# CO-3

## What is Machine Learning ?

Arthur Samuel, a pioneer in the field of artificial intelligence and computer gaming, coined the term "Machine Learning". He defined machine learning as – a "Field of study that gives computers the capability to learn without being explicitly programmed". In a very layman's manner, Machine Learning(ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance.

The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

Example: Training of students during exams. While preparing for the exams students don't actually cram the subject but try to learn it with complete understanding. Before the examination, they feed their machine(brain) with a good amount of high- quality data (questions and answers from different books or teachers' notes, or online video lectures). Actually, they are training their brain with input as well as output i.e. what kind of approach or logic do they have to solve a different kinds of questions. Each time they solve practice test papers and find the performance (accuracy /score) by comparing answers with the answer key given, Gradually, the performance keeps on increasing, gaining more confidence with the adopted approach. That's how actually models are built, train machine with data (both inputs and outputs are given to the model), and when the time comes test on data (with input only) and achieve our model scores by comparing its answer with the actual output which has not been fed while training. Researchers are working with assiduous efforts to improve algorithms, and techniques so that these models perform even much better.

## Basic Difference in ML and Traditional Programming?

**Traditional Programming:** We feed in DATA (Input) + PROGRAM (logic), run it on the machine, and get the output.

**Machine Learning:** We feed in DATA(Input) + Output, run it on the machine during training and the machine creates its own program(logic), which can be evaluated while testing.

**What does exactly learning mean for a computer?** A computer is said to be learning from **Experiences** with respect to some class of **Tasks** if its performance in a given task improves with the Experience.

A computer program is said to learn from experience **E** with respect to some class oftasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E Example:** playing checkers. **E** = the experience of playing many games of checkers **T** = the task of playing checkers. **P** = the probability that the program will win the next game In general, any machine learning problem can be assigned to one of two broad classifications: Supervised learning and Unsupervised learning.

**How does ML work?**

Gathering past data in any form suitable for processing. The better the quality of data, the more suitable it will be for modeling

Data Processing – Sometimes, the data collected is in raw form and it needs to be pre-processed. Example: Some tuples may have missing values for certain attributes, and, in this case, it has to be filled with suitable values in order to perform machine learning or any form of data mining. Missing values for numerical attributes such as the price of the house may be replaced with the mean value of the attribute whereas missing values for categorical attributes may be replaced with the attribute with the highest mode. This invariably depends on the types of filters we use. If data is in the form of text or images then converting it to numerical form will be required, be it a list or array or matrix. Simply, Data is to be made relevant and consistent. It is to be converted into a format understandable by the machine

- Divide the input data into training, cross-validation, and test sets. The ratio between the respective sets must be 6:2:2

- Building models with suitable algorithms and techniques on the training set.
- Testing our conceptualized model with data that was not fed to the model at the time of training and evaluating its performance using metrics such as F1 score, precision, and recall.
- Linear Algebra
- Statistics and Probability
- Calculus
- Graph theory
- Programming Skills – Languages such as Python, R, MATLAB, C++, or Octave

Machine learning is programming computers to optimize a performance criterion using example data or past experience . We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.

- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

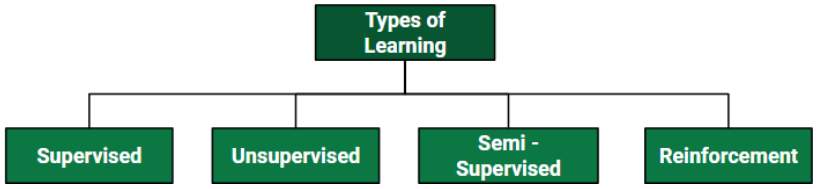
**Definition of learning:** A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks T, as measured by P , improves with experience E.

**Examples**

- Handwriting recognition learning problem
  - Task T : Recognizing and classifying handwritten words within images
  - Performance P : Percent of words correctly classified
  - Training experience E : A dataset of handwritten words with given classifications
- A robot driving learning problem
  - Task T : Driving on highways using vision sensors
  - Performance P : Average distance traveled before an error
  - Training experience E : A sequence of images and steering commands recorded while observing a human driver

**Definition:** A computer program which learns from experience is called a machine learning program or simply a learning program .

## Classification of Machine Learning




Machine learning implementations are classified into four major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:

**A. Supervised learning:**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. The given data is labeled . Both *classification* and *regression* problems are supervised learning problems .

- Example — Consider the following data regarding patients entering a clinic . The data consists of the gender and age of the patients and each patient is labeled as "healthy" or "sick".

| gender | age | label |
|---|---|---|
| M | 48 | sick |
| M | 67 | sick |
| F | 53 | healthy |
| M | 49 | sick |
| F | 32 | healthy |
| M | 34 | healthy |
| M | 21 | healthy |

**Unsupervised learning:**

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, classification or categorization is not included in the observations. Example: Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients.

| gender | age |
|---|---|
| M | 48 |
| M | 67 |
| F | 53 |
| M | 49 |
| F | 34 |
| M | 21 |

As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects. Some recommendation systems that you find on the web in the form of marketing automation are basedon this type of learning.

**C. Reinforcement learning:**

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

A learner is not told what actions to take as in most forms of machine learning but instead must discover which actions yield the most reward by trying them. For example — Consider teaching a dog a new trick: we cannot tell him what to do, what not to do, but we can reward/punish it if it does the right/wrong thing.

When watching the video, notice how the program is initially clumsy and unskilled but steadily improves with training until it becomes a champion.

## Semi-supervised learning:

Where an incomplete training signal is given: a training set with some (often many) of the target outputs missing. There is a special case of this principle known as Transduction where the entire set of problem instances is known at learning time, except that part of the targets are missing. Semi-supervised learning is an approach to machine learning that combines small labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning and supervised learning.

## Supervised learning

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labelled data.

**For instance**, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this:

- If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –**Apple**.
- If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –**Banana**.

Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, andasked to identify it.

Since the machine has already learned the things from previous data and this time has to use it wisely. It willfirst classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit).

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 datais 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.

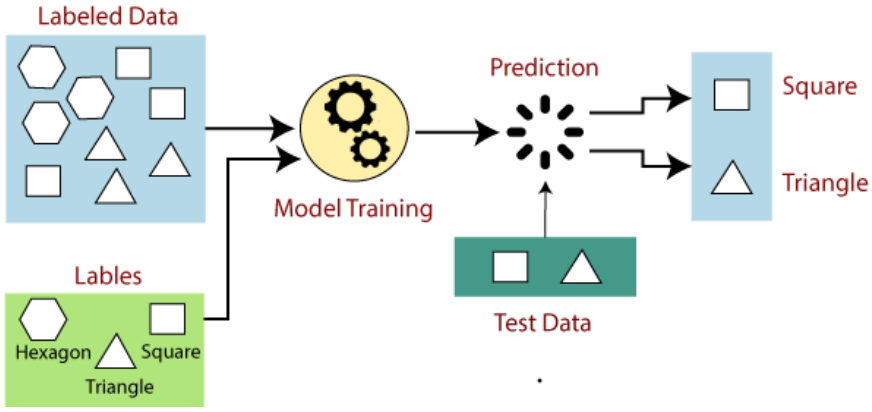
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

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 onthe bases of a number of sides, and predicts the output.

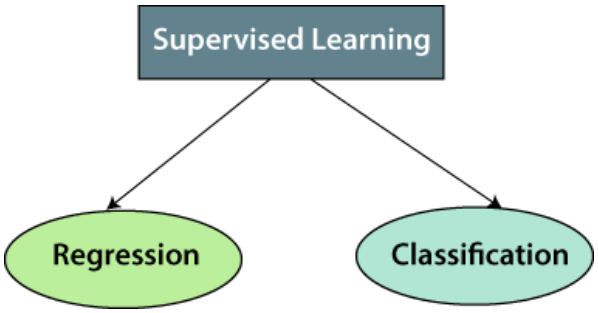
## Steps Involved in Supervised Learning:

- 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 modelcan 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

## 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.

### Supervised learning is classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” , “disease” or “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.

**Types:-**

- Regression
- Logistic Regression
- Classification
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

**Advantages:-**

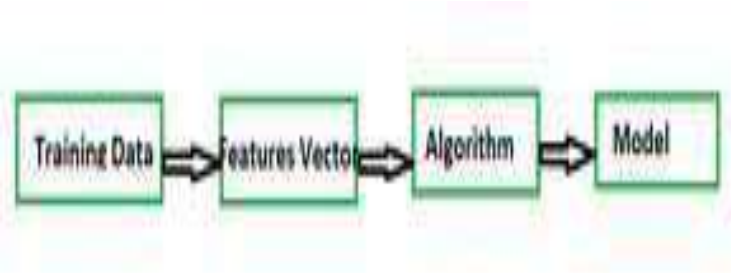
- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.

- Supervised machine learning helps to solve various types of real-world computation problems.

**Disadvantages:-**

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.

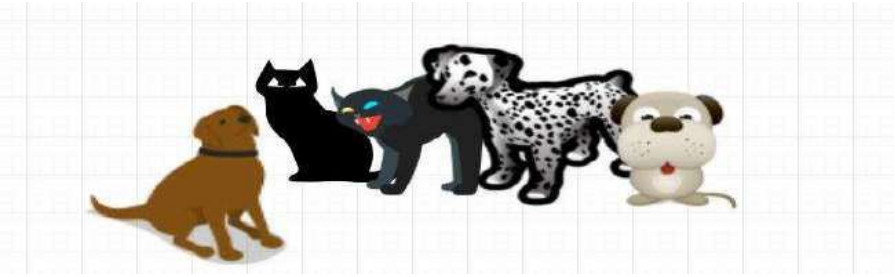
*Steps*




**Unsupervised learning**

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself. **For instance**, suppose it is given an image having both dogs and cats which it has never seen.



Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats'. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having **dogs** in them and the second part may contain all pics having **cats** in them. Here you didn't learn anything before, which means no training data or examples.

It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

Unsupervised learning is classified into two categories of algorithms:

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

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?

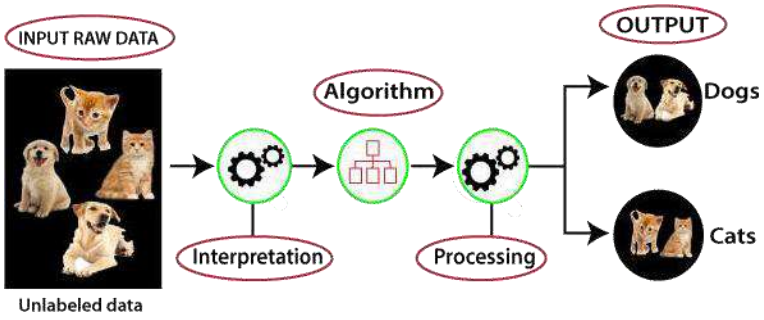
Below are some main reasons which describe the importance of Unsupervised Learning:

- 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:

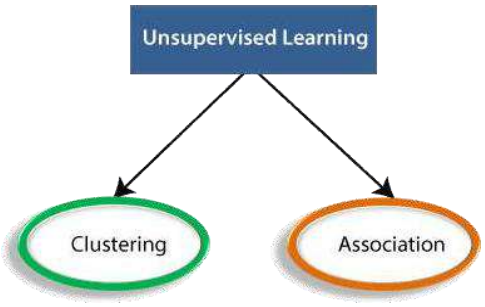



Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and 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.

## Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principle Component Analysis**
- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

## Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

## Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Types of Unsupervised Learning:-

**Clustering**

1. Exclusive (partitioning)
2. Agglomerative
3. Overlapping
4. Probabilistic

**Clustering Types:-**

1. Hierarchical clustering
2. K-means clustering
3. Principal Component Analysis
4. Singular Value Decomposition
5. Independent Component Analysis

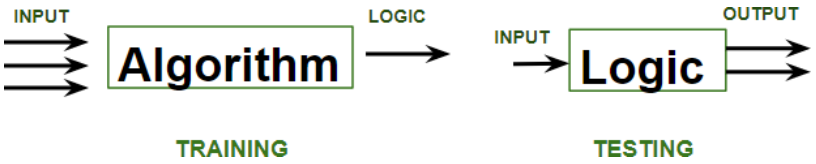
**Supervised vs. Unsupervised Machine Learning**

| Parameters | Supervised machine learning | Unsupervised machine learning |
|---|---|---|
| Input Data | Algorithms are trained using labeled data. | Algorithms are used against data that is not labeled |
| Computational Complexity | Simpler method | Computationally complex |
| Accuracy | Highly accurate | Less accurate |

| | | |
|---|---|---|
| No. of classes | No. of classes is known | No. of classes is not known |
| Data Analysis | Uses offline analysis | Uses real-time analysis of data |
| Algorithms used | Linear and Logistics regression,Random forest, Support Vector Machine, Neural Network, etc. | K-Means clustering, Hierarchical clustering,Apriori algorithm, etc. |

**Types of Learning – Supervised Learning**

Let us discuss what is learning for a machine is as shown below media as follows:




A machine is said to be learning from **past Experiences**(data feed-in) with respect to some cla of **tasks** if

its **Performance** in a given Task improves with the Experience. For example, assume that a machine has to predict

whether a customer will buy a specific product let's say "Antivirus" this year or not. The machine will do it by looking at

the **previous knowledge/past experiences** i.e the data of products that the customer had bought every year and if he

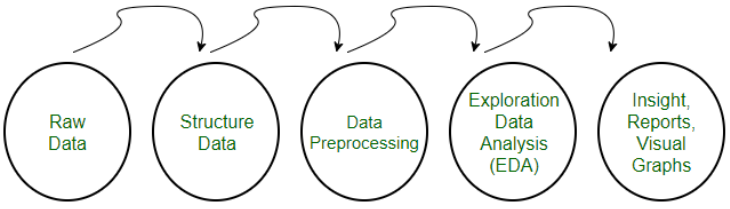
buys Antivirus every year, then there is a high probability that the customer is going to buy an antivirus this year as well.

This is how machine learning works at the basic conceptual level.

## Data Preprocessing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.




**Steps Involved in Data Preprocessing:**

**1. Data Cleaning:**

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

- **(a). Missing Data:**

This situation arises when some data is missing in the data. It can be handled in various ways.

Some of them are:

1. **Ignore the tuples:**

This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

**2. Fill the Missing values:**

There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

**3.(b). Noisy Data:**

Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

2. **Binning Method:**

This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

3. **Regression:**

Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

4. **Clustering:**

This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

**2. Data Transformation:**

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

**Normalization:**

It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

**Attribute Selection:**

In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

**Discretization:**

This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

**Concept Hierarchy Generation:**

Here attributes are converted from lower level to higher level in hierarchy. For Example- The attribute "city" can be converted to "country".

### 3. Data Reduction:

Data reduction is a crucial step in the data mining process that involves reducing the size of the dataset while preserving the important information. This is done to improve the efficiency of data analysis and to avoid overfitting of the model. Some common steps involved in data reduction are:

**Feature Selection:** This involves selecting a subset of relevant features from the dataset. Feature selection is often performed to remove irrelevant or redundant features from the dataset. It can be done using various techniques such as correlation analysis, mutual information, and principal component analysis (PCA).

**Feature Extraction:** This involves transforming the data into a lower-dimensional space while preserving the important information. Feature extraction is often used when the original features are high-dimensional and complex. It can be done using techniques such as PCA, linear discriminant analysis (LDA), and non-negative matrix factorization (NMF).

**Sampling:** This involves selecting a subset of data points from the dataset. Sampling is often used to reduce the size of the dataset while preserving the important information. It can be done using techniques such as random sampling, stratified sampling, and systematic sampling.

**Clustering:** This involves grouping similar data points together into clusters. Clustering is often used to reduce the size of the dataset by replacing similar data points with a representative centroid. It can be done using techniques such as k-means, hierarchical clustering, and density-based clustering.

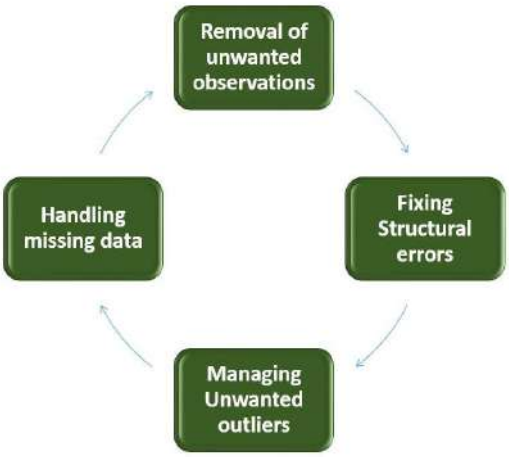
**Compression:** This involves compressing the dataset while preserving the important information. Compression is often used to reduce the size of the dataset for storage and transmission purposes. It can be done using techniques such as wavelet compression, JPEG compression, and gzip compression.

### Why is Data Cleaning Important?

Data cleansing is a crucial step in the data preparation process, playing an important role in ensuring the accuracy, reliability, and overall quality of a dataset.For decision-making, the integrity of the conclusions drawn heavily relies on the cleanliness of the underlying data. Without proper data cleaning, inaccuracies, outliers, missing values, and inconsistencies can compromise the validity of analytical results. Moreover, clean data facilitates more effective modeling and pattern recognition, as algorithms perform optimally when fed high-quality, error-free input.

### Steps to Perform Data Cleanliness

Performing data cleaning involves a systematic process to identify and rectify errors, inconsistencies, and inaccuracies in a dataset. The following are essential steps to perform data cleaning.




- **Removal of Unwanted Observations**: Identify and eliminate irrelevant or redundant observations from the dataset. The step involves scrutinizing data entries for duplicate records, irrelevant information, or data points that do not contribute meaningfully to the analysis. Removing unwanted observations streamlines the dataset, reducing noise and improving the overall quality.
- **Fixing Structure errors:** Address structural issues in the dataset, such as inconsistencies in data formats, naming conventions, or variable types. Standardize formats, correct naming discrepancies, and ensure uniformity in data representation. Fixing structure errors enhances data consistency and facilitates accurate analysis and interpretation.
- **Managing Unwanted outliers:** Identify and manage outliers, which are data points significantly deviating from the norm. Depending on the context, decide whether to remove outliers or transform them to minimize their impact on analysis. Managing outliers is crucial for obtaining more accurate and reliable insights from the data.
- **Handling Missing Data:** Devise strategies to handle missing data effectively. This may involve imputing missing values based on statistical methods, removing records with missing values, or employing advanced imputation techniques. Handling missing data ensures a more complete dataset, preventing biases and maintaining the integrity of analyses.

## Splitting Data

Splitting facts for system mastering models is an crucial step within the version improvement process. It includes dividing the to be had dataset into separate subsets for education, validation, and trying out the version. Here are a few common processes for splitting data:

**1. Train-Test Split:** The dataset is divided right into a training set and a trying out set. The education set is used to educate the model, even as the checking out set is used to assess the model's overall performance. The regular cut up is 70-eighty% for training and 20-30% for checking out, but this may vary depending on the scale of the dataset and the precise use case.

**2. Train-Validation-Test Split:** The dataset is split into three subsets – a schooling set, a validation set, and a trying out set. The training set is used to train the version, the validation set is used to tune hyperparameters and validate the version's overall performance for the duration of training, and the testing set is used to evaluate the very last version's overall performance.

**3. K-fold Cross Validation:** The dataset is divided into ok equally sized folds, and the version is educated and evaluated okay instances. Each time, k-1 folds are used for training, and 1 fold is used for validation/testing. This allows in acquiring greater strong overall performance estimates and reduces the variance in version evaluation.

**4. Stratified Sampling:** This technique guarantees that the distribution of training or other essential features is preserved in the training and trying out units. This is in particular beneficial when coping with imbalanced datasets, wherein some classes may additionally have only a few samples.

**5. Time-primarily based Split:** When coping with time collection facts, consisting of stock costs or weather statistics, the dataset is regularly cut up into schooling and checking out sets based on a chronological order. This facilitates in comparing the model's performance on future unseen facts

**What is Data Normalization?**

Data normalization is a vital pre-processing, mapping, and scaling method that helps forecasting and prediction models become more accurate. The current data range is transformed into a new, standardized range using this method. Normalization is extremely important when it comes to bringing disparate prediction and forecasting techniques into harmony. Data normalization improves the consistency and comparability of different predictive models by standardizing the range of independent variables or features within a dataset, leading to more steady and dependable results.

Normalisation, which involves reshaping numerical columns to conform to a standard scale, is essential for datasets with different units or magnitudes across different features. Finding a common scale for the data while maintaining the intrinsic variations in value ranges is the main goal of normalization. This usually entails rescaling the features to a standard range, which is typically between 0 and 1. Alternatively, the features can be adjusted to have a mean of 0 and a standard deviation of 1.

Z-Score Normalisation (Standardisation) and Min-Max Scaling are two commonly used normalisation techniques. In order to enable more insightful and precise analyses in a variety of predictive modelling scenarios, these techniques are essential in bringing disparate features to a comparable scale.

## Data Normalization Techniques

**Min-Max normalization:**

This method of normalising data involves transforming the original data linearly. The data's minimum and maximum values are obtained, and each value is then changed using the formula that follows.

The formula works by subtracting the minimum value from the original value to determine how far the value is from the minimum. Then, it divides this difference by the range of the variable (the difference between the maximum and minimum values). This division scales the variable to a proportion of the entire range. As a result, the normalized value falls between 0 and 1.

- When the feature $X$ is at its minimum, the normalized value ( ) is 0. This is because the numerator becomes zero.
- Conversely, when $X$ is at its maximum, is 1, indicating full-scale normalization.

- For values between the minimum and maximum, ranges between 0 and 1, preserving the relative position of *X* within the original range.

**Normalisation through decimal scaling:**

The data is normalised by shifting the decimal point of its values. By dividing each data value by the maximum absolute value of the data, we can use this technique to normalise the data. The following formula is used to normalise the data value, v, of the data to v':

where is the normalized value, is the original value, and is the smallest integer such that . This formula involves dividing each data value by an appropriate power of 10 to ensure that the resulting normalized values are within a specific range.

**Normalisation of Z-score or Zero Mean (Standardisation):**

Using the mean and standard deviation of the data, values are normalised in this technique to create a standard normal distribution (mean: 0, standard deviation: 1). The equation that is applied is:

**where,**

is the mean of the data A is the standard deviation.

## Data Shuffling

Why Should the Data Be Shuffled for Machine Learning Tasks?

**Shuffling the data helps prevent bias during training, ensures randomness in batch selection, and prevents the model from learning patterns based on the order of the data.**

Shuffling the data is a crucial step in machine learning tasks for several reasons:

1.
    1. **Preventing Bias**: Without shuffling, the model might learn patterns based on the order of the data, leading to biased training and potentially poor generalization to unseen data.

2.
**2 . Randomness in Batch Selection**: When training in mini-batches, shuffling ensures that each batch contains a diverse set of samples from the dataset. This randomness helps the model learn more effectively and prevents it from memorizing specific patterns within a batch.

3. **Improving Generalization**: Shuffling the data helps to ensure that the model generalizes well to unseen data by exposing it to a variety of samples during training. This can lead to better performance on validation and test datasets.
4. **Breaking Patterns**: In some cases, the dataset might have inherent patterns based on the order of samples (e.g., temporal or spatial patterns). Shuffling disrupts these patterns, forcing the model to learn more robust features.
5. **Avoiding Overfitting**: Shuffling helps to mitigate overfitting by preventing the model from memorizing the specific characteristics of the training data, which may not generalize well to new data.

## Overfitting and Underfitting

## Underfitting in Machine Learning

A statistical model or a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data. In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples. It mainly happens when we uses very simple model with overly simplified assumptions. To address underfitting problem of the model, we need to use more complex models, with enhanced feature representation, and less regularization.

**Note: The underfitting model has High bias and low variance.**

## Reasons for Underfitting

1. The model is too simple, So it may be not capable to represent the complexities in the data.
2. The input features which is used to train the model is not the adequate representations of underlying factors
   influencing the target variable.
3. The size of the training dataset used is not enough.
4. Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.
5. Features are not scaled.

## Overfitting in Machine Learning

A statistical model is said to be overfitted when the model does not make accurate predictions
on testing data. When a model gets trained with so much data, it starts learning from the noise
and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes
of overfitting are the non-parametric and non-linear methods because these types of machine
learning algorithms have more freedom in building the model based on the dataset and therefore
they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on
training data is different from unseen data.

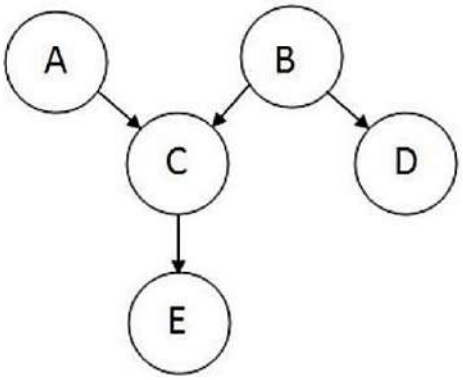
### Reasons for Overfitting:

1. High variance and low bias.
2. The model is too complex.
3. The size of the training data.

# Bayesian Belief Networks (BBN)

- A Bayesian network, or belief network, shows conditional probability and causality relationships between variables.
- The probability of an event occurring given that another event has already occurred is called a conditional probability.
- The probabilistic model is described qualitatively by a **directed acyclic graph, or DAG**. The vertices of the graph, which represent variables, are called nodes.
- The nodes are represented as circles containing the variable name. The connections between the nodes are called arcs, or edges.
- The edges are drawn as arrows between the nodes and represent the dependence between the variables.
- Therefore, any pairs of nodes indicate that one node is the parent of the other so there are no independence assumptions. Independence assumptions are implied in Bayesian networks by the absence of a link.

Here is a sample DAG:




Sample DAG

- The node where the arc originates is called the parent, while the node where the arc ends is called the child. In this case, A is a parent of C, and C is a child of A.
- Nodes that can be reached from other nodes are called descendants. Nodes that lead a path to a specific node are called ancestors. For example, C and E are descendants of A, and A and C are ancestors of E.
- There are no loops in Bayesian networks since no child can be its ancestor or descendent.
- Bayesian networks will generally also include a set of probability tables, stating the probabilities for the true/false values of the variables.
- The main point of Bayesian Networks is to allow for probabilistic inference to be performed. This means that the probability of each value of a node in the Bayesian network can be computed when the values of the other variables are known.

## Joint Probability Distributions

Joint probability is defined as the probability that a series of events will happen concurrently. The joint probability of several variables can be calculated from the product of individual probabilities of the nodes:

$$\mathrm{P}\left(X_1, \ldots, X_n\right) = \prod_{i=1}^{n} \mathrm{P}\left(X_i \mid \text{ parents } \left(X_i\right)\right)$$

**Using the above sample DAG,** the joint probability distribution is:

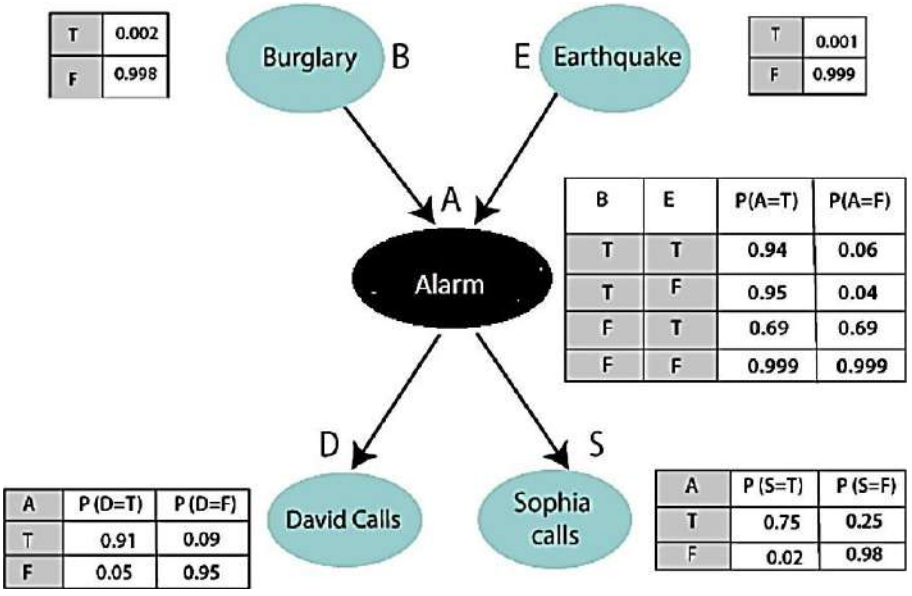
$$P(A, B, C, D, E) = P(A)P(B)P(C \mid A, B)P(D \mid B)P(E \mid C)$$

If a node does not have a parent, like node A, its probability distribution is described as unconditional. Otherwise, the local probability distribution of the node is conditional on other nodes.

# Example 1:

To detect burglary, Harry put a new burglar alarm at his home. The alarm is not only capable of detecting a burglary, but it can also detect mild earthquakes. Harry has two next-door neighbors, David and Sophia, who have agreed to notify Harry at work if they hear the alarm. When David hears the alarm, he always phones Harry, but sometimes he gets confused with the phone ringing and calls at that time as well. Sophia, on the other hand, enjoys listening to loud music and occasionally misses the alarm. We'd like to calculate the likelihood of a burglary alarm in this case.

Calculate the probability that the alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.




| T | 0.002 |
|---|---|
| F | 0.998 |

| T | 0.001 |
|---|---|
| F | 0.999 |

| B | E | P(A=T) | P(A=F) |
|---|---|---|---|
| T | T | 0.94 | 0.06 |
| T | F | 0.95 | 0.04 |
| F | T | 0.69 | 0.69 |
| F | F | 0.999 | 0.999 |

| A | P (D=T) | P (D=F) |
|---|---|---|
| T | 0.91 | 0.09 |
| F | 0.05 | 0.95 |

| A | P (S=T) | P (S=F) |
|---|---|---|
| T | 0.75 | 0.25 |
| F | 0.02 | 0.98 |

**List of all events occurring in this network:**

- Burglary (B)
- Earthquake(E)
- Alarm(A)
- David Calls(D)
- Sophia calls(S)

We can express the occurrences in the problem statement as probability: Using jointprobability distribution, P[D, S, A, B, E] may rewrite the preceding probability statement:

$$P[D, S, A, B, E] = P[D \mid S, A, B, E].\ P[S, A, B, E]$$
$$= P[D \mid S, A, B, E].\ P[S \mid A, B, E].\ P[A, B, E]$$
$$= P[D \mid A].\ P[S \mid A, B, E].\ P[A, B, E]$$
$$= P[D \mid A].\ P[S \mid A].\ P[A \mid B, E].\ P[B, E]$$
$$= P[D \mid A].\ P[S \mid A].\ P[A \mid B, E].\ P[B \mid E].\ P[E]$$

Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary. P(B= False)= 0.998, which is the probability of no burglary.
P(E= True)= 0.001, which is the probability of a minor earthquake
P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

**Conditional probability table for Alarm A:**

The Conditional probability of Alarm A depends on Burglar and earthquake:

| B | E | P(A=True) | P(A=False) |
|---|---|---|---|
| True | True | 0.94 | 0.06 |
| True | False | 0.95 | 0.04 |
| False | True | 0.31 | 0.69 |
| False | False | 0.001 | 0.999 |

**Conditional probability table for David Calls:**

David's conditional probability of calling is determined by the probability of Alarm.

| A | P(D= True) | P(D= False) |
|---|---|---|
| True | 0.91 | 0.09 |
| False | 0.05 | 0.95 |

**Conditional probability table for Sophia Calls:**

Sophia's conditional chance of calling is determined by its Parent Node "Alarm."

| A | P(S= True) | P(S= False) |
|---|---|---|
| True | 0.75 | 0.25 |
| False | 0.02 | 0.98 |

**We may formulate the issue statement in the form of a probability distribution using the joint distribution formula:**

$$P(S, D, A, \neg B, \neg E) = P(S|A) * P(D|A) * P(A|\neg B \wedge \neg E) * P(\neg B) * P(\neg E).$$

= 0.75 * 0.91 * 0.001 * 0.998 * 0.999

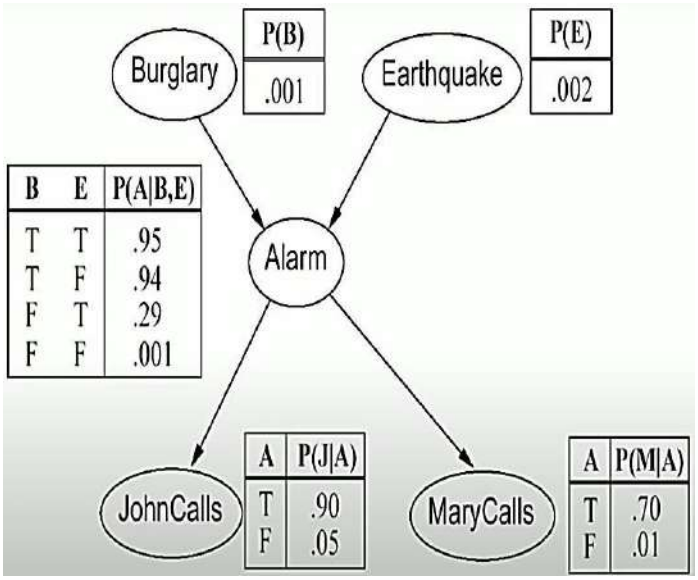
= 0.00068045

Example 2:

**Implement the Bayesian belief network for the following events and probability:**

**(i)** What is the probability that the alarm has sounded but neither a burglary nor an earthquake has occurred, and both John & Mary call?

**(ii)** **What is the probability that John calls?**




Here we need to calculate negations if not given in the problem.

$P(\neg B) = 1 - P(B) = 0.999$, $P(\neg E) = 1 - P(E) = 0.998$

| B | E | P(A\|B,E) | NEGATIONS: P(¬A\|B,E) |
|---|---|---|---|
| T | T | $P(A\|B,E) = 0.95$ | $P(\neg A\|B,E) = 1 - P(A\|B,E) = 1-0.95 = 0.05$ |
| T | F | $P(A\|B,\neg E) = 0.94$ | $P(\neg A\|B,\neg E) = 1 - P(A\|B,\neg E) = 1-0.94 = 0.06$ |
| F | T | $P(A\|\neg B,E) = 0.29$ | $P(\neg A\|\neg B,E) = 1 - P(A\|\neg B,E) = 1-0.29 = 0.71$ |
| F | F | $P(A\|\neg B,\neg E) = 0.01$ | $P(\neg A\|\neg B,\neg E) = 1- P(A\|\neg B,\neg E) = 1-0.01 = 0.99$ |

| A | P(J\|A) | NEGATIONS: P(¬J\|A) |
|---|---|---|
| T | P(J\|A) = 0.90 | P(¬J\|A) = 1 - P(J\|A) = 1-0.90 =0.1 |
| F | P(J\|¬A) = 0.05 | P(¬J\|A) = 1 - P(J\|¬A) = 1-0.05 =0.95 |

| A | P(M\|A) | NEGATIONS: P(¬M\|A) |
|---|---|---|
| T | P(M\|A) = 0.70 | P(¬M\|A) = 1 - P(M\|A) = 1-0.70 =0.3 |
| F | P(M\|¬A) = 0.01 | P(¬M\|¬A) = 1 - P(M\|¬A) = 1-0.01 =0.99 |

What is the probability that the alarm has sounded but neither a Burglary nor an Earthquake has Occurred, and both John & Mary Call?

$$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E) = P(J \mid A)\, P(M \mid A)\, P(A \mid \neg B, \neg E)\, P(\neg B)\, P(\neg E)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$

$$= 0.00062$$

## What is the Probability that John Call?

$$P(J) = P(J \mid A)\, P(A) + P(J \mid \neg A)\, P(\neg A)$$

$$= P(J \mid A)\{P(A|B,E)*P(B,E)+P(A|\neg B,E)*P(\neg B,E)+P(A|B,\neg E)*P(B,\neg E)+P(A|\neg B,\neg E)*P(\neg B,\neg E)\}$$
$$+ P(J \mid \neg A)\{P(\neg A|B,E)*P(B,E)+P(\neg A|\neg B,E)*P(\neg B,E)+P(\neg A|B,\neg E)*P(B,\neg E)+P(\neg A|\neg B,\neg E)*P(\neg B,\neg E)\}$$

$$= 0.90 * 0.00252 + 0.05 * 0.9974$$

$$= 0.002268 + 0.04987$$

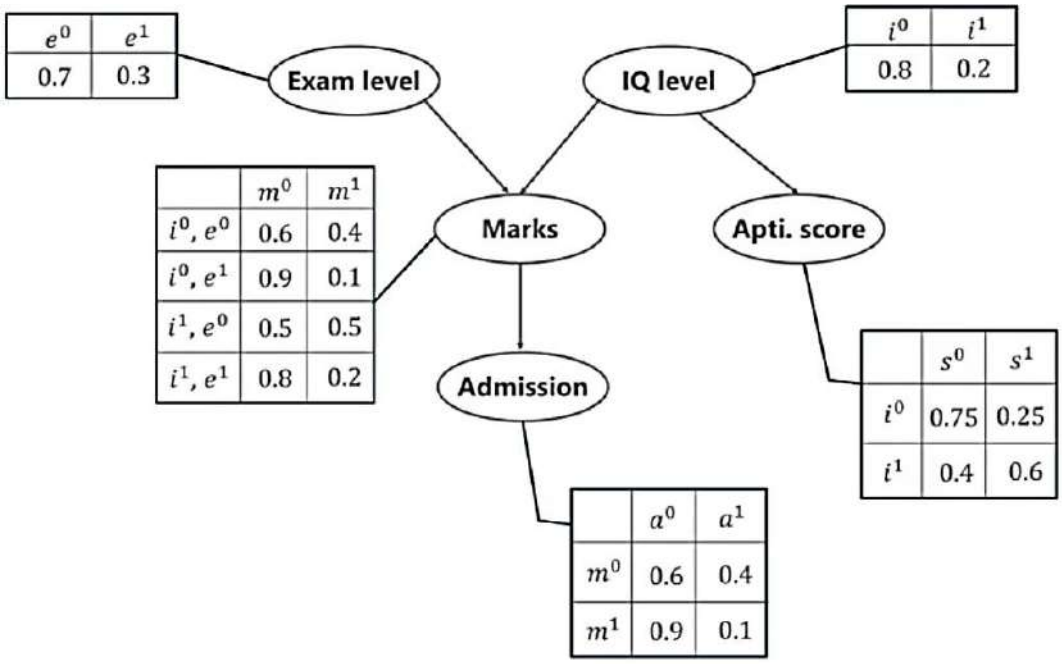
$$= 0.052138$$

# Example 3:

Let us imagine that we are given the task of modeling a student's marks (***m***) for an exam he has just given. From the given Bayesian Network Graph below, we see that the marks depend upon two other variables. They are,

- Exam Level (***e***)– This discrete variable denotes the difficulty of the exam and has two values (0 for easy and 1 for difficult)
- IQ Level (***i***) – This represents the Intelligence Quotient level of the student and is also discrete in nature having two values (0 for low and 1 for high)

Additionally, the IQ level of the student also leads us to another variable, which is the Aptitude Score of the student (***s***). Now, with marks the student has scored, he can secure admission to a particular university. The probability distribution for getting admitted (***a***) to a university is also given below.




| $e^0$ | $e^1$ |
|---|---|
| 0.7 | 0.3 |

| $i^0$ | $i^1$ |
|---|---|
| 0.8 | 0.2 |

| | $m^0$ | $m^1$ |
|---|---|---|
| $i^0, e^0$ | 0.6 | 0.4 |
| $i^0, e^1$ | 0.9 | 0.1 |
| $i^1, e^0$ | 0.5 | 0.5 |
| $i^1, e^1$ | 0.8 | 0.2 |

| | $s^0$ | $s^1$ |
|---|---|---|
| $i^0$ | 0.75 | 0.25 |
| $i^1$ | 0.4 | 0.6 |

| | $a^0$ | $a^1$ |
|---|---|---|
| $m^0$ | 0.6 | 0.4 |
| $m^1$ | 0.9 | 0.1 |

Let us first list all the possible events that are occurring in the above-given table. Exam Level (e)

IQ Level (i) Aptitude Score (s) Marks (m) Admission (a)

These five variables are represented in the form of a Directed Acyclic Graph (DAG) in a Bayesian Network format with their Conditional Probability tables. Now, to calculate the Joint Probability Distribution of the 5 variables the formula is given by,

$$P[a, m, i, e, s] = P(a \mid m) \cdot P(m \mid i, e) \cdot P(i) \cdot P(e) \cdot P(s \mid i)$$

From the above formula,

- P(a | m) denotes the conditional probability of the student getting admission based on the marks he has scored in the examination.
- P(m | i, e) represents the marks that the student will score given his IQ level and difficulty of the Exam Level.
- P(i) and P(e) represent the probability of the IQ Level and the Exam Level.
- P(s | i) is the conditional probability of the student's Aptitude Score, given his IQ Level.

With the following probabilities calculated, we can find the Joint Probability Distribution of the entire Bayesian Network.

## Calculate the probability that the exam level is difficult, the student having a low IQ level and a low Aptitude Score manages to pass the exam and secure admission to the university.

From the above word problem statement, the Joint Probability Distribution can be written as below,

P[a=1, m=1, i=0, e=1, s=0]

From the above Conditional Probability tables, the values for the given conditions are fed to the formula and is calculated as below.

P[a=1, m=1, i=0, e=0, s=0] = P(a=1 | m=1) . P(m=1 | i=0, e=1) . P(i=0) . P(e=1) . P(s=0 | i=0)

= 0.1 * 0.1 * 0.8 * 0.3 * 0.75

= **0.0018**

## Calculate the probability that the student has a High IQ level and Aptitude Score, the exam being easy yet fails to pass and does not secure admission to the university.

From the above word problem statement, the Joint Probability Distribution can be written as below,

P[a=0, m=0, i=1, e=0, s=1]

Thus,

P[a=0, m=0, i=1, e=0, s=1]= P(a=0 | m=0) . P(m=0 | i=1, e=0) . P(i=1) . P(e=0) . P(s=1 | i=1)

= 0.6 * 0.5 * 0.2 * 0.7 * 0.6

= **0.0252**

# Performance Metrics

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.

## Performance Metrics for Classification

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**

### 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:

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

### 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

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

**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:

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

# K-NEAREST NEIGHBOR CLASSIFICATION (K-NN)

- The K-Nearest Neighbors (K-NN) algorithm is a simple and popular machine learning algorithm used mostly for solving classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately. Instead, it stores the dataset and performs an action on it during classification.
- The KNN algorithm, at the training phase, just stores the dataset, and when it gets new data, it classifies that data into a category that is very similar to the new data.

# Steps in K-NN Classification:

Step #1 - Assign a value to K.

Step #2 - Calculate the distance between the new data entry and all other existing data entries. Arrange them in ascending order or use majority voting based on their distance values.

Step #3 - Find the K-nearest neighbors to the new entry based on the calculated distances. Step

#4 - Assign the new data entry to the majority class in the nearest neighbors.

# Example 1:

| Brightness | Saturation | Class |
|---|---|---|
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |

There are two columns — Brightness and Saturation. Each row in the table has a class of either Red or Blue. Let's assume the value of K is 5.

Given the dataset and new test instance, we need to find the distance from the new test instance to every training example. Here we use the Euclidean distance formula to find the distance.

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

Here's the new data entry:

| Brightness | Saturation | Class |
|---|---|---|
| 20 | 35 | ? |

Here's the formula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Where:

- $X_2$ = New entry's brightness (20).
- $X_1$= Existing entry's brightness.
- $Y_2$ = New entry's saturation (35).
- $Y_1$ = Existing entry's saturation.

## Distance #1

For the first row, d1:

| Brightness | Saturation | Class |
|---|---|---|
| 40 | 20 | Red |

$d1 = \sqrt{(20 - 40)^2 + (35 - 20)^2}$

$= \sqrt{400 + 225}$

$= \sqrt{625}$

$= 25$

## Distance #2

For the second row, d2:

| Brightness | Saturation | Class | Distance |
|---|---|---|---|
| 50 | 50 | Blue | ? |

$d2 = \sqrt{(20 - 50)^2 + (35 - 50)^2}$

$= \sqrt{900 + 225}$

$= \sqrt{1125}$

$= 33.54$

### Distance #3

For the third row, d3:

| Brightness | Saturation | Class | Distance |
|---|---|---|---|
| 60 | 90 | Blue | ? |

$d2 = \sqrt{(20 - 60)^2 + (35 - 90)^2}$

$= \sqrt{1600 + 3025}$

$= \sqrt{4625}$

$= 68.01$

Similarly, calculate the distance for the last four rows. Then the table will look like this after all the distances have been calculated:

| Brightness | Saturation | Class | Distance |
|---|---|---|---|
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | 10 |
| 70 | 70 | Blue | 61.03 |
| 60 | 10 | Red | 47.17 |
| 25 | 80 | Blue | 45 |

Rearrange the distances in ascending order:

| Brightness | Saturation | Class | Distance |
|---|---|---|---|
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |
| 70 | 70 | Blue | 61.03 |
| 60 | 90 | Blue | 68.01 |

Since, the value of K=5, we'll only consider the first five rows. That is:

| Brightness | Saturation | Class | Distance |
|---|---|---|---|
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |

The majority class within the 5 nearest neighbors to the new entry is Red. Therefore, we'll classify the new entry as Red.

Here's the updated table:

| Brightness | Saturation | Class |
|---|---|---|
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |
| 20 | 35 | Red |

# Example 2:

Apply K-NN classification to predict the Sugar of Diabetic Patient given BMI and Age. Assume K=3.

## Test Example BMI=43.6, Age=40, Sugar=?

| BMI | Age | Sugar |
|---|---|---|
| 33.6 | 50 | 1 |
| 26.6 | 30 | O |
| 23.4 | 40 | O |
| 43.1 | 67 | O |
| 35.3 | 23 | 1 |
| 35.9 | 67 | 1 |
| 36.7 | 45 | 1 |
| 25.7 | 46 | O |
| 23.3 | 29 | O |
| 31 | 56 | 1 |

**Solution:**

The following table consists of the calculated distances from the text example to training instances.

| BMI | Age | Sugar | Formula | Distance |
|---|---|---|---|---|
| 33.6 | 50 | 1 | √((43.6-33.6)^2+(40-50)^2 ) | 14.14 |
| 26.6 | 30 | O | √((43.6-26.6)^2+(40-30)^2 ) | 19.72 |
| 23.4 | 40 | O | √((43.6-23.4)^2+(40-40)^2 ) | 20.20 |
| 43.1 | 67 | O | √((43.6-43.1)^2+(40-67)^2 ) | 27.00 |
| 35.3 | 23 | 1 | √((43.6-35.3)^2+(40-23)^2 ) | 18.92 |
| 35.9 | 67 | 1 | √((43.6-35.9)^2+(40-67)^2 ) | 28.08 |
| 36.7 | 45 | 1 | √((43.6-36.7)^2+(40-45)^2 ) | 8.52 |
| 25.7 | 46 | O | √((43.6-25.7)^2+(40-46)^2 ) | 18.88 |
| 23.3 | 29 | O | √((43.6-23.3)^2+(40-29)^2 ) | 23.09 |
| 31 | 56 | 1 | √((43.6-31)^2+(40-56)^2 ) | 20.37 |

The next step is to find the nearest neighbors based on the value of k. In this case, the value of k is 3. Hence, we need to find 3 nearest neighbors.

| BMI | Age | Sugar | Distance | Rank |
|---|---|---|---|---|
| 33.6 | 50 | 1 | 14.14 | 2 |
| 26.6 | 30 | O | 19.72 | |
| 23.4 | 40 | O | 20.20 | |
| 43.1 | 67 | O | 27.00 | |
| 35.3 | 23 | 1 | 18.92 | |
| 35.9 | 67 | 1 | 28.08 | |
| 36.7 | 45 | 1 | 8.52 | 1 |
| 25.7 | 46 | O | 18.88 | 3 |
| 23.3 | 29 | O | 23.09 | |
| 31 | 56 | 1 | 20.37 | |

Now, we need to apply the majority voting technique to decide the resulting label for the new example.

Here the 1st and 2nd nearest neighbors have target label 1 and the 3rd nearest neighbor has target label 0.

Target label 1 has the majority. Hence the new example is classified as 1, That is the diabetic patient has Sugar.

Therefore, the Test Example BMI=43.6, Age=40, **Sugar=1**

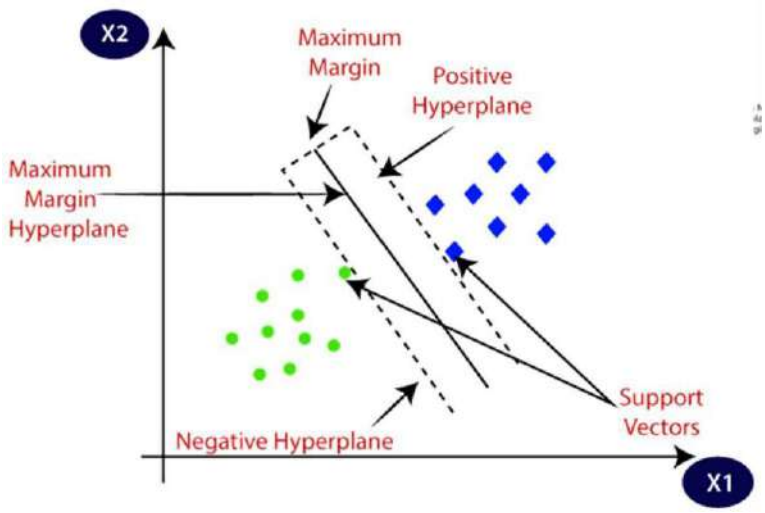
# SUPPORT VECTOR MACHINE

## Support Vector Machine Algorithm

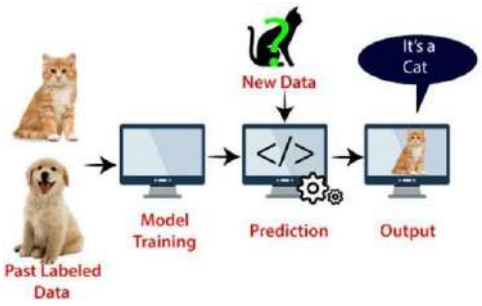
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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.

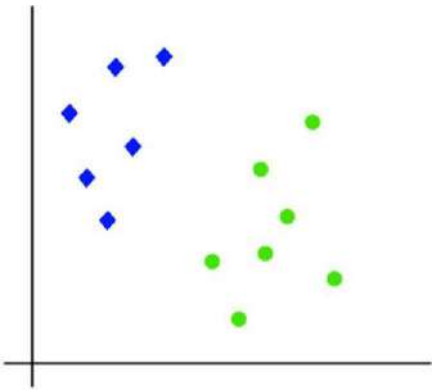
**Support Vectors:**

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.

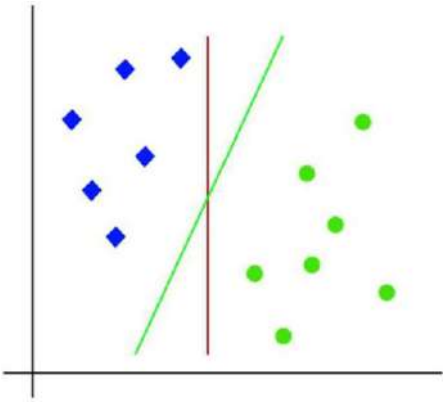
## How does SVM works?

**Linear SVM:**

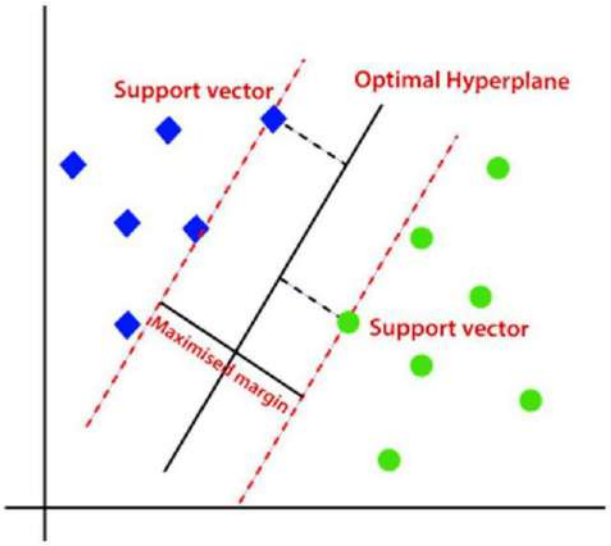
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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:

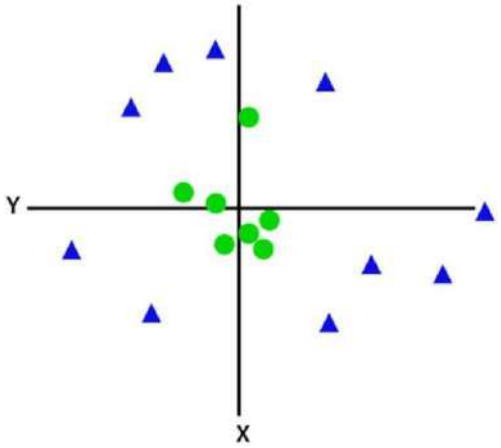


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane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

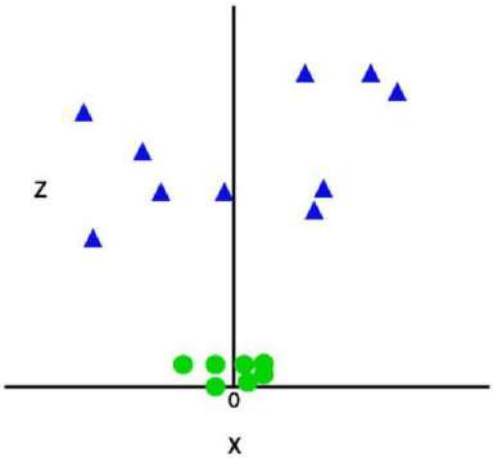
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



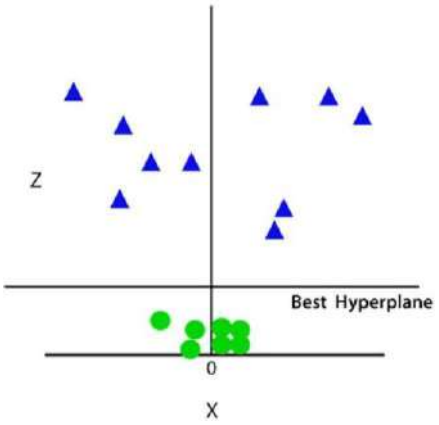

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

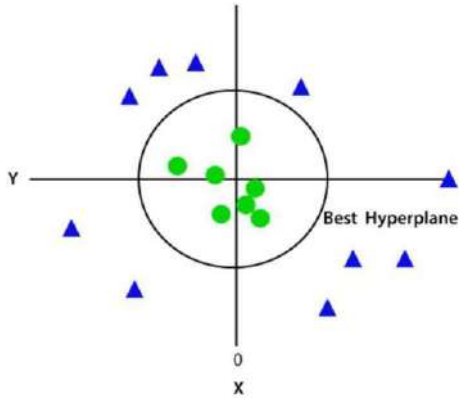
By adding the third dimension, the sample space will become as below image:




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



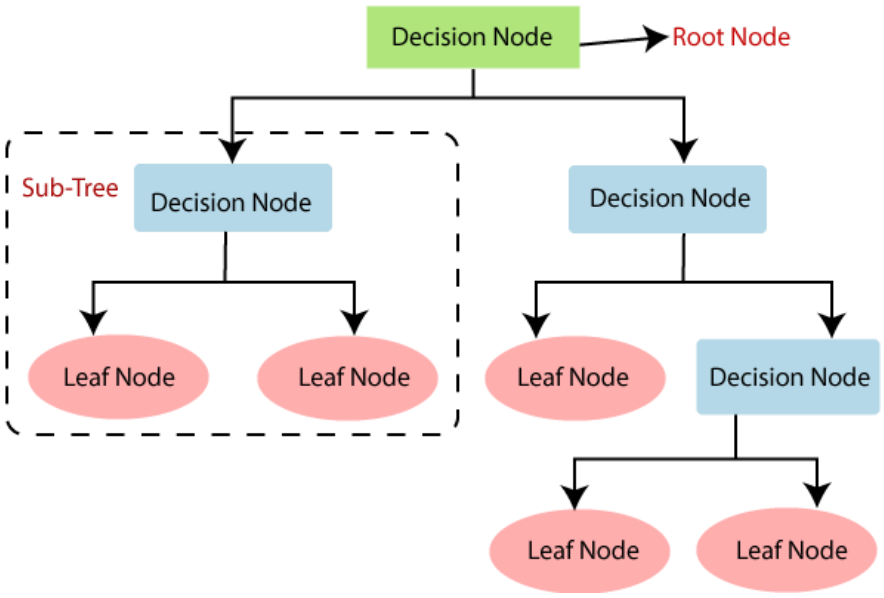

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:




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

## 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:




### 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.

### Decision Tree 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.

#### 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:

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**

| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

## Attribute: Outlook

$Values\ (Outlook) = Sunny, Overcast, Rain$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Sunny} \leftarrow [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$

$S_{Overcast} \leftarrow [4+, 0-]$ $\quad Entropy(S_{Overcast}) = -\frac{4}{4}log_2\frac{4}{4} - \frac{0}{4}log_2\frac{0}{4} = 0$

$S_{Rain} \leftarrow [3+, 2-]$ $\quad Entropy(S_{Rain}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$

$$Gain\ (S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Outlook) = Entropy(S) - \frac{5}{14}Entropy(S_{Sunny}) - \frac{4}{14}Entropy(S_{Overcast}) - \frac{5}{14}Entropy(S_{Rain})$$

$$Gain(S, Outlook) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.2464$$



| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

## Attribute: Temp

$Values\ (Temp) = Hot, Mild, Cool$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Hot} \leftarrow [2+, 2-]$ $\quad Entropy(S_{Hot}) = -\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4} = 1.0$

$S_{Mild} \leftarrow [4+, 2-]$ $\quad Entropy(S_{Mild}) = -\frac{4}{6}log_2\frac{4}{6} - \frac{2}{6}log_2\frac{2}{6} = 0.9183$

$S_{Cool} \leftarrow [3+, 1-]$ $\quad Entropy(S_{Cool}) = -\frac{3}{4}log_2\frac{3}{4} - \frac{1}{4}log_2\frac{1}{4} = 0.8113$

$$Gain\ (S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Temp) = Entropy(S) - \frac{4}{14}Entropy(S_{Hot}) - \frac{6}{14}Entropy(S_{Mild}) - \frac{4}{14}Entropy(S_{Cool})$$

$$Gain(S, Temp) = 0.94 - \frac{4}{14}1.0 - \frac{6}{14}0.9183 - \frac{4}{14}0.8113 = 0.0289$$

| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

## Attribute: Humidity

$$Values\ (Humidity) = High, Normal$$

$$S = [9+, 5-] \qquad Entropy(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.94$$

$$S_{High} \leftarrow [3+, 4-] \qquad Entropy(S_{High}) = -\frac{3}{7} log_2 \frac{3}{7} - \frac{4}{7} log_2 \frac{4}{7} = 0.9852$$

$$S_{Normal} \leftarrow [6+, 1-] \qquad Entropy(S_{Normal}) = -\frac{6}{7} log_2 \frac{6}{7} - \frac{1}{7} log_2 \frac{1}{7} = 0.5916$$

$$Gain\ (S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Humidity)$$

$$= Entropy(S) - \frac{7}{14} Entropy(S_{High}) - \frac{7}{14} Entropy(S_{Normal})$$

$$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14} 0.5916 = 0.1516$$



| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

## Attribute: Wind

$$Values\ (Wind) = Strong, Weak$$

$$S = [9+, 5-] \qquad Entropy(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.94$$

$$S_{Strong} \leftarrow [3+, 3-] \qquad Entropy(S_{Strong}) = 1.0$$

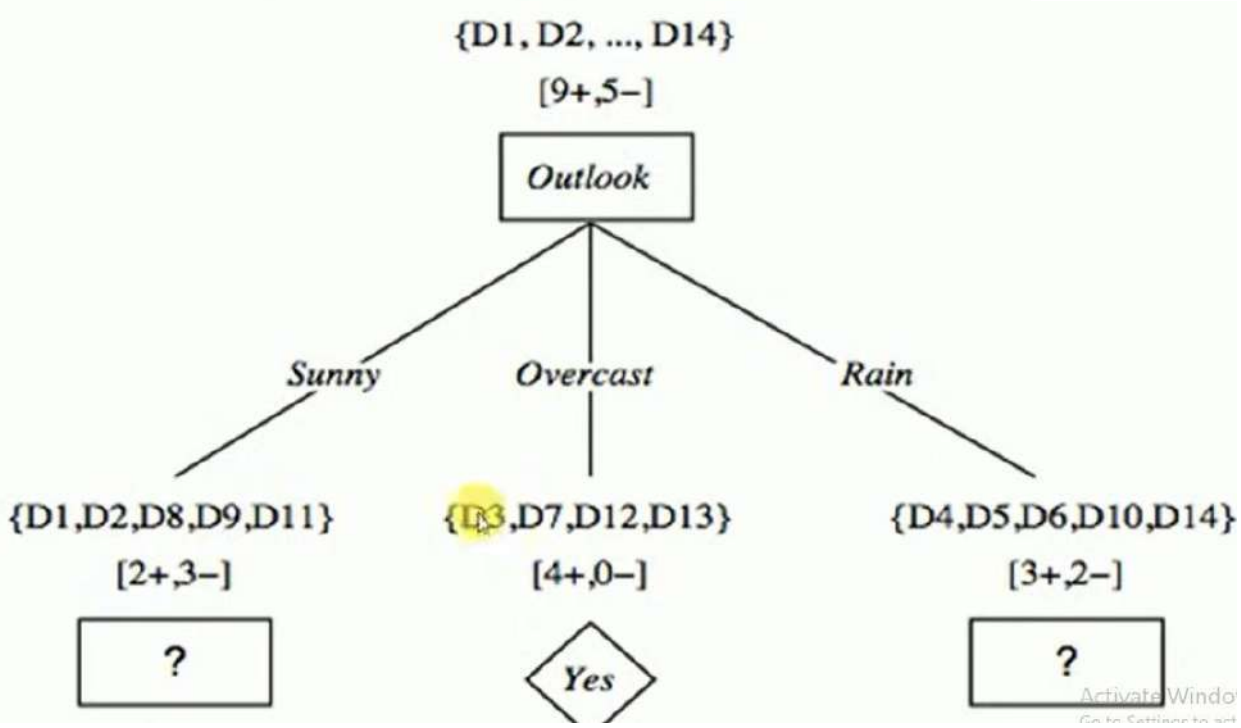
$$S_{Weak} \leftarrow [6+, 2-] \qquad Entropy(S_{Weak}) = -\frac{6}{8} log_2 \frac{6}{8} - \frac{2}{8} log_2 \frac{2}{8} = 0.8113$$

$$Gain\ (S, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Wind) = Entropy(S) - \frac{6}{14} Entropy(S_{Strong}) - \frac{8}{14} Entropy(S_{Weak})$$

$$Gain(S, Wind) = 0.94 - \frac{6}{14} 1.0 - \frac{8}{14} 0.8113 = 0.0478$$

| Day | Outlook | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |



| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D1 | Hot | High | Weak | No |
| D2 | Hot | High | Strong | No |
| D8 | Mild | High | Weak | No |
| D9 | Cool | Normal | Weak | Yes |
| D11 | Mild | Normal | Strong | Yes |

## Attribute: Temp

$Values\ (Temp) = Hot, Mild, Cool$

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 2-]$ $\quad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Mild}) = 1.0$

$S_{Cool} \leftarrow [1+, 0-]$ $\quad Entropy(S_{Cool}) = 0.0$

$$Gain\ (S_{Sunny}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Temp)$$

$$= Entropy(S) - \frac{2}{5}Entropy(S_{Hot}) - \frac{2}{5}Entropy(S_{Mild})$$

$$- \frac{1}{5}Entropy(S_{Cool})$$

$$Gain(S_{sunny}, Temp) = 0.97 - \frac{2}{5}0.0 - \frac{2}{5}1 - \frac{1}{5}0.0 = 0.570$$



| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| Dl | Hot | High | Weak | No |
| D2 | Hot | High | Strong | No |
| D8 | Mild | High | Weak | No |
| D9 | Cool | Normal | Weak | Yes |
| Dl1 | Mild | Normal | Strong | Yes |

## Attribute: Humidity

$Values\ (Humidity) = High, Normal$

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{high} \leftarrow [0+, 3-]$ $\quad Entropy(S_{High}) = 0.0$

$S_{Normal} \leftarrow [2+, 0-]$ $\quad Entropy(S_{Normal}) = 0.0$

$$Gain\ (S_{Sunny}, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Humidity) = Entropy(S) - \frac{3}{5}Entropy(S_{High}) - \frac{2}{5}Entropy(S_{Normal})$$

$$Gain(S_{sunny}, Humidity) = 0.97 - \frac{3}{5}0.0 - \frac{2}{5}0.0 = 0.97$$

| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| Dl | Hot | High | Weak | No |
| D2 | Hot | High | Strong | No |
| D8 | Mild | High | Weak | No |
| D9 | Cool | Normal | Weak | Yes |
| Dl1 | Mild | Normal | Strong | Yes |

**Attribute: Wind**

$Values\,(Wind) = Strong, Weak$

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{Strong} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Strong}) = 1.0$

$S_{Weak} \leftarrow [1+, 2-]$ $\quad Entropy(S_{Weak}) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$

$$Gain\,(S_{Sunny}, Wind) = Entropy(S) - \sum_{v\in\{Strong,Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

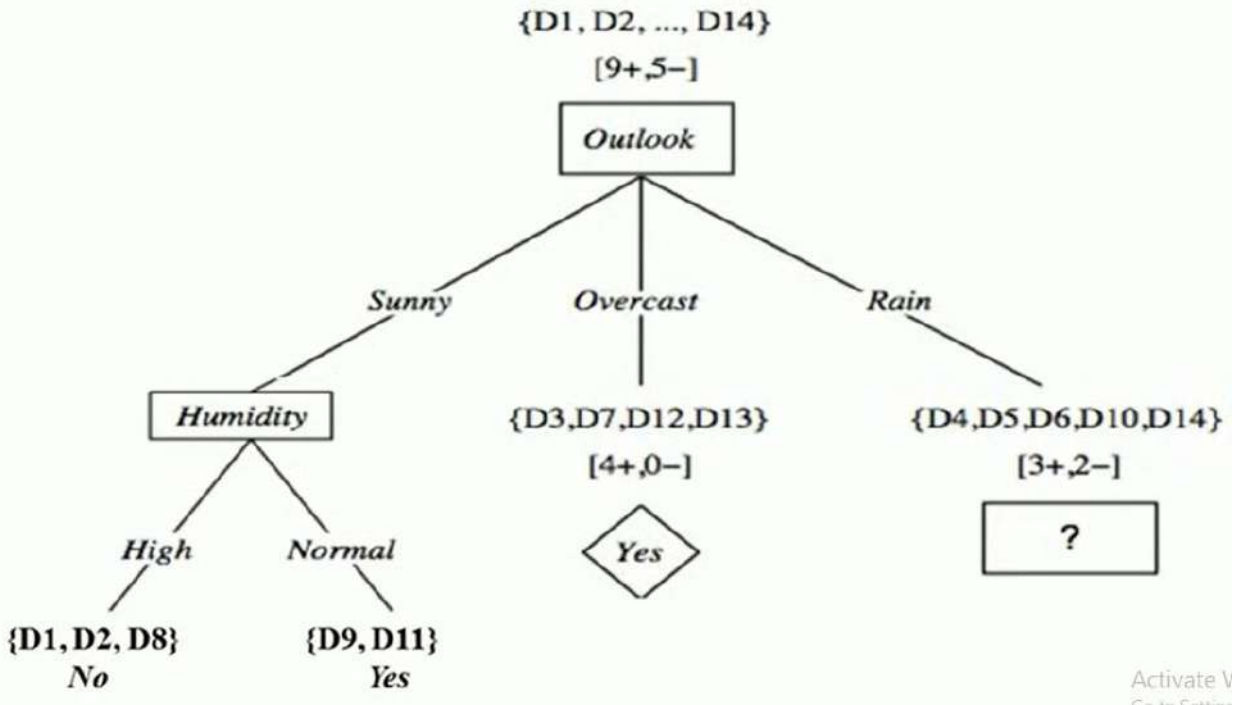
$$Gain(S_{sunny}, Wind) = 0.97 - \frac{2}{5}1.0 - \frac{3}{5}\,0.918 = 0.0192$$

| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D1 | Hot | High | Weak | No |
| D2 | Hot | High | Strong | No |
| D8 | Mild | High | Weak | No |
| D9 | Cool | Normal | Weak | Yes |
| D11 | Mild | Normal | Strong | Yes |

$$Gain(S_{sunny}, Temp) = 0.570$$

$$Gain(S_{sunny}, Humidity) = 0.97$$

$$Gain(S_{sunny}, Wind) = 0.0192$$



| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D4 | Mild | High | Weak | Yes |
| D5 | Cool | Normal | Weak | Yes |
| D6 | Cool | Normal | Strong | No |
| D10 | Mild | Normal | Weak | Yes |
| D14 | Mild | High | Strong | No |

## Attribute: Temp

$Values\ (Temp) = Hot, Mild, Cool$

$S_{Rain} = [3+, 2-]$ $\quad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 0-]$ $\quad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [2+, 1-]$ $\quad Entropy(S_{Mild}) = -\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.9183$

$S_{Cool} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Cool}) = 1.0$

$$Gain\ (S_{Rain}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Temp)$$

$$= Entropy(S) - \frac{0}{5}Entropy(S_{Hot}) - \frac{3}{5}Entropy(S_{Mild})$$

$$-\frac{2}{5}Entropy(S_{Cool})$$

$$Gain(S_{Rain}, Temp) = 0.97 - \frac{0}{5}0.0 - \frac{3}{5}0.918 - \frac{2}{5}1.0 = 0.0192$$



| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D4 | Mild | High | Weak | Yes |
| D5 | Cool | Normal | Weak | Yes |
| D6 | Cool | Normal | Strong | No |
| Dl0 | Mild | Normal | Weak | Yes |
| Dl4 | Mild | High | Strong | No |

## Attribute: Humidity

$Values\ (Humidity) = High, Normal$

$S_{Rain} = [3+, 2-]$ $\quad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$

$S_{High} \leftarrow [1+, 1-]$ $\quad Entropy(S_{High}) = 1.0$

$S_{Normal} \leftarrow [2+, 1-]$ $\quad Entropy(S_{Normal}) = -\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.9183$

$$Gain\ (S_{Rain}, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Humidity) = Entropy(S) - \frac{2}{5}Entropy(S_{High}) - \frac{3}{5}Entropy(S_{Normal})$$

$$Gain(S_{Rain}, Humidity) = 0.97 - \frac{2}{5}1.0 - \frac{3}{5}0.918 = 0.0192$$

| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D4 | Mild | High | Weak | Yes |
| D5 | Cool | Normal | Weak | Yes |
| D6 | Cool | Normal | Strong | No |
| Dl0 | Mild | Normal | Weak | Yes |
| Dl4 | Mild | High | Strong | No |

## Attribute: Wind

$Values\ (wind) = Strong, Weak$

$S_{Rain} = [3+, 2-]$ $\quad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$

$S_{Strong} \leftarrow [0+, 2-]$ $\quad Entropy(S_{Strong}) = 0.0$

$S_{Weak} \leftarrow [3+, 0-]$ $\quad Entropy(S_{weak}) = 0.0$

$$Gain\ (S_{Rain}, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

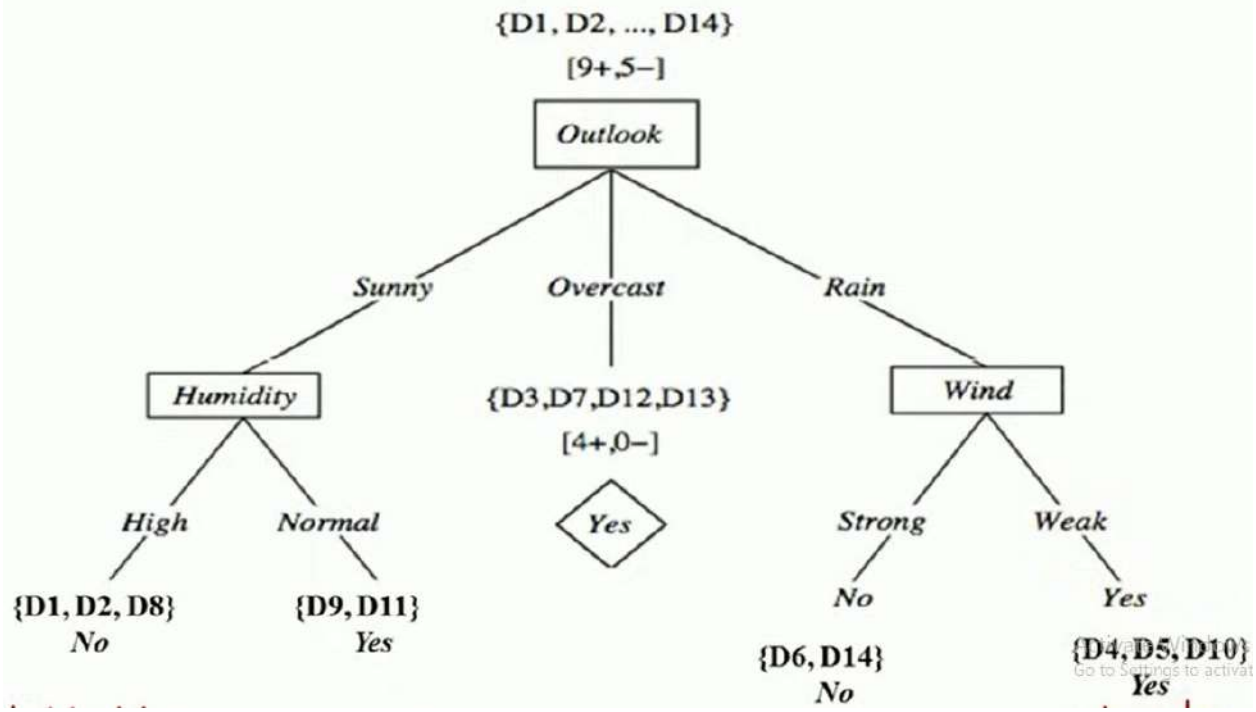
$$Gain(S_{Rain}, Wind) = 0.97 - \frac{2}{5}0.0 - \frac{3}{5}0.0 = 0.97$$

| Day | Temp | Humidity | Wind | Play Tennis |
|---|---|---|---|---|
| D4 | Mild | High | Weak | Yes |
| D5 | Cool | Normal | Weak | Yes |
| D6 | Cool | Normal | Strong | No |
| Dl0 | Mild | Normal | Weak | Yes |
| Dl4 | Mild | High | Strong | No |

$$Gain(S_{Rain}, Temp) = 0.0192$$

$$Gain(S_{Rain}, Humidity) = 0.0192$$

$$Gain(S_{Rain}, Wind) = 0.97$$




# ENSEMBLE LEARNING

Ensemble learning is a machine learning paradigm that proposes to use multiple models to create a stronger model. The fundamental idea behind ensemble learning is that a group of models when working together, can outperform any individual model.

## Ensemble learning techniques

In ensemble learning, multiple techniques can be used for various cases.

## Bagging (Bootstrap Aggregating)

Bagging involves training multiple instances of the same learning algorithm on different subsets of

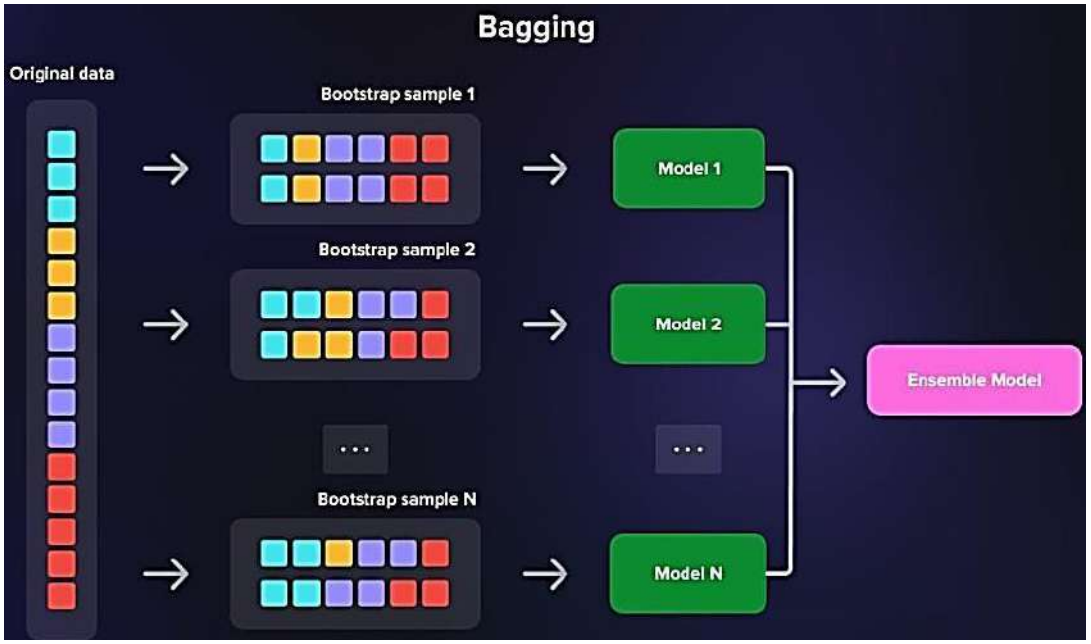
the training data.

The dataset is divided into smaller samples, and then each model is exposed to a random sample of the dataset, and predictions are averaged to produce the final output.

Imagine that you have a large dataset that you divide into smaller portions and feed into several similar models at the same time. Each model produces a certain output which is either voted upon for classification tasks or calculated as an average for regression.

To understand how classification works, imagine that you need to put oranges and apples into different baskets.

| | Model 1 | Model 2 | Model 3 | Model output |
|---|---|---|---|---|
| Sample 1 | apple | orange | apple | apple |
| Sample 2 | orange | orange | orange | orange |
| Sample 3 | apple | ? | apple | apple |



Bagging is helpful when you have a high-variance model and want to reduce overfitting.

### Example tasks:

- Decision tree-based models in high-dimensional data such as random forest for classification or regression.
- Ensemble of neural networks for image classification.

# Boosting

Boosting focuses on improving the performance of a weak learner over iterations. Weak learners are models that perform slightly better than random chance.

To improve their performance, ML engineers can use algorithms like AdaBoost and Gradient Boosting, where each model is trained sequentially.

A subset of training data is fed into the model during training. The model provides false predictions. The sample is then fed to a different weak learner, and the process is repeated for several more iterations. Then the model outputs the overall prediction based on the previous iterations.

The emphasis is on correcting the errors made during the previous steps. The final prediction is a weighted sum of the individual predictions, where models that perform well are given more weight.

Imagine the same classification task with fruit baskets.

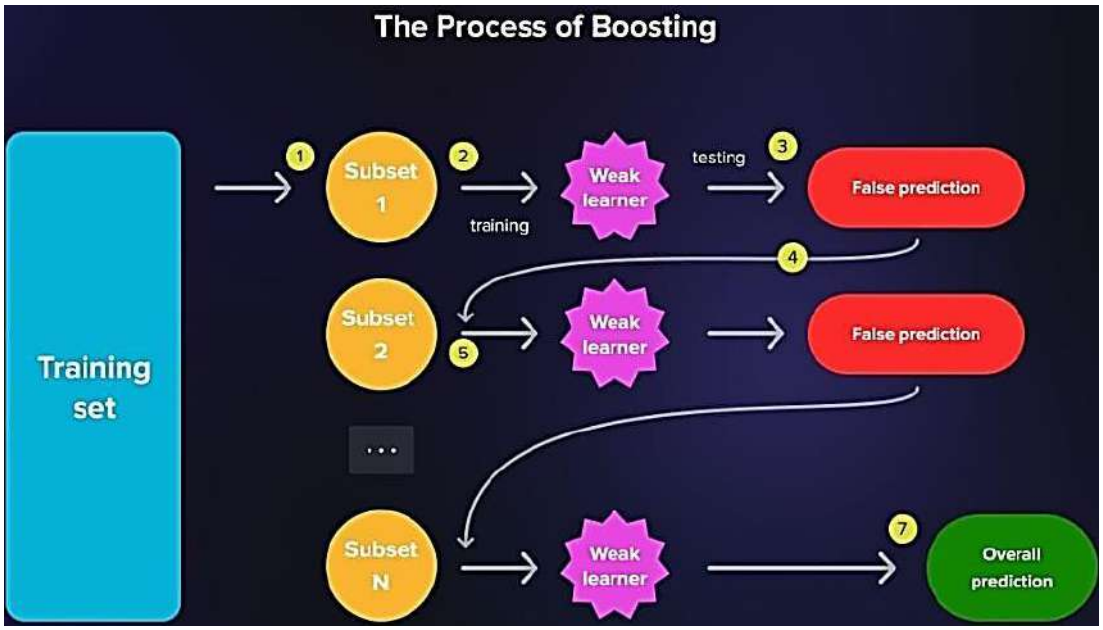
| | Model 1 | Model 2 | Model 3 | Model output |
|---|---|---|---|---|
| Sample 1 | false | true | true | true |
| Sample 1 | false | false | true | true |
| Sample 1 | false | false | true | true |

Boosting is used when you have a weak base model and you want to convert it into a strong model.

### Example tasks:

- Classification problems where the goal is to improve accuracy, such as in fraud detection or spam filtering.

Regression problems where you want to predict a continuous variable such as predicting house prices.

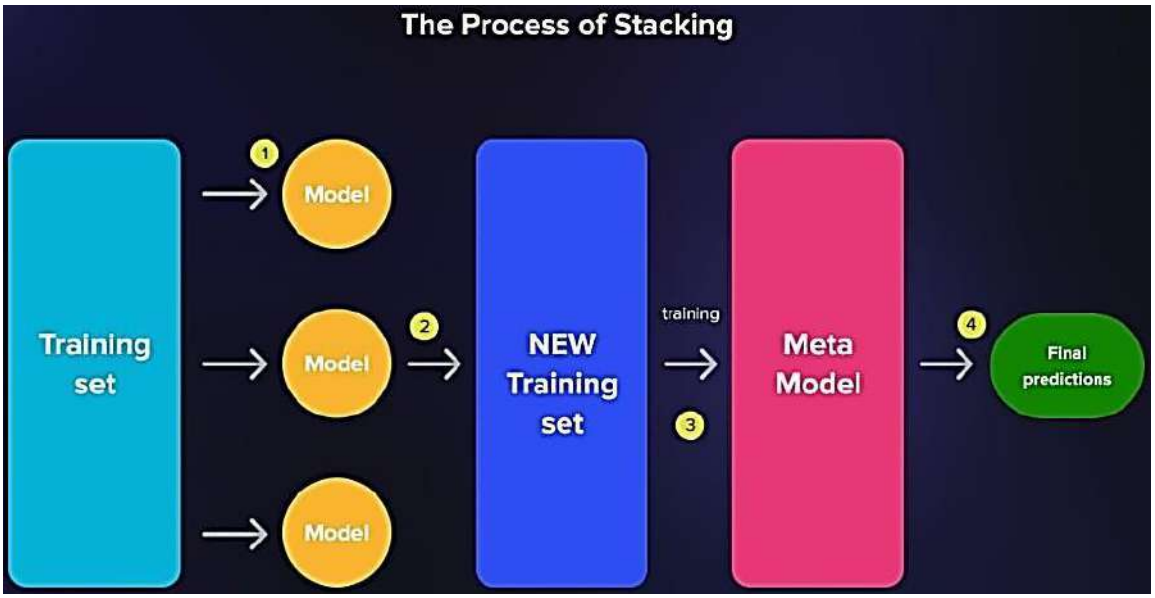



# Stacking

Stacking uses a meta-learner, which is another model that takes the outputs of the base models as input.

This method is usually used when different models in the ensemble have different skills, and their errors are uncorrelated.

The meta-learner then makes the final prediction based on these combined outputs. Stacking is effective when different base models specialize in capturing different aspects of the underlying data patterns.




Stacking is used when you want to combine the predictions of multiple models, capture the strengths of individual models, and improve overall performance.

## Example tasks:

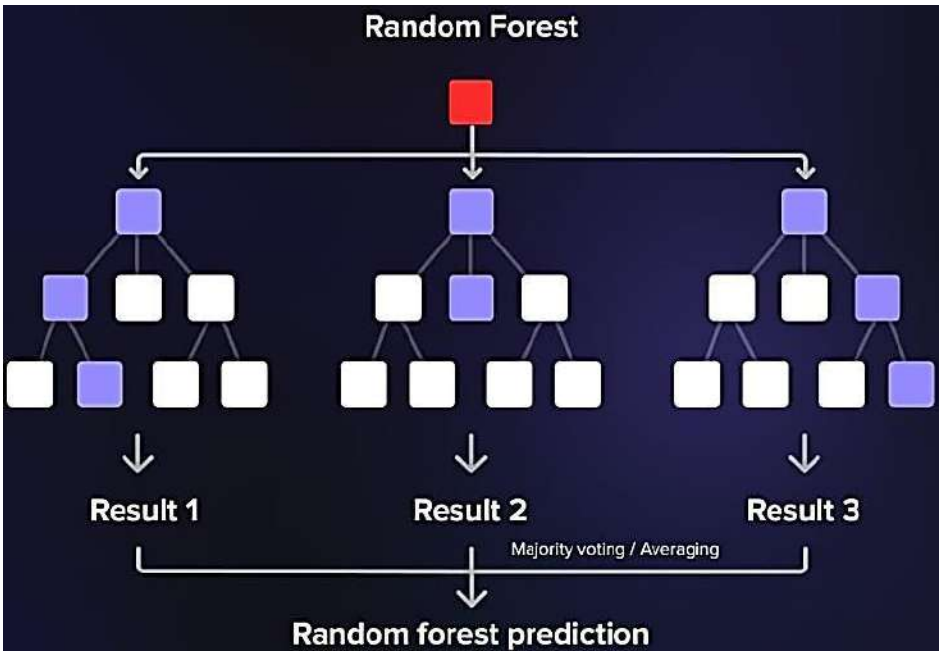
- Any complex problem where different types of models might excel in different aspects such as combining a decision tree model with a neural network for a diverse set of features.
- Multi-modal tasks where different models handle different types of data (text, images, etc.).

# Types of Ensemble Learning Algorithms

Ensemble learning is often used for classification and regression tasks. For each task, there are many algorithms to choose from, depending on the type of the problem, the dataset size, and the desired trade-offs between interpretability, speed, and accuracy.

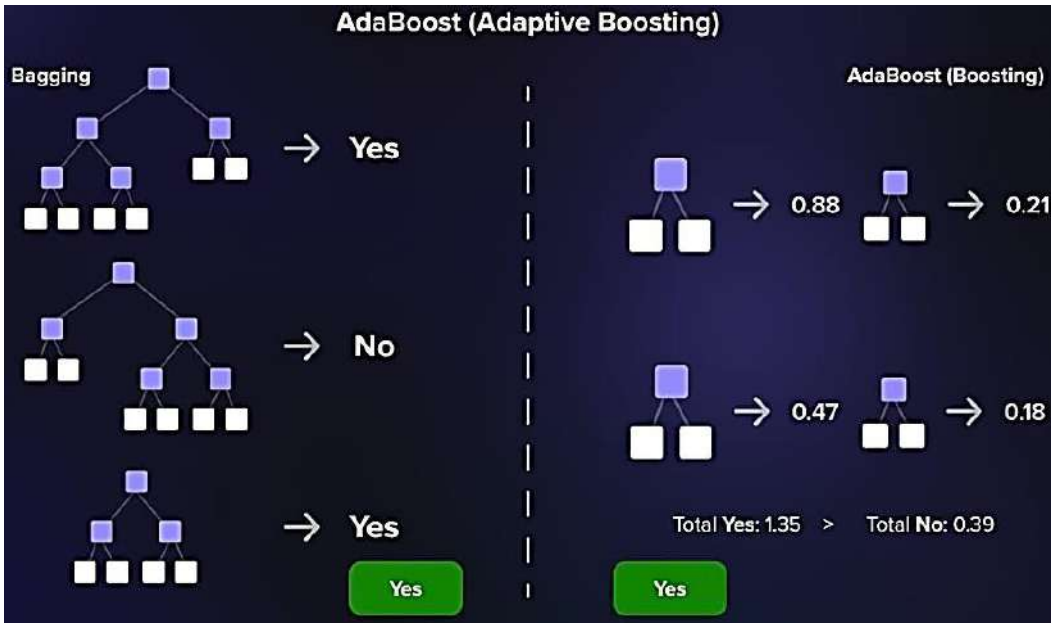
## Random forest

Random forest builds multiple decision trees during training and merges their predictions. Each tree is trained on a random subset of the training data, and the final prediction is determined by a majority vote. It can be used for classification, regression, and other tasks.



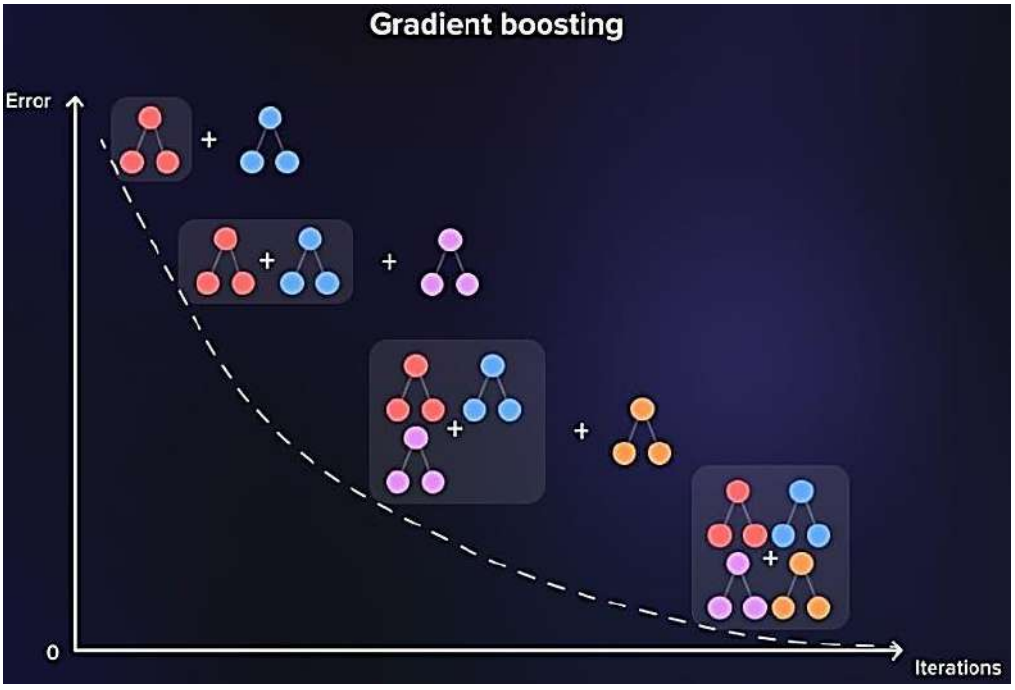
# AdaBoost (Adaptive Boosting)

AdaBoost assigns weights to misclassified instances and focuses on training subsequent models to correct these errors. The final prediction is a weighted sum of the individual model predictions. It is used to solve the problem of weak learners for regression problems.




# Gradient boosting

Gradient boosting builds trees sequentially, with each tree correcting the errors of the previous one. It minimizes a loss function, often using gradient descent, to find the optimal weights for each tree.



It's also often used to solve the problem of weak learners and is used for both regression and classification tasks.

# XGBoost (Extreme Gradient Boosting)

XGBoost is a supervised learning technique that uses an ensemble approach based on the gradient-boosting algorithm.

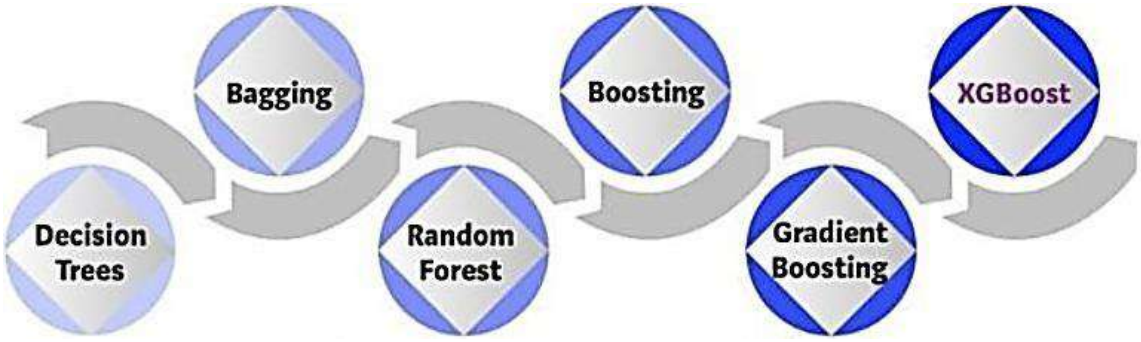
XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed.

With XGBoost, trees are built in parallel, instead of sequentially like Gradient Boosting Decision Trees. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

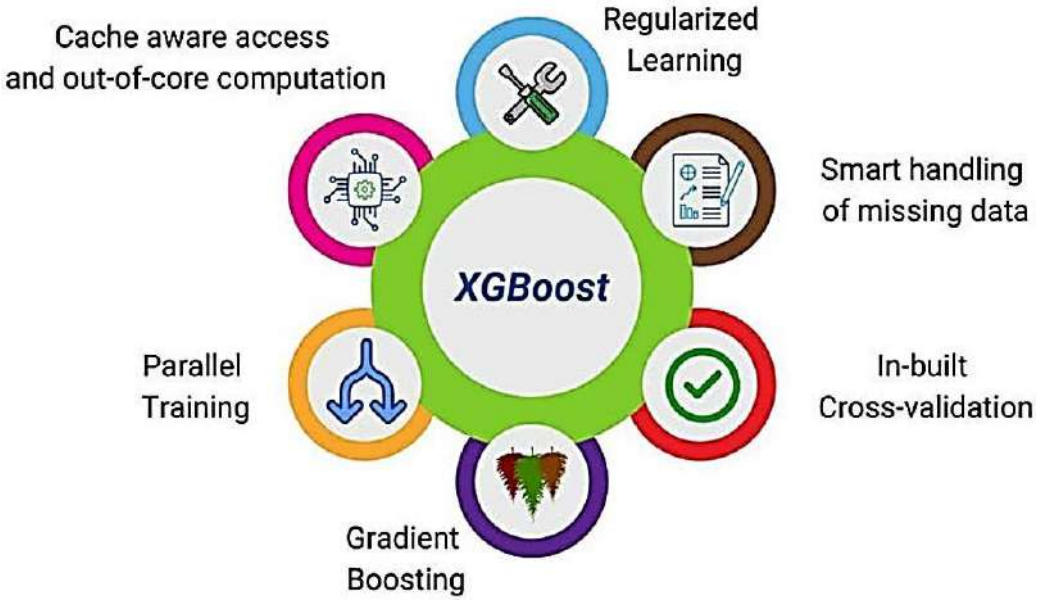
XGBoost has been integrated with a wide variety of other tools and packages such as scikit-learn for Python enthusiasts and caret for R users. In addition, XGBoost is integrated with distributed processing frameworks like Apache Spark and Dask.

It has been used by data scientists and researchers worldwide to optimize their machine- learning models.

**Evolution of Tree-based Algorithms**



**Features Supported by XGBoost**




**Bagging Meta-Estimator**

BaggingClassifier and BaggingRegressor provide a general framework for bagging. They can be used with various base classifiers or regressors.

# VotingClassifier

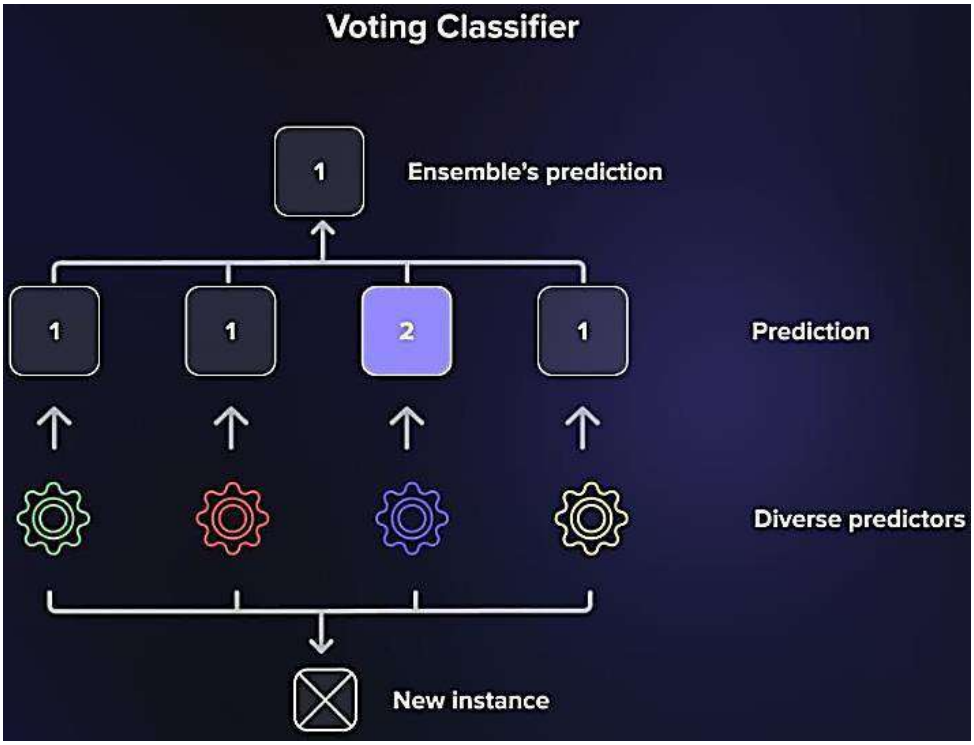
VotingClassifier allows combining multiple classifiers (e.g., SVM, random forest, logistic regression) by majority voting. It supports soft and hard voting strategies.

# Soft and hard voting

Ensemble learning models perform a voting to determine the overall outcome. There are two voting strategies: soft and hard.

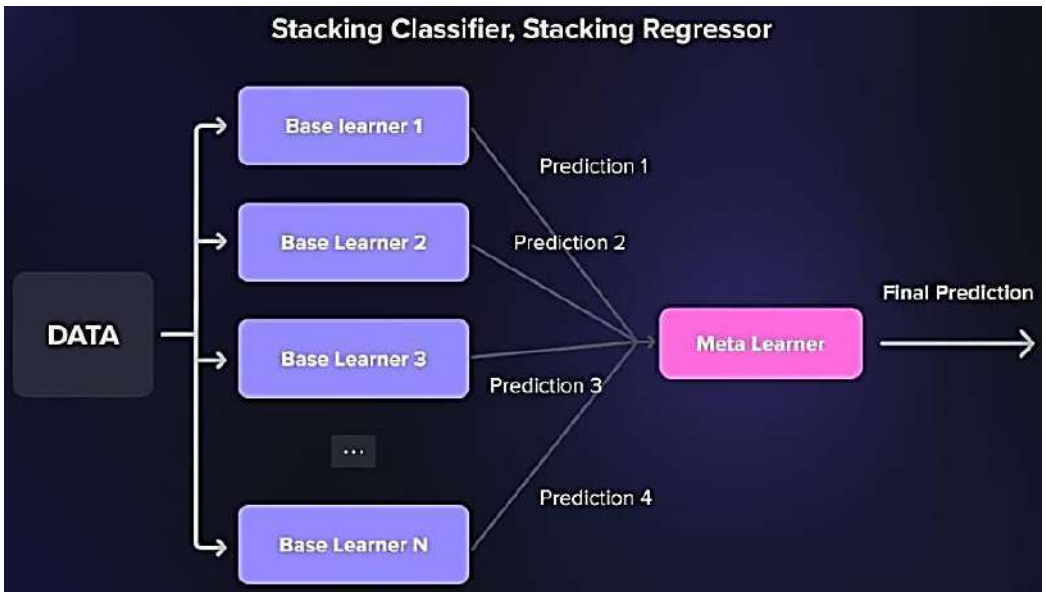
- In **hard voting**, each model in the ensemble makes a prediction, and the final prediction is determined by a majority vote.
- In **soft voting**, each model in the ensemble provides a probability estimate for each class. The final prediction is then based on the average of these probabilities or values.

Hard voting is usually used to predict class labels. Soft voting can be applied when the base models provide probability estimates or confidence scores.



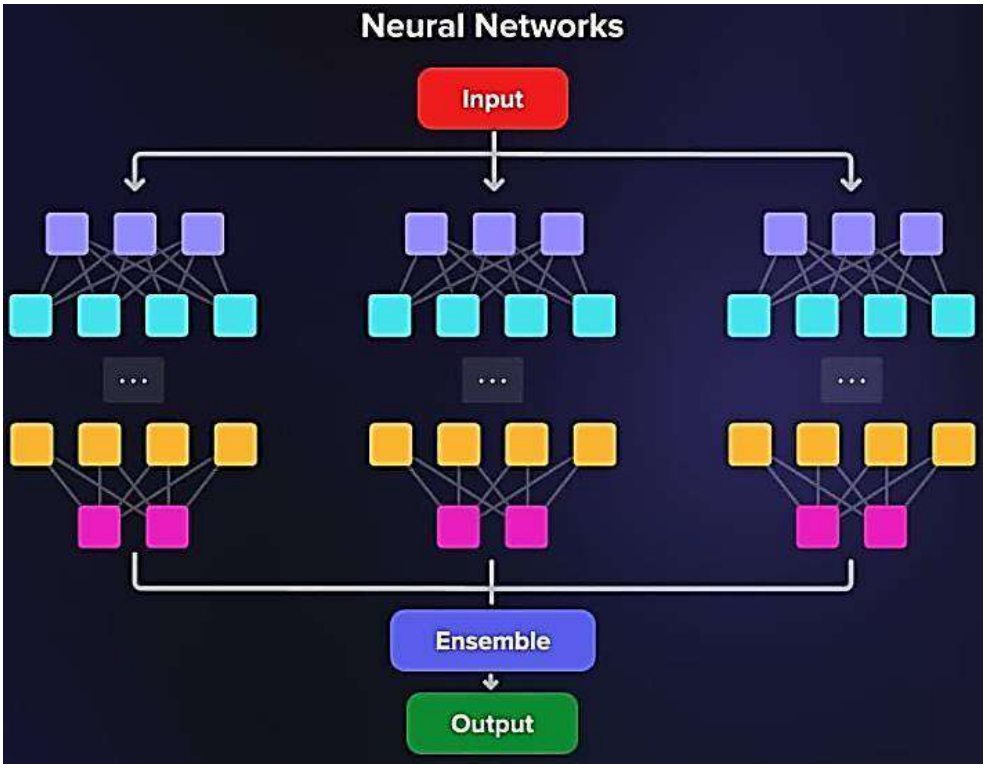
# StackingClassifier, StackingRegressor

StackingClassifier and StackingRegressor enable stacking with multiple base learners and a meta-learner. They provide a flexible way to experiment with stacking in a scikit- learn compatible interface.




# Neural networks

Neural networks can also be used in an ensemble. Bagging can be used to train multiple networks independently and combine their predictions.



## Advantages of ensemble learning

Ensemble learning offers several advantages that contribute to its popularity in the machine learning community.

1. **Increased accuracy**
   Ensemble learning can enhance predictive accuracy. By combining the strengths of multiple models, ensemble methods can often outperform individual models.
2. **Improved robustness**
   Ensemble learning tends to be more robust to outliers and noisy data. Outliers that significantly affect one model's performance might have less influence on the overall ensemble prediction. The diversity among models helps mitigate the impact of individual model errors.
3. **Generalization to new data**
   Ensemble models often generalize well to new and unseen data. The combination of diverse models helps capture complex patterns in the data.
4. **Versatility across tasks**
   Ensemble learning techniques can be applied to various machine learning tasks, including classification, regression, and even unsupervised learning. They are adaptable to different types of models and datasets.

## Disadvantages of ensemble learning

However, engineers also must deal with some disadvantages and trade-offs when using ensemble learning.

1. **Computational complexity**
   Ensemble methods can be computationally expensive, especially when dealing with many models or complex base learners. Training and maintaining multiple models require more resources.
2. **Increased model complexity**
   The combination of multiple models can increase model complexity, making it more challenging to interpret and understand the inner workings of the ensemble.
3. **Overfitting risk**
   Although ensemble methods can reduce overfitting in some cases, there is a risk of overfitting the training data, especially if the base models are too complex or if there is not enough diversity among them.
4. **4. Dependency on base models**
   The effectiveness of ensemble learning depends on the quality and diversity of the base models. If the base models are weak or similar, the ensemble might not provide significant improvements.

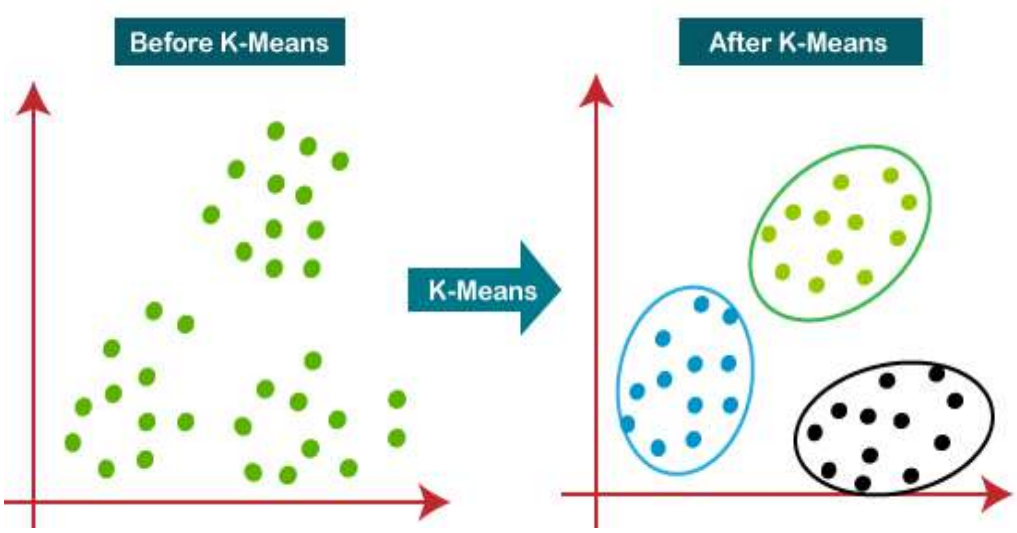
# K MEANS CLUSTERING

**"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."**

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.

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.




How does the K-Means Algorithm Work?

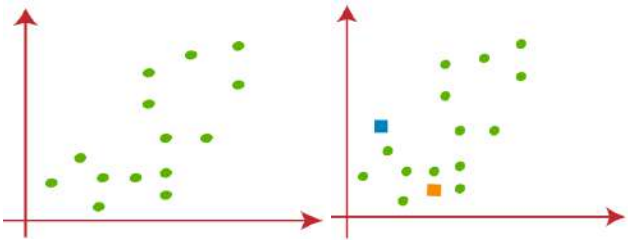
The working of the K-Means algorithm is explained in the below steps:

- **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.

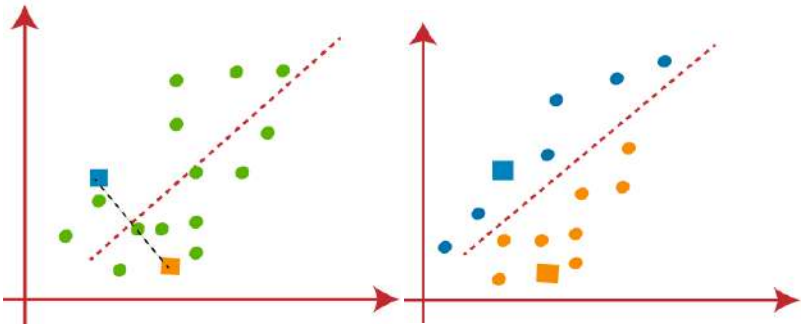
- **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
- **Step-7**: The model is ready.

**Example**

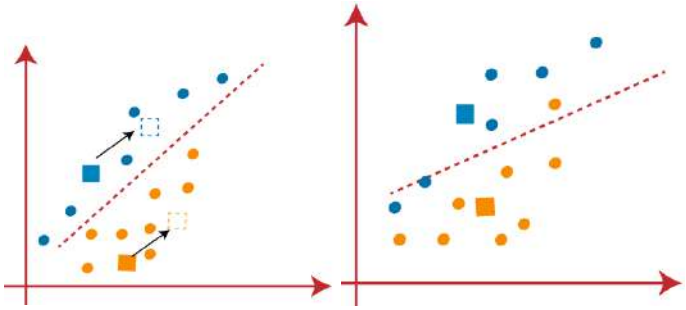
- Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We 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:



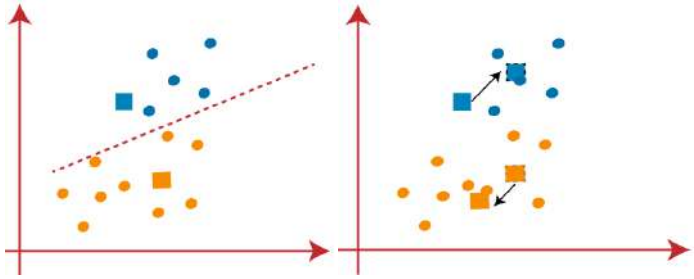
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:
- From the above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



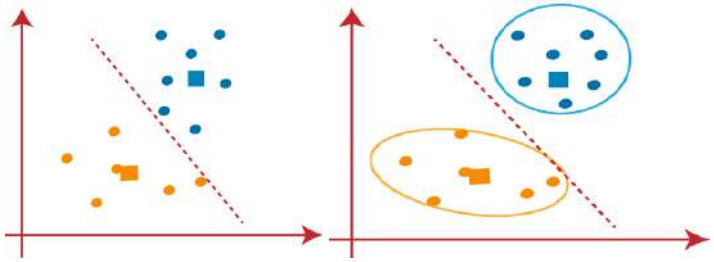
- As we need to find the closest cluster, so we will repeat the process by choosing a new centroid. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:
- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



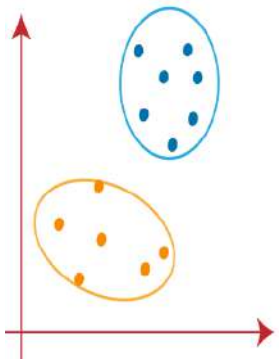
- From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.
- As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.
- 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:



Example Problem

- Let's imagine we have 5 objects (say 5 people) and for each of them we know two features (height and weight). We want to group them into k=2 clusters.
- Our dataset will look like this:

| | Height (H) | Weight (W) |
|---|---|---|
| Person 1 | 167 | 55 |
| Person 2 | 120 | 32 |
| Person 3 | 113 | 33 |
| Person 4 | 175 | 76 |
| Person 5 | 108 | 25 |

First of all, we have to initialize the value of the centroids for our clusters. For instance, let's choose Person 2 and Person 3 as the two centroids c1 and c2, so that c1=(120,32) and c2=(113,33). Now we compute the euclidian distance between each of the two centroids and each point in the data. If you did all the calculations, you should have come up with the following numbers:

| | **Distance of object from *c1*** | **Distance of object from *c2*** |
|---|---|---|
| Person 1 | 52.3 | 58.3 |
| Person 2 | 0 | 7.1 |
| Person 3 | 7.1 | 0 |
| Person 4 | 70.4 | 75.4 |
| Person 5 | 13.9 | 9.4 |

- At this point, we will assign each object to the cluster it is closer to (that is taking the minimum between the two computed distances for each object).
- We can then arrange the points as follows:

Person 1 → cluster 1
Person 2 → cluster 1
Person 3 → cluster 2
Person 4 → cluster 1
Person 5→ cluster 2

Let's iterate, which means to redefine the centroids by calculating the mean of the members of each of the two clusters.

- So c'1 = ((167+120+175)/3, (55+32+76)/3) = (154, 54.3) and c'2 = ((113+108)/2, (33+25)/2) = (110.5, 29)
- Then, we calculate the distances again and re-assign the points to the new centroids.
- We repeat this process until the centroids don't move anymore (or the difference between them is under a certain small threshold).
- In our case, the result we get is given in the figure below. You can see the two different clusters labelled with two different colours and the position of the centroids, given by the crosses.

# HIERARCHICAL CLUSTERING

- Hierarchical clustering is a popular method for grouping objects.
- It creates groups so that objects within a group are similar to each other and different from objects in other groups.
- Clusters are visually represented in a hierarchical tree called a dendrogram.
- Hierarchical clustering is the most popular and widely used method to analyze social network data.
- In this method, nodes are compared with one another based on their similarity.
- Larger groups are built by joining groups of nodes based on their similarity.

A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

1. Identify the 2 clusters which can be closest together, and

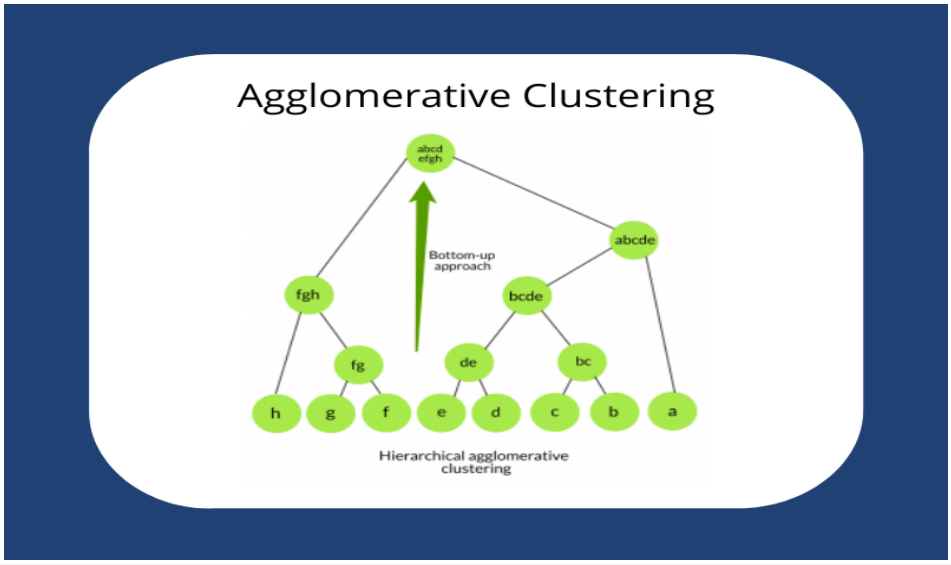
2. Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

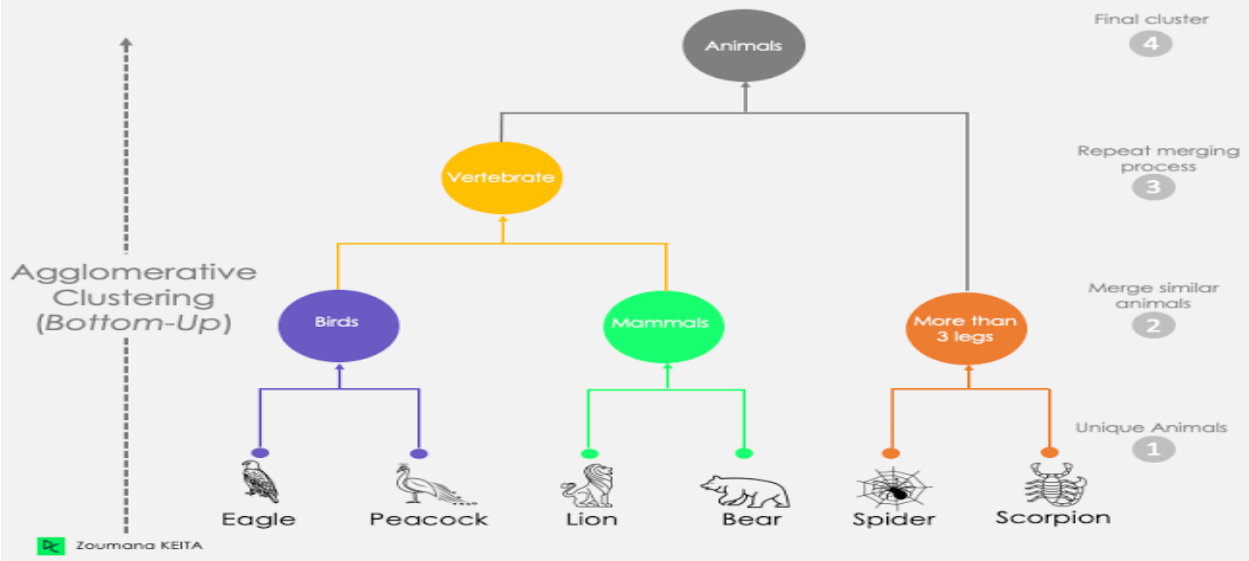
- In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters.
- A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits.

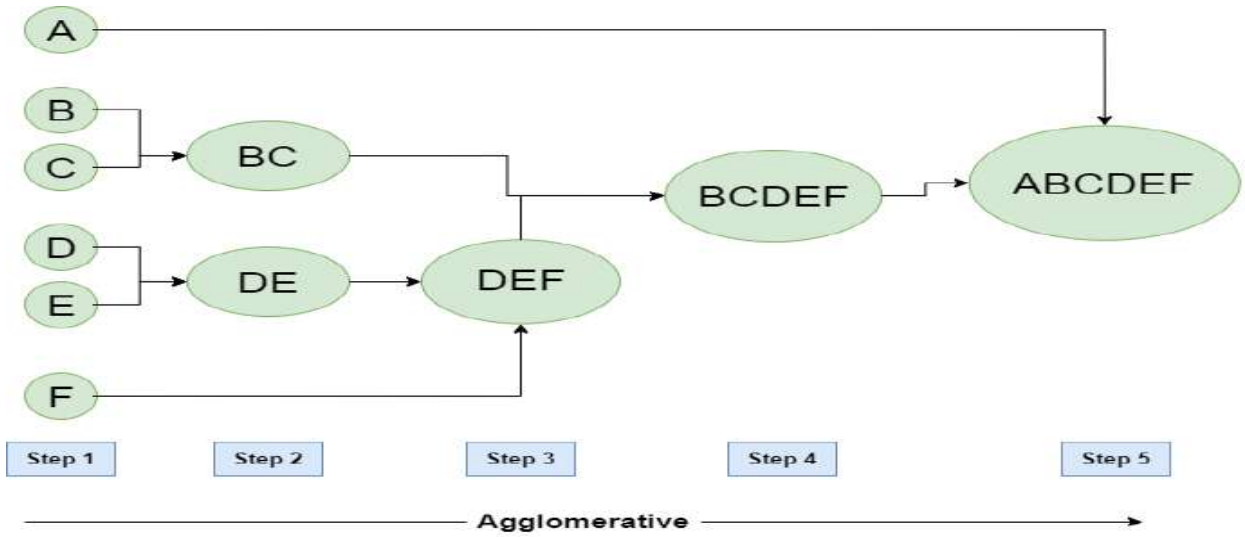
**The hierarchical clustering technique has two approaches:**

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

**AGGLOMERATIVE CLUSTERING**

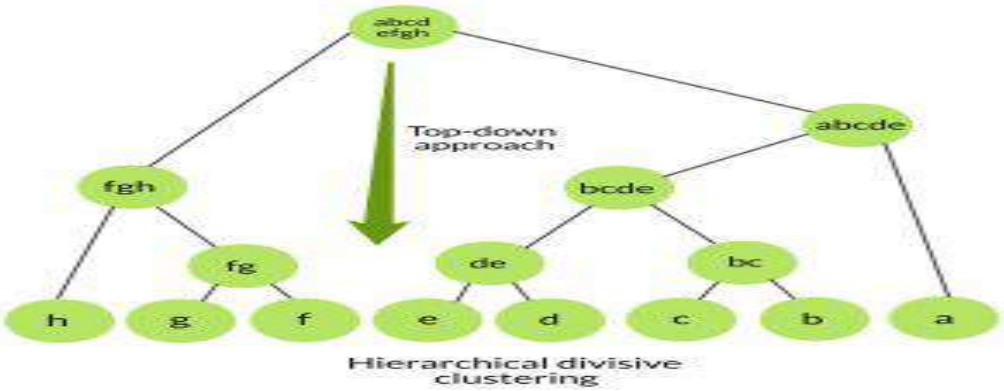


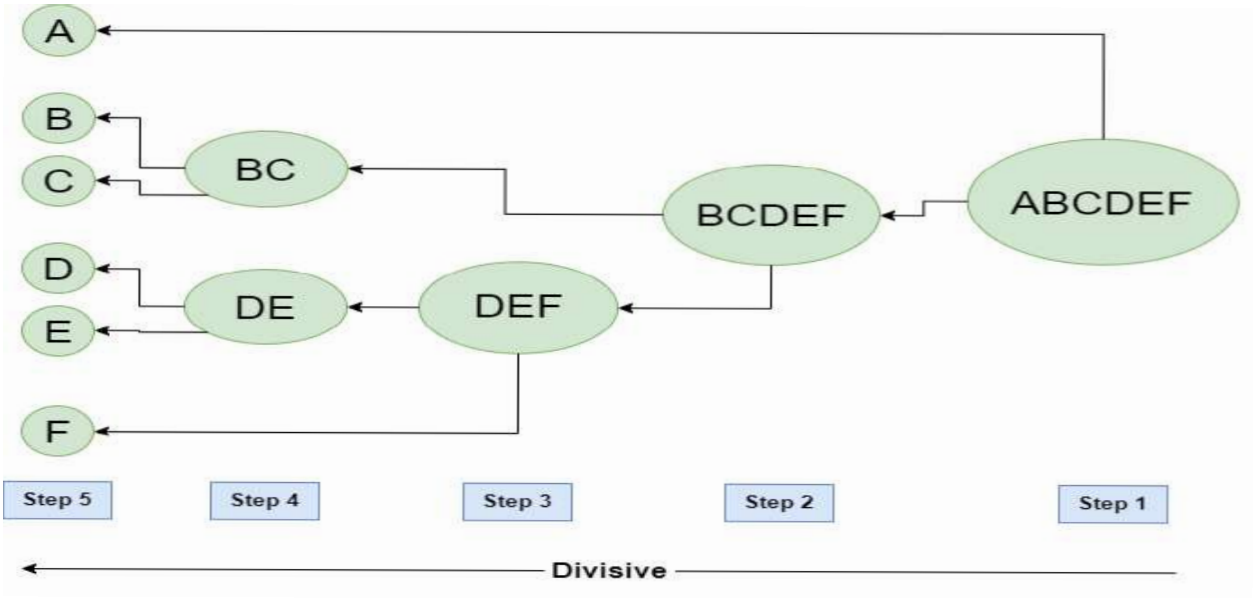






- The algorithm for Agglomerative Hierarchical Clustering is:
- Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
- Consider every data point as an individual cluster
- Merge the clusters which are highly similar or close to each other.
- Recalculate the proximity matrix for each cluster
- Repeat Steps 3 and 4 until only a single cluster remains.

**Divisive Hierarchical Clustering**



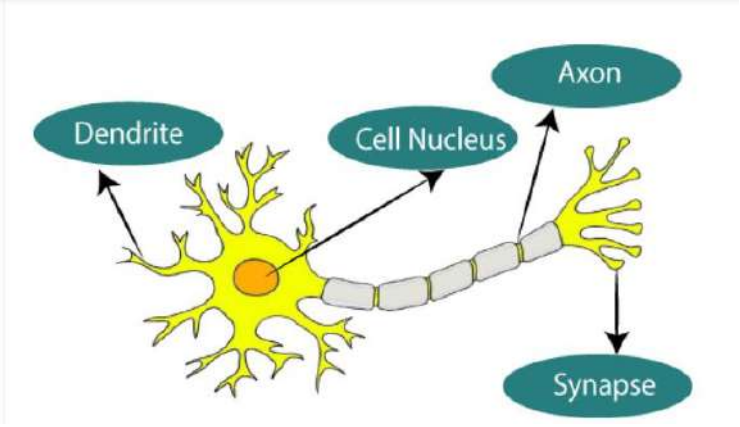



- **This approach starts with all of the objects in the same cluster.**
- **In the continuous iteration, a cluster is split up into smaller clusters.**
- **It is down until each object in one cluster or the termination condition holds.**
- **This method is rigid, i.e., once a merging or splitting is done, it can never be undone.**

**CO-4**

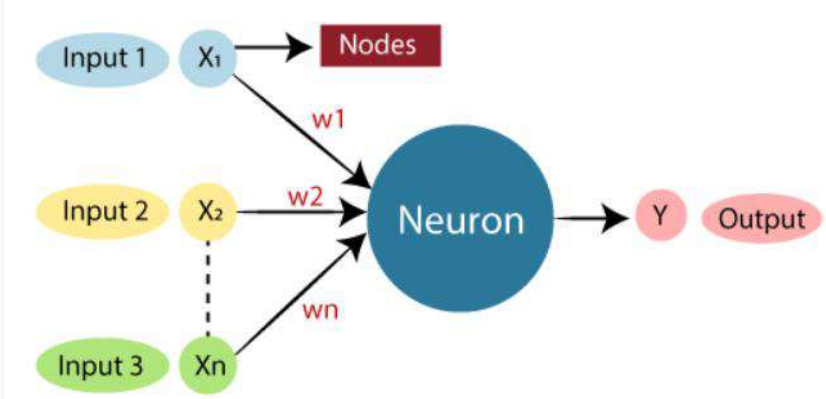
## Artificial Neural Network(ANN)

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.




**The given figure illustrates the typical diagram of Biological Neural Network.**

**The typical Artificial Neural Network looks something like the given figure.**




Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

| Biological Neural Network | Artificial Neural Network |
|---|---|
| Dendrites | Inputs |
| Cell nucleus | Nodes |
| Synapse | Weights |
| Axon | Output |

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

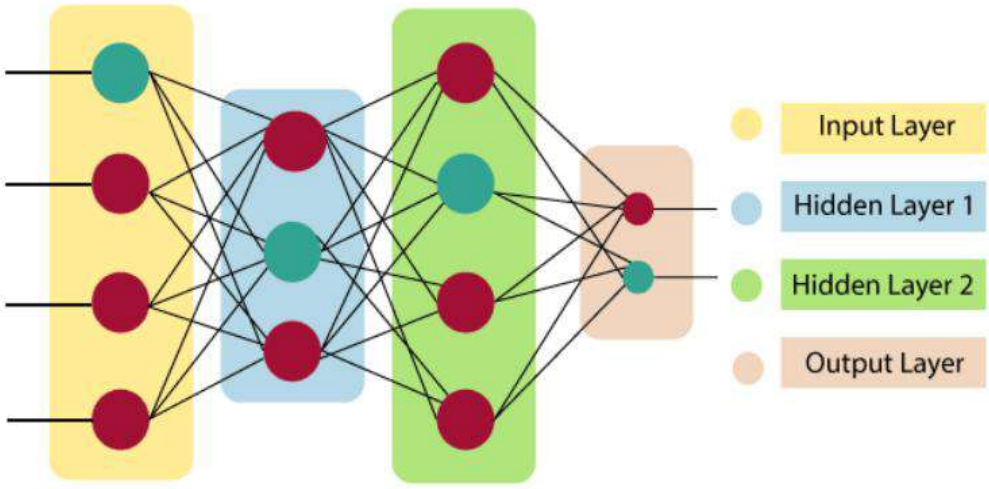
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

**Artificial Neural Network primarily consists of three layers:**




**Input Layer:**

As the name suggests, it accepts inputs in several different formats provided by the programmer.

**Hidden Layer:**

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

**Output Layer:**

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

**Advantages of Artificial Neural Network (ANN)**

**Parallel processing capability:**

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

**Storing data on the entire network:**

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

**Capability to work with incomplete knowledge:**

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

**Having a memory distribution:**

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

**Having fault tolerance:**

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network:

**Assurance of proper network structure:**

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

**Unrecognized behavior of the network:**

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

**Hardware dependence:**

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

**Difficulty of showing the issue to the network:**

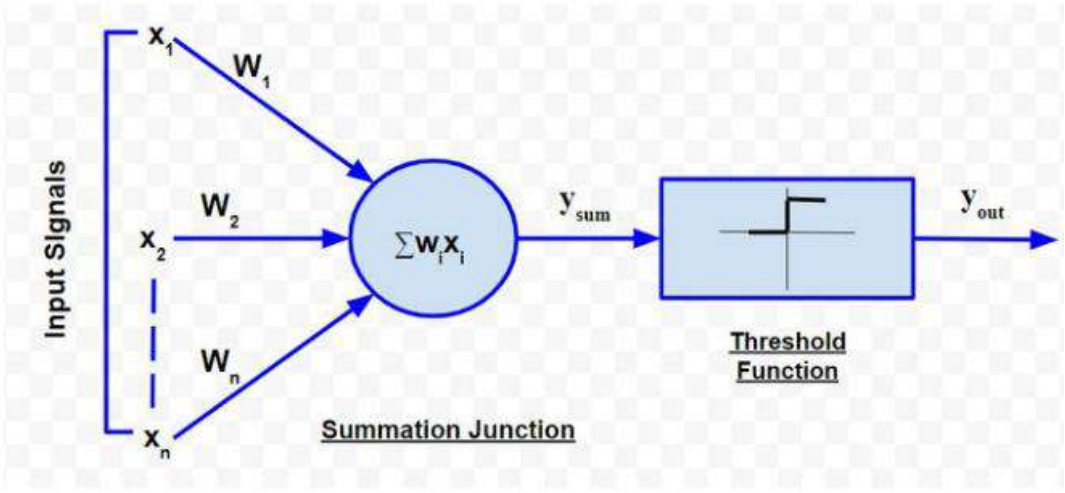
ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

**The duration of the network is unknown:**

The network is reduced to a specific value of the error, and this value does not give us optimum results.

**McCulloch-Pitts Model of Neuron**

The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — **Excitatory and Inhibitory.** The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal *yout* is 1 if the input *ysum* is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:




Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function

(rather than the threshold value of the activation function). For better understanding purpose, let me consider an example:

John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

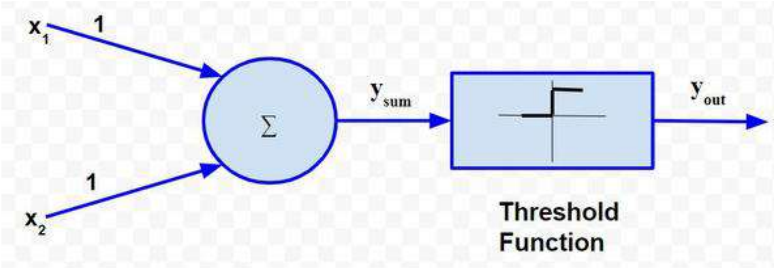
- First scenario: It is not raining, nor it is sunny
- Second scenario: It is not raining, but it is sunny
- Third scenario: It is raining, and it is not sunny
- Fourth scenario: It is raining as well as it is sunny

To analyse the situations using the McCulloch-Pitts neural model, I can consider the input signals as follows:

X1: Is it raining?

X2 : Is it sunny?

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X1 and X2 as 1 and a threshold function as 1. So, the neural network model will look like:




**Truth Table for this case will be:**

| Situation | $x_1$ | $x_2$ | $y_{sum}$ | $y_{out}$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 1 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 2 | 1 |

So, I can say that,

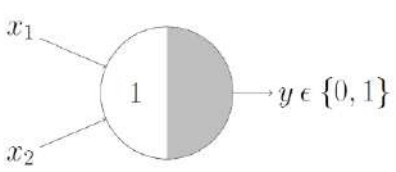
$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

The truth table built with respect to the problem is depicted above. From the truth table, I can conclude that in the situations where the value of *yout* is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4

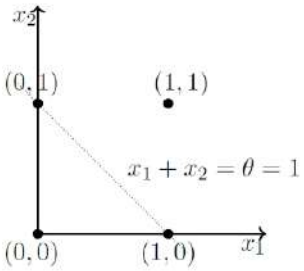
**OR Function**

We know that the thresholding parameter for OR function is 1, i.e. theta is 1. The possible combinations of inputs are: (0,0), (0,1), (1,0), and (1,1). Considering the OR function's aggregation equation, i.e. x_1+x_2≥1, let us plot the graph.




*OR function*

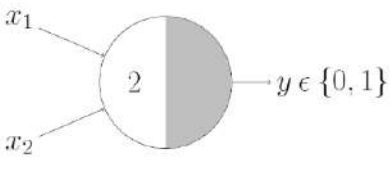
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$




The graph shows that the inputs for which the output when passed through OR function M-P neuron lie ON or ABOVE (output is 1, positive) that line and all inputs that lie BELOW (output is 0, negative) that line give the output as 0.
Therefore, the McCulloh Pitt's Model has made a linear decision boundary which splits the inputs into two classes, which are positive and negative.

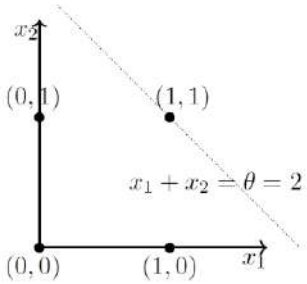
**AND Function**

Similar to OR Function, we can plot the graph for AND function considering the equation is x_1+x_2=2.




*AND function*

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$

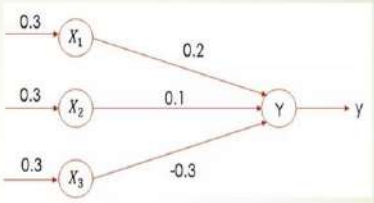



Here, the decision boundary separates all the input points that lie ON or ABOVE and give output 1 with just (1,1) when passed through the AND function.

From these examples, we can understand that with increase in the number of inputs, the dimensions which are plotted on the graph will also increase, which means that if we consider 3 inputs with OR function, we will plot a graph on a three-dimensional (3D) plane and draw a decision boundary in 3 dimensions.

Example-1

**Q1.** For the network shown in Figure 1, calculate the weights are net input to the output ne




**Solution:** The given neural net consists of three input neurons and one output neuron. The inputs and weights are

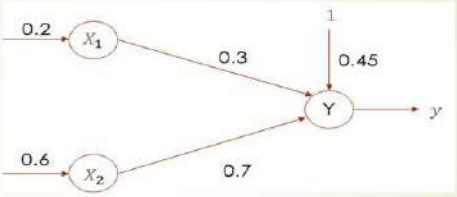
$$[x_1, x_2, x_3] = [0.3, 0.5, 0.6]$$
$$[w_1, w_2, w_3] = [0.2, 0.1, -0.3]$$

The net input can be calculated as

$$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3$$
$$= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times (-0.3)$$
$$= 0.06 + 0.05 - 0.18 = -0.07$$

Example-2

**Q2.** Calculate the net input for the network shown in Figure 2 with bias included in the network.




**Solution:** The given net consists of two input neurons a bias and an output neuron. The inputs are $[x_1, x_2] = [0.2, 0.6]$ and the weights are $[w_1, w_2] = [0.3, 0.7]$. Since the bias is included b=0.45 and bias input $x_0$ is equal to 1. The net input is calculated as

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0.45 + 0.2 \times 0.3 + 0.6 \times 0.7$$
$$= 0.45 + 0.06 + 0.42 = 0.93$$

Therefore, $y_{in} = 0.93$ is the net input.

# Perceptron

In Machine Learning and Artificial Intelligence, Perceptron is the most commonly used term for all folks. It is the primary step to learn Machine Learning and Deep Learning technologies, which consists of a set of weights, input values or scores, and a threshold. ***Perceptron is a building block of an Artificial Neural Network***. Initially, in the mid of 19th century, **Mr. Frank Rosenblatt** invented the Perceptron for performing certain calculations to detect input data capabilities or business intelligence. Perceptron is a linear Machine Learning algorithm used for supervised learning for various binary classifiers. This

algorithm enables neurons to learn elements and processes them one by one during preparation. In this tutorial, "Perceptron in Machine Learning," we will discuss in-depth knowledge of Perceptron and its basic functions in brief. Let's start with the basic introduction of Perceptron.

**What is the Perceptron model in Machine Learning?**

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

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

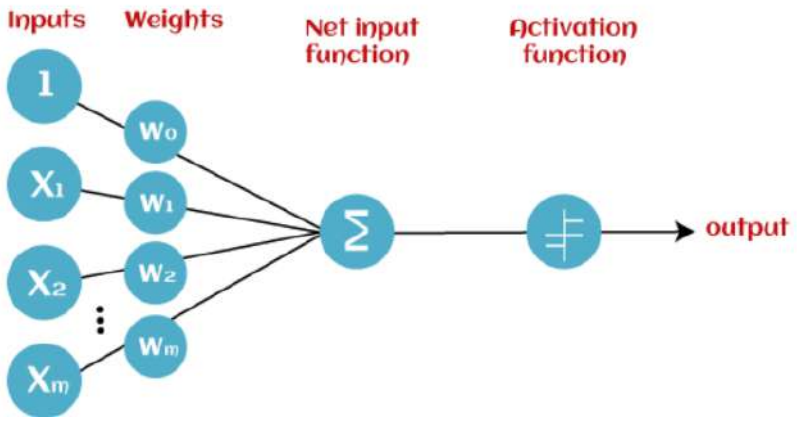
**What is Binary classifier in Machine Learning?**

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

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

**Basic Components of Perceptron**

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




- **Input Nodes or Input Layer:**

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

- **Wight and Bias:**

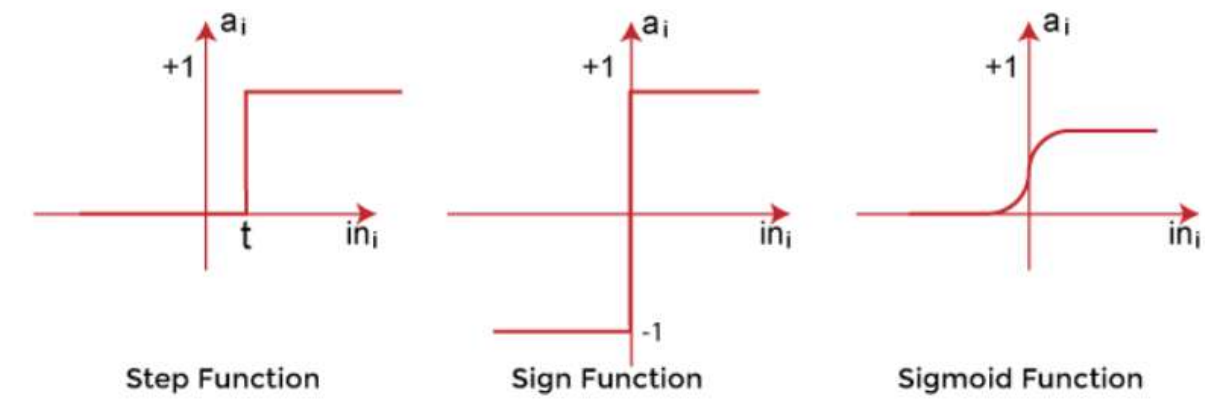
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept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

- **Activation Function:**

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

Types of Activation functions:

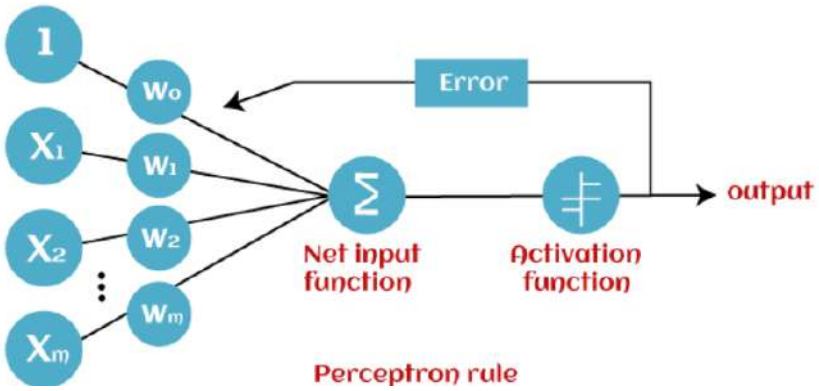
- Sign function
- Step function, and
- Sigmoid function




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

**How does Perceptron work?**

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




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

Perceptron model works in two important steps as follows:

**Step-1**

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

$$\sum wi*xi = x1*w1 + x2*w2 + ...wn*xn$$

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

$$\sum wi*xi + b$$

**Step-2**

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

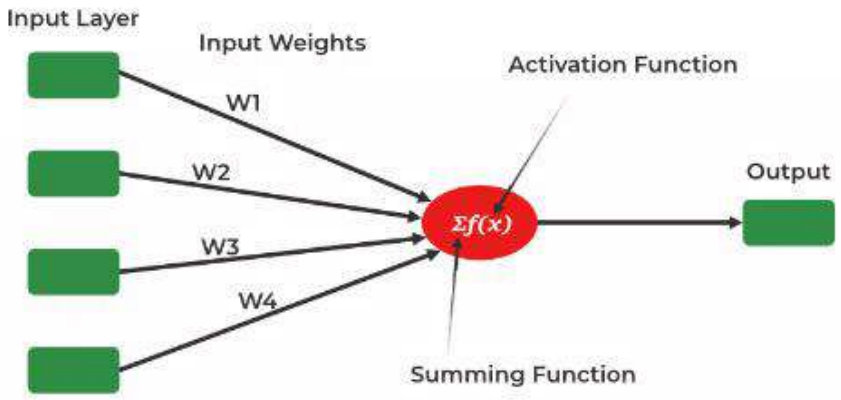
$$Y = f(\sum wi*xi + b)$$

Types of Perceptron Models

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

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

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




The main functionality of the perceptron is:-

- Takes input from the input layer
- Weight them up and sum it up.
- Pass the sum to the nonlinear function to produce the output.

Here activation functions can be anything like **sigmoid, tanh, relu** Based on the requirement we will be choosing the most appropriate nonlinear activation function to produce the better result. Now let us implement a single-layer perceptron.

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

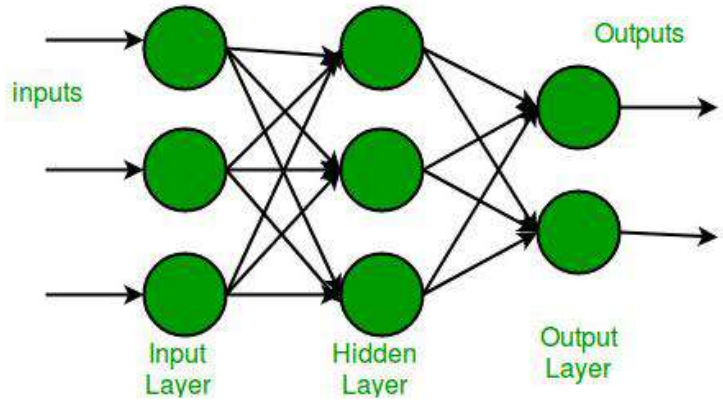
If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output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

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

**Multi-Layered Perceptron Model:**

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

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.




In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

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

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

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

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

**Advantages of Multi-Layer Perceptron:**

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

**Disadvantages of Multi-Layer Perceptron:**

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

Example Problem

Calculate the output value Y of the given perceptron for each of the following input patterns (given in table): Weights corresponding to the three inputs have the following values: w1 = -0.2, w2 = 0.4, and w3 = -0.3 and the activation of the unit is given by the step function.

| Pattern | P1 | P2 | P3 |
|---|---|---|---|
| X1 | 1 | 0 | 1 |
| X2 | 0 | 1 | 0 |
| X3 | 0 | 1 | 1 |

Solution

To solve this problem, we need to calculate the output YYY of a perceptron for each input pattern, using the weights w1=−0.2w_1 = -0.2w1=−0.2, w2=0.4w_2 = 0.4w2=0.4, and w3=−0.3w_3 = -0.3w3=−0.3, and applying the step function as the activation function.

The step function activates as follows:

$$Y = \begin{cases} 1 & \text{if weighted sum} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

For each pattern X1, X2, and X3, we'll calculate the weighted sum and apply the step function.

### Given Patterns and Weights:

- Weights: w1=−0.2w_1 = -0.2w1=−0.2, w2=0.4w_2 = 0.4w2=0.4, w3=−0.3w_3 = -0.3w3=−0.3
- Patterns:
  - X1=(1,0,1)X_1 = (1, 0, 1)X1=(1,0,1)
  - X2=(0,1,0)X_2 = (0, 1, 0)X2=(0,1,0)
  - X3=(0,1,1)X_3 = (0, 1, 1)X3=(0,1,1)

### Calculations:

1. **For X1=(1,0,1)X_1 = (1, 0, 1)X1=(1,0,1):**

   Weighted sum=(1·−0.2)+(0·0.4)+(1·−0.3)=−0.2+0−0.3=−0.5

   Since −0.5<0, Y=0

2. **For X2=(0,1,0)X_2 = (0, 1, 0)X2=(0,1,0):**

   Weighted sum=(0·−0.2)+(1·0.4)+(0·−0.3)=0+0.4+0=0.4

   Since 0.4≥0, Y=1.

3. **For X3=(0,1,1)X_3 = (0, 1, 1)X3=(0,1,1):**

   Weighted sum=(0·−0.2)+(1·0.4)+(1·−0.3)=0+0.4−0.3=0.1

   Since 0.1≥0 , Y=1.

### Summary:

- For X1X_1X1: Y=0Y = 0Y=0
- For X2X_2X2: Y=1Y = 1Y=1
- For X3X_3X3: Y=1Y = 1Y=1

So, the output values for each pattern are:

- Y(X1)=0Y(X_1) = 0Y(X1)=0

- Y(X2)=1Y(X_2) = 1Y(X2)=1
- Y(X3)=1Y(X_3) = 1Y(X3)=1

---

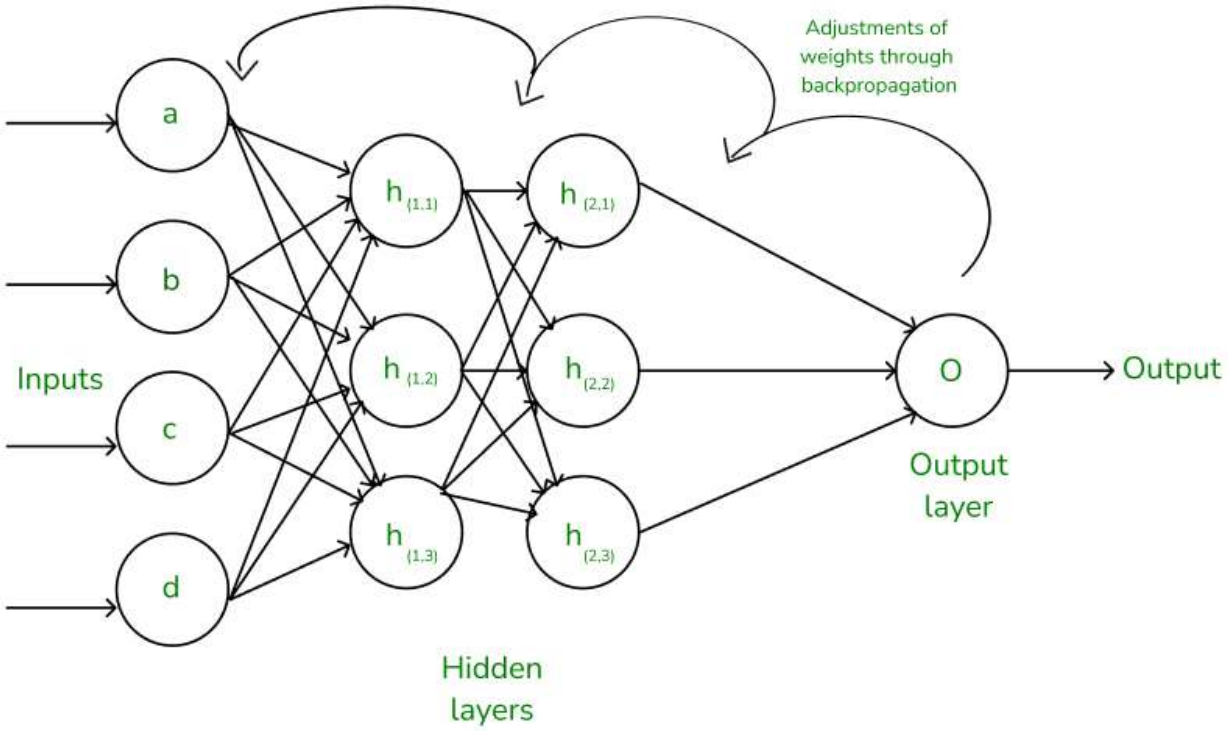
## Backpropagation

**Backpropagation** is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.

For a single training example, **Backpropagation** algorithm calculates the gradient of the **error function**. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation.

Now, how error function is used in Backpropagation and how Backpropagation works? Let start with an example and do it mathematically to understand how exactly updates the weight using Backpropagation.

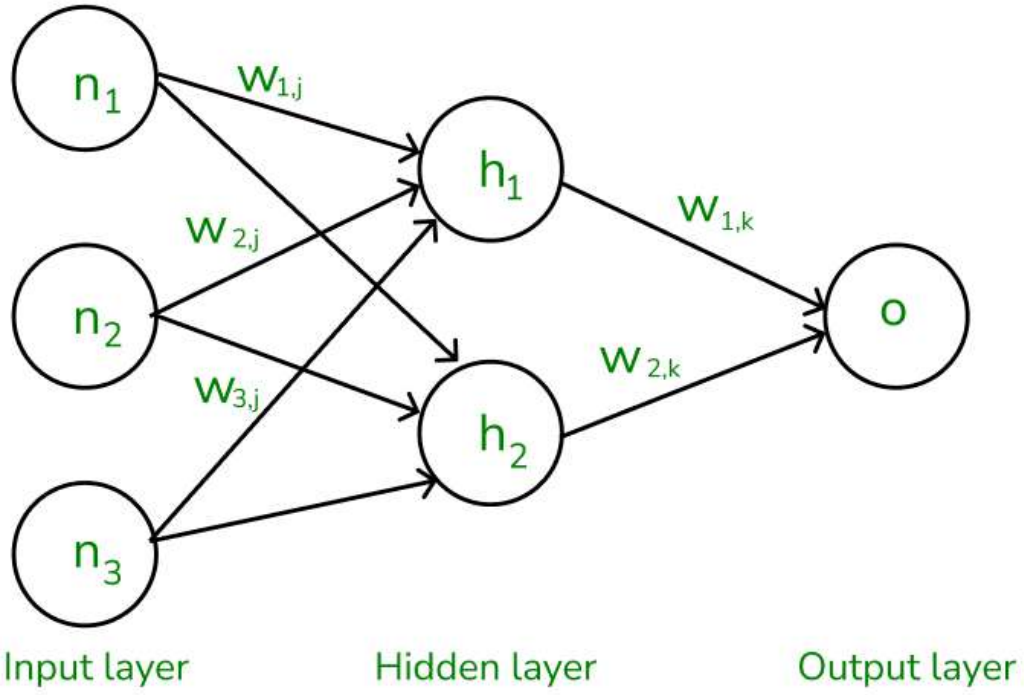



**How Does the Forward Pass Work?**

In the **forward pass**, the input data is fed into the input layer. These inputs, combined with their respective weights, are passed to hidden layers.

For example, in a network with two hidden layers (h1 and h2 as shown in Fig. (a)), the output from h1 serves as the input to h2. Before applying an activation function, a bias is added to the weighted inputs.

Each hidden layer applies an activation function like **ReLU (Rectified Linear Unit)**, which returns the input if it's positive and zero otherwise. This adds non-linearity, allowing the model to learn complex relationships in the data. Finally, the outputs from the last hidden layer are passed to the output layer, where an activation function, such as **softmax**, converts the weighted outputs into probabilities for classification.



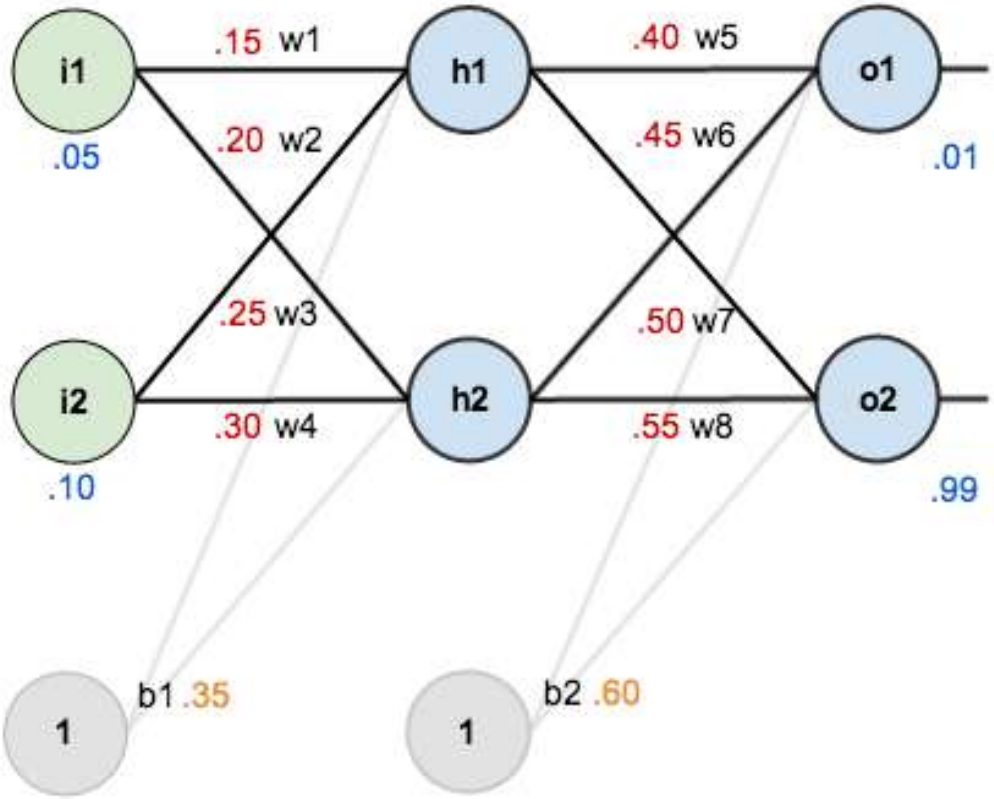

**How Does the Backward Pass Work?**

In the backward pass, the error (the difference between the predicted and actual output) is propagated back through the network to adjust the weights and biases. One common method for error calculation is the **Mean Squared Error (MSE)**, given by:

MSE=(Predicted Output−Actual Output)2*MSE*=(Predicted Output−Actual Output)2

Once the error is calculated, the network adjusts weights using **gradients**, which are computed with the chain rule. These gradients indicate how much each weight and bias should be adjusted to minimize the error in the next iteration. The backward pass continues layer by layer, ensuring that the network learns and improves its performance. The activation function, through its derivative, plays a crucial role in computing these gradients during backpropagation.

## Example of Backpropagation in Machine Learning




The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.
To work with a single training set: given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.

**Calculating the Total Error**

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

For example, the target output for $O_1$ is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for $O_2$ (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer to the target output, thereby minimizing the error for each output neuron and the network as a whole.

**Output Layer**

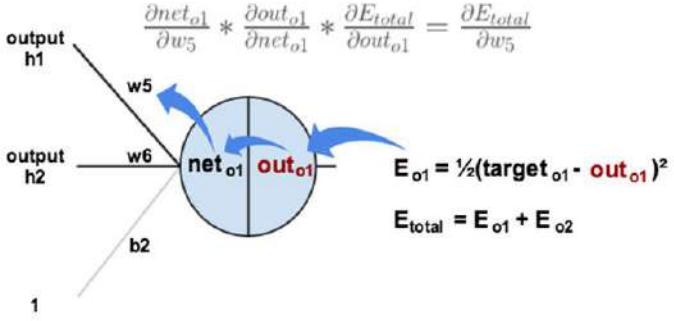
Consider $w_5$ we want to know how much a change in $w_5$ affects the total error, $\frac{\partial E_{total}}{\partial w_5}$

$\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of $E_{total}$ with respect to $w_5$". You can also say "the gradient with respect to $w_5$".

By applying the chain rule we know that:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$




We need to figure out each piece in this equation.
First, how much does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

When we take the partial derivative of the total error with respect to $out_{o1}$ the quantity $\frac{1}{2}(target_{o2} - out_{o2})^2$ becomes zero because $out_{o1}$ does not affect it which means we're taking the derivative of a constant which is zero.

---

Next, how much does the output of $o_1$ change with respect to its total net input?
The partial derivative of the logistic function is the output multiplied by 1 minus the output:

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of $o1$ change with respect to $w_5$?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$



Putting it all together: $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

You'll often see this calculation combined in the form of the delta rule:

The Delta rule in machine learning and neural network environments is a specific type of backpropagation that helps to refine connectionist ML/AI networks, making connections between inputs and outputs with layers of artificial neurons.

In general, backpropagation has to do with recalculating input weights for artificial neurons using a gradient method. Delta learning does this using the difference between a target activation and an actual obtained activation. Using a linear activation function, network connections are adjusted.

Another way to explain the Delta rule is that it uses an error function to perform gradient descent learning.

Putting it all together: $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

You'll often see this calculation combined in the form of the delta rule:

The Delta rule in machine learning and neural network environments is a specific type of backpropagation that helps to refine connectionist ML/AI networks, making connections between inputs and outputs with layers of artificial neurons.

In general, backpropagation has to do with recalculating input weights for artificial neurons using a gradient method. Delta learning does this using the difference between a target activation and an actual obtained activation. Using a linear activation function, network connections are adjusted.

Another way to explain the Delta rule is that it uses an error function to perform gradient descent learning.

## Apply the Delta rule

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

We can repeat this process to get the new weights $w_6$, $w_7$ and $w_8$

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$