1. INTRODUCTION

ARTIFICIAL INTELLIGENCE

Artificial Intelligence is one of the booming technologies of computer science, which is ready to create a new revolution in the world by making intelligent machines. AI holds a tendency to cause a machine to work as a human. The word "Artificial Intelligence" first adopted by American Computer scientist **John McCarthy** at the Dartmouth Conference in the year **1956**.

What is Artificial 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."

So, we can define AI as, "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

Why Artificial Intelligence?

- With the help of AI, we can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, we can create your personal virtual Assistant, such as Cortana, Google Assistant etc.
- With the help of AI, we can build such Robots which can work in an environment where survival of humans can be at risk.
- AI opens a path for other new technologies, new devices, and new Opportunities.

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.

Definition: "AI is the study of how to make computers do things at which, at the moment, people are better".

Advantages of Artificial Intelligence:

1. High Accuracy with fewer errors: AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
2. 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.
3. High reliability: AI machines are highly reliable and can perform the same action multiple times with high accuracy.
4. 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.
5. 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.
6. 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:

1. High Cost: The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
2. 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.
3. 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.
4. Increase dependency on machines: With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
5. 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.

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

Applicationsof AI:

1. Gaming: AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
2. Natural Language Processing: It is possible to interact with the computer that understands natural language spoken by humans.
3. Expert Systems: There are some applications which integrate machine, software and special information to impart reasoning and advising. They provide explanation and advice to the users.
4. Vision Systems: These systems understand, interpret and comprehend visual input on the computer. For example,
   a. A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.
   b. Doctors use clinical expert system to diagnose the patient.
   c. Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
5. Speech Recognition: Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
6. Handwriting Recognition: The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and converts it into editable text.
7. Intelligent Robots: Robots are able to perform the tasks given by humans. They have special sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

MACHINE LEARNING

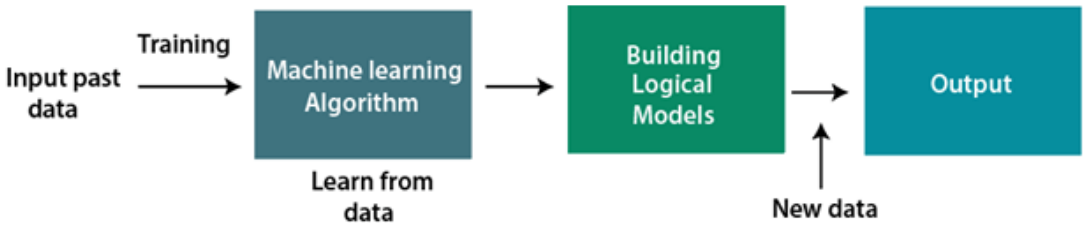
Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using

historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

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

"Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed."

- A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.
- The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.
- Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output.
- Machine learning has changed our way of thinking about the problem.



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

A checkers learning problem:

- Task T: playing checkers
- Performance measure P: percent of games won against opponents
- Training experience E: playing practice games against itself

We can specify many learning problems in this fashion, such as learning to recognize handwrittenwords (or) learning to drive a robotic automobile autonomously.

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

Importance of Machine Learning:

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

TYPES (OR) CLASSIFICATION OF MACHINE LEARNING:

- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

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. The example of supervised learning is spam filtering.Supervised learning can be grouped further in two categories of algorithms:

- 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
    - Linear Regression
    - Regression Trees

    - Non-Linear Regression
    - Bayesian Linear Regression
    - Polynomial Regression
- 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.
    - 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

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: 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.

- K-Means Clustering algorithm
- Mean-shift algorithm
- DBSCAN Algorithm
- Principal Component Analysis
- Independent Component Analysis

- 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.
  - Apriori algorithm
  - FP-growth algorithm

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.

Semi-Supervised Learning: To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced.The main aim of semi-supervised learning is to effectively use all the available data, rather than only labeled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labeled data.

Advantages of Semi-Supervised Learning:

- It is simple and easy to understand the algorithm

- It is highly efficient
- It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms

Disadvantages of Semi-Supervised Learning:

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

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

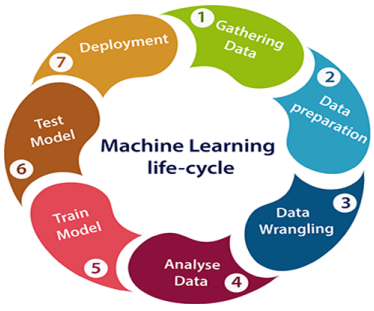
Advantages of Reinforcement Learning:

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

Disadvantages of Reinforcement Learning:

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

Machine learning Life cycle: Machine learning life cycle involves seven major steps



1. Gathering Data:
    - The goal of this step is to identify and obtain all data-related problems
    - In this step, we need to identify the different data sources, as data can be collected from various sources such as files, database, internet, or mobile devices
    - The more will be the data, the more accurate will be the prediction
2. Data Preparation:
    - Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training
    - First, we put all data together, and then randomize the ordering of data
    - This step can be further divided into two processes:
        - Data exploration:It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.A better understanding of data leads to an effective outcome
        - Data pre-processing:Now the next step is preprocessing of data for its analysis
3. Data Wrangling:
    - Data wrangling is the process of cleaning and converting raw data into a useable format
    - Cleaning of data is required to address the quality issues
    - In real-world applications, collected data may have various issues like Missing Values, Duplicate data, Invalid data, Noise
4. Analyse Data:
    - This step involves: Selection of analytical techniques, Building models, Review the result
    - The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome
5. Train the model:
    - In this step we train our model to improve its performance for better outcome of the problem
    - We use datasets to train the model using various machine learning algorithms
    - Training a model is required so that it can understand the various patterns, rules, and, features
6. Test the model:
    - In this step, we check for the accuracy of our model by providing a test dataset to it
    - Testing the model determines the percentage accuracy of the model as per the requirement of project or problem
7. Deployment:

- Here, we deploy the model in the real-world system
- If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system.
- But before deploying the project, we will check whether it is improving its performance using available data or not

Data Preprocessing in Machine Learning:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. It involves below steps:

1. Getting the dataset: The collected data for a particular problem in a proper format is known as the dataset. Dataset may be of different formats for different purposes. To use the dataset in our code, we usually put it into a CSV (Comma-Separated Values) file. Sometimes, we may also need to use an HTML or xlsx file
2. Importing libraries: In order to perform data preprocessing using Python, we need to import some predefined Python libraries like numpy (used for including any type of mathematical operation, supports to add multidimensional arrays and matrices), pandas (used for importing and managing the datasets), matplotlib(used to plot any type of charts) etc
3. Importing datasets: Before importing a dataset, we need to set the current directory as a working directory
4. Finding Missing Data: If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.There are mainly two ways to handle missing data. They are:
    - By deleting the particular row
    - By calculating the mean

    To handle missing values, we will use Scikit-learn library in our code, which contains various libraries for building machine learning models.
5. Encoding Categorical Data: Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.
6. Splitting dataset into training and test set:

- Training Set: A subset of dataset to train the machine learning model, and we already know the output.
- Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

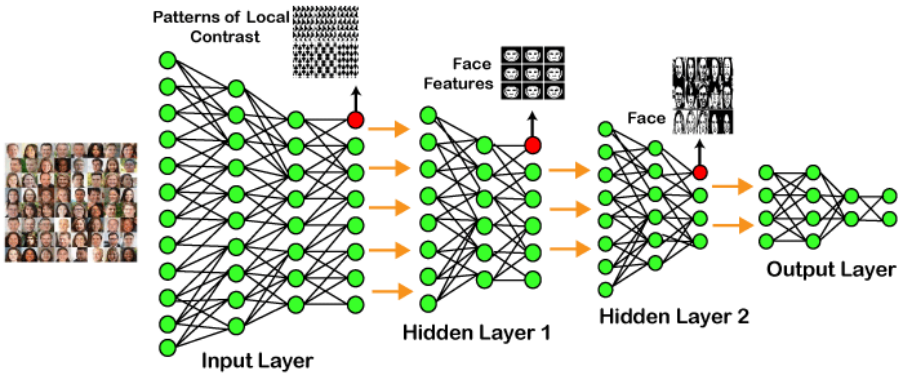
7. Feature scaling: It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

DEEP LEARNING

Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers.

Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.

Definition: "Deep learning is a collection of statistical techniques of machine learning for learning feature hierarchies that are actually based on artificial neural networks".



- Here, we provide the raw data of images to the first layer of the input layer.
- After then, input layer will determine the patterns of local contrast that means it will differentiate on the basis of colors, luminosity, etc.

- Then the 1st hidden layer will determine the face feature, i.e., it will fixate on eyes, nose, and lips, etc. And then, it will fixate those face features on the correct face template.
- So, in the 2nd hidden layer, it will actually determine the correct face here as it can be seen in the above image, after which it will be sent to the output layer.
- Likewise, more hidden layers can be added to solve more complex problems, for example, if you want to find out a particular kind of face having large or light complexions.
- So, as and when the hidden layers increase, we are able to solve complex problems.

Types of Deep Learning Networks:

- Feed Forward Neural Network
- Recurrent Neural Network
- Convolutional Neural Network
- Restricted Boltzmann Machine
- Autoencoders

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

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

Convolutional Neural Network:

- Convolutional Neural Networks are a special kind of neural network mainly used for image classification, clustering of images and object recognition
- To achieve the best accuracy, deep convolutional neural networks are preferred more than any other neural network

- The applications of CNN are Identify Faces, Street Signs, Tumors, Image Recognition, Video Analysis, NLP, Anomaly Detection, Drug Discovery, Checkers Game, Time Series Forecasting

Restricted Boltzmann Machine:

- 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

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
- The autoencoders are mainly used for the smaller representation of the input
- It helps in the reconstruction of the original data from compressed data

Applications of Deep Learning:

1. 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, to reduce the accidents.
2. Voice control assistance: Siri is one of the examples of Voice control assistance. We can tell Siri whatever we want it to do, and it will search and display it.
3. Automatic Image Caption Generation: Whatever image is uploaded, the algorithm will work in such a way that it will generate caption accordingly. Supposeif we say blue colored eye, it will display a blue-colored eye with a caption at the bottom of the image.
4. Automatic Machine Translation: With the help of automatic machine translation, we are able to convert one language into another.

Advantages of Deep Learning:

- 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

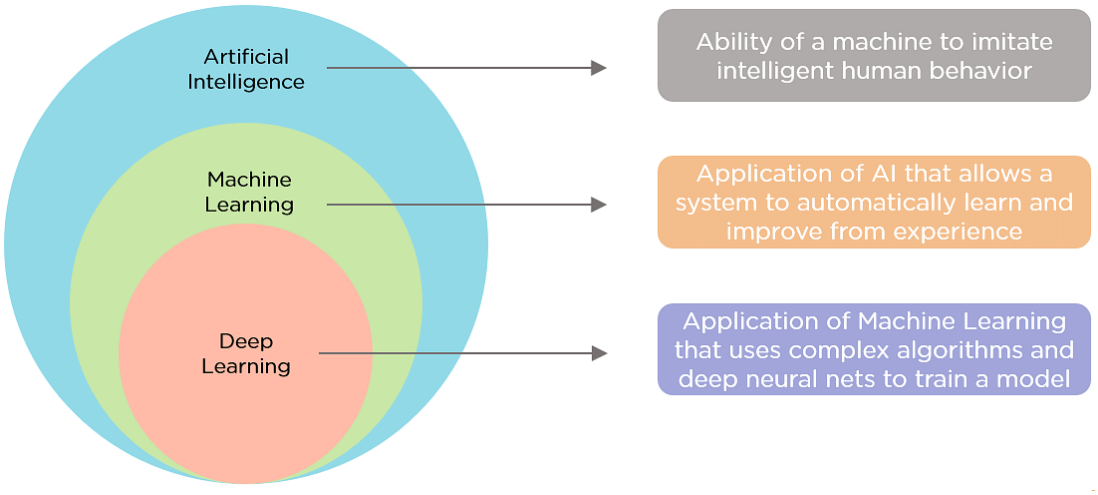
Disadvantages of Deep Learning:

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

Deep Learning Algorithms: The Deep Learning Algorithms are as follows:

- Convolutional Neural Networks (CNNs)
- Long Short Term Memory Networks (LSTMs)
- Recurrent Neural Networks (RNNs)
- Generative Adversarial Networks (GANs)
- Radial Basis Function Networks (RBFNs)
- Multilayer Perceptrons (MLPs)
- Self Organizing Maps (SOMs)
- Deep Belief Networks (DBNs)
- Restricted Boltzmann Machines (RBMs)

DIFFERENCES BETWEEN AI, ML & DL



Artificial Intelligence (AI)	Machine Learning (ML)	Deep Learning (DL)
It is the study which enables machines to mimic human behaviour through particular algorithm.	It is the study that uses statistical methods enabling machines to improve with experience.	It is the study that makes use of Neural Networks to imitate functionality just like a human brain.

AI is the broader family consisting of ML and DL as it's components.	ML is the subset of AI.	DL is the subset of ML.
AI is a computer algorithm which exhibits intelligence through decision making.	ML is an AI algorithm which allows system to learn from data.	DL is a ML algorithm that uses deep (more than one layer) neural networks to analyze data and provide output accordingly.
Search Trees and much complex math is involved in AI.	If you have a clear idea about the logic (math) involved in behind and you can visualize the complex functionalities like K-Mean, Support Vector Machines, etc., then it defines the ML aspect.	If you are clear about the math involved in it but don't have idea about the features, so you break the complex functionalities into linear/lower dimension features by adding more layers, then it defines the DL aspect.
The aim is to basically increase chances of success and not accuracy.	The aim is to increase accuracy not caring much about the success ratio.	It attains the highest rank in terms of accuracy when it is trained with large amount of data.
Three broad categories/types Of AI are: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI)	Three broad categories/types Of ML are: Supervised Learning, Unsupervised Learning and Reinforcement Learning	DL can be considered as neural networks with a large number of parameters layers lying in one of the four fundamental network architectures: Unsupervised Pre-trained Networks, Convolutional Neural Networks, Recurrent Neural Networks and Recursive Neural Networks
The efficiency Of AI is basically the efficiency provided by ML and DL respectively.	Less efficient than DL as it can't work for longer dimensions or higher amount of data.	More powerful than ML as it can easily work for larger sets of data.
Examples of AI applications include: Google's AI-Powered	Examples of ML applications include: Virtual Personal	Examples of DL applications include: Sentiment based news

Predictions, Ridesharing Apps Like Uber and Lyft, Commercial Flights Use an AI Autopilot, etc.	Assistants: Siri, Alexa, Google, etc., Email Spam and Malware Filtering.	aggregation, Image analysis and caption generation, etc.

MAIN CHALLENGES OF MACHINE LEARNING

1. Inadequate Training Data: The major issue that comes while using machine learning algorithms is the lack of quality as well as quantity of data.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. Poor quality of data: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.
3. Non-representative 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.
4. Overfitting and Underfitting: 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. The methods to reduce overfitting are:
    - Increase training data in a dataset
    - Reduce model complexity by simplifying the model by selecting one with fewer parameters
    - Ridge Regularization and Lasso Regularization
    - Early stopping during the training phase
    - Reduce the noise
    - Reduce the number of attributes in training data.
    - Constraining the model

    Underfitting is just the opposite of overfitting. Whenever a machine learning model is trained with fewer amounts of data, and as a result, it provides incomplete and inaccurate data and destroys the accuracy of the machine learning model. The methods to reduce Underfitting are:

- Increase model complexity
- Remove noise from the data
- Trained on increased and better features
- Reduce the constraints
- Increase the number of epochs to get better results

5. Monitoring and maintenance: Different results for different actions require data change; hence editing of codes as well as resources for monitoring them also become necessary.
6. Getting bad recommendations: A machine learning model operates under a specific context which results in bad recommendations and concept drift in the model. Let's understand with an example where at a specific time customer is looking for some gadgets, but now customer requirement changed over time but still machine learning model showing same recommendations to the customer while customer expectation has been changed. This incident is called a Data Drift.
7. Lack of skilled resources: Although Machine Learning is continuously growing in the market, the absence of skilled resources in the form of manpower is also an issue. Hence, we need manpower having in-depth knowledge of mathematics, science, and technologies for developing and managing scientific substances for machine learning.
8. Customer Segmentation:Customer segmentation is also an important challenge to identify the customers who paid for the recommendations shown by the model and who don't even check them. Hence, an algorithm is necessary to recognize the customer behavior and trigger a relevant recommendation for the user based on past experience.
9. Process Complexity of Machine Learning: The machine learning process is very complex. It also includes analyzing the data, removing data bias, training data, applying complex mathematical calculations, etc., making the procedure more complicated and quite tedious.
10. Data Bias: These errors exist when certain elements of the dataset are heavily weighted or need more importance than others. Biased data leads to inaccurate results, skewed outcomes, and other analytical errors. The methods to remove Data Bias are:
    - Research more for customer segmentation
    - Be aware of your general use cases and potential outliers
    - Combine inputs from multiple sources to ensure data diversity
    - Include bias testing in the development process
    - Analyze data regularly and keep tracking errors to resolve them easily
    - Review the collected and annotated data

11. Lack of Explain ability: This basically means the outputs cannot be easily comprehended as it is programmed in specific ways to deliver for certain conditions. Hence, a lack of explain ability is also found in machine learning algorithms which reduce the credibility of the algorithms.
12. Slow implementations and results: Machine learning models are highly efficient in producing accurate results but are time-consuming. Slow programming, excessive requirements' and overloaded data take more time to provide accurate results than expected.
13. 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.

1. STATISTICAL LEARNING

TRAINING AND TEST LOSS

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

Training Dataset:

- The training data is the biggest (in -size) subset of the original dataset, which is used to train or fit the machine learning model.
- Firstly, the training data is fed to the ML algorithms, which lets them learn how to make predictions for the given task.
- The training data varies depending on whether we are using Supervised Learning or Unsupervised Learning Algorithms.
- The type of training data that we provide to the model is highly responsible for the model's accuracy and prediction ability.
- Training data is approximately more than or equal to 60% of the total data for an ML project.

Test Dataset:

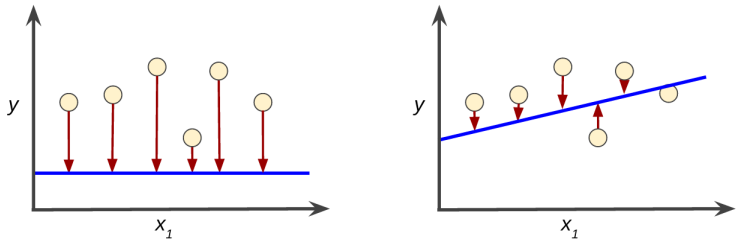
- Once we train the model with the training dataset, it's time to test the model with the test dataset.
- The test dataset is another subset of original data, which is independent of the training dataset.
- Usually, the test dataset is approximately 20-25% of the total original data for an ML project.

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

If we train our model with a training set and then test it with a completely different test dataset, and then our model will not be able to understand the correlations between the features.Therefore, if we train and test the model with two different datasets, then it will decrease the performance of the model. Hence it is important to split a dataset into two parts, i.e., train and test set.

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

Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.



Here, The arrows represent loss

The blue lines represent predictions

We can notice that the arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the line in the right plot is a much better predictive model than the line in the left plot.

Mean square error (MSE): MSE is a popular loss function. It is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediction(x))^2$$

Where,

(x , y) is an example in which x is the set of features that the model uses to make predictions& y is the example's label

prediction(x) is a function of the weights and bias in combination with the set of features x

D is a data set containing many labeled examples, which are (x , y) pairs

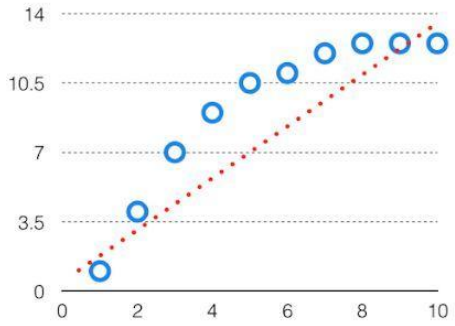
N is the number of examples in D

TRADEOFFS IN STATISTICAL LEARNING

Bias:

The bias is known as the difference between the prediction of the values by the ML model and the correct value. Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.

By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as Underfitting of Data. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation



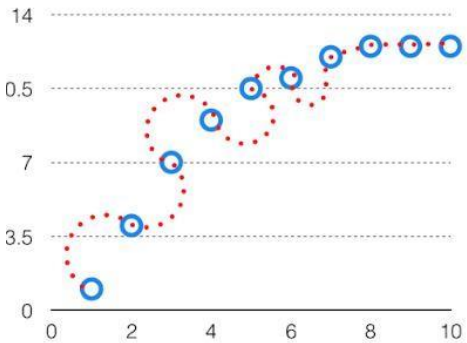
In such a problem, a hypothesis looks like follows

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Variance:

The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

When a model is high on variance, it is then said to as Overfitting of Data. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high.While training a data model variance should be kept low.The high variance data looks like follows



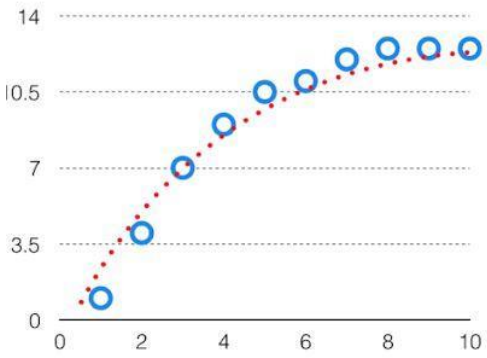
In such a problem, a hypothesis looks like follows

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

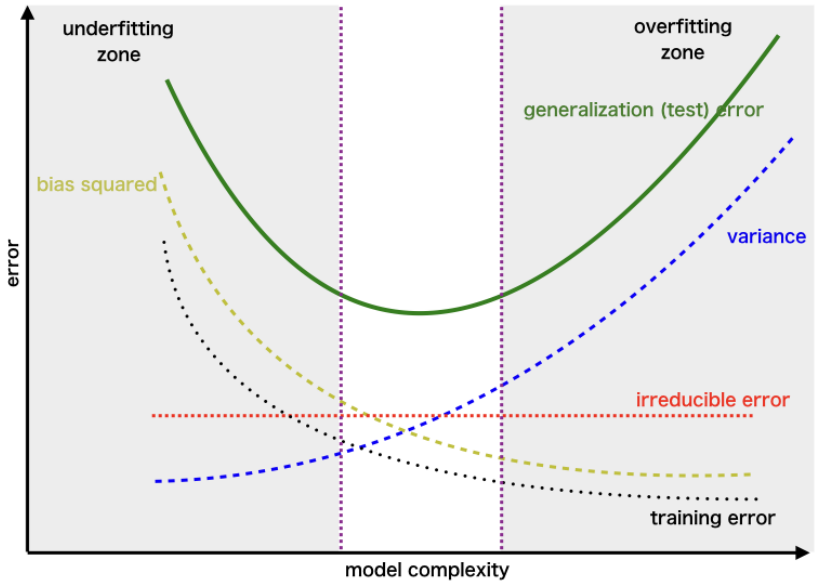
Bias Variance Tradeoff:

If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as Trade-off or Bias Variance Trade-off.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like.



The best fit will be given by hypothesis on the tradeoff point.The error to complexity graph to show trade-off is given as



This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

ESTIMATING RISK STATISTICS

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data.

In machine learning, each task or problem is divided into classification and Regression. Different evaluation metrics are used for both Regression and Classification tasks.

Performance Metrics for Classification:

- Confusion Matrix: The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known.

N=total Predictions	Actual : Positive	Actual : Negative
Predicted : Positive	True Positive	False Positive
Predicted : Negative	False Negative	True Negative

  - True Negative: Model has given prediction No, and the real or actual value was also No.
  - True Positive: The model has predicted Yes, and the actual value was also Yes.
  - False Negative: The model has predicted No, but the actual value was Yes, it is also called as Type-II error.
  - False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

Example of Confusion Matrix is as follows:

N=165	Actual : Positive	Actual : Negative
Predicted : Positive	100	10
Predicted : Negative	5	50

In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No. However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

- Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

- Precision: It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Recall (or) Sensitivity: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

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

- F-Score: If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision.

$$\text{F-measure} = \frac{2*Recall*Precision}{Recall+Precision}$$

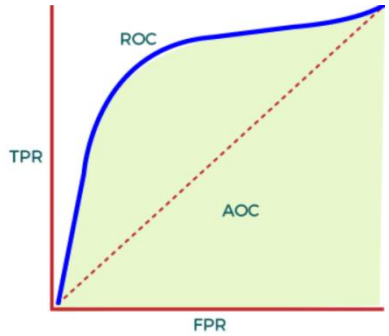
- AUC(Area Under the Curve)-ROC: Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. Firstly, ROC means Receiver Operating Characteristic curve. ROC represents a graph to show the performance of a classification model at different threshold levels. The curve is plotted between two parameters, which are:
    - True Positive Rate: TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

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

    - False Positive Rate: FPR or False Positive Rate can be calculated as:

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

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



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

Performance Metrics for Regression:

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

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

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

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

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

- R2 Score: R squared error is also known as Coefficient of Determination, which is another popular metric used for Regression model evaluation. The R-squared metric enables us to compare model with a constant baseline to determine the performance of the model. To select the constant baseline, we need to take the mean of the data and draw the line at the mean. The R squared score will always be less than or equal to 1 without concerning if the values are too large or small.

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

- Adjusted R2: Adjusted R squared, as the name suggests, is the improved version of R squared error. R square has a limitation of improvement of a score on increasing the terms, even though the model is not improving, and it may mislead the data scientists.

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

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

  Here, n is the number of observations, k denotes the number of independent variables and R_a^2 denotes the adjusted R^2

SAMPLING DISTRIBUTION OF AN 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:

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

Where σ_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. he sampling distribution of the standard deviation is the standard error of the standard deviation. It is defined as:

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

EMPIRICAL RISK MINIMIZATION

Empirical risk minimization (ERM) is a principle in statistical learning theory which defines a family of learning algorithms and is used to give theoretical bounds on their performance.

Empirical risk minimization (ERM) is a principle in statistical learning theory which defines a family of learning algorithms and is used to give theoretical bounds on their performance. The idea is that we don't know exactly how well an algorithm will work in practice because we don't know the true distribution of data that the algorithm will work on but as an alternative we can measure its performance on a known set of training data.

We assumed that our samples come from this distribution and use our dataset as an approximation. If we compute the loss using the data points in our dataset, it's called empirical risk. It is "empirical" and not "true" because we are using a dataset that's a subset of the whole population.

When our learning model is built, we have to pick a function that minimizes the empirical risk that is the delta between predicted output and actual output for data points in the dataset. This process of finding this function is called empirical risk minimization (ERM). We want to minimize the true risk. We don't have information that allows us to achieve that, so we hope that this empirical risk will almost be the same as the true empirical risk.

However, we can compute an approximation, called empirical risk, by averaging the loss function on the training set; more formally, computing the expectation with respect to the empirical measure:

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i)$$

For example if we to build a model that can differentiate between a male and a female based on specific features. If we select 150 random people where women are really short, and men are really tall, then the model might incorrectly assume that height is the differentiating feature. For building a truly accurate model, we have to gather all the women and men in the world to extract differentiating features. Unfortunately, that is not possible! So we select a small number of people and hope that this sample is representative of the whole population.

2. SUPERVISED LEARNING (REGRESSION/CLASSIFICATION)

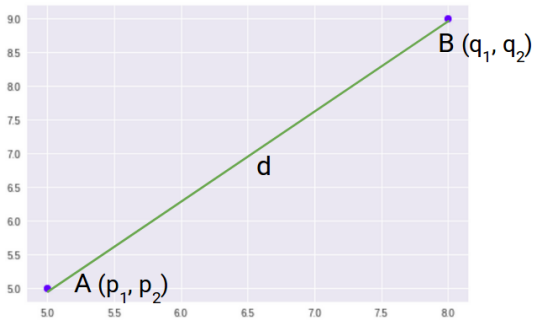
DISTANCE BASED METHODS

- Distance based model work on the concept of distance.
- Good distance based model helps in increasing the performance of classification and clustering process significantly.
- These models help algorithms to recognize the similarities between the contents.

Distance Function

- The distance function provides distance between the elements of a set. If the distance is zero then the elements are equivalent else they are different from each other.
- The distance function is nothing but a mathematical formula used by distance metrics. The distance function can differ across different function metrics.
- Distance metrics are a key part of several machine learning algorithms. These distance metrics are used in both supervised and unsupervised learning, generally to calculate the similarity between data points.
- There are 4 Types of Distance Metrics in Machine Learning
    1. Euclidean Distance
    2. Manhattan Distance
    3. Minkowski Distance
    4. Hamming Distance

Euclidean Distance: Euclidean Distance represents the shortest distance between two points. Most machine learning algorithms including K-Means use this distance metric to measure the similarity between observations.



Here's the formula for Euclidean Distance:

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$

We use the above formula when we are dealing with 2 dimensions. We can generalize this for an n-dimensional space as:

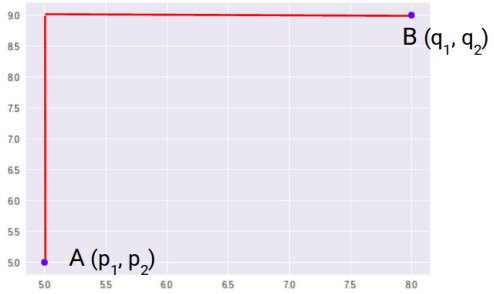
$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

Where,

- n = number of dimensions
- p_i, q_i = data points

Euclidean Distance between (1, 2, 3) and (4, 5, 6) is 5.196152422706632.

Manhattan Distance: Manhattan Distance is the sum of absolute differences between points across all the dimensions. We can represent Manhattan Distance as:



Since the above representation is 2 dimensional, to calculate Manhattan Distance, we will take the sum of absolute distances in both the x and y directions. So, the Manhattan distance in a 2-dimensional space is given as:

$$d = |p_1 - q_1| + |p_2 - q_2|$$

And the generalized formula for an n-dimensional space is given as:

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

Manhattan Distance between (1, 2, 3) and (4, 5, 6) is 9.

Minkowski Distance: Minkowski Distance is the generalized form of Euclidean and Manhattan Distance. The formula for Minkowski Distance is given as:

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Here, p represents the order of the norm.

Minkowski Distance between (1, 2, 3) and (4, 5, 6) is 4.3267487109222245.

Hamming Distance: Hamming Distance measures the similarity between two strings of the same length. The Hamming Distance between two strings of the same length is the number of positions at which the corresponding characters are different. Let's understand the concept using an example. Let's say we have two strings: "euclidean" and "manhattan"

Since the length of these strings is equal, we can calculate the Hamming Distance. We will go character by character and match the strings. The first character of both the strings (e and m respectively) is different. Similarly, the second character of both the strings (u and a) is different. and so on. Therefore seven characters are different whereas two characters (the last two characters) are similar:

euclidean and manhattan

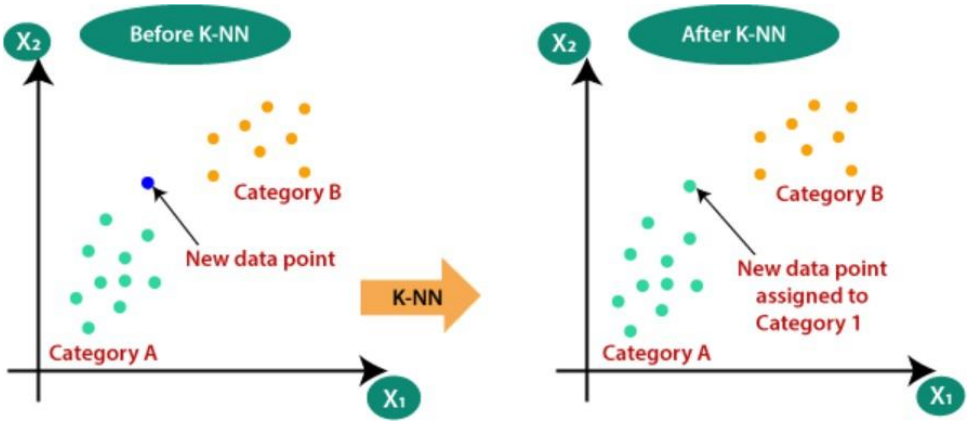
Hence, the Hamming Distance here will be 7. Note that larger the Hamming Distance between two strings, more dissimilar will be those strings (and vice versa).

K-NEAREST-NEIGHBOR ALGORITHM

- 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.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type

of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



K-NN Algorithm:

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

How to select 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:

- 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:

- 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.

Example: Consider the below dataset & decide whether in what sport can Angelina with age as 5 & gender as female can be put into which sport (Football, Cricket, Neither)

Name	Age	Gender	Sport
Ajay	32	M	Football
Mark	40	M	Neither
Sara	16	F	Cricket
Zaira	14	F	Cricket
Sachin	55	M	Neither
Rahul	40	M	Cricket
Pooja	20	F	Neither
Smith	15	M	Cricket
Laxmi	55	F	Football
Michael	15	M	Football

Assuming k=3 for the above dataset. In the above dataset the attribute Age is numeric & the attribute Gender is discrete. Now, convert the discrete data to the numeric data with male as 0 & female as 1. So, we got both the attributes Age & Gender in numeric form.

Now, we need to find out the distance between Angelina with all the others individually by using the Euclidean distance formula

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

Note: The distance can also be calculated by using Manhattan distance (or) Minkowski distance.

Now let us calculate the distance between Ajay & Angelina by using the Euclidean distance formula which is as follows

$$\sqrt{(5\text{-}32)^2 + (1\text{-}0)^2}$$

$$\sqrt{729 + 1} = 27.02$$

Similarly, we need to calculate all the other distances also by using Euclidean distance formula. Then the dataset will be as follows

Name	Age	Gender	Distance	Sport
Ajay	32	0	27.02	Football
Mark	40	0	35.01	Neither
Sara	16	1	11.00	Cricket
Zaira	14	1	9.00	Cricket
Sachin	55	0	50.01	Neither
Rahul	40	0	35.01	Cricket
Pooja	20	1	15.00	Neither
Smith	15	0	10.05	Cricket
Laxmi	55	1	50.00	Football
Michael	15	0	10.05	Football

As assumed k=3, now we need to look to the 3 closest values. In the above table Zaira, Smith & Michael are closest records to Angelina. Looking at the classification of the 3 nearest neighbors Zaira & Smith like Cricket & Michael likes Football. Since the class of Cricket is most common among the 3 nearest neighbors, then KNN will predict that Angelina will be in the sport of Cricket.

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.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Decision Tree Terminologies:

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

- Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- Branch/Sub Tree: A tree formed by splitting the tree.
- Pruning: Pruning is the process of removing the unwanted branches from the tree.
- Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Tree Representation:

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.
- Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.
- An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute.
- This process is then repeated for the subtree rooted at the new node.
- In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

Decision tree based ID3 Algorithm:

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

Entropy: Entropy measures the impurity of a collection of examples.

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Where, p_+ is the proportion of positive examples in S

p_- is the proportion of negative examples in S

Information Gain: Information Gain is the expected reduction in entropy caused by partitioning the examples according to this attribute. The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

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

Where, Values (A) is the set of all possible values for attribute A

S_V is the subset of S for which attribute A has value v

Example:

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

The above dataset consists of 14 examples& 4 attributes (Outlook, Temperature, Humidity, Wind).If we want to draw a decision tree using ID3 algorithm, first we need to understand which attribute is giving maximum information out of available attributes.

First we need to calculate the information gain of every attribute. The attribute which is having maximum information gain is considered as the root node.

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

Attribute: Outlook

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

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

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

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

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

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

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

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

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

Attribute: Temp

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

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

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

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

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

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

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

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

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

Attribute: Humidity

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

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

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

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

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

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

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

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

Attribute: Wind

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

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

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

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

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

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

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

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

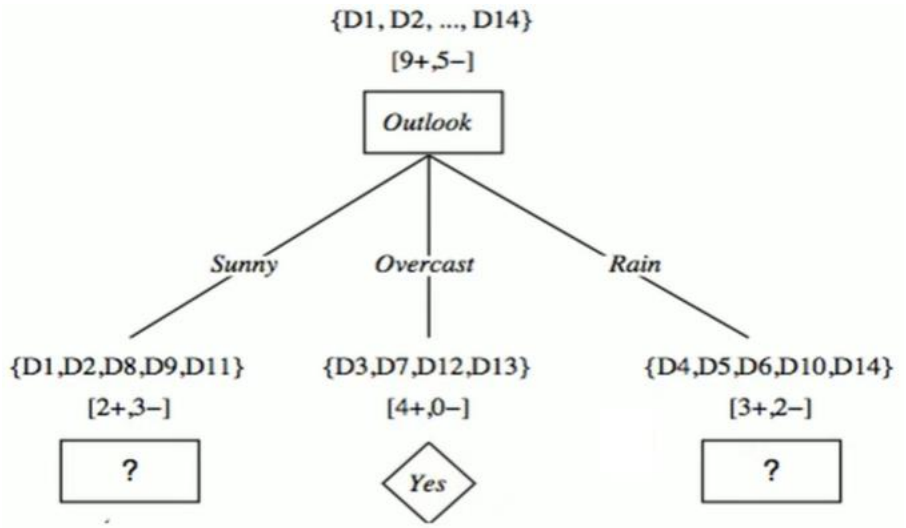
$$Gain(S, Outlook) = 0.2464$$

$$Gain(S, Temp) = 0.0289$$

$$Gain(S, Humidity) = 0.1516$$

$$Gain(S, Wind) = 0.0478$$

From the above, Outlook attribute has the maximum information gain. So, Outlook is the root node.



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

Attribute: Temp

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

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

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

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

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

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

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

$$= Entropy(S) - \frac{2}{5}Entropy(S_{Hot}) - \frac{2}{5}Entropy(S_{Mild}) - \frac{1}{5}Entropy(S_{Cool})$$

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

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

Attribute: Humidity

$Values\ (Humidity) = High, Normal$

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

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

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

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

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

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

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

Attribute: Wind

$Values\ (Wind) = Strong, Weak$

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

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

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

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

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

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

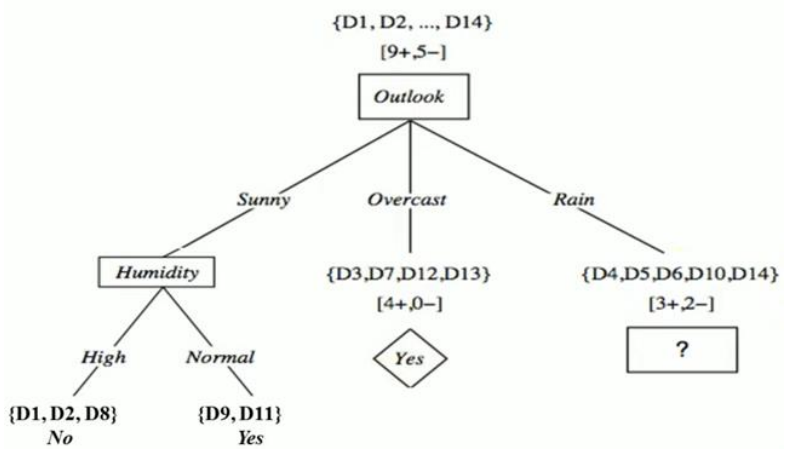
Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

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

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

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

From the above, Humidity attribute has the maximum information gain. So, the tree is as follows



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

Attribute: Temp

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

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

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

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

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

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

$$Gain(S_{Rain}, Temp) = Entropy(S) - \frac{0}{5}Entropy(S_{Hot}) - \frac{3}{5}Entropy(S_{Mild}) - \frac{2}{5}Entropy(S_{Cool})$$

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

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

Attribute: Humidity

$Values\ (Humidity) = High, Normal$

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

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

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

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

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

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

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

Attribute: Wind

$Values\ (wind) = Strong, Weak$

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

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

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

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

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

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

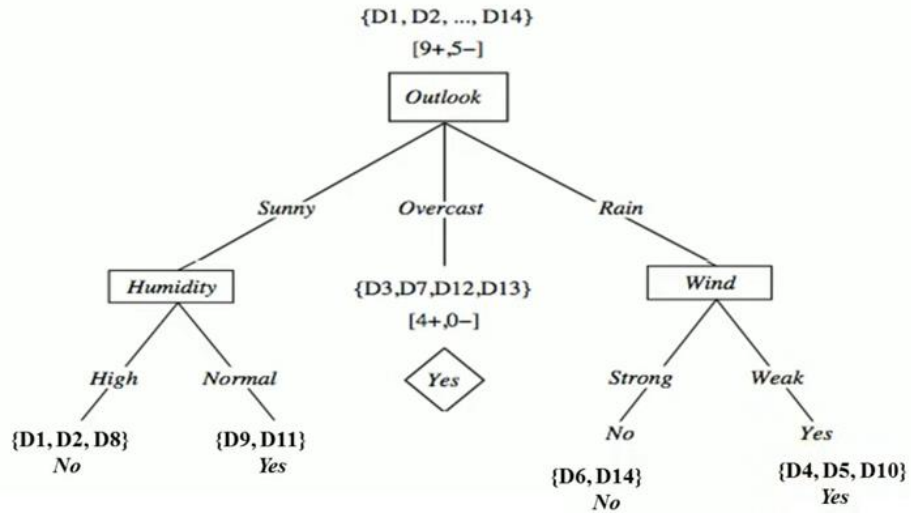
Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

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

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

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

From the above, Wind attribute has the maximum information gain. Finally, the tree is as follows



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.

NAIVE 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 help 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.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes:

- 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

Working of Naïve Bayes' Classifier: Consider a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So, to solve this, we need to follow the below steps:

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

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

S.No	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	4

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14=0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes' Theorem:

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

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

P(Sunny)= 0.35

P(Yes)=0.71

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

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

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

P(No)= 0.29

P(Sunny)= 0.35

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

So as we can see from the above calculation that P(Yes|Sunny)>P(No|Sunny). Hence on a Sunny day, Player can play the game.

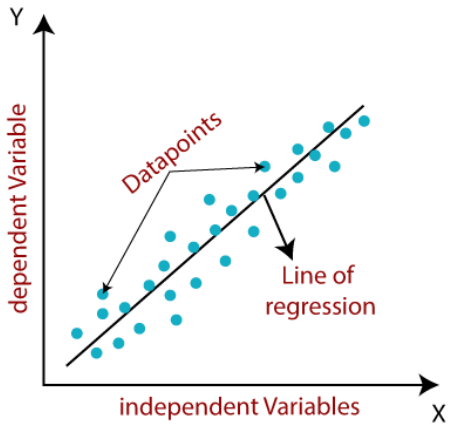
2. LINEAR MODELS

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 (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. 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.

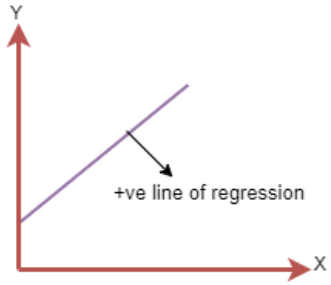
Types of Linear Regression

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

Linear Regression Line

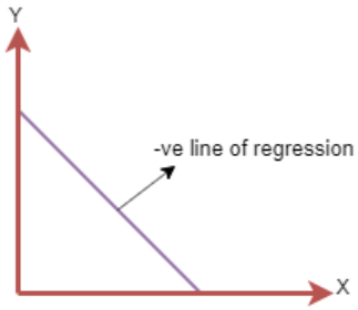
A linear line showing the relationship between the dependent and independent variables is called a regression line.

- 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: $Y= a_0+a_1x$

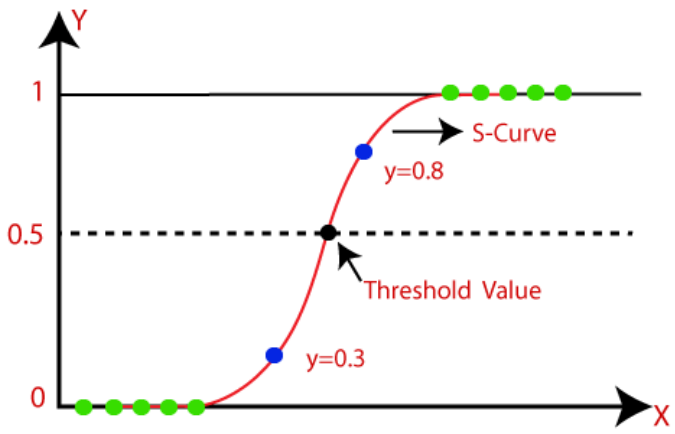
- Negative Linear Relationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



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

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).



Logistic Function (Sigmoid Function)

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

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

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

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

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

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

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

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

Type of Logistic Regression

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

Linear Regression VS Logistic Regression

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear Regression is used for solving Regression problem.	Logistic regression is used for solving Classification problems.
In Linear regression, we predict the value of continuous variables.	In logistic Regression, we predict the values of categorical variables.
In linear regression, we find the best fit line, by which we can easily predict the output.	In Logistic Regression, we find the S-curve by which we can classify the samples.

Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for estimation of accuracy.
The output for Linear Regression must be a continuous value, such as price, age, etc.	The output of Logistic Regression must be a Categorical value such as 0 or 1, Yes or No, etc.
In Linear regression, it is required that relationship between dependent variable and independent variable must be linear.	In Logistic regression, it is not required to have the linear relationship between the dependent and independent variable.
In linear regression, there may be collinearity between the independent variables,	In logistic regression, there should not be collinearity between the independent variable.

GENERALIZED LINEAR MODELS (GLM)

Generalized Linear Model (GLM) is an advanced statistical modelling technique formulated by John Nelder and Robert Wedderburn in 1972. Generalized linear models (GLMs) explain how Linear regression and Logistic regression are a member of a much broader class of models. GLMs can be used to construct the models for regression and classification problems by using the type of distribution which best describes the data or labels given for training the model.

- Binary classification data – Bernoulli distribution
- Real valued data – Gaussian distribution
- Count-data – Poisson distribution

Bernoulli Distribution: Bernoulli distribution is a discrete distribution in which the random variable has only two possible outcomes and a single trial known as a Bernoulli trial. p is the expected value of the Bernoulli random variable, which is known as the Bernoulli distribution parameter. The experiment's outcome can have only two values i.e, 0 or 1

$$P_X(x) = \begin{cases} 1 & \text{For } x = 1 \\ 0 & \text{otherwise} \end{cases}$$

Here x is the discrete random variable that can take only one value.

Examples of Bernoulli Distributions:

- When tossed a coin, will it be head or tail?
- Will a student pass or fail a test?
- Will India win or lose the match?
- Chances of job application getting accepted or rejected?

Conditions of Bernoulli Distribution:

- There should be only two possible outcomes of your trial.
- Each of the two outcomes should have a fixed probability of occurrence.
- The trials should be independent of each other.

If these conditions are met, it can be considered a Bernoulli trial.

Gaussian distribution: The Gaussian distribution, also known as the normal distribution, is a continuous probability distribution that is widely used in statistical modeling and Machine Learning. It is a bell-shaped curve that is symmetrical around its mean and is characterized by its mean and standard deviation.

- A wide variety of real-world occurrences may be adequately modeled by the Gaussian distribution.
- The Gaussian distribution is important because it is a common and useful model for data, and because it allows for efficient calculations in statistical analysis.

Gaussian Distribution formula is

$$f(x) = (1 / \text{sqrt}(2 * \text{pi} * \text{sigma}^2)) * \exp(-((x - \text{mu})^2) / (2 * \text{sigma}^2))$$

Where,

- X is a real number representing a possible value of a continuous random variable;
- mu is the mean of the distribution, and sigma is the standard deviation;
- (1 / sqrt(2 * pi * sigma^2))– is the normalization factor that ensures that the area under the curve of the distribution is equal to 1; and
- exp(-((x – mu)^2) / (2 * sigma^2))– is a bell-shaped curve that is centered at the mean, mu, and has a standard deviation of sigma.

Poisson Distribution: Poisson Distribution is a probability distribution that is used to show how many times an event occurs over a specific period. It is the discrete probability distribution of the number of events occurring in a given time period, given the average number of times the event occurs over that time period. It is the distribution related to probabilities of events that are extremely rare but have a large number of independent opportunities of occurrence.

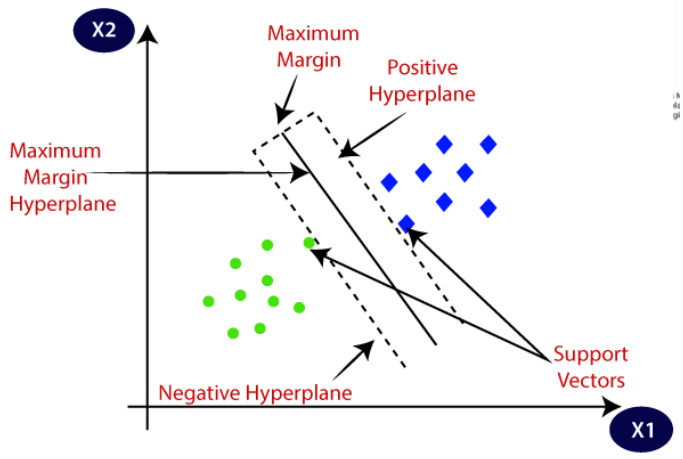
$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

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. SVM algorithm can be used for Face detection, image classification, text categorization, etc. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM:

- 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.

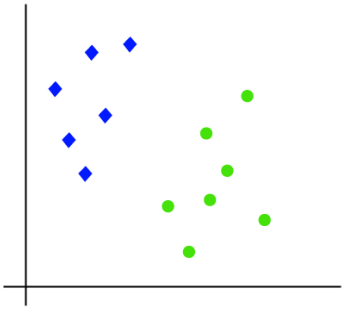
Hyperplaneand Support Vectors in the SVM Algorithm:

- 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.

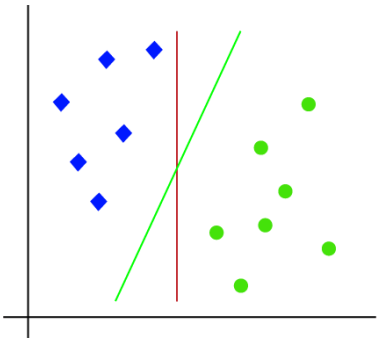
  The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum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

Working of SVM:

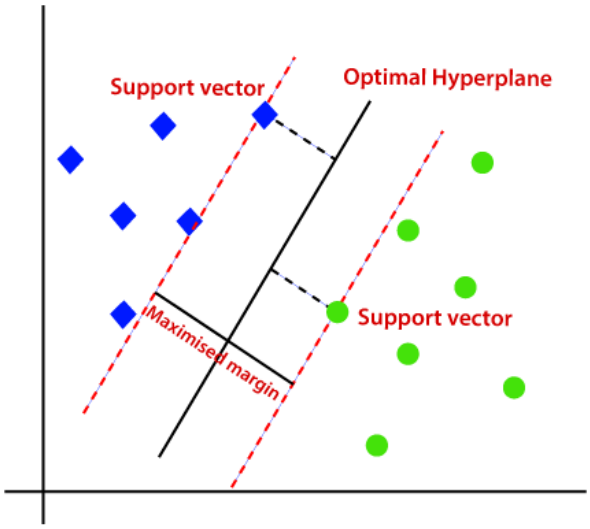
Linear SVM: The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below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



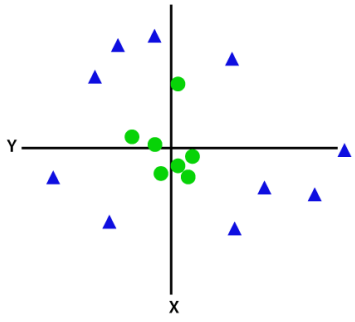
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



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.



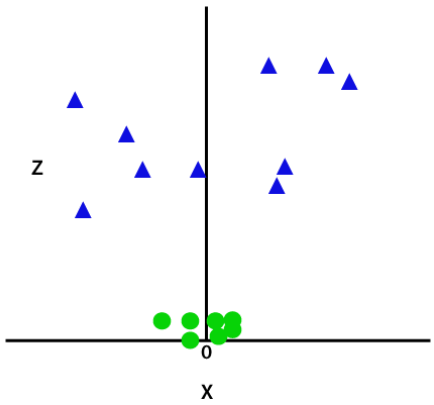
Non-Linear SVM: If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



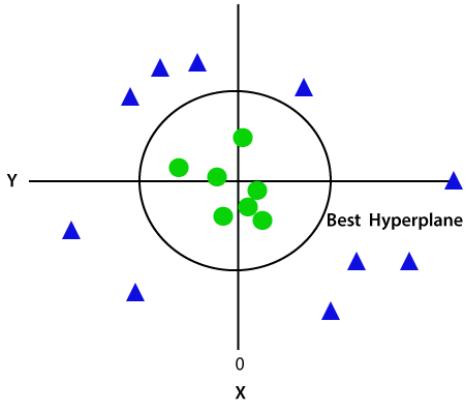
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$\mathbf{Z = x^2 + y^2}$$

By adding the third dimension, the sample space will become as 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.

2. BINARY CLASSIFICATION

MULTICLASS/STRUCTURED OUTPUTS

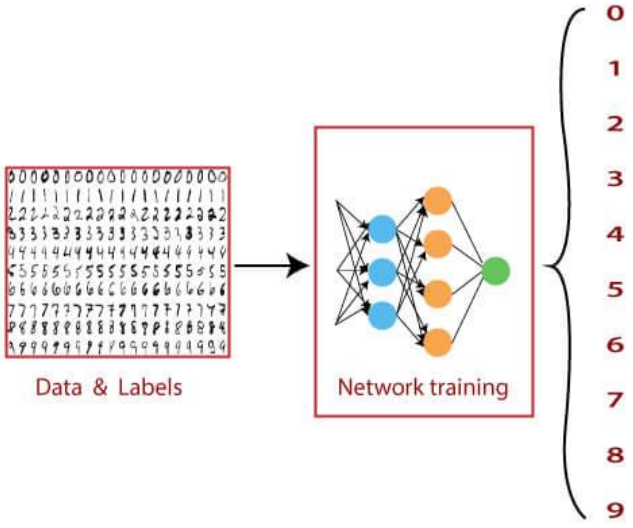
Multi-output classification is a type of machine learning that predicts multiple outputs simultaneously. In multi-output classification, the model will give two or more outputs after making any prediction. In other types of classifications, the model usually predicts only a single output.

An example of a multi-output classification model is a model that predicts the type and color of fruit simultaneously. The type of fruit can be orange, mango and pineapple. The color can be red, green, yellow, and orange. The multi-output classification solves this problem and gives two prediction results.

MNIST (**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology)

MNIST database is a large database of handwritten numbers or digits that are used for training various image processing systems. The dataset also widely used for training and testing in the field of machine learning. The set of images in the MNIST database are a combination of two of NIST's databases: Special Database 1 and Special Database 3.

- The MNIST dataset has 60,000 training images and 10,000 testing images.
- The MNIST dataset can be online, and it is essentially a database of various handwritten digits.
- The MNIST dataset has a large amount of data and is commonly used to demonstrate the real power of deep neural networks.



- Our brain and eyes work together to recognize any numbered image. Our mind is a potent tool, and it's capable of categorizing any image quickly.

- There are so many shapes of a number, and our mind can easily recognize these shapes and determine what number is it, but the same task is not simple for a computer to complete. There is only one way to do this, which is the use of deep neural network which allows us to train a computer to classify the handwritten digits effectively.

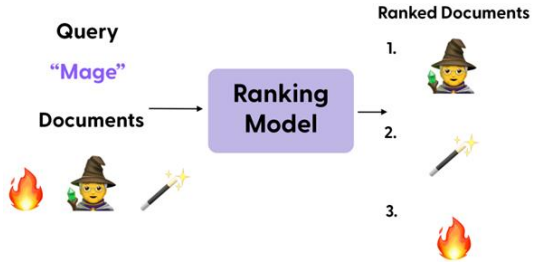
In the MNIST dataset, a single data point comes in the form of an image. These images included in the MNIST dataset are typical of 28*28 pixels such as 28 pixels crossing the horizontal axis and 28 pixels crossing the vertical axis. This means that a single image from the MNIST database has a total of 784 pixels that must be analyzed. The input layer of our neural network has 784 nodes to explain one of these images.

RANKING

- Ranking is a type of supervised machine learning (ML) that uses labeled datasets to train its data and models to classify future data to predict outcomes.
- The goal of a ranking model is to sort data in an optimal and relevant order.
- Ranking is a machine learning technique to rank items.
- Ranking is useful for many applications in information retrieval such as e-commerce, social networks, recommendation systems, and so on. For example, a user searches for an article or an item to buy online.
- People search for a topic, while the ranking algorithm reorders search results based on the PageRank, and the search engine is able to display the most relevant results to its customers.

How does ranking work?

Ranking is incredibly versatile and dependent on the data. Ranking models are made up of 2 main factors: queries and documents. Queries are any input value, such as a question on Google or an interaction on an e-commerce site. Documents are the output value or results of the query. Given the query, and the associated documents, a function, given a list of parameters to rank on, will score the documents to be sorted in order of relevancy.



The machine learning algorithm learning to rank takes the scores from this model, and uses them to predict future outcomes on a new and unseen list of documents.

3. ENSEMBLE LEARNING AND RANDOM FORESTS

INTRODUCTION

Ensemble learning is one of the most powerful machine learning techniques that use the combined output of two or more models/weak learners and solve a particular computational intelligence problem. E.g., a Random Forest algorithm is an ensemble of various decision trees combined.

Ensemble learning is primarily used to improve the model performance, such as classification, prediction, function approximation, etc. In simple words, we can summarise the ensemble learning as follows:

"An ensembled model is a machine learning model that combines the predictions from two or more models." There are 3 most common ensemble learning methods in machine learning

- Bagging
- Boosting
- Stacking

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:

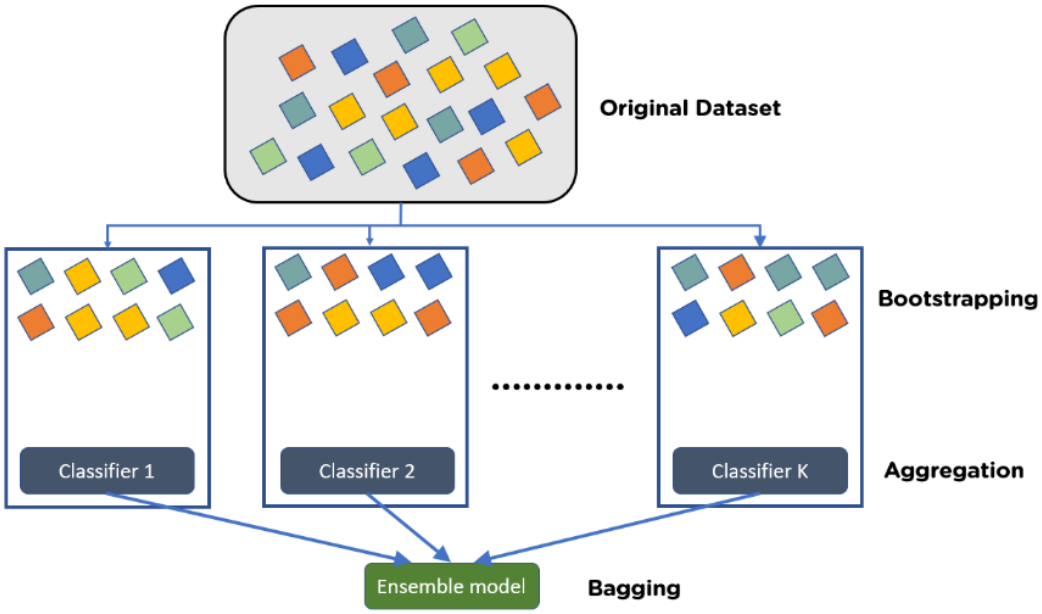
- Hard Voting: In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
- 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.

Note: Make sure to include a variety of models to feed a Voting Classifier to be sure that the error made by one might be resolved by the other.

BAGGING

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



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

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

Advantages of using Random Forest technique:

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

Disadvantages of using Random Forest technique:

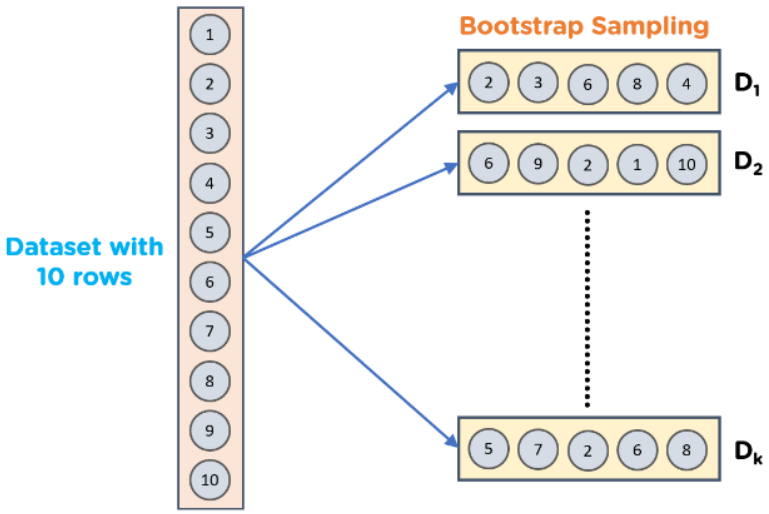
- Since the last prediction depends on the mean predictions from subset trees, it won't give precise value for the regression model.

BOOSTING

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

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

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.

Difference between Bagging and Boosting:

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

PASTING

Pasting is an ensemble technique similar to bagging except for the fact that in pasting sampling is done without replacement i.e. an observation can be present in only one subset. Since pasting limits diversity of models its performance with is suboptimal when compared to bagging, particularly in case of small datasets. However, pasting is preferred over bagging in case of so large datasets, owing to computational efficiency.

STACKING

Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future.

This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved. In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

Stacking is also known as a stacked generalization and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

Architecture of Stacking:

The architecture of the stacking model is designed in such as way that it consists of two or more base/learner's models and a meta-model that combines the predictions of the base models. These base models are called level 0 models, and the meta-model is known as the level 1 model. So, the Stacking ensemble method includes original (training) data, primary level models, primary level prediction, secondary level model, and final prediction. The basic architecture of stacking can be represented as shown below the image.

- Original data: This data is divided into n-folds and is also considered test data or training data.
- Base models: These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- Level-0 Predictions: Each base model is triggered on some training data and provides different predictions, which are known as level-0 predictions.
- Meta Model: The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the level-1 model.
- Level-1 Prediction: The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models,

i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

Steps to implement Stacking models:

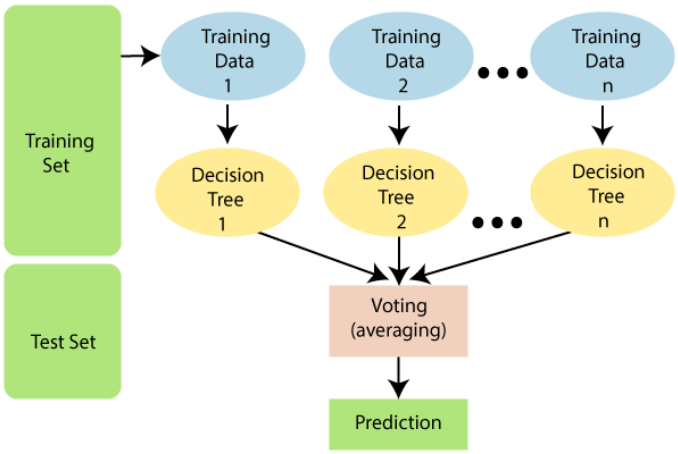
- Split training data sets into n-folds using the RepeatedStratifiedKFold as this is the most common approach to preparing training datasets for meta-models.
- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.
- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model.
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

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.



Working of Random Forest Algorithm:

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:

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:

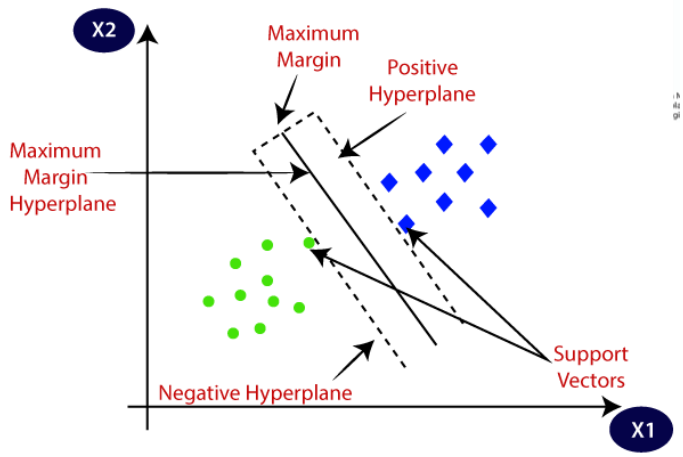
- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

3. 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. SVM algorithm can be used for Face detection, image classification, text categorization, etc. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM:

- 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.

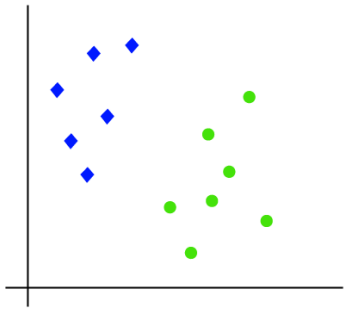
Hyperplaneand Support Vectors in the SVM Algorithm:

- 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.

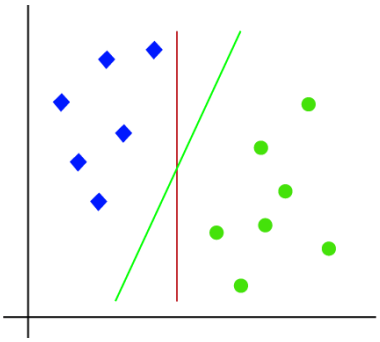
  The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum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

Working of SVM:

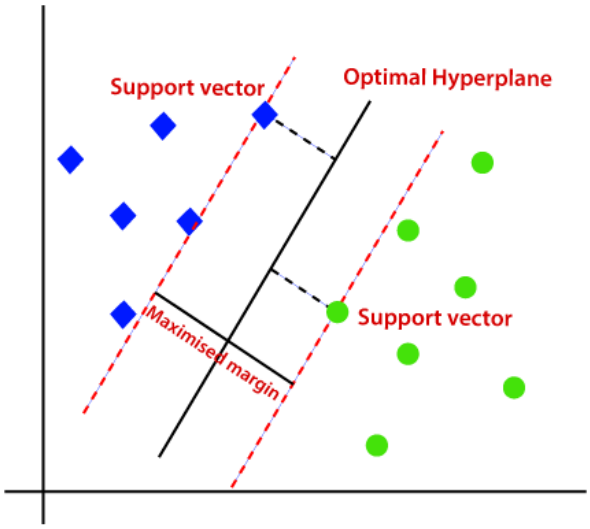
Linear SVM: The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below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



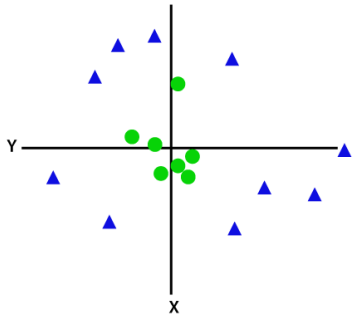
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



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.



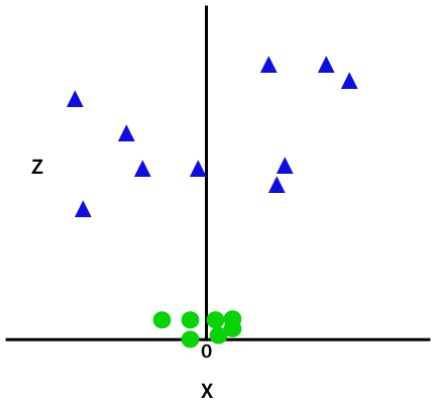
Non-Linear SVM: If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



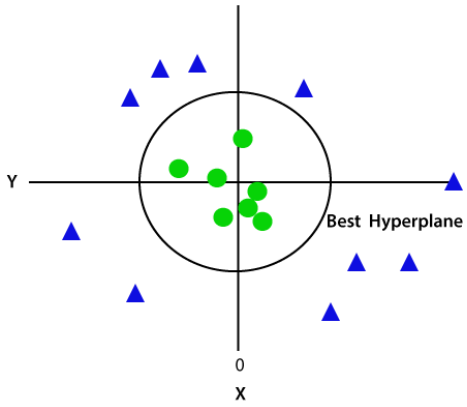
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$\mathbf{Z = x^2 + y^2}$$

By adding the third dimension, the sample space will become as 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.

4. UNSUPERVISED LEARNING TECHNIQUES

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 is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset. The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

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

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

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). Below are the main clustering methods used in Machine learning:

1. 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 examples of partitioning clustering are the K-Means Clustering algorithm, CLARANS (Clustering Large Applications based upon Randomized Search), etc.
2. Density-Based Clustering: These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Examples are DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure), etc.

3. Distribution Model-Based Clustering: In this, 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).
4. 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. Examples are Agglomerative (bottom-up approach) & Divisive (top-down approach).
5. 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:

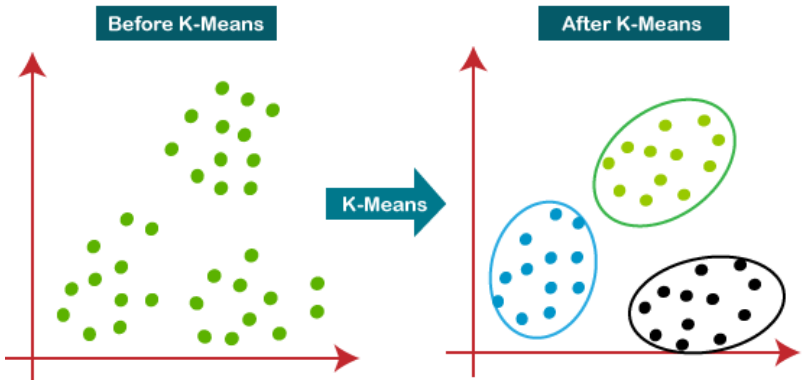
- Identification of Cancer Cells: The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- Search Engines: Search engines also work on the clustering technique. The search result appears based on 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.
- Biology: It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- 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 CLUSTERING ALGORITHM

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters "It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties."

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

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



<u>Working of K-Means Algorithm</u>:

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

Example of K-Means Algorithm:

Data Points	X	Y
A1	2	10
A2	2	5
A3	8	4
B1	5	8
B2	7	5
B3	6	4
C1	1	2
C2	4	9

In the given example, there are 8 data points. Consider the initial centroids as A1, B1 & C1. So, the value of K=3. If initially any centroids are given we need to consider them if not we can selects any of the data points as centroid. The formula for calculating distance is

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

Initial centroids are A1: (2, 10) , B1: (5, 8) , C1: (1, 2). Initially we need to calculate the distance by using above formula. Once the distance has been calculated, we need to assign the data points to one of the cluster. So, we need to consider the distances and the one which is having the smallest value, to that we can assign the cluster.

Data Points			Distance to						Cluster	New Cluster
			2	10	5	8	1	2		
A1	2	10	0.00		3.61		8.06		1	
A2	2	5	5.00		4.24		3.16		3	
A3	8	4	8.49		5.00		7.28		2	
B1	5	8	3.61		0.00		7.21		2	
B2	7	5	7.07		3.61		6.71		2	
B3	6	4	7.21		4.12		5.39		2	
C1	1	2	8.06		7.21		0.00		3	
C2	4	9	2.24		1.41		7.62		2	

After the 1st iteration the data points A1 is assigned to 1st Cluster, A3, B1, B2, B3, C2 are assigned to the 2nd cluster and A2, C1 are assigned to the 3rd cluster. Now we need to calculate the new centroids. The new centroids are A1: (2, 10), B1: (6, 6) & C1: (1.5, 3.5). After considering the new centroids:

Data Points			Distance to						Cluster	New Cluster
			2	10	6	6	1.5	3.5		
A1	2	10	0.00		5.66		6.52		1	1
A2	2	5	5.00		4.12		1.58		3	3
A3	8	4	8.49		2.83		6.52		2	2
B1	5	8	3.61		2.24		5.70		2	2
B2	7	5	7.07		1.41		5.70		2	2
B3	6	4	7.21		2.00		4.53		2	2
C1	1	2	8.06		6.40		1.58		3	3
C2	4	9	2.24		3.61		6.04		2	1

If we look into the above table C2 was assigned to 2nd Cluster, but now it is assigned to 1st cluster, which means the data point has been moved from one cluster to other. So, we need to calculate the new centroids again. The new centroids are A1: (3, 9.5), B1: (6.5, 5.25) & C1: (1.5, 3.5). Here, the new cluster will become the current cluster. After considering the new centroids:

Data Points			Distance to						Cluster	New Cluster
			3	9.5	6.5	5.25	1.5	3.5		
A1	2	10	1.12		6.54		6.52		1	1
A2	2	5	4.61		4.51		1.58		3	3
A3	8	4	7.43		1.95		6.52		2	2
B1	5	8	2.50		3.13		5.70		2	1
B2	7	5	6.02		0.56		5.70		2	2
B3	6	4	6.26		1.35		4.53		2	2
C1	1	2	7.76		6.39		1.58		3	3
C2	4	9	1.12		4.51		6.04		1	1

If we look into the above table B1 was assigned to 2nd Cluster, but now it is assigned to 1st cluster, which means the data point has been moved from one cluster to other. So, we need to calculate the new centroids again. The new centroids are A1: (3.67, 9), B1: (7, 4.33)

& C1: (1.5, 3.5). Here, the new cluster will become the current cluster. After considering the new centroids:

Data Points			Distance to						Cluster	New Cluster
			3.67	9	7	4.33	1.5	3.5		
A1	2	10	1.94		7.56		6.52		1	1
A2	2	5	4.33		5.04		1.58		3	3
A3	8	4	6.62		1.05		6.52		2	2
B1	5	8	1.67		4.18		5.70		1	1
B2	7	5	5.21		0.67		5.70		2	2
B3	6	4	5.52		1.05		4.53		2	2
C1	1	2	7.49		6.44		1.58		3	3
C2	4	9	0.33		5.55		6.04		1	1

Finally, cluster & new cluster became exactly same. A1, B1, C2 belongs to 1st cluster. A3, B2, B3 belongs to 2nd cluster & A2, C1 belongs to 3rd cluster.

LIMITATIONS OF K-MEANS

- Choosing the right number of clusters: K-means requires the user to specify the number of clusters to be generated, which can be difficult to determine. A poor choice of K can lead to suboptimal clustering results.
- Outliers can skew results: K-means assumes that all data points are equally important in the clustering process, which can lead to outliers skewing the results. Outliers may be assigned to a cluster, leading to inaccurate cluster assignments for the other data points.
- Sensitive to outliers: K-means is sensitive to outliers or noise data, which can distort the resulting clusters.
- Assumes spherical clusters: K-means assumes that the clusters are spherical and have equal variances, which is not always true in real-world scenarios.
- Limited applicability to non-numerical data: K-means is designed for numerical data and does not handle categorical or textual data well.
- Lack of robustness: K-means can be sensitive to the distribution of the data, and the resulting clusters may not be robust to small changes in the input data.

- Difficulty in handling high-dimensional data: K-means is less effective in high-dimensional data, where the "curse of dimensionality" makes it harder to identify meaningful clusters.

USING CLUSTERING FOR IMAGE SEGMENTATION

It is a method to perform Image Segmentation of pixel-wise segmentation. In this type of segmentation, we try to cluster the pixels that are together. There are two approaches for performing the Segmentation by clustering

- Clustering by Merging
- Clustering by Divisive

Clustering by merging or Agglomerative Clustering:

In this approach, we follow the bottom-up approach, which means we assign the pixel closest to the cluster. The algorithm for performing the agglomerative clustering as follows:

- Take each point as a separate cluster.
    - For a given number of epochs or until clustering is satisfactory.
- Merge two clusters with the smallest inter-cluster distance.
- Repeat the above step

The agglomerative clustering is represented by Dendrogram. It can be performed in 3 methods: by selecting the closest pair for merging, by selecting the farthest pair for merging, or by selecting the pair which is at an average distance (neither closest nor farthest).

Clustering by division or Divisive splitting:

In this approach, we follow the top-down approach, which means we assign the pixel closest to the cluster. The algorithm for performing the agglomerative clustering as follows:

- Construct a single cluster containing all points.
- For a given number of epochs or until clustering is satisfactory.
    - Split the cluster into two clusters with the largest inter-cluster distance.
- Repeat the above steps.

K-Means Clustering:

K-means clustering is a very popular clustering algorithm which applied when we have a dataset with labels unknown. The goal is to find certain groups based on some kind of similarity in the data with the number of groups represented by K. This algorithm is generally used in areas like market segmentation, customer segmentation, etc. But, it can also be used to segment different objects in the images on the basis of the pixel values.

Algorithm for Image Segmentation:

1. First, we need to select the value of K in K-means clustering.
2. Select a feature vector for every pixel (color values such as RGB value, texture etc.).
3. Define a similarity measure b/w feature vectors such as Euclidean distance to measure the similarity b/w any two points/pixel.
4. Apply K-means algorithm to the cluster centers
5. Apply connected component's algorithm.
6. Combine any component of size less than the threshold to an adjacent component that is similar to it until you can't combine more.

USING CLUSTERING FOR PREPROCESSING

Clustering is a useful technique for preprocessing data in machine learning. It can help in identifying patterns and structures within the data that may be useful for subsequent modeling steps.

One common use case for clustering in preprocessing is to identify groups of similar instances in the data. This can be helpful in situations where there are a large number of instances and it is difficult to manually label each one for classification. By clustering the instances, we can group similar instances together and then assign a label to each cluster based on the majority class of the instances within the cluster. This can save time and effort in labeling the data and can also improve the quality of the labels by leveraging the similarities between instances.

Another use case for clustering in preprocessing is feature selection or feature engineering. Clustering can be used to identify groups of highly correlated features, which can then be combined or reduced to a smaller set of features that better capture the underlying structure of the data. This can help to reduce the dimensionality of the data and improve the performance of subsequent modeling steps.

Finally clustering can be a useful preprocessing technique in machine learning for identifying patterns and structures in the data, grouping similar instances together, and selecting or engineering features.

USING CLUSTERING FOR SEMI-SUPERVISED LEARNING

Clustering can be a useful technique for semi-supervised learning, which is a type of machine learning where some of the data is labeled and some of it is not. In semi-supervised learning, the goal is to use the labeled data to help guide the learning process for the unlabeled data.

One way to use clustering for semi-supervised learning is to first cluster the unlabeled data using an unsupervised clustering algorithm such as k-means or hierarchical clustering or spectral clustering. Once the data has been clustered, the labels of the clustered data can be propagated to the individual data points within each cluster. This means that each data point within a cluster will be assigned the same label as the centroid of the cluster.

Once the labels have been propagated, the labeled and unlabeled data can be used together to train a supervised learning model. The labeled data provides the model with examples of what the correct output should be, while the unlabeled data helps the model learn the underlying patterns and structure of the data.

One important consideration when using clustering for semi-supervised learning is the choice of clustering algorithm and the number of clusters to use. The number of clusters can have a significant impact on the quality of the labels that are propagated to the individual data points, and choosing the optimal number of clusters can be challenging. There are several methods for semi-supervised clustering that can be divided into two classes which are as follows:

- Constraint-based semi-supervised clustering: It can be used based on user-provided labels or constraints to support the algorithm toward a more appropriate data partitioning. This contains modifying the objective function depending on constraints or initializing and constraining the clustering process depending on the labeled objects.
- Distance-based semi-supervised clustering: It can be used to employ an adaptive distance measure that is trained to satisfy the labels or constraints in the supervised data. Multiple adaptive distance measures have been utilized, including string-edit

distance trained using Expectation-Maximization (EM), and Euclidean distance changed by the shortest distance algorithm.

Overall, clustering can be a useful technique for semi-supervised learning, but it should be used in conjunction with other methods and with careful consideration given to the choice of clustering algorithm and number of clusters.

DBSCAN (Density-based spatial clustering of applications with noise)

Partitioning and hierarchical methods are designed to find spherical-shaped clusters. They have difficulty finding clusters of arbitrary shape such as "S" shape and oval clusters. Density based clustering methods can be used to find clusters of arbitrary shape (or) non spherical shape.

DBSCAN is one of the most popular unsupervised learning algorithms. The DBSCAN algorithm is based on the intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

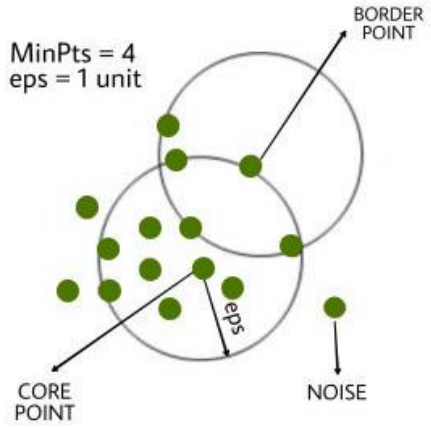
DBSCAN Algorithm requires two parameters:

- eps : It defines the neighbourhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers.
- MinPts: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen.

DBSCAN Algorithm has 3 types of data points:

- Core Point: A point is a core point if it has more than MinPts points within eps

- Border Point: A point which has fewer than MinPts within eps but it is in the neighbourhood of a core point
- Noise or outlier: A point which is not a core point or border point



Other Parameters:

- A point X is **directly density-reachable** from point Y w.r.t epsilon, minPoints if,
    - X belongs to the neighborhood of Y, i.e, dist(X, Y) <= epsilon
    - Y is a core point
- X is **density-reachable** from Y with X being directly density-reachable from P2, P2 from P3, and P3 from Y. But, the inverse of this is not valid.
- A point X is **density-connecte**d from point Y w.r.t epsilon and minPoints if there exists a point O such that both X and Y are density-reachable from O w.r.t to epsilon and minPoints.

DBSCAN Algorithm:

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

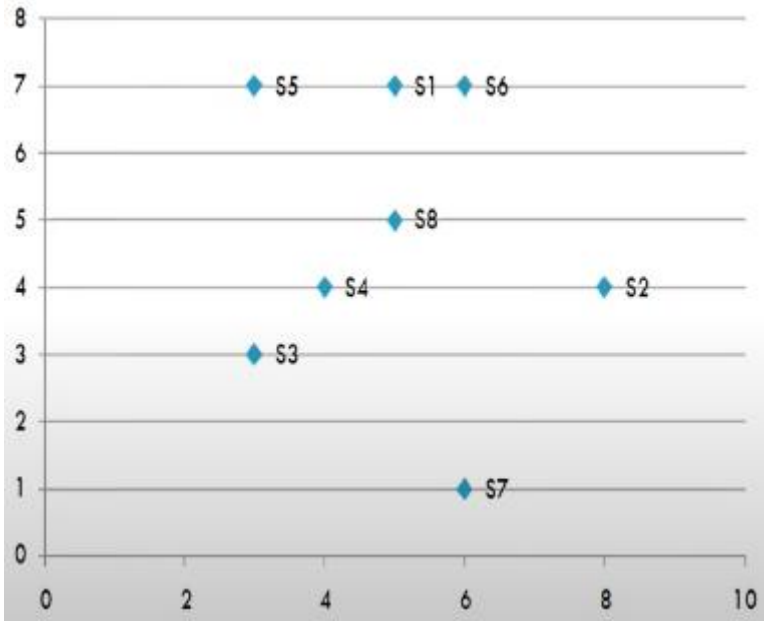
neighbor of d, d is neighbor of e, which in turn is neighbor of a implies that b is neighbor of a.

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

Example:

S1	5	7
S2	8	4
S3	3	3
S4	4	4
S5	3	7
S6	6	7
S7	6	1
S8	5	5

Form the clusters with ε = 3.5 and MinPts = 3



Calculate the Euclidean distance among all the points

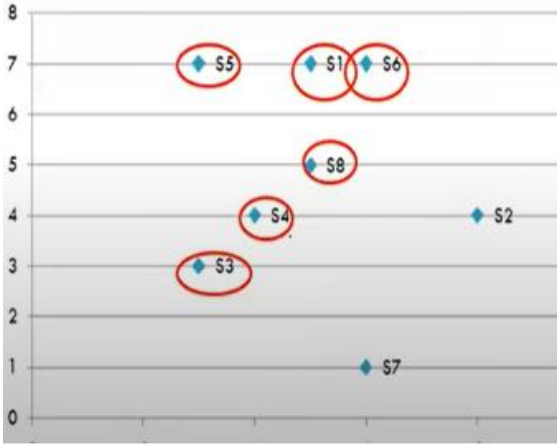
	S1	S2	S3	S4	S5	S6	S7	S8
S1	0							
S2	4.24	0						
S3	4.47	5.1	0					
S4	3.16	4	1.41	0				
S5	2	5.83	4	3.16	0			
S6	1	3.61	5	3.61	3	0		
S7	6.08	3.61	3.61	3.61	6.71	6	0	
S8	2	3.16	2.83	1.41	2.83	2.24	4.12	0

Identify the neighbours of each point for ε = 3.5

Point and their neighbours within the boundary of radius ε = 3.5			
S1 : S4, S5, S6, S8	S2 : S8	S3 : S4, S8	S4 : S1, S5, S3, S8
S5 : S1, S4, S6, S8	S6 : S1, S5, S8	S7 : None	S8 : S1, S2, S3, S4, S5, S6

Point and their neighbours within the boundary of radius ε= 3.5			
S1: S4, S5, S6, S8	S2: S8	S3:S4, S8	S4: S1, S3, S8
S5:S1, S4, S8	S6: S1, S5, S8	S7: None	S8: S1, S2, S3, S4, S5, S6

S1, S3, S4, S5, S6 & S8 become the core points



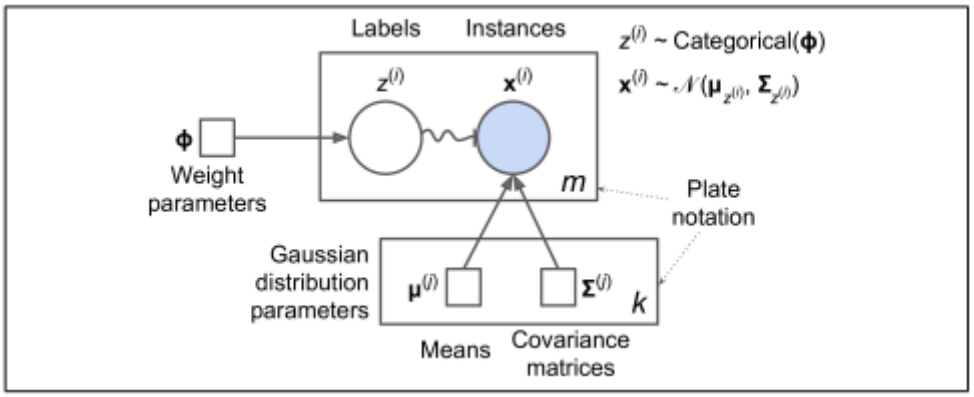
Identify the core points & noise points. Check for direct density reachable condition for noise points. If the density reachable condition is satisfied, convert noise to boundary point.

Point	Core/Noise	
S1	Core	
S2	Noise	Boundary
S3	Core	
S4	Core	
S5	Core	
S6	Core	
S7	Noise	Noise
S8	Core	

GAUSSIAN MIXTURES

A Gaussian mixture model (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown. All the instances generated from a single Gaussian distribution form a cluster that typically looks like an ellipsoid. Each cluster can have a different ellipsoidal shape, size, density and orientation.

This generative process can be represented as a graphical model. This is a graph which represents the structure of the conditional dependencies between random variables.



Gaussian Mixture Model

- The circles represent random variables
- The squares represent fixed values
- The large rectangles are called plates: they indicate that their content is repeated several times
- The number indicated at the bottom right hand side of each plate indicates how many times its content is repeated
- Each variable $z^{(i)}$ is drawn from the categorical distribution with weights ϕ. Each variable $x^{(i)}$ is drawn from the normal distribution with the mean and covariance matrix defined by its cluster $z^{(i)}$.
- The solid arrows represent conditional dependencies
- The squiggly arrow from $z^{(i)}$ to $x^{(i)}$ represents a switch: depending on the value of $z^{(i)}$, the instance $x^{(i)}$ will be sampled from a different Gaussian distribution
- Shaded nodes indicate that the value is known, so in this case only the random variables $x^{(i)}$ have known values: they are called observed variables. The unknown random variables $z^{(i)}$ are called latent variables.

Scikit-Learn's Gaussian Mixture class is as follows:

```
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=3, n_init=10)
gm.fit(X)
>>> gm.weights_
 >>> gm.means_
>>> gm.covariances_
```

We can check whether or not the algorithm converged and how many iterations it took:

```
>>> gm.converged_
True
 >>> gm.n_iter_
3
```

Now that you have an estimate of the location, size, shape, orientation and relative weight of each cluster, the model can easily assign each instance to the most likely cluster (hard clustering) or estimate the probability that it belongs to a particular cluster (soft clustering). For this, just use the predict() method for hard clustering, or the predict_proba() method for soft clustering:

```
>>> gm.predict(X)
>>> gm.predict_proba(X)
```

It is a generative model, meaning we can actually sample new instances from it

```
>>> X_new, y_new = gm.sample(6)
>>> X_new
>>> y_new
```

It is also possible to estimate the density of the model at any given location. This is achieved using the score_samples() method: for each instance it is given, this method estimates the log of the probability density function (PDF) at that location. The greater the score, the higher the density

```
>>> gm.score_samples(X)
```

If we compute the exponential of these scores, we get the value of the PDF at the location of the given instances. These are not probabilities, but probability densities: they can take on any positive value, not just between 0 and 1. To estimate the probability that an instance will fall within a particular region, we would have to integrate the PDF over that region.

Bayesian Gaussian Mixture Models:

Rather than manually searching for the optimal number of clusters, it is possible to use instead the Bayesian Gaussian Mixture class which is capable of giving weights equal (or close) to zero to unnecessary clusters. Just set the number of clusters n_components to a value that we have good reason to believe is greater than the optimal number of clusters, and the algorithm will eliminate the unnecessary clusters automatically

```
>>> from sklearn.mixture import BayesianGaussianMixture
>>>bgm = BayesianGaussianMixture(n_components=10, n_init=10, random_state=42)
>>> bgm.fit(X)
>>> np.round(bgm.weights_, 2)
```

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.

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.

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:

- 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
- It removes the redundant features (if present) by taking care of multicollinearity

Disadvantages of dimensionality Reduction:

- 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 OF DIMENSION REDUCTION

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

1. 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:
    - 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.
    - 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
    - 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.

2. 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:
    - Principal Component Analysis
    - Linear Discriminant Analysis
    - Kernel PCA
    - Quadratic Discriminant Analysis

COMMON TECHNIQUES OF DIMENSIONALITY REDUCTION

1. Principal Component Analysis: 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.
2. Backward Elimination: The backward feature elimination technique is mainly used while developing Linear Regression or Logistic Regression model.
3. Forward Feature Selection: Forward feature selection follows the inverse process of the backward elimination process. It means, in this technique, we don't eliminate the feature; instead, we will find the best features that can produce the highest increase in the performance of the model.
4. Missing Value Ratio: If a dataset has too many missing values, then we drop those variables as they do not carry much useful information. To perform this, we can set a threshold level, and if a variable has missing values more than that threshold, we will drop that variable.
5. Low Variance Filter: As same as missing value ratio technique, data columns with some changes in the data have less information. Therefore, we need to calculate the variance of each variable, and all data columns with variance lower than a given threshold are dropped because low variance features will not affect the target variable.
6. High Correlation Filter: High Correlation refers to the case when two variables carry approximately similar information. Due to this factor, the performance of the model can be degraded. This correlation between the independent numerical variable gives the calculated value of the correlation coefficient. If this value is higher than the threshold value, we can remove one of the variables from the dataset. We can

consider those variables or features that show a high correlation with the target variable.

7. Random Forest: Random Forest is a popular and very useful feature selection algorithm in machine learning. In this technique, we need to generate a large set of trees against the target variable, and with the help of usage statistics of each attribute, we need to find the subset of features.
8. Factor Analysis: Factor analysis is a technique in which each variable is kept within a group according to the correlation with other variables, it means variables within a group can have a high correlation between themselves, but they have a low correlation with variables of other groups.
9. Auto-Encoder: One of the popular methods of dimensionality reduction is auto-encoder. In this, the input is compressed into latent-space representation, and output is occurred using this representation. It has mainly two parts
    - Encoder: The function of the encoder is to compress the input to form the latent-space representation.
    - Decoder: The function of the decoder is to recreate the output from the latent-space representation.

PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It 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. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

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. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

PCA Algorithm:

Step 1. Data

We consider a dataset having n features or variables denoted by $X_1, X_2, \ldots, X_n$. Let there be N examples. Let the values of the i-th feature X_i be $X_{i1}, X_{i2}, \ldots, X_{iN}$

Features	Example 1	Example 2	$\cdots$	Example N
X_1	X_{11}	X_{12}	$\cdots$	X_{1N}
X_2	X_{21}	X_{22}	$\cdots$	X_{2N}
$\vdots$				
X_i	X_{i1}	X_{i2}	$\cdots$	X_{iN}
$\vdots$				
X_n	X_{n1}	X_{n2}	$\cdots$	X_{nN}

Step 2. Compute the means of the variables

We compute the mean $\bar{X}_i$ of the variable X_i:

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \cdots + X_{iN}).$$

Step 3. Calculate the covariance matrix

Consider the variables X_i and X_j (i and j need not be different). The covariance of the ordered pair (X_i, X_j) is defined as[1]

$$\text{Cov}(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j).$$

We calculate the following $n \times n$ matrix S called the covariance matrix of the data. The element in the i-th row j-th column is the covariance $\text{Cov}(X_i, X_j)$:

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \cdots & \text{Cov}(X_2, X_n) \\ \vdots & & & \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \cdots & \text{Cov}(X_n, X_n) \end{bmatrix}$$

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

Let S be the covariance matrix and let I be the identity matrix having the same dimension as the dimension of S.

i) Set up the equation:

$$\det(S - \lambda I) = 0.$$

This is a polynomial equation of degree n in λ. It has n real roots (some of the roots may be repeated) and these roots are the eigenvalues of S. We find the n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$

ii) If $\lambda = \lambda'$ is an eigenvalue, then the corresponding eigenvector is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

such that

$$(S - \lambda' I)U = 0.$$

(This is a system of n homogeneous linear equations in $u_1, u_2, \ldots, u_n$ and it always has a nontrivial solution.) We next find a set of n orthogonal eigenvectors $U_1, U_2, \ldots, U_n$ such that U_i is an eigenvector corresponding to λ_i.[2]

iii) We now normalise the eigenvectors. Given any vector X we normalise it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is defined as

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Given any eigenvector U, the corresponding normalised eigenvector is computed as

$$\frac{1}{\|U\|} U.$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{\|U_i\|} U_i, \quad i = 1, 2, \ldots, n.$$

Step 5. Derive new data set

Order the eigenvalues from highest to lowest. The unit eigenvector corresponding to the largest eigenvalue is the first principal component. The unit eigenvector corresponding to the next highest eigenvalue is the second principal component, and so on.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{bmatrix},$$

where T in the superscript denotes the transpose.

iv) We form the following $n \times N$ matrix:

$$X = \begin{bmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \cdots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \cdots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \cdots & X_{nN} - \bar{X}_n \end{bmatrix}$$

v) Next compute the matrix:

$$X_{\text{new}} = FX.$$

Note that this is a $p \times N$ matrix. This gives us a dataset of N samples having p features.

Step 6. New dataset

The matrix X_{new} is the new dataset. Each row of this matrix represents the values of a feature. Since there are only p rows, the new dataset has only features.

Step 7. Conclusion

This is how the principal component analysis helps us in dimensional reduction of the dataset. Note that it is not possible to get back the original n-dimensional dataset from the new dataset.

Example:

Given the data in Table, reduce the dimension from 2 to 1 using the Principal Component Analysis (PCA) algorithm.

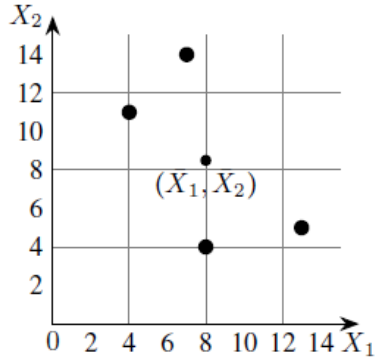
Feature	Example 1	Example 2	Example 3	Example 4
X1	4	8	13	7
X2	11	4	5	14

Step 1:

No. of features, n=2

No. of samples, N=4

The scatter plot of the given data points are



Step 2: Calculate the mean of X_1 and X_2

$$\bar{X}_1 = \frac{1}{4}(4+8+13+7) = 8,$$
$$\bar{X}_2 = \frac{1}{4}(11+4+5+14) = 8.5.$$

Step 3: Calculation of the Covariance Matrix

The Covariance's are calculated as follows:

$$\begin{aligned}
\text{Cov}(X_1, X_1) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)^2 \\
&= \tfrac{1}{3}\left((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right) \\
&= 14 \\
\text{Cov}(X_1, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)(X_{2k} - \bar{X}_2) \\
&= \tfrac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) \\
&\quad + (13-8)(5-8.5) + (7-8)(14-8.5) \\
&= -11 \\
\text{Cov}(X_2, X_1) &= \text{Cov}(X_1, X_2) \\
&= -11 \\
\text{Cov}(X_2, X_2) &= \frac{1}{N-1}\sum_{k=1}^{N}(X_{2k} - \bar{X}_2)^2 \\
&= \tfrac{1}{3}\left((11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2\right) \\
&= 23
\end{aligned}$$

The covariance matrix is,

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix} = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 4: Calculate the Eigen Values, Eigen Vectors & Normalized Eigen Vector of the Covariance Matrix

(i) Eigen values:

$\det(S - \lambda I) = 0$ where, det = determenant

$\det\left(\begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix}\right) = 0$ S = Covariance matrix, λ = Eigen values, I = Identity matrix

$(14-\lambda)(23-\lambda) - (-11)(-11) = 0$

$\lambda^2 - 37\lambda + 201 = 0$

To find out the roots of Equation, Formula is

$\frac{1}{2a}(-b \pm \sqrt{b^2 - 4ac})$

Here $a=1$, $b=-37$, $c=201$

$\frac{1}{2a}(-b \pm \sqrt{(37)^2 - 4(1)(201)})$

$\frac{1}{2a}(-b \pm \sqrt{1369 - 804})$

$\frac{1}{2a}(-b \pm \sqrt{565}$

$\frac{1}{2}(-37 \pm 23.7697)$

∴ the roots of the Equation are 30.3849, 6.6151

consider highest value as λ_1 and smallest as λ_2

∴ $\lambda_1 = 30.3849$, $\lambda_2 = 6.6151$

(ii) Eigen vector of λ_1

$(S - \lambda_1 I)U = 0$

$\begin{bmatrix} 14-\lambda_1 & -11 \\ -11 & 23-\lambda_1 \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = 0$

$\begin{bmatrix} (14-\lambda_1)U_1 & -11U_2 \\ -11U_1 & +(23-\lambda_1)U_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$(14-\lambda_1)U_1 - 11U_2 = 0 \longrightarrow (1)$

$-11U + (23-\lambda_1)U_2 = 0 \longrightarrow (2)$

Consider Equation (1)

$(14-\lambda_1)U_1 - 11U_2 = 0$

$(14-\lambda_1)U_1 = 11U_2$

$\frac{U_1}{11} = \frac{U_2}{14-\lambda_1} = t$

where $t=1$, $U_1 = 11$, $U_2 = 14-\lambda_1$

Eigen vector U_1 of λ_1 $= \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix} = \begin{bmatrix} 11 \\ 14-30.3848 \end{bmatrix} = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix}$

(iii) Normalized Eigen vector U_1 is

$e_1 = \begin{bmatrix} \frac{11}{\sqrt{11^2 + (-16.3849)^2}} \\ \frac{-16.3849}{\sqrt{11^2 + (-16.3849)^2}} \end{bmatrix} = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$

By carrying out similar Computation the unit Eigen vector e_2 is

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

Step 5: Deriving New Dataset

First PCA (PC1)	Example 1	Example 2	Example 3	Example 4
	P11	P12	P13	P14

$$P_{11} = e_1^T \begin{bmatrix} 4-8 \\ 11-8.5 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} -4 \\ 2.5 \end{bmatrix} = -4.3052$$

$$P_{12} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 8-8 \\ 4-8.5 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 0 \\ -4.5 \end{bmatrix} = 3.7361$$

$$P_{13} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 13-8 \\ 5-8.5 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 5 \\ -3.5 \end{bmatrix} = 5.6928$$

$$P_{14} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 7-8 \\ 14-8.5 \end{bmatrix}$$

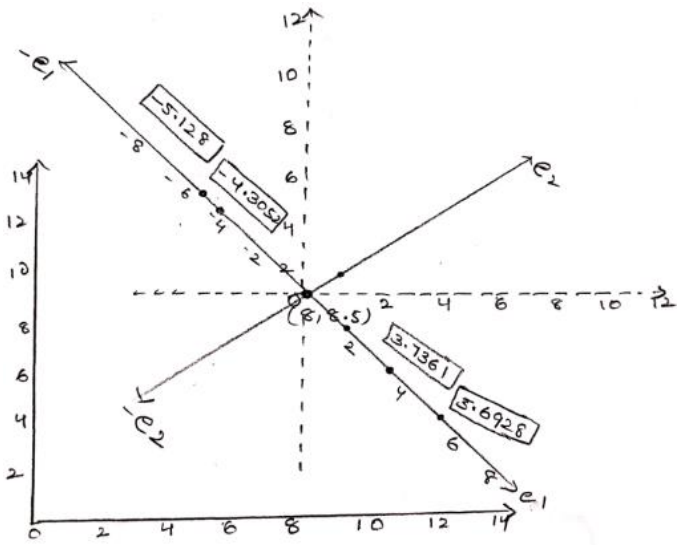
$$\begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} -1 \\ 5.5 \end{bmatrix} = -5.1238$$

First PCA (PC1)	Example 1	Example 2	Example 3	Example 4
	-4.3052	3.7361	5.6928	-5.1238

Step 6: The new Dataset is

Feature	**Example 1**	**Example 2**	**Example 3**	**Example 4**
X1	4	8	13	7
X2	11	4	5	14
First PCA (PC1)	-4.3052	3.7361	5.6928	-5.1238

Step 7: The scatter plot of the given data points are



USING SCIKIT-LEARN

Scikit-Learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

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:

```
import numpy as np
from sklearn.decomposition import PCA
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
pca = PCA(n_components=2)
pca.fit(X)

print(pca.explained_variance_ratio_)

print(pca.singular_values_)
```

```
[0.99244289 0.00755711]
[6.30061232 0.54980396]
```

RANDOMIZED PCA

If we set the svd_solver hyperparameter to "randomized", Scikit-Learn uses a stochastic 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:

```
pca = PCA(n_components=2, svd_solver='randomized')
pca.fit(X)

print(pca.explained_variance_ratio_)

print(pca.singular_values_)
```

```
[0.99244289 0.00755711]
[6.30061232 0.54980396]
```

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 we want to force Scikit-Learn to use full SVD, we can set the svd_solver hyperparameter to "full"

```
pca = PCA(n_components=2, svd_solver='full')
pca.fit(X)

print(pca.explained_variance_ratio_)

print(pca.singular_values_)
```

```
[0.99244289 0.00755711]
[6.30061232 0.54980396]
```

KERNEL PCA

Kernel PCA a mathematical technique that implicitly maps instances into a very high-dimensional space (called the feature space), enabling nonlinear classification and regression with Support Vector Machines.

A linear decision boundary in the high-dimensional feature space corresponds to a complex nonlinear decision boundary in the original space. It turns out that the same trick can be applied to PCA, making it possible to perform complex nonlinear projections for dimensionality reduction. This is called Kernel PCA. It is often good at preserving clusters of instances after projection, or sometimes even unrolling datasets that lie close to a twisted manifold.

For example, the following code uses Scikit-Learn's KernelPCA class to perform kPCA with an RBF kernel

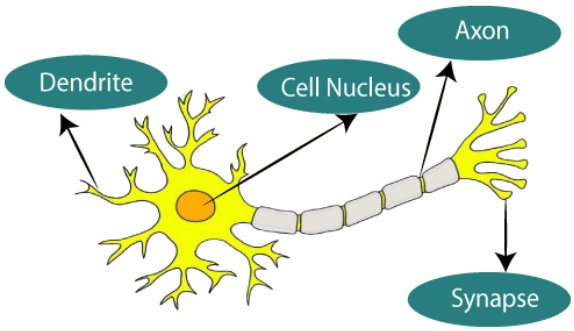
```python
from sklearn.decomposition import KernelPCA
rbf_pca = KernelPCA(n_components = 2, kernel="rbf", gamma=0.04)
X_reduced = rbf_pca.fit_transform(X)
X_reduced
```

```
array([[ 0.35376229, -0.21816667],
       [ 0.51755714, -0.05228295],
       [ 0.65529111,  0.27044962],
       [-0.35376229, -0.21816667],
       [-0.51755714, -0.05228295],
       [-0.65529111,  0.27044962]])
```

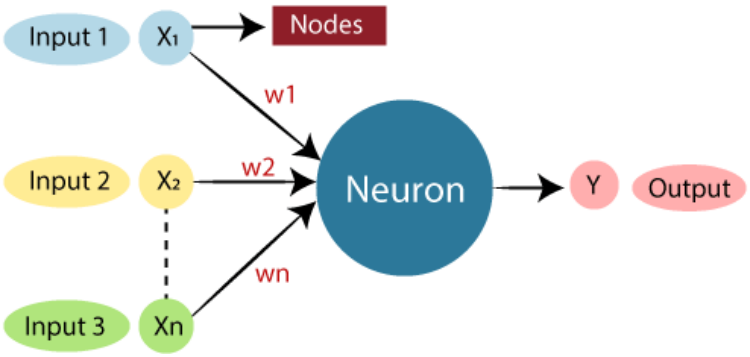
5. NEURAL NETWORKS AND DEEP LEARNING

INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS WITH KERAS

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. Below illustrates the typical diagram of Biological Neural Network.



The typical Artificial Neural Network is as follows

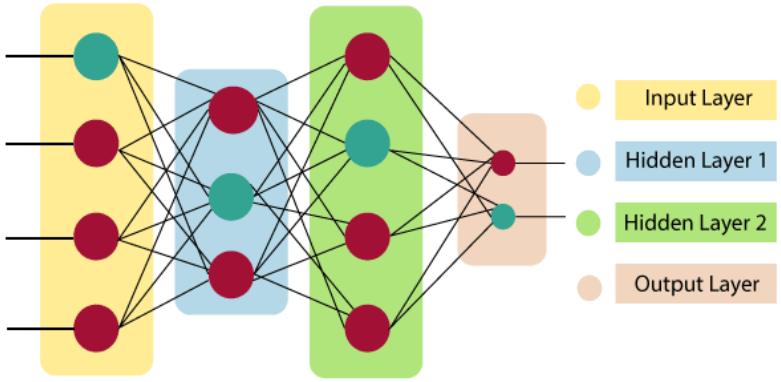


Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output. Relationship between Biological neural network and artificial neural network is as follows:

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

Architecture of an Artificial Neural Network:

Artificial Neural Network primarily consists of three layers:



1. Input Layer: It accepts inputs in several different formats
2. Hidden Layer: The layer present in-between input layer and output layer is the hidden layer. It performs all the calculations to find hidden features and patterns.
3. 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.

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

Here, W_i is Weight, X_i is input value & b is bias.

Types of Artificial Neural Network (ANN):

1. Feedforward Neural Network: The feedforward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided travels in a single direction. It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist. So the feedforward neural network has a front propagated wave only and usually does not have backpropagation.
2. Recurrent Neural Network: The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer. The first layer in the RNN is quite similar to the feed-forward neural network and the recurrent neural network starts once the output of the first layer is computed. After this layer, each unit will remember some information from the previous step so that it can act as a memory cell in performing computations.

3. Convolutional Neural Network: A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layers that use a convolution operation on the input and then pass the result obtained in the form of output to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.
4. Modular Neural Network: A Modular Neural Network contains a collection of different neural networks that work independently towards obtaining the output with no interaction between them. Each of the different neural networks performs a different sub-task by obtaining unique inputs compared to other networks. The advantage of this modular neural network is that it breaks down a large and complex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.
5. Radial basis function Neural Network: Radial basis functions are those functions that consider the distance of a point concerning the center. RBF functions have two layers. In the first layer, the input is mapped into all the Radial basis functions in the hidden layer and then the output layer computes the output in the next step. Radial basis function nets are normally used to model the data that represents any underlying trend or function.

Applications of Artificial Neural Networks (ANN):

- Social Media
- Marketing and Sales
- Healthcare
- Personal Assistants

Advantages of Artificial Neural Network (ANN):

- Parallel processing capability
- Storing data on the entire network
- Capability to work with incomplete knowledge
- Having a memory distribution

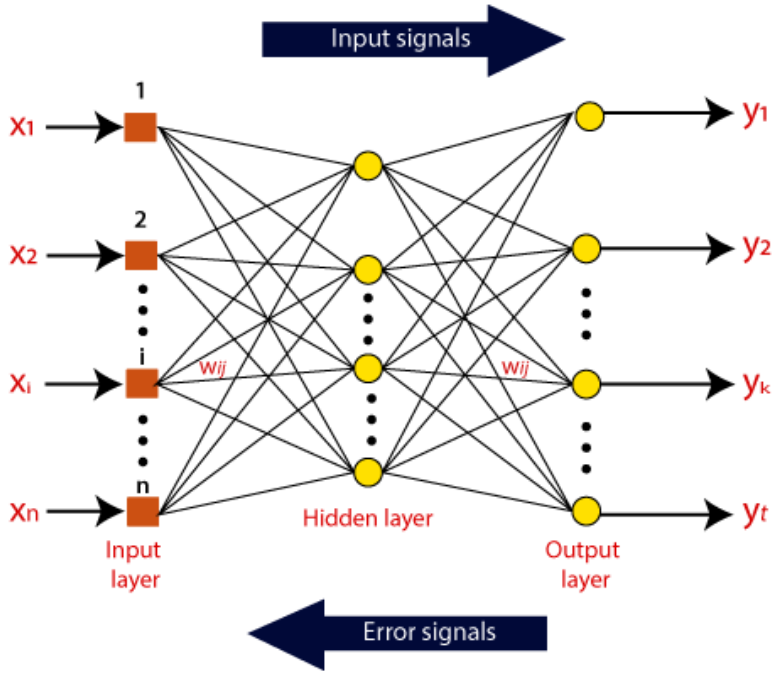
Disadvantages of Artificial Neural Network (ANN):

- Assurance of proper network structure
- Unrecognized behaviour of the network
- Hardware dependence

- Difficulty of showing the issue to the network

<u>Working of Artificial Neural Networks</u> (ANN):

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of

functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions.

Keras:

Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, or Cognitive Toolkit (CNTK) etc., for creating deep learning models.

Features:

- Consistent, simple and extensible API
- Minimal structure - easy to achieve the result without any frills
- It supports multiple platforms and backend
- It is user friendly framework which runs on both CPU and GPU
- Highly scalability of computation

Benefits:

- Larger community support
- Easy to test
- Keras neural networks are written in Python which makes things simpler
- Keras supports both convolution and recurrent networks
- Deep learning models are discrete components, so that, we can combine into many ways

IMPLEMENTING MLPS WITH KERAS

Multi-Layer Perceptrons (MLPs) are artificial neural networks that are widely used for classification and regression tasks. Keras is a high-level neural network API written in Python, which makes it easy to build and train deep learning models, including MLPs. Here are the steps to build and train an MLP using Keras:

1. Import the required libraries:

```
import tensorflow as tf
from tensorflow import keras
```

2. Load the dataset:

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

3. Preprocess the data:

```
x_train = x_train.reshape((60000, 28 * 28))
x_train = x_train.astype('float32') / 255
x_test = x_test.reshape((10000, 28 * 28))
x_test = x_test.astype('float32') / 255

y_train = keras.utils.to_categorical(y_train)
y_test = keras.utils.to_categorical(y_test)
```

4. Define the model architecture:

```
model = keras.Sequential([
   keras.layers.Dense(512, activation='relu', input_shape=(28 * 28,)),
   keras.layers.Dropout(0.5),
   keras.layers.Dense(10, activation='softmax')
])
```

5. Compile the model:

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

6. Train the model:

```
history = model.fit(x_train, y_train, epochs=10, batch_size=128, validation_data=(x_test, y_test))
```

7. Evaluate the model on the test set:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

The above creates a simple MLP with two dense layers and a dropout layer. The first dense layer has 512 units and uses the ReLU activation function. The output layer has 10 units (one for each digit) and uses the softmax activation function. The model is compiled using the RMSprop optimizer and the categorical cross-entropy loss function. Finally, the model is trained for 10 epochs using a batch size of 128 and is evaluated on the test set.

INSTALLING TENSORFLOW 2

TensorFlow 2:

- TensorFlow 2 is a popular open-source machine learning library developed by Google. It is a major update to the original TensorFlow library, and it introduces many new features and improvements.
- TensorFlow 2 is designed to be more user-friendly and accessible than its predecessor. It simplifies the process of building, training, and deploying machine learning models. It also includes many high-level APIs that make it easier to use machine learning techniques, such as deep neural networks.
- One of the key features of TensorFlow 2 is its support for eager execution, which allows us to run operations immediately as they are called, rather than building up a graph of operations to be executed later. This makes it easier to debug and experiment with models, and it also enables a more natural coding style.
- TensorFlow 2 also includes support for Keras, a high-level API for building and training neural networks. This integration allows us to take advantage of Keras's simplicity and ease of use, while still benefiting from TensorFlow's powerful low-level capabilities.

The following are the steps for installing TensorFlow 2:

- First, check if the system has the necessary hardware and software requirements to install TensorFlow 2. We need a 64-bit operating system, Python 3.6-3.9, and a compatible graphics card if we plan to use GPU(Graphics Processing Unit) acceleration.

- Create a new virtual environment to install TensorFlow 2. This step is optional, but it's a good practice to keep the TensorFlow installation isolated from other Python packages on thes ystem. We can create a virtual environment using the following command in terminal or command prompt:

  python3 -m venv myenv

  This creates a new virtual environment named myenv.

- Activate the virtual environment using the following command:

  source myenv/bin/activate

  If we're using Windows, we can activate the virtual environment using the following command instead:

  myenv\Scripts\activate

- Install TensorFlow 2 using pip by running the following command:

  pip install tensorflow

  If we have a compatible GPU and want to use GPU acceleration, we can install the GPU version of TensorFlow by running the following command instead:

  pip install tensorflow-gpu

- Verify that TensorFlow 2 has been installed correctly by importing it in a Python script or in a Python shell:

  import tensorflow as tf

  print(tf.__version__)

  This should print the version number of TensorFlow 2 that we installed.

Program to check whether TensorFlow 2 has been installed (or) not:

```
import tensorflow as tf
try:
    tf_version = tf.__version__
    print("TensorFlow version is:", tf_version)
except ImportError:
    print("TensorFlow is not installed.")
```

LOADING AND PREPROCESSING DATA WITH TENSORFLOW 2

Loading and preprocessing data is an important step in building machine learning models using TensorFlow 2.

Loading Data:

TensorFlow 2 supports various data sources, including NumPy arrays, pandas dataframes, and TFRecords files. The most commonly used data sources are NumPy arrays and pandas dataframes.

NumPy arrays: NumPy arrays are a popular data structure in Python for numerical computations. They can be used to represent tabular data, image data, and other types of data. To load data from a NumPy array, we can use the tf.data.Dataset.from_tensor_slices() method. Here's an example of how to load data from a NumPy array:

```
import tensorflow as tf
```

```
import numpy as np

# create a NumPy array with random data
data = np.random.random((100, 10))

# create a TensorFlow dataset from the NumPy array
dataset = tf.data.Dataset.from_tensor_slices(data)
```

Pandas dataframes: Pandas dataframes are another popular data structure in Python for data analysis. They can be used to represent tabular data with labeled rows and columns. To load data from a pandas dataframe, we can use the tf.data.Dataset.from_tensor_slices() method as well. Here's an example of how to load data from a pandas dataframe:

```
import tensorflow as tf
import pandas as pd

# create a pandas dataframe with random data
data = pd.DataFrame(np.random.random((100, 10)))

# create a TensorFlow dataset from the pandas dataframe
dataset = tf.data.Dataset.from_tensor_slices(data.values)
```

TFRecords files: TFRecords files are a binary format used to store large amounts of data efficiently. They are often used to store image data, audio data, and other types of data that can be represented as a sequence of bytes. To load data from a TFRecords file, we can use the tf.data.TFRecordDataset() method. Here's an example of how to load data from a TFRecords file:

```
import tensorflow as tf

# create a TensorFlow dataset from a TFRecords file
dataset = tf.data.TFRecordDataset('path/to/tfrecords/file')
```

Preprocessing Data:

After loading data, the next step is to preprocess it to prepare it for training. Preprocessing typically involves transforming the data into a format that can be easily fed into a machine learning model.

Mapping: The map() method can be used to apply a function to each element of a dataset. This is often used to transform the input data into a format that is suitable for a machine learning model. Here's an example of how to use the map() method to apply a function to each element of a dataset:

```
import tensorflow as tf

# create a TensorFlow dataset
dataset = tf.data.Dataset.from_tensor_slices([1, 2, 3, 4, 5])

# define a function to square each element
def square_fn(x):
    return x ** 2

# apply the function to each element of the dataset
dataset = dataset.map(square_fn)
```

Batch: The batch() method can be used to group the elements of a dataset into batches. This is often used to feed multiple inputs to a machine learning model at once. Here's an example of how to use the batch() method to group the elements of a dataset into batches:

```
import tensorflow as tf

# Create a dataset
dataset = tf.data.Dataset.range(10)

# Batch the dataset into groups of 3 elements
batched_dataset = dataset.batch(3)

# Print the batches
for batch in batched_dataset:
  print(batch)
```