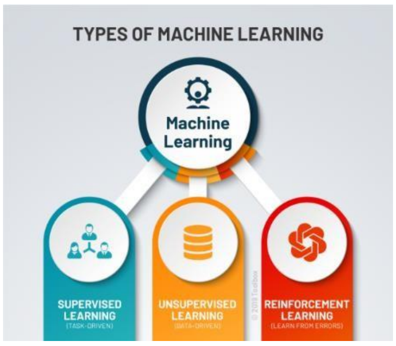
- **Machine Leaíning** is defined as the study of computeí píogíams that leveíage algoíithms and statistical models to leaín thíough infeíence and patteíns without being explicitly píogíammed.



<u>Supervised Machine learning:</u>

- Data is labeled and the algorithms learn to predict the output from the input data
- Learning stops when the algorithm achieves an acceptable level of performance
- It is classified as
    1. Regression
    2. Classification

Regression Algorithm	Classification Algorithm
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.

ML Modelling Process Flow:

- **Get data:** Gather data from different sources
- **Clean, prepare and Manipulate data:**
    1. Handling null values and outliers
    2. Feature selection/ Variable Selection based on domain knowledge
    3. Converting Categorical data to Numerical
    4. Scaling the data
- **EDA:** Generating reports and Graphs
- **Splitting the Data:** Creating train and test data
- **Train Model:** Build a model using train data (usually 70% to 80% of whole data)
- **Evaluation/Test Model:** Test the model using the test data(20% to 30% of the whole data)
- **Model Tuning:** Optimize the model to increase its accuracy

Linear Regression

- **Linear Regression Steps:**

1. Create the dataframe properly--> pd.read_csv(), pd.read_excel()
2. Pre-processing the data:
    a. Feature selection-->domain knowledge-->drop()
    b. Handling the missing values-->isnull().sum(),fillna(),dropna()
    c. Converting the categorical data to numerical-->map()
3. Assumption 1: There should be no outliers in the data-->boxplot()
4. Assumption 2: Assumption of Linearity: Every ind var should have a linear relationship with the dep var-->pairplot(), drop()
5. Create X and Y--> X= ind vars, Y=dep var
6. Assumption 3: Assumption of Normality: The dependent variable should follow an approximate normal distribution-->distplot(),log()
7. Check and handle the skewness in the X vars-->hist(),skew(),log1p()
8. Assumption 4: Assumption of no multicollinearity: There should be no multicollinearity between the independent variables-->corr(),heatmap(),VIF(),drop()
9. Splitting the data into train and test(validation)-->train_test_split()
10. Building the model:
    a. Create the model-->obj=AlgoName()
    b. Train the model-->obj.fit(X_train,Y_train)
    c. Predict using the model-->Y_pred=obj.predict(X_test)
11. Evaluate the model:
    score, R-squared, Adj R-squared, RMSE, AIC/BIC
12. Assumption 5: There should be no auto-correlation in the data-->Durbin Watson test
13. Assumption 6: Errors should be random-->Residual v/s Fitted plot
14. Assumption 7: Errors should follow an approx normal distribution-->Normal QQ plot
15. Assumption 8: Errors should follow a constant variance(Homoskedasticity)-->Scale-Location plot
16. Tuning the model:
    a. Feature selection-->domain knowledge, p-values
    b. Regularization techniques-->Ridge(),Lasso()
    c. Stochastic Gradient Descent

Linear Regression Definition:

- Regression is a statistical technique which helps you to measure the relationship between the independent variables and dependent variables
- It helps you to understand one unit change in the independent variables is going to cause how many units change in the dependent variable
- Dependent or predicted variable is represented as 'y'

LR Approaches:

1. Statistical – Ordinary least Squares (OLS)
2. Machine Learning – Stochastic Gradient Descent (SGD)
    - **OLS is used for linear regression as it will return the best fit line which gives least errors**

Types of Linear Regression:

1. <u>Simple Linear Regression</u>: 1 X- variable and Y- Variable

   Example -predicting salary on the basis of experience

   Equation:

   $$Y = \beta 0 + \beta 1 . X$$

   were,

   $$\beta 0 = Point\ on\ Y\ axis\ where\ best\ fit\ line\ intersect\ it,\ the\ value\ of\ Y\ when\ X\ is\ zero$$

   Y = Dependent variable

   X = Independent variable

   $$\beta 1 = Slope\ coefficient\ of\ X - variable$$

   It indicates one unit change in X cause how many units change in Y.

Positive Slope Coefficient → X increases Y increases

Negative Slope Coefficient → X increases Y decreases

0 or close to 0 → No Relation

1. Multiple Linear Regression: Multiple X- variable and 1 Y- Variable
   Example: Predicting Salary on the basis of Department, age, domain etc.

   Equation:

   $$Y = \beta 0 + \beta 1 . X1 + \beta 2 . X2 + \cdots . \beta n Xn$$

   where,

   $\beta 0 = Point\ on\ Y\ axis\ where\ best\ fit\ line\ intersect\ it,$
   $the\ value\ of\ Y\ when\ X\ is\ zero$

   Y = Dependent variable

   X1, X2, Xn = Independent variables

   $\beta 1, \beta 2, \beta n = Slope\ coefficients\ of\ X - variables$

   It indicates one unit change in X cause how many units change in Y.

Positive Slope Coefficient → X increases Y increases

Negative Slope Coefficient → X increases Y decreases

0 or close to 0 → No Relation

Multicollinearity

Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model.

How to Deal with Multicollinearity

- Remove some of the highly correlated independent variables.
- Linearly combine the independent variables, such as adding them together.

No Auto-Correlation:

Auto-Correlation measures the relationship between a variable's current value and its past value, mean errors are assumed to be uncorrelated i.e. randomly spread around the regression line

No patterns in Graph indicates no Auto - Correlation

Durbin Watson Test for Auto-Correlation:

It takes a value between 0 – 4

- If value is close to 0 → No Auto-correlation
- If value is close to 2 → Positive Auto-correlation
- If value is close to 4 → Negative Auto-correlation

<u>Decomposition Variability/ Evaluation matrices:</u>

- It is a determinant for a good regression
- It includes
  1. Sum of Squares total
  2. Sum of Squares regression
  3. Sum of Squares Error
  4. R – Squared
  5. Adjusted R – Squared
  6. AIC & BIC
  7. RMSE

Sum of Squares total (SST):

The square of the difference between the observed dependent variable and its mean

$$SST = \sum (y - \hat{y})^2$$

Sum of Squares regression (SSR):

The sum of the difference between the predicted value and the mean of dependent variable

$$SSR = \sum (y' - \hat{y}')^2$$

Sum of Squares Error (SSE):

The sum of the difference between the observed value and predicted value of the dependent variable

$$SSE = \sum (y - y')^2$$

Relation between SST, SSR, and SSE:

SST = SSR + SSE

Total Variability = Explained Variability + Unexplained Variability

AIC & BIC:

- It is used to compare different models with same algorithms but with different number of Independent Variables

1. **AIC**
   - Akaike Information criteria to penalize the inclusion of additional variables to the model
   - It adds penalty to those variables that increases the error
   - The model with lowest AIC is selected
   - The penalty of AIC is less compared to BIC, causing AIC to pick More complex models
2. **BIC**
   - Bayesian Information Criterion is a variant of AIC with a stronger penalty for including additional variables to the model
   - It adds penalty to those variables that increases the error
   - The model with lowest BIC is selected
   - As compared to AIC, it penalizes model complexity more heavily

RMSE:

- Root Mean Square error is an absolute measure of the goodness for the fit
- It gives an absolute number on how much your predicted results deviate from the actual number
- Low the RMSE better the model

R Squared:

- It tells you how well the regression model is predicting as compared to the mean model
- Lies between (0-1)
- If R squared is close to 1 → very good model
- If R squared is close to 0.5 → Needs tuning
- If R squared is close to 0 → Not a good model
- If R squared is less than 0 → Mean model is better than the regression model

Adjusted – R Squared:

- Penalized R Squared Value
- It will always be lower than R Squared
- As more data is added R Squared goes on increasing whereas Adj -R Squared will increase only when significant data is added to the model hence it is more reliable to quote the adjusted R Squared to the client

Tuning the Model:

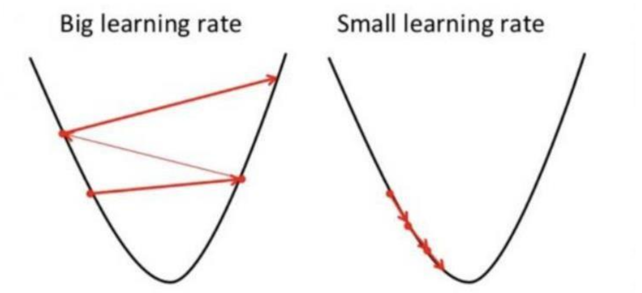
1. **Feature Selection** – It is a process of selecting relevant features from the dataset

2. **Dedicated approach** – Regularization techniques (penalty-based models) to avoid overfitting and underfitting
    - **Ridge regression** identifies the insignificant variables and lowers its β values such that the overall impact of those variables on the model is reduces
    - Ridge is preferred when we have less number of variables

    - **Lasso Regression** identifies the insignificant variables and forces its β vales to be 0 such that the overall impact of those variables on the model is eliminated
    - Lasso is preferred when we have larger number of variables

Stochastic Gradient Descent:

- Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs.
- In other words, SGD is used to minimize the error/difference between the y – actual and y – predicted
- In SGD we take one observation at a time, model learns on the data multiple times and then we use the model for predictions.
- **Hyper Parameters in SGD:**
    1. Learning rate(α) – how quickly an algorithm updates its parameter, mostly learning rates between 0.0001 to 1 is considered, if learning rate is high, algorithm overshoots the minima (lowest error), and if low it requires many updates to reach minima.



2. Number of Epochs – Number of times it has to run through the training data, one epoch is a single cycle to the entire data.

- **Stopping Criteria:**
  1. Maximum Number of Epochs
  2. Zero Convergence

In SGD we find best 'w' and 'b' values

Where w is weight of independent variables & b is the biasness of the model to a particular weight initially b=0

SGD Workflow:

1. Initialise weights
2. Feed data (as it is SGD one data at a time)
3. Predict Y
4. Compute loss (Ypred -Yactual) ^2
5. Compute bias
6. Update new weights

These six steps complete one epoch

Scaling and its Importance

1. Normalization → MinMaxScaler() → 0 to 1
2. Standardization → StandardScaler() → -3 to 3
   - Scaling is important for making the values unitless
   - Scaling technique helps every variable to follow standard normal distribution
   - In this technique data is well distributed
   - Outliers doesn't have any impact
   - It is not a mandatory step but a good practice

Summary

OLS → We take the entire data at once model learns on the entire data we use the model for prediction

Batch Gradient descent → We take the entire data at once model learns on the entire data multiple times, we use the model for prediction

Mini – Batch Gradient Descent → We take the subset of the data at a time model learns on the data multiple times, we use the model for prediction

SGD → We take one observation at a time, model learns on the data multiple times, we use the model for prediction

Why SGD is better than Batch GD?

- In batch GD it takes time to go through all observations, and errors are given after a long time so its time consuming whereas SGD travers through one observation at a time and error is given fast and it moves to the next observation quickly so it takes less amount of time
- Neural Networks are complex models and time consuming so if we use Batch GD it becomes more time consuming so to avoid that SGD is used.

Logistic Regression

Classification Steps:

1. Create the dataframe properly-->pd.read_csv(),pd.read_excel()
2. Preprocessing the data:
    a. Feature Selection-->domain knowledge-->drop()
    b. Handling the missing values-->isnull().sum(),fillna(),dropna()
3. Converting the categorical data into numerical-->map(),get_dummies(),LabelEncoder()
4. Create X and Y-->X=ind var, Y=dep var-->X=df.values[:,:-1], Y=df.values[:,-1]
5. Scale the data[Optional]-->MinMaxScaler(),StandardScaler()
6. Splitting the data into Train and Test(validation)-->train_test_split()
7. Build the model:
    a. Create the model-->obj=AlgoName()
    b. Train the model-->obj.fit(X_train, Y_train)
    c. Predict using the model-->Y_pred=obj.predict(X_test)
8. Evaluate the model:
    a. confusion_matrix(Y_test,Y_pred)
    b. accuracy_score(Y_test,Y_pred)
    c. classification_report(Y_test,Y_pred)
9. Tuning the model:
    a. Feature Selection
    b. Dedicated approach-->Adjusting the threshold
    c. Stochastic Gradient Descent

Model is ready--classifier-->adult_data.csv

Follow the steps on the adult_test.csv:

1. Create a dataframe properly-->adult_test
2. Preprocessing the data:
    a. Feature selection-->eliminate fnlwgt,education
    b. Handling the missing values
3. Converting categorical values to numerical
4. Create X_test_new and Y_test_new
5. Scaling the data-->X_test_new-->X_test_new=scaler.transform(X_test_new),fit() not to be implemented
6. Y_pred_prob=classifier.predict_proba(X_test_new)
7. Use the if-else code with threshold=0.46(optimum) and generate Y_pred_new

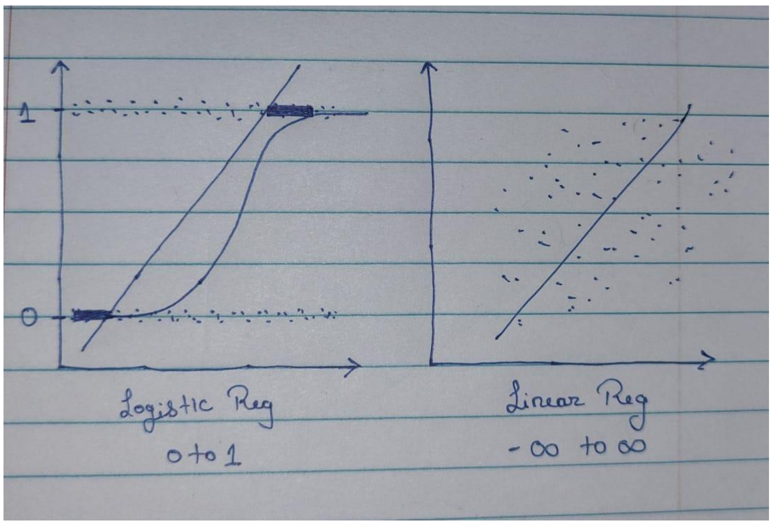
8. Evaluating the model:
    a. confusion_matrix(Y_test_new,Y_pred_new)
    b. accuracy_score(Y_test_new,Y_pred_new)
    c. classification_report(Y_test_new,Y_pred_new)

- Logistic Regression Used for Classification
- It is Supervised Machine Learning algorithm
- Only Suitable for binary Classification
- It is not Suitable for Multiple Classes

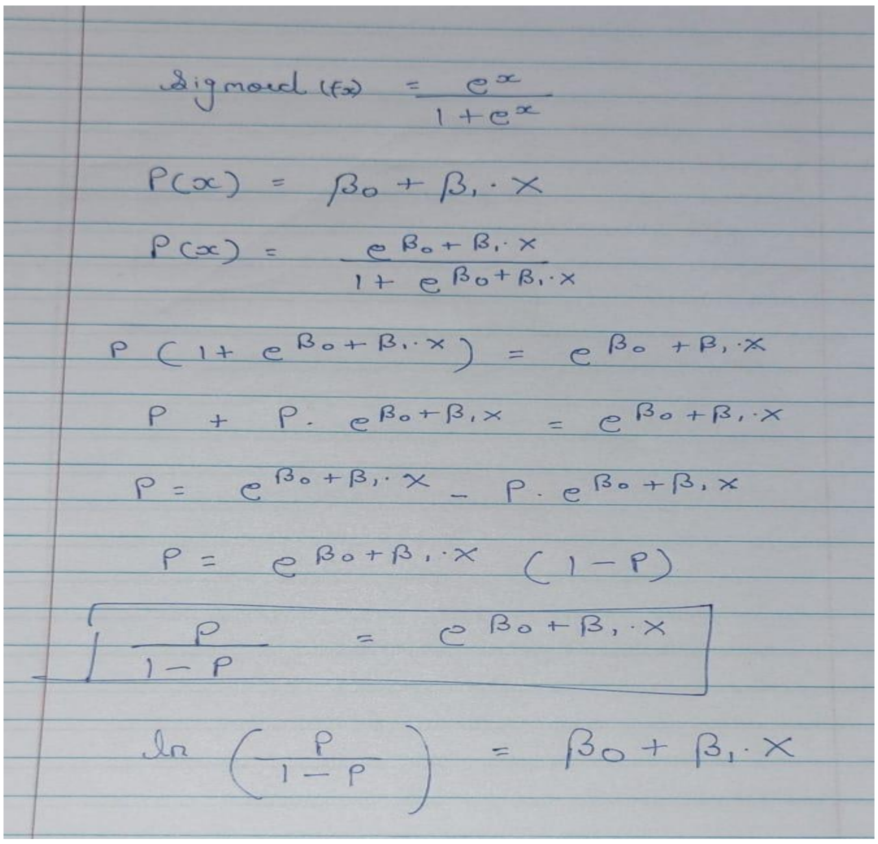
In linear regression we have range from -∞ to +∞, but in logistic Regression we have range from 0 to 1 as we are working on binary classification of data, so the data points would be either at 0 or 1 which makes the best fit line from linear regression not suitable for such type of data, so the solution to this would be any value going above one would be considered as 1 and any value going below zero would be considered as 0 , which again gives rise to a problem of having three straight lines, so this was also solved by making it as a S- shaped curve giving rise to the sigmoid function.

Sigmoid Function – It allows to convert any real value data to the range of 0 to 1

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



- **Logistic regression equation is derived from the sigmoid function**



$$\mathrm{sigmoid}(fx) = \frac{e^{x}}{1+e^{x}}$$

$$P(x) = \beta_0 + \beta_1 \cdot X$$

$$P(x) = \frac{e^{\beta_0+\beta_1 \cdot X}}{1+e^{\beta_0+\beta_1 \cdot X}}$$

$$P\left(1+e^{\beta_0+\beta_1 \cdot X}\right) = e^{\beta_0+\beta_1 \cdot X}$$

$$P + P \cdot e^{\beta_0+\beta_1 X} = e^{\beta_0+\beta_1 \cdot X}$$

$$P = e^{\beta_0+\beta_1 \cdot X} - P \cdot e^{\beta_0+\beta_1 X}$$

$$P = e^{\beta_0+\beta_1 \cdot X}(1-P)$$

$$\boxed{\frac{P}{1-P} = e^{\beta_0+\beta_1 \cdot X}}$$

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 \cdot X$$

Were,

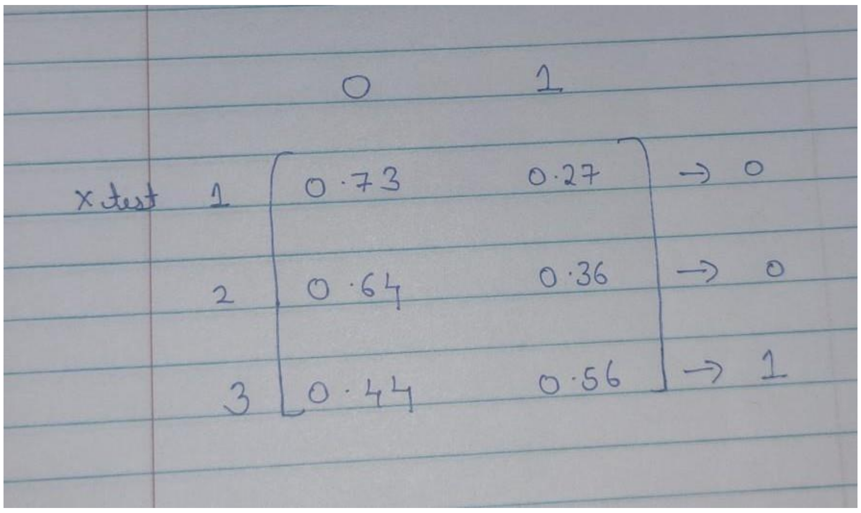
- P/1 - P = Odd's Ratio → probability of success divided by the probability of failure
- We take log to eliminate 'e' to get the linear regression equation hence the name **Logistic regression**
- when we take log the LHS is called as **logit function** So while performing logistic regression in the background we have Linear Regression running, it is because of the logit function the values are in the range of 0-1

<u>Probability Matrix</u>

Some data points are at 0 and some at 1, while doing prediction assumption is midpoint 0.5 is default threshold anything less than 0.5 is belong to class zero and more than 0.5 belong to class 1, however the values being continuous it reaches to 0 or 1 internally with the help of **probability matrix**

Probability matrix will be generated in the background based on y values

Probability matrix is finally responsible for giving the final y value

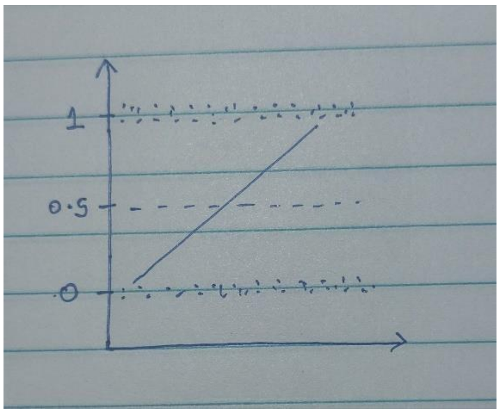


$$
\begin{array}{cc}
 & \begin{array}{cc} 0 & 1 \end{array} \\
\begin{array}{c} x\,test\ 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0.73 & 0.27 \\ 0.64 & 0.36 \\ 0.44 & 0.56 \end{bmatrix}
\end{array}
\begin{array}{c} \rightarrow 0 \\ \rightarrow 0 \\ \rightarrow 1 \end{array}
$$

- **Threshold**

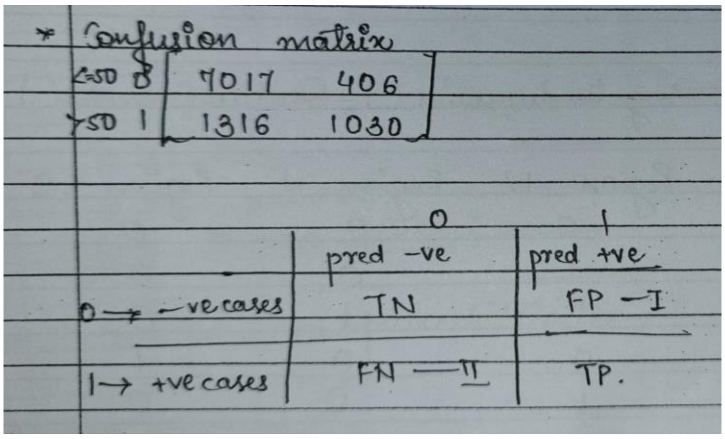
  Anything < 0.5 == 0

  Anything > 0.5 ==1



Evaluation

- Confusion Matrix



* Confusion matrix

$$\begin{array}{cc|cc} \leq 50 & 0 & 7017 & 406 \\ >50 & 1 & 1316 & 1030 \end{array}$$

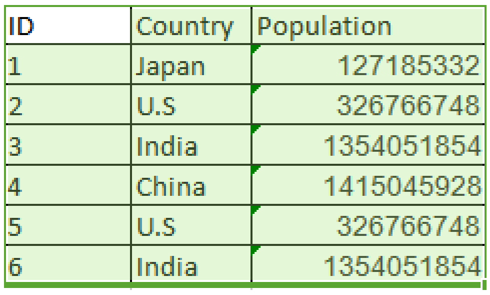
	0 pred -ve	1 pred +ve
0 → -ve cases	TN	FP — I
1 → +ve cases	FN — II	TP.

- Accuracy Score → percentage of correct predictions, higher the accuracy better the model
  - TN+TP/TP+FP+FN+TP
- Recall Value → accuracy score of individual classes
  1. Class 0 – TN/TN + FP
  2. Class 1 – TP/TP + FN
- Precision → It tells you about how relevant are the predictions
  1. Class 0 : How many negative predictions are correct?
     = TN/TN + FN
  2. Class 1 - How many positive predictions are correct?
     = TP/TP + FP

- F1 Score Value → harmonic mean of precision and recall
  2*(precision*recall) / precision + recall

Encoding (converting categorical data to numeric):

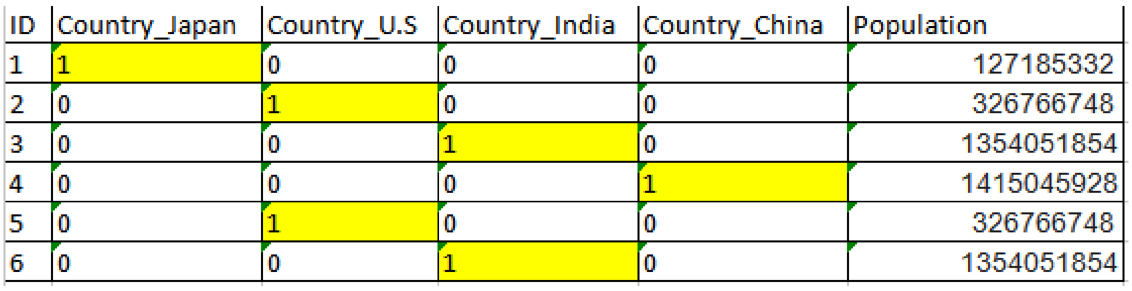
1. Manual Encoding → map()



ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

2. Dummy Variable → get_dummies(), OneHotEncoder()

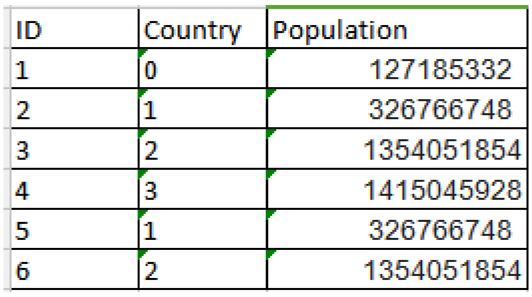
- one hot encoding takes a column which has categorical data, which has been label encoded and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.
- It becomes more confusing to understand
- No of steps goes on increasing so it is not so preferred function.



ID	Country_Japan	Country_U.S	Country_India	Country_China	Population
1	1	0	0	0	127185332
2	0	1	0	0	326766748
3	0	0	1	0	1354051854
4	0	0	0	1	1415045928
5	0	1	0	0	326766748
6	0	0	1	0	1354051854

3. Creating levels →labelEncoder ()

    - Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.
    - It assigns values to the variable in ascending alphabetical order



ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

<u>Overfitting and Underfitting:</u>

- Overfitting: Good performance on the training data, poor performance to unseen data.
- Underfitting: Poor performance on the training data poor performance to unseen data.

Tuning:

1. **Feature Selection** → Done with the help of some inbuild functions
   I. Recursive feature elimination – RFE ()
   Code = RFE (classifier, no of variables to be retained in the model)
   It will Start will all variables and start eliminating the variable one by one till retained number of variables are obtained
   II. Univariant feature selection – SelectKBest ()
   Code = SelectKBest (k = no of variables to be retained in the model, score_func=chi2)
   III. VarianceThreshold – On the basis of variance we eliminate those variables where there is less variety
   Code = VarianceThreshold (desired threshold)
2. **Dedicated approach**

- Adjusting the threshold

The default threshold is 0.5 that means Anything < 0.5 == 0 and anything > 0.5 ==1, we can do train and error and adjust the threshold from 0.4 to 0.6 depending upon which class is not performing good

If class 0 needs to be improved, the threshold is increased from 0.5 to 0.6 approx. so that the no of observations belonging to class 0 increases and model gets trained better

It is always advisable to keep the threshold between 0.4 to 0.6 to avoid overfitting and under fitting

3. **SGD**

- SGD for logistic Regression is same to that of linear Regression except 2 changes
  1. SGD Classifier is used in place of SGD Regressor
  2. One more Hyperparameter is added loss='log'

Cross Validation:

- How durable your model is on unseen data.
- Cross validation can only be implemented on train data
- K – fold divides or subsets the data into ‘K’ no of folds
  By default, k=3
- The entire data is divided into k- folds with the help of random sampling
- In each iteration k -1 data will be used as training data and remaining will be used for testing
- At the end of this step, we get the accuracy score of each iteration
- No of accuracy scores = No of k – folds if k=10 we will get 10 accuracy score
- We take mean of these accuracy scores and then compare it with the base model accuracy if both are similar, we conclude that the original model was doing a good job, it wasn’t overfitting or underfitting the data and we can go ahead with the model for further predictions
- If accuracy scores differ drastically, we go ahead with the cross-validation model instead of base model

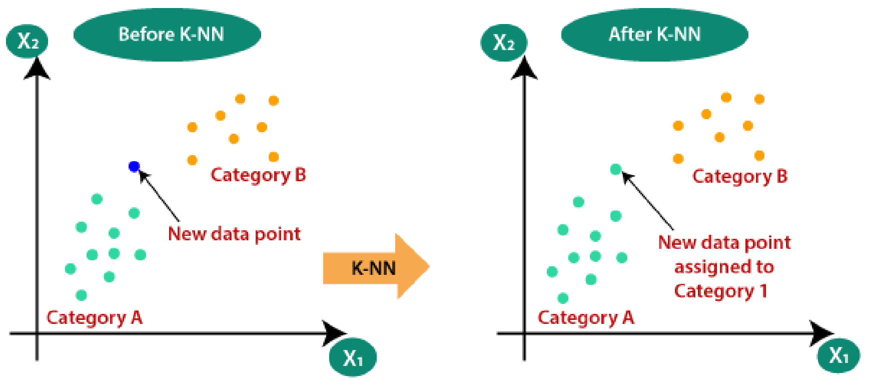
MLE

- In Linear Regression we use OLS to minimize the error in logistic Regression we use MLE
- MLE stands for Maximum Likelihood Estimation
- This method works upon finding the best beta values such that the predictions are accurate similar to best fit line in case of linear regression

K – Nearest Neighbour (KNN) (not effected by outliers)

- 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 data and available data and put the new data 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.
- KNN is Suitable for noisy data and smaller 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:



How does K-NN work?

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

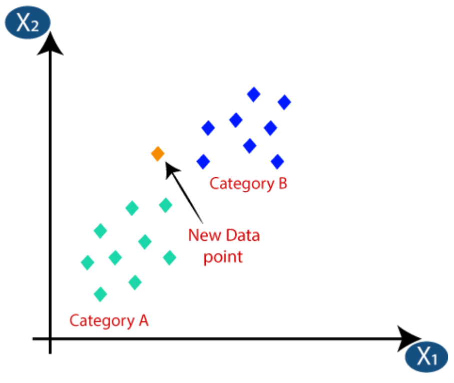
- **Step-1:** Select the number K of the neighbors

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

I. 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.
II. 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.
III. Large values for K are good, but it may find some difficulties.

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

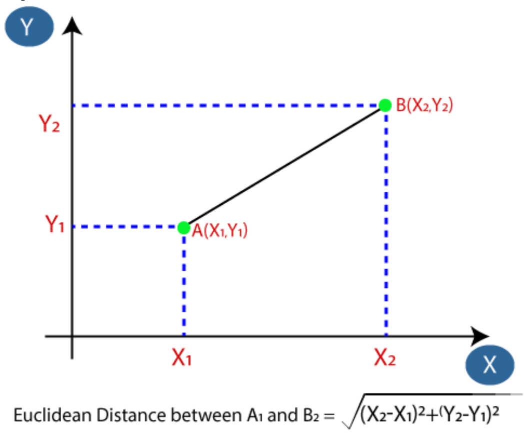
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



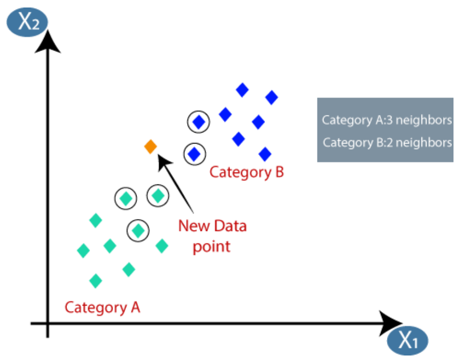
- Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points.

Euclidean Distance represents the shortest distance between two points.



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



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

<u>Tuning</u>

1. Feature Selection
2. Dedicated approach → Hyperparameter Tuning
   - Change the value ok K or the metric
   - We can apply for loop and find optimum value of k from 1 to its square root also the value of k should not be even its should be an odd number
   - Metric is by default **minkowski** we can change it to Euclidean, Manhattan or Hamming

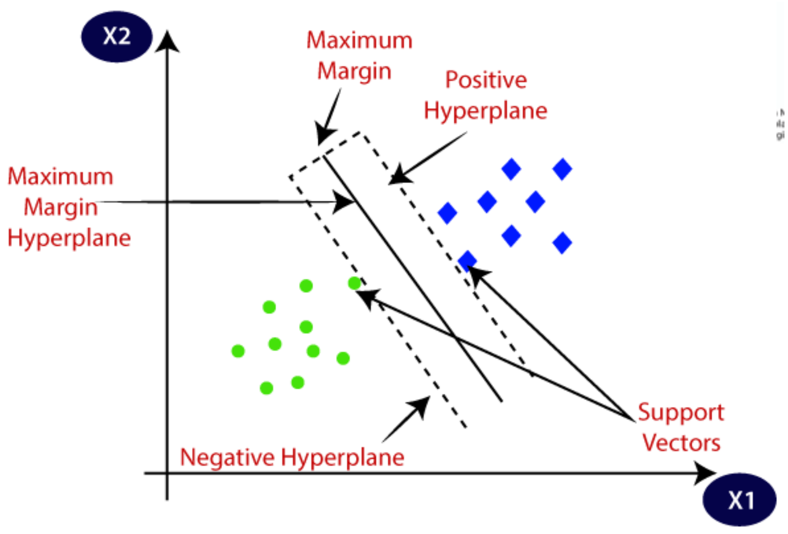
<u>**Support Vector Machine (SVM)**</u>

(Not affected by outliers)

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

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

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



SVM is suitable for high dimensional data (no of variables are high compared to number of observations)

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, 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:

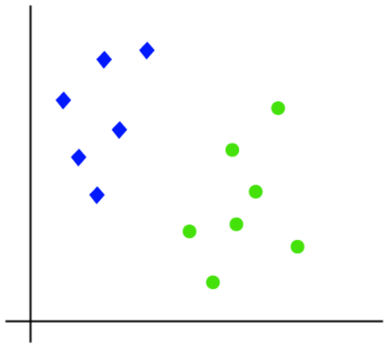
- Support vectors are those data points which represent the entire class and are nearest to the opposite class

SVM can be of two types:

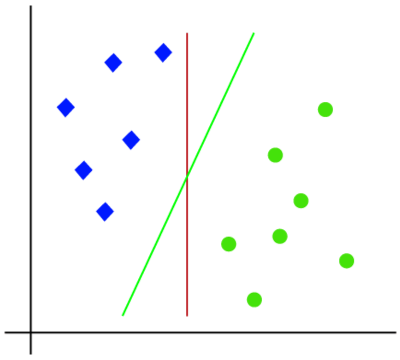
- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear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

Linear SVM:

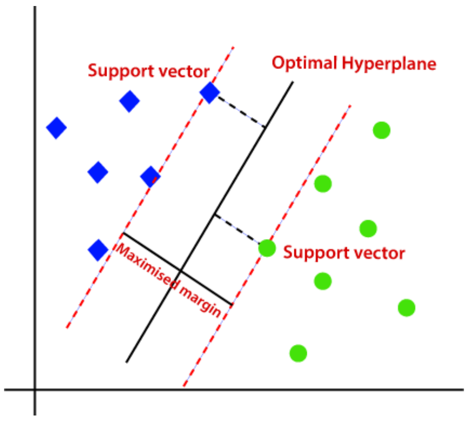
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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:

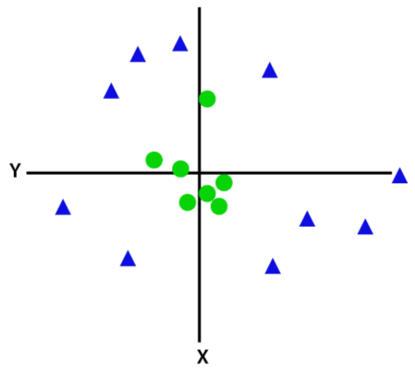


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane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

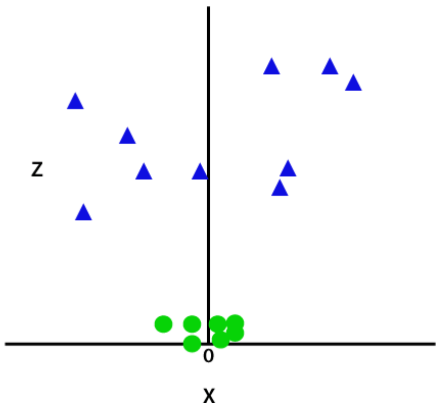


Non-Linear SVM:

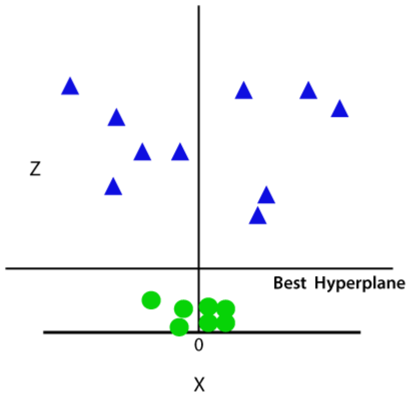
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



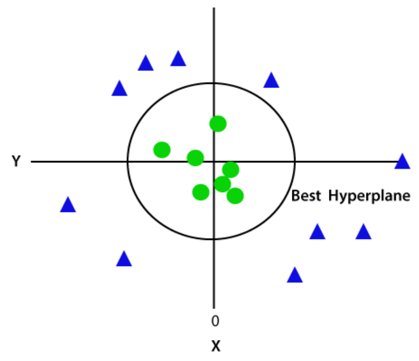
So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. By adding the third dimension, the sample space will become as below image:



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



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



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

Training:

Input → x, y

Output → best fit hyperplane equation

Wx +b =0 → hyperplane equation

Were,

b = bias

W = Weight

Testing:

Input → x

Output → y on the basis of equation

If the value of y < 0 towards -1, left class towards the hyperplane is the predicted class

If the value of y > 0 towards 1, right class towards the hyperplane is the predicted class

- In case of regression after substituting x values of the test data point into the equation, whatever value we get becomes the predicted value

Tuning:

1. Feature Selection
2. Hyperparameter Tuning (Kernel, Gamma, C)

- **Kernel**
  I. Linear → one of the least used kernel and it is only used when we have high dimensional data because studies have proved that very high dimensional data is usually linearly separable.
  II. Sigmoid → one of the least used kernel mostly suitable for binary classification
  III. Poly → It stands for Polynomial kernel used to fit non-linear data
  IV. rbf → radial basis function kernel which if required will perform the kernel trick transformation and fit an almost linear hyperplane into data, rbf is a **default kernel** used by sklearn because it can handle both linear and non-linear data.

What is kernel trick transformation?

- It allows to convert non – linearly separable data into a linearly separable data, It will try to map the non – linear data by adding a new dimension where it usually squares the original value such that it becomes linearly separable data and we can fit a linear hyperplane into it.

Gamma:

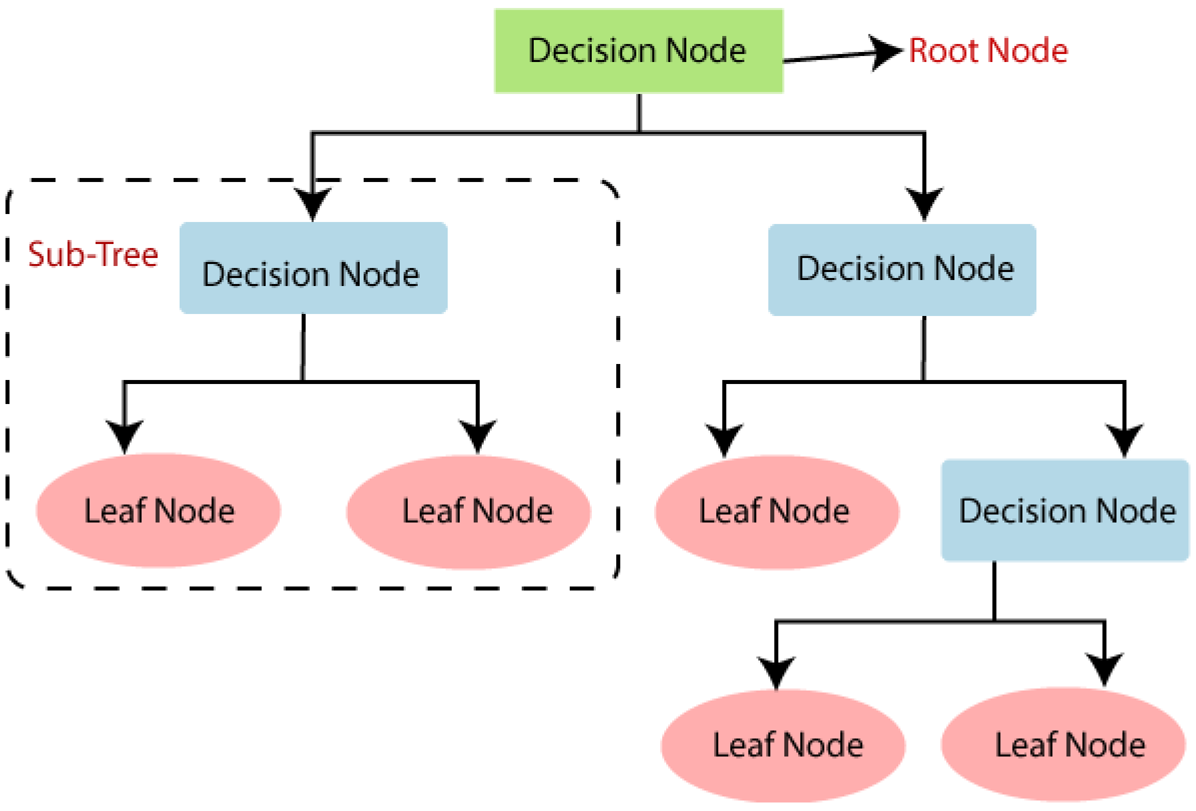
- Gamma value indicates how many farther support vectors should be considered while fitting the best fit hyperplane
- It is not applicable when kernel = Linear
- We can use it only when kernels are poly, rbf and sigmoid
- For the base model gamma = 0.1
- If we try to increase the gamma value i.e., 1,10, It will go very close to data points and fit the hyperplane which might lead to overfitting
- So, the ideal way to tune the gamma value is to reduce it toward 0 i.e., 0.01, 0.001 such that we end up getting good amount of boundary towards each of the class

C → Cast Parameter (regularization)

- It is a cost penalty implemented to penalize the model for the misclassification
- By default, C = 1, which is a low penalty which indicates we are ok with misclassification
- We can tune the c value we can increase it to 10, 20, 30 to indicate we are not ok with misclassifications
- But a very high value of c could shift the hyperplane a lot and may lead to an over – fitted model
- So ideally we should gradually increase the value of c only till the time we notice that model improves

Decision Tree

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



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

How does the Decision Tree algorithm Work?

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

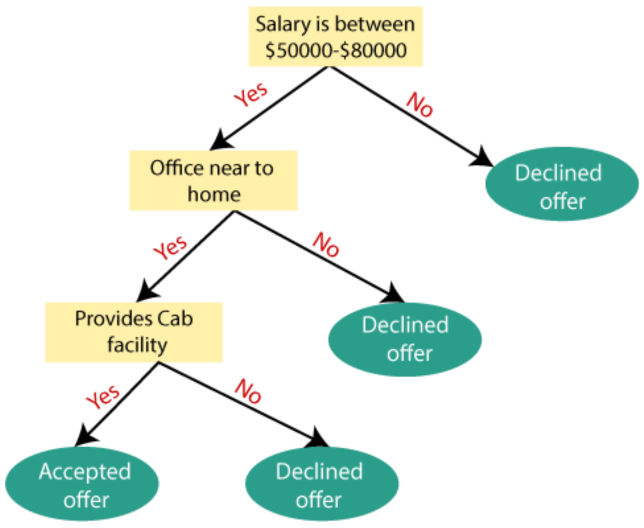
Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

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



Attribute Selection Measures

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

- **Information Gain:**
    - It calculates how much information a feature provides us about a class.
    - According to the value of information gain, we split the node and build the decision tree.
    - A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first

- It can be calculated using the below formula

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

Were,

Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations

- **Gini Index:**

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

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

Pruning:

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

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.

Hyperparameter Tuning:

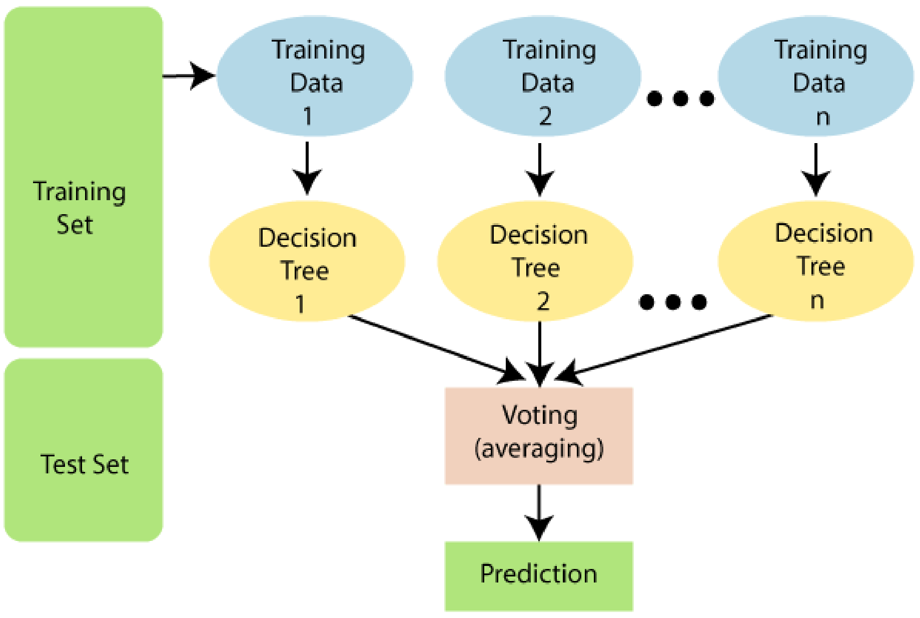
Hyperparameters in decision tree are:

1. Max_depth → Indicates how deep decision tree can be, deeper the tree more splits it has and captures more information about data, A tree overfits for larger depth values

2. Min_samples_split → minimum number of samples required to split an internal node
3. Min_samples_leaf → minimum number of samples required to be at a leaf node
4. Max_features → No of features to consider when looking for the best split
5. Criterion → supported criterion are 'gini' and 'entropy'

Random forest

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.



Ensemble Modelling:

- What is Ensembling?
  Ensembling can be defined as a machine learning technique which combines several base models in order to produce one optimal predictive model
  Ensemble methods helps in decreasing variance, bias and improve predictions

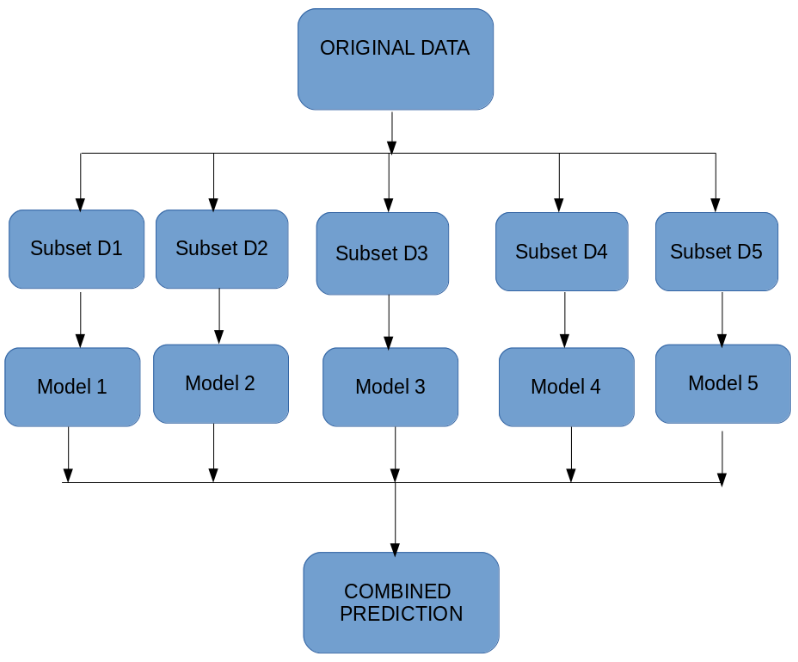
Ensemble Techniques:

- **Bagging** is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.

- **Bagging** works as follows: -

1. Multiple subsets are created from the original dataset, selecting observations with replacement.
2. A base model (weak model) is created on each of these subsets.
3. The models run in parallel and are independent of each other.
4. The final predictions are determined by combining the predictions from all the models.

Now, bagging can be represented diagrammatically as follows:



Boosting is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.
Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model.

In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analysing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.

- **Let's understand the way boosting works in the below steps**.

1. A subset is created from the original dataset.
2. Initially, all data points are given equal weights.
3. A base model is created on this subset.
4. This model is used to make predictions on the whole dataset.
5. Errors are calculated using the actual values and predicted values.
6. The observations which are incorrectly predicted, are given higher weights
7. Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)
8. Similarly, multiple models are created, each correcting the errors of the previous model.
9. The final model (strong learner) is the weighted mean of all the models (weak learners).
10. Thus, the boosting algorithm combines a number of weak learners to form a strong learner.
11. The individual models would not perform well on the entire dataset, but they work well for some part of the dataset.
12. Thus, each model actually boosts the performance of the ensemble.

Very roughly, we can say that bagging will mainly focus at getting an ensemble model with less variance than its components whereas boosting will mainly try to produce strong models less biased than its components (even if variance can also be reduced).

XG-Boost:

- Stands for e**X**treme **G**radient **Boost**ing
- XG boosting only focuses on increasing the weights of the error data points

AdaBoost:

- Stands for Adaptive Boosting
- ADA boost focuses on increasing the weights of the error data points and at the same time decreasing the weight of the correctly classified point