MACHINE LEARNING

Artificial Intelligence:-

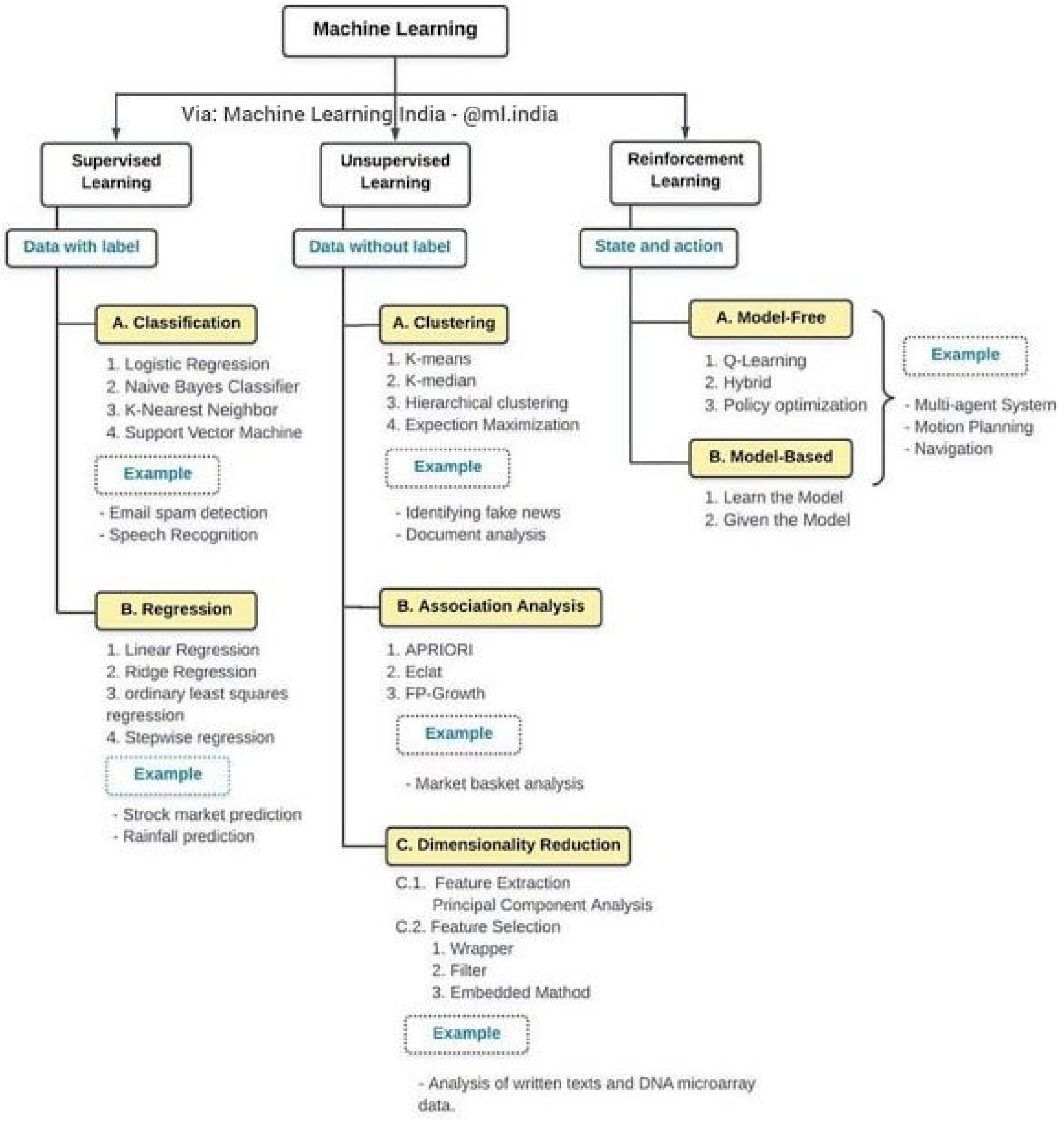
Any technology that mimics human intelligence and thinking

Machine Learning:-

The method of machines to improve , understand and learn from the data without explicitly coding is called machine learning

ML is a field of AI consisting of learning algorithms that −

- Improve their performance (P)
- At executing some task (T)
- Over time with experience (E)



Challenges in Machines Learning

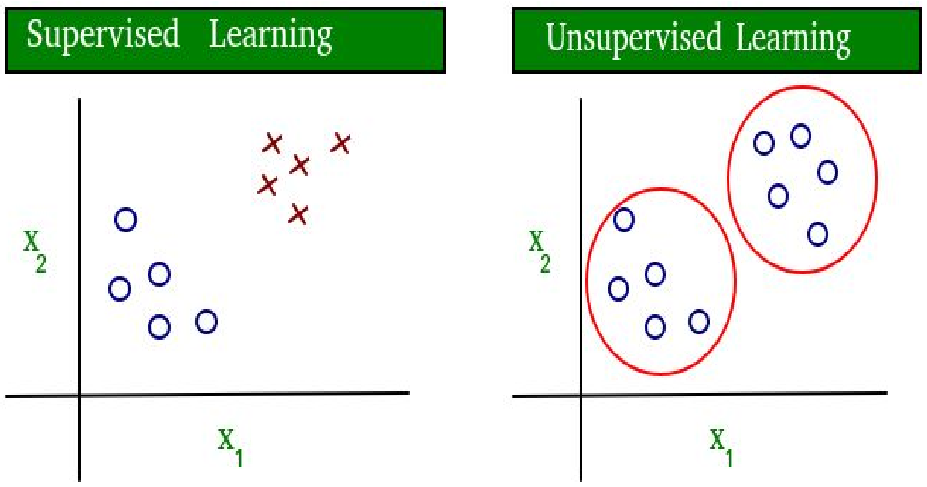
- Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
- Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
- Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
- No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
- Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
- Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
- Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Types of machine learning problems

1. On basis of the nature of the learning "signal" or "feedback" available to a learning system

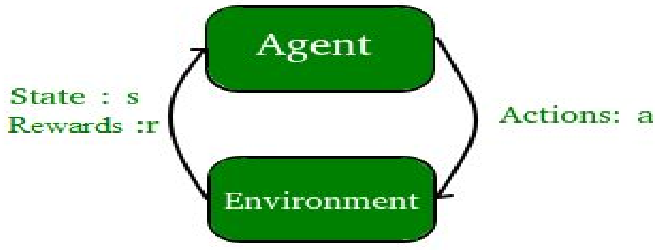
- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. The training process continues until the model achieves the desired level of accuracy on the training data. Some real-life examples are:
    - Image Classification: You train with images/labels. Then in the future you give a new image expecting that the computer will recognize the new object.
    - Market Prediction/Regression: You train the computer with historical market data and ask the computer to predict the new price in the future.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. It is used for clustering populations in different groups. Unsupervised learning can be a goal in itself (discovering hidden patterns in data).
    - Clustering: You ask the computer to separate similar data into clusters, this is essential in research and science.
    - High Dimension Visualization: Use the computer to help us visualize high dimensional data.
    - Generative Models: After a model captures the probability distribution of your input data, it will be able to generate more data. This can be very useful to make your classifier more robust.

A simple diagram which clears the concept of supervised and unsupervised learning is shown below:



As you can see clearly, the data in supervised learning is labelled, where as data in unsupervised learning is unlabelled.

- Semi-supervised learning: Problems where you have a large amount of input data and only some of the data is labeled, are called semi-supervised learning problems. These problems sit in between both supervised and unsupervised learning. For example, a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.



2. On the basis of "output" desired from a machine learned system

- Classification: Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

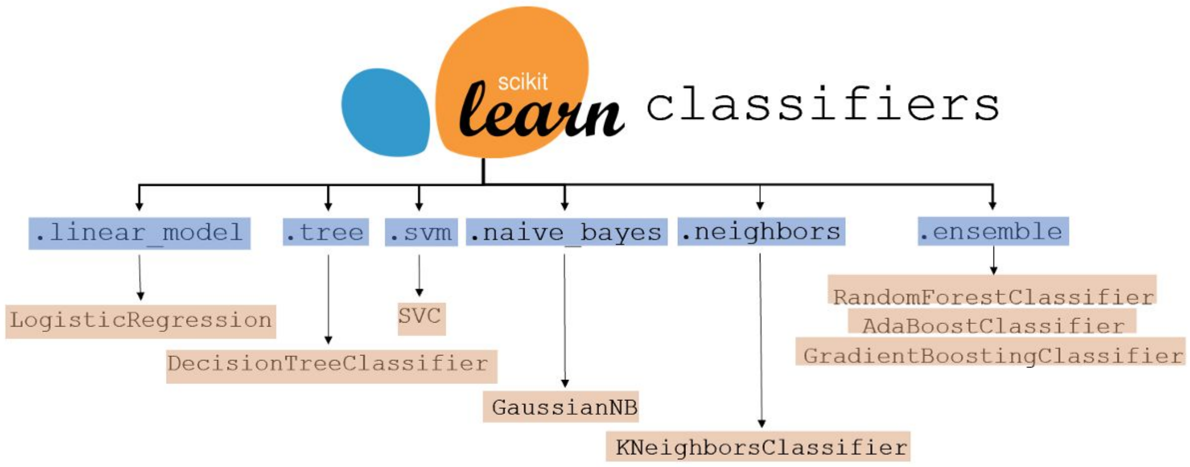
Types of Learners in Classification

Lazy Learners

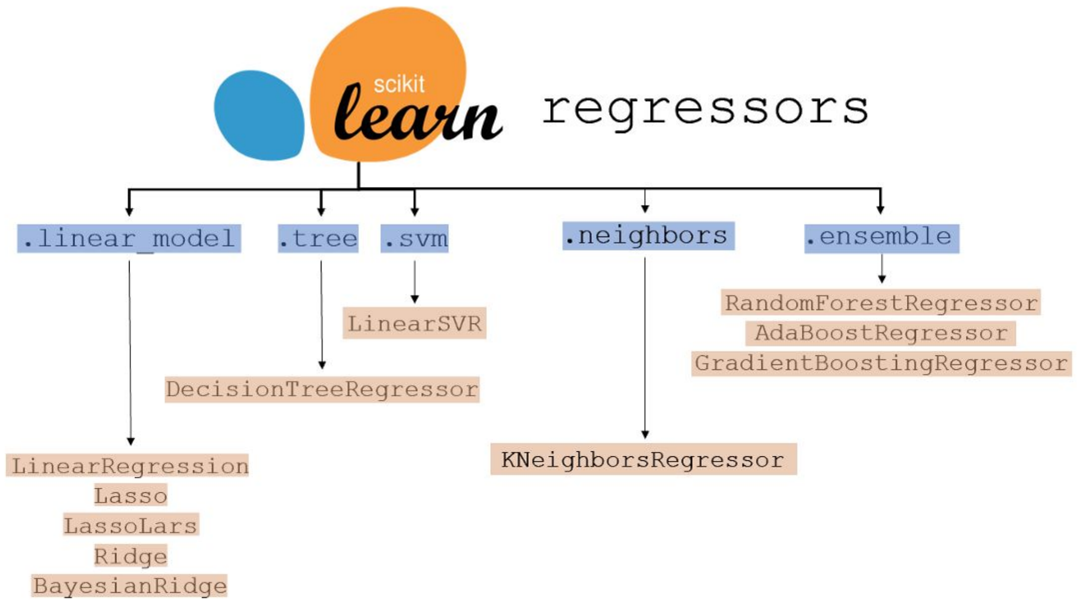
As the name suggests, such learners wait for the testing data to appear after storing the training data. Classification is done only after getting the testing data. They spend less time on training but more time on predicting. Examples of lazy learners are K-nearest neighbors and case-based reasoning.

Eager Learners

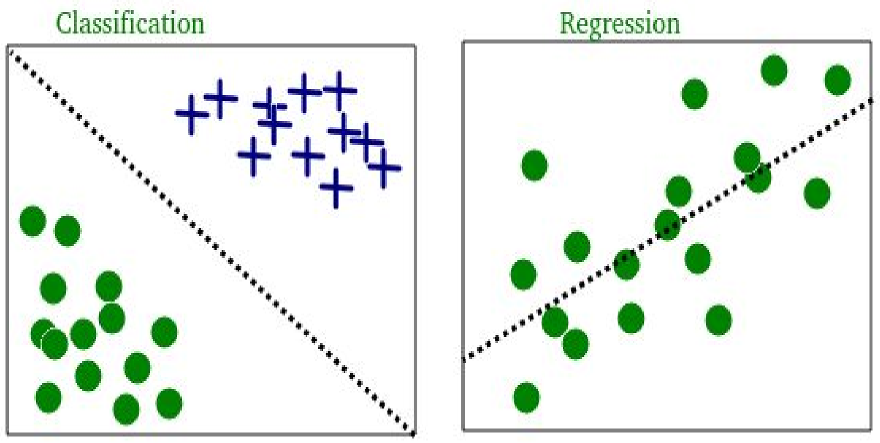
Eager learners construct classification models without waiting for the testing data to appear after storing the training data. They spend more time on training but less time on predicting. Examples of eager learners are Decision Trees, Naïve Bayes and Artificial Neural Networks (ANN).



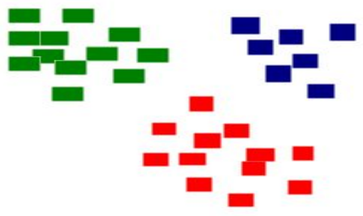
- Regression: It is also a supervised learning problem, but the outputs are continuous rather than discrete. For example, predicting the stock prices using historical data.



An example of classification and regression on two different datasets is shown below:



- Clustering: Here, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
  As you can see in the example below, the given dataset points have been divided into groups identifiable by the colors red, green and blue.



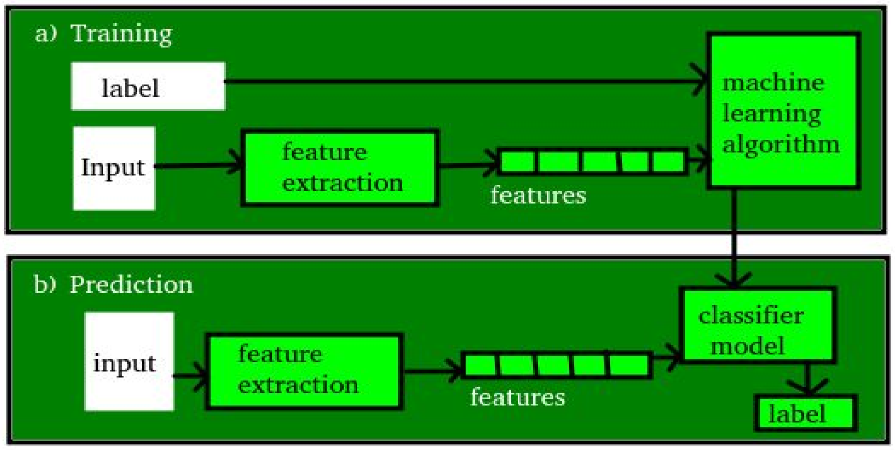
- Density estimation: The task is to find the distribution of inputs in some space.
- Dimensionality reduction: It simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

Terminologies of Machine Learning

- Model
  A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- Feature
  A feature is an individual measurable property of our data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
  Note: Choosing informative, discriminating and independent features is a crucial step for effective algorithms. We generally employ a feature extractor to extract the relevant features from the raw data.
- Target (Label)
  A target variable or label is the value to be predicted by our model. For the fruit example

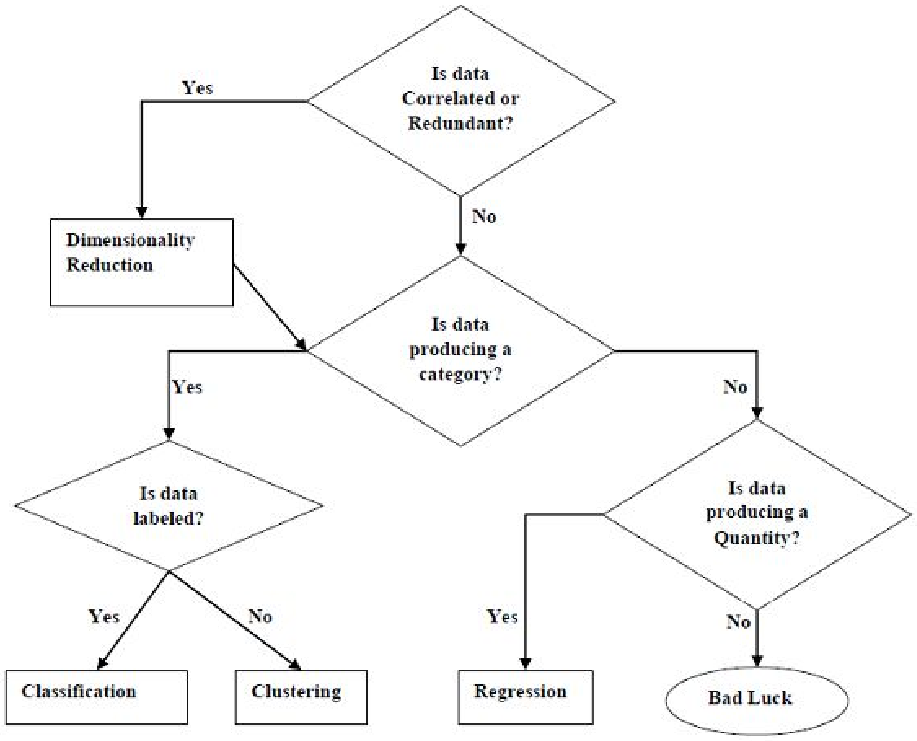
discussed in the features section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

- Training
  The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- Prediction
  Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).



Tasks Suited for Machine Learning

The following diagram shows what type of task is appropriate for various ML problems –



Based on learning ability

❖ Batch or Offline Learning

In many cases, we have end-to-end Machine Learning systems in which we need to train the model in one go by using whole available training data. Such a learning method or algorithm is called Batch or Offline learning. It is called Batch or Offline learning because it is a one-time procedure and the model will be trained with data in one single batch.

❖ Mini-batches or Online Learning

In these learning methods, the training data is supplied in multiple incremental batches, called mini-batches, to the algorithm.

Based on Generalization Approach

- Instance based Learning

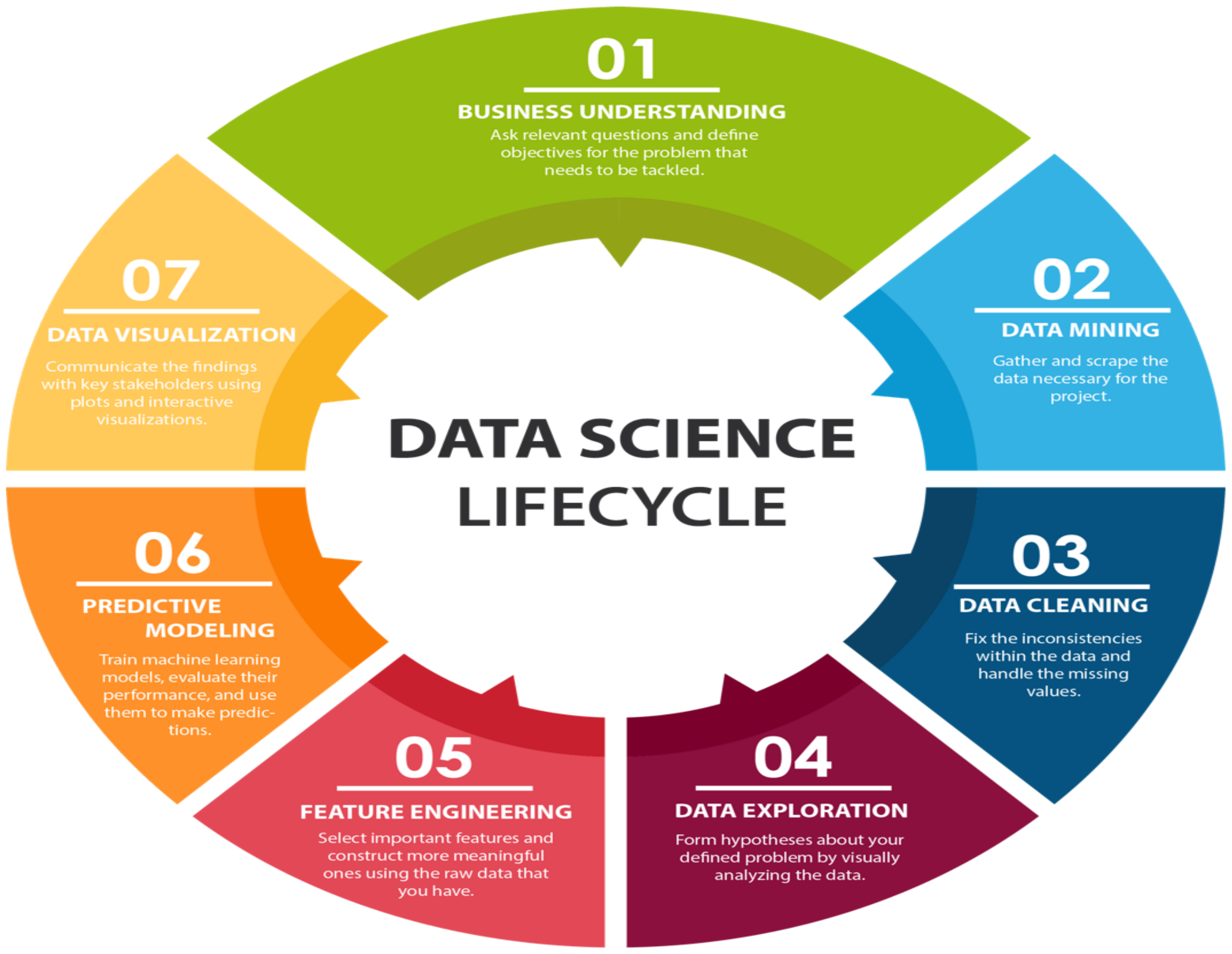
Instance based learning method is one of the useful methods that build the ML models by doing generalization based on the input data. It is opposite to the previously studied learning methods in the way that this kind of learning involves ML systems as well as methods that use the raw data points themselves to draw the outcomes for newer data samples without building an explicit model on training data.

In simple words, instance-based learning basically starts working by looking at the input data points and then using a similarity metric, it will generalize and predict the new data points.

- Model based Learning

In Model based learning methods, an iterative process takes place on the ML models that are built based on various model parameters, called hyperparameters and in which input data is used to extract the features. In this learning, hyperparameters are optimized based on various model validation techniques. That is why we can say that Model based learning methods use a more traditional ML approach towards generalization.

Data Science life cycle:-


DATA SCIENCE LIFECYCLE
01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.

Data Visualization Technique:-

Multivariate Plots:

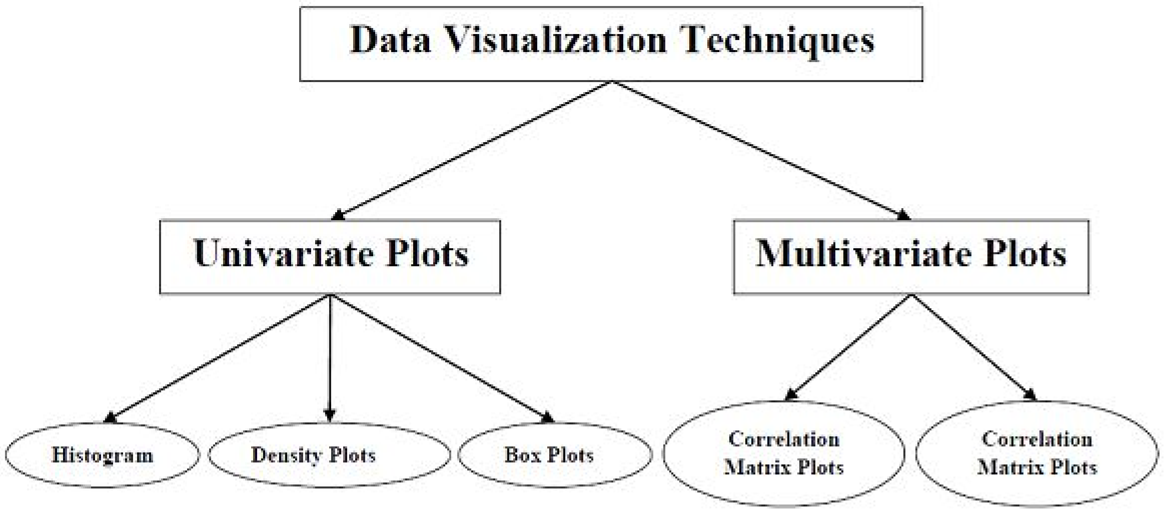
visualization is multivariable or “multivariate” visualization. With the help of multivariate visualization, we can understand interaction between multiple attributes of our dataset.

Ex: correlation plot and pair plot

Univariate Plots

The simplest type of visualization is single-variable or “univariate” visualization. With the help of univariate visualization, we can understand each attribute of our dataset independently.

Ex: box,histogram



Data Pre-processing Techniques

1. Scaling
2. Normalization
3. Binarization
4. Standardization
5. Data Labeling

Scaling

Most probably our dataset comprises the attributes with varying scale, but we cannot provide such data to ML algorithms hence it requires rescaling. Data rescaling makes sure that attributes are at the same scale. Generally, attributes are rescaled into the range of 0 and 1. ML algorithms like gradient descent and k-Nearest Neighbors require scaled data. We can rescale the data with the help of the **MinMaxScaler** class of scikit-learn Python library.

Normalization

Another useful data preprocessing technique is Normalization. This is used to rescale each row of data to have a length of 1. It is mainly useful in Sparse dataset where we have lots of zeros. We can rescale the data with the help of the Normalizer class of scikit-learn Python library.

Types of Normalization

In machine learning, there are two types of normalization preprocessing techniques as follows −

- **L1 Normalization**

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the absolute values will always be up to 1. It is also called **Least Absolute Deviations.**

- **L2 Normalization**

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1. It is also called **least squares.**

Binarization

As the name suggests, this is the technique with the help of which we can make our data binary. We can use a binary threshold for making our data binary. The values above that threshold value will be converted to 1 and below that threshold will be converted to 0. For example, if we choose threshold value = 0.5, then the dataset value above it will become 1 and below this will become 0. That is why we can call it binarizing the data or thresholding the data. This technique is useful when we have probabilities in our dataset and want to convert them into crisp values.We can binarize the data with the help of the **Binarizer class** of scikit-learn Python library.

Standardization

Another useful data preprocessing technique which is basically used to transform the data attributes with a Gaussian distribution. It differs the mean and SD (Standard Deviation) to a standard Gaussian distribution with a mean of 0 and a SD of 1. This technique is useful in ML algorithms like linear **regression, logistic regression** that assumes a Gaussian distribution in input dataset and produces better results with rescaled data. We can standardize the data (mean = 0 and SD =1) with the help of the **StandardScaler** class of scikit-learn Python library.

Data Labeling

We discussed the importance of good Data for ML algorithms as well as some techniques to pre-process the data before sending it to ML algorithms. One more aspect in this regard is data labeling. It is also very important to send the data to ML algorithms having proper labeling.

What is Label Encoding?

Most of the sklearn functions expect the data with number labels rather than word labels. Hence, we need to convert such labels into number labels. This process is called label encoding. We can perform label encoding of data with the help of **LabelEncoder()** function of scikit-learn Python library.

Feature Selection Techniques

1. Univariate selection
2. Recursive Feature Elimination
3. Principal Component Analysis (PCA)
4. Feature Importance

Univariate Selection

This feature selection technique is very useful in selecting those features, with the help of statistical testing, having the strongest relationship with the prediction variables. We can implement a univariate feature selection technique with the help of **SelectKBest0class** of scikit-learn Python library.

Recursive Feature Elimination

As the name suggests, RFE (Recursive feature elimination) feature selection technique removes the attributes recursively and builds the model with remaining attributes. We can implement RFE feature selection technique with the help of the **RFE** class of scikit-learn Python library.

Principal Component Analysis (PCA)

PCA, generally called data reduction technique, is a very useful feature selection technique as it uses linear algebra to transform the dataset into a compressed form. We can implement PCA feature selection techniques with the help of the **PCA** class of scikit-learn Python library. We can select the number of principal components in the output.

Feature Importance

As the name suggests, feature importance technique is used to choose the important features. It basically uses a trained supervised classifier to select features. We can implement this feature selection technique with the help of **ExtraTreeClassifier** class of scikit-learn Python library.

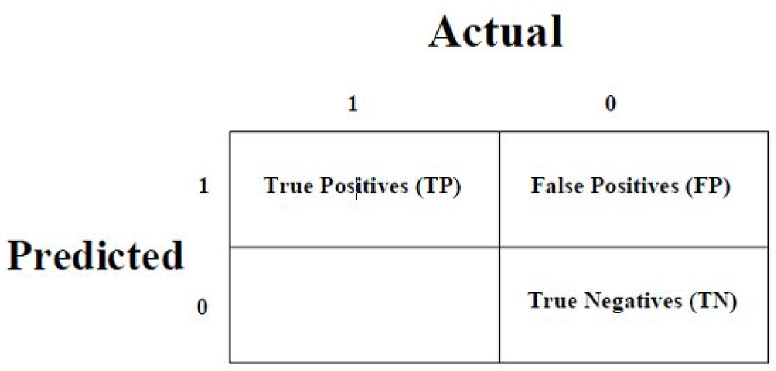
Classification Evaluation Metrics

The job is not done even if you have finished implementation of your Machine Learning application or model. We must have to find out how effective our model is? There can be different evaluation metrics, but we must choose it carefully because the choice of metrics influences how the performance of a machine learning algorithm is measured and compared.

1. Accuracy
2. Precision
3. Recall or sensitivity
4. specificity

Confusion Matrix

It is the easiest way to measure the performance of a classification problem where the output can be of two or more types of classes. A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)” as shown below −



The explanation of the terms associated with confusion matrix are as follows −

- True Positives (TP) − It is the case when both actual class & predicted class of data point is 1.
- True Negatives (TN) − It is the case when both actual class & predicted class of data point is 0.
- False Positives (FP) − It is the case when the actual class of data point is 0 & predicted class of data point is 1.
- False Negatives (FN) − It is the case when the actual class of data point is 1 & predicted class of data point is 0.

Accuracy
It may be defined as the number of correct predictions made by our ML model. We can easily calculate it by confusion matrix with the help formula −

Accuracy=TP+TN / TP+FP+FN+TN

Precision
Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

Precision=TP / TP+FP

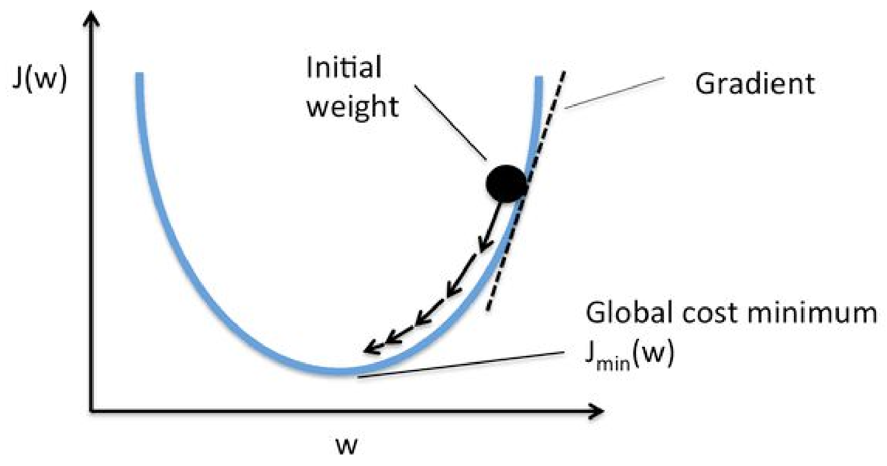
Recall or Sensitivity
Recall may be defined as the number of positives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −
Recall=TP /TP+FN0

Specificity
Specificity, in contrast to recall, may be defined as the number of negatives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −
Specificity=TN/TN+FP

Validate the function

- ***Cost Function(r8 dependent variable)***
- ***Gradient descent(r8 weights)***
- ***Feature scaling(normalization)***
- ***Generalization(over/underfitting)***
- ***Regularization(take care of overfitting)***

- **Cost function:**
  Sum of the square function(squaring because to increase the error value)
  Cost function=sqr(previous-predict)
- **Gradient descent:** (how to minimize the cost function)
  Graph between cost function and the prediction value
  If the slope is **Negative** we move the point to **Right**
  If the slope is **Positive** we move the point to **Left**
  j(w)= cost function
  w=predicted value
  alpha= coefficient in the equation



- **FEATURE SCALING**: normalization

  why 0-1 (to increase cpu power)
  - ➢ MIN-MAX NORMALIZATION(range is 0-1)

    Not changing the value variation just changing the range
  - ➢ STANDARDIZATION

    Mean and Standard deviation

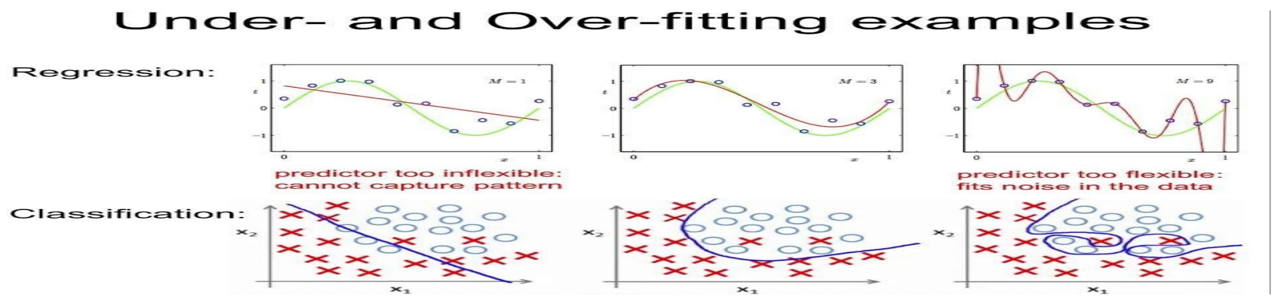
- **Generalization**

How effective the model learned the programm
- Overfitting:- model learned data too well not concept
  - ❖ Regularization
    - ➢ Cost function(mse) L1 or lasso regularization(lambda+weight)
    - ➢ Ridge regularization L2(lambda+weight^2) (more error is added)
    - ➢ Elastic net regularization(L1&L2)
- underfitting:-model not performing in training and test
- God fitting- fallowing and learning at a time

1. **Regularization(reduce overfitting)**

When c is higher value regression will Overfit

When c is lower value regression will Underfit



Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes.

Types of Logistic Regression

Based on those number of categories, Logistic regression can be divided into following types −

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variables can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

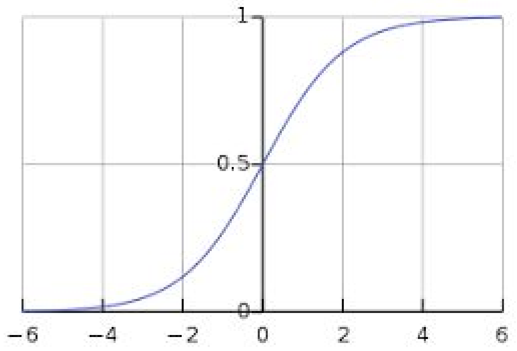
Ordinal

In such a kind of classification, dependent variables can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have scores like 0,1,2,3.

Logistic Regression Assumptions

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

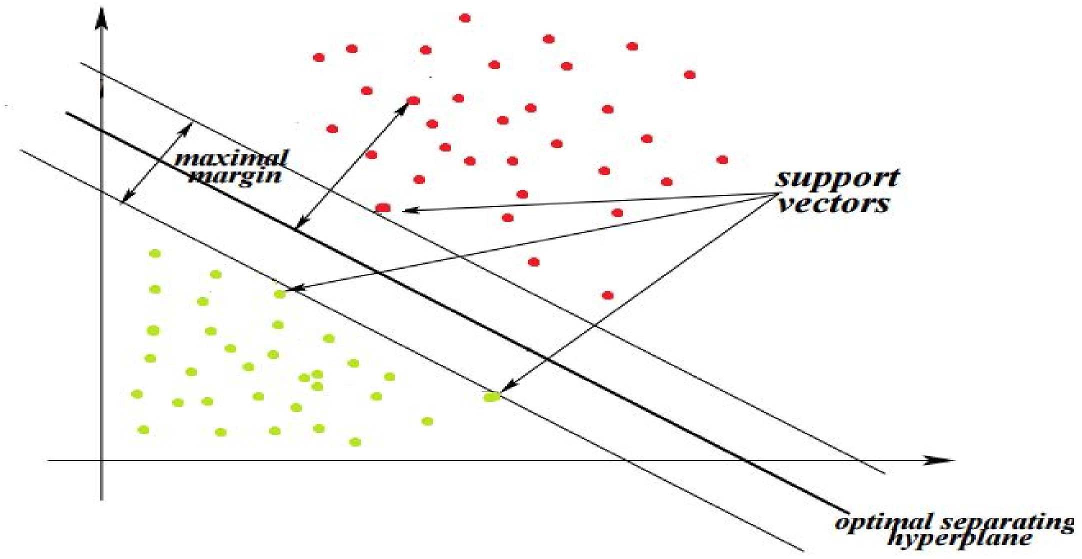
- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other .
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.



The classes can be divided into positive or negative. The output comes under the probability of positive class if it lies between 0 and 1. For our implementation, we are interpreting the output of the hypothesis function as positive if it is ≥0.5, otherwise negative.

SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).



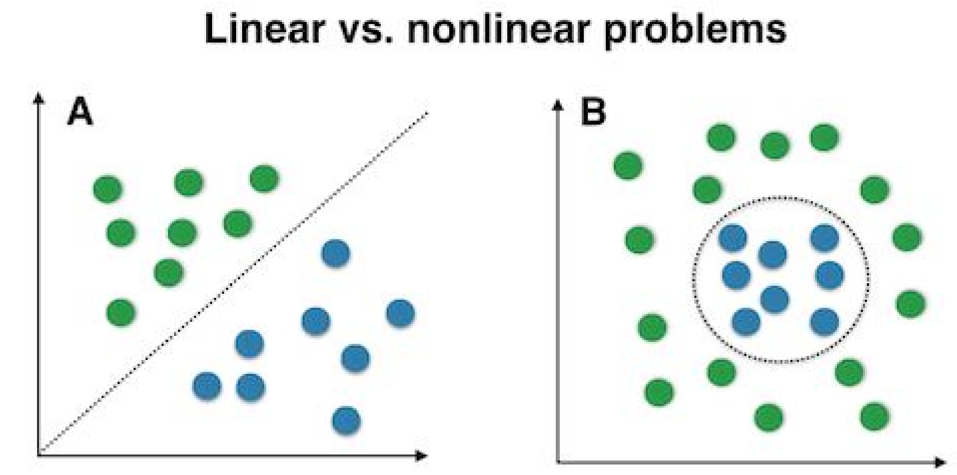
The followings are important concepts in SVM −

- Support Vectors − Data Points that are closest to the hyperplane is called support vectors. Separating lines will be defined with the help of these data points.
- Hyperplane − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- Margin − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in the best way.
- Then, it will choose the hyperplane that separates the classes correctly.

Hard margin :-No real time application
Soft margin:-This can accept outliers(noise) in the data
Non-Separable case(non-linear SVM)
Extra dimensions are create and then data can be separated and general linearly separable



SVM Kernels

In practice, an SVM algorithm is implemented with a kernel that transforms an input data space into the required form. SVM uses a technique called the **kernel t**rick in which the kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, the kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM −

- **Linear Kernel**

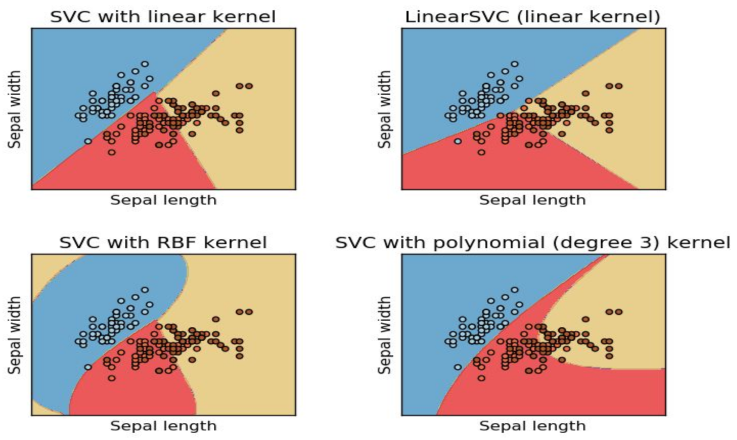
It can be used as a dot product between any two observations. The formula of linear kernel is as below −
k(x,xi) = sum(x*xi)
From the above formula, we can see that the product between two vectors say *x* & *xi* is the sum of the multiplication of each pair of input values.

- **Polynomial Kernel**

It is a more generalized form of linear kernel and distinguishes curved or nonlinear input space. Following is the formula for polynomial kernel −
K(x, xi) = 1 + sum(x * xi)^d
Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

- **Radial Basis Function (RBF) Kernel**

RBF kernel, mostly used in SVM classification, maps input space in infinite dimensional space.
Following formula explains it mathematically −
K(x,xi) = exp(-gamma * sum((x – xi^2))
Here, gamma ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of gamma is 0.1.
As we implemented SVM for linearly separable data, we can implement it in Python for the data that is not linearly separable. It can be done by using kernels.



Pros and Cons of SVM Classifiers

Pros

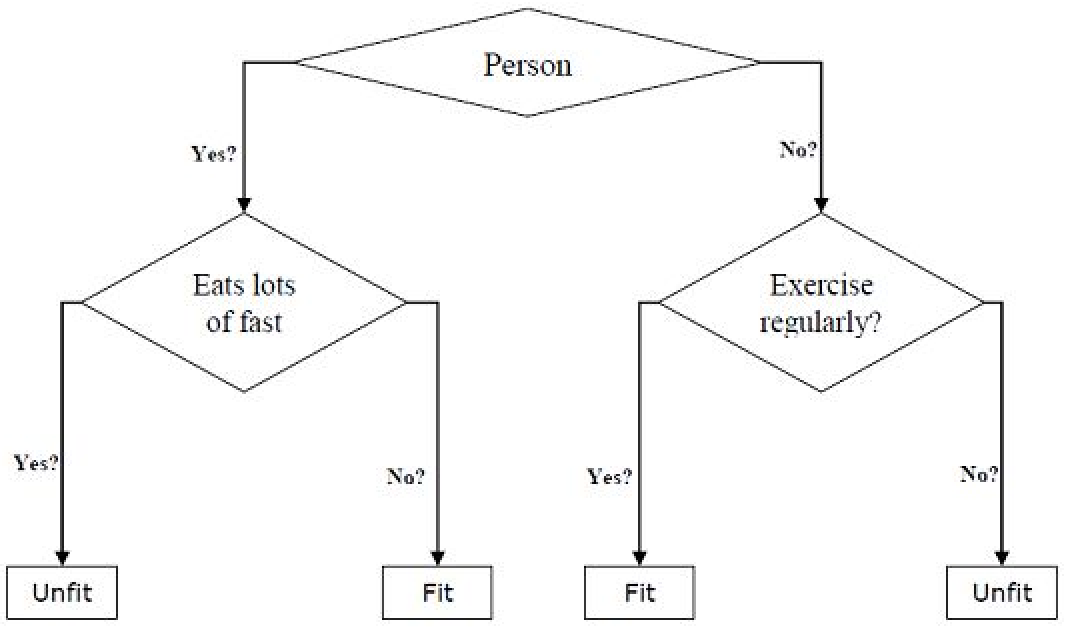
SVM classifiers offer great accuracy and work well with high dimensional space. SVM classifiers basically use a subset of training points hence in result uses very less memory.

Cons

They have high training time hence in practice not suitable for large datasets. Another disadvantage is that SVM classifiers do not work well with overlapping classes.

DECISION TREE

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are the most powerful algorithms that fall under the category of supervised algorithms.
They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below −



In the above decision tree, the questions are decision nodes and final outcomes are leaves. We have the following two types of decision trees −

- Classification decision trees − In this kind of decision tree, the decision variable is categorical. The above decision tree is an example of a classification decision tree.
- Regression decision trees − In this kind of decision trees, the decision variable is continuous.

Implementing Decision Tree Algorithm

- **Gini Index**

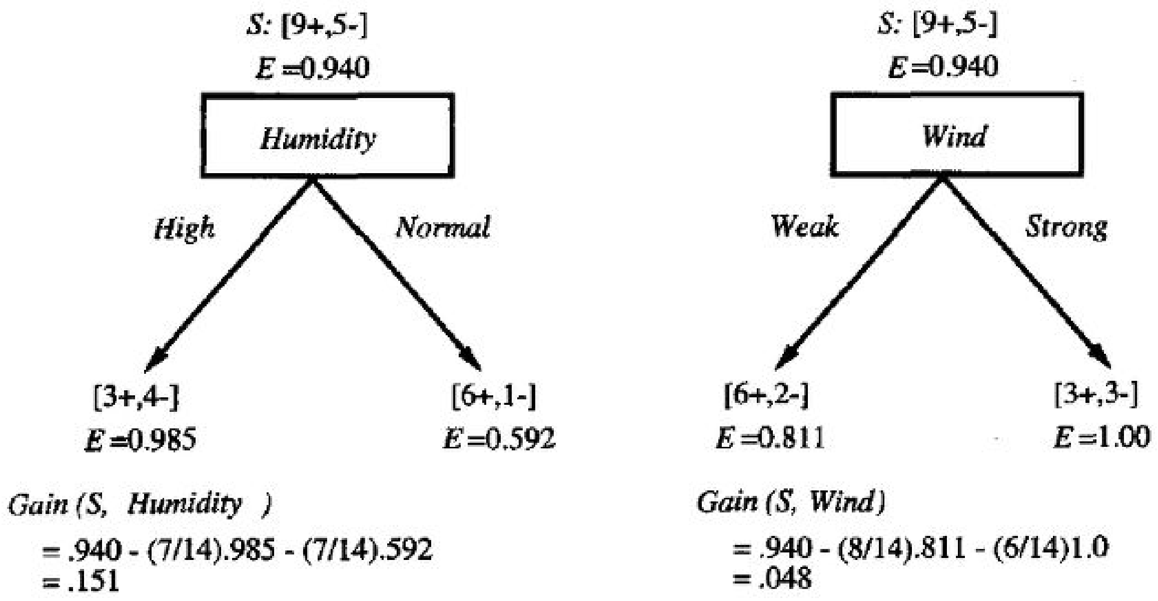
It is the name of the cost function that is used to evaluate the binary splits in the dataset and works with the categorical target variable “Success” or “Failure”.

Higher the value of the Gini index, higher the homogeneity. A perfect Gini index value is 0 and worst is 0.5 (for 2 class problems). Gini index for a split can be calculated with the help of following steps −

- First, calculate the Gini index for sub-nodes by using the formula p^2+q^2 , which is the sum of the square of probability for success and failure.
- Next, calculate the Gini index for split using the weighted Gini score of each node of that split.

Classification and Regression Tree (CART) algorithm uses Gini method to generate binary splits.

- **Entropy Gain**
  More computationally heavy due to log in the equation
  Entropy=-(p(red)logp(red)+p(blue)log(p(blue)))
- **Information Gain:-**
  Information Gain is calculated for a split by subtracting the weighted entropies of each branch from the original entropy. When training a Decision Tree using these metrics, the best split is chosen by maximizing Information Gain

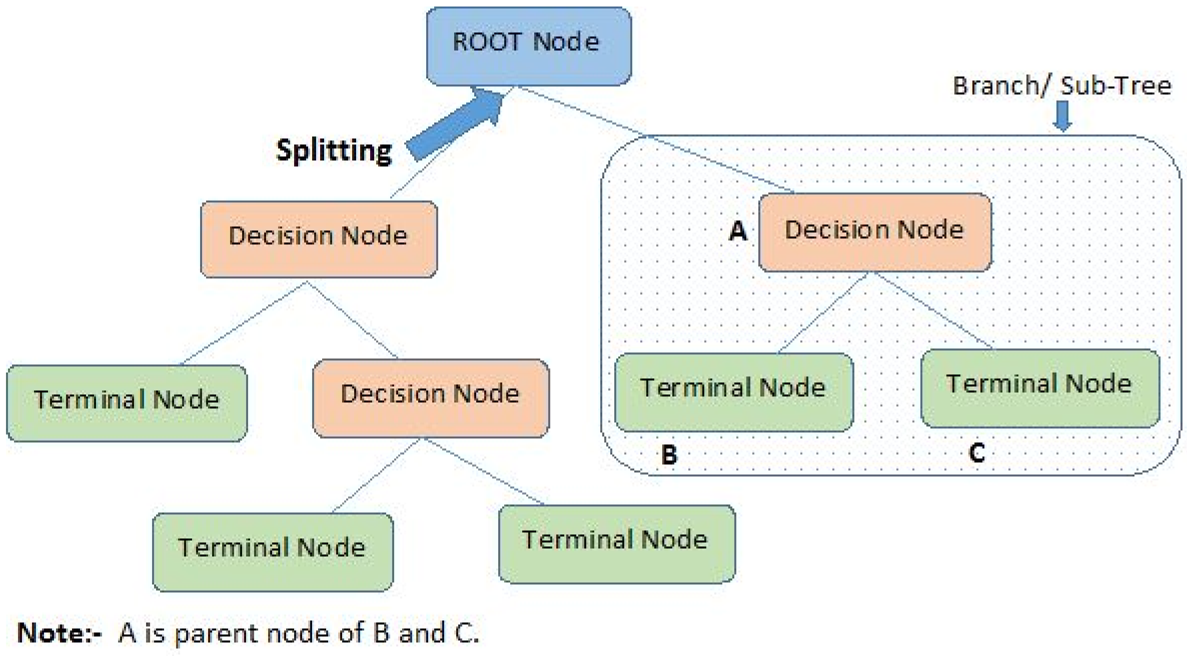


Split Creation
A split is basically including an attribute in the dataset and a value. We can create a split in dataset with the help of following three parts −

- Part1: Calculating Gini Score − We have just discussed this part in the previous section.
- Part2: Splitting a dataset − It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups - right and left, from the dataset, we can calculate the value of the split by using the Gini score calculated in the first part. Split value will decide in which group the attribute will reside.
- Part3: Evaluating all splits − Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

Building a Tree
As we know that a tree has root nodes and terminal nodes. After creating the root node, we can build the tree by following two parts −

- **Terminal node creation**
- **Recursive Splitting**



- **Part1: Terminal node creation**

While creating terminal nodes of decision trees, one important point is to decide when to stop growing trees or creating further terminal nodes. It can be done by using two criteria namely maximum tree depth and minimum node records as follows −

1. Maximum Tree Depth − As name suggests, this is the maximum number of the nodes in a tree after root node. We must stop adding terminal nodes once a tree reaches a maximum depth i.e. once a tree has a maximum number of terminal nodes.
2. Minimum Node Records − It may be defined as the minimum number of training patterns that a given node is responsible for. We must stop adding terminal nodes once a tree reaches these minimum node records or below this minimum.

- **Part2: Recursive Splitting**

As we understood about when to create terminal nodes, now we can start building our tree. Recursive splitting is a method to build the tree. In this method, once a node is created, we can create the child nodes (nodes added to an existing node) recursively on each group of data, generated by splitting the dataset, by calling the same function again and again.

Assumptions

The following are some of the assumptions we make while creating decision tree −

- While preparing decision trees, the training set is a root node.
- Decision tree classifier prefers the features values to be categorical. In case if you want to use continuous values then they must be done discretized prior to model building.
- Based on the attribute’s values, the records are recursively distributed.
- Statistical approach will be used to place attributes at any node position i.e.as root node or internal node.

Naive Bayes

Naïve Bayes Algorithm

Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, **the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class.** For example, a phone may be considered as smart if it has a touch screen, internet facility, good camera etc. Though all these features are dependent on each other, they contribute independently to the probability that the phone is a smart phone.

In Bayesian classification, the main interest is to find the posterior probabilities i.e. the probability of a label given some observed features, $P(L \mid features)$. With the help of Bayes theorem, we can express this in quantitative form as follows −

P(L|features)*P(features)=P(L)*P(features|L)

Here, $P(L \mid features)$ is the posterior probability of class.

$P(L)$ is the prior probability of class.

$P(features \mid L)$ is the likelihood which is the probability of predictor given class.

$P(features)$ is the prior probability of predictor.

We have the following three types of Naïve Bayes model under Scikit learn Python library −

Gaussian Naïve Bayes

It is the simplest Naïve Bayes classifier having the assumption that the data from each label is drawn from a simple Gaussian distribution.

Multinomial Naïve Bayes

Another useful Naïve Bayes classifier is Multinomial Naïve Bayes in which the features are assumed to be drawn from a simple Multinomial distribution. Such kinds of Naïve Bayes are most appropriate for the features that represent discrete counts.

Bernoulli Naïve Bayes

Another important model is Bernoulli Naïve Bayes in which features are assumed to be binary (0s and 1s). Text classification with the 'bag of words' model can be an application of Bernoulli Naïve Bayes.

Pros & Cons

Advantages

The followings are some pros of using Naïve Bayes classifiers −

- Naïve Bayes classification is easy to implement and fast.
- It will converge faster than discriminative models like logistic regression.
- It requires less training data.
- It is highly scalable in nature, or they scale linearly with the number of predictors and data points.
- It can make probabilistic predictions and can handle continuous as well as discrete data.

- Naïve Bayes classification algorithm can be used for binary as well as multi-class classification problems both.

Cons

The followings are some cons of using Naïve Bayes classifiers −

- One of the most important cons of Naïve Bayes classification is its strong feature independence because in real life it is almost impossible to have a set of features which are completely independent of each other.
- Another issue with Naïve Bayes classification is its **'zero frequency'** which means that if a categorical variable has a category but is not being observed in a training data set, then Naïve Bayes model will assign a zero probability to it and it will be unable to make a prediction.

Types of ML Regression Algorithms

The most useful and popular ML regression algorithm is Linear regression algorithm which further divided into two types namely −

- Simple Linear Regression algorithm
- Multiple Linear Regression algorithm.

We will discuss it and implement it in Python in the next chapter.

Simple Linear Regression (SLR)

It is the most basic version of linear regression which predicts a response using a single feature. The assumption in SLR is that the two variables are linearly related.

Multiple Linear Regression (MLR)

It is the extension of simple linear regression that predicts a response using two or more features

Assumptions

The following are some assumptions about dataset that is made by Linear Regression model −

- Multicollinearity − Linear regression models assume that there is very little or no multicollinearity in the data. Basically, multicollinearity occurs when the independent variables or features have dependency in them.
- Auto-correlation − Another assumption Linear regression model assumes is that there is very little or no auto-correlation in the data. Basically, auto-correlation occurs when there is dependency between residual errors.
- Relationship between variables − Linear regression model assumes that the relationship between response and feature variables must be linear.

RANDOM FOREST

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest.

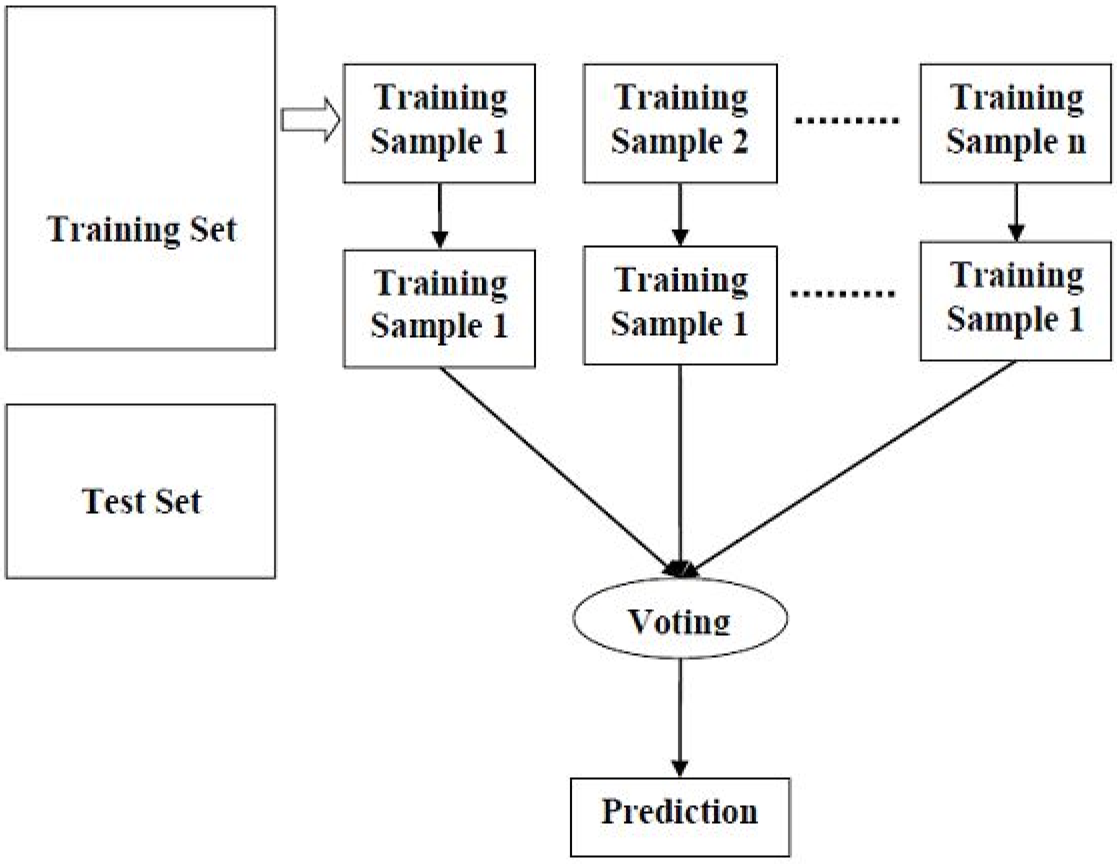
Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps −

- Step 1 − First, start with the selection of random samples from a given dataset.
- Step 2 − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- Step 3 − In this step, voting will be performed for every predicted result.
- Step 4 − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −



Pros and Cons of Random Forest

Pros

The following are the advantages of Random Forest algorithm −

- It overcomes the problem of overfitting by averaging or combining the results of different decision trees.
- Random forests work well for a larger range of data items than a single decision tree does.
- Random forest has less variance than a single decision tree.
- Random forests are very flexible and possess very high accuracy.
- Scaling of data does not require a random forest algorithm. It maintains good accuracy even after providing data without scaling.
- Scaling of data does not require a random forest algorithm. It maintains good accuracy even after providing data without scaling.

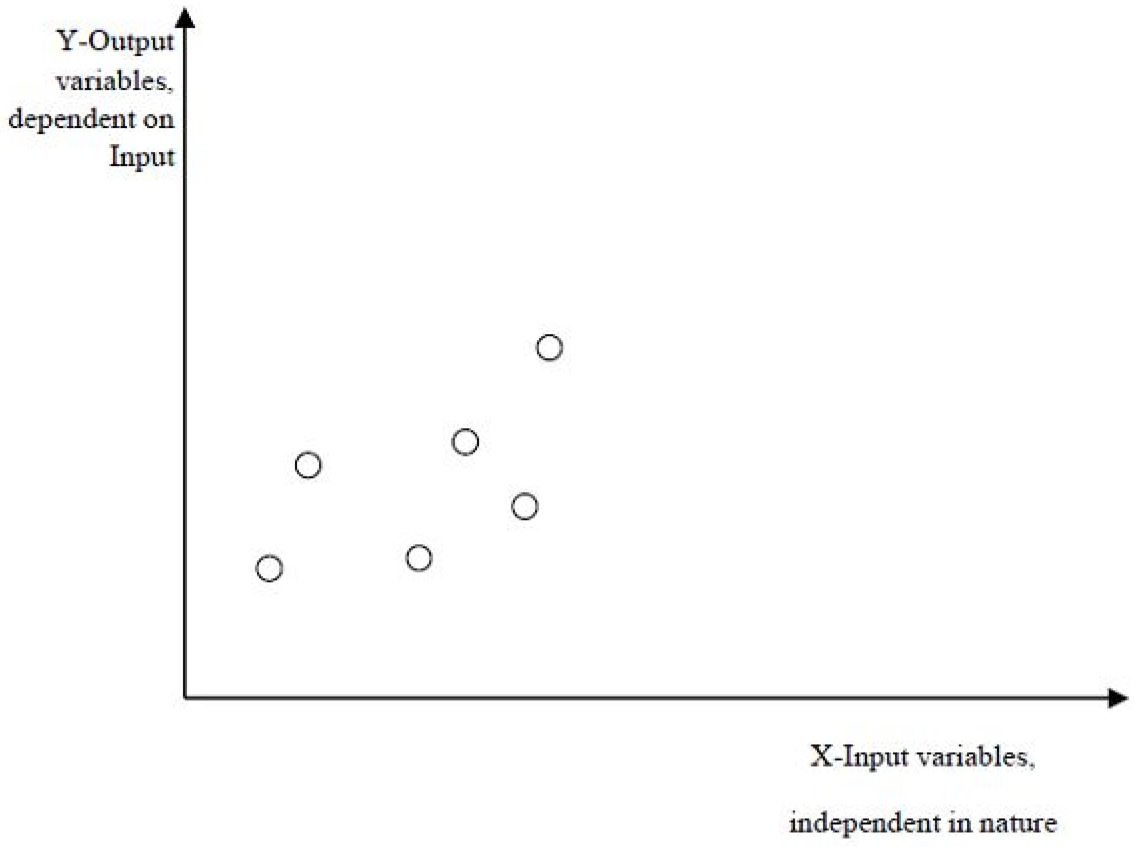
Cons

The following are the disadvantages of Random Forest algorithm −

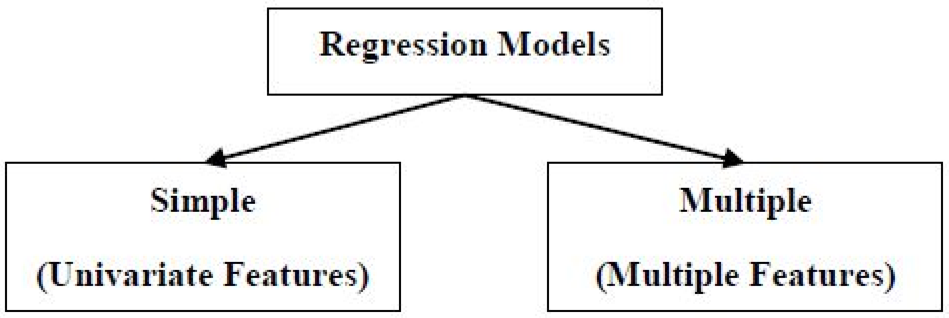
- Complexity is the main disadvantage of Random forest algorithms.
- Construction of Random forests are much harder and time-consuming than decision trees.
- More computational resources are required to implement the Random Forest algorithm.
- It is less intuitive in case when we have a large collection of decision trees .
- The prediction process using random forests is very time-consuming in comparison with other algorithms.

Regression

Regression is another important and broadly used statistical and machine learning tool. The key objective of regression-based tasks is to predict output labels or responses which are continuous numeric values, for the given input data. The output will be based on what the model has learned in the training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific associations between inputs and corresponding outputs.



Types of Regression Models



Regression models are of following two types −
Simple regression model − This is the most basic regression model in which predictions are formed from a single, univariate feature of the data.
Multiple regression model − As name implies, in this regression model the predictions are formed from multiple features of the data.

K-NEAREST NEIGHBORS (KNN) ALGORITHM

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of KNN Algorithm
K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

- Step 1 − For implementing any algorithm, we need a dataset. So during the first step of KNN, we must load the training as well as test data.
- Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.
- Step 3 − For each point in the test data do the following −
  3.1 − Calculate the distance between test data and each row of training data with the help of any of the methods namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
  3.2 − Now, based on the distance value, sort them in ascending order.
  3.3 − Next, it will choose the top K rows from the sorted array.
  3.4 − Now, it will assign a class to the test point based on the most frequent class of these rows.
- Step 4 − End

Pros and Cons of KNN
Pros

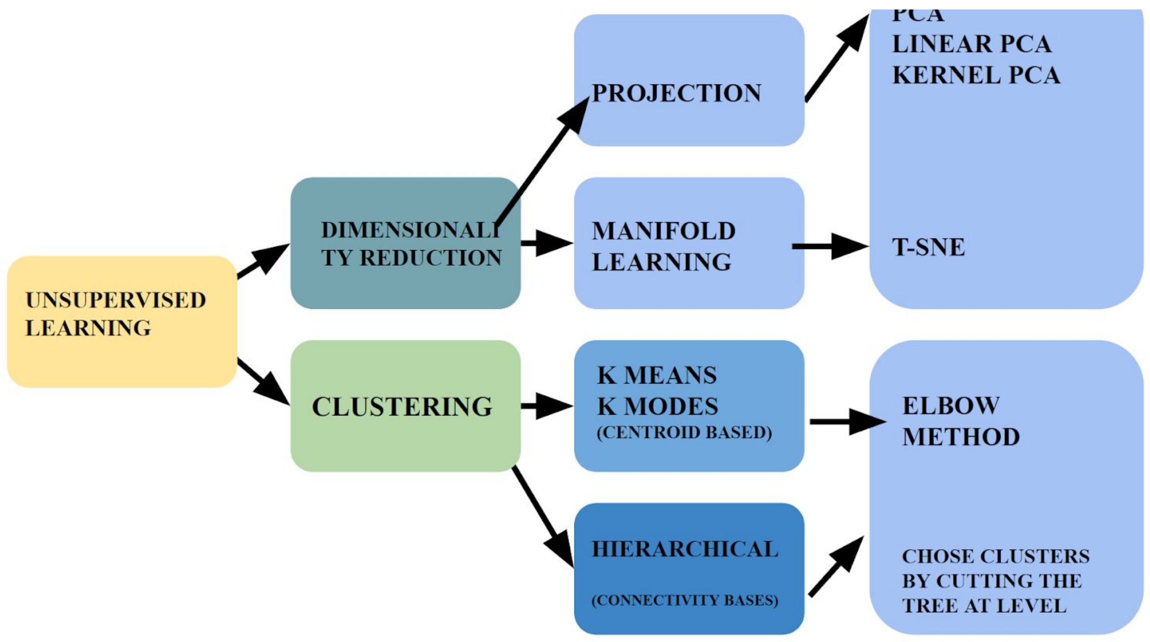
- It is a very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than KNN.

Cons

- It is computationally a bit expensive because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.

- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

UNSUPERVISED LEARNING



- DIMENSIONALITY REDUCTION
  - Projection Approach
    - Principal Component Analysis (PCA)
    - Incremental PCA
    - Kernelized PCA
  - Manifold Learning

    t-Distributed Stochastic Neighbor Embedding(manifold)

Curse of Dimensionality:-

Increase in number of feature make the problems more complex the values increase more exponentially

Covariance

In probability theory and statistics, covariance is a measure of the joint variability of two random variables. If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values, the covariance is positive

Covariance Matrix **:** Matrix consists of covariance between pairs of variables.

Correlation

is a statistical technique that can show whether and how strongly pairs of variables are related.

Eg : Work experience and Age of an employee are related. The relationship isn't perfect. As people with the same age can have different work experience .

Correlation coefficients (r) are used to measure how strong a relationship is between two variables.

It not only shows the kind of relation (in terms of direction) but also how strong the relationship is. Thus, we can say the correlation values have standardized notions, whereas the covariance values are not standardized and cannot be used to compare how strong or weak the relationship is because the magnitude has no direct significance. It can assume values from -1 to +1.

Eigenvectors and Eigenvalues:-

The vector that does not change the axis even when we did the mathematical operations and eigenvalue is the stretch of the vector caused due to addition multiplication and so on

The eigenvector with the highest eigenvalue is therefore the **principal component.**

PRINCIPAL COMPONENT ANALYSIS:-

- Principal components are eigenvectors of a covariance matrix, hence they are orthogonal
- The algorithm proceeds by finding the direction of maximum variance (Component 1)
- Then it finds the direction that contains the most information while being orthogonal to the first component.

Number of dimensions in PCA:-

1. Kaiser's Stopping Rule(Choose all components whose eigenvalues are greater than 1)
2. Scree Test(plot the values and take the dimension when becomes level)
3. Percentage of Cumulative Variance

Pros:

- Can deal with large datasets.
- No special assumptions on the data .

Cons :

- Non-linear structure is hard to model
- Too expensive for some task.

PCA VS T-SNE

PCA is a linear dimension reduction technique that seeks to maximize variance and preserves large pairwise distances

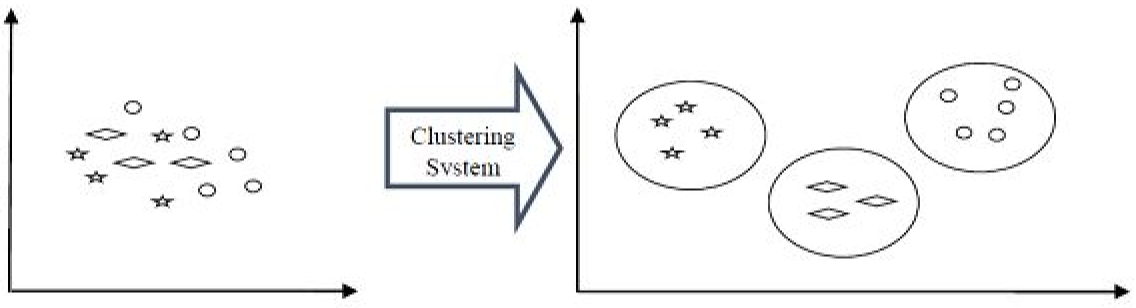
t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.

CLUSTERING

It is the task of dividing the dataset into groups, called clusters. These methods are used to find similarity as well as the relationship patterns among data samples and then cluster those samples into groups having similarity based on features.

Clustering is important because it determines the intrinsic grouping among the present unlabeled data. They basically make some assumptions about data points to constitute their similarity. Each assumption will construct different but equally valid clusters.

For example, below is the diagram which shows clustering system grouped together the similar kind of data in different clusters −



TYPES OF CLUSTERING

- **Hard Clustering**: In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each customer is assigned a probability to be in either of 10 clusters of the retail store.

CLUSTER FORMATION METHODS

It is not necessary that clusters will be formed in spherical form.

- Connectivity-based Model (hierarchical Clustering)
- Centroid-based Model (k-MEANS,k-MODES classification data)
- Distribution-based Model (assume data is distributed in gaussian distribution)
- Density-based Model(connects areas of high example density into clusters)

Density-based

In these methods, the clusters are formed as the dense region. The advantage of these methods is that they have good accuracy as well as good ability to merge two clusters. Ex. Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Ordering Points to identify Clustering structure (OPTICS) etc.

Hierarchical-based(connectivity based)

In these methods, the clusters are formed as a tree type structure based on the hierarchy. They have two categories namely, **Agglomerative** (Bottom up approach) and **Divisive** (Top down approach). Ex. Clustering using Representatives (CURE), Balanced iterative Reducing Clustering using Hierarchies (BIRCH) etc.

Partitioning

In these methods, the clusters are formed by portioning the objects into k clusters. Number of clusters will be equal to the number of partitions. Ex. **K-means**, Clustering Large Applications based upon randomized Search (CLARANS).

Grid

In these methods, the clusters are formed as a grid like structure. The advantage of these methods is that all the clustering operations done on these grids are fast and independent of the number of data objects. Ex. Statistical Information Grid (STING), Clustering in Quest (CLIQUE).

MEASURING CLUSTERING PERFORMANCE

1. Silhouette Index
2. Dunn Index
3. DB Index
4. CH Index
5. I- Index
6. XB or Xie Beni Index

CLUSTER VALIDATION

Internal cluster validation : The clustering result is evaluated based on the data clustered itself (internal information) without reference to external information.

- Single Linkage

In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster.

- Complete Linkage

In complete linkage hierarchical clustering, the distance between two clusters is defined as the longest distance between two points in each cluster.

- Average Linkage

In average linkage hierarchical clustering, the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.

Intra-cluster distance *D(a)* of a cluster a can be –

- **Complete diameter linkage distance:** Distance between two farthest objects belonging to cluster a.
- **Average diameter linkage distance:** Average distance between all the objects belonging to cluster a.
- **Centroid diameter linkage distance:** Twice the average distance between all the objects and the centroid of the cluster

External cluster validation : Clustering results are evaluated based on some externally known result, such as externally provided class labels.

Relative cluster validation : The clustering results are evaluated by varying different parameters for the same algorithm (e.g. **changing the number of clusters).**

we are not that blessed because we deal with unlabeled data. But still we have some metrics that give the practitioner an insight about the happening of change in clusters depending on the algorithm.

Before we deep dive into such metrics, we must understand that these metrics only evaluate the comparative performance of models against **each other rather than measuring the validity of the model's prediction**.

Silhouette Index

Silhouette analysis used to check the quality of the clustering model by measuring the distance between the clusters. It basically provides us a way to assess the parameters like number of clusters with the help of Silhouette score. This score measures how close each point in one cluster is to points in the neighboring clusters.

Analysis of Silhouette Score

The range of Silhouette scores is [-1, 1]. Its analysis is as follows −

- +1 Score − Near +1 Silhouette score indicates that the sample is far away from its neighboring cluster.
- 0 Score − 0 Silhouette score indicates that the sample is on or very close to the decision boundary separating two neighboring clusters.
- -1 Score minus -1 Silhouette score indicates that the samples have been assigned to the wrong clusters.

Davies-Bouldin Index

DB index is another good metric to perform the analysis of clustering algorithms. With the help of DB index, we can understand the following points about clustering model −

- Whether the clusters are well-spaced from each other or not?
- How dense the clusters are?

Dunn Index

It works same as DB index but there are following points in which both differs −

- The Dunn index considers only the worst case i.e. the clusters that are close together while DB index considers dispersion and separation of all the clusters in the clustering model.
- Dunn index increases as the performance increases while DB index gets better when clusters are well-spaced and dense.

TYPES OF CLUSTERING ALGORITHMS

K-Means Algorithm

K-means clustering algorithm computes the **centroids** and iterates until we find the optimal centroid. It assumes that the number of clusters are already known. It is also called a **flat clustering algorithm.** The number of clusters identified from data by algorithm is represented by 'K' in K-means.

In this algorithm, the data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within the same cluster.

Working of K-Means Algorithm

We can understand the working of K-Means clustering algorithm with the help of following steps −

- Step 1 − First, we need to specify the number of clusters, K, need to be generated by this algorithm.

  Elbow method(selecting k):-

  The basic idea behind this method is that it plots the various values of cost with changing k. The lesser number of elements means closer to the centroid. So, the point where this inertia declines the most is the elbow point.

- Step 2 − Next, randomly select K data points and assign each data point to a cluster. In simple words, classify the data based on the number of data points.
- Step 3 − Now it will compute the cluster centroids.
- Step 4 − Next, keep iterating the following until we find optimal centroid which is the assignment of data points to the clusters that are not changing any more −

4.1 − First, the sum of squared distance between data points and centroids would be computed.
4.2 − Now, we have to assign each data point to the cluster that is closer than the other cluster (centroid).
4.3 − At last compute the centroids for the clusters by taking the average of all data points of that cluster.

K-means follows Expectation-Maximization approach to solve the problem. The Expectation-step is used for assigning the data points to the closest cluster and the Maximization-step is used for computing the centroid of each cluster.

While working with K-means algorithm we need to take care of the following things −

- While working with clustering algorithms including K-Means, it is recommended to **standardize** the data because such algorithms use distance-based measurement to determine the similarity between data points.
- Due to the iterative nature of K-Means and random initialization of centroids, K-Means may stick in a local optimum and may not converge to **global optimum**. That is why it is recommended to use different initializations of centroids.

Advantages and Disadvantages

Advantages

The following are some advantages of K-Means clustering algorithms −

- It is very easy to understand and implement.
- If we have a large number of variables then, K-means would be faster than Hierarchical clustering.
- On re-computation of centroids, an instance can change the cluster.
- Tighter clusters are formed with K-means as compared to Hierarchical clustering.

Disadvantages

The following are some disadvantages of K-Means clustering algorithms −

- It is a bit difficult to predict the number of clusters i.e. the value of k.
- Output is strongly impacted by initial inputs like number of clusters (value of k).

- Order of data will have a strong impact on the final output.
- It is very sensitive to rescaling. If we will rescale our data by means of normalization or standardization, then the output will completely change.final output.
- It is not good to do clustering jobs if the clusters have a complicated geometric shape.

MEAN-SHIFT ALGORITHM

As discussed earlier, it is another powerful clustering algorithm used in unsupervised learning. Unlike K-means clustering, it does not make any assumptions; hence it is a non-parametric algorithm.

Mean-shift algorithm basically assigns the data points to the clusters iteratively by shifting points towards the highest density of data points i.e. cluster centroid.

The difference between K-Means algorithm and Mean-Shift is that later one does not need to specify the number of clusters in advance because the number of clusters will be determined by the algorithm w.r.t data.

Working of Mean-Shift Algorithm

We can understand the working of Mean-Shift clustering algorithm with the help of following steps −

- Step 1 − First, start with the data points assigned to a cluster of their own.
- Step 2 − Next, this algorithm will compute the centroids.
- Step 3 − In this step, location of new centroids will be updated.
- Step 4 − Now, the process will be iterated and moved to the higher density region.
- Step 5 − At last, it will be stopped once the centroids reach a position from where it cannot move further.

Advantages and Disadvantages

Advantages

The following are some advantages of Mean-Shift clustering algorithm −

- It does not need to make any model assumption as in K-means or Gaussian mixture.
- It can also model the complex clusters which have a convex shape.
- It only needs one parameter named bandwidth which automatically determines the number of clusters.
- There is no issue of local minima as in K-means.
- No problem generated from outliers.

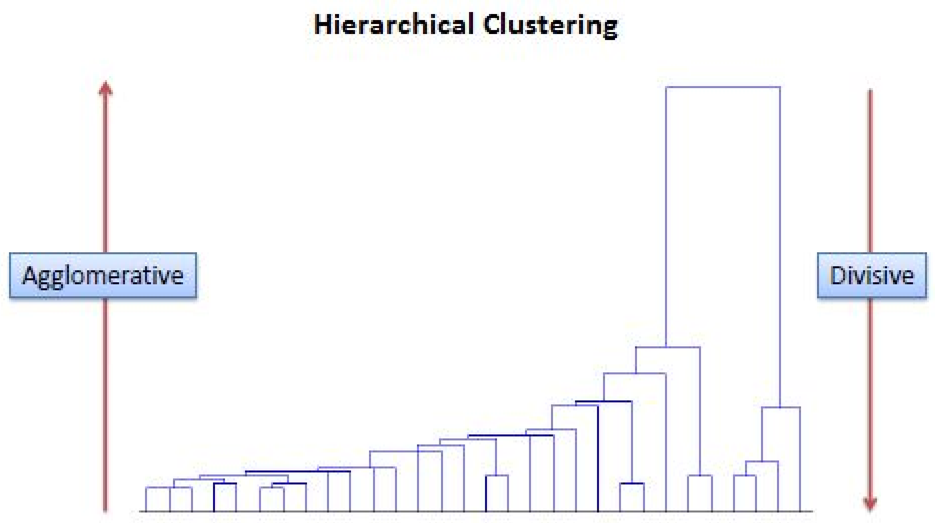
Disadvantages

The following are some disadvantages of Mean-Shift clustering algorithm −

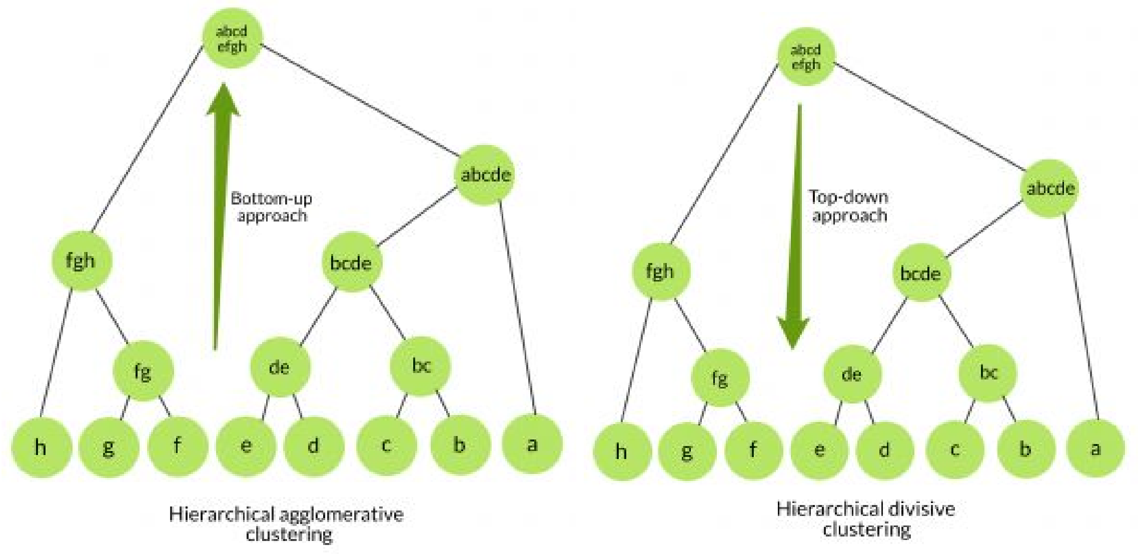
Mean-shift algorithm does not work well in case of high dimension, where the number of clusters changes abruptly.

- We do not have any direct control on the number of clusters but in some applications, we need a specific number of clusters.
- It cannot differentiate between meaningful and meaningless modes.

Hierarchical Clustering



Agglomerative hierarchical algorithms − In agglomerative hierarchical algorithms, each data point is treated as a single cluster and then successively merge or agglomerate (**bottom-up approach**) the pairs of clusters. The hierarchy of the clusters is represented as a dendrogram or tree structure.



Divisive hierarchical algorithms − On the other hand, in divisive hierarchical algorithms, all the data points are treated as one big cluster and the process of clustering involves dividing (**Top-down approach**) the one big cluster into various small clusters.

Agglomerative Hierarchical Clustering

We are going to explain the most used and important Hierarchical clustering i.e. agglomerative. The steps to perform the same is as follows −

- Step 1 − Treat each data point as a single cluster. Hence, we will be having, say K clusters at start. The number of data points will also be K at start.
- Step 2 − Now, in this step we need to form a big cluster by joining two closet datapoints. This will result in a total of K-1 clusters.
- Step 3 − Now, to form more clusters we need to join two closest clusters. This will result in a total of K-2 clusters.
- Step 4 − Now, to form one big cluster repeat the above three steps until K would become 0 i.e. no more data points left to join.
- Step 5 − At last, after making one single big cluster, dendrograms will be used to divide into multiple clusters depending upon the problem.

METRICS

There are various metrics which we can use to evaluate the performance of ML algorithms, classification as well as regression algorithms. We must carefully choose the metrics for evaluating ML performance because −

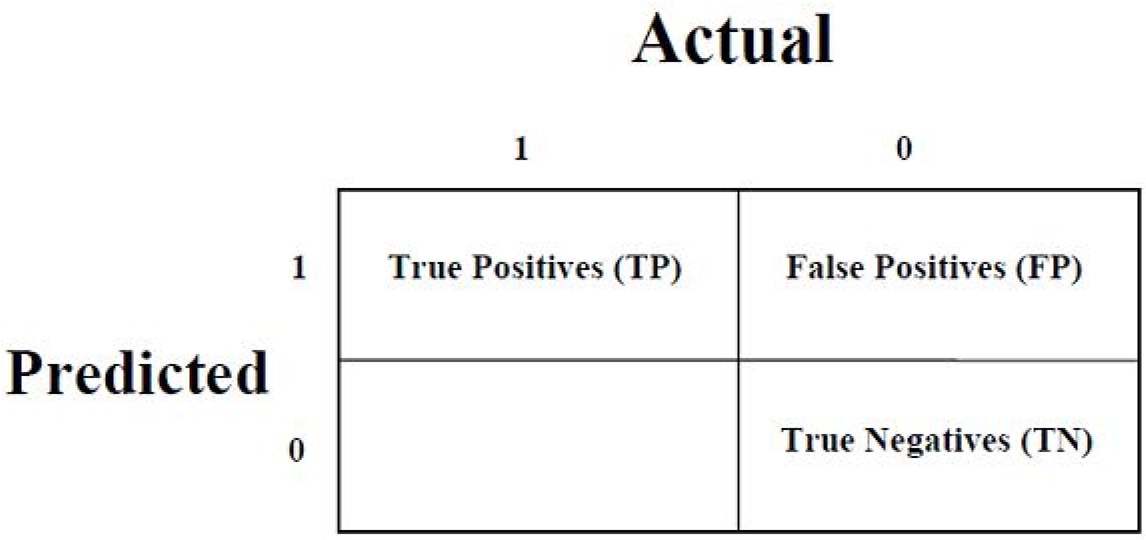
- How the performance of ML algorithms is measured and compared will be dependent entirely on the metric you choose.
- How you weigh the importance of various characteristics in the result will be influenced completely by the metric you choose.

Performance Metrics for Classification Problems

We have discussed classification and its algorithms in the previous chapters. Here, we are going to discuss various performance metrics that can be used to evaluate predictions for classification problems.

- **Confusion Matrix**

It is the easiest way to measure the performance of a classification problem where the output can be of two or more types of classes. A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have
“True Positives (TP)”,
“True Negatives (TN)”,
“False Positives (FP)”,
“False Negatives (FN)”



Explanation of the terms associated with confusion matrix are as follows −

- True Positives (TP) − It is the case when both actual class & predicted class of data point is 1.
- True Negatives (TN) − It is the case when both actual class & predicted class of data point is 0.
- False Positives (FP) − It is the case when the actual class of data point is 0 & predicted class of data point is 1.
- False Negatives (FN) − It is the case when the actual class of data point is 1 & predicted class of data point is 0.

We can use the confusion_matrix function of sklearn.metrics to compute the Confusion Matrix of our classification model.

- **Classification Accuracy**

It is the most common performance metric for classification algorithms. It may be defined as the number of correct predictions made as a ratio of all predictions made.
We can use the accuracy_score function of sklearn.metrics to compute the accuracy of our classification model.

- **Classification Report**

This report consists of the scores of Precisions, Recall, F1 and Support.

- **Precision**

Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model.

- **Recall or Sensitivity**

Recall may be defined as the number of positives returned by our ML model.

- **Specificity**

Specificity, in contrast to recall, may be defined as the number of negatives returned by our ML model.
We can easily calculate it by confusion matrix with the help of following formula −

- **Support**

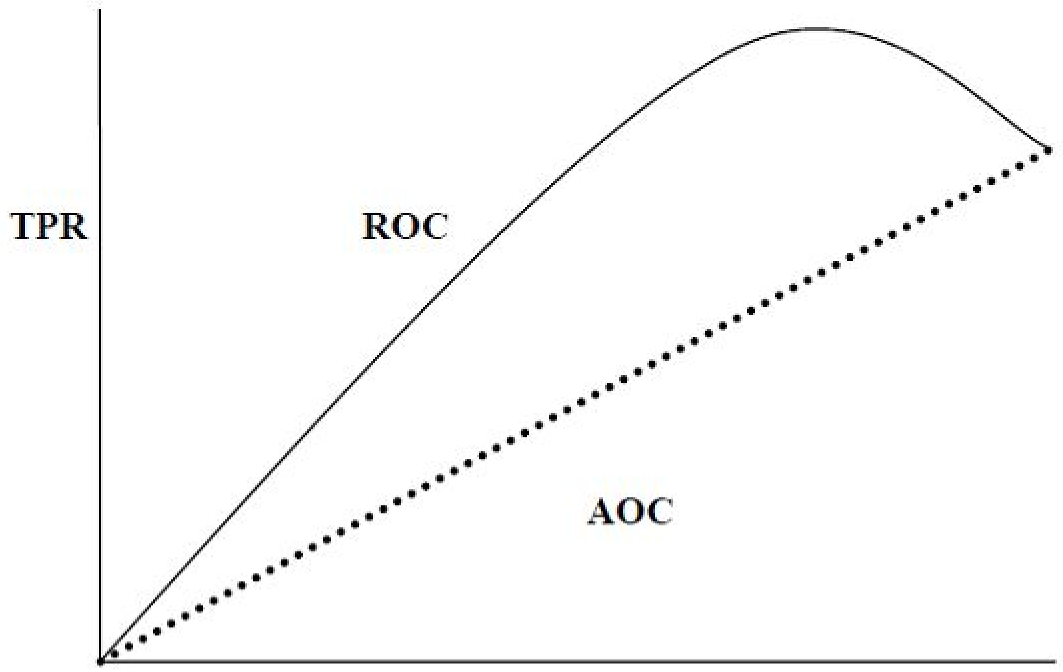
Support may be defined as the number of samples of the true response that lies in each class of target values.

- **F1 Score**

This score will give us the harmonic mean of precision and recall. Mathematically, F1 score is the weighted average of the precision and recall. **The best value of F1 would be 1 and worst would be 0**. We can calculate F1 score with the help of following formula −
***F*1 = 2 ∗ (*precision* ∗ *recall*) / (*precision* + *recall*)**
F1 score is having equal relative contribution of precision and recall.

AUC (Area Under ROC curve)
AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric, based on varying threshold values, for classification problems. As the name suggests, ROC is a probability curve and AUC measures the separability. In simple words, AUC-ROC metric will tell us about the capability of the model in distinguishing the classes. Higher the AUC, better the model.

Mathematically, it can be created by plotting TPR (True Positive Rate) i.e. Sensitivity or recall vs FPR (False Positive Rate) i.e. 1-Specificity, at various threshold values. Following is the graph showing ROC, AUC having TPR at y-axis and FORmat x-axis −



We can use the roc_auc_score function of sklearn.metrics to compute AUC-ROC.

- **LOG LOSS (Logarithmic Loss)**

It is also called Logistic regression loss or cross-entropy loss. It is basically defined on probability estimates and measures the performance of a classification model where the input is a probability value between 0 and 1. It can be understood more clearly by differentiating it with accuracy. As we know that accuracy is the count of predictions (predicted value = actual value) in our model whereas Log Loss is the amount of uncertainty of our prediction based on how much it varies from the actual label. With the help of Log Loss value, we can have a more accurate view of the performance of our model. We can use the log_loss function of sklearn.metrics to compute Log Loss.

Performance Metrics for Regression Problems
We have discussed regression and its algorithms in previous chapters. Here, we are going to discuss various performance metrics that can be used to evaluate predictions for regression problems.

- **Mean Absolute Error (MAE)**

It is the simplest error metric used in regression problems. It is basically the sum of the average of the absolute difference between the predicted and actual values. In simple words, with MAE, we can get an idea of how wrong the predictions were. MAE does not indicate the direction of the model i.e. no indication about underperformance or overperformance of the model.

- **Mean Square Error (MSE)**

MSE is like the MAE, but the only difference is that it squares the difference of actual and predicted output values before summing them all instead of using the absolute value. The difference can be noticed in the following equation −

- **R Squared (R2)**

R Squared metric is generally used for explanatory purpose and provides an indication of the goodness or fit of a set of predicted output values to the actual output values. The following formula will help us understanding it −
In the above equation, numerator is MSE and the denominator is the variance in *Y* values.
We can use the r2_score function of sklearn.metrics to compute R squared value.

ENSEMBLE TECHNIQUES

Ensembles can give us a boost in the machine learning result by combining several models. Basically, ensemble models consist of several individually trained supervised learning models and their results are merged in various ways to achieve better predictive performance compared to a single model. Ensemble methods can be divided into following two groups −

Sequential ensemble methods

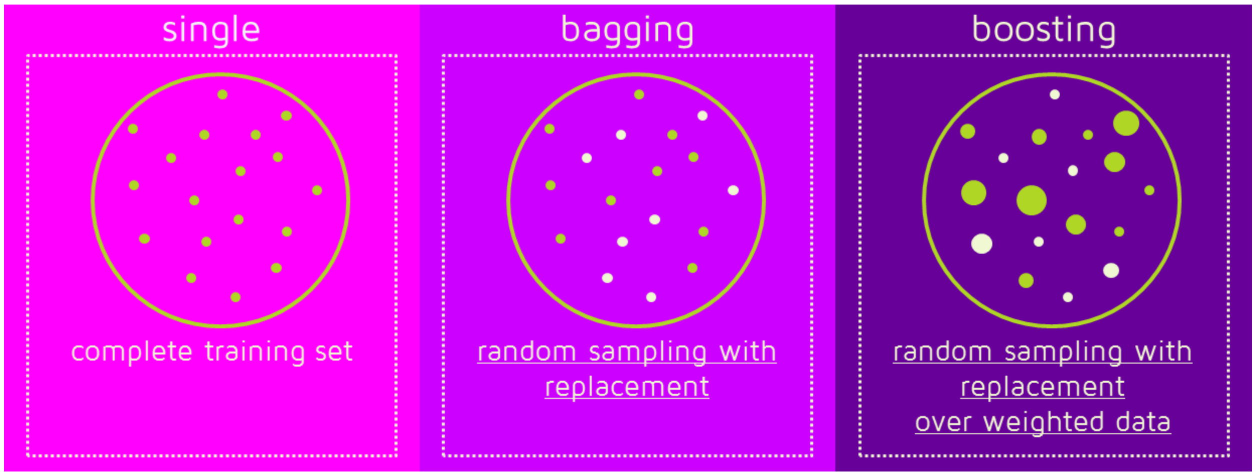
As the name implies, in these kinds of ensemble methods, the base learners are generated sequentially. The motivation of such methods is to exploit the dependency among base learners.

Parallel ensemble methods

As the name implies, in these kinds of ensemble methods, the base learners are generated in parallel. The motivation of such methods is to exploit the independence among base learners.

Ensemble Learning Methods

The following are the most popular ensemble learning methods i.e. the methods for combining the predictions from different models −


single
complete training set
bagging
random sampling with replacement
boosting
random sampling with replacement over weighted data

Bagging

The term bagging is also known as bootstrap aggregation. In bagging methods, ensemble models try to improve prediction accuracy and decrease model variance by combining predictions of individual models trained over randomly generated training samples. The final prediction of the ensemble model will be given by calculating the average of all predictions from the individual estimators. One of the best examples of bagging methods are random forests.

Boosting

In boosting methods, the main principle of building ensemble models is to build it incrementally by training each base model estimator sequentially. As the name suggests, it basically combines several weak base learners, trained sequentially over multiple iterations of training data, to build a powerful ensemble. During the training of week base learners, higher weights are assigned to those learners which were misclassified earlier. An example of a boosting method is AdaBoost.

Voting

In this ensemble learning model, multiple models of different types are built and some simple statistics, like calculating mean or median etc., are used to combine the predictions. This prediction will serve as the additional input for training to make the final prediction.

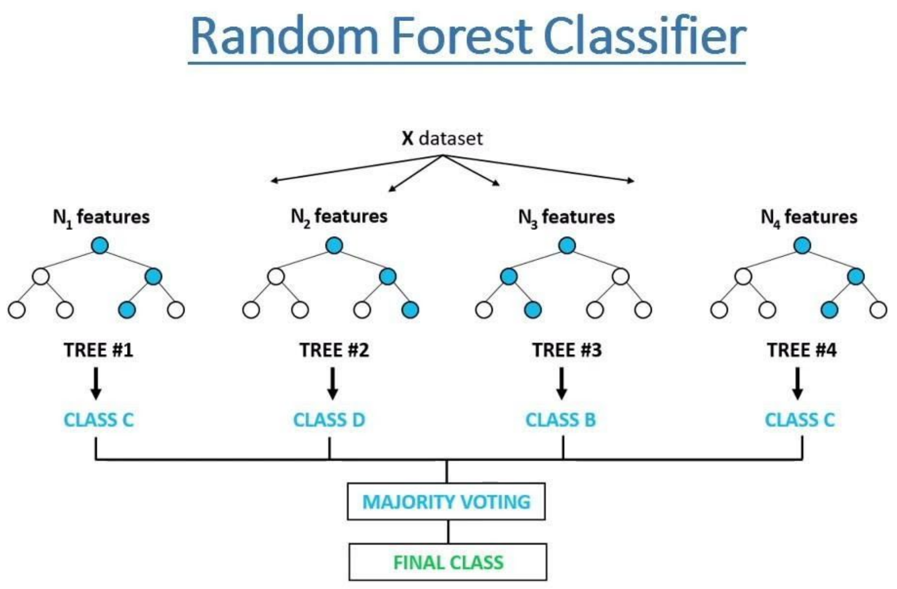
Bagging Ensemble Algorithms

The following are three bagging ensemble algorithms −

- **Bagged Decision Tree**

As we know that bagging ensemble methods work well with the algorithms that have high variance and, in this concern, the best one is decision tree algorithm. In the following Python recipe, we are going to build a bagged decision tree ensemble model by using the BaggingClassifier function of sklearn with DecisionTreeClasifier (a classification & regression trees algorithm) on Pima Indians diabetes dataset.

- **Random Forest**



It is an extension of bagged decision trees. For individual classifiers, the samples of training dataset are taken with replacement, but the trees are constructed in such a way that reduces the correlation between them. Also, a random subset of features is considered to choose each split point rather than greedily choosing the best split point in construction of each tree.

In the following Python recipe, we are going to build a bagged random forest ensemble model by using RandomForestClassifier class of sklearn on Pima Indians diabetes dataset.

- **Extra Trees**

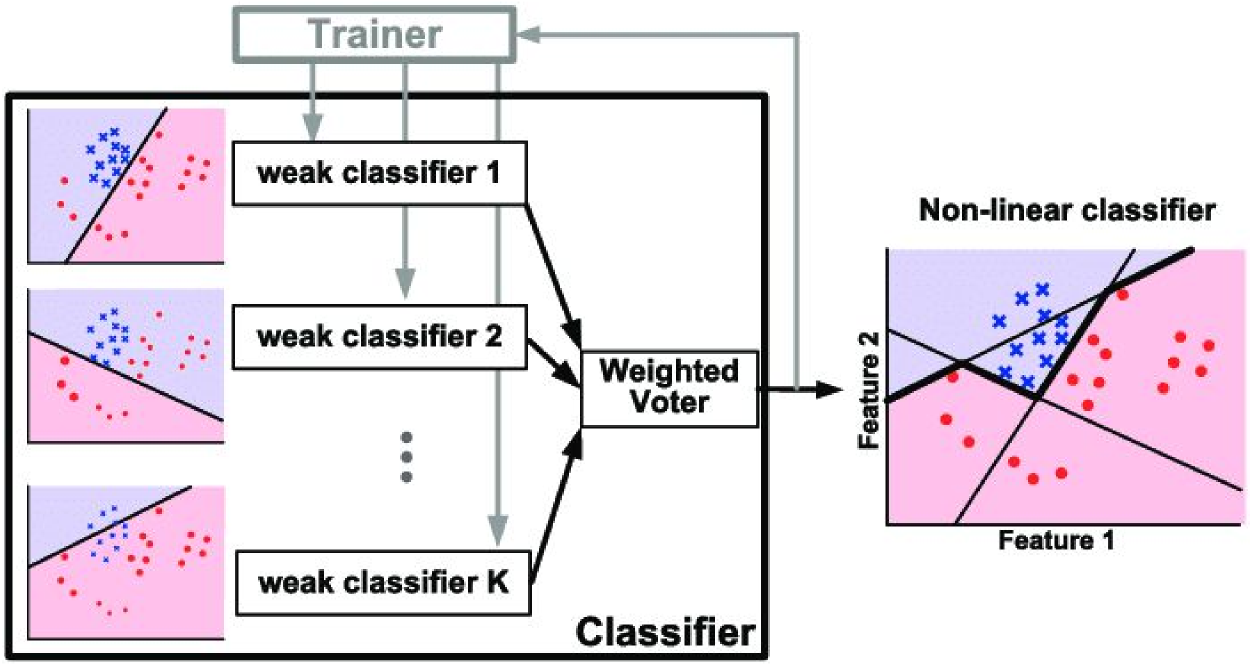
It is another extension of the bagged decision tree ensemble method. In this method, the random trees are constructed from the samples of the training dataset.

In the following Python recipe, we are going to build an extra tree ensemble model by using ExtraTreesClassifier class of sklearn on Pima Indians diabetes dataset.

Boosting Ensemble Algorithms

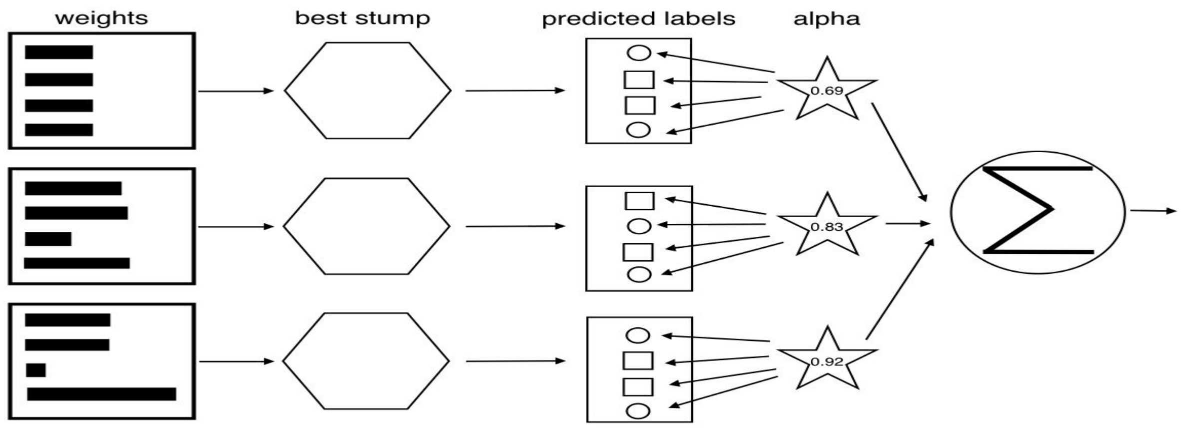
The followings are the two most common boosting ensemble algorithms −

- **AdaBoost**



It is one the most successful boosting ensemble algorithms. The main key of this algorithm is in the way they give weights to the instances in the dataset. Due to this the algorithm needs to pay less attention to the instances while constructing subsequent models.

In the following Python recipe, we are going to build AdaBoost ensemble model for classification by using AdaBoostClassifier class of sklearn on Pima Indians diabetes dataset.



- **Stochastic Gradient Boosting**

It is also called Gradient Boosting Machines. In the following Python recipe, we are going to build a Stochastic Gradient Boosting Ensemble model for classification by using GradientBoostingClassifier class of sklearn on Pima Indians diabetes dataset.

- **Voting Ensemble Algorithms**

As discussed, voting first creates two or more standalone models from training dataset and then a voting classifier will wrap the model along with taking the average of the predictions of the sub-model whenever needed new data.

In the following Python recipe, we are going to build a Voting ensemble model for classification by using the VotingClassifier class of sklearn on Pima Indians diabetes dataset. We are combining the predictions of logistic regression, Decision Tree classifier and SVM together for a classification problem

TUNING

As we know that ML models are parameterized in such a way that their behavior can be adjusted for a specific problem. Algorithm tuning means finding the best combination of these parameters so that the performance of the ML model can be improved. This process is sometimes called hyperparameter optimization and the parameters of the algorithm itself are called hyperparameters and coefficients found by ML algorithms are called parameters.

Here, we are going to discuss some methods for algorithm parameter tuning provided by Python Scikit-learn.

Grid Search Parameter Tuning(GridSearchCV)

It is a parameter tuning approach. The key point of working on this method is that it builds and evaluates the model methodically for every possible combination of algorithm parameters specified in a grid. Hence, we can say that this algorithm is having search nature.

Random Search Parameter Tuning(RandomizedSearchCV)

It is a parameter tuning approach. The key point of working of this method is that it samples the algorithm parameters from a random distribution for a fixed number of iterations.