ML Unit 1 to 5

MACHINE LEARNING

UNIT-1

Introduction- Artificial Intelligence, Machine Learning, Deep learning, Types of Machine Learning Systems, Main Challenges of Machine Learning. Statistical Learning: Introduction, Supervised and Unsupervised Learning, Training and Test Loss, Trade-offs in Statistical Learning, Estimating Risk Statistics, Sampling distribution of an estimator, Empirical Risk Minimization.

Artificial Intelligence:

Artificial intelligence is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.

Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power."*

Goals Of Artificial Intelligence:

Following are the main goals of Artificial Intelligence:

1. Replicate human intelligence
2. Solve Knowledge-intensive tasks
3. An intelligent connection of perception and action
4. Building a machine which can perform tasks that requires human intelligence such as:
    - Proving a theorem
    - Playing chess
    - Plan some surgical operation
    - Driving a car in traffic
5. Creating some system which can exhibit intelligent behavior, learn new things by itself, demonstrate, explain, and can advise to its user.

Advantages of Artificial Intelligence:

Following are some main advantages of Artificial Intelligence:

- **High Accuracy with less errors:** AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
- **High-Speed:** AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.

- **High reliability:** AI machines are highly reliable and can perform the same action multiple times with high accuracy.
- **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
- **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.
- **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security purpose, Natural language processing to communicate with the human in human-language, etc.

Disadvantages Of Artificial Intelligence:

Every technology has some disadvantages, and thesame goes for Artificial intelligence. Being so advantageous technology still, it has some disadvantages which we need to keep in our mind while creating an AI system. Following are the disadvantages of AI:

- **High Cost:** The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
- **Can't think out of the box:** Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
- **No feelings and emotions:** AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
- **Increase dependency on machines:** With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
- **No Original Creativity:** As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

Machine Learning:

Machine Learning is the science (and art) of programming computers so they can learn from data.

Here is a slightly more general definition: [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. —Arthur Samuel, 1959

And a more engineering-oriented one: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. —Tom Mitchell, 1997

Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Applications of Machine Learning:

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

2. Speech Recognition**:** While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning. Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3. Traffic prediction: If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions. It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

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

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

5. Self-driving cars: One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering: Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as **Multi-Layer Perceptron**, **Decision tree**, and **Naïve Bayes classifier** are used for email spam filtering and malware detection.

7. Virtual Personal Assistant: We have various virtual personal assistants such as **Google assistant**, **Alexa**, **Cortana**, **Siri**. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

8. Online Fraud Detection: Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as **fake accounts**, **fake ids**, and **steal money** in the middle of a transaction. So to detect this, **Feed Forward Neural network** helps us by checking whether it is a genuine transaction or a fraud transaction.

9. Stock Market trading: Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

10. Medical Diagnosis: In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain. It helps in finding brain tumors and other brain-related diseases easily.

11. Automatic Language Translation: Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.

Deep Learning:

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

Types of Deep Learning:

1. Feed Forward Neural Network: A feed-forward neural network is none other than an Artificial Neural Network, which ensures that the nodes do not form a cycle. In this kind of

neural network, all the perceptrons are organized within layers, such that the input layer takes the input, and the output layer generates the output. Since the hidden layers do not link with the outside world, it is named as hidden layers. Each of the perceptrons contained in one single layer is associated with each node in the subsequent layer. It can be concluded that all of the nodes are fully connected. It does not contain any visible or invisible connection between the nodes in the same layer. There are no back-loops in the feed-forward network. To minimize the prediction error, the backpropagation algorithm can be used to update the weight values.

2. Recurrent Neural Network: Recurrent neural networks are yet another variation of feed-forward networks. Here each of the neurons present in the hidden layers receives an input with a specific delay in time. The Recurrent neural network mainly accesses the preceding info of existing iterations. For example, to guess the succeeding word in any sentence, one must have knowledge about the words that were previously used. It not only processes the inputs but also shares the length as well as weights crossways time. It does not let the size of the model to increase with the increase in the input size. However, the only problem with this recurrent neural network is that it has slow computational speed as well as it does not contemplate any future input for the current state. It has a problem with reminiscing prior information.

3. Convolutional Neural Network: Convolutional Neural Networks are a special kind of neural network mainly used for image classification, clustering of images and object recognition. DNNs enable unsupervised construction of hierarchical image representations. To achieve the best accuracy, deep convolutional neural networks are preferred more than any other neural network.

4. Restricted Boltzmann Machine: RBMs are yet another variant of Boltzmann Machines. Here the neurons present in the input layer and the hidden layer encompasses symmetric connections amid them. However, there is no internal association within the respective layer. But in contrast to RBM, Boltzmann machines do encompass internal connections inside the hidden layer. These restrictions in BMs helps the model to train efficiently.

5. Autoencoders: An autoencoder neural network is another kind of unsupervised machine learning algorithm. Here the number of hidden cells is merely small than that of the input cells. But the number of input cells is equivalent to the number of output cells. An autoencoder network is trained to display the output similar to the fed input to force AEs to find common patterns and generalize the data. The autoencoders are mainly used for the smaller representation of the input. It helps in the reconstruction of the original data from compressed data. This algorithm is comparatively simple as it only necessitates the output identical to the input.

- **Encoder:** Convert input data in lower dimensions.
- **Decoder:** Reconstruct the compressed data.

Applications:

- ***Self-Driving Cars***
  In self-driven cars, it is able to capture the images around it by processing a huge amount of data, and then it will decide which actions should be incorporated to take a left or right or should it stop. So, accordingly, it will decide what actions it should take, which will further reduce the accidents that happen every year.
- ***Voice Controlled Assistance***
  When we talk about voice control assistance, then **Siri** is the one thing that comes into our mind. So, you can tell Siri whatever you want it to do it for you, and it will search it for you and display it for you.
- ***Automatic Image Caption Generation***
  Whatever image that you upload, the algorithm will work in such a way that it will generate caption accordingly. If you say blue colored eye, it will display a blue-colored eye with a caption at the bottom of the image.
- ***Automatic Machine Translation***
  With the help of automatic machine translation, we are able to convert one language into another with the help of deep learning.

Advantages:

- It lessens the need for feature engineering.
- It eradicates all those costs that are needless.
- It easily identifies difficult defects.
- It results in the best-in-class performance on problems.

Disadvantages:

- It requires an ample amount of data.
- It is quite expensive to train.
- It does not have strong theoretical groundwork.

Types of Machine Learning Systems:

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

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

1. Supervised Learning:

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

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

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

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

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

Applications:

Advertisement Popularity: Selecting advertisements that will perform well is often a supervised learning task. Many of the ads you see as you browse the internet are placed there because a learning algorithm said that they were of reasonable popularity (and clickability). Furthermore, its placement associated on a certain site or with a certain query (if you find yourself using a search engine) is largely due to a learned algorithm saying that the matching between ad and placement will be effective.

Spam Classification: If you use a modern email system, chances are you've encountered a spam filter. That spam filter is a supervised learning system. Fed email examples and labels (spam/not spam), these systems learn how to pre-emptively filter out malicious emails so that their user is not harassed by them. Many of these also behave in such a way that a user can provide new labels to the system and it can learn user preference.

Face Recognition: Most likely your face has been used in a supervised learning algorithm that is trained to recognize your face. Having a system that takes a photo, finds faces, and guesses who that is in the photo (suggesting a tag) is a supervised process. It has multiple layers to it, finding faces and then identifying them, but is still supervised nonetheless.

2. Unsupervised Learning:

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

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

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

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

Applications:

Recommender Systems: If you've ever used YouTube or Netflix, you've most likely encountered a video recommendation system. These systems are often times placed in the unsupervised domain. We know things about videos, maybe their length, their genre, etc. We also know the watch history of many users. Taking into account users that have watched similar videos as you and then enjoyed other videos that you have yet to see, a recommender system can see this relationship in the data and prompt you with such a suggestion.

Buying Habits: It is likely that your buying habits are contained in a database somewhere and that data is being bought and sold actively at this time. These buying habits can be used in unsupervised learning algorithms to group customers into similar purchasing segments. This helps companies market to these grouped segments and can even resemble recommender systems.

Grouping User Logs: Less user facing, but still very relevant, we can use unsupervised learning to group user logs and issues. This can help companies identify central themes to issues their customers face and rectify these issues, through improving a product or designing an FAQ to handle common issues. Either way, it is something that is actively done and if you've ever submitted an issue with a product or submitted a bug report, it is likely that it was fed to an unsupervised learning algorithm to cluster it with other similar issues.

3. Reinforcement Learning:

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

Applications:

Video Games: One of the most common places to look at reinforcement learning is in learning to play games. Look at Google's reinforcement learning application, AlphaZero and AlphaGo which learned to play the game Go. Our Mario example is also a common example. Currently, I don't know any production-grade game that has a reinforcement learning agent deployed as its game AI, but I can imagine that this will soon be an interesting option for game devs to employ.

Industrial Simulation: For many robotic applications (think assembly lines), it is useful to have our machines learn to complete their tasks without having to hardcode their processes. This can be a cheaper and safer option; it can even be less prone to failure. We can also incentivize our machines to use less electricity, so as to save us money. More than that, we can start this all within a simulation so as to not waste money if we potentially break our machine.

Resource Management: Reinforcement learning is good for navigating complex environments. It can handle the need to balance certain requirements. Take, for example, Google's data centers. They used reinforcement learning to balance the need to satisfy our power requirements, but do it as efficiently as possible, cutting major costs. How does this affect us and the average person? Cheaper data storage costs for us as well and less of an impact on the environment we all share.

Main Challenges of Machine Learning:

Although machine learning is being used in every industry and helps organizations make more informed and data-driven choices that are more effective than classical methodologies, it still has so many problems that cannot be ignored. Here are some challenges of Machine Learning:

1. **Insufficient Quantity of Training Data:** The major issue that comes while using machine learning algorithms is the lack of quality as well as quantity of data. Although data plays a vital role in the processing of machine learning algorithms, many data scientists claim that inadequate data, noisy data, and unclean data are extremely exhausting the machine learning

algorithms. For example, a simple task requires thousands of sample data, and an advanced task such as speech or image recognition needs millions of sample data examples. Further, data quality is also important for the algorithms to work ideally, but the absence of data quality is also found in Machine Learning applications. Data quality can be affected by some factors as follows:

- **Noisy Data-** It is responsible for an inaccurate prediction that affects the decision as well as accuracy in classification tasks.
- **Incorrect data-** It is also responsible for faulty programming and results obtained in machine learning models. Hence, incorrect data may affect the accuracy of the results also.
- **Generalizing of output data-** Sometimes, it is also found that generalizing output data becomes complex, which results in comparatively poor future actions.

2. **Nonrepresentative Training Data:** To make sure our training model is generalized well or not, we have to ensure that sample training data must be representative of new cases that we need to generalize. The training data must cover all cases that are already occurred as well as occurring.

Further, if we are using non-representative training data in the model, it results in less accurate predictions. A machine learning model is said to be ideal if it predicts well for generalized cases and provides accurate decisions. If there is less training data, then there will be a sampling noise in the model, called the non-representative training set. It won't be accurate in predictions. To overcome this, it will be biased against one class or a group.

Hence, we should use representative data in training to protect against being biased and make accurate predictions without any drift.

3. **Poor Quality of Data:** As we have discussed above, data plays a significant role in machine learning, and it must be of good quality as well. Noisy data, incomplete data, inaccurate data, and unclean data lead to less accuracy in classification and low-quality results. Hence, data quality can also be considered as a major common problem while processing machine learning algorithms.

4. **Irrelevant Features:** Although machine learning models are intended to give the best possible outcome, if we feed garbage data as input, then the result will also be garbage. Hence, we should use relevant features in our training sample. A machine learning model is said to be good if training data has a good set of features or less to no irrelevant features.

5. **Overfitting the training data:** Overfitting is one of the most common issues faced by Machine Learning engineers and data scientists. Whenever a machine learning model is trained with a huge amount of data, it starts capturing noise and inaccurate data into the training data set. It negatively affects the performance of the model. Let's understand with a simple example where we have a few training data sets such as 1000 mangoes, 1000 apples, 1000 bananas, and 5000 papayas. Then there is a considerable probability of identification of an apple as papaya

because we have a massive amount of biased data in the training data set; hence prediction got negatively affected. The main reason behind overfitting is using non-linear methods used in machine learning algorithms as they build non-realistic data models. We can overcome overfitting by using linear and parametric algorithms in the machine learning models.

6. **Underfitting the training data:** As you might guess, underfitting is the opposite of overfitting, it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training exam- ples. The main options to fix this problem are:

• Selecting a more powerful model, with more parameters

• Feeding better features to the learning algorithm (feature engineering)

• Reducing the constraints on the model (e.g., reducing the regularization hyper- parameter)

Statistical Learning:

Statistical learning refers to a vast set of tools for understanding data. These tools can be classified as supervised or unsupervised. Broadly speaking, supervised statistical learning involves building a statistical model for predicting, or estimating, an output based on one or more inputs. Problems of this nature occur in fields as diverse as business, medicine, astrophysics, and public policy. With unsupervised statistical learning, there are inputs but no supervising output; nevertheless we can learn relationships and structure from such data.

Supervised Learning and Unsupervised Learning:

Supervised Machine Learning:

Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: **Classification** and **Regression**.

Example: Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.

Unsupervised Machine Learning:

Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

Unsupervised learning can be used for two types of problems: **Clustering** and **Association**.

Example: To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

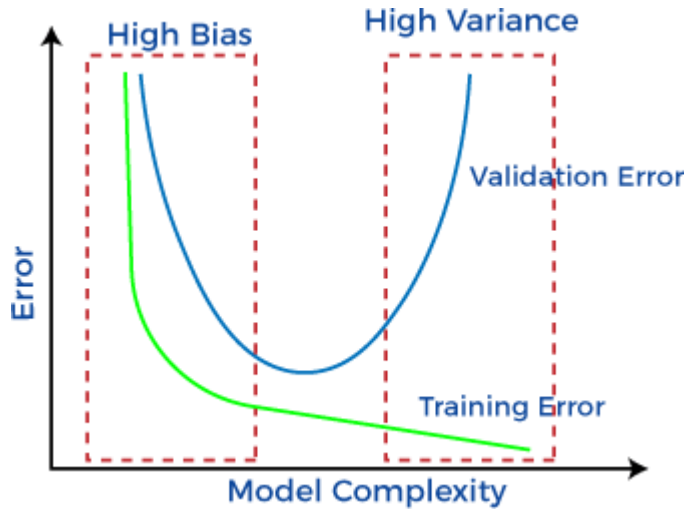
The main differences between Supervised and Unsupervised learning are given below:

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** proble ms.	Unsupervised Learning can be classified in **Clustering** and **Associations** proble ms.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first	Unsupervised learning is more close to the true Artificial Intelligence as it learns

train the model for each data, and then only it can predict the correct output.	similarly as a child learns daily routine things by his experiences.

Training and Test Loss:

Machine learning is a branch of Artificial Intelligence, which allows machines to perform data analysis and make predictions. However, if the machine learning model is not accurate, it can make predictions errors, and these prediction errors are usually known as Bias and Variance. In machine learning, these errors will always be present as there is always a slight difference between the model predictions and actual predictions. The main aim of ML/data science analysts is to reduce these errors in order to get more accurate results.



Errors in Machine Learning?

In machine learning, an error is a measure of how accurately an algorithm can make predictions for the previously unknown dataset. On the basis of these errors, the machine learning model is selected that can perform best on the particular dataset. There are mainly two types of errors in machine learning, which are:

Reducible errors: These errors can be reduced to improve the model accuracy. Such errors can further be classified into bias and Variance.
Irreducible errors: These errors will always be present in the model regardless of which algorithm has been used. The cause of these errors is unknown variables whose value can't be reduced.

What is Bias?

In general, a machine learning model analyses the data, find patterns in it and make predictions. While training, the model learns these patterns in the dataset and applies them to test data for prediction. ***While making predictions, a difference occurs between prediction values made by the model and actual values/expected values, and this difference is known as bias errors or Errors due to bias***. It can be defined as an inability of machine learning algorithms such as Linear Regression to capture the true relationship between the data points. Each algorithm

begins with some amount of bias because bias occurs from assumptions in the model, which makes the target function simple to learn. A model has either:

- **Low Bias:** A low bias model will make fewer assumptions about the form of the target function.
- **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset. **A high bias model also cannot perform well on new data.**

Generally, a linear algorithm has a high bias, as it makes them learn fast. The simpler the algorithm, the higher the bias it has likely to be introduced. Whereas a nonlinear algorithm often has low bias.

Some examples of machine learning algorithms with low bias **are Decision Trees, k-Nearest Neighbours and Support Vector Machines**. At the same time, an algorithm with high bias is **Linear Regression, Linear Discriminant Analysis and Logistic Regression.**

Ways to reduce High Bias:

High bias mainly occurs due to a much simple model. Below are some ways to reduce the high bias:

- Increase the input features as the model is underfitted.
- Decrease the regularization term.
- Use more complex models, such as including some polynomial features.

What is a Variance Error?

The variance would specify the amount of variation in the prediction if the different training data was used. In simple words, ***variance tells that how much a random variable is different from its expected value.*** Ideally, a model should not vary too much from one training dataset to another, which means the algorithm should be good in understanding the hidden mapping between inputs and output variables. Variance errors are either of **low variance or high variance.**

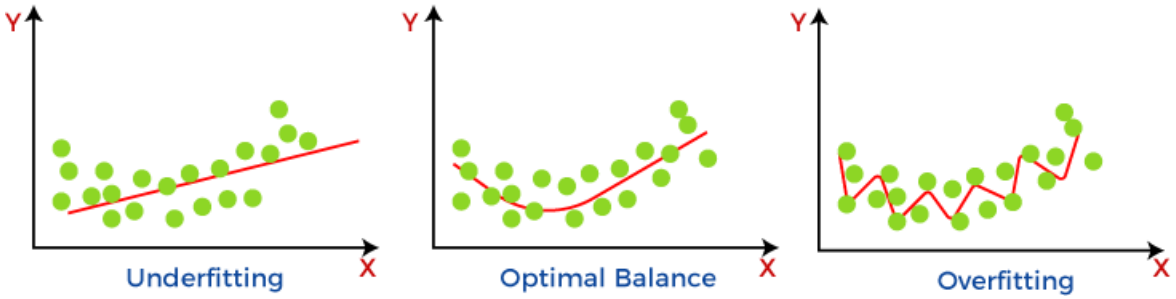
Low variance means there is a small variation in the prediction of the target function with changes in the training data set. At the same time, **High variance** shows a large variation in the prediction of the target function with changes in the training dataset.

A model that shows high variance learns a lot and perform well with the training dataset, and does not generalize well with the unseen dataset. As a result, such a model gives good results with the training dataset but shows high error rates on the test dataset.

Since, with high variance, the model learns too much from the dataset, it leads to overfitting of the model. A model with high variance has the below problems:

- A high variance model leads to overfitting.
- Increase model complexities.

Usually, nonlinear algorithms have a lot of flexibility to fit the model, have high variance.



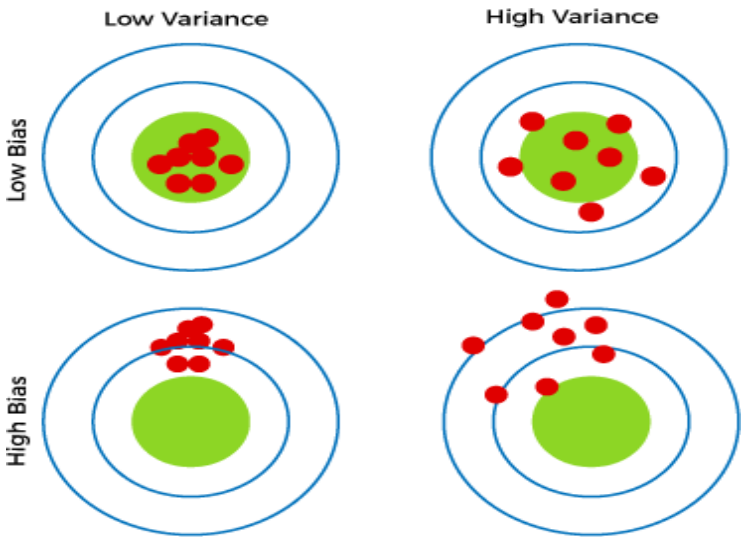
Some examples of machine learning algorithms with low variance are, **Linear Regression, Logistic Regression, and Linear discriminant analysis**. At the same time, algorithms with high variance are **decision tree, Support Vector Machine, and K-nearest neighbours.**

Ways to Reduce High Variance:

- Reduce the input features or number of parameters as a model is overfitted.
- Do not use a much complex model.
- Increase the training data.
- Increase the Regularization term.

Different Combinations of Bias-Variance

There are four possible combinations of bias and variances, which are represented by the below diagram:

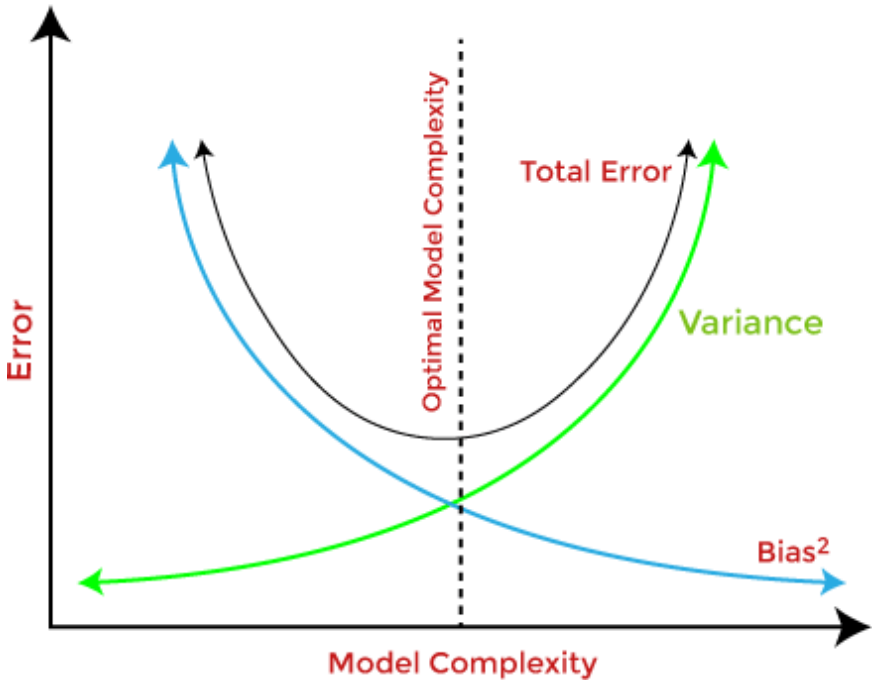


1. **Low-Bias, Low-Variance:** The combination of low bias and low variance shows an ideal machine learning model. However, it is not possible practically.
2. **Low-Bias, High-Variance:** With low bias and high variance, model predictions are inconsistent and accurate on average. This case occurs when the model learns with a large number of parameters and hence leads to an **overfitting**
3. **High-Bias, Low-Variance:** With High bias and low variance, predictions are consistent but inaccurate on average. This case occurs when a model does not learn well with the training dataset or uses few numbers of the parameter. It leads to **underfitting** problems in the model.
4. **High-Bias, High-Variance:** With high bias and high variance, predictions are inconsistent and also inaccurate on average.

Tradeoffs in Statistical Learning:

Bias vs variance: A trade-off

While building the machine learning model, it is really important to take care of bias and variance in order to avoid overfitting and underfitting in the model. If the model is very simple with fewer parameters, it may have low variance and high bias. Whereas, if the model has a large number of parameters, it will have high variance and low bias. So, it is required to make a balance between bias and variance errors, and this balance between the bias error and variance error is known as **the Bias-Variance trade-off.**



For an accurate prediction of the model, algorithms need a low variance and low bias. But this is not possible because bias and variance are related to each other:

- If we decrease the variance, it will increase the bias.
- If we decrease the bias, it will increase the variance.

Bias-Variance trade-off is a central issue in supervised learning. Ideally, we need a model that accurately captures the regularities in training data and simultaneously generalizes well with the unseen dataset. Unfortunately, doing this is not possible simultaneously. Because a high variance algorithm may perform well with training data, but it may lead to overfitting to noisy data. Whereas, high bias algorithm generates a much simple model that may not even capture important regularities in the data. So, we need to find a sweet spot between bias and variance to make an optimal model.

Hence, the ***Bias-Variance trade-off is about finding the sweet spot to make a balance between bias and variance errors.***

Estimating risk statistics:

Nowadays, Machine Learning is playing a big role in helping organizations in different aspects such as analyzing structured and unstructured data, detecting risks, automating manuals tasks, making data-driven decisions for business growth, etc. It is capable of replacing the huge amount of human labour by applying automation and providing insights to make better decisions for assessing, monitoring, and reducing the risks for an organization.

Although machine learning can be used as a risk management tool, it also contains many risks itself. While 49% of companies are exploring or planning to use machine learning, only a small minority recognize the risks it poses. In which, only 41% of organizations in a global McKinsey survey say they can comprehensively identify and prioritize machine learning risks. Hence, it is necessary to be aware of some of the risks of machine learning-and how they can be adequately evaluated and managed.

Below are a few risks associated with Machine Learning:

1. Poor Data

As we know, a machine learning model only works on the data that we provide to it, or we can say it completely depends on human-given training data to work. What we will be input that we will get as an output, so if we will enter the poor data, the ML model will generate abrupt output. Poor data or dirty data includes errors in training data, outliers, and unstructured data, which cannot be adequately interpreted by the model.

2. Overfitting

Overfitting is commonly found in non-parametric and non-linear models that are more flexible to learn target function.

An overfitted model fits the training data so perfectly that it becomes unable to learn the variability for the algorithm. It means it won't be able to generalize well when it comes to testing real data.

3. Biased data

Biased data means that human biases can creep into your datasets and spoil outcomes. For instance, the popular selfie editor FaceApp was initially inadvertently trained to make faces "hotter" by lightening the skin tone-a result of having been fed a much larger quantity of photos of people with lighter skin tones.

4. Lack of strategy and experience:

Machine learning is a very new technology in the IT sector; hence, less availability of trained and skilled resources is a very big issue for the industries. Further, lack of strategy and experience due to fewer resources leads to wastage of time and money as well as negatively affect the organization's production and revenue. According to a survey of over 2000 people, 860 reported to lack of clear strategy and 840 were reported to lack of talent with appropriate skill sets. This survey shows how lack of strategy and relevant experience creates a barrier in the development of machine learning for organizations.

5. Security Risks

Security of data is one of the major issues for the IT world. Security also affects the production and revenue of organizations. When it comes to machine learning, there are various types of security risks exist that can compromise machine learning algorithms and systems. Data scientists and machine learning experts have reported 3 types of attacks, primarily for machine learning models. These are as follows:

Evasion attacks: These attacks are commonly arisen due to adversarial input introduced in the models; hence they are also known as adversarial attacks.
An evasion attack happens when the network uses adversarial examples as input which can influence the classifiers, i.e., disrupting ML models. When a security violation involves supplying malicious data that gets classified as genuine. A targeted attack attempts to allow a specific intrusion or disruption, or alternatively to create general mayhem.

Data Poisoning attacks: In data poisoning attacks, the source of raw data is known, which is used to train the ML models. Further, it strives to bias or "poison" the data to compromise the resulting machine learning model's accuracy. The effects of these attacks can be overcome by prevention and detection. Through proper monitoring, we can prevent ML models from data poisoning. **Model skewing** is one the most common type of data poisoning attacks in which spammers categorise the classifiers with bad input as good.

Model Stealing: Model stealing is one of the most important security risks in machine learning. Model stealing techniques are used to create a clone model based on information or data used in the training of a base model. Why we are saying model stealing is a major concern for ML

experts because ML models are the valuable intellectual property of organizations that consist of sensitive data of users such as account details, transactions, financial information, etc. The attackers use public API and sample data of the original model and reconstruct another model having a similar look and feel.

6. Data privacy and confidentiality

Data is one of the main key players in developing Machine learning models. We know machine learning requires a huge amount of structured and unstructured data for training models so they can predict accurately in future. Hence, to achieve good results, we need to secure data by defining some privacy terms and conditions as well as making it confidential. Hackers can launch data extraction attacks that can fly under the radar, which can put your entire machine learning system at risk.

7. Third-party risks

These types of security risks are not so famous in industries as there are very minimal chances of these risks in industries. Third-party risks generally exist when someone outsources their business to third-party service providers who may fail to properly govern a machine learning solution. This leads to various types of data breaches in the ML industry.

8. Regulatory challenges

Regulatory challenges occur whenever a knowledge gap is found in an organization, such as teammates do not aware of how ML algorithms work and create decisions. Hence, a lack of knowledge to justify decisions to regulators can also be a major security risk for industries.

Sampling Distribution of the Estimator:

In statistics, it is the probability distribution of the given statistic estimated on the basis of a random sample. It provides a generalized way to statistical inference. The estimator is the generalized mathematical parameter to calculate sample statistics. An estimate is the result of the estimation.

The sampling distribution of estimator depends on the sample size. The effect of change of the sample size has to be determined. An estimate has a single numerical value and hence they are called point estimates. There are various estimators like sample mean, sample standard deviation, proportion, variance, range etc.

Sampling distribution of the mean: It is the population mean from which the samples are drawn. For all the sample sizes, it is likely to be normal if the population distribution is normal. The population mean is equal to the mean of the sampling distribution of the mean. Sampling distribution of mean has the standard deviation, which is as follows:

$$\sigma_M = \frac{\sigma}{\sqrt{n}}$$

Where $\sigma_{\bar{x}}$ is the standard deviation of the sampling mean, σ is the population standard deviation and *n* is the sample size.

As the size of the sample increases, the spread of the sampling distribution of the mean decreases. But the mean of the distribution remains the same and it is not affected by the sample size.

The sampling distribution of the standard deviation is the standard error of the standard deviation. It is defined as:

$$\sigma_s = \frac{\sigma}{\sqrt{2n}}$$

Here, σ_s is the sampling distribution of the standard deviation. It is positively skewed for small *n* but it approximately becomes normal for sample sizes greater than 30.

Empirical Risk Minimization(ERM):

The Empirical Risk Minimization (ERM) principle is a learning paradigm which consists in selecting the model with minimal average error over the training set. This so-called training error can be seen as an estimate of the risk (due to the law of large numbers), hence the alternative name of empirical risk.

By minimizing the empirical risk, we hope to obtain a model with a low value of the risk. The larger the training set size is, the closer to the true risk the empirical risk is.

If we were to apply the ERM principle without more care, we would end up learning by heart, which we know is bad. This issue is more generally related to the overfitting phenomenon, which can be avoided by restricting the space of possible models when searching for the one with minimal error. The most severe and yet common restriction is encountered in the contexts of linear classification or linear regression. Another approach consists in controlling the complexity of the model by regularization.

While building our machine learning model, we choose a function that reduces the differences between the actual and the predicted output i.e. empirical risk. We aim to reduce/minimize the empirical risk as an attempt to minimize the true risk by hoping that the empirical risk is almost the same as the true risk.

Empirical risk minimization depends on four factors:

- The size of the dataset - the more data we get, the more the empirical risk approaches the true risk.
- The complexity of the true distribution - if the underlying distribution is too complex, we might need more data to get a good approximation of it.

- The class of functions we consider - the approximation error will be very high if the size of the function is too large.
- The loss function - It can cause trouble if the loss function gives very high loss in certain conditions.

The L2 Regularization is an example of empirical risk minimization.

L2 Regularization

In order to handle the problem of overfitting, we use the regularization techniques. A regression problem using L2 regularization is also known as **ridge regression**.
In ridge regression, the predictors that are insignificant are penalized. This method constricts the coefficients to deal with independent variables that are highly correlated. Ridge regression adds the "squared magnitude" of coefficient, which is the sum of squares of the weights of all features as the penalty term to the loss function.

- $$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda \sum_{i=1}^{N} \theta_i^2$$

  Here, λ is the regularization parameter.

UNIT-2

Supervised Learning: (Regression/Classification):Basic Methods: Distance based Methods, Nearest Neighbours, Decision Trees, Naive Bayes.

Linear Models: Linear Regression, Logistic Regression, Generalized Linear Models, Support Vector Machines.

Binary Classification: Multiclass/Structured outputs, MNIST, Ranking.

Supervised Learning:

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

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

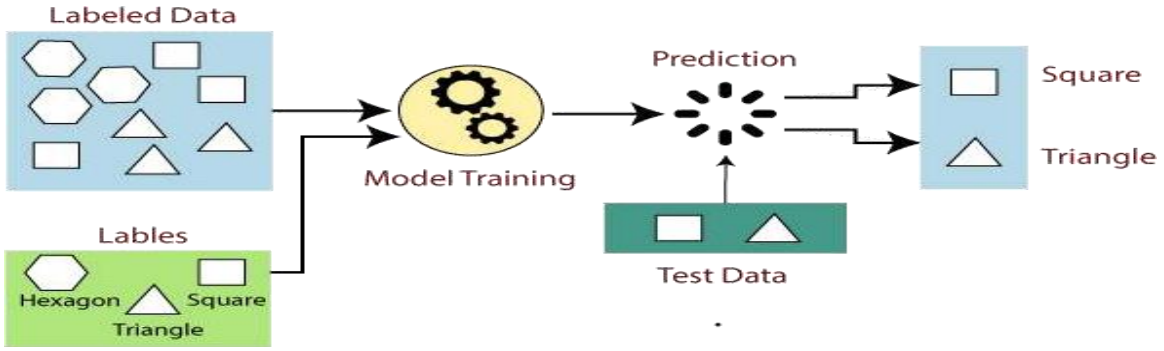
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

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

How Supervised Learning Works?

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

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



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

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

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

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

Steps Involved in Supervised Learning:

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

Types of supervised Machine learning Algorithms:

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

1. Regression

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

- Linear Regression
- Regression Trees
- Non-Linear Regression

- Bayesian Linear Regression
- Polynomial Regression

2. Classification

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

Spam Filtering,

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

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.

Basic Methods:

Distance Based Methods:

Distance measures play an important role in machine learning.

They provide the foundation for many popular and effective machine learning algorithms like k-nearest neighbors for supervised learning and k-means clustering for unsupervised learning.

Different distance measures must be chosen and used depending on the types of the data. As such, it is important to know how to implement and calculate a range of different popular distance measures and the intuitions for the resulting scores.

There are mainly four types of distance based methods in supervised learning. They are:

1. Hamming distance
2. Euclidean distance
3. Manhattan distance
4. Minkowski distance

1. Hamming distance:

Hamming distance calculates the distance between two binary vectors, also referred to as binary strings or bitstrings for short.
You are most likely going to encounter bitstrings when you one-hot encode categorical columns of data.
For example, if a column had the categories '*red*,' '*green*,' and '*blue*,' you might one hot encode each example as a bitstring with one bit for each column.

- red = [1, 0, 0]
- green = [0, 1, 0]
- blue = [0, 0, 1]

The distance between red and green could be calculated as the sum or the average number of bit differences between the two bitstrings. This is the Hamming distance.

For a one-hot encoded string, it might make more sense to summarize to the sum of the bit differences between the strings, which will always be a 0 or 1.

- HammingDistance = sum for i to N abs(v1[i] – v2[i])

For bitstrings that may have many 1 bits, it is more common to calculate the average number of bit differences to give a hamming distance score between 0 (identical) and 1 (all different).

- HammingDistance = (sum for i to N abs(v1[i] – v2[i])) / N

2. Euclidean distance:

Euclidean distance calculates the distance between two real-valued vectors.
You are most likely to use Euclidean distance when calculating the distance between two rows of data that have numerical values, such a floating point or integer values.

If columns have values with differing scales, it is common to normalize or standardize the numerical values across all columns prior to calculating the Euclidean distance. Otherwise, columns that have large values will dominate the distance measure.

Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors.

- EuclideanDistance = sqrt(sum for i to N (v1[i] – v2[i])^2)

If the distance calculation is to be performed thousands or millions of times, it is common to remove the square root operation in an effort to speed up the calculation. The resulting scores will have the same relative proportions after this modification and can still be used effectively within a machine learning algorithm for finding the most similar examples.

- EuclideanDistance = sum for i to N (v1[i] – v2[i])^2

This calculation is related to the L2 vector norm and is equivalent to the sum squared error and the root sum squared error if the square root is added.

3. **Manhattan distance: (Taxicab or City Block distance)**

The Manhattan distance, also called the Taxicab distance or the City Block distance, calculates the distance between two real-valued vectors.
It is perhaps more useful to vectors that describe objects on a uniform grid, like a chessboard or city blocks. The taxicab name for the measure refers to the intuition for what the measure calculates: the shortest path that a taxicab would take between city blocks (coordinates on the grid).

It might make sense to calculate Manhattan distance instead of Euclidean distance for two vectors in an integer feature space.

Manhattan distance is calculated as the sum of the absolute differences between the two vectors.

- ManhattanDistance = sum for i to N sum |v1[i] – v2[i]|

The Manhattan distance is related to the L1 vector norm and the sum absolute error and mean absolute error metric.

4. **Minkowski distance:**

Minkowski distance calculates the distance between two real-valued vectors.
It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the "*order*" or "*p*", that allows different distance measures to be calculated.
The Minkowski distance measure is calculated as follows:

- EuclideanDistance = (sum for i to N (abs(v1[i] – v2[i]))^p)^(1/p)

Where "*p*" is the order parameter.
When p is set to 1, the calculation is the same as the Manhattan distance. When p is set to 2, it is the same as the Euclidean distance.

- *p=1*: Manhattan distance.
- *p=2*: Euclidean distance.

Intermediate values provide a controlled balance between the two measures.

It is common to use Minkowski distance when implementing a machine learning algorithm that uses distance measures as it gives control over the type of distance measure used for real-valued vectors via a hyperparameter "*p*" that can be tuned.

Nearest Neighbor:

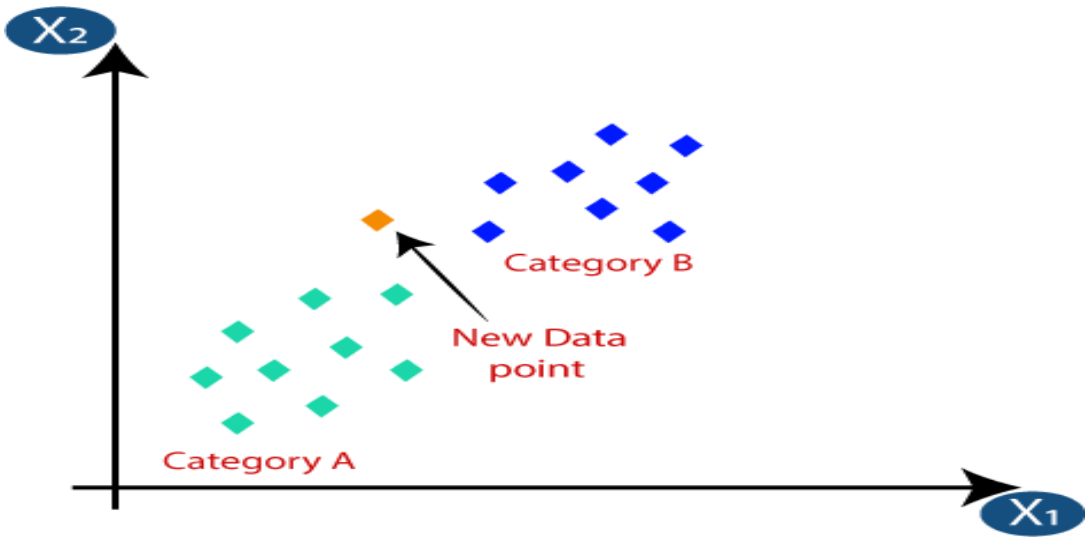
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm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

How does K-NN work?

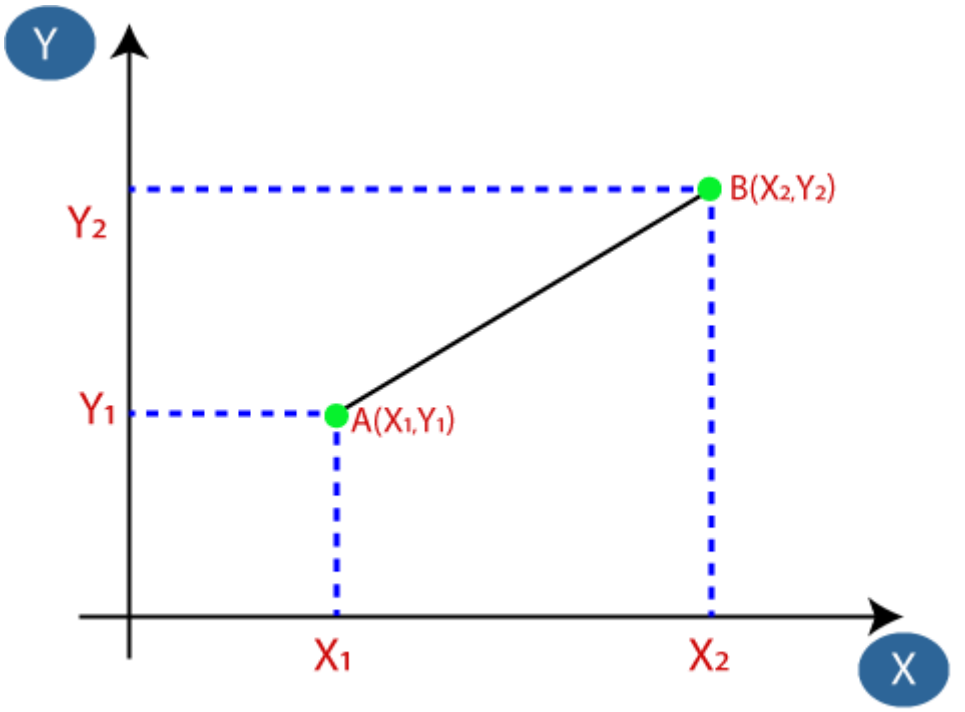
The K-NN working can be explained on the basis of the below algorithm:

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

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



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



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

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

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

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

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

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

Advantages of KNN Algorithm:

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

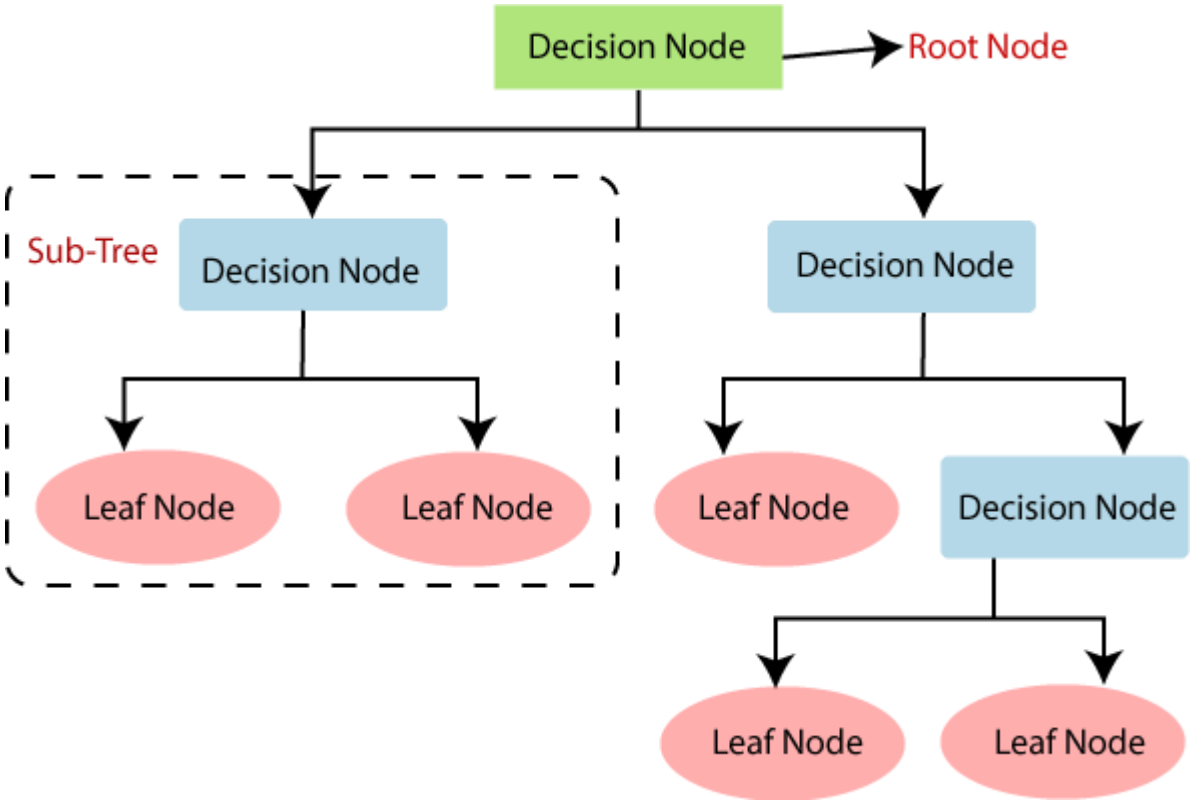
Disadvantages of KNN Algorithm:

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

Decision Trees:

- 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:



Decision Tree Terminologies

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

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

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

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

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

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

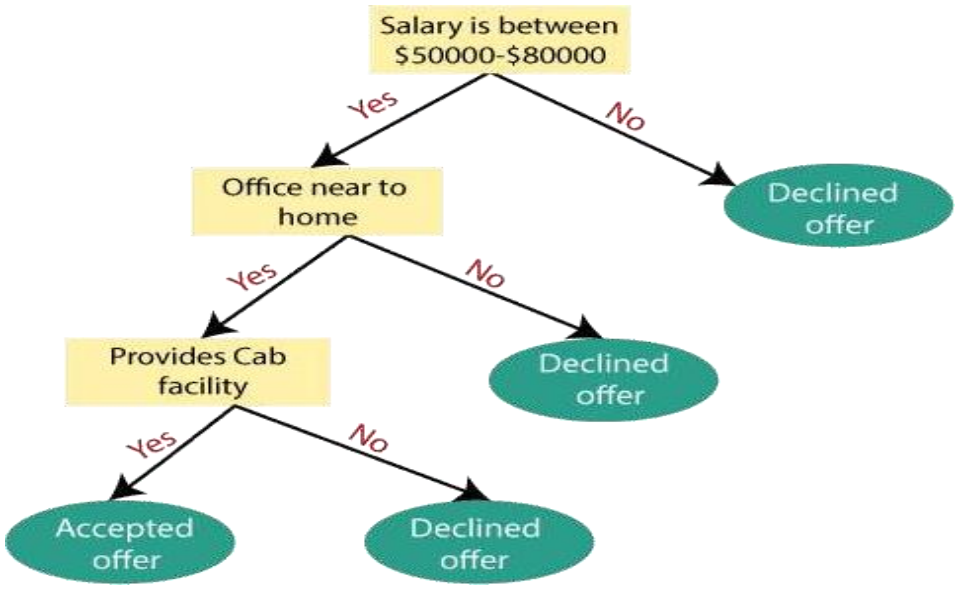
How does the Decision Tree algorithm Work?

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

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

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

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



Attribute Selection Measures

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

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

1. Information Gain:

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

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

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

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

Where,

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

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.

- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

Pruning: Getting an Optimal Decision tree

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

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

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

Advantages of the Decision Tree

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

Disadvantages of the Decision Tree

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

Naïve Bayes:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

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

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

Bayes' Theorem:

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

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

Where,

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

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

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

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

Advantages of Naïve Bayes Classifier:

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

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

Disadvantages of Naïve Bayes Classifier:

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

Applications of Naïve Bayes Classifier:

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

Types of Naïve Bayes Model:

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

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

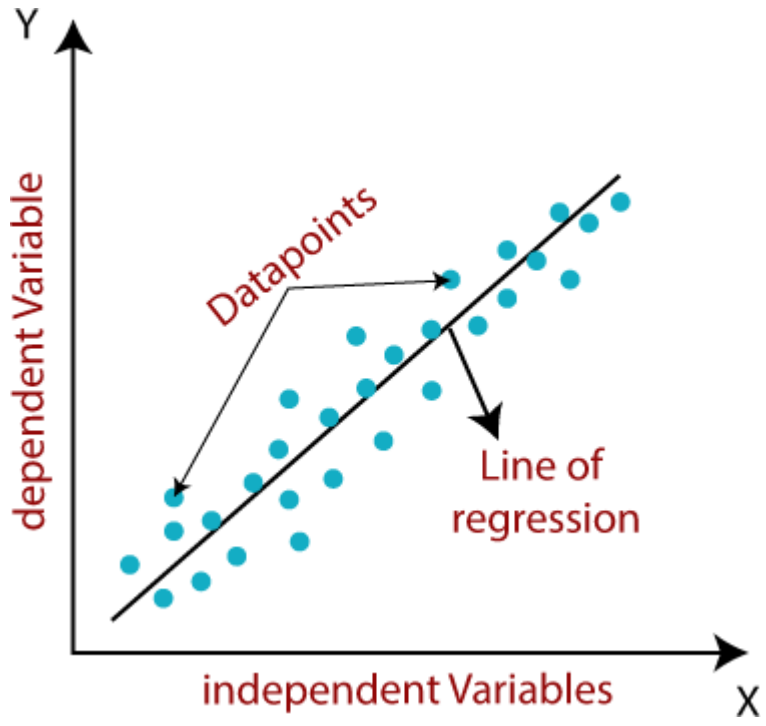
Linear Models:

Linear Regression:

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

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

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



Mathematically, we can represent a linear regression as:

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

Here,

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

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

Types of Linear Regression

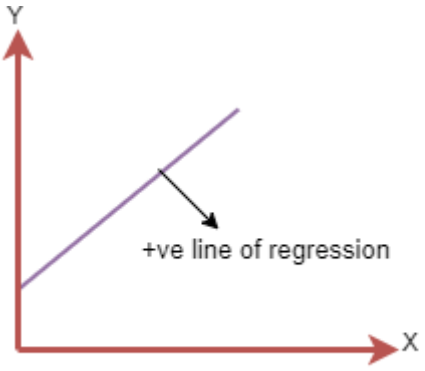
Linear regression can be further divided into two types of the algorithm:

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

Linear Regression Line

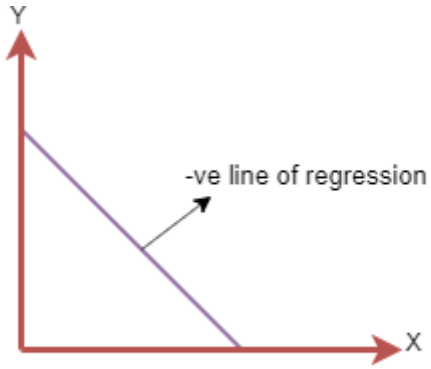
A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

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



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

- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



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

Assumptions of Linear Regression

Below are some important assumptions of Linear Regression. These are some formal checks while building a Linear Regression model, which ensures to get the best possible result from the given dataset.

- **Linear relationship between the features and target:**
  Linear regression assumes the linear relationship between the dependent and independent variables.
- **Small or no multicollinearity between the features:**
  Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.
- **Homoscedasticity Assumption:**
  Homoscedasticity is a situation when the error term is the same for all the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.
- **Normal distribution of error terms:**
  Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.
  It can be checked using the **q-q plot**. If the plot shows a straight line without any deviation, which means the error is normally distributed.

- **No autocorrelations:**
  The linear regression model assumes no autocorrelation in error terms. If there will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

Simple Linear Regression in Machine Learning

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

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

Simple Linear regression algorithm has mainly two objectives:

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

Simple Linear Regression Model:

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

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

Where,

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

Multiple Linear Regression

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

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

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

Example:

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

Some key points about MLR:

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

MLR equation:

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

1. Y= b₀+b₁x₁+ b₂x₂ + b₃x₃+..... bnxn .....(a)

Where,

Y= Output/Response variable

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

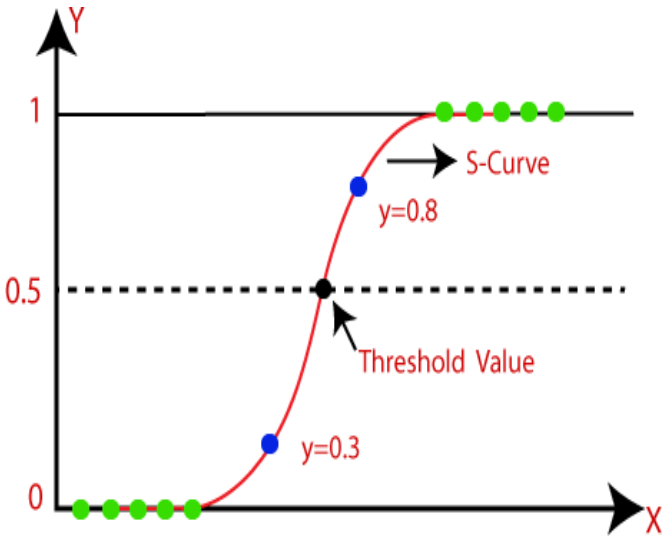
x_1, x_2, x_3, x_4,. = Various Independent/feature variable

Assumptions for Multiple Linear Regression:

- A **linear relationship** should exist between the Target and predictor variables.
- The regression residuals must be **normally distributed**.
- MLR assumes little or **no multicollinearity** (correlation between the independent variable) in data.

Logistic Regression:

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Logistic Function (Sigmoid Function):

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

Assumptions for Logistic Regression:

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

Logistic Regression Equation:

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

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

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

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

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

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

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

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

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

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

Generalized Linear Models:

The term "general" linear model (GLM) usually refers to conventional linear regression models for a continuous response variable given continuous and/or categorical predictors. It includes multiple linear regression, as well as ANOVA and ANCOVA (with fixed effects only). The form is $y_i \sim N(x_i T\beta, \sigma 2)$, where x_i contains known covariates and β contains the coefficients to be estimated. These models are fit by least squares and weighted least squares using, for example, SAS's GLM procedure or R's lm() function.

The term "generalized" linear model (GLIM or GLM) refers to a larger class of models popularized by McCullagh and Nelder (1982, 2nd edition 1989). In these models, the response variable y_i is assumed to follow an exponential family distribution with mean μ_i, which is assumed to be some (often nonlinear) function of $x_i T\beta$. Some would call these “nonlinear” because μ_i is often a nonlinear function of the covariates, but McCullagh and Nelder consider them to be linear because the covariates affect the distribution of y_i only through the linear combination $x_i T\beta$.

The first widely used software package for fitting these models was called GLIM. Because of this program, "GLIM" became a well-accepted abbreviation for generalized linear models, as opposed to "GLM" which often is used for general linear models. Today, GLIMs are fit by many packages, including SAS's Genmod procedure and R's glm() function. Unfortunately, different authors and texts may use GLM to mean either "general" or "generalized" linear model, so it's best to rely on context to determine which is meant.

There are three components to any GLM:

- **Random Component** - specifies the probability distribution of the response variable; e.g., normal distribution for Y in the classical regression model, or binomial distribution for Y in the binary logistic regression model. This is the only random component in the model; there is not a separate error term.

- **Systematic Component** - specifies the explanatory variables $(x1, x2, \ldots, xk)$ in the model, more specifically, their linear combination; e.g., $\beta 0+\beta 1x1+\beta 2x2$, as we have seen in a linear regression, and as we will see in the logistic regression in this lesson.

- **Link Function, η or $g(\mu)$** - specifies the link between the random and the systematic components. It indicates how the expected value of the response relates to the linear combination of explanatory variables; e.g., $\eta = g(E(Yi)) = E(Yi)$ for classical regression, or $\eta = \log_{f0}(\pi 1 - \pi) = \text{logit}(\pi)$ for logistic regression.

Assumptions

- The data Y1,Y2,…,Yn are independently distributed, i.e., cases are independent.
- The dependent variable Yi does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal, etc.).
- A GLM does NOT assume a linear relationship between the response variable and the explanatory variables, but it does assume a linear relationship between the transformed expected response in terms of the link function and the explanatory variables; e.g., for binary logistic regression logit(π)=β0+β1x.
- Explanatory variables can be nonlinear transformations of some original variables.
- The homogeneity of variance does NOT need to be satisfied. In fact, it is not even possible in many cases given the model structure.
- Errors need to be independent but NOT normally distributed.
- Parameter estimation uses maximum likelihood estimation (MLE) rather than ordinary least squares (OLS).

The following are three popular examples of GLMs.

Simple Linear Regression

SLR models how the mean of a continuous response variable Y depends on a set of explanatory variables, where i indexes each observation:

$$\mu i=\beta 0+\beta xi$$

- **Random component** - The distribution of Y has a normal distribution with mean μ and constant variance σ2.
- **Systematic component** - x is the explanatory variable (can be continuous or discrete) and is linear in the parameters β0+β1x. This can be extended to multiple linear regression where we may have more than one explanatory variable, e.g., (x1,x2,…,xk). Also, the explanatory variables themselves could be transformed, e.g., x2, or log(x), provided they are combined with the parameter coefficients in a linear fashion.
- **Link function** - the identity link, η=g(E(Y))=E(Y), is used; this is the simplest link function.

Binary Logistic Regression

Binary logistic regression models how the odds of "success" for a binary response variable Y depend on a set of explanatory variables:

$$\text{logit}(\pi i)=\log(\pi i1-\pi i)=\beta 0+\beta 1xi$$

- **Random component** - The distribution of the response variable is assumed to be binomial with a single trial and success probability E(Y)=π.
- **Systematic component** - x is the explanatory variable (can be continuous or discrete) and is linear in the parameters. As with the above example, this can be extended to multiple variables of non-linear transformations.
- **Link function** - the log-odds or logit link, η=g(π)=log(πi1−πi), is used.

Poisson Regression

models how the mean of a discrete (count) response variable Y depends on a set of explanatory variables

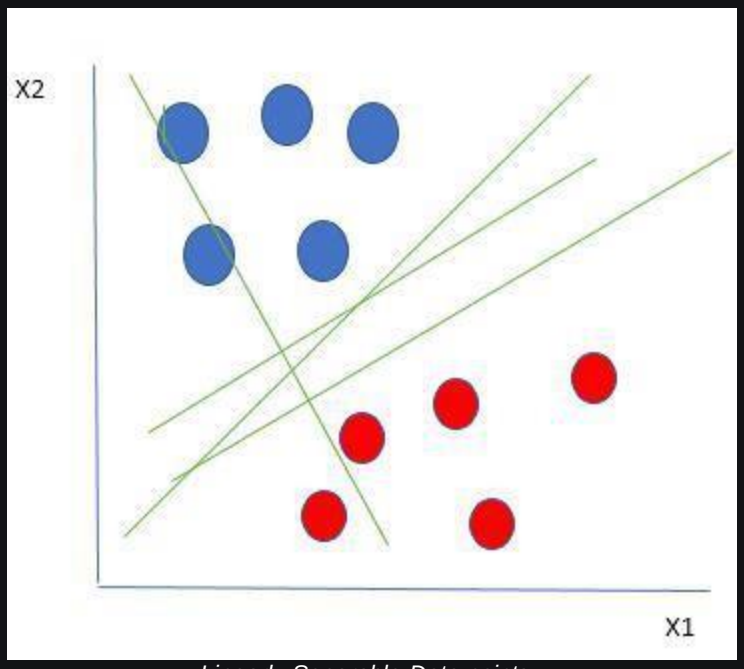
$$\log\lambda i=\beta 0+\beta xi$$

- **Random component** - The distribution of Y is Poisson with mean λ.
- **Systematic component** - x is the explanatory variable (can be continuous or discrete) and is linear in the parameters. As with the above example, this can be extended to multiple variables of non-linear transformations.
- **Link function** - the log link is used.

Support Vector Machines:

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

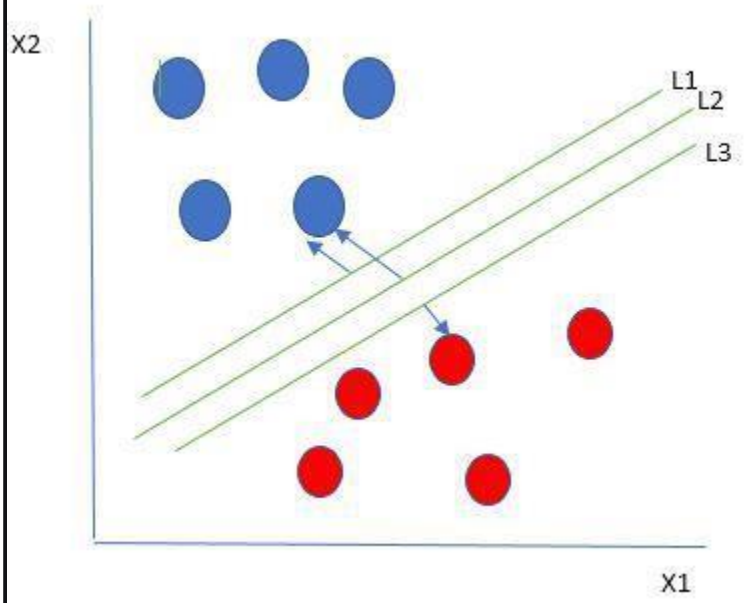
Let's consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red circle.



Linearly Separable Data points

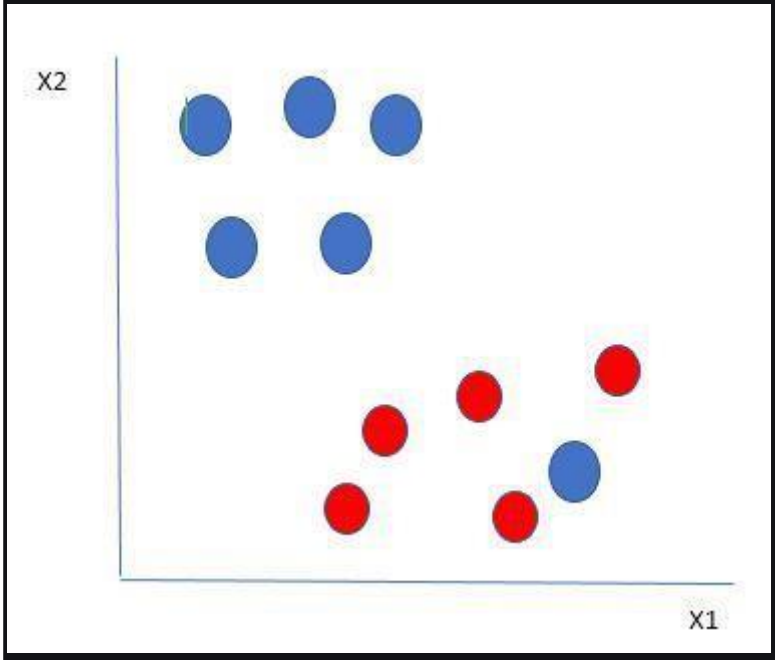
From the figure above its very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregates our data points or does a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points.

Selecting the best hyper-plane:
One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

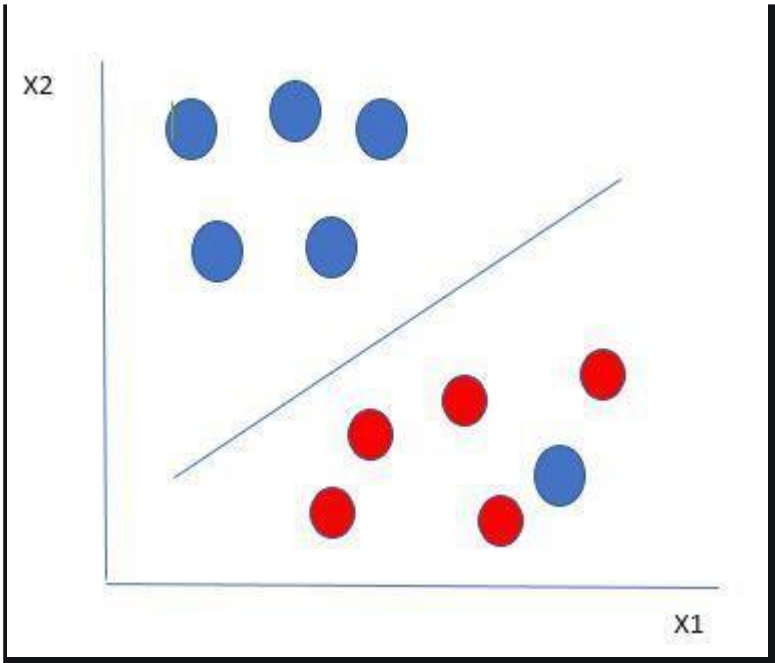


So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from the above figure, we choose L2.

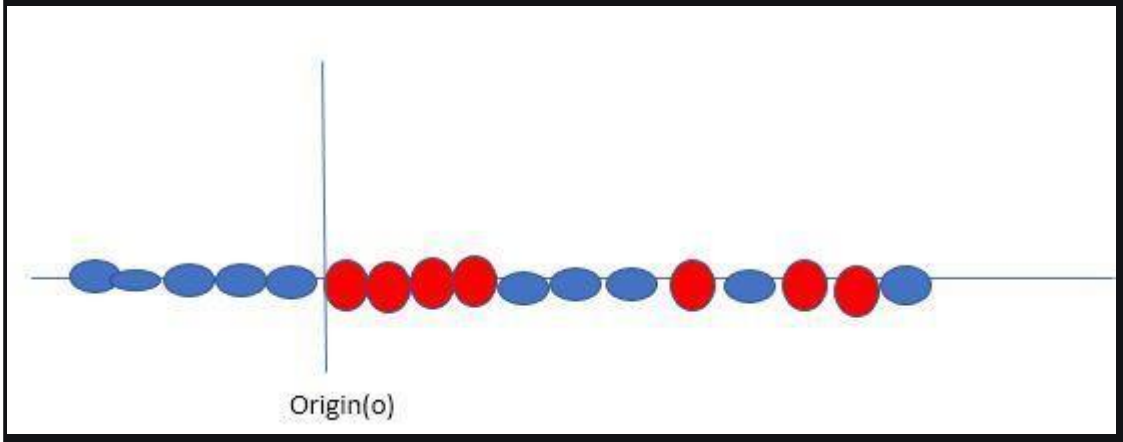
Let’s consider a scenario like shown below



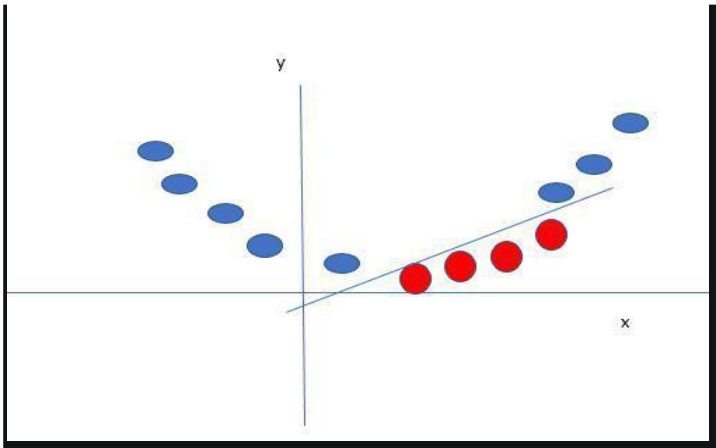
Here we have one blue ball in the boundary of the red ball. The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



So in this type of data points what SVM does is, it finds maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these type of cases are called soft margin. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+A(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss. If violations hinge loss proportional to the distance of violation.



Say, our data is like shown in the figure above. SVM solves this by creating a new variable using a kernel. We call a point x_i on the line and we create a new variable y_i as a function of distance from origin o.so if we plot this we get something like as shown below



In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as kernel.

SVM Kernel:
The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts non separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

Advantages of SVM:

- Effective in high dimensional cases
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

Binary Classification:

Binary classification refers to those classification tasks that have two class labels.
Examples include:

- Email spam detection (spam or not).
- Churn prediction (churn or not).
- Conversion prediction (buy or not).

Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state.

For example "*not spam*" is the normal state and "*spam*" is the abnormal state. Another example is "*cancer not detected*" is the normal state of a task that involves a medical test and "*cancer detected*" is the abnormal state.
The class for the normal state is assigned the class label 0 and the class with the abnormal state is assigned the class label 1.

It is common to model a binary classification task with a model that predicts a Bernoulli probability distribution for each example.
The Bernoulli distribution is a discrete probability distribution that covers a case where an event will have a binary outcome as either a 0 or 1. For classification, this means that the model predicts a probability of an example belonging to class 1, or the abnormal state.

Popular algorithms that can be used for binary classification include:

- Logistic Regression
- k-Nearest Neighbors
- Decision Trees
- Support Vector Machine
- Naive Bayes

Some algorithms are specifically designed for binary classification and do not natively support more than two classes; examples include Logistic Regression and Support Vector Machines.

Multiclass/Structured Outputs Classification:

Multi-class classification refers to those classification tasks that have more than two class labels.
Examples include:

- Face classification.
- Plant species classification.
- Optical character recognition.

Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes.

The number of class labels may be very large on some problems. For example, a model may predict a photo as belonging to one among thousands or tens of thousands of faces in a face recognition system.

Problems that involve predicting a sequence of words, such as text translation models, may also be considered a special type of multi-class classification. Each word in the sequence of words to be predicted involves a multi-class classification where the size of the vocabulary defines the number of possible classes that may be predicted and could be tens or hundreds of thousands of words in size.

It is common to model a multi-class classification task with a model that predicts a Multinoulli probability distribution for each example.

The Multinoulli distribution is a discrete probability distribution that covers a case where an event will have a categorical outcome, e.g. K in {1, 2, 3, …, K}. For classification, this means that the model predicts the probability of an example belonging to each class label.
Many algorithms used for binary classification can be used for multi-class classification.

Popular algorithms that can be used for multi-class classification include:

- k-Nearest Neighbors.
- Decision Trees.
- Naive Bayes.
- Random Forest.
- Gradient Boosting.

Algorithms that are designed for binary classification can be adapted for use for multi-class problems.

This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

- **One-vs-Rest**: Fit one binary classification model for each class vs. all other classes.
- **One-vs-One**: Fit one binary classification model for each pair of classes.

Binary classification algorithms that can use these strategies for multi-class classification include:

- Logistic Regression.
- Support Vector Machine.

MNIST:

Using the MNIST dataset, which is a set of 70,000 small images of digits handwritten by high school students and employees of the US Cen- sus Bureau. Each image is labeled with the digit it represents. This set has been stud- ied so much that it is often called the “Hello World” of Machine Learning: whenever people come up with a new classification algorithm, they are curious to see how it will perform on MNIST. Whenever someone learns Machine Learning, sooner or later they tackle MNIST. Scikit-Learn provides many helper functions to download popular datasets. MNIST is one of them. The following code fetches the MNIST dataset:

```
>>> from sklearn.datasets import fetch_openml

>>> mnist = fetch_openml('mnist_784', version=1)
```

>>> mnist.keys() dict_keys(['data', 'target', 'feature_names', 'DESCR', 'details', 'categories', 'url'])

Datasets loaded by Scikit-Learn generally have a similar dictionary structure including:

• A DESCR key describing the dataset

• A data key containing an array with one row per instance and one column per feature

• A target key containing an array with the labels

Let's look at these arrays:

```
>>> X, y = mnist["data"], mnist["target"]
>>> X.shape
(70000, 784)
>>> y.shape
(70000,)
```

There are 70,000 images, and each image has 784 features. This is because each image is 28×28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black). Let's take a peek at one digit from the dataset. All you need to do is grab an instance's feature vector, reshape it to a 28×28 array, and display it using Matplotlib's imshow() function:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
some_digit = X[0]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap = mpl.cm.binary, interpolation="nearest")
plt.axis("off")
plt.show()
```

This looks like a 5, and indeed that's what the label tells us:

```
>>> y[0]
'5'
```

Note that the label is a string. We prefer numbers, so let's cast y to integers:

```
>>> y = y.astype(np.uint8)
```

A few digits from the MNIST dataset

Ranking:

There are many different types of ranking algorithms, each with its own set of advantages and disadvantages. Some of the most common types of ranking algorithms are:

Binary Ranking Algorithms: Binary ranking algorithms are the simplest type of ranking algorithm. A binary ranking algorithm ranks items in a dataset according to their relative importance. The two most common types of binary ranking algorithms are the rank-by-feature and the rank-by-frequency algorithms. Rank-by-feature algorithms rank items by the number of features that they have in common with the reference item. The reference item is the item that is used to calculate the similarity value for each of the other items in the dataset. Rank-by-frequency algorithms rank items by the number of times that they occur in the dataset. Both rank-by-feature and rank-by-frequency algorithms have their own set of advantages and disadvantages. Rank-by-feature algorithms are more accurate than rank-by-frequency algorithms, but they are also more computationally expensive. Rank-by-frequency algorithms are faster than rank-by-feature algorithms, but they are less accurate.

Ranking by Similarity: Ranking by similarity is a type of probabilistic ranking algorithm that ranks items in a dataset according to their similarity to a reference item. The reference item is the item that is used to calculate the similarity value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the

relevance score, the more similar the item is to the reference item. There are many different types of ranking by similarity algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by similarity algorithms are clustering ranking algorithm, vector space ranking algorithm, etc.

Ranking by Distance: Ranking by distance algorithms are a type of probabilistic ranking algorithm that rank items in a dataset according to their distance from a reference item. The reference item is the item that is used to calculate the distance value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more distant the item is from the reference item. There are many different types of ranking by distance algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by distance algorithms are Euclidean distance algorithm, Mahalanobis distance algorithm, etc.

Ranking by Preference: Preferential ranking algorithms are a type of probabilistic ranking algorithm that rank items in a dataset according to their preference for a reference item. The reference item is the item that is used to calculate the preference value for each of the other items in the dataset. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more preferred the item is for the reference item.

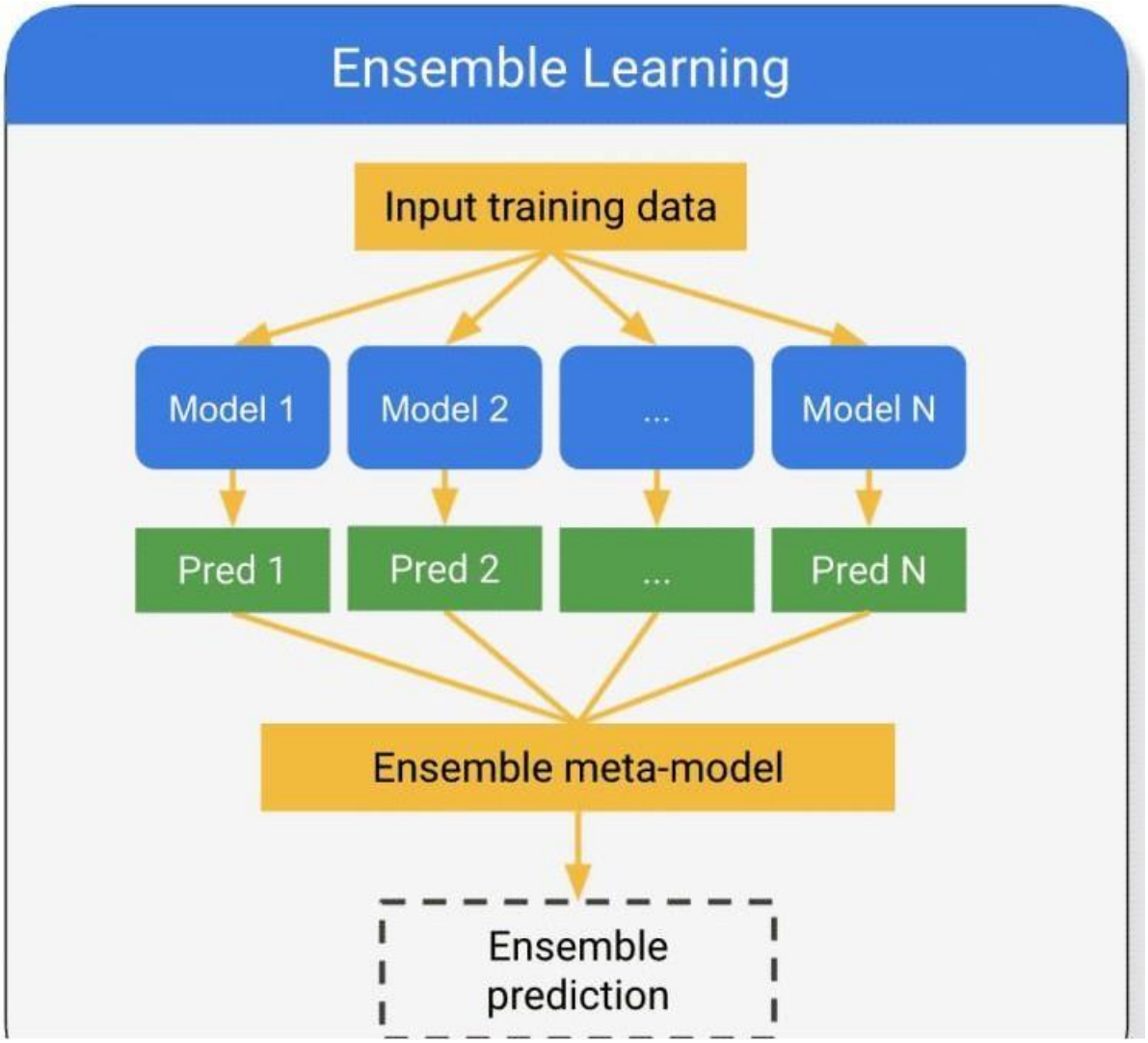
Ranking by Probability: Ranking by probability is a type of probabilistic ranking algorithm that ranks items in a dataset according to their probability of being a positive example. The ranking algorithm uses the input data, such as the number of features that are common to both positive and negative examples, to calculate the item's relevance score. The higher the relevance score, the more likely the item is to be a positive example. Ranking by probability is different from other types of ranking algorithms because it takes into account the uncertainty of the data. This makes it more accurate than other types of ranking algorithms. There are many different types of ranking by probability algorithms, each with its own set of advantages and disadvantages. Some common types of ranking by probability algorithms are Bayesian Ranking Algorithm, AUC Ranking Algorithm, etc.

UNIT- 3

Ensemble Learning and Random Forests: Introduction, Voting Classifiers, Bagging and Pasting, Random Forests, Boosting, Stacking. Support Vector Machine: Linear SVM Classification, Nonlinear SVM Classification SVM Regression, Naïve Bayes Classifiers

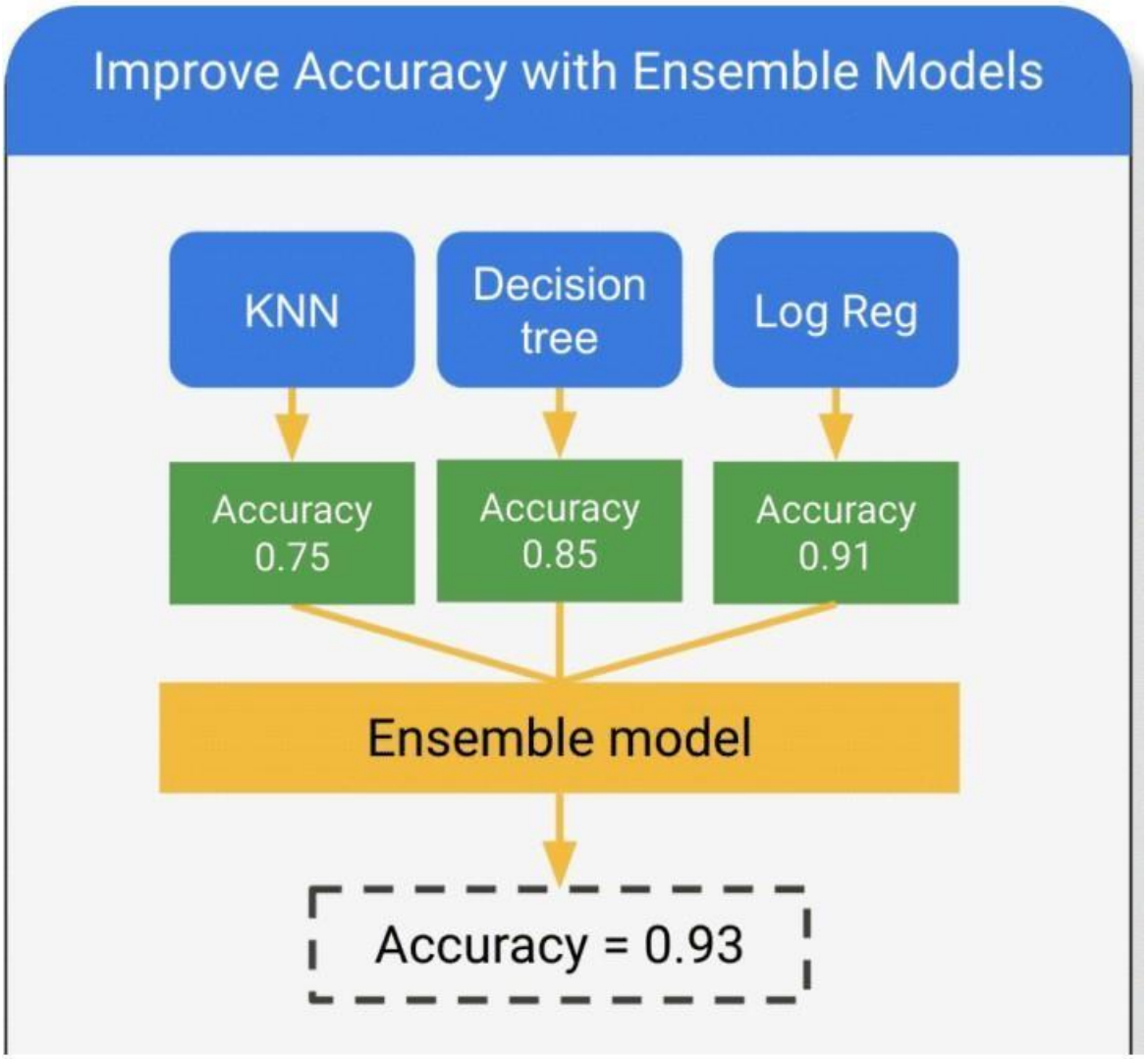
Ensemble Learning:

Ensemble learning is a supervised learning technique used in machine learning to improve overall performance by combining the predictions from multiple models.



How does Ensemble Learning Work?

Ensemble learning works on the principle of the "wisdom of the crowd". By combing multiple models, we can improve the accuracy of the predictions.



Types of Ensemble Methods

• Voting

• Bootstrap aggregation (bagging)

• Random Forests

• Boosting

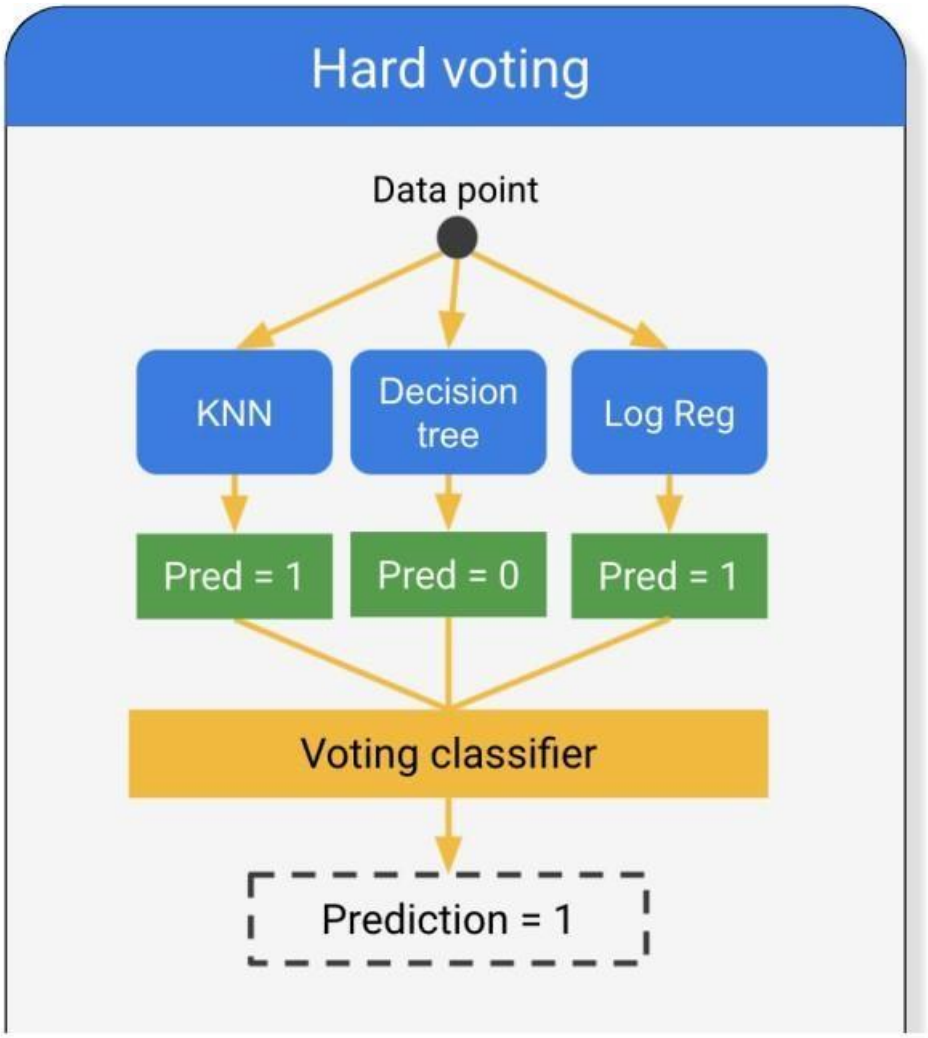
• Stacked Generalization (Blending)

1. Voting Classifiers:

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.
It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

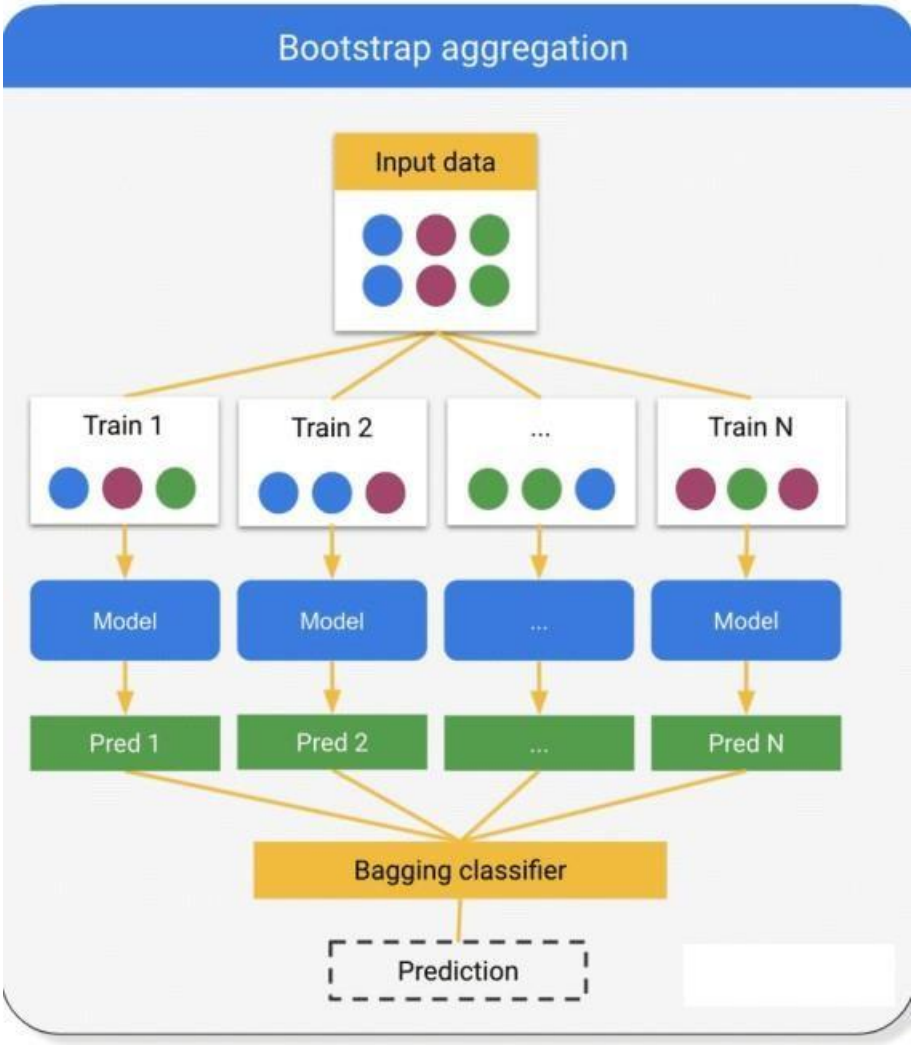
Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the *output class(A, A, B)*, so here the majority predicted *A* as output. Hence *A* will be the final prediction.
2. **Soft Voting:** In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class *A = (0.30, 0.47, 0.53)* and *B = (0.20, 0.32, 0.40)*. So the average for class *A is 0.4333* and *B is 0.3067*, the winner is clearly class *A* because it had the highest probability averaged by each classifier.



2. Bagging and Pasting:

Bagging, or bootstrap aggregation, is an ensemble method that reduces the variance of individual models by fitting a decision tree on different bootstrap samples of a training set.



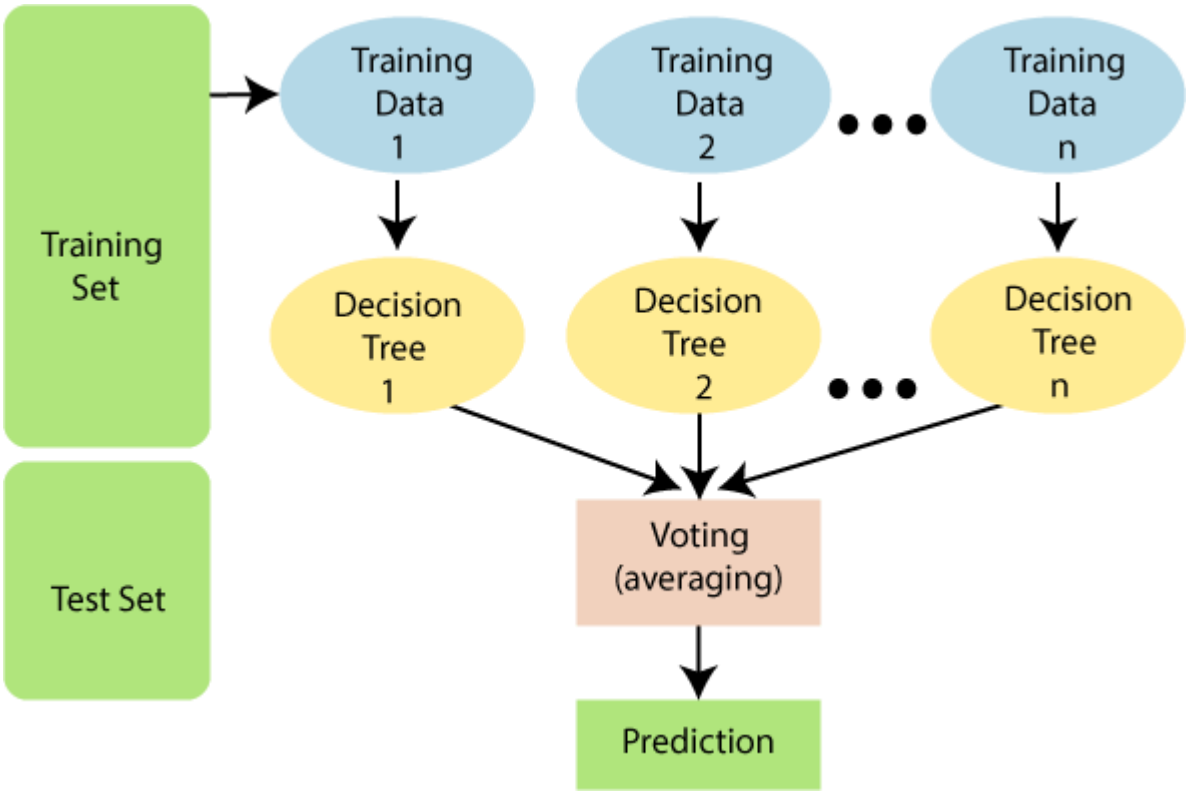
3. Random Forests:

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

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

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

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

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

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

How does Random Forest algorithm work?

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

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

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

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

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

Step-4: Repeat Step 1 & 2.

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

Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

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

Disadvantages of Random Forest

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

4. Boosting:

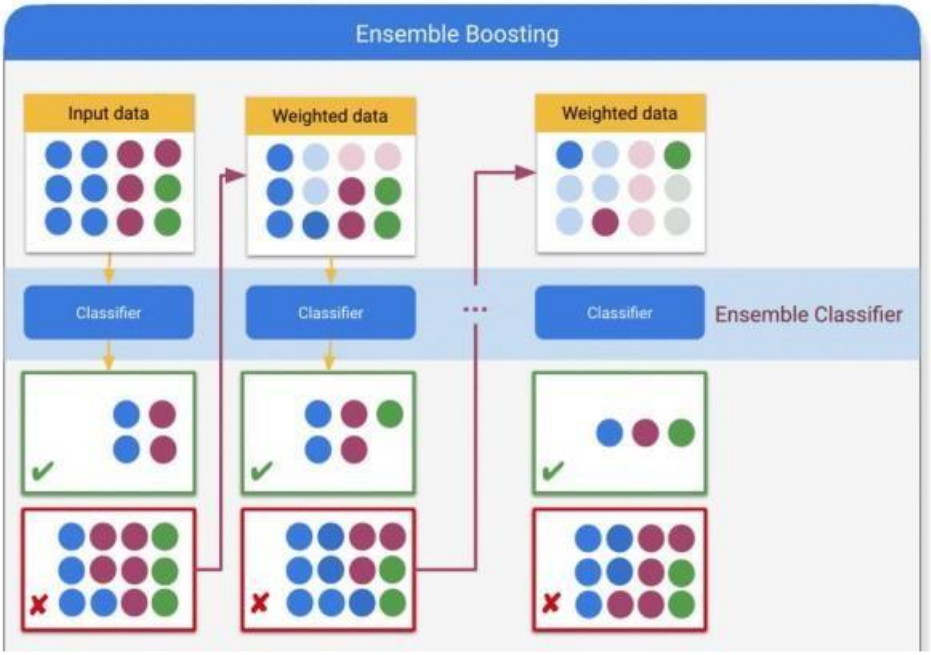
Boosting is an ensemble method that converts weak learners into strong learners by having each predictor fix the errors of its predecessor.

Boosting can be used in classification and regression problems. Boosting machine learning algorithms work by:

• Instantiating a weak learner (e.g. CART with max_depth of 1)

• Making a prediction and passing the wrong predictions to the next predictor

• Paying more and more attention at each iteration to the observations. having prediction errors
• Making new prediction until the limit is reached or the higher accuracy is achieved



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

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

Gradient Boosting is an expansion of the boosting procedure.

Gradient Boosting = Gradient Descent + Boosting

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

Advantages of using Gradient Boosting methods:

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

Disadvantages of using a Gradient Boosting methods:

- It requires cautious tuning of different hyper-parameters.

Multiple boosting Algorithms

• Gradient Boosting: Gradient boosting machines, Gradient Boosted Regression Trees

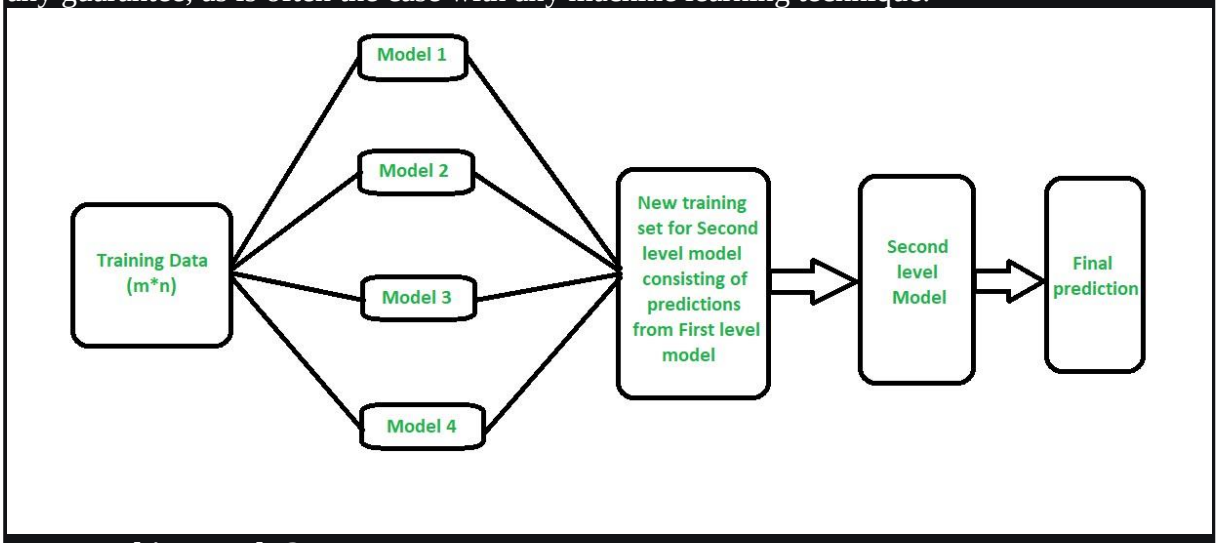
• Adaboost • XGBoost

4. Stacking:

Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are ***Bagging*** or ***Boosting***. Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small.

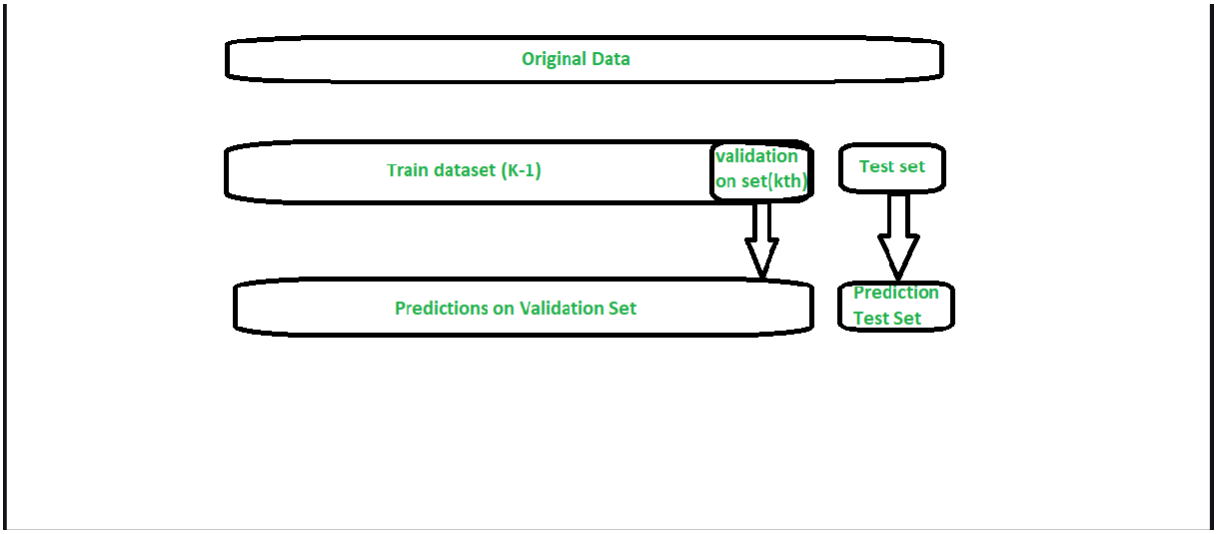
Stacking (sometimes called *Stacked Generalization*) is a different paradigm. The point of stacking is to explore a space of different models for the same problem. The idea is that you can attack a learning problem with different types of models which are capable to learn some part of the problem, but not the whole space of the problem. So, you can build multiple different learners and you use them to build an intermediate prediction, one prediction for each learned model. Then you add a new model which learns from the intermediate predictions the same target.

This final model is said to be stacked on the top of the others, hence the name. Thus, you might improve your overall performance, and often you end up with a model which is better than any individual intermediate model. Notice however, that it does not give you any guarantee, as is often the case with any machine learning technique.



How stacking works?

1. We split the training data into K-folds just like K-fold cross-validation.
2. A base model is fitted on the K-1 parts and predictions are made for Kth part.
3. We do for each part of the training data.
4. The base model is then fitted on the whole train data set to calculate its performance on the test set.
5. We repeat the last 3 steps for other base models.
6. Predictions from the train set are used as features for the second level model.
7. Second level model is used to make a prediction on the test set.



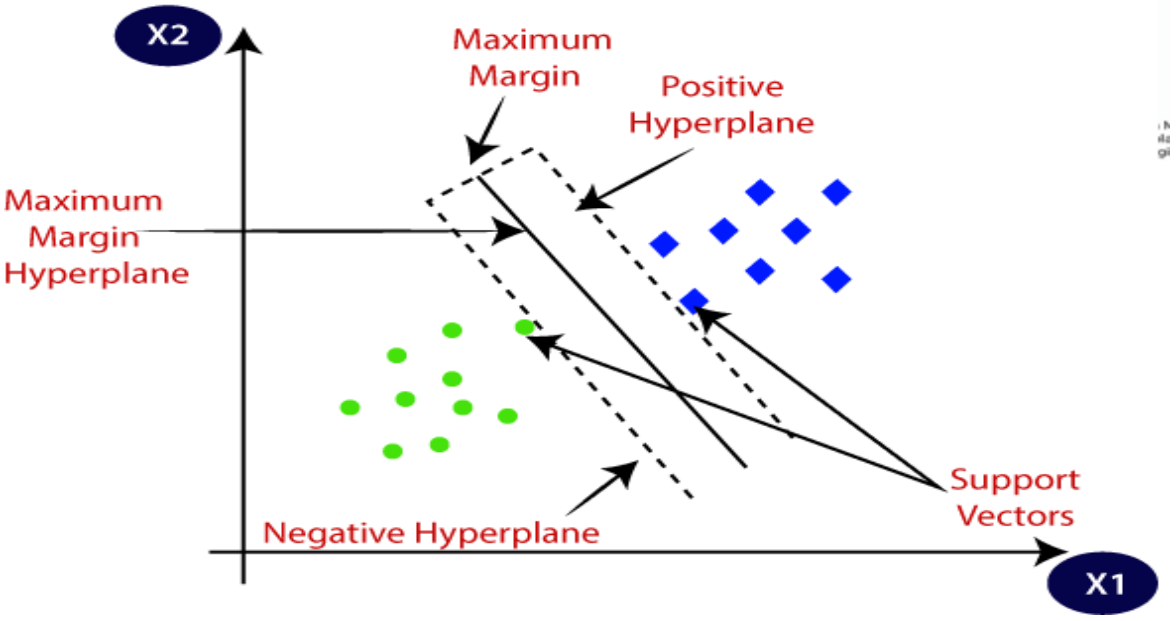
Support Vector Machines:

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

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

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

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



Types of SVM

SVM can be of two types:

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

Hyperplane and Support Vectors in the SVM algorithm:

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

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

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

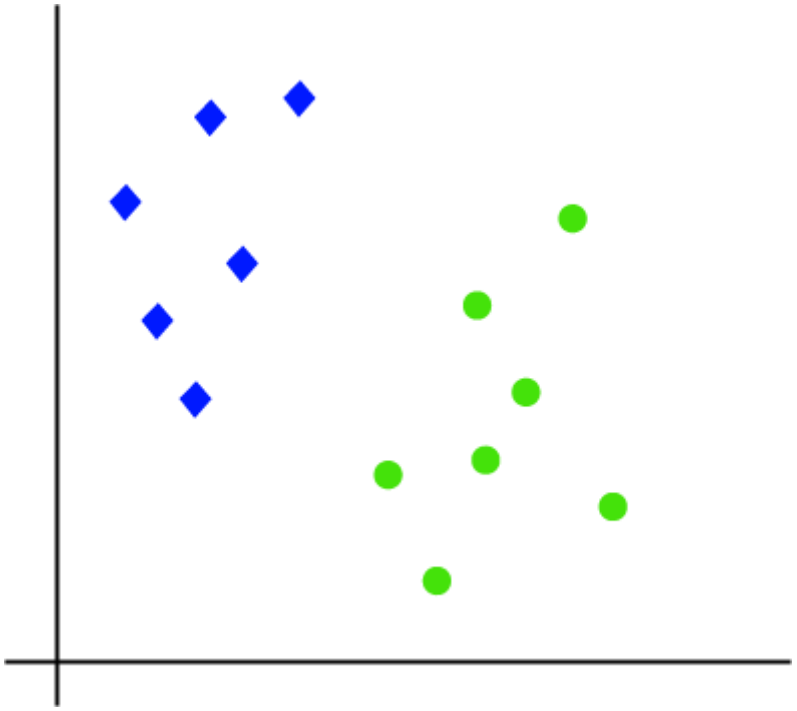
Support Vectors:

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

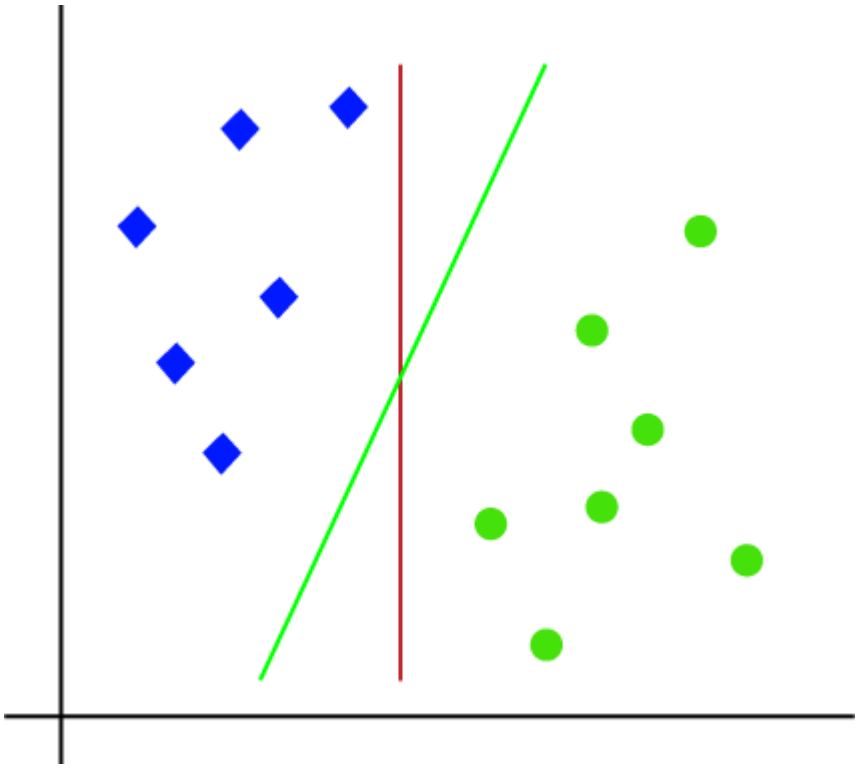
How does SVM works?

Linear SVM:

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

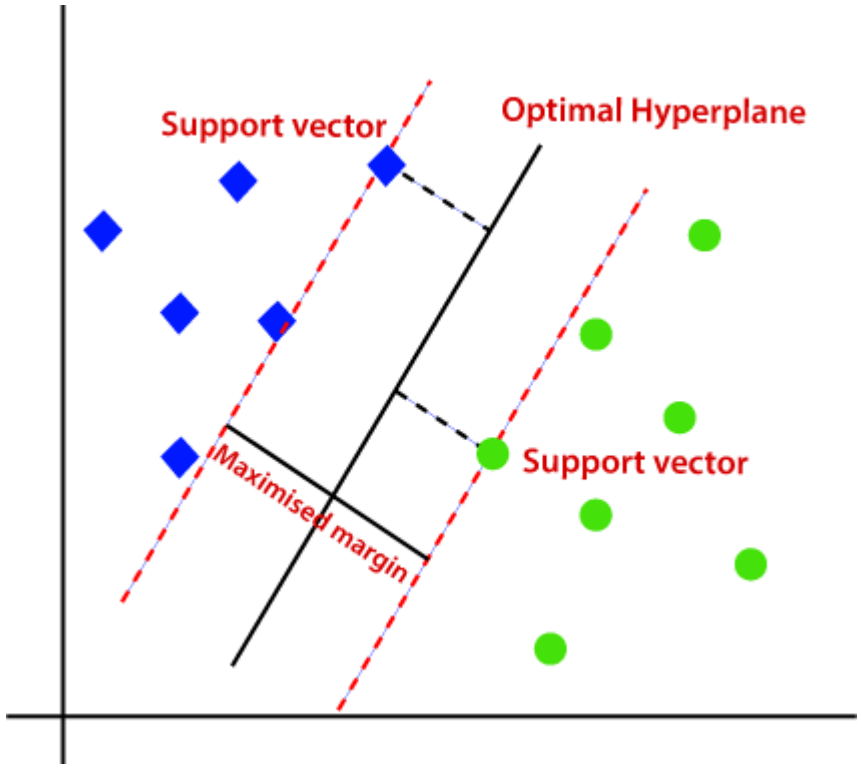


So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



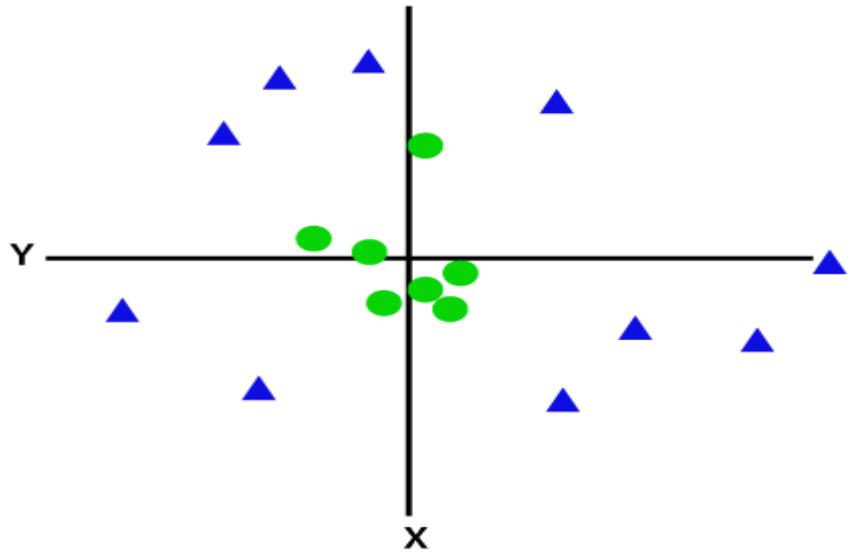
Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and

the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

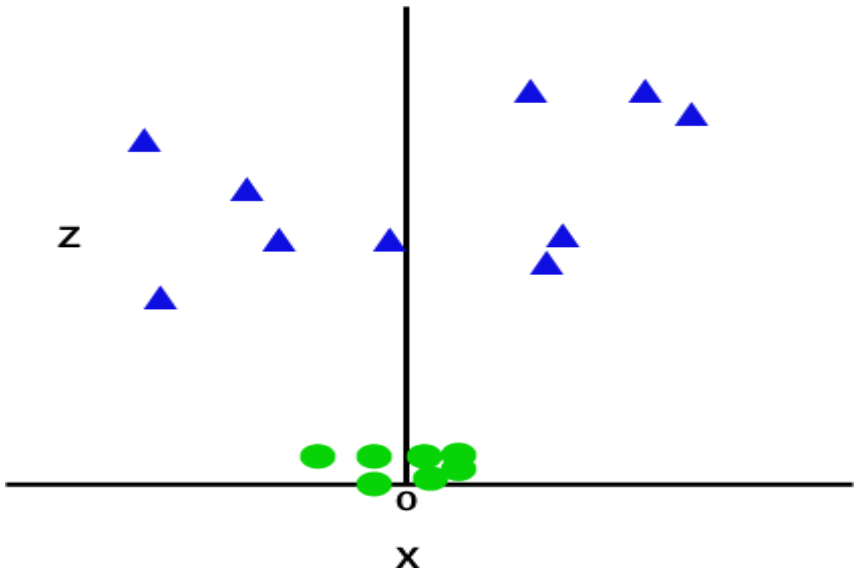
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



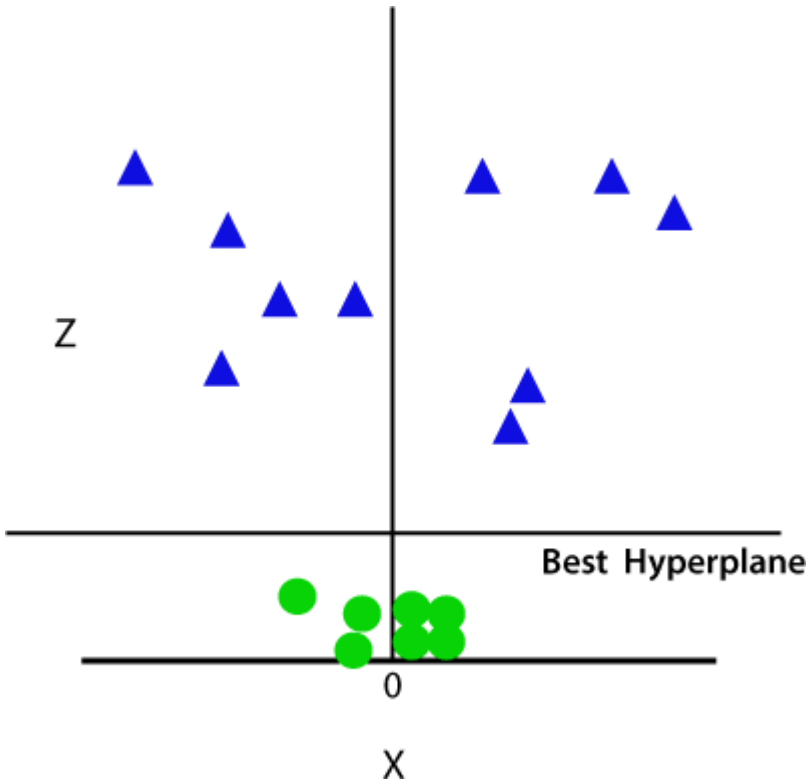
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

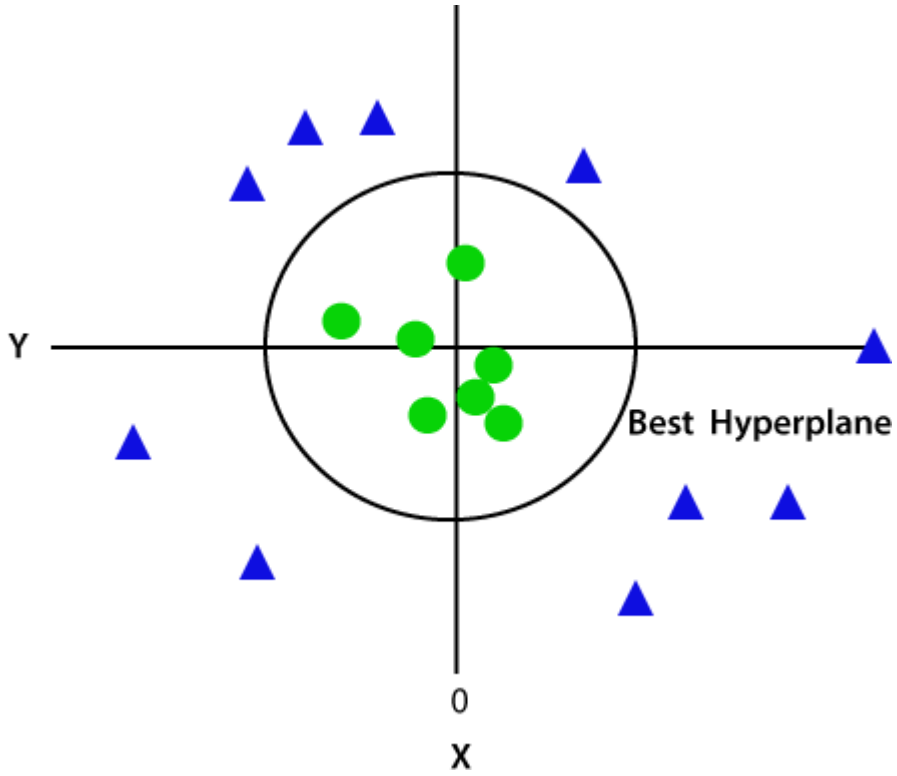
By adding the third dimension, the sample space will become as below image:



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



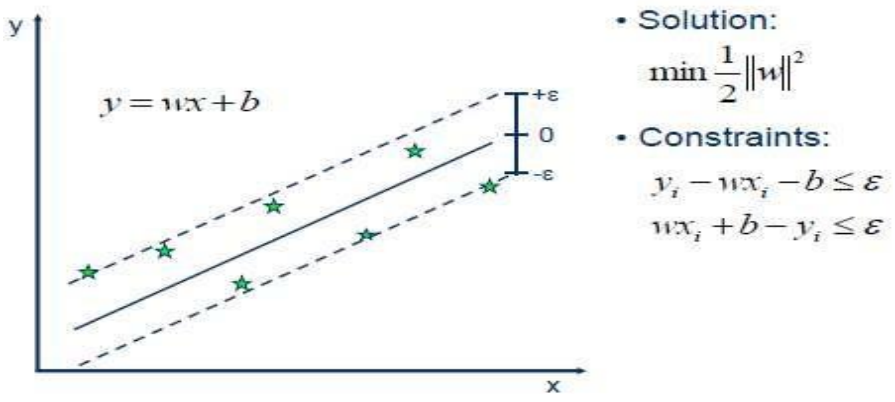
Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

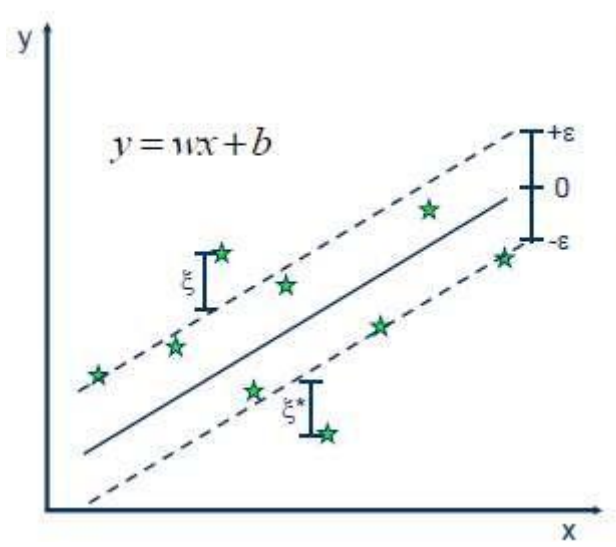


Hence we get a circumference of radius 1 in case of non-linear data.

SVM Regression:

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.





• Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

• Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Linear SVR

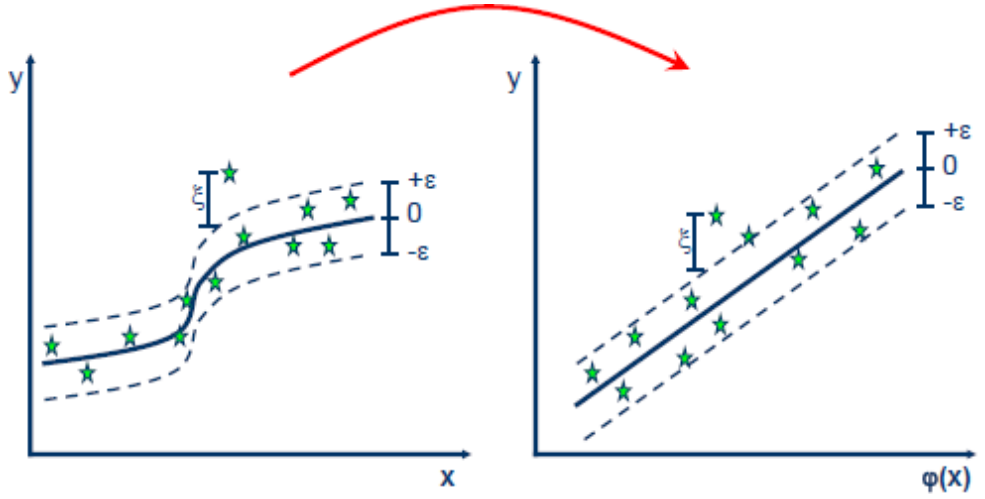
$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\left\langle x_i, x\right\rangle + b$$

Non-linear SVR

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\left\langle \varphi(x_i), \varphi(x)\right\rangle + b$$

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot K(x_i, x) + b$$



Kernel functions

Polynomial

$$k(x_i, x_j) = (x_i . x_j)^d$$

Gaussian Radial Basis function

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Naïve Bayes Classifier:

Naïve Bayes theorem is also a supervised algorithm, which is based on Bayes theorem and used to solve classification problems. It is one of the most simple and effective classification algorithms in Machine Learning which enables us to build various ML models for quick predictions. It is a probabilistic classifier that means it predicts on the basis of probability of an object. Some popular Naïve Bayes algorithms are **spam filtration, Sentimental analysis, and classifying articles.**

Advantages of Naïve Bayes Classifier in Machine Learning:

- It is one of the simplest and effective methods for calculating the conditional probability and text classification problems.
- A Naïve-Bayes classifier algorithm is better than all other models where assumption of independent predictors holds true.
- It is easy to implement than other models.
- It requires small amount of training data to estimate the test data which minimize the training time period.
- It can be used for Binary as well as Multi-class Classifications.

Disadvantages of Naïve Bayes Classifier in Machine Learning:

The main disadvantage of using Naïve Bayes classifier algorithms is, it limits the assumption of independent predictors because it implicitly assumes that all attributes are independent or unrelated but in real life it is not feasible to get mutually independent attributes.

UNIT-4

Unsupervised Learning Techniques: Clustering, K-Means, Limits of K-Means, Using Clustering for Image Segmentation, Using Clustering for Preprocessing, Using Clustering for Semi- Supervised Learning, DBSCAN, Gaussian Mixtures. Dimensionality Reduction: The Curse of Dimensionality, Main Approaches for Dimensionality Reduction, PCA, Using Scikit-Learn, Randomized PCA, Kernel PCA

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**.

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.

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.

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.

Clustering:

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

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

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

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

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

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

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

Types of Clustering Methods

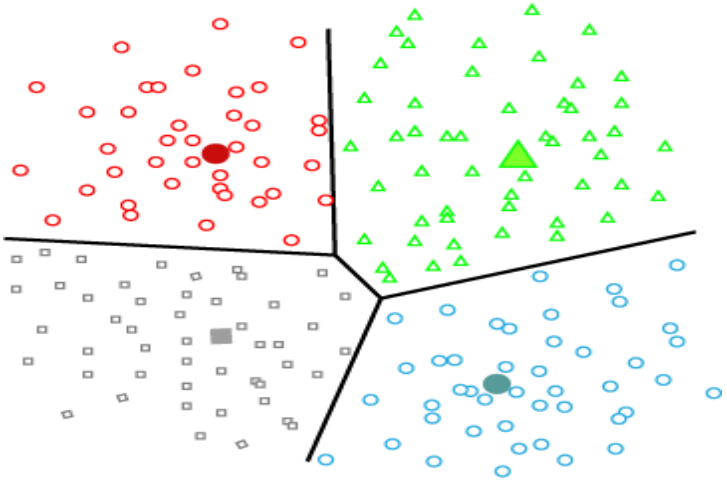
The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

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

Partitioning Clustering

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

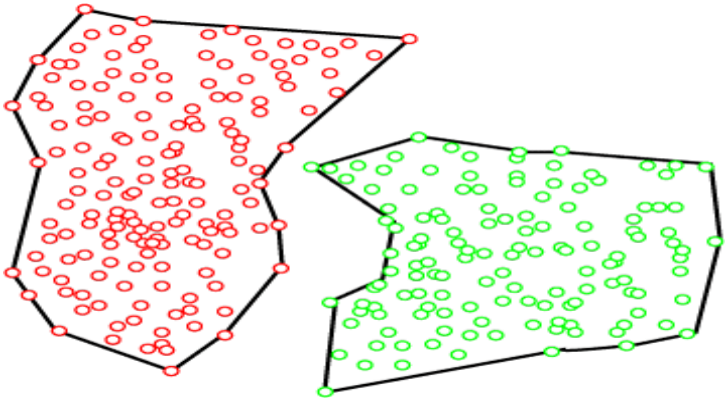
In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

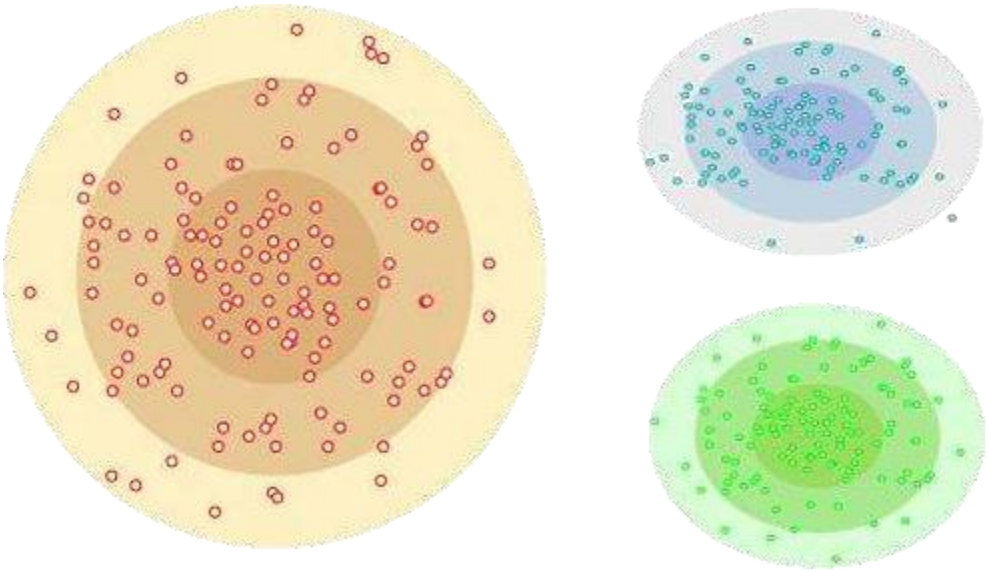
These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Distribution Model-Based Clustering

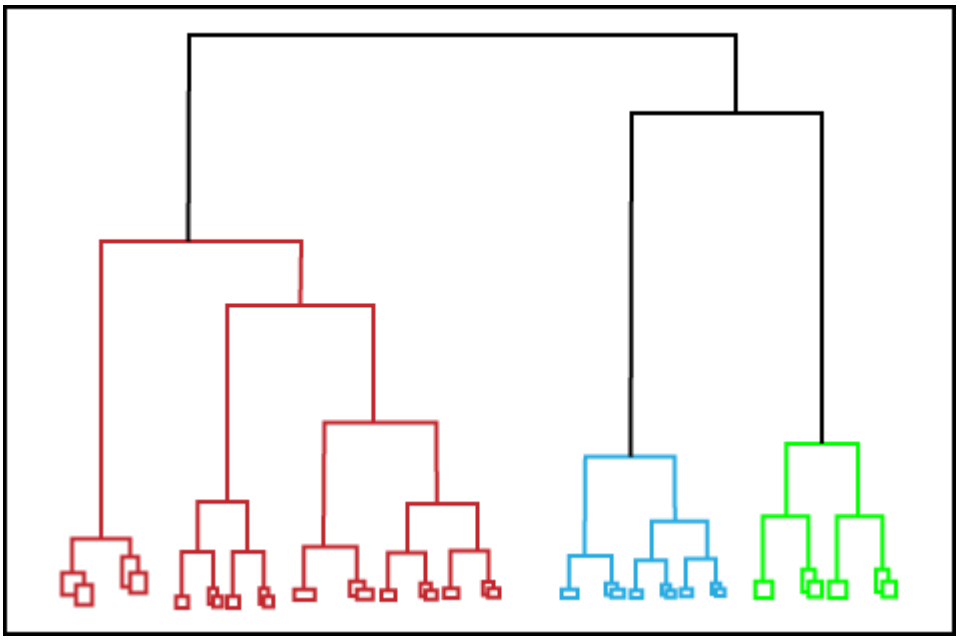
In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

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



Hierarchical Clustering

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



Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Applications of Clustering

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

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

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

K-Means:

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

What is K-Means Algorithm?

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

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

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

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

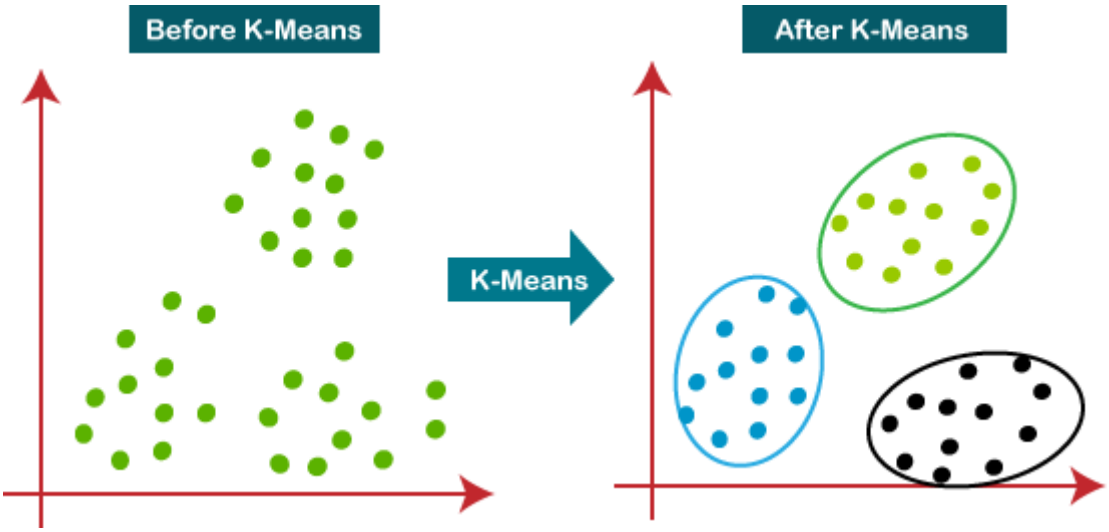
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

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

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

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



How does the K-Means Algorithm Work?

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

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

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

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

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

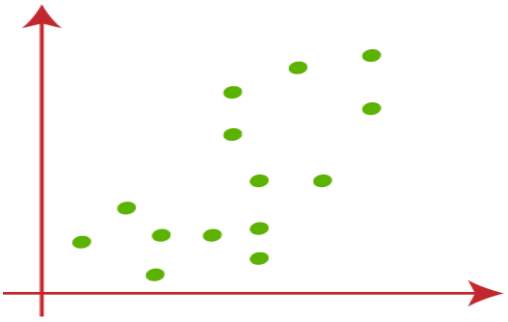
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

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

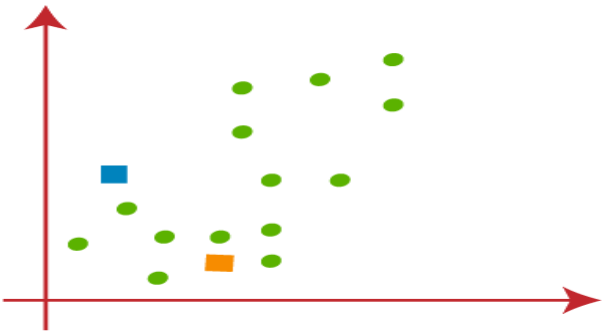
Step-7: The model is ready.

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

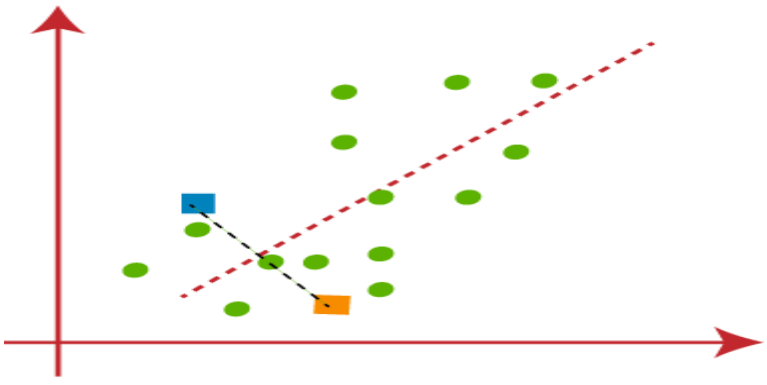
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



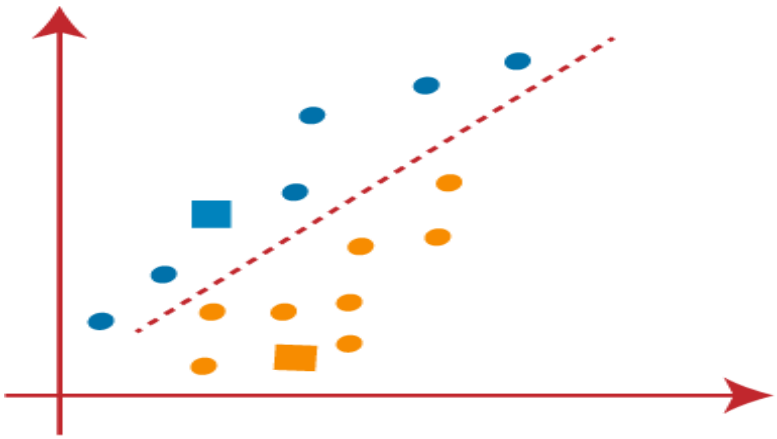
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points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



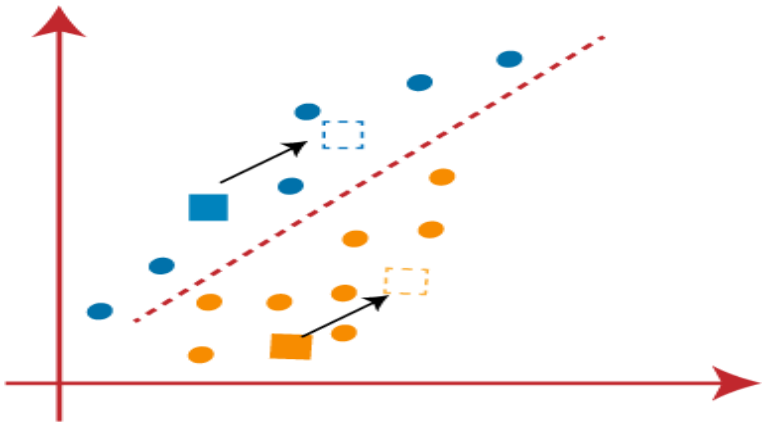
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



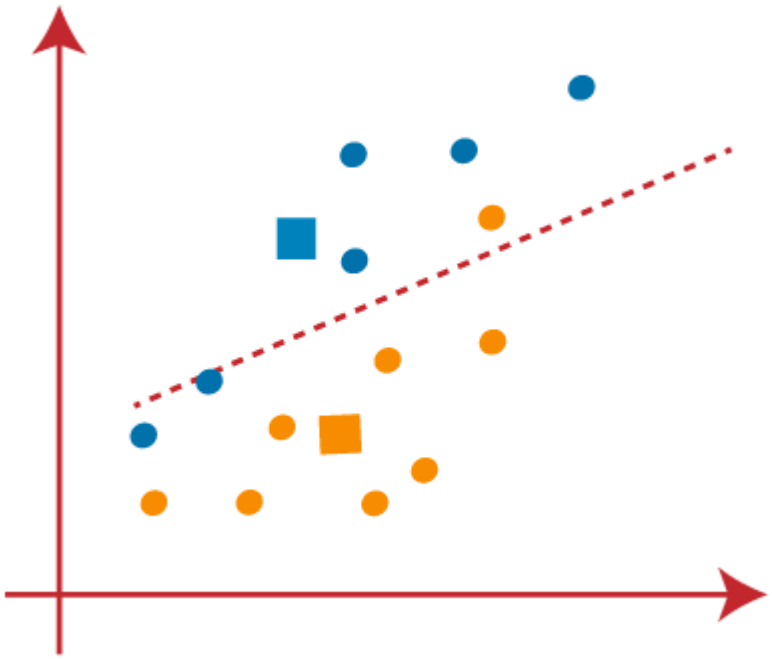
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



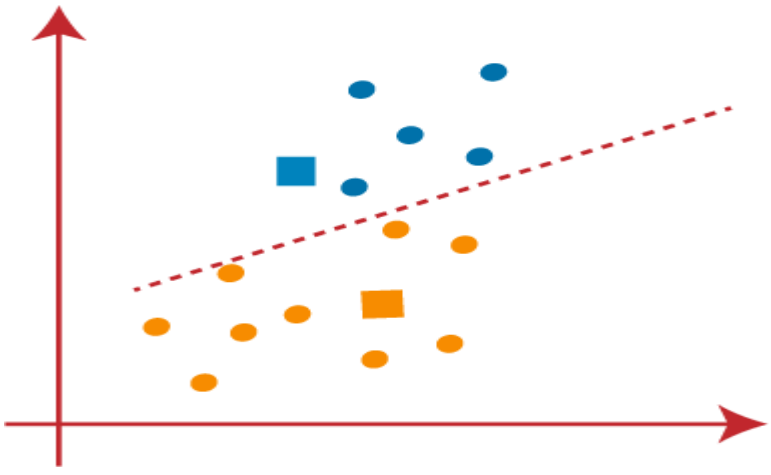
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



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

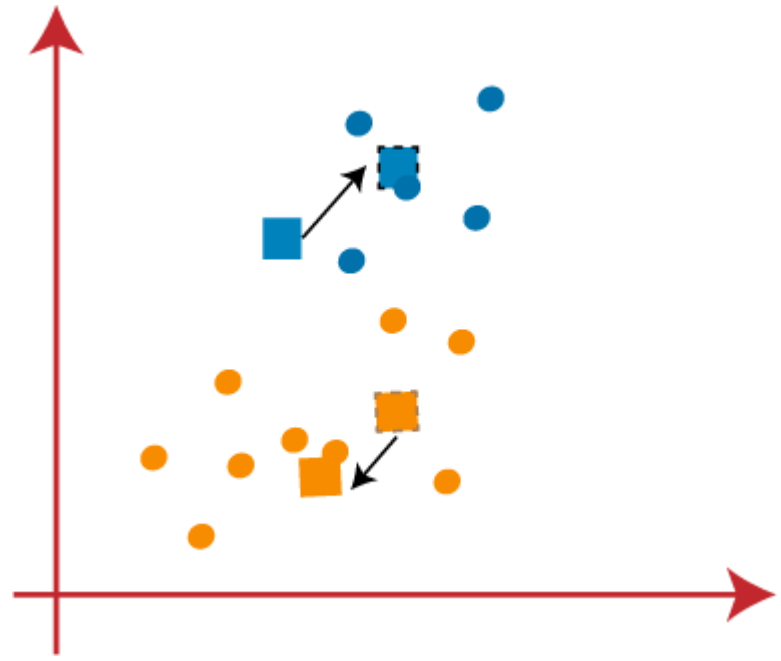


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

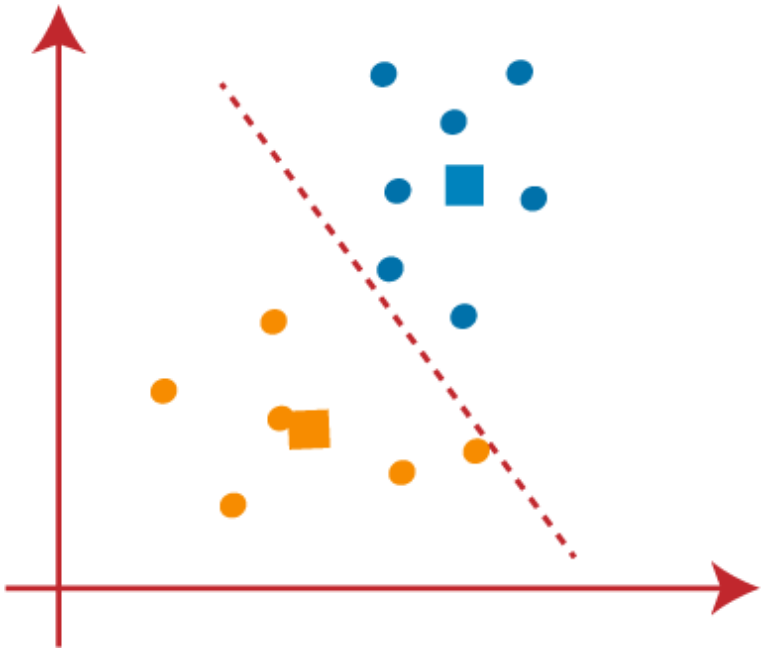


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

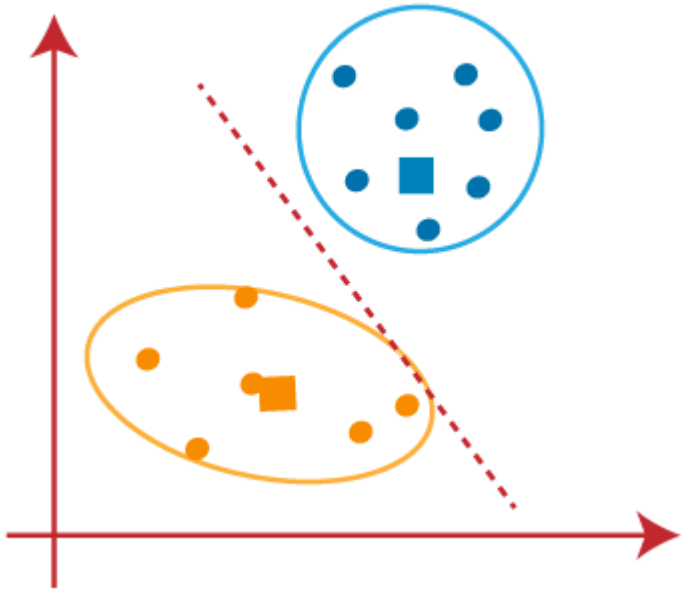
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



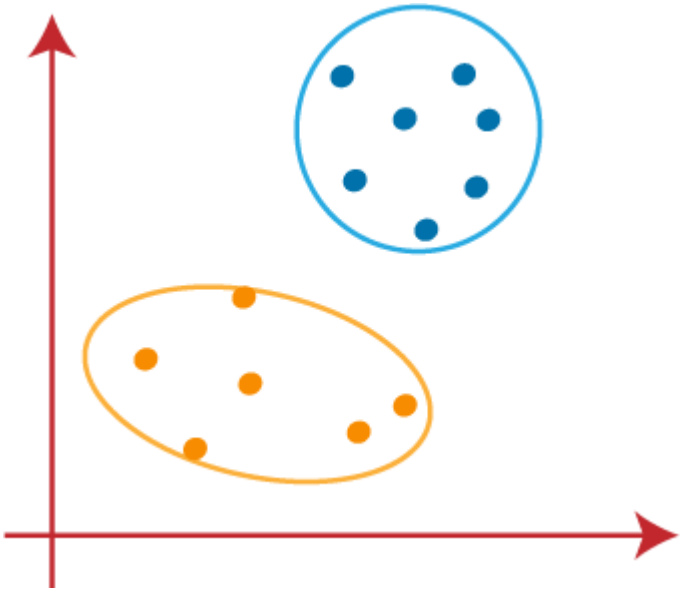
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



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



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



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

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different

ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

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

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

In the above formula of WCSS,

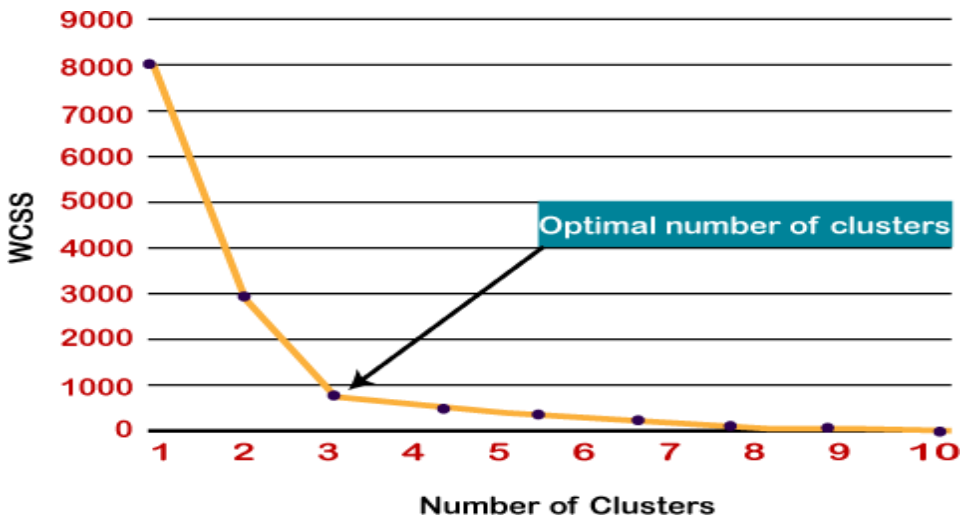
$\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

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

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

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

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



Limits of KMeans:

Similar to other algorithm, K-Mean clustering has many weaknesses:

- When the numbers of data are not so many, initial grouping will determine the cluster significantly.
- The number of cluster, K, must be determined before hand.
- We never know the real cluster, using the same data, if it is inputted in a different order may produce different cluster if the number of data is a few.
- Sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the local optimum.
- We never know which attribute contributes more to the grouping process since we assume that each attribute has the same weight.
- Weakness of arithmetic mean is not robust to outliers. Very far data from the centroid may pull the centroid away from the real one.
- The result is circular cluster shape because based on distance.

Using Clustering for Image Segmentation:

Image segmentation is the task of partitioning an image into multiple segments. In semantic segmentation, all pixels that are part of the same object type get assigned to the same segment. For example, in a self-driving car's vision system, all pixels that are part of a pedestrian's image might be assigned to the "pedestrian" segment (there would just be one segment containing all the pedestrians). In instance segmentation, all pixels that are part of the same individual object are assigned to the same segment. In this case there would be a different segment for each pedestrian. The state of the art in semantic or instance segmentation today is achieved using complex architectures based on convolutional neural networks. Here, we are going to do something much simpler: color segmentation. We will simply assign pixels to the same segment if they have a similar color. In some applications, this may be sufficient, for example if you want to analyze satellite images to measure how much total forest area there is in a region, color segmentation may be just fine.

First, let's load the image using Matplotlib's imread() function:

```
>>> from matplotlib.image import imread # you could also use `imageio.imread()`

>>> image = imread(os.path.join("images","clustering","ladybug.png"))

>>> image.shape (533, 800, 3)
```

The image is represented as a 3D array: the first dimension's size is the height, the second is the width, and the third is the number of color channels, in this case red, green and blue (RGB). In other words, for each pixel there is a 3D vector containing the intensities of red, green and blue, each between 0.0 and 1.0 (or between 0 and 255 if you use imageio.imread()). Some images may have less channels, such as gray- scale images (one channel), or more channels, such as images with an additional alpha channel for transparency, or satellite images which often contain channels for many light frequencies (e.g., infrared). The following code reshapes the array to get a long list of RGB colors, then it clusters these colors using K-Means. For example, it may identify a color cluster for all shades of green.

Next, for each color (e.g., dark green), it looks for the mean color of the pixel's color cluster. For example, all shades of green may be replaced with the same light green color (assuming the mean color of the green cluster is light green). Finally it reshapes this long list of colors to get the same shape as the original image.

```
X = image.reshape(-1, 3)

kmeans = KMeans(n_clusters=8).fit(X)

segmented_img = kmeans.cluster_centers_[kmeans.labels_]

segmented_img = segmented_img.reshape(image.shape)
```

Using Clustering for Preprocessing:

Clustering can be an efficient approach to dimensionality reduction, in particular as a preprocessing step before a supervised learning algorithm. For example, let's tackle the digits dataset which is a simple MNIST-like dataset containing 1,797 grayscale 8×8 images representing digits 0 to 9. First, let's load the dataset:

```
from sklearn.datasets

import load_digits X_digits, y_digits = load_digits(return_X_y=True)
```

Now, let's split it into a training set and a test set:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_digits, y_digits)
```

Next, let's fit a Logistic Regression model:

```
 from sklearn.linear_model import

LogisticRegression log_reg = LogisticRegression(random_state=42)

log_reg.fit(X_train, y_train)
```

Let's evaluate its accuracy on the test set:

```
>>> log_reg.score(X_test, y_test)

0.9666666666666667
```

Okay, that's our baseline: 96.7% accuracy. Let's see if we can do better by using KMeans as a preprocessing step. We will create a pipeline that will first cluster the training set into 50 clusters and replace the images with their distances to these 50 clusters, then apply a logistic regression model.

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([ ("kmeans", KMeans(n_clusters=50)), ("log_reg", LogisticRegression()), ]) pipeline.fit(X_train, y_train)
```

Now let's evaluate this classification pipeline:

```
>>> pipeline.score(X_test, y_test)
```

0.9822222222222222

We almost divided the error rate by a factor of 2! But we chose the number of clusters k completely arbitrarily, we can surely do better. Since K-Means is just a preprocessing step in a classification pipeline, finding a good value for k is much simpler than earlier: there's no need to perform silhouette analysis or minimize the inertia, the best value of k is simply the one that results in the best classification performance during cross-validation. Let's use GridSearchCV to find the optimal number of clusters:

```
from sklearn.model_selection import GridSearchCV

param_grid = dict(kmeans__n_clusters=range(2, 100))

grid_clf = GridSearchCV(pipeline, param_grid, cv=3, verbose=2)

grid_clf.fit(X_train, y_train)
```

Let's look at best value for k, and the performance of the resulting pipeline:

```
>>> grid_clf.best_params_

{'kmeans__n_clusters': 90}

>>> grid_clf.score(X_test, y_test)

0.9844444444444445
```

With k=90 clusters, we get a small accuracy boost, reaching 98.4% accuracy on the test set.

Using Clustering for Semi-Supervised Learning:

Another use case for clustering is in semi-supervised learning, when we have plenty of unlabeled instances and very few labeled instances. Let's train a logistic regression model on a sample of 50 labeled instances from the digits dataset:

```
n_labeled = 50

log_reg = LogisticRegression()

log_reg.fit(X_train[:n_labeled], y_train[:n_labeled])
```

The performance of this model on the test set

```
>>> log_reg.score(X_test, y_test)

0.8266666666666667
```

The accuracy is just 82.7%: it should come as no surprise that this is much lower than earlier, when we trained the model on the full training set. Let's see how we can do better. First, let's cluster the training set into 50 clusters, then for each cluster let's find the image closest to the centroid. We will call these images the representative images:

```
k = 50

kmeans = KMeans(n_clusters=k)

X_digits_dist = kmeans.fit_transform(X_train)
```

```
representative_digit_idx = np.argmin(X_digits_dist, axis=0)

X_representative_digits = X_train[representative_digit_idx]
```

Now let's look at each image and manually label it:

```
y_representative_digits = np.array([4, 8, 0, 6, 8, 3, ..., 7, 6, 2, 3, 1, 1])
```

Now we have a dataset with just 50 labeled instances, but instead of being completely random instances, each of them is a representative image of its cluster. Let's see if the performance is any better:

```
>>> log_reg = LogisticRegression()

>>> log_reg.fit(X_representative_digits, y_representative_digits)

>>> log_reg.score(X_test, y_test)

0.9244444444444444
```

We jumped from 82.7% accuracy to 92.4%, although we are still only training the model on 50 instances. Since it is often costly and painful to label instances, especially when it has to be done manually by experts, it is a good idea to label representa- tive instances rather than just random instances. This is called label propagation:

```
y_train_propagated = np.empty(len(X_train), dtype=np.int32)

for i in range(k):

    y_train_propagated[kmeans.labels_==i] = y_representative_digits[i]
```

Now let's train the model again and look at its performance:

```
>>> log_reg = LogisticRegression()

>>> log_reg.fit(X_train, y_train_propagated)

>>> log_reg.score(X_test, y_test)

0.9288888888888889
```

We got a tiny little accuracy boost. Better than nothing, but not astounding. The problem is that we propagated each representative instance's label to all the instances in the same cluster, including the instances located close to the cluster boundaries, which are more likely to be mislabeled. Let's see what happens if we only propagate the labels to the 20% of the instances that are closest to the centroids:

```
percentile_closest = 20

X_cluster_dist = X_digits_dist[np.arange(len(X_train)), kmeans.labels_]

for i in range(k):

    in_cluster = (kmeans.labels_ == i)

    cluster_dist = X_cluster_dist[in_cluster]

    cutoff_distance = np.percentile(cluster_dist, percentile_closest)
```

```
    above_cutoff = (X_cluster_dist > cutoff_distance)
    X_cluster_dist[in_cluster & above_cutoff] = -1
partially_propagated = (X_cluster_dist != -1)
X_train_partially_propagated = X_train[partially_propagated]
y_train_partially_propagated = y_train_propagated[partially_propagated]
```

Now let's train the model again on this partially propagated dataset:

```
>>> log_reg = LogisticRegression()
>>> log_reg.fit(X_train_partially_propagated, y_train_partially_propagated)
>>> log_reg.score(X_test, y_test)
0.9422222222222222
```

With just 50 labeled instances (only 5 examples per class on average!), we got 94.2% performance, which is pretty close to the performance of logistic regression on the fully labeled digits dataset (which was 96.7%). This is because the propagated labels are actually pretty good, their accuracy is very close to 99%:

```
>>> np.mean(y_train_partially_propagated == y_train[partially_propagated])
0.9896907216494846
```

DBSCAN:

Density-Based Clustering refers to one of the most popular unsupervised learning methodologies used in model building and machine learning algorithms. The data points in the region separated by two clusters of low point density are considered as noise. The surroundings with a radius ε of a given object are known as the ε neighborhood of the object. If the ε neighborhood of the object comprises at least a minimum number, MinPts of objects, then it is called a core object.

Density-Based Clustering - Background

There are two different parameters to calculate the density-based clustering

E_{PS}: It is considered as the maximum radius of the neighborhood.

MinPts: MinPts refers to the minimum number of points in an Eps neighborhood of that point.

NEps (i) : { k belongs to D and dist (i,k) < = Eps}

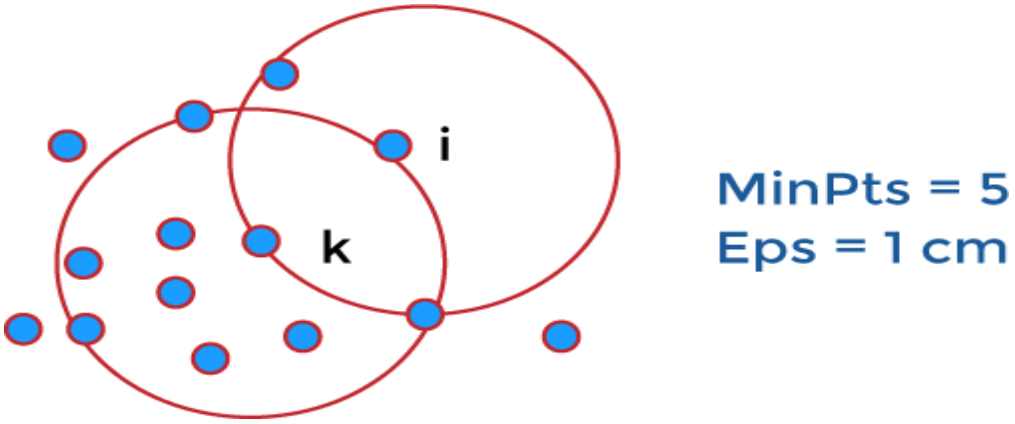
Directly density reachable:

A point i is considered as the directly density reachable from a point k with respect to Eps, MinPts if

i belongs to NEps(k)

Core point condition:

NEps (k) >= MinPts



Working of Density-Based Clustering

Suppose a set of objects is denoted by D', we can say that an object I is directly density reachable form the object j only if it is located within the ε neighborhood of j, and j is a core object.

An object i is density reachable form the object j with respect to ε and MinPts in a given set of objects, D' only if there is a sequence of object chains point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly density reachable from i_i with respect to ε and MinPts.

An object i is density connected object j with respect to ε and MinPts in a given set of objects, D' only if there is an object o belongs to D such that both point i and j are density reachable from o with respect to ε and MinPts.

Major Features of Density-Based Clustering

The primary features of Density-based clustering are given below.

- It is a scan method.
- It requires density parameters as a termination condition.
- It is used to manage noise in data clusters.
- Density-based clustering is used to identify clusters of arbitrary size.

DBSCAN algorithm can be abstracted in the following steps:

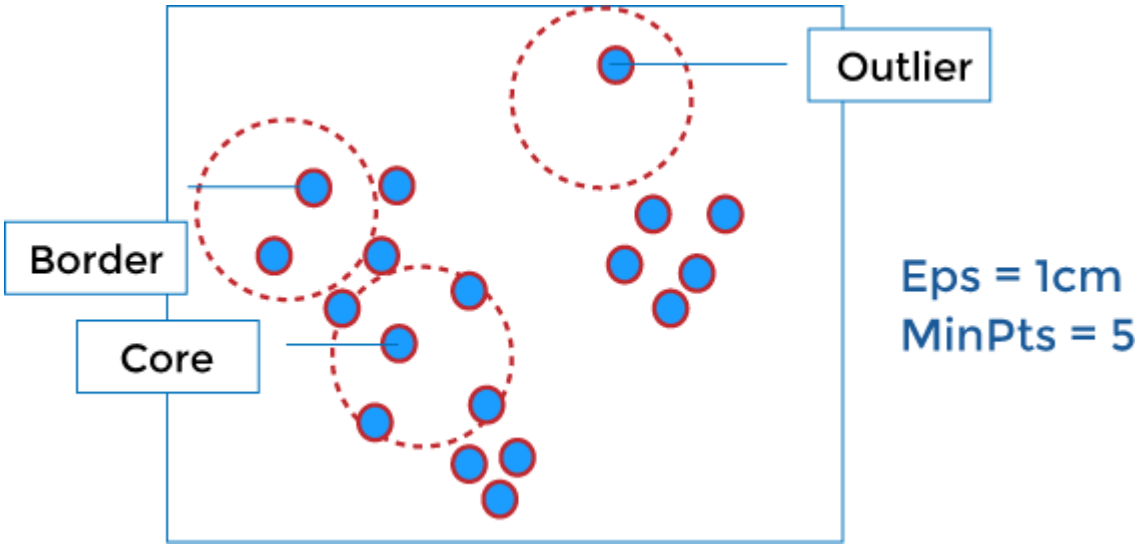
1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.

2. For each core point if it is not already assigned to a cluster, create a new cluster.

3. Find recursively all its density connected points and assign them to the same cluster as the core point.

A point a and b are said to be density connected if there exist a point c which has a sufficient number of points in its neighbors and both the points a and b are within the eps distance. This is a chaining process. So, if b is neighbor of c, c is nighbor of d, d is neighbor of e, which in turn is neighbor of a implies that b is neighbor of a.

4. Iterate through the remaining unvisited point in the dataset. Those points that do not belong to any cluster are noise.

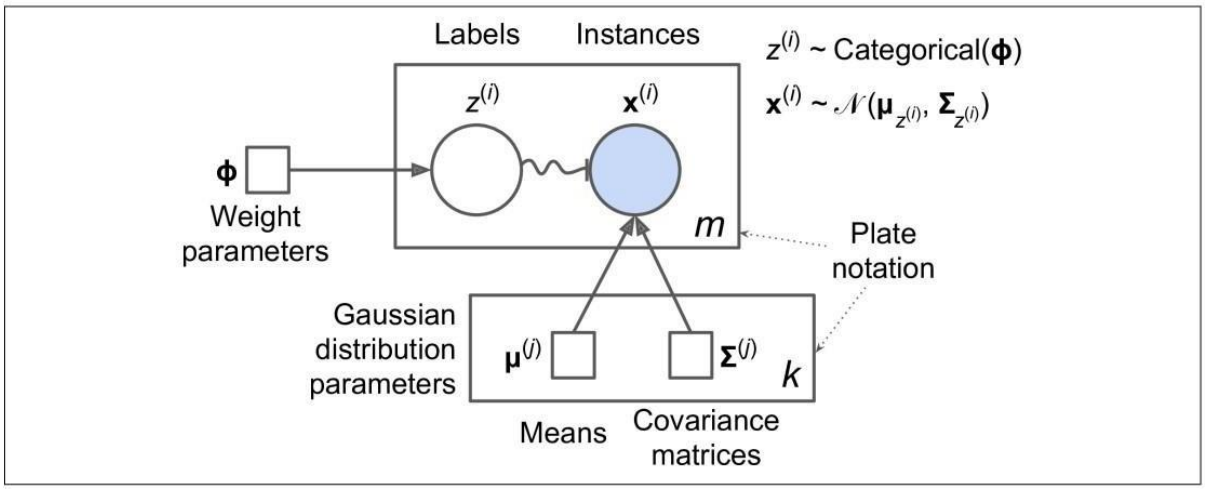


Gaussian Mixtures Model:

Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together.

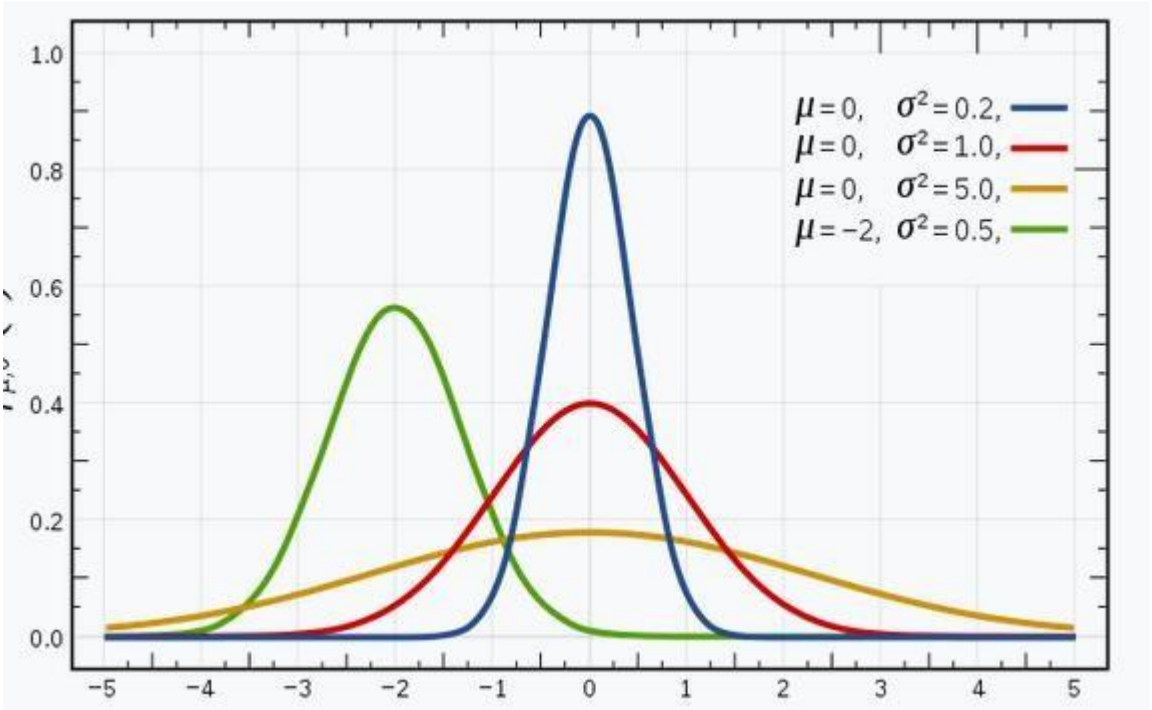
Let's say we have three Gaussian distributions (more on that in the next section) – GD1, GD2, and GD3. These have a certain mean (μ1, μ2, μ3) and variance (σ1, σ2, σ3) value respectively. For a given set of data points, our GMM would identify the probability of each data point belonging to each of these distributions.

Gaussian Mixture Models are probabilistic models and use the soft clustering approach for distributing the points in different clusters.



The Gaussian distribution:

The below image has a few Gaussian distributions with a difference in mean (μ) and variance (σ^2). Remember that the higher the σ value more would be the spread:



In a one dimensional space, the probability density function of a Gaussian distribution is given by:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ^2 is the variance.

But this would only be true for a single variable. In the case of two variables, instead of a 2D bell-shaped curve, we will have a 3D bell curve.

The probability density function would be given by:

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

where x is the input vector, μ is the 2D mean vector, and Σ is the 2×2 covariance matrix. The covariance would now define the shape of this curve. We can generalize the same for d-dimensions.

Thus, this multivariate Gaussian model would have x and μ as vectors of length d, and Σ would be a *d x d* covariance matrix.

Hence, for a dataset with *d* features, we would have a mixture of *k* Gaussian distributions (where *k* is equivalent to the number of clusters), each having a certain mean vector and variance matrix.

These values are determined using a technique called Expectation-Maximization (EM). We need to understand this technique before we dive deeper into the working of Gaussian Mixture Models.

Expectation-Maximization

Expectation-Maximization (EM) is a statistical algorithm for finding the right model parameters. We typically use EM when the data has missing values, or in other words, when the data is incomplete.

These missing variables are called **latent variables**. We consider the target (or cluster number) to be unknown when we're working on an unsupervised learning problem.

It’s difficult to determine the right model parameters due to these missing variables. Think of it this way – if you knew which data point belongs to which cluster, you would easily be able to determine the mean vector and covariance matrix.

Since we do not have the values for the latent variables, Expectation-Maximization tries to use the existing data to determine the optimum values for these variables and then finds the model parameters. Based on these model parameters, we go back and update the values for the latent variable, and so on.

Broadly, the Expectation-Maximization algorithm has two steps:

- **E-step:** In this step, the available data is used to estimate (guess) the values of the missing variables
- **M-step:** Based on the estimated values generated in the E-step, the complete data is used to update the parameters

Expectation-Maximization is the base of many algorithms, including Gaussian Mixture Models.

Expectation-Maximization in Gaussian Mixture Models

Let’s say we need to assign *k* number of clusters. This means that there are k Gaussian distributions, with the mean and covariance values to be μ_1, $\mu_{2, .. \mu k}$ and Σ_1, Σ_2, .. Σk . Additionally, there is another parameter for the distribution that defines the number of points for the distribution. Or in other words, the density of the distribution is represented with Π_i.

Now, we need to find the values for these parameters to define the Gaussian distributions. We already decided the number of clusters, and randomly assigned the values for the mean, covariance, and density. Next, we’ll perform the E-step and the M-step

E-step:

For each point x_i, calculate the probability that it belongs to cluster/distribution c_1, c_2, … c_k. This is done using the below formula:

$$r_{ic} = \frac{\text{Probability Xi belongs to c}}{\text{Sum of probability Xi belongs to } c_1, c_2, .. c_k} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

This value will be high when the point is assigned to the right cluster and lower otherwise.

M-step:

Post the E-step, we go back and update the Π, μ and Σ values. These are updated in the following manner:

1. The new density is defined by the ratio of the number of points in the cluster and the total number of points:

$$\Pi = \frac{\text{Number of points assigned to cluster}}{\text{Total number of points}}$$

2. The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion with the probability values for the data point. Hence, a data point that has a higher probability of being a part of that distribution will contribute a larger portion:

$$\mu = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} x_i$$

$$\Sigma_c = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} (x_i - \mu_c)^T (x_i - \mu_c)$$

Based on the updated values generated from this step, we calculate the new probabilities for each data point and update the values iteratively. This process is repeated in order to maximize the log-likelihood function.

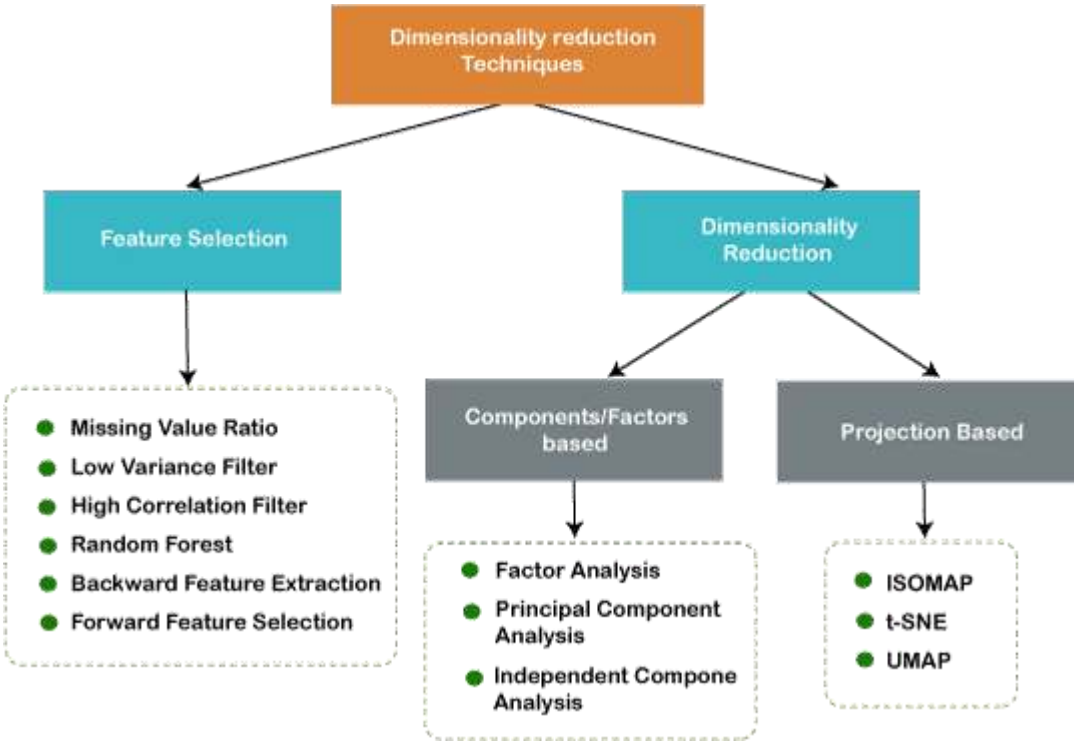
Dimensionality Reduction:

The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction. A dataset contains a huge number of input features in various cases, which makes the predictive

modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."*** These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.



The Curse of Dimensionality

Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Benefits of applying Dimensionality Reduction:

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- **I**t removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of dimensionality Reduction

There are also some disadvantages of applying the dimensionality reduction, which are given below:

- Some data may be lost due to dimensionality reduction.
- In the PCA dimensionality reduction technique, sometimes the principal components required to consider are unknown.

Main Approaches for Dimensionality Reduction:

There are two ways to apply the dimension reduction technique, which are given below:

Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation**
- **Chi-Square Test**
- **ANOVA**
- **Information Gain, etc.**

2. Wrappers Methods

The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase

the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection
- Bi-directional Elimination

3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

a. Principal Component Analysis

b. Linear Discriminant Analysis

c. Kernel PCA

d. Quadratic Discriminant Analysis

Principal Component Analysis(PCA):

Principal Component Analysis is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***

Using Scikit-Learn:

Scikit-Learn's PCA class implements PCA using SVD decomposition just like we did before. The following code applies PCA to reduce the dimensionality of the dataset down to two

dimensions (note that it automatically takes care of centering the data): from sklearn.decomposition import PCA pca = PCA(n_components = 2) X2D = pca.fit_transform(X) After fitting the PCA transformer to the dataset, you can access the principal compo- nents using the components_ variable (note that it contains the PCs as horizontal vectors, so, for example, the first principal component is equal to pca.components_.T[:, 0]).

Randomized PCA:

If you set the svd_solver hyperparameter to "randomized", Scikit-Learn uses a sto- chastic algorithm called Randomized PCA that quickly finds an approximation of the first d principal components. Its computational complexity is $O(m \times d^2) + O(d^3)$, instead of $O(m \times n^2) + O(n^3)$ for the full SVD approach, so it is dramatically faster than full SVD when d is much smaller than n:

```
rnd_pca = PCA(n_components=154, svd_solver="randomized")

X_reduced = rnd_pca.fit_transform(X_train)
```

By default, svd_solver is actually set to "auto": Scikit-Learn automatically uses the randomized PCA algorithm if m or n is greater than 500 and d is less than 80% of m or n, or else it uses the full SVD approach. If you want to force Scikit-Learn to use full SVD, you can set the svd_solver hyperparameter to "full".

Incremental PCA:

One problem with the preceding implementations of PCA is that they require the whole training set to fit in memory in order for the algorithm to run. Fortunately, Incremental PCA (IPCA) algorithms have been developed: you can split the training set into mini-batches and feed an IPCA algorithm one mini-batch at a time. This is useful for large training sets, and also to apply PCA online (i.e., on the fly, as new instances arrive). The following code splits the MNIST dataset into 100 mini-batches (using NumPy's array_split() function) and feeds them to Scikit-Learn's IncrementalPCA class5 to reduce the dimensionality of the MNIST dataset down to 154 dimensions (just like before). Note that you must call the partial_fit() method with each mini-batch rather than the fit() method with the whole training set:

```
from sklearn.decomposition import IncrementalPCA

n_batches = 100

inc_pca = IncrementalPCA(n_components=154)

for X_batch in np.array_split(X_train, n_batches):

    inc_pca.partial_fit(X_batch)

X_reduced = inc_pca.transform(X_train)
```

Alternatively, you can use NumPy's memmap class, which allows you to manipulate a large array stored in a binary file on disk as if it were entirely in memory; the class loads only the

data it needs in memory, when it needs it. Since the IncrementalPCA class uses only a small part of the array at any given time, the memory usage remains under control. This makes it possible to call the usual fit() method, as you can see in the following code:

X_mm = np.memmap(filename, dtype="float32", mode="readonly", shape=(m, n))

batch_size = m // n_batches inc_pca = IncrementalPCA(n_components=154, batch_size=batch_size)

inc_pca.fit(X_mm)

Kernel PCA:

Kernel PCA can be implemented using kernalPCA() function in sklearn. The basic idea behind kernal PCA is that we use kernal function to project the non-linear data into higher dimensional space where the groups are linearly separable. And then use regular PCA to do the dimentionality reduction.

Here use KernelPCA() function with "rbf" kernel function to perform kernel PCA.

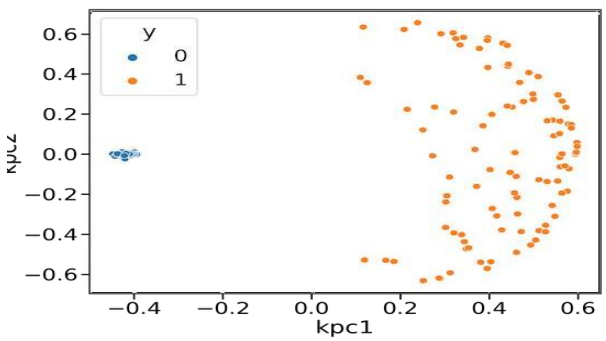
```
kpca = KernelPCA(kernel="rbf",
                 fit_inverse_transform=True,
                 gamma=10,
                n_components=2)
X_kpca = kpca.fit_transform(X)
```

Let us save the results into a dataframe as before.

```
kpca_res = pd.DataFrame(X_kpca)
kpca_res.columns=["kpc1","kpc2"]
kpca_res['y']=y
kpca_res.head()
```

Now, we can visualize the PCs from kernel PCA using scatter plot and we can clearly see that the data is linearly separable.

sns.scatterplot(data=kpca_res,x='kpc1',y='kpc2', hue="y")



PCA plot of non linear data with kernel PCA

UNIT-5

Neural Networks: Introduction to Artificial Neural Networks with Keras, Implementing MLPs with Keras, Installing TensorFlow 2, Loading and Preprocessing Data with TensorFlow.

Neural Networks:

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

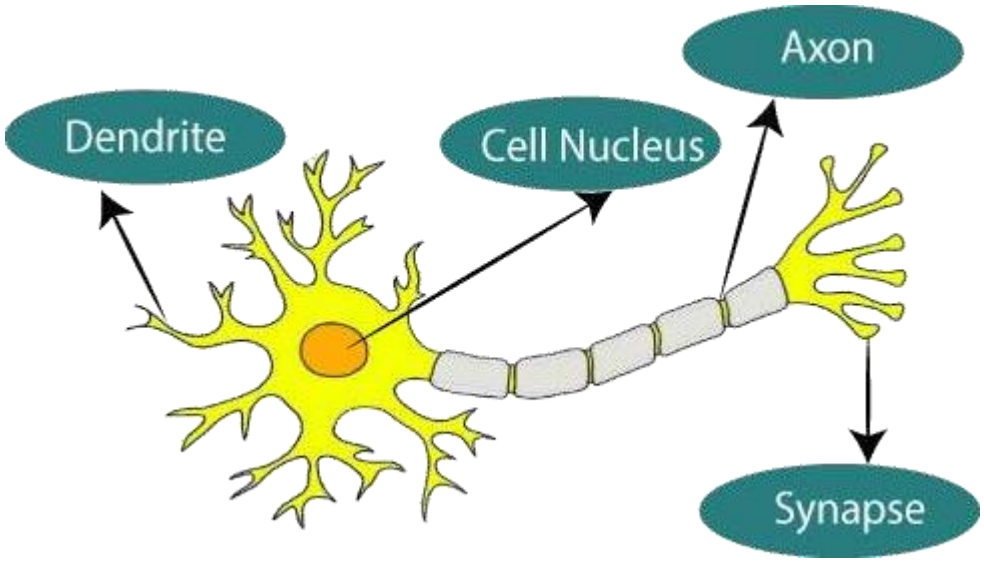
Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity.

Introduction to Artificial Neural Networks:

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

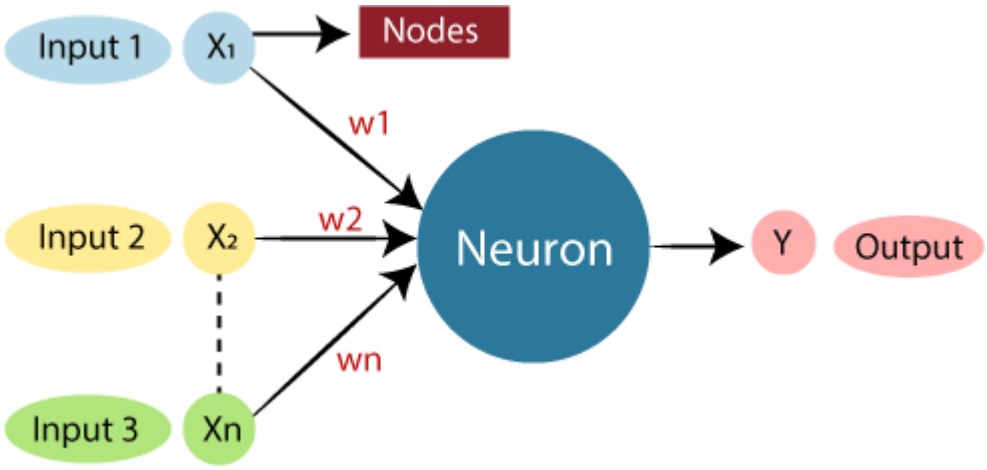
What is Artificial Neural Network?

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.



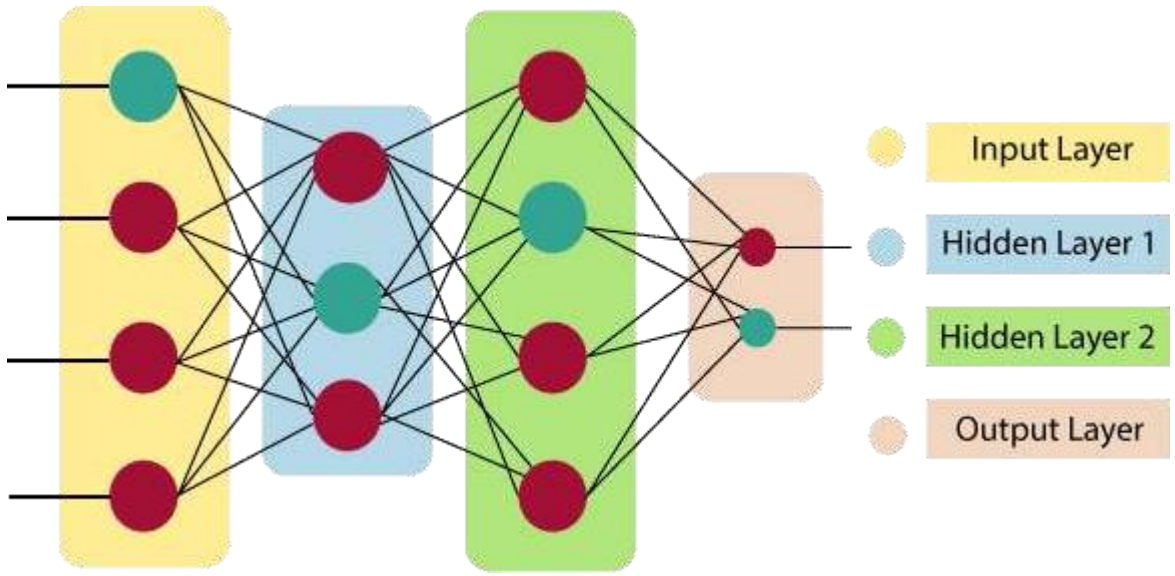
Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

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.

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.

Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

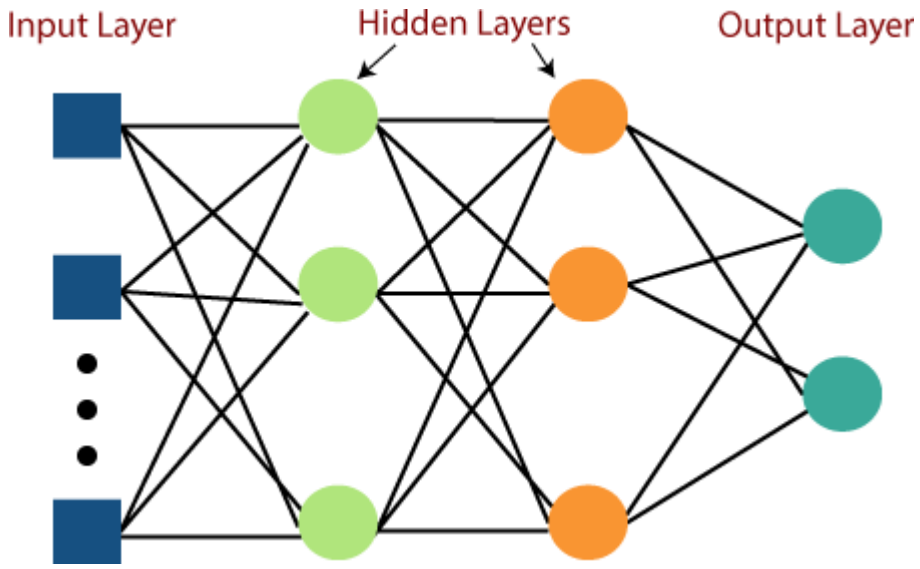
Feed-Forward ANN:

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

Multi-layer Perceptron:

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron. TensorFlow is a very popular deep learning framework released by, and this notebook will guide to build a neural network with this library. If we want to understand what is a Multi-layer perceptron, we have to develop a multi-layer perceptron from scratch using Numpy.

The pictorial representation of multi-layer perceptron learning is as shown below-



MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called **back propagation's algorithm**.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method.

The algorithm for the MLP is as follows:

1. Just as with the perceptron, the inputs are pushed forward through the MLP by taking the dot product of the input with the weights that exist between the input layer and the hidden layer (WH). This dot product yields a value at the hidden layer. We do not push this value forward as we would with a perceptron though.

2. MLPs utilize activation functions at each of their calculated layers. There are many activation functions to discuss: rectified linear units (ReLU), sigmoid function, tanh. Push the calculated output at the current layer through any of these activation functions.

3. Once the calculated output at the hidden layer has been pushed through the activation function, push it to the next layer in the MLP by taking the dot product with the corresponding weights.

4. Repeat steps two and three until the output layer is reached.

5. At the output layer, the calculations will either be used for a backpropagation algorithm that corresponds to the activation function that was selected for the MLP (in the case of training) or a decision will be made based on the output (in the case of testing).

MLPs form the basis for all neural networks and have greatly improved the power of computers when applied to classification and regression problems. Computers are no longer limited by XOR cases and can learn rich and complex models thanks to the multilayer perceptron.

Installing Tensorflow 2:

You can simply use pip to install TensorFlow. If you created an isolated environment using virtualenv, you first need to activate it:

$ cd $ML_PATH # Your ML working directory (e.g., $HOME/ml)

$ source env/bin/activate # on Linux or MacOSX

$.\env\Scripts\activate # on Windows

Next, install TensorFlow 2 (if you are not using a virtualenv, you will need adminis- trator rights, or to add the --user option):

$ python3 -m pip install --upgrade tensorflow

To test your installation, open a Python shell or a Jupyter notebook, then import Ten- sorFlow and tf.keras, and print their versions:

>>> import tensorflow as tf

>>> from tensorflow import keras

>>> tf.__version__

'2.0.0'

>>> keras.__version__

'2.2.4-tf'

The second version is the version of the Keras API implemented by tf.keras. Note that it ends with -tf, highlighting the fact that tf.keras implements the Keras API, plus some extra TensorFlow-specific features.