the Accuracy measure will be 95%, which is not correct.

II. Confusion Matrix

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.

The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.

A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. **True Positive(TP):** In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

III. Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive).

$$Precision = \frac{TP}{(TP + FP)}$$

IV. Recall or Sensitivity

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).

The formula for calculating Recall is given below:

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

When to use Precision and Recall?

From the above definitions of Precision and Recall, we can say that recall determines the performance of a classifier with respect to a false negative, whereas precision gives information about the performance of a classifier with respect to a false positive.

So, if we want to minimize the false negative, then, Recall should be as near to 100%, and if we want to minimize the false positive, then precision should be close to 100% as possible.

In simple words, *if we maximize precision, it will minimize the FP errors, and if we maximize recall, it will minimize the FN error.*

V. F-Scores

F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, ***the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.***

The formula for calculating the F1 score is given below:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

When to use F-Score?

As F-score make use of both precision and recall, so it should be used if both of them are important for evaluation, but one (precision or recall) is slightly more important to consider than the other. For example, when False negatives are comparatively more important than false positives, or vice versa.

VI. AUC-ROC

Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. It is one of the popular and important metrics for evaluating the performance of the classification model.

Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ***ROC represents a graph to show the performance of a classification model at different threshold levels***. The curve is plotted between two parameters, which are:

- **True Positive Rate**
- **False Positive Rate**

TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

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

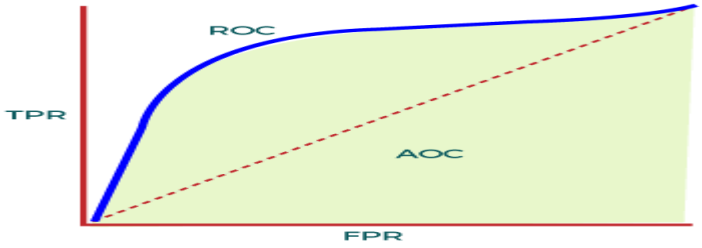
FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

To calculate value at any point in a ROC curve, we can evaluate a logistic regression model multiple times with different classification thresholds, but this would not be much efficient. So, for this, one efficient method is used, which is known as AUC.

AUC: Area Under the ROC curve

AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:



AUC calculates the performance across all the thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1. It means a model with 100% wrong prediction will have an AUC of 0.0, whereas models with 100% correct predictions will have an AUC of 1.0.

When to Use AUC

AUC should be used to measure how well the predictions are ranked rather than their absolute values. Moreover, it measures the quality of predictions of the model without considering the classification threshold.

When not to use AUC

As AUC is scale-invariant, which is not always desirable, and we need calibrating probability outputs, then AUC is not preferable.

Further, AUC is not a useful metric when there are wide disparities in the cost of false negatives vs. false positives, and it is difficult to minimize one type of classification error.

2. Performance Metrics for Regression

Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The metrics used for regression are different from the classification metrics. It means we cannot use the Accuracy metric (explained above) to evaluate a regression model; instead, the performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models.

- **Mean Absolute Error**
- **Mean Squared Error**
- **R2 Score**
- **Adjusted R2**

I. Mean Absolute Error (MAE)

Mean Absolute Error or MAE is one of the simplest metrics, which measures the absolute difference between actual and predicted values, where absolute means taking a number as Positive.

To understand MAE, let's take an example of Linear Regression, where the model draws a best fit line between dependent and independent variables. To measure the MAE or error in prediction, we need to calculate the difference between actual values and predicted values. But in order to find the absolute error for the complete dataset, we need to find the mean absolute of the complete dataset.

The below formula is used to calculate MAE:

$$MAE = 1/N \sum |Y - Y'|$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

MAE is much more robust for the outliers. One of the limitations of MAE is that it is not differentiable, so for this, we need to apply different optimizers such as Gradient Descent. However, to overcome this limitation, another metric can be used, which is Mean Squared Error or MSE.

II. Mean Squared Error

Mean Squared error or MSE is one of the most suitable metrics for Regression evaluation. It measures the average of the Squared difference between predicted values and the actual value given by the model.

Since in MSE, errors are squared, therefore it only assumes non-negative values, and it is usually positive and non-zero.

Moreover, due to squared differences, it penalizes small errors also, and hence it leads to over-estimation of how bad the model is.

MSE is a much-preferred metric compared to other regression metrics as it is differentiable and hence optimized better.

The formula for calculating MSE is given below:

$$MSE = 1/N \sum (Y - Y')^2$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

III. R Squared Score

R squared error is also known as Coefficient of Determination, which is another popular metric used for Regression model evaluation. The R-squared metric enables us to compare our model with a constant baseline to determine the performance of the model. To select the constant baseline, we need to take the mean of the data and draw the line at the mean.

The R squared score will always be less than or equal to 1 without concerning if the values are too large or small.

$$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$$

IV. Adjusted R Squared

Adjusted R squared, as the name suggests, is the improved version of R squared error. R square has a limitation of improvement of a score on increasing the terms, even though the model is not improving, and it may mislead the data scientists.

To overcome the issue of R square, adjusted R squared is used, which will always show a lower value than R^2. It is because it adjusts the values of increasing predictors and only shows improvement if there is a real improvement.

We can calculate the adjusted R squared as follows:

$$R_a^2 = 1-\left[\left(\frac{n-1}{n-k-1}\right)\times\left(1-R^2\right)\right]$$

Unit 2

1. **Explain simple linear regression.**

Simple Linear Regression

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

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

where:

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

Simple Linear Regression in Machine Learning

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

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

Simple Linear regression algorithm has mainly two objectives:

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

Simple Linear Regression Model:

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

$y= a_0+a_1x+ \varepsilon$

Where,

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

Implementation of Simple Linear Regression Algorithm using Python

Problem Statement example for Simple Linear Regression:

Here we are taking a dataset that has two variables: salary (dependent variable) and experience (Independent variable). The goals of this problem is:

- **We want to find out if there is any correlation between these two variables**
- **We will find the best fit line for the dataset.**
- **How the dependent variable is changing by changing the independent variable.**

In this section, we will create a Simple Linear Regression model to find out the best fitting line for representing the relationship between these two variables.

By executing the above lines of code, we will get the below graph plot as an output.

In the above plot, we can see the real values observations in green dots and predicted values are covered by the red regression line. The regression line shows a correlation between the dependent and independent variable.

The good fit of the line can be observed by calculating the difference between actual values and predicted values. But as we can see in the above **plot, most of the observations are close to the regression line, hence our model is good for the training set**.

---

2. Explain gradient descent for simple linear regression.

Gradient Descent in Linear Regression

We know that in any machine learning project our main aim relies on how good our project accuracy is or how much our model prediction differs from the actual data point. Based on the difference between model prediction and actual data points we try to find the parameters of the model which give better accuracy on our dataset\, In order to find these parameters we apply gradient descent on the cost function of the machine learning model.

What is Gradient Descent

Gradient Descent is an iterative optimization algorithm that tries to find the optimum value (Minimum/Maximum) of an objective function. It is one of the most used optimization techniques in machine learning projects for updating the parameters of a model in order to minimize a cost function.

The main aim of gradient descent is to find the best parameters of a model which gives the highest accuracy on training as well as testing datasets. In gradient descent, The gradient is a vector that points in the direction of the steepest increase of the function at a specific point. Moving in the opposite direction of the gradient allows the algorithm to gradually descend towards lower values of the function, and eventually reaching to the minimum of the function.

Steps Required in Gradient Descent Algorithm

- **Step 1** we first initialize the parameters of the model randomly
- **Step 2** Compute the gradient of the cost function with respect to each parameter. It involves making partial differentiation of cost function with respect to the parameters.
- **Step 3** Update the parameters of the model by taking steps in the opposite direction of the model. Here we choose a hyperparameter learning rate which is denoted by alpha. It helps in deciding the step size of the gradient.
- **Step 4** Repeat steps 2 and 3 iteratively to get the best parameter for the defined model

o apply this gradient descent on data using any programming language we have to make four new functions using which we can update our parameter and apply it to data to make a prediction. We will see each function one by one and understand it

1. **gradient_descent –** In the gradient descent function we will make the prediction on a dataset and compute the difference between the predicted and actual target value and accordingly we will update the parameter and hence it will return the updated parameter.
2. **compute_predictions –** In this function, we will compute the prediction using the parameters at each iteration.
3. **compute_gradient –** In this function we will compute the error which is the difference between the actual and predicted target value and then compute the gradient using this error and training data.
4. **update_parameters –** In this separate function we will update the parameter using learning rate and gradient that we got from the compute_gradient function.

Mathematics Behind Gradient Descent

In the Machine Learning Regression problem, our model targets to get the best-fit regression line to predict the value y based on the given input value (x). While training the model, the model calculates the cost function like Root Mean Squared error between the predicted value (pred) and true value (y). Our model targets to minimize this cost function. To minimize this cost function, the model needs to have the best value of θ_1 and θ_2(for Univariate linear regression problem). Initially model selects θ_1 and θ_2 values randomly and then iteratively update these value in order to minimize the cost function until it reaches the minimum. By the time model achieves the minimum cost function, it will have the best θ_1 and θ_2 values. Using these updated values of θ_1 and θ_2 in the hypothesis equation of linear equation, our model will predict the output value y.

How do θ_1 and θ_2 values get updated?

Linear Regression Cost

Function:

so our model aim is to Minimize \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) – y^{(i)})^2 and store the parameters which makes it minimum.

Gradient Descent Algorithm For Linear Regression

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y_i]^2$$

($h_\Theta(x_i)$: Predicted Value; y_i: True Value)

Gradient Descent

$$\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)$$

(α: Learning Rate)

Now,

$$\frac{\partial}{\partial\Theta}J_\Theta = \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y)$$

$$= \frac{1}{m}(h_\Theta(x_i) - y)x_i$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

Gradient descent algorithm for linear regression

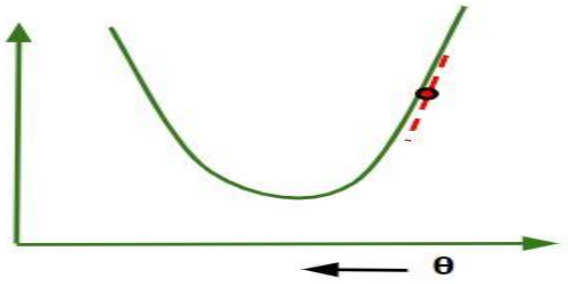
-> **θ_j** : Weights of the hypothesis.
-> **$h_{\theta(xi)}$** : predicted y value for i^{th} input.
-> **i** : Feature index number (can be 0, 1, 2, ..., n).
-> **α** : Learning Rate of Gradient Descent.

How Does Gradient Descent Work

Gradient descent works by moving downward toward the pits or valleys in the graph to find the minimum value. This is achieved by taking the derivative of the cost function, as illustrated in the figure below. During each iteration, gradient descent step-downs the cost function in the direction of the steepest descent. By adjusting the parameters in this direction, it seeks to reach the minimum of the cost function and find the best-fit values for the parameters. The size of each step is determined by parameter **α** known as **Learning Rate**.

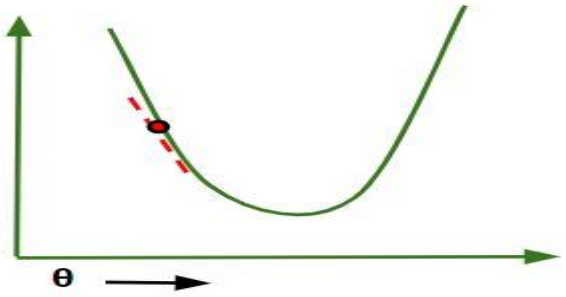
In the Gradient Descent algorithm, one can infer two points :

- **If slope is +ve** : $\theta_j = \theta_j$ – (+ve value). Hence the value of θ_j decreases.



If slope is +ve in Gradient Descent

- **If slope is -ve** : $\theta_j = \theta_j$ – (-ve value). Hence the value of θ_j increases.

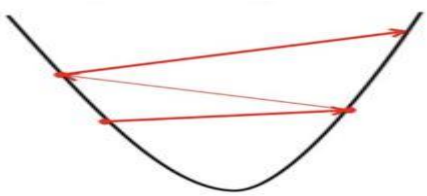


If slope is -ve in Gradient Descent

How To Choose Learning Rate

The choice of correct learning rate is very important as it ensures that Gradient Descent converges in a reasonable time. :

- If we choose **α to be very large**, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.



Effect of large alpha value on Gradient Descent

- If we choose α to be very small, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.

Effect of small alpha value on Gradient Descent

Python Implementation of Gradient Descent

At first, we will import all the necessary Python libraries that we will need for mathematical computation and plotting like **numpy** for mathematical operations and **matplotlib** for plotting. Then we will define a class **Linear_Regression** that represents the linear regression model.

We will make a **update_coeffs** method inside the class to update the coefficients (parameters) of the linear regression model using gradient descent. To calculate the error between the predicted output and the actual output we will make a **predict** method that will make predictions using the current model coefficients.

For updating and calculating the gradient of the error we will make **compute_cost** which will apply gradient descent on (mean squared error) between the predicted values and the actual values

---

3. What is hypothesis function for simple linear regression?

In simple linear regression, the hypothesis function $h\theta(x)h\theta(x)$ is represented as:

$h\theta(x)=\theta 0+\theta 1xh\theta(x)=\theta 0+\theta 1x$

Where:

- $h\theta(x)h\theta(x)$ is the predicted output (the dependent variable) based on the input xx.
- $\theta 0\theta 0$ is the y-intercept (bias term).
- $\theta 1\theta 1$ is the coefficient (slope) associated with the input feature xx.

This function essentially represents a straight line that best fits the relationship between the input variable xx and the output variable. The goal of linear regression is to find the optimal values for $\theta 0\theta 0$ and $\theta 1\theta 1$ that minimize the difference between the actual output and the predicted output.

Hypothesis Function:
The hypothesis function in simple linear regression is a linear equation that describes the relationship between the input variable (predictor) and the output variable (response). It's represented as:

$h\theta(x)=\theta0+\theta1xh\theta(x)=\theta0+\theta1x$

Here,

- $h\theta(x)h\theta(x)$ represents the predicted output (response variable) for a given input xx.
- $\theta0\theta0$ is the intercept term or bias. It represents the value of $h\theta(x)h\theta(x)$ when $x=0x=0$.
- $\theta1\theta1$ is the coefficient of the input variable xx, also known as the slope. It indicates the rate of change of $h\theta(x)h\theta(x)$ with respect to xx.

Interpretation:

- The intercept term $\theta0\theta0$ determines the position of the regression line on the y-axis. It indicates the predicted value of yy when $x=0x=0$.
- The coefficient $\theta1\theta1$ represents the change in the predicted output $h\theta(x)h\theta(x)$ for a one-unit change in the input variable xx. If $\theta1>0\theta1>0$, it implies a positive relationship between xx and $h\theta(x)h\theta(x)$, whereas $\theta1<0\theta1<0$ indicates a negative relationship.

Optimization:
The goal of linear regression is to find the optimal values of $\theta0\theta0$ and $\theta1\theta1$ that minimize the difference between the actual output and the predicted output. This is typically achieved by minimizing a cost function, often the mean squared error (MSE), which measures the average squared difference between the actual and predicted values over the entire dataset.

Learning:
Linear regression models are trained using various optimization algorithms such as gradient descent or normal equation. Gradient descent iteratively updates the parameters ($\theta0\theta0$ and $\theta1\theta1$) in the direction of steepest descent of the cost function until convergence is reached. Normal equation provides a closed-form solution for the optimal parameters by solving a system of linear equations.

Assumptions:
Simple linear regression makes several assumptions, including linearity (the relationship between xx and yy is linear), independence (observations are independent of each other), homoscedasticity (constant variance of residuals), and normality (residuals are normally distributed).

Extensions:
While simple linear regression models the relationship between one input variable and one output variable, multiple linear regression extends this to multiple input variables. Polynomial regression involves fitting a nonlinear relationship between the input and output variables by introducing polynomial terms.

Understanding the hypothesis function and its implications is crucial for interpreting and using linear regression models effectively in various applications.

4. **Explain simple regression in matrix form.**

Chapter 5: Linear Regression in Matrix Form

The SLR Model in Scalar Form

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim^{iid} N(0, \sigma^2)$$

Consider now writing an equation for each observation:

$$\begin{aligned} Y_1 &= \beta_0 + \beta_1 X_1 + \epsilon_1 \\ Y_2 &= \beta_0 + \beta_1 X_2 + \epsilon_2 \\ &\vdots \quad \vdots \quad \vdots \\ Y_n &= \beta_0 + \beta_1 X_n + \epsilon_n \end{aligned}$$

The SLR Model in Matrix Form

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 X_1 \\ \beta_0 + \beta_1 X_2 \\ \vdots \\ \beta_0 + \beta_1 X_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

(I will try to use **bold** symbols for matrices. At first, I will also indicate the dimensions as a subscript to the symbol.)

- $\mathbf{X}$ is called the design matrix.
- β is the vector of parameters
- ϵ is the error vector
- $\mathbf{Y}$ is the response vector

The Design Matrix

$$\mathbf{X}_{n\times 2} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix}$$

Vector of Parameters

$$\beta_{2\times 1} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Vector of Error Terms

$$\epsilon_{n\times 1} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Vector of Responses

$$\mathbf{Y}_{n\times 1} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

Thus,

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\beta + \epsilon \\ \mathbf{Y}_{n\times 1} &= \mathbf{X}_{n\times 2}\beta_{2\times 1} + \epsilon_{n\times 1} \end{aligned}$$

5. **Explain Least Squares in Matrix Form.**

Least Square Method

In statistics, when we have data in the form of data points that can be represented on a cartesian plane by taking one of the variables as the independent variable represented as the x-coordinate and the other one as the dependent variable represented as the y-coordinate, it is called scatter data. This data might not be useful in making interpretations or predicting the

values of the dependent variable for the independent variable where it is initially unknown. So, we try to get an equation of a line that fits best to the given data points with the help of the **Least Square Method**.
The method uses averages of the data points and some formulae discussed as follows to find the slope and intercept of the line of best fit. This line can be then used to make further interpretations about the data and to predict the unknown values. The **Least Squares Method** provides accurate results only if the scatter data is evenly distributed and does not contain outliers.

What is Least Square Method?

The Least Squares Method is used to derive a generalized linear equation between two variables, one of which is independent and the other dependent on the former. The value of the independent variable is represented as the x-coordinate and that of the dependent variable is represented as the y-coordinate in a 2D cartesian coordinate system. Initially, known values are marked on a plot. The plot obtained at this point is called a scatter plot. Then, we try to represent all the marked points as a straight line or a linear equation. The equation of such a line is obtained with the help of the least squares method. This is done to get the value of the dependent variable for an independent variable for which the value was initially unknown. This helps us to fill in the missing points in a data table or forecast the data. The method is discussed in detail as follows.

Least Square Method Definition

The least-squares method can be defined as a statistical method that is used to find the equation of the line of best fit related to the given data. This method is called so as it aims at reducing the sum of squares of deviations as much as possible. The line obtained from such a method is called a **regression line**.

Formula of Least Square Method

The formula used in the least squares method and the steps used in deriving the line of best fit from this method are discussed as follows:

- **Step 1:** Denote the independent variable values as x_i and the dependent ones as y_i.
- **Step 2:** Calculate the average values of x_i and y_i as X and Y.
- **Step 3:** Presume the equation of the line of best fit as y = mx + c, where m is the slope of the line and c represents the intercept of the line on the Y-axis.
- **Step 4:** The slope m can be calculated from the following formula:

$$m = [\Sigma (X - x_i) \times (Y - y_i)] / \Sigma (X - x_i)^2$$

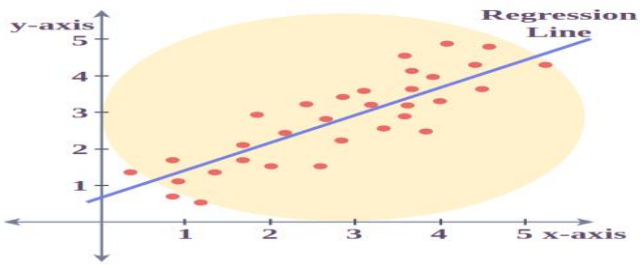
- **Step 5:** The intercept c is calculated from the following formula:

$$c = Y - mX$$

Thus, we obtain the line of best fit as y = mx + c, where values of m and c can be calculated from the formulae defined above.

Least Square Method Graph

Let us have a look at how the data points and the line of best fit obtained from the least squares method look when plotted on a graph.



The red points in the above plot represent the data points for the sample data available. Independent variables are plotted as x-coordinates and dependent ones are plotted as y-coordinates. The equation of the line of best fit obtained from the least squares method is plotted as the red line in the graph.
We can conclude from the above graph that how the least squares method helps us to find a line that best fits the given data points and hence can be used to make further predictions about the value of the dependent variable where it is not known initially.

Limitations of the Least Square Method

The least squares method assumes that the data is evenly distributed and doesn't contain any outliers for deriving a line of best fit. But, this method doesn't provide accurate results for unevenly distributed data or for data containing outliers.

Check: Least Square Regression Line

Least Square Method Formula

The Least Square Method formula is used to find the best-fitting line through a set of data points by minimizing the sum of the squares of the vertical distances (residuals) of the points from the line. For a simple linear regression, which is a line of the form $y=ax+b$, where y is the dependent variable, x is the independent variable, a is the slope of the line, and b is the y-intercept, the formulas to calculate the slope (a) and intercept (b) of the line are derived from the following equations:

1. **Slope (a) Formula:** $a=n(\sum xy)-(\sum x)(\sum y) \,/\, n(\sum x2)-(\sum x)^2$
2. **Intercept (b) Formula:** $b=(\sum y)-a(\sum x) \,/\, n$

Where:

- n is the number of data points,
- $\sum xy$ is the sum of the product of each pair of x and y values,
- $\sum x$ is the sum of all x values,
- $\sum y$ is the sum of all y values,
- $\sum x^2$ is the sum of the squares of x values.

These formulas are used to calculate the parameters of the line that best fits the data according to the criterion of the least squares, minimizing the sum of the squared differences between the observed values and the values predicted by the linear model.

Least Square Method Solved Examples

Problem 1: Find the line of best fit for the following data points using the least squares method: (x,y) = (1,3), (2,4), (4,8), (6,10), (8,15).

Solution:

Here, we have x as the independent variable and y as the dependent variable. First, we calculate the means of x and y values denoted by X and Y respectively.

$X = (1+2+4+6+8)/5 = 4.2$

$Y = (3+4+8+10+15)/5 = 8$

x_i	y_i	$X - x_i$	$Y - y_i$	$(X-x_i)*(Y-y_i)$	$(X - x_i)^2$
1	3	3.2	5	16	10.24
2	4	2.2	4	8.8	4.84
4	8	0.2	0	0	0.04
6	10	-1.8	-2	3.6	3.24
8	15	-3.8	-7	26.6	14.44
Sum (Σ)		0	0	55	32.8

The slope of the line of best fit can be calculated from the formula as follows:

$$m = (\Sigma (X - x_i)*(Y - y_i)) / \Sigma(X - x_i)^2$$

$$m = 55/32.8 = 1.68 \text{ (rounded upto 2 decimal places)}$$

Now, the intercept will be calculated from the formula as follows:

$$c = Y - mX$$

$$c = 8 - 1.68*4.2 = 0.94$$

Thus, the equation of the line of best fit becomes, $y = 1.68x + 0.94$.

How Do You Calculate Least Squares?

To calculate the least squares solution, you typically need to:

1. Determine the equation of the line you believe best fits the data.
2. Calculate the residuals (differences) between the observed values and the values predicted by your model.
3. Square each of these residuals and sum them up.
4. Adjust the model to minimize this sum.

---

5. Explain Sampling Distribution of Estimators.

6.3 Sampling Distributions and Estimators

Key Concept: In addition to knowing how individual data values vary about the mean for a population, statisticians are interested in knowing how the means of samples of the same size taken from the same population vary about the population mean.

We now consider the concept of a **sampling distribution of a statistic.** Instead of working with values from the original population, we want to focus on the values of **statistics** (such as sample proportions or sample means) obtained from the population.

6.3 Sampling Distributions and Estimators

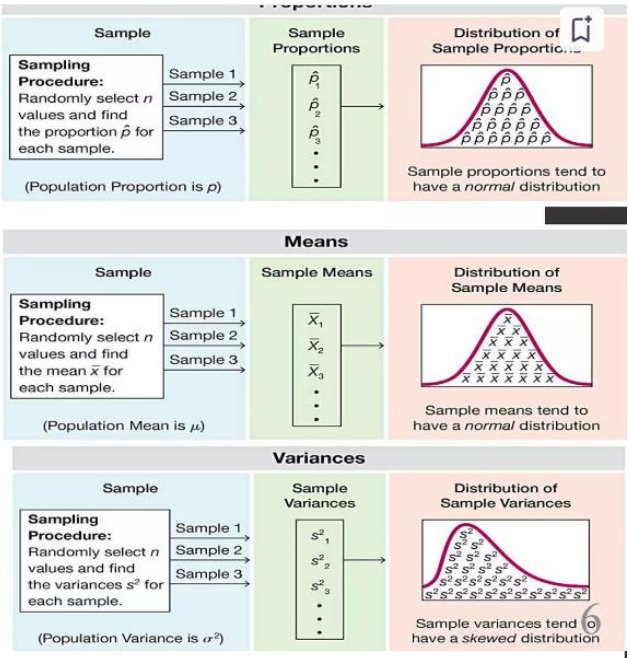
Sampling Distribution of a Statistic: The **sampling distribution of a statistic** (such as a sample proportion or sample mean) **is the distribution of all values of the statistic when all possible samples of the same size *n* are taken from the same population.** (The sampling distribution of a statistic is typically represented as a probability distribution in the format of a probability histogram, formula, or table.)

population proportion: p

sample proportion: $\hat{p}$ $(p - hat)$

General Behavior of Sampling Distributions:

1. **Sample proportions tend to be normally distributed.**
2. **The mean of sample proportions is the same as the population mean.**

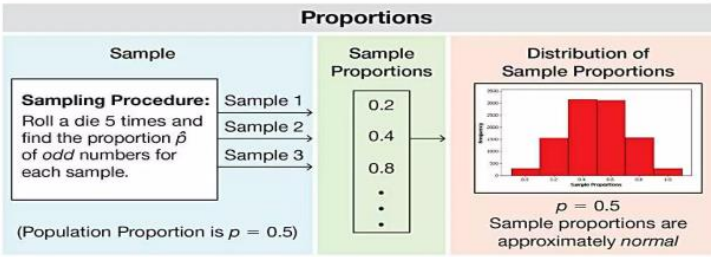


Behavior of Sample Proportions: $\hat{p}$

1. The distribution of sample proportions tends to approximate a normal distribution.
2. Sample proportions target the values of the population proportion (the mean of all the sample proportion: $\hat{p}$ $(p - hat)$ is equal to the population proportion p.).

Example 1: Consider repeating this process:

Roll a die 5 times and find the proportion of **odd** numbers (1 or 3 or 5). What do we know about the behavior of all sample proportions that are generated as this process continues indefinitely?



The figure illustrates this process repeated for 10,000 times (the sampling distribution of the sample proportion should be repeated process indefinitely). The figure shows that **the sample proportions are approximately normally distributed.**

(1, 2, 3, 4, 5, 6 are all equally likely: 1/6)

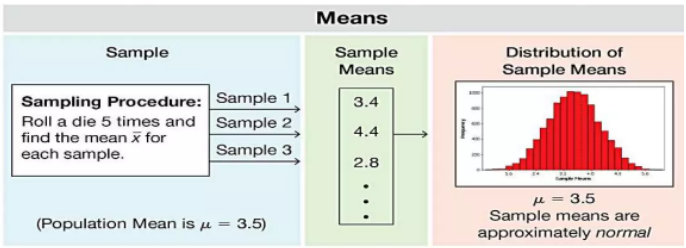
the proportion of odd numbers in the population: 0.5,

The figure shows that **the sample proportions have a mean of 0.50**.)

Behavior of Sample Mean: $\bar{x}$

1. The distribution of sample means tends to be a normal distribution. (This will be discussed further in the following section, but the distribution tends to become closer to a normal distribution as the sample size increases.)
2. The sample means **target** the value of the population mean. (That is, the mean of the sample means is the population mean. The expected value of the sample mean is equal to the population mean.)

Example 2: Consider repeating this process: Roll a die 5 times to randomly select 5 values from the population {1, 2, 3, 4 ,5, 6}, then find the mean $\bar{x}$ of the result. What do we know about the behavior of all sample means that are generated as this process continues indefinitely?



The figure illustrates a process of rolling a die 5 times and finding the mean of the results. The figure shows results from repeating this process 10,000 times, but the true sampling distribution of the mean involves repeating the process indefinitely.

(1, 2, 3, 4, 5, 6 are all equally likely: 1/6), the population has a mean of

$$\mu = E(x) = \sum x \cdot p(x) = 3.5.$$

The 10,000 sample means included in the figure have a mean of 3.5. If the process is continued indefinitely, **the mean of the**

Sampling Distribution of the Sample Variance

The **sampling distribution of the sample variance** is the distribution of sample variances (the variable s^2), with all samples having the same sample size n taken from the same population. (The sampling distribution of the sample variance is typically represented as a probability distribution in the format of a table, probability histogram, or formula.)

Behavior of Sample Variances:

The distribution of sample variances tends to be a distribution skewed to the right.

1. The sample variances **target** the value of the population variance. (That is, the mean of the sample variances is the population variance. The expected value of the sample variance is equal to the population variance.)

Population variance:

$$\sigma^2 = \frac{\sum (x-\mu)^2}{N}$$

Population standard deviation:

$$\sigma = \sqrt{\frac{\sum (x-\mu)^2}{N}}$$

---

8. Explain multivariate linear regression.

Multivariate linear regression

Introduction

In the previous tutorial we just figured out how to solve a simple linear regression model. A dependent variable guided by a single independent variable is a good start but of very less use in real world scenarios. Generally one dependent variable depends on multiple factors. For example, the rent of a house depends on many factors like the neighborhood it is in, size of it, no.of rooms, attached facilities, distance of nearest station from it, distance of nearest shopping area from it, etc. How do we deal with such scenarios? Let's jump into **multivariate linear regression** and figure this out.

Multivariate Linear Regression

This is quite similar to the simple linear regression model we have discussed previously, but with multiple independent variables contributing to the dependent variable and hence multiple coefficients to determine and complex computation due to the added variables. Jumping straight into the equation of multivariate linear regression,

$$Yi=\alpha+\beta 1xi(1)+\beta 2xi(2)+....+\beta nxi(n)$$

Yi is the estimate of ith component of dependent variable y, where we have **n** independent variables and xij denotes the ith component of the jth independent variable/feature.
Similarly cost function is as follows,

$$E(\alpha,\beta 1,\beta 2,...,\beta n)=12m\sum i=1m(yi-Yi)$$

where we have **m** data points in training data and y is the observed data of dependent variable.
As per the formulation of the equation or the cost function, it is pretty straight forward generalization of simple linear regression. But computing the parameters is the matter of interest here.

Computing parameters

Generally, when it comes to multivariate linear regression, we don't throw in all the independent variables at a time and start minimizing the error function. First one should focus on selecting the best possible independent variables that contribute well to the dependent variable. For this, we go on and construct a correlation matrix for all the independent variables and the dependent variable from the observed data. The correlation value gives us an idea about which variable is significant and by what factor. From this matrix we pick independent variables in decreasing order of correlation value and run the regression model to estimate the coefficients by minimizing the error function. We stop when there is no prominent improvement in the estimation function by inclusion of the next independent feature. This method can still get complicated when there are large no.of independent features that have significant contribution in deciding our dependent variable. Let's discuss the normal method first which is similar to the one we used in univariate linear regression.

Normal Equation

Now let us talk in terms of matrices as it is easier that way. As discussed before, if we have n independent variables in our training data, our matrix X has n+1 rows, where the first row is the $0th$ term added to each vector of independent variables which has a value of 1 (this is the coefficient of the constant term α). So, X is as follows,

$$X=[X1..Xm]$$

Xi contains n entries corresponding to each feature in training data of ith entry. So, matrix **X** has m rows and n+1 columns ($0thcolumn$ is all $1s$ and rest for one independent variable each).

$$Y=[Y1Y2..Ym\ ]$$

and coefficient matrix C,

$$C=[\alpha\beta 1..\beta n]$$

and our final equation for our hypothesis is,

$$Y=XC$$

To calculate the coefficients, we need **n+1** equations and we get them from the minimizing condition of the error function. Equating partial derivative of $E(\alpha,\beta 1,\beta 2,...,\beta n)$ with each of the coefficients to 0 gives a system of n+1 equations. Solving these is a complicated step and gives the following nice result for matrix C,

$$C=(XTX)-1XTy$$

where y is the matrix of the observed values of dependent variable.
This method seems to work well when the n value is considerably small (approximately for 3-digit values of n). As n grows big the above computation of matrix inverse and multiplication take large amount of time. In future tutorials lets discuss a different method that can be used for data with large no.of features.

---

9. **What is hypothesis function for multivariate linear regression?**
   Unit 3

Logistic Regression in Machine Learning

Logistic regression is a **supervised machine learning algorithm** used for **classification tasks** where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors. The article explores the fundamentals of logistic regression, it's types and implementations.

What is Logistic Regression?

Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.
For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 otherwise it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.

Key Points:

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

Logistic Function – Sigmoid Function

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

Types of Logistic Regression

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

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".

Assumptions of Logistic Regression

We will explore the assumptions of logistic regression as understanding these assumptions is important to ensure that we are using appropriate application of the model. The assumption include:

1. Independent observations: Each observation is independent of the other. meaning there is no correlation between any input variables.
2. Binary dependent variables: It takes the assumption that the dependent variable must be binary or dichotomous, meaning it can take only two values. For more than two categories SoftMax functions are used.
3. Linearity relationship between independent variables and log odds: The relationship between the independent variables and the log odds of the dependent variable should be linear.
4. No outliers: There should be no outliers in the dataset.
5. Large sample size: The sample size is sufficiently large

Terminologies involved in Logistic Regression

Here are some common terms involved in logistic regression:

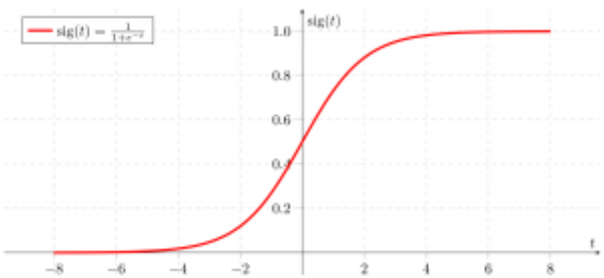
- **Independent variables:** The input characteristics or predictor factors applied to the dependent variable's predictions.
- **Dependent variable:** The target variable in a logistic regression model, which we are trying to predict.
- **Logistic function:** The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.
- **Odds:** It is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur.
- **Log-odds:** The log-odds, also known as the logit function, is the natural logarithm of the odds. In logistic regression, the log odds of the dependent variable are modeled as a linear combination of the independent variables and the intercept.
- **Coefficient:** The logistic regression model's estimated parameters, show how the independent and dependent variables relate to one another.
- **Intercept:** A constant term in the logistic regression model, which represents the log odds when all independent variables are equal to zero.
- **Maximum likelihood estimation:** The method used to estimate the coefficients of the logistic regression model, which maximizes the likelihood of observing the data given the model.

How does Logistic Regression work?

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

Sigmoid Function

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.



Sigmoid function

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

- tends towards 1 as
- tends towards 0 as
- is always bounded between 0 and 1

where the probability of being a class can be measured as:

Logistic Regression Equation

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

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

then the final logistic regression equation will be:

Likelihood Function for Logistic Regression

The predicted probabilities will be:

- for y=1 The predicted probabilities will be: p(X;b,w) = p(x)
- for y = 0 The predicted probabilities will be: 1-p(X;b,w) = 1-p(x)

Taking natural logs on both sides

Gradient of the log-likelihood function

To find the maximum likelihood estimates, we differentiate w.r.t w,

Code Implementation for Logistic Regression

Binomial Logistic regression:

Target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fail”, “dead” vs “alive”, etc., in this case, sigmoid functions are used, which is already discussed above.

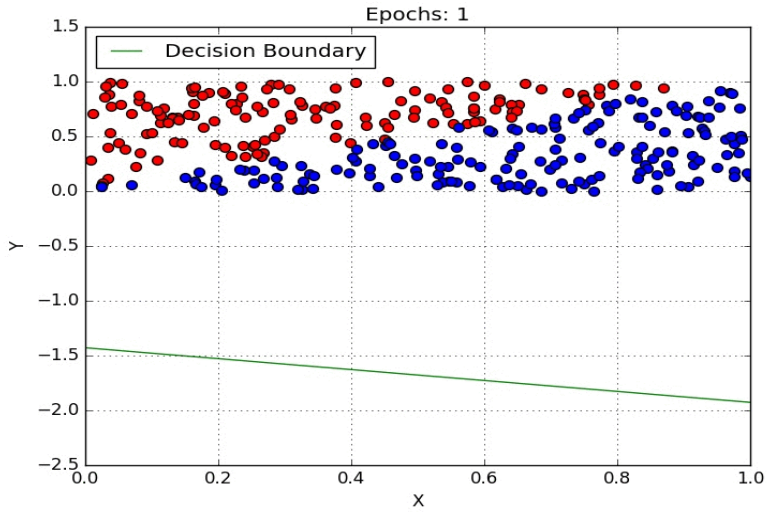
Importing necessary libraries based on the requirement of model. This Python code shows how to use the breast cancer dataset to implement a Logistic Regression model for classification.

---

2. What is Hypothesis representation in logistic regression?

Introduction

In this blog, we will discuss the basic concepts of Logistic Regression and what kind of problems can it help us to solve.



GIF: University of Toronto

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

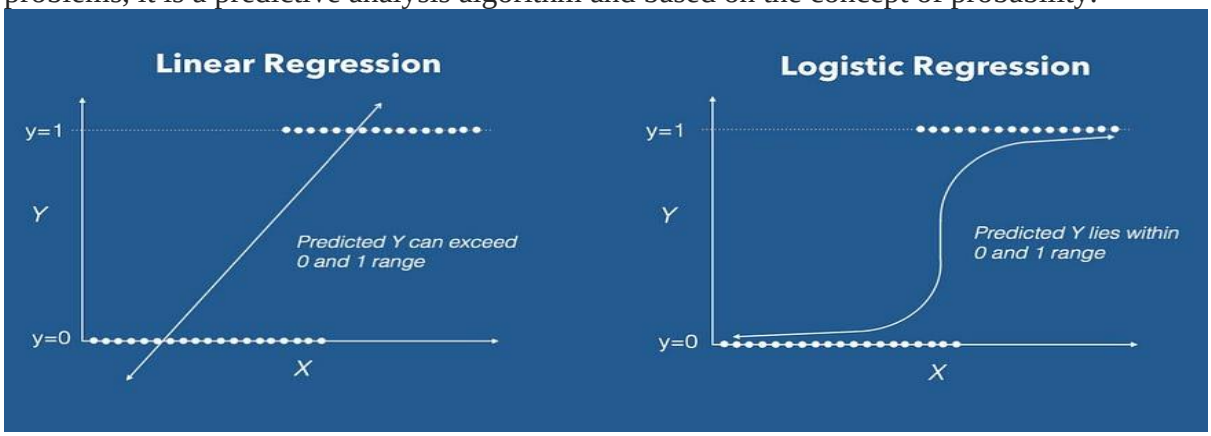
What are the types of logistic regression

1. Binary (eg. Tumor Malignant or Benign)

2. Multi-linear functions failsClass (eg. Cats, dogs or Sheep's)

Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



Linear Regression VS Logistic Regression Graph| Image: Data Camp

We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.

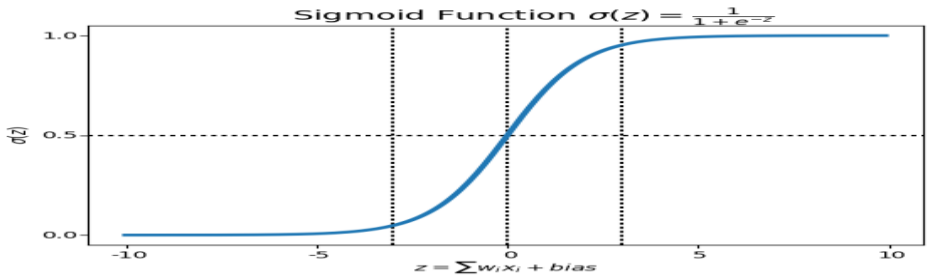
The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

Logistic regression hypothesis expectation

What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



Sigmoid Function Graph

$$f(x) = \frac{1}{1+e^{-(x)}}$$

Formula of a sigmoid function | Image: Analytics India Magazine

Hypothesis Representation

When using *linear regression* we used a formula of the hypothesis i.e.

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

For logistic regression we are going to modify it a little bit i.e.

$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$

We have expected that our hypothesis will give values between 0 and 1.

$Z = \beta_0 + \beta_1 X$

$h\Theta(x) = sigmoid(Z)$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

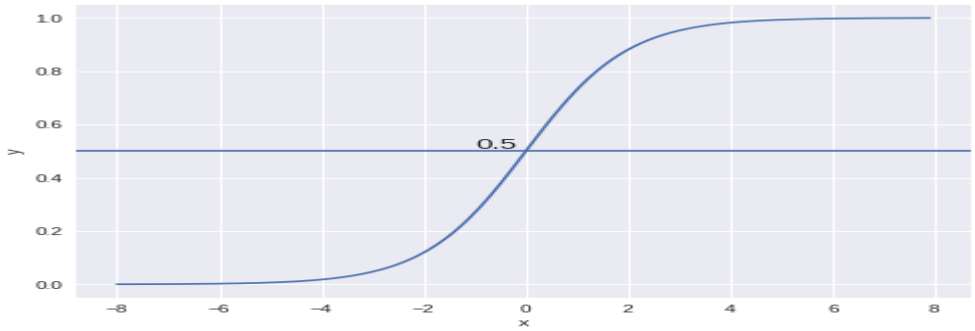
$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The Hypothesis of logistic regression

Decision Boundary

We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

For Example, We have 2 classes, let's take them like cats and dogs(1 — dog , 0 — cats). We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.



Example

As shown in the above graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1(DOG). If our prediction returned a value of 0.2 then we would classify the observation as Class 2(CAT).

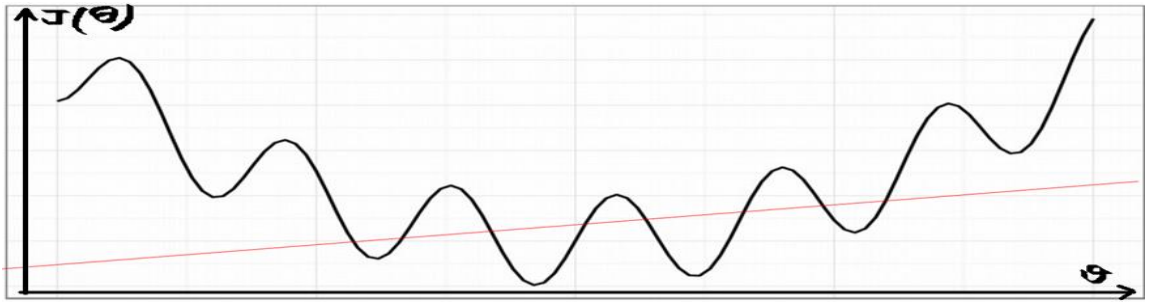
Cost Function

We learnt about the cost function $J(\theta)$ in the *Linear regression*, the cost function represents optimization objective i.e. we create a cost function and minimize it so that we can develop an accurate model with minimum error.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

The Cost function of Linear regression

If we try to use the cost function of the linear regression in 'Logistic Regression' then it would be of no use as it would end up being a **non-convex** function with many local minimums, in which it would be very **difficult** to **minimize the cost value** and find the global minimum.



Non-convex function

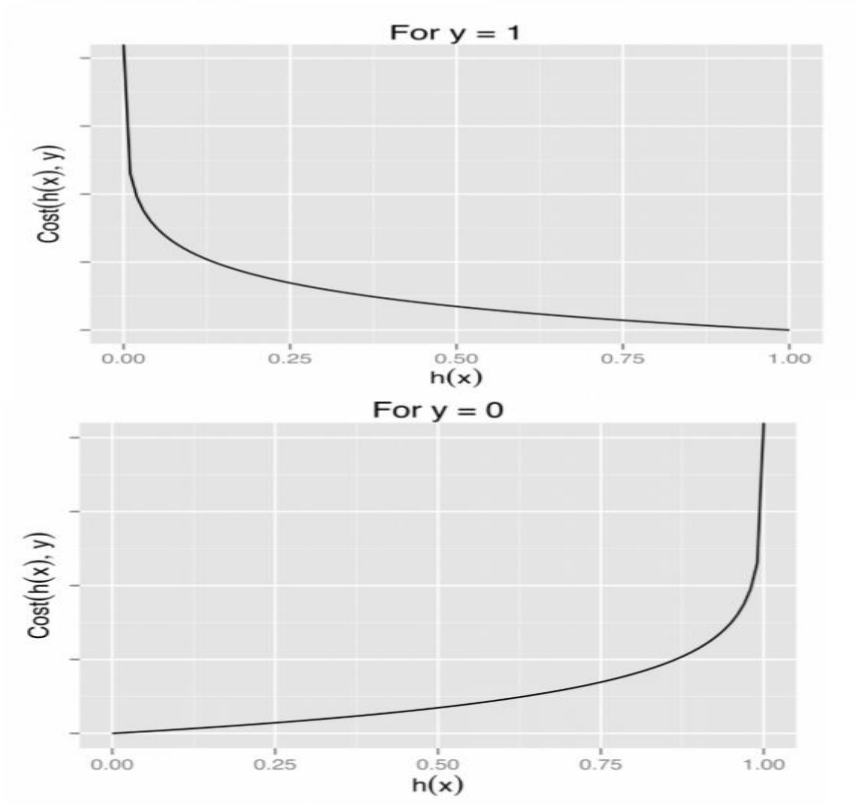
For logistic regression, the Cost function is defined as:

$-log(h\theta(x))$ *if* $y = 1$

$-log(1-h\theta(x))$ *if* $y = 0$

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Cost function of Logistic Regression



Graph of logistic regression

The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

Above functions compressed into one cost function

Gradient Descent

Now the question arises, how do we reduce the cost value. Well, this can be done by using **Gradient Descent.** The main goal of Gradient descent is to **minimize the cost value.** i.e. min J(θ).

Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

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

Objective: To minimize the cost function we have to run the gradient descent function on each parameter

Want $\min_\theta J(\theta)$:

Repeat {

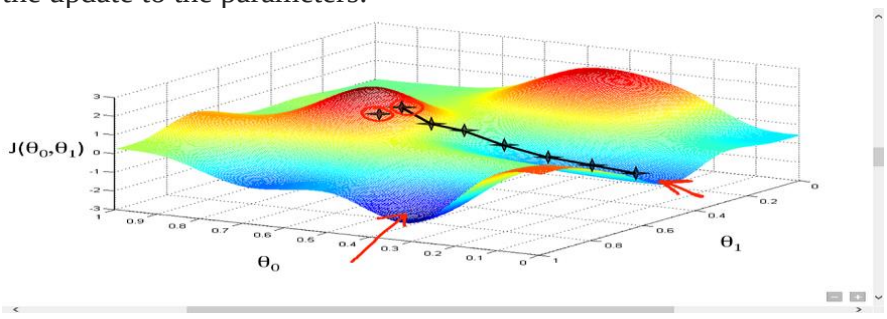
$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

}

Gradient Descent Simplified | Image: Andrew Ng Course

Gradient descent has an analogy in which we have to imagine ourselves at the top of a mountain valley and left stranded and blindfolded, our objective is to reach the bottom of the hill. Feeling the slope of the terrain around you is what everyone would do. Well, this action is analogous to calculating the gradient descent, and taking a step is analogous to one iteration of the update to the parameters.



Gradient Descent analogy

Unit 4 1.

What is decision tree? State the advantages, and limitations.

1. What is a decision tree?

In its simplest form, a decision tree is a type of flowchart that shows a clear pathway to a decision. In terms of data analytics, it is a type of algorithm that includes conditional 'control' statements to classify data. A decision tree starts at a single point (or 'node') which then branches (or 'splits') in two or more directions. Each branch offers different possible outcomes, incorporating a variety of decisions and chance events until a final outcome is achieved. When shown visually, their appearance is tree-like…hence the name!

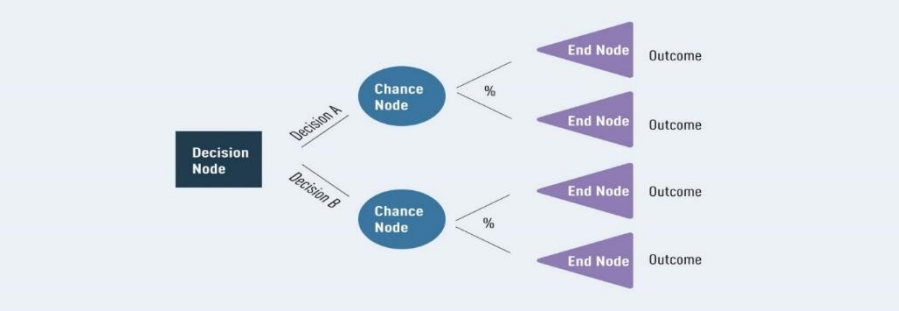
Decision trees are extremely useful for data analytics and machine learning because they break down complex data into more manageable parts. They're often used in these fields for prediction analysis, data classification, and regression. Don't worry if this all sounds a bit abstract—we'll provide some examples below to help clear things up. First though, let's look at the different aspects that make up a decision tree.

2. What are the different parts of a decision tree?

Decision trees can deal with complex data, which is part of what makes them useful. However, this doesn't mean that they are difficult to understand. At their core, all decision trees ultimately consist of just three key parts, or 'nodes':

- Decision nodes: Representing a decision (typically shown with a square)
- Chance nodes: Representing probability or uncertainty (typically denoted by a circle)
- End nodes: Representing an outcome (typically shown with a triangle)

Connecting these different nodes are what we call 'branches'. Nodes and branches can be used over and over again in any number of combinations to create trees of various complexity. Let's see how these parts look before we add any data.



Luckily, a lot of decision tree terminology follows the tree analogy, which makes it much easier to remember! Some other terms you might come across will include:

Root nodes

In the diagram above, the blue decision node is what we call a 'root node.' This is always the first node in the path. It is the node from which all other decision, chance, and end nodes eventually branch.

Leaf nodes

In the diagram above, the lilac end nodes are what we call 'leaf nodes.' These show the end of a decision path (or outcome). You can always identify a leaf node because it doesn't split, or branch any further. Just like a real leaf!

Internal nodes

Between the root node and the leaf nodes, we can have any number of internal nodes. These can include decisions and chance nodes (for simplicity, this diagram only uses chance nodes). It's easy to identify an internal node—each one has branches of its own while also connecting to a previous node.

Splitting

Branching or 'splitting' is what we call it when any node divides into two or more sub-nodes. These sub-nodes can be another internal node, or they can lead to an outcome (a leaf/ end node.)

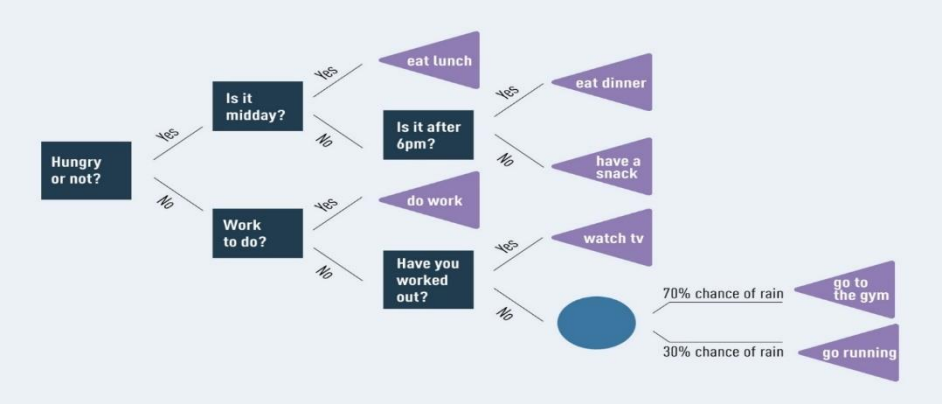
Pruning

Sometimes decision trees can grow quite complex. In these cases, they can end up giving too much weight to irrelevant data. To avoid this problem, we can remove certain nodes using a process known as 'pruning'. Pruning is exactly what it sounds like—if the tree grows branches we don't need, we simply cut them off. Easy!

Curious about a career in Data Analytics?
Start learning for free!

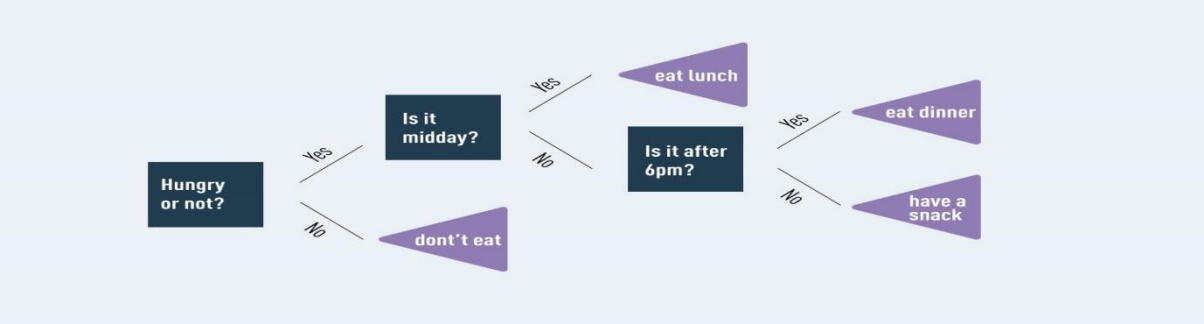
3. An example of a simple decision tree

Now that we've covered the basics, let's see how a decision tree might look. We'll keep it really simple. Let's say that we're trying to classify what options are available to us if we are hungry. We might show this as follows:



In this diagram, our different options are laid out in a clear, visual way. Decision nodes are navy blue, chance nodes are light blue, and end nodes are purple. It is easy for anybody to understand and to see the possible outcomes.

However, let's not forget: our aim was to classify what to do in the event of being hungry. By including options for what to do in the event of not being hungry, we've overcomplicated our decision tree. Cluttering a tree in this way is a common problem, especially when dealing with large amounts of data. It often results in the algorithm extracting meaning from irrelevant information. This is known as overfitting. One option to fix overfitting is simply to prune the tree:



As you can see, the focus of our decision tree is now much clearer. By removing the irrelevant information (i.e. what to do if we're not hungry) our outcomes are focused on the goal we're aiming for. This is one example of a pitfall that decision trees can fall into, and how to get around it. However, there are several pros and cons for decision trees. Let's touch on these next.

4. Pros and cons of decision trees

Used effectively, decision trees are very powerful tools. Nevertheless, like any algorithm, they're not suited to every situation. Here are some key advantages and disadvantages of decision trees.

Advantages of decision trees

- Good for interpreting data in a highly visual way.
- Good for handling a combination of numerical and non-numerical data.
- Easy to define rules, e.g. 'yes, no, if, then, else…'
- Requires minimal preparation or **data cleaning** before use.
- Great way to choose between best, worst, and likely case scenarios.
- Can be easily combined with other decision-making techniques.

Disadvantages of decision trees

- Overfitting (where a model interprets meaning from irrelevant data) can become a problem if a decision tree's design is too complex.
- They are not well-suited to continuous variables (i.e. variables which can have more than one value, or a spectrum of values).
- In predictive analysis, calculations can quickly grow cumbersome, especially when a decision path includes many chance variables.
- When using an imbalanced dataset (i.e. where one class of data dominates over another) it is easy for outcomes to be biased in favor of the dominant class.
- Generally, decision trees provide lower prediction accuracy compared to other predictive algorithms.

5. What are decision trees used for?

Despite their drawbacks, decision trees are still a powerful and popular tool. They're commonly used by data analysts to carry out predictive analysis (e.g. to develop operations strategies in businesses). They're also a popular tool for machine learning and artificial intelligence, where they're used as training algorithms for supervised learning (i.e. categorizing data based on different tests, such as 'yes' or 'no' classifiers.)

Broadly, decision trees are used in a wide range of industries, to solve many types of problems. Because of their flexibility, they're used in sectors from technology and health to financial planning. Examples include:

- A technology business evaluating expansion opportunities based on analysis of past sales data.
- A toy company deciding where to target its limited advertising budget, based on what demographic data suggests customers are likely to buy.
- Banks and mortgage providers using historical data to predict how likely it is that a borrower will default on their payments.
- Emergency room triage might use decision trees to prioritize patient care (based on factors such as age, gender, symptoms, etc.)
- Automated telephone systems guiding you to the outcome you need, e.g. 'For option A, press 1. For option B, press 2', and so on.

---

2. What is the need of decision tree?

he need for decision trees in machine learning arises due to several factors:

1. **Interpretability**: Decision trees are easily interpretable and understandable by humans. They represent decisions in a tree-like structure where each internal node denotes a decision based on a feature, each branch represents the outcome of that decision, and each leaf node represents the final decision or outcome.
2. **Versatility**: Decision trees can be applied to both classification and regression problems. They are versatile and can handle both numerical and categorical data.
3. **Handling Non-Linearity**: Decision trees can capture complex nonlinear relationships between features and the target variable without the need for feature engineering or transformation.
4. **Handling Missing Values**: Decision trees can handle missing values in the dataset without requiring imputation techniques. They can make decisions based on available features at each node.
5. **Feature Selection**: Decision trees implicitly perform feature selection by identifying the most informative features at each decision point. This can help in reducing overfitting and improving model performance.
6. **Ease of Implementation**: Decision trees are relatively easy to implement and understand compared to other machine learning algorithms. They provide a simple and intuitive way to model decision-making processes.

Overall, decision trees serve as a powerful tool in machine learning due to their interpretability, versatility, ability to handle nonlinear relationships, and ease of implementation

---

3. Explain decision tress algorithm.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

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

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

Why use Decision Trees?

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

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

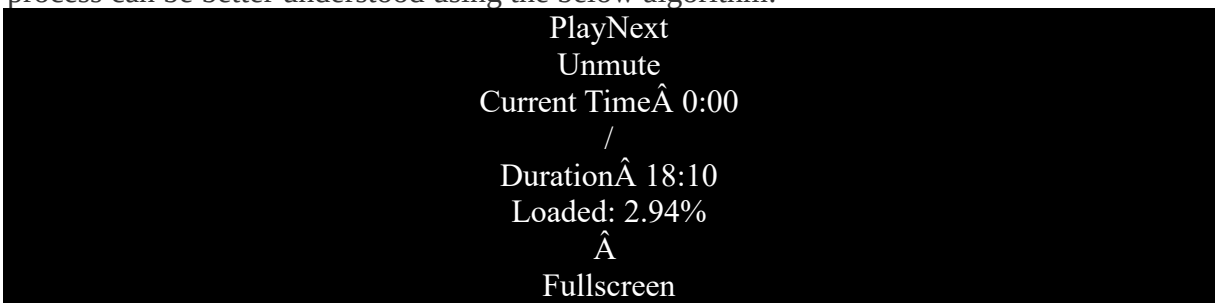
Decision Tree Terminologies

 Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
 Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
 Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
 Branch/Sub Tree: A tree formed by splitting the tree.
 Pruning: Pruning is the process of removing the unwanted branches from the tree.
 Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

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

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



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

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

Attribute Selection Measures

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

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

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

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

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

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

2. Gini Index:

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

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

Pruning: Getting an Optimal Decision tree

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

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

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

Advantages of the Decision Tree

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

Disadvantages of the Decision Tree

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

Python Implementation of Decision Tree

Now we will implement the Decision tree using Python. For this, we will use the dataset "**user_data.csv**," which we have used in previous classification models. By using the same dataset, we can compare the Decision tree classifier with other classification models such as KNN SVM, LogisticRegression, etc.

Steps will also remain the same, which are given below:

- **Data Pre-processing step**
- **Fitting a Decision-Tree algorithm to the Training set**
- **Predicting the test result**
- **Test accuracy of the result(Creation of Confusion matrix)**
- **Visualizing the test set result.**

---

3. **What is information gain and entropy in decision tree?**
4. Entropy and information gain are key concepts in domains such as information theory, data science, and machine learning. Information gain is the amount of knowledge acquired during a certain decision or action, whereas entropy is a measure of uncertainty or unpredictability. People can handle difficult situations and make wise judgments across a variety of disciplines when they have a solid understanding of these principles. Entropy can be used in data science, for instance, to assess the variety or unpredictable nature of a dataset, whereas Information Gain can assist in identifying the qualities that would be most useful to include in a model. In this article, we'll examine the main distinctions between entropy and information gain and how they affect machine learning.
5. What is Entropy?
6. The term "entropy" comes from the study of thermodynamics, and it describes how chaotic or unpredictable a system is. Entropy is a measurement of a data set's impurity in the context of machine learning. In essence, it is a method of calculating the degree of uncertainty in a given dataset.
7. The following formula is used to compute entropy −
8. Entropy(S)=−p1log2p1−p2log2p2−…−pnlog2pnEntropy(S)=−p1log2p1−p2log2p2−…−pnlog2pn
9. S is the data set, and p1 through pn are the percentages of various classes inside the data. The resultant entropy value is expressed in bits since the base 2 logarithm used in this method is typical.
10. Consider a dataset with two classes, A and B, in order to comprehend this formula. The entropy can be determined as follows if 80% of the data is in class A and 20% is in class B −
11. Entropy(S)=−0.8log20.8−0.2log20.2=0.72bitsEntropy(S)=−0.8log20.8−0.2log20.2=0.72bits
12. This indicates that the dataset is impurity-rich, with an entropy of 0.72 bits.
13. What is Information Gain?
14. Information Gain is a statistical metric used to assess a feature's applicability in a dataset. It is an important idea in machine learning and is frequently utilized in decision tree algorithms. By contrasting the dataset's entropy before and after a feature is separated, information gain is estimated. A feature's relevance to the categorization of the data increases with information gain.
15. When the dataset has been divided based on a feature, information gain calculates the entropy decrease. The amount of knowledge a feature imparts about the class is

measured by this metric. Selecting the characteristic that provides the most information about the class will help you achieve your aim of maximizing information gain.

16. The following formula is used to compute information gain −

17. InformationGain(S,A)=Entropy(S)−∑(|Sv|/|S|)∗Entropy(Sv)InformationGain(S,A)=Entropy(S)−∑(|Sv|/|S|)∗Entropy(Sv)

18. The number of elements in Sv is given by |Sv|, where S is the set of data, A is a feature, Sv is the subset of S for which feature A takes the value v, and S is the total number of elements in S.
19. Think of a dataset with two characteristics, X and Y, to better comprehend this formula. The information gain can be calculated as follows if the data is to be divided based on characteristic X −

20. InformationGain(S,X)=Entropy(S)−[(3/5)∗Entropy(S1)+(2/5)∗Entropy(S2)]InformationGain(S,X)=Entropy(S)−[(3/5)∗Entropy(S1)+(2/5)∗Entropy(S2)]

21. where S1 is the subset of data where feature X takes a value of 0, and S2 is the subset of data where feature X takes a value of 1. These two subsets' entropies, Entropy(S1) and Entropy(S2), can be determined using the formula we previously covered.
22. The amount by which the dataset will be divided based on characteristic X will be shown by the information gain that results.
23. Key Differences between Entropy and Information Gain

Entropy	**Information Gain**
Entropy is a measurement of the disorder or impurity of a set of occurrences. It determines the usual amount of information needed to classify a sample taken from the collection.	Information gain is a metric for the entropy reduction brought about by segmenting a set of instances according to a feature. It gauges the amount of knowledge a characteristic imparts to the class of an example.
Entropy is calculated for a set of examples by calculating the probability of each class in the set and using that information in the entropy calculation.	By dividing the collection of instances depending on the feature and calculating the entropies of the resulting subsets, information gain is determined for each feature. The difference between the entropy of the original set and the weighted sum of the entropies of the subsets is thus the information gain.
Entropy quantifies the disorder or impurity present in a collection of instances and aims to be minimized by identifying the ideal division.	By choosing the feature with the maximum information gain, the objective of information gain is to maximize the utility of a feature for categorization.
Entropy is typically taken into account by decision trees for determining the best split.	Decision trees frequently employ information gain as a criterion for choosing the optimal feature to split on.
Entropy usually favors splits that result in balanced subgroups.	Splits that produce imbalanced subsets with pure classes are frequently preferred by information gain.

Entropy can control continuous characteristics by discretizing them into bins.	By choosing the split point that maximizes the information acquisition, continuous features may also be handled.
Calculating probabilities and logarithms, which can be computationally costly, is necessary to determine entropy.	Entropies and weighted averages must be calculated in order to gather information, which can be computationally costly.
Entropy is a versatile indicator of impurity that may be applied to a variety of classification issues.	For binary classification issues, information gain is a particular measure of feature usefulness that works well.
Entropy, which is given in bits, calculates the typical amount of data required to categorize an example.	Information gain, which is also stated in bits, indicates the reduction in uncertainty attained by splitting based on a feature.
If there are too many characteristics or the tree is too deep, entropy might result in overfitting.	If the tree is too deep or there are too many irrelevant characteristics, information gain may potentially result in overfitting.

---

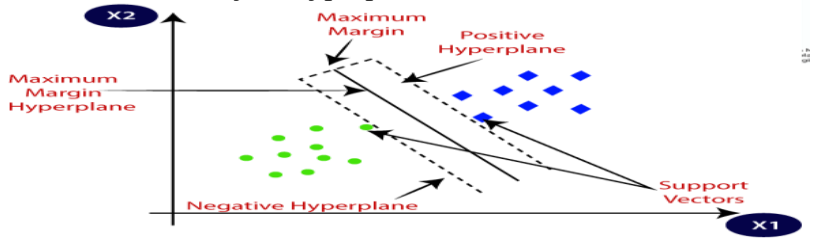
6. What is SVM? Explain in detail.

Support Vector Machine Algorithm

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

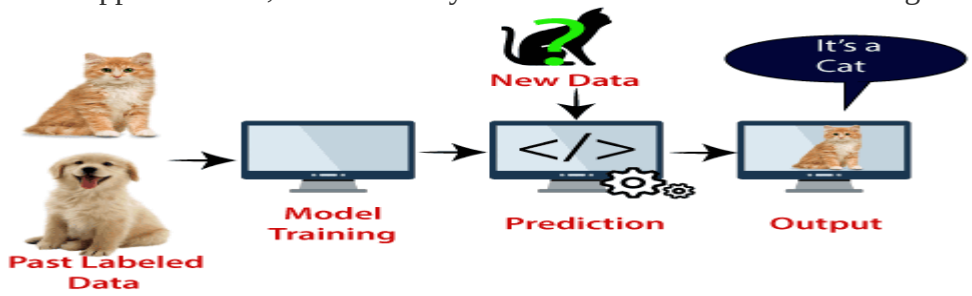
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

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



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

choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



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

Types of SVM

SVM can be of two types:

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

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

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

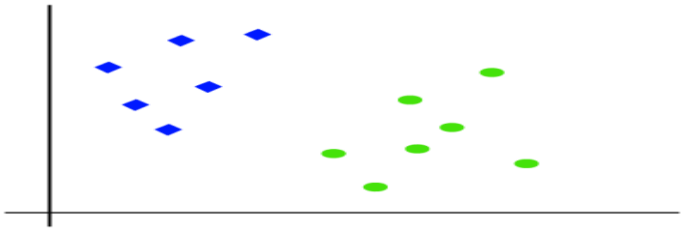
Support Vectors:

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

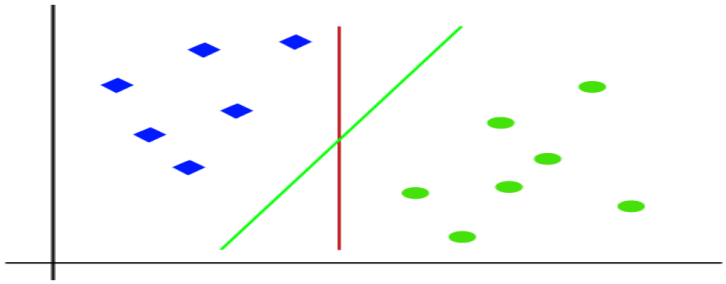
How does SVM works?

Linear SVM:

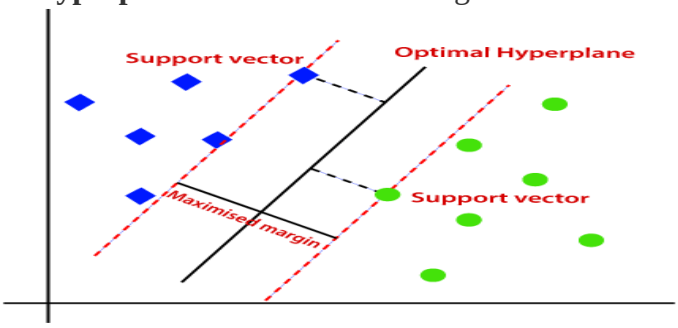
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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

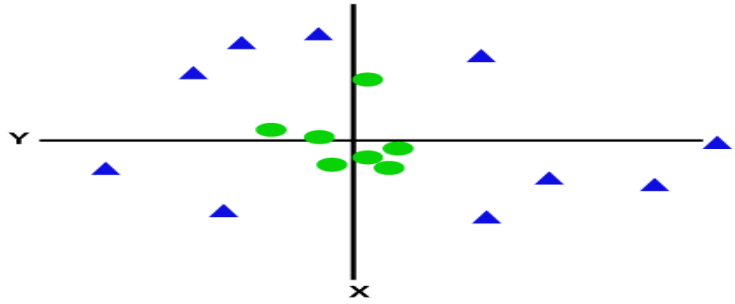


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane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



Non-Linear SVM:

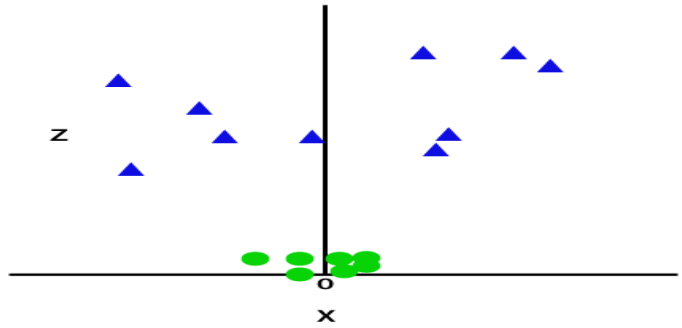
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



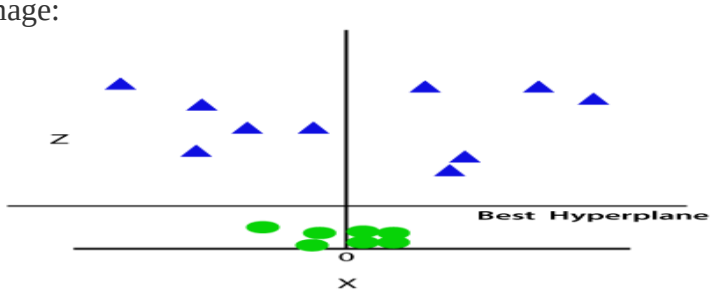
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

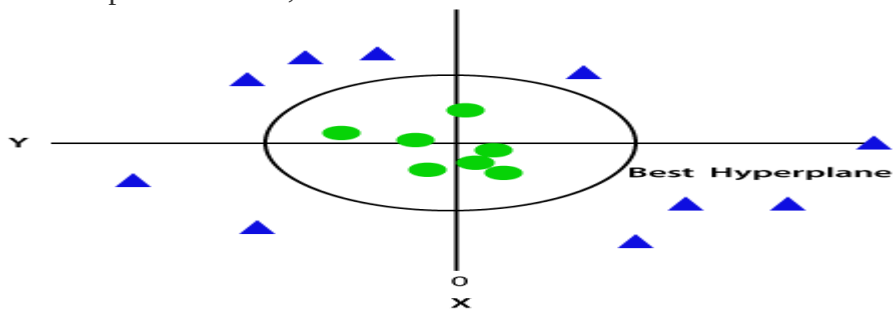
By adding the third dimension, the sample space will become as below image:



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



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



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

---

7. Explain Hyperplane and Support Vectors in the SVM algorithm

What is SVM?

In the SVM algorithm, we plot each observation as a point in an n-dimensional space (where n is the number of features in the dataset). Our task is to find an optimal hyperplane that successfully classifies the data points into their respective classes.

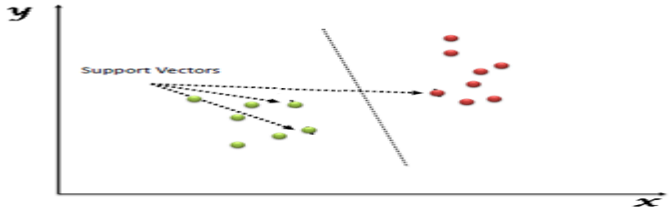
Before diving into the working of SVM let's first understand the two basic terms used in the algorithm "The support vector " and " Hyper-Plane".

Hyper-Plane

A hyperplane is a decision boundary that differentiates the two classes in SVM. A data point falling on either side of the hyperplane can be attributed to different classes. The dimension of the hyperplane depends on the number of input features in the dataset. If we have 2 input features the hyper-plane will be a line. likewise, if the number of features is 3, it will become a two-dimensional plane.

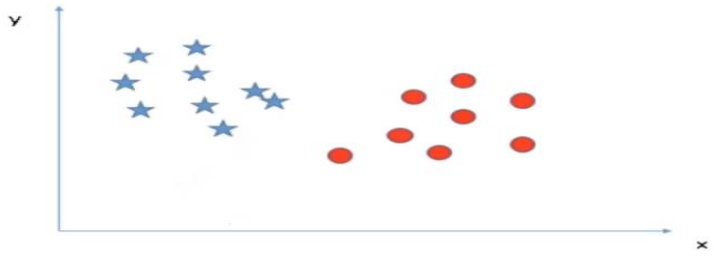
Support-Vectors

Support vectors are the data points that are nearest to the hyper-plane and affect the position and orientation of the hyper-plane. We have to select a hyperplane, for which the margin, i.e the distance between support vectors and hyper-plane is maximum. Even a little interference in the position of these support vectors can change the hyper-plane.

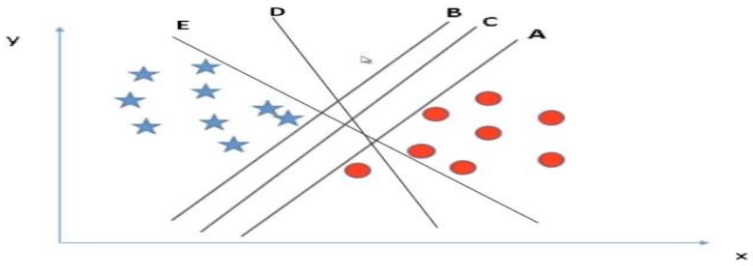


How SVM works?

As we have a clear idea of the terminologies related to SVM, let's now see how the algorithm works. For example, we have a classification problem where we have to separate the red data points from the blue ones.



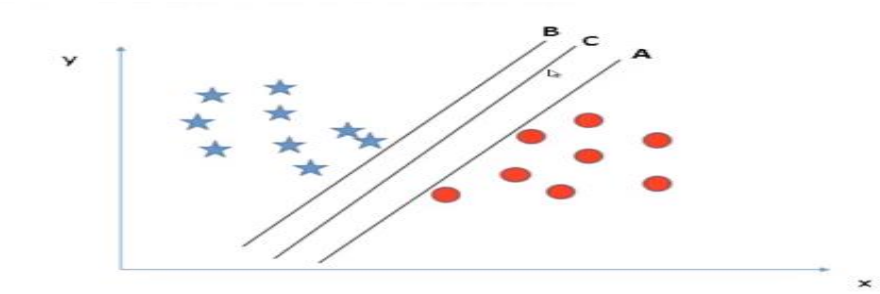
Since it is a two-dimensional problem, our decision boundary will be a line, for the 3-dimensional problem we have to use a plane, and similarly, the complexity of the solution will increase with the rising number of features.



As shown in the above image, we have multiple lines separating the data points successfully. But our objective is to look for the best solution.

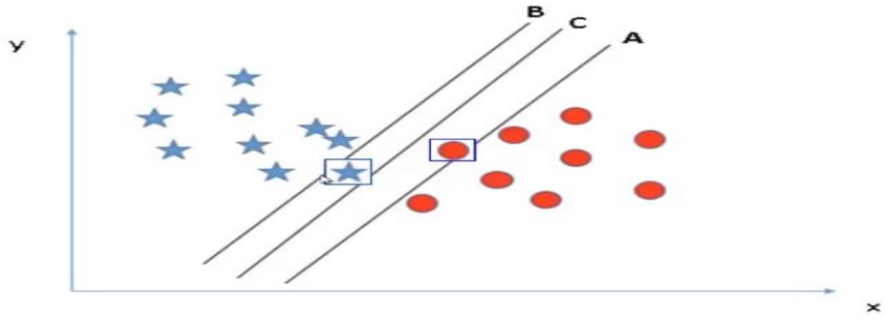
There are few rules that can help us to identify the best line.

Maximum classification, i.e the selected line must be able to successfully segregate all the data points into the respective classes. In our example, we can clearly see lines E and D are miss classifying a red data point. Hence, for this problem lines A, B, C is better than E and D. So we will drop them.



The second rule is **Best Separation**, which means, we must choose a line such that it is perfectly able to separate the points.

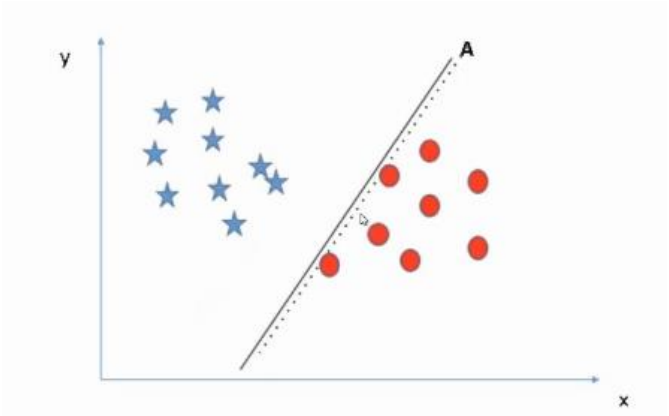
If we talk about our example, if we get a new red data point closer to line A as shown in the image below, line A will miss classifying that point. Similarly, if we got a new blue instance closer to line B, then line A and C will classify the data successfully, whereas line B will miss classifying this new point.

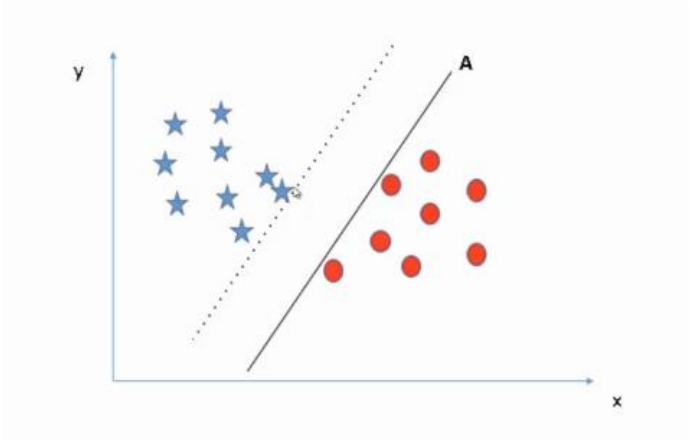


The point to be noticed here, In both the cases line C is successfully classifying all the data points why? To understand this let's take all the lines one by one.

Why not Line A?

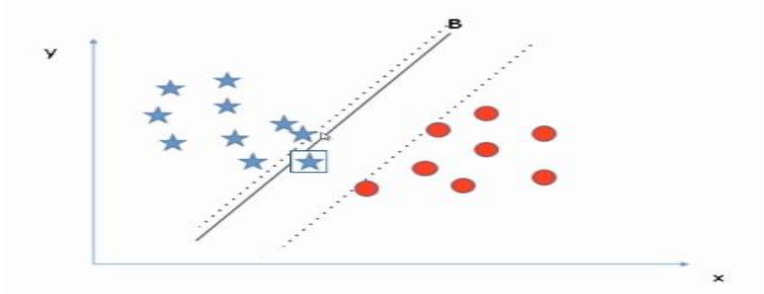
First, consider line A. If I move line A towards the left, we can see it has very little chance to miss classify the blue points. on the other hand, if I shift line A towards the right side it will very easily miss-classify the red points. The reason is on the left side of the margin i.e the distance between the nearest data point and the line is large whereas on the right side the margin is very low.





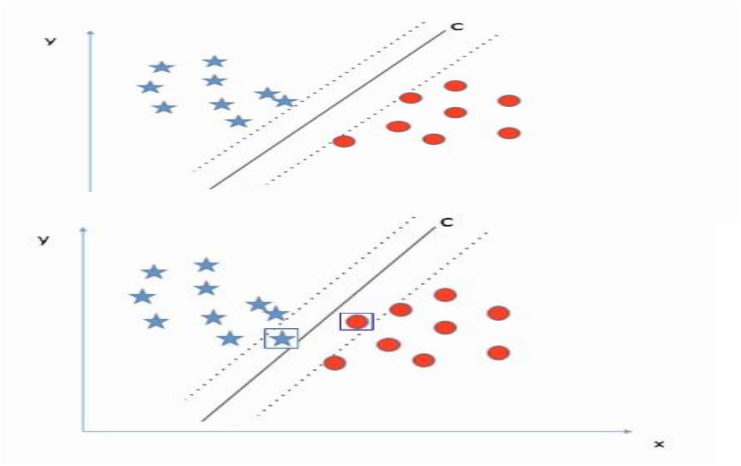
Why Not Line B?

Similarly, in the case of line B. If we shift line B towards the right side, it has a sufficient margin on the right side whereas it will wrongly classify the instances on the left side as the margin towards the left is very low. Hence, B is not our perfect solution.



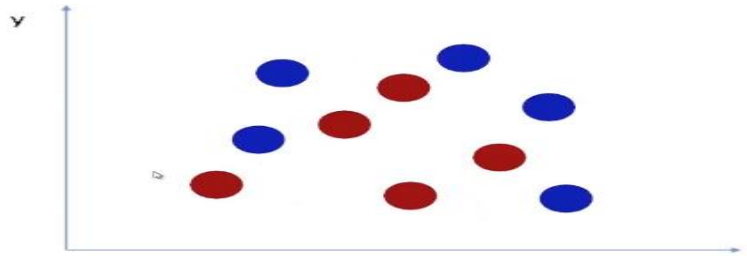
Why Line C?

In the case of line C, It has sufficient margin on the left as well as the right side. This maximum margin makes line C more robust for the new data points that might appear in the future. Hence, C is the best fit in that case that successfully classifies all the data points with the maximum margin.

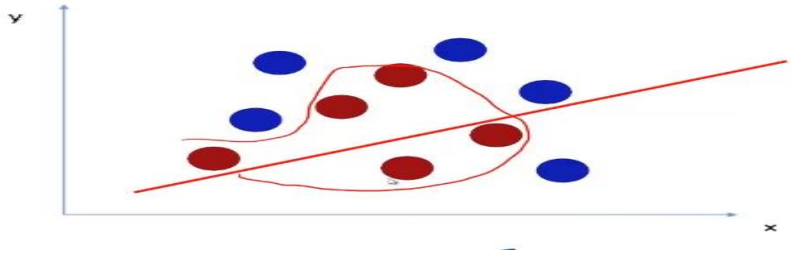


This is what SVM looks for, it aims for the maximum margin and creates a line that is equidistant from both sides, which is line C in our case. so we can say C represents the SVM classifier with the maximum margin.

Now let's look at the data below, As we can see this is not linearly separable data, so SVM will not work in this situation. If anyhow we try to classify this data with a line, the result will not be promising.

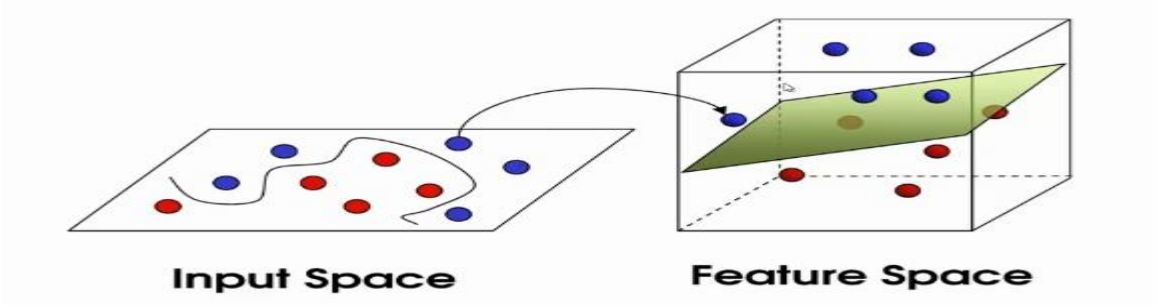


So, is there any way that SVM can classify this kind of data? For this problem, we have to create a decision boundary that looks something like this.



The question is, is it possible to create such a decision boundary using SVM. Well, the answer is **Yes**. SVM does this by projecting the data in a higher dimension. As shown in the following image. In the first case, data is not linearly separable, hence, we project into a higher dimension.

If we have more complex data then SVM will continue to project the data in a higher dimension till it becomes linearly separable. Once the data become linearly separable, we can use SVM to classify just like the previous problems.



8. **Which are the Pros and Cons of SVM Classifiers?**

Pros and Cons of SVM

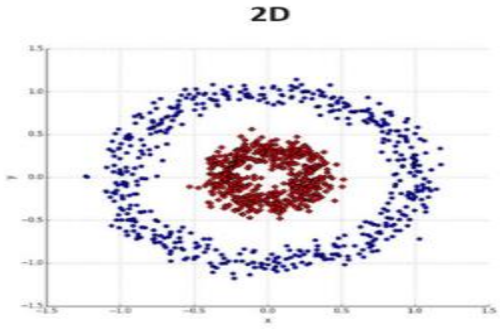
Pros:

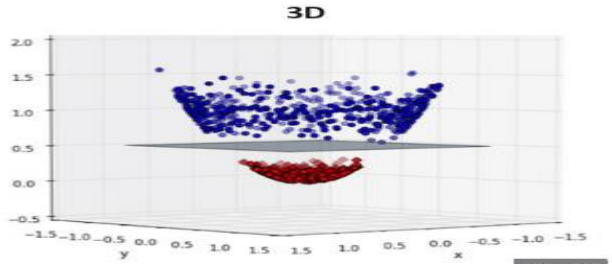
- It works really well with a clear margin of separation.
- It is effective in high-dimensional spaces.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It uses a subset of the training set in the decision function (called support vectors), so it is also memory efficient.

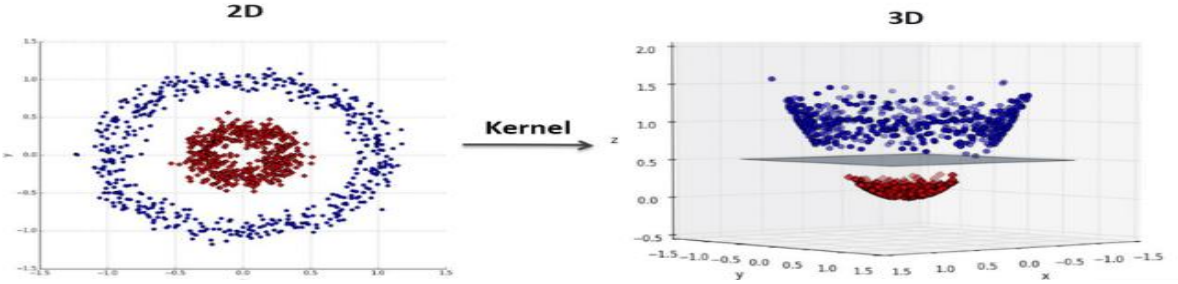
Cons:

- It doesn't perform well when we have a large data set because the required training time is higher.
- It also doesn't perform very well when the data set has more noise, i.e., target classes are overlapping.
- SVM doesn't directly provide probability estimates; these are calculated using an expensive five-fold cross-validation. It is included in the related SVC method of the Python scikit-learn library.

9. **What is kernel trick in SVM?**
10. Support Vector Machines (SVM) is a popular and effective machine learning algorithm used in classification and regression tasks. It works by finding the best possible boundary that can separate two classes of data points with maximum margin, also known as the hyperplane. SVM utilizes kernel functions to map the input data points into a higher-dimensional space where the separation between the two classes becomes easier. This allows SVM to solve complex non-linear problems as well. SVM has been shown to be very effective in many real-world applications such as text classification, image classification, and bioinformatics. Its ability to handle high-dimensional feature spaces, excellent generalization performance, and robustness to noisy data make it one of the most preferred algorithms in the field of machine learning.
11. **What is Kernel Trick ?**
12. The "Kernel Trick" is a method used in Support Vector Machines (SVMs) to convert data (that is not linearly separable) into a higher-dimensional feature space where it may be linearly separated.

13. This technique enables the SVM to identify a hyperplane that separates the data with the maximum margin, even when the data is not linearly separable in its original space. The kernel functions are used to compute the inner product between pairs of points in the transformed feature space without explicitly computing the transformation itself. This makes it computationally efficient to deal with high dimensional feature spaces.
14. The most widely used kernels in SVM are the linear kernel, polynomial kernel, and Gaussian (radial basis function) kernel. The choice of kernel relies on the nature of the data and the job at hand. The linear kernel is used when the data is roughly linearly separable, whereas the polynomial kernel is used when the data has a complicated curved border. The Gaussian kernel is employed when the data has no clear boundaries and contains complicated areas of overlap.
15. Let's take an example to understand the kernel trick in more detail. Consider a binary classification problem where we have two classes of data points: red and blue. The data is not linearly separable in the 2D space. We can see this in the plot below:
16. 

17. To make this data linearly separable, we can use the kernel trick.
18. By applying the kernel trick to the data, we transform it into a higher-dimensional feature space where the data becomes linearly separable. We can see this in the plot below, where the red and blue data points have been separated by a hyperplane in the 3D space:
19. 

20. 

21. As we can see, the kernel trick has helped us find a solution for a non-linearly separable dataset.
22. The kernel trick is a powerful technique that enables SVMs to solve non-linear classification problems by implicitly mapping the input data to a higher-dimensional feature space. By doing so, it allows us to find a hyperplane that separates the different classes of data

10. What is cost function of SVM?

11. The Cost Function

$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

12.

13. The Cost Function is used to train the SVM. By minimizing the value of J(theta), we can ensure that the SVM is as accurate as possible. In the equation, the functions cost1 and cost0 refer to the cost for an example where y=1 and the cost for an example where y=0. For SVMs, cost is determined by kernel (similarity) functions.

Unit 5

1. What is clustering? Explain in detail.

Clustering in Machine Learning

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as ***"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."***

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique is commonly used for **statistical data analysis.**

Note: Clustering is somewhere similar to the classification algorithm, but the difference is the type of dataset that we are using. In classification, we work with the labeled data set, whereas in clustering, we work with the unlabelled dataset.

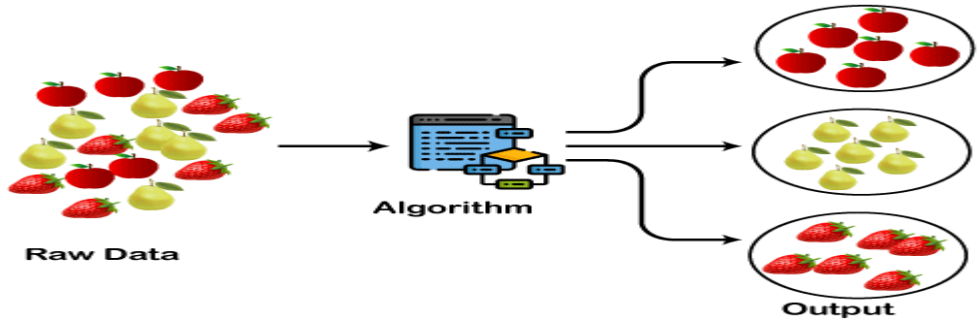
Example: Let's understand the clustering technique with the real-world example of Mall: When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation system to provide the recommendations as per the past search of products. **Netflix** also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods

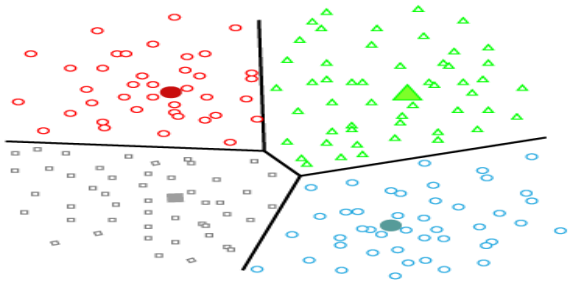
The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. **Partitioning Clustering**
2. **Density-Based Clustering**
3. **Distribution Model-Based Clustering**
4. **Hierarchical Clustering**
5. **Fuzzy Clustering**

Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.

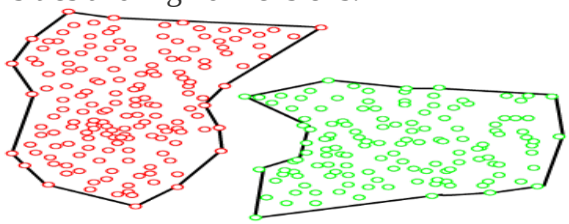


Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This

algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

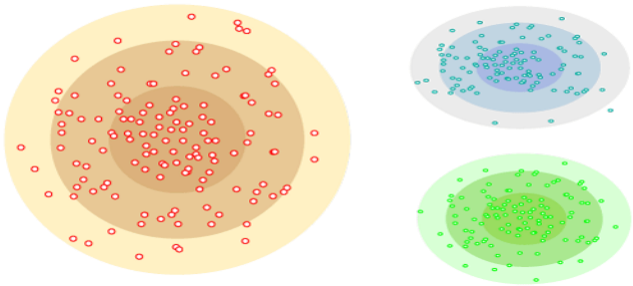
These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Distribution Model-Based Clustering

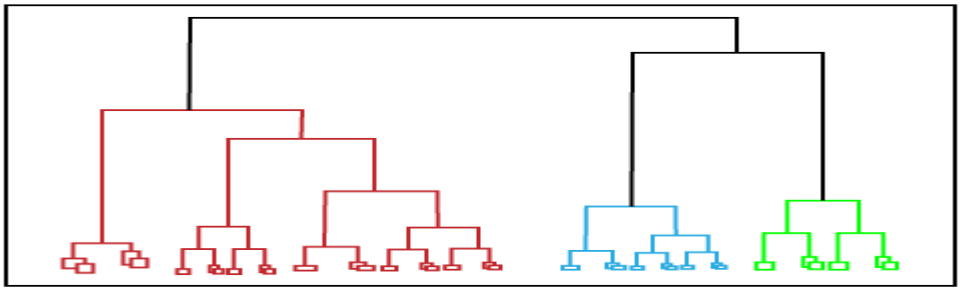
In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).



Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.



Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree

of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Clustering Algorithms

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. **K-Means algorithm:** The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of **O(n).**
2. **Mean-shift algorithm:** Mean-shift algorithm tries to find the dense areas in the smooth density of data points. It is an example of a centroid-based model, that works on updating the candidates for centroid to be the center of the points within a given region.
3. **DBSCAN Algorithm:** It stands **for Density-Based Spatial Clustering of Applications with Noise**. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.
4. **Expectation-Maximization Clustering using GMM:** This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.
5. **Agglomerative Hierarchical algorithm:** The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.
6. **Affinity Propagation:** It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has $O(N^2T)$ time complexity, which is the main drawback of this algorithm.

Applications of Clustering

Below are some commonly known applications of clustering technique in Machine Learning:

- **In Identification of Cancer Cells:** The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- **In Search Engines:** Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.
- **Customer Segmentation:** It is used in market research to segment the customers based on their choice and preferences.

- **In Biology:** It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- **In Land Use:** The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

---

2. Explain K Means clustering.

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled 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 allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

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.

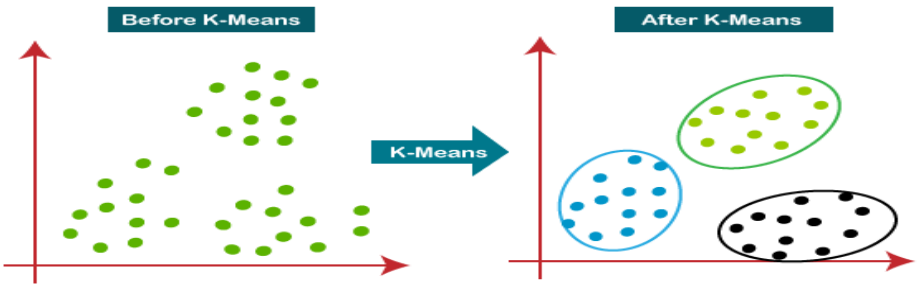
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

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

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

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



How does the K-Means Algorithm Work?

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

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

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

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

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

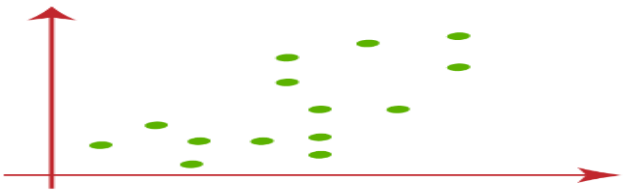
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

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

Step-7: The model is ready.

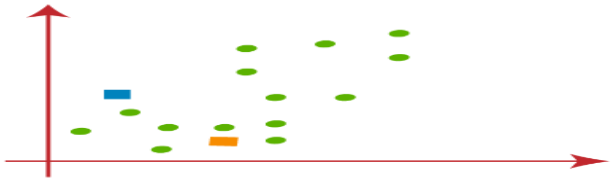
Let's understand the above steps by considering the visual plots:

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

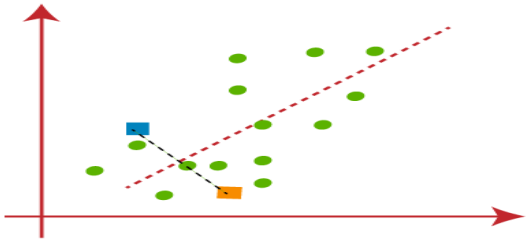


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points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

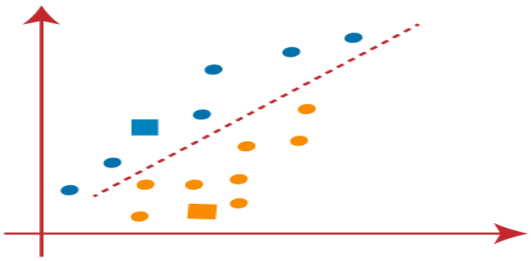
below image:



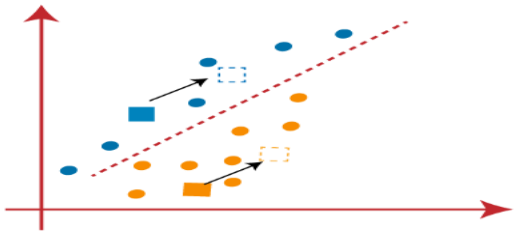
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



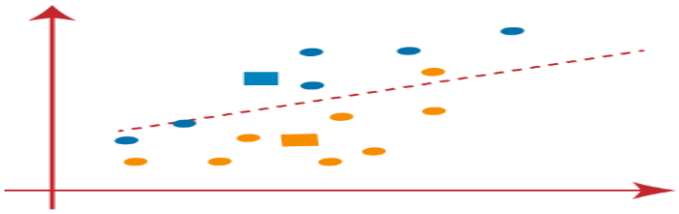
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



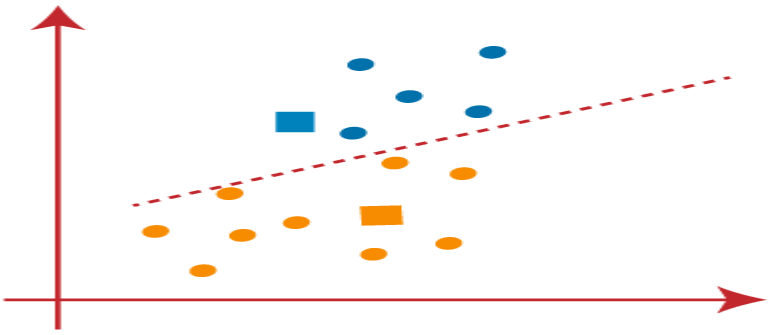
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



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

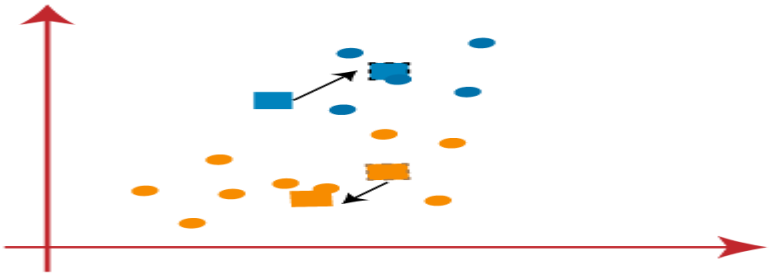


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

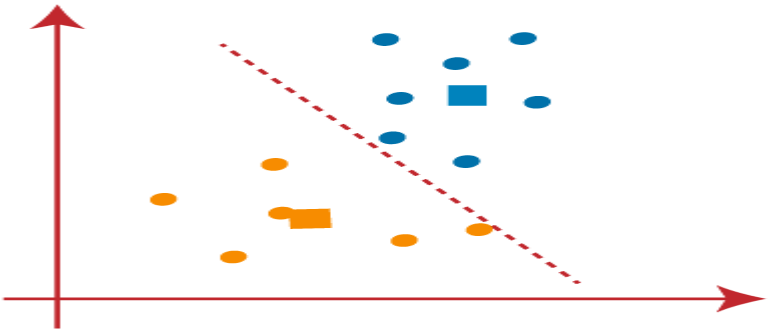


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

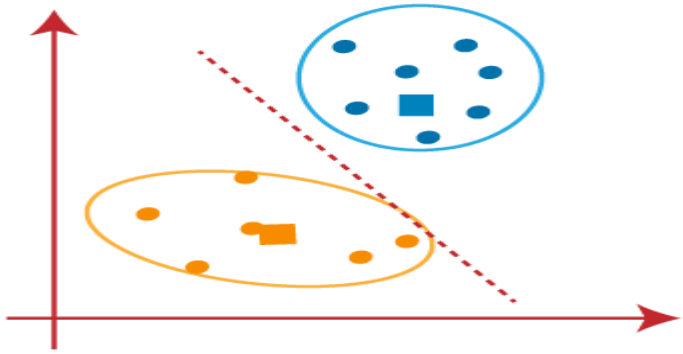
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



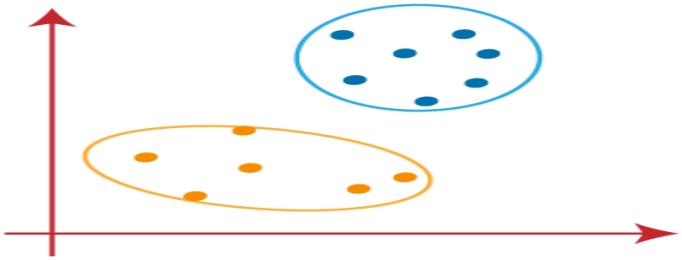
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



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



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



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

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

Elbow Method

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

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

In the above formula of WCSS,

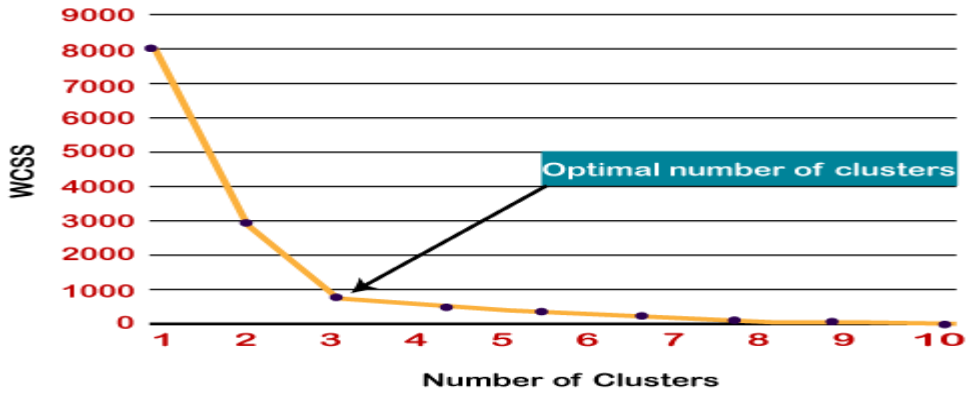
$\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

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

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

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

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



Note: We can choose the number of clusters equal to the given data points. If we choose the number of clusters equal to the data points, then the value of WCSS becomes zero, and that will be the endpoint of the plot.

Python Implementation of K-means Clustering Algorithm

In the above section, we have discussed the K-means algorithm, now let's see how it can be implemented using Python.

Before implementation, let's understand what type of problem we will solve here. So, we have a dataset of **Mall_Customers**, which is the data of customers who visit the mall and spend there.

In the given dataset, we have **Customer_Id, Gender, Age, Annual Income ($), and Spending Score** (which is the calculated value of how much a customer has spent in the mall, the more the value, the more he has spent). From this dataset, we need to calculate some patterns, as it is an unsupervised method, so we don't know what to calculate exactly.

The steps to be followed for the implementation are given below:

- **Data Pre-processing**
- **Finding the optimal number of clusters using the elbow method**
- **Training the K-means algorithm on the training dataset**
- **Visualizing the clusters**

---

3. Explain Elbow Method in K Means clustering

Introduction

Clustering is an unsupervised machine-learning technique. It is the process of division of the dataset into groups in which the members in the same group possess similarities in features. The commonly used clustering techniques are K-Means clustering, Hierarchical clustering, Density-based clustering, Model-based clustering, etc. It can even handle large datasets. We can implement the K-Means clustering machine learning algorithm in the elbow method using the scikit-learn library in Python.

Elbow Method Definition

The Elbow Method is a technique used in data analysis and machine learning for determining the optimal number of clusters in a dataset. It involves plotting the variance explained by different numbers of clusters and identifying the "elbow" point, where the rate of variance decreases sharply levels off, suggesting an appropriate cluster count for analysis or model training.

---

3. Explain Hierarchical clustering.

Hierarchical Clustering in Machine Learning

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled 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.

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.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

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

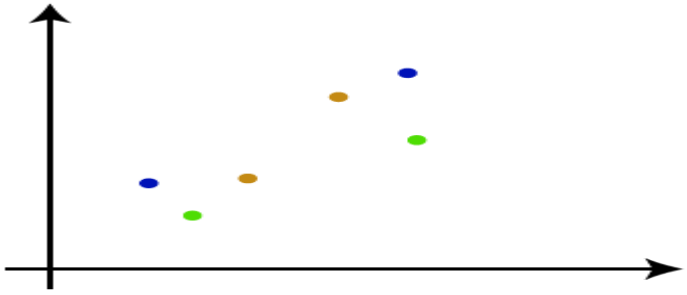
pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

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

How the Agglomerative Hierarchical clustering Work?

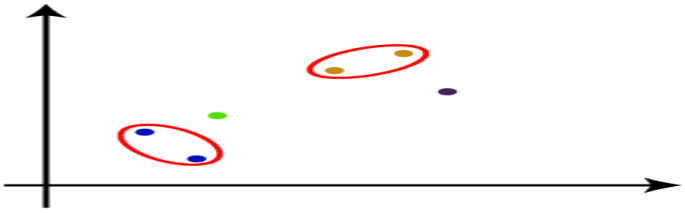
The working of the AHC algorithm can be explained using the below steps:

- **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.

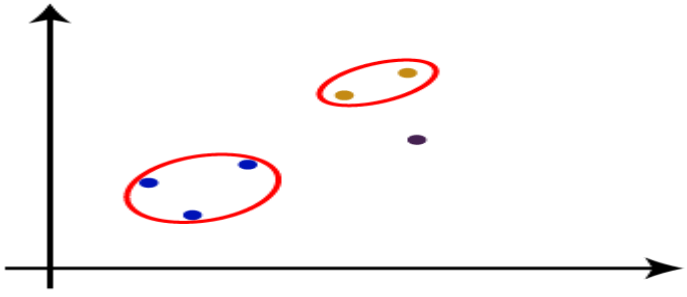


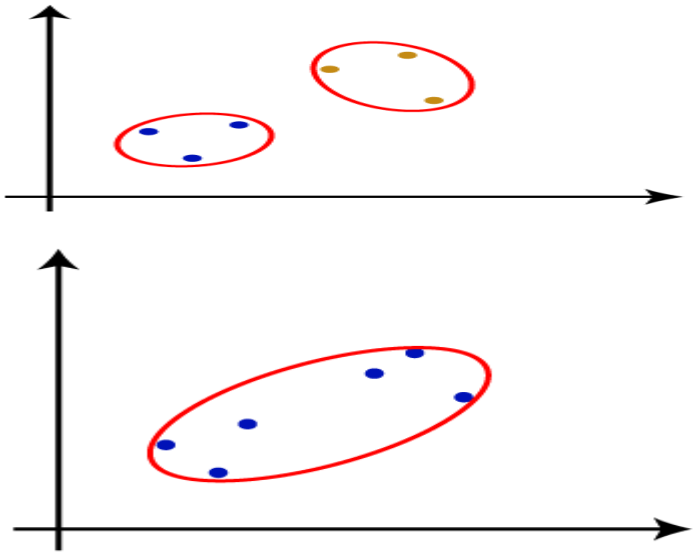
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

- **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:





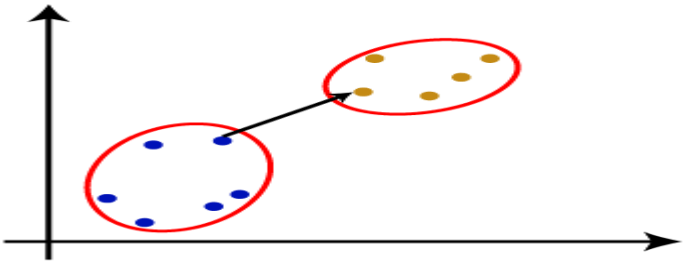
- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

Note: To better understand hierarchical clustering, it is advised to have a look on k-means clustering

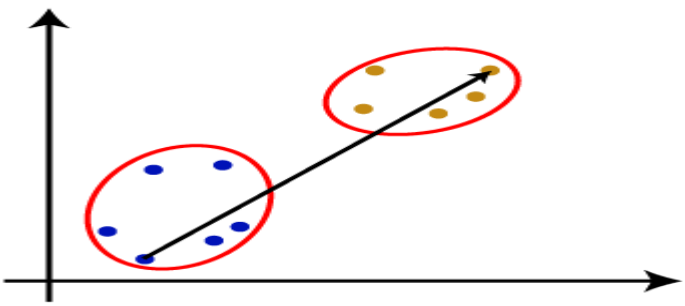
Measure for the distance between two clusters

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:

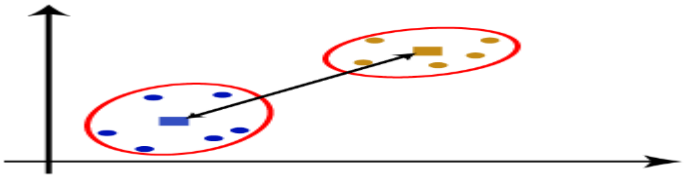
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



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:

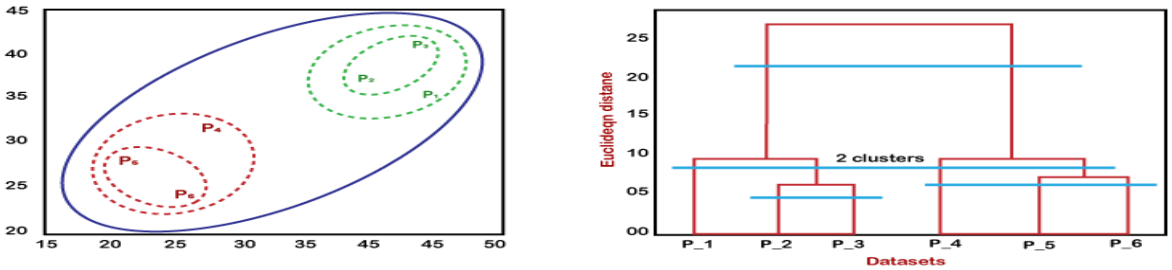


From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

Woking of Dendrogram in Hierarchical clustering

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

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



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

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

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

---

5. What is Agglomerative Hierarchical clustering?

Agglomerative Hierarchical Clustering

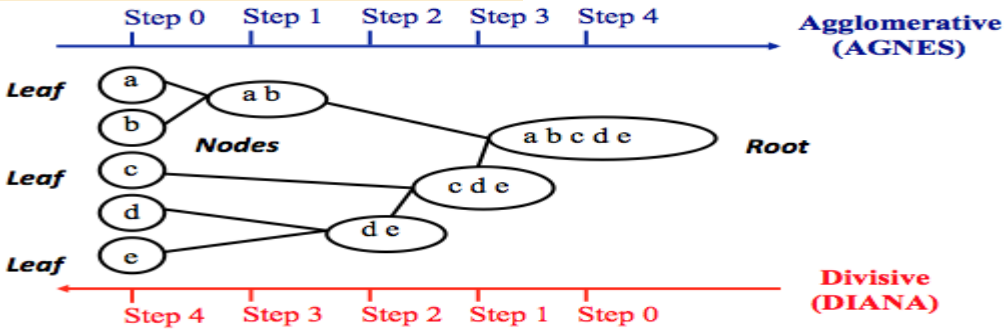
The **agglomerative clustering** is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as *AGNES* (*Agglomerative*

Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named *dendrogram*.

Algorithm

Agglomerative clustering works in a "bottom-up" manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below). The inverse of agglomerative clustering is *divisive clustering*, which is also known as DIANA (*Divise Analysis*) and it works in a "top-down" manner. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster (see figure below).

Note that, agglomerative clustering is good at identifying small clusters. Divisive clustering is good at identifying large clusters. In this article, we'll focus mainly on agglomerative hierarchical clustering.



Steps to agglomerative hierarchical clustering

We'll follow the steps below to perform agglomerative hierarchical clustering using R software:

1. Preparing the data
2. Computing (dis)similarity information between every pair of objects in the data set.
3. Using linkage function to group objects into hierarchical cluster tree, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.
4. Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

We'll describe each of these steps in the next section.

Data structure and preparation

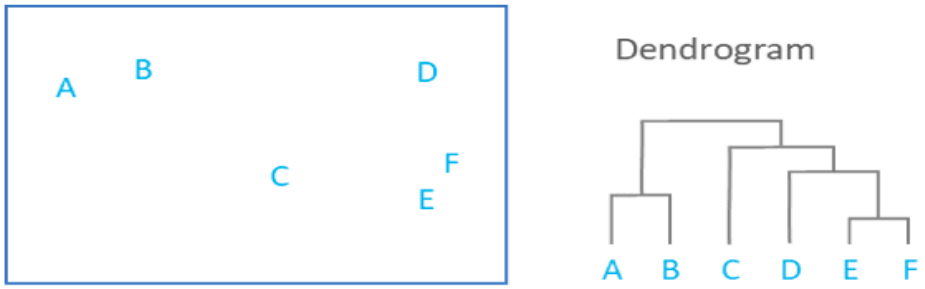
The data should be a numeric matrix with:

- rows representing observations (individuals);
- and columns representing variables.

---

5. **Explain working of dendrogram in Hierarchical clustering.**
6. A *dendrogram* is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from *hierarchical clustering*. The main use of a dendrogram is to work out the best way to allocate objects to clusters. The

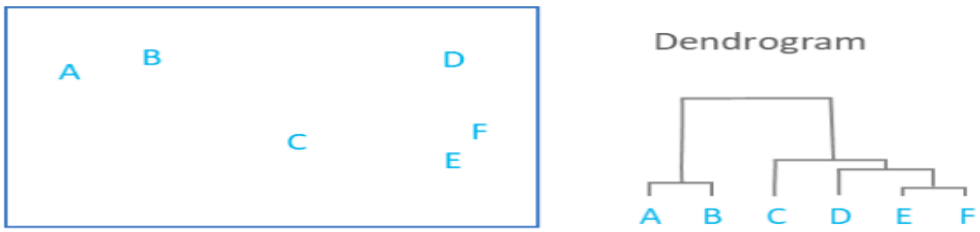
dendrogram below shows the hierarchical clustering of six *observations* shown on the *scatterplot* to the left. (Dendrogram is often miswritten as dendogram.)



7.
8.

9. **Create your own hierarchical cluster analysis**

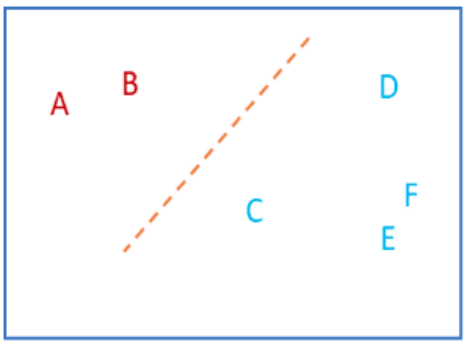
10.
11. To create your own dendrogram using hierarchical clustering, simply click the button above!
12. How to read a dendrogram
13. The key to interpreting a dendrogram is to focus on the height at which any two objects are joined together. In the example above, we can see that E and F are most similar, as the height of the link that joins them together is the smallest. The next two most similar objects are A and B.
14. In the dendrogram above, the height of the dendrogram indicates the order in which the clusters were joined. A more informative dendrogram can be created where the heights reflect the distance between the clusters as is shown below. In this case, the dendrogram shows us that the big difference between clusters is between the cluster of A and B versus that of C, D, E, and F.

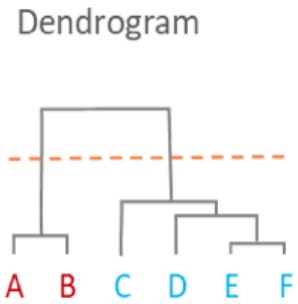


15.
16. It is important to appreciate that the dendrogram is a summary of the distance matrix, and, as occurs with most summaries, information is lost. For example, the dendrogram suggests that C and D are much closer to each other than is C to B, but the original data (shown in the scatterplot), shows us that this is not true. To use some jargon, a dendrogram is only accurate when data satisfies the *ultrametric tree inequality*, and this is unlikely for any real-world data.
17. The consequence of the information loss is that the dendrograms are most accurate at the bottom, showing which items are very similar.
18.

19. **Create your own hierarchical cluster analysis**

20.
21. Allocating observations to clusters
22. Observations are allocated to clusters by drawing a horizontal line through the dendrogram. Observations that are joined together below the line are in clusters. In the example below, we have two clusters. One cluster combines A and B, and a second cluster combining C, D, E, and F.





23.
24.
25. **Create your own hierarchical cluster analysis**
26.
27. Dendrograms cannot tell you how many clusters you should have
28. A common mistake people make when reading dendrograms is to assume that the shape of the dendrogram gives a clue as to how many clusters exist. In the example above, the (incorrect) interpretation is that the dendrogram shows that there are two clusters, as the distance between the clusters (the vertical segments of the dendrogram) are highest between two and three clusters.
29. Interpretation of this kind is justified only when the ultrametric tree inequality holds, which, as mentioned above, is very rare. In general, it is a mistake to use dendrograms as a tool for determining the number of clusters in data. Where there is an obviously "correct" number of clusters this will often be evident in a dendrogram. However, dendrograms often suggest a correct number of clusters when there is no real evidence to support the conclusion.

---

7. Explain Association Rule mining.

Association rule mining is a technique used to uncover hidden relationships between variables in large datasets. It is a popular method in data mining and machine learning and has a wide range of applications in various fields, such as **market basket analysis**, customer segmentation, and fraud detection.

In this article, we will explore association rule mining in Python, including its use cases, algorithms, and implementation. We will start with a brief overview of association rule mining and its applications and then delve into the details of the algorithms and their implementation in Python. Finally, we will conclude with a summary of the key points covered in this article.

What is Association Rule Mining?

Association rule mining is a technique used to identify patterns in large data sets. It involves finding relationships between variables in the data and using those relationships to make predictions or decisions. The goal of association rule mining is to uncover rules that describe the relationships between different items in the data set.

For example, consider a dataset of transactions at a grocery store. Association rule mining could be used to identify relationships between items that are frequently purchased together. For example, the rule "If a customer buys bread, they are also likely to buy milk" is an association rule that could be mined from this data set. We can use such rules to inform decisions about store layout, product placement, and marketing efforts.

Association rule mining typically involves using algorithms to analyze the data and identify the relationships. These algorithms can be based on statistical methods or machine learning techniques. The resulting rules are often expressed in the form of "if-then" statements, where

the "if" part represents the antecedent (the condition being tested) and the "then" part represents the consequent (the outcome that occurs if the condition is met).
Association rule mining is an important technique in data analysis because it allows users to discover patterns or relationships within data that may not be immediately apparent. By identifying associations between variables, association rule mining can help users understand the relationships between different variables and how those variables may be related to one another.
This can be useful for various purposes, such as identifying market trends, detecting fraudulent activity, or understanding customer behavior. Association rule mining can also be used as a stepping stone for other types of data analysis, such as predicting outcomes or identifying key drivers of certain phenomena. Overall, association rule mining is a valuable tool for extracting insights and understanding the underlying structure of data.

Use Cases of Association Rule Mining:
Association rule mining is commonly used in a variety of applications, some common ones are:

Market Basket Analysis
One of the most well-known applications of association rule mining is in market basket analysis. This involves analyzing the items customers purchase together to understand their purchasing habits and preferences.
For example, a retailer might use association rule mining to discover that customers who purchase diapers are also likely to purchase baby formula. We can use this information to optimize product placements and promotions to increase sales.
To learn more about Market Basket Analysis, check out our **Market Basket Analysis in R tutorial**.

Customer Segmentation
Association rule mining can also be used to segment customers based on their purchasing habits.
For example, a company might use association rule mining to discover that customers who purchase certain types of products are more likely to be younger. Similarly, they could learn that customers who purchase certain combinations of products are more likely to be located in specific geographic regions.
We can use this information to tailor marketing campaigns and personalized recommendations to specific customer segments. Discover more about this topic in our **introduction to customer segmentation in Python tutorial**.

Fraud Detection
You can also use association rule mining to detect fraudulent activity. For example, a credit card company might use association rule mining to identify patterns of fraudulent transactions, such as multiple purchases from the same merchant within a short period of time.
We can then use this information can to flag potentially fraudulent activity and take preventative measures to protect customers. In our **Data Sciece in Banking: Fraud Detection** blog, you can learn more about how the process works.

Social network analysis
Various companies use association rule mining to identify patterns in social media data that can inform the analysis of social networks.

For example, an analysis of Twitter data might reveal that users who tweet about a particular topic are also likely to tweet about other related topics, which could inform the identification of groups or communities within the network.

Recommendation systems
Association rule mining can be used to suggest items that a customer might be interested in based on their past purchases or browsing history. For example, a music streaming service might use association rule mining to recommend new artists or albums to a user based on their listening history.

---

8. Explain apriori algorithm.

Apriori Algorithm

Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.

The primary objective of the apriori algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions. Let's understand the apriori algorithm with the help of an example; suppose you go to Big Bazar and buy different products. It helps the customers buy their products with ease and increases the sales performance of the Big Bazar. In this tutorial, we will discuss the apriori algorithm with examples.

Introduction

We take an example to understand the concept better. You must have noticed that the Pizza shop seller makes a pizza, soft drink, and breadstick combo together. He also offers a discount to their customers who buy these combos. Do you ever think why does he do so? He thinks that customers who buy pizza also buy soft drinks and breadsticks. However, by making combos, he makes it easy for the customers. At the same time, he also increases his sales performance.

Similarly, you go to Big Bazar, and you will find biscuits, chips, and Chocolate bundled together. It shows that the shopkeeper makes it comfortable for the customers to buy these products in the same place.

The above two examples are the best examples of Association Rules in Data Mining. It helps us to learn the concept of apriori algorithms.

What is Apriori Algorithm?

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Let's take an example to understand this concept.

We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.

Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 percent of customers who bought biscuits bought chocolates also.

Lift

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

= 50/10 = 5

It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

How does the Apriori Algorithm work in Data Mining?

We will understand this algorithm with the help of an example

Consider a Big Bazar scenario where the product set is P = {Rice, Pulse, Oil, Milk, Apple}. The database comprises six transactions where 1 represents the presence of the product and 0 represents the absence of the product.

Transaction ID	Rice	Pulse	Oil Milk	Apple	
t1	1	1	1	0	0
t2	0	1	1	1	0
t3	0	0	0	1	1
t4	1	1	0	1	0
t5	1	1	1	0	1
t6	1	1	1	1	1

The Apriori Algorithm makes the given assumptions

- All subsets of a frequent itemset must be frequent.
- The subsets of an infrequent item set must be infrequent.
- Fix a threshold support level. In our case, we have fixed it at 50 percent.

Step 1

Make a frequency table of all the products that appear in all the transactions. Now, short the frequency table to add only those products with a threshold support level of over 50 percent. We find the given frequency table.

Product	Frequency (Number of transactions)

Rice (R)	4
Pulse(P)	5
Oil(O)	4
Milk(M)	4

The above table indicated the products frequently bought by the customers.

Step 2

Create pairs of products such as RP, RO, RM, PO, PM, OM. You will get the given frequency table.

Itemset	Frequency (Number of transactions)
RP	4
RO	3
RM	2
PO	4
PM	3
OM	2

Step 3

Implementing the same threshold support of 50 percent and consider the products that are more than 50 percent. In our case, it is more than 3

Thus, we get RP, RO, PO, and PM

Step 4

Now, look for a set of three products that the customers buy together. We get the given combination.

1. RP and RO give RPO
2. PO and PM give POM

Step 5

Calculate the frequency of the two itemsets, and you will get the given frequency table.

Itemset	Frequency (Number of transactions)
RPO	4
POM	3

If you implement the threshold assumption, you can figure out that the customers' set of three products is RPO.

We have considered an easy example to discuss the apriori algorithm in data mining. In reality, you find thousands of such combinations.

How to improve the efficiency of the Apriori Algorithm?

There are various methods used for the efficiency of the Apriori algorithm

Hash-based itemset counting

In hash-based itemset counting, you need to exclude the k-itemset whose equivalent hashing bucket count is least than the threshold is an infrequent itemset.

Transaction Reduction

In transaction reduction, a transaction not involving any frequent X itemset becomes not valuable in subsequent scans.

Apriori Algorithm in data mining

We have already discussed an example of the apriori algorithm related to the frequent itemset generation. Apriori algorithm has many applications in data mining.

The primary requirements to find the association rules in data mining are given below.

Use Brute Force

Analyze all the rules and find the support and confidence levels for the individual rule. Afterward, eliminate the values which are less than the threshold support and confidence levels.

The two-step approaches

The two-step approach is a better option to find the associations rules than the Brute Force method.

Step 1

In this article, we have already discussed how to create the frequency table and calculate itemsets having a greater support value than that of the threshold support.

Step 2

To create association rules, you need to use a binary partition of the frequent itemsets. You need to choose the ones having the highest confidence levels.

In the above example, you can see that the RPO combination was the frequent itemset. Now, we find out all the rules using RPO.

RP-O, RO-P, PO-R, O-RP, P-RO, R-PO

You can see that there are six different combinations. Therefore, if you have n elements, there will be 2^n - 2 candidate association rules.

Advantages of Apriori Algorithm

- It is used to calculate large itemsets.
- Simple to understand and apply.

Disadvantages of Apriori Algorithms

- Apriori algorithm is an expensive method to find support since the calculation has to pass through the whole database.
- Sometimes, you need a huge number of candidate rules, so it becomes computationally more expensive.

---

9. Explain Eclat algorithm

The ECLAT algorithm stands for **Equivalence Class Clustering and bottom-up Lattice Traversal**. It is one of the popular methods of Association Rule mining. It is a more efficient and scalable version of the Apriori algorithm. While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph. This vertical approach of the ECLAT algorithm makes it a faster algorithm than the Apriori algorithm.

How the algorithm work? :

The basic idea is to use Transaction Id Sets(tidsets) intersections to compute the support value of a candidate and avoiding the generation of subsets which do not exist in the prefix tree. In the first call of the function, all single items are used along with their tidsets. Then the function is called recursively and in each recursive call, each item-tidset pair is verified and combined with other item-tidset pairs. This process is continued until no candidate item-tidset pairs can be combined.

Let us now understand the above stated working with an example:-

Consider the following transactions record:-

Transaction Id	Bread	Butter	Milk	Coke	Jam
T1	1	1	0	0	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	1	0	1	0	0
T6	0	1	1	0	0
T7	1	0	1	0	0
T8	1	1	1	0	1
T9	1	1	1	0	0

The above-given data is a boolean matrix where for each cell (i, j), the value denotes whether the j'th item is included in the i'th transaction or not. 1 means true while 0 means false.
We now call the function for the first time and arrange each item with it's tidset in a tabular fashion:-

k = 1, minimum support = 2

Item	Tidset
Bread	{T1, T4, T5, T7, T8, T9}
Butter	{T1, T2, T3, T4, T6, T8, T9}
Milk	{T3, T5, T6, T7, T8, T9}
Coke	{T2, T4}
Jam	{T1, T8}

We now recursively call the function till no more item-tidset pairs can be combined:-

k = 2

Item	Tidset
{Bread, Butter}	{T1, T4, T8, T9}
{Bread, Milk}	{T5, T7, T8, T9}
{Bread, Coke}	{T4}
{Bread, Jam}	{T1, T8}
{Butter, Milk}	{T3, T6, T8, T9}
{Butter, Coke}	{T2, T4}
{Butter, Jam}	{T1, T8}
{Milk, Jam}	{T8}

k = 3

Item	Tidset
{Bread, Butter, Milk}	{T8, T9}
{Bread, Butter, Jam}	{T1, T8}

k = 4

Item	Tidset
{Bread, Butter, Milk, Jam}	{T8}

We stop at k = 4 because there are no more item-tidset pairs to combine.
Since minimum support = 2, we conclude the following rules from the given dataset:-

Items Bought	Recommended Products
Bread	Butter
Bread	Milk
Bread	Jam
Butter	Milk
Butter	Coke
Butter	Jam
Bread and Butter	Milk
Bread and Butter	Jam

Advantages over Apriori algorithm:-

1. **Memory Requirements:** Since the ECLAT algorithm uses a Depth-First Search approach, it uses less memory than Apriori algorithm.
2. **Speed:** The ECLAT algorithm is typically faster than the Apriori algorithm.
3. **Number of Computations:** The ECLAT algorithm does not involve the repeated scanning of the data to compute the individual support values.

---

10. Explain F-P Growth algorithm

FP Growth Algorithm in Data Mining

In Data Mining, finding frequent patterns in large databases is very important and has been studied on a large scale in the past few years. Unfortunately, this task is computationally expensive, especially when many patterns exist.

The FP-Growth Algorithm proposed by ***Han in***. This is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the TreeProjection. In some later works, it was proved that FP-Growth performs better than other methods, including ***Eclat*** and ***Relim***. The popularity and efficiency of the FP-Growth Algorithm contribute to many studies that propose variations to improve its performance.

What is FP 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.

This algorithm works as follows:

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

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.

In large databases, holding the FP tree in the main memory is impossible. A strategy to cope with this problem is to partition the database into a set of smaller databases (called projected databases) and then construct an FP-tree from each of these smaller databases.

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. The purpose of the FP tree is to mine the most frequent pattern. 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. The associations of the nodes with the lower nodes, that is, the item sets with the other item sets, are maintained while forming the tree.

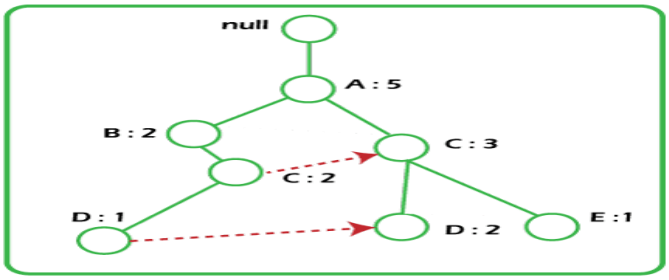
Han defines the FP-tree as the tree structure given below:

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:

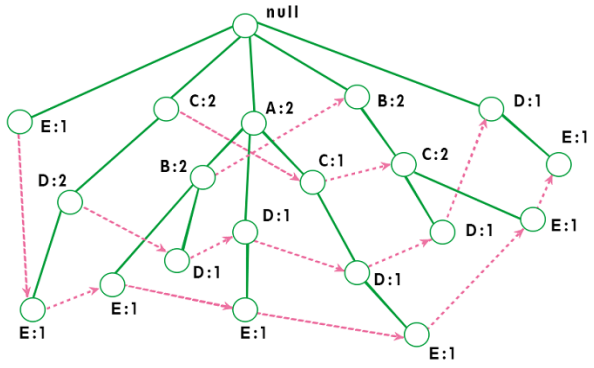
- Item-name: registers which item is represented by the node;
- Count: the number of transactions represented by the portion of the path reaching the node;
- Node-link: links to the next node in the FP-tree carrying the same item name or null if there is none.

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 below diagram is an example of a best-case scenario that occurs when all transactions have the same itemset; the size of the FP-tree will be only a single branch of nodes.



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. The diagram below shows how a worst-case scenario FP-tree might appear. As you can see, the tree's complexity grows with each transaction's uniqueness.



Algorithm by Han

The original algorithm to construct the FP-Tree defined by Han is given below:

Algorithm 1: FP-tree construction

***Input*:** A transaction database DB and a minimum support threshold?

***Output*:** FP-tree, the frequent-pattern tree of DB.

***Method*:** The FP-tree is constructed as follows.

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, and then the next itemset with the lower count. 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, 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. The common node and new node count are 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 is called a conditional pattern base.
   A 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, 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.

Using this algorithm, the FP-tree is constructed in two database scans. The first scan collects and sorts the set of frequent items, and the second constructs the FP-Tree.

Example

Support threshold=50%, Confidence= 60%

Table 1:

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5

T6	I1,I2,I3,I4

Solution: Support threshold=50% => 0.5*6= 3 => min_sup=3

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

Table 2: Count of each item
Table 3: Sort the itemset in descending order.

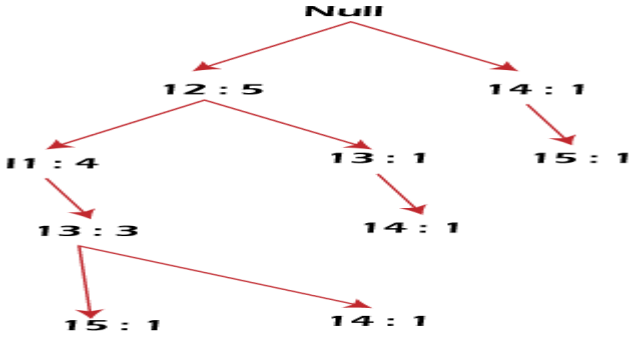
Item	Count
I2	5
I1	4
I3	4
I4	4

Build FP Tree

Let's build the FP tree in the following steps, such as:

1. Considering the root node null.
2. The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child, I1 is linked to I2 and I3 is linked to I1.
3. T2: I2, I3, and I4 contain I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share the I2 node as common as it is already used in T1.
4. Increment the count of I2 by 1, and I3 is linked as a child to I2, and I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.
5. T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.
6. T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node. Hence it will be incremented by 1. Similarly I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.
7. T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.

8. T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.



Mining of FP-tree is summarized below:

1. The lowest node item, I5, is not considered as it does not have a min support count. Hence it is deleted.
2. The next lower node is I4. I4 occurs in 2 branches , {I2,I1,I3:,I41},{I2,I3,I4:1}. Therefore considering I4 as suffix the prefix paths will be {I2, I1, I3:1}, {I2, I3: 1} this forms the conditional pattern base.
3. The conditional pattern base is considered a transaction database, and an FP tree is constructed. This will contain {I2:2, I3:2}, I1 is not considered as it does not meet the min support count.
4. This path will generate all combinations of frequent patterns : {I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
5. For I3, the prefix path would be: {I2,I1:3},{I2:1}, this will generate a 2 node FP-tree : {I2:4, I1:3} and frequent patterns are generated: {I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}.
6. For I1, the prefix path would be: {I2:4} this will generate a single node FP-tree: {I2:4} and frequent patterns are generated: {I2, I1:4}.

Item	Conditional Pattern Base	Conditional FP-tree	Frequent P
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I
I1	{I2:4}	{I2:4}	{I2,I1:4}

The diagram given below depicts the conditional FP tree associated with the conditional node I3.