Mithilesh singh

Logistic Regression: classification model

Logistic regression is a machine learning classification model with similar as Linear Regression

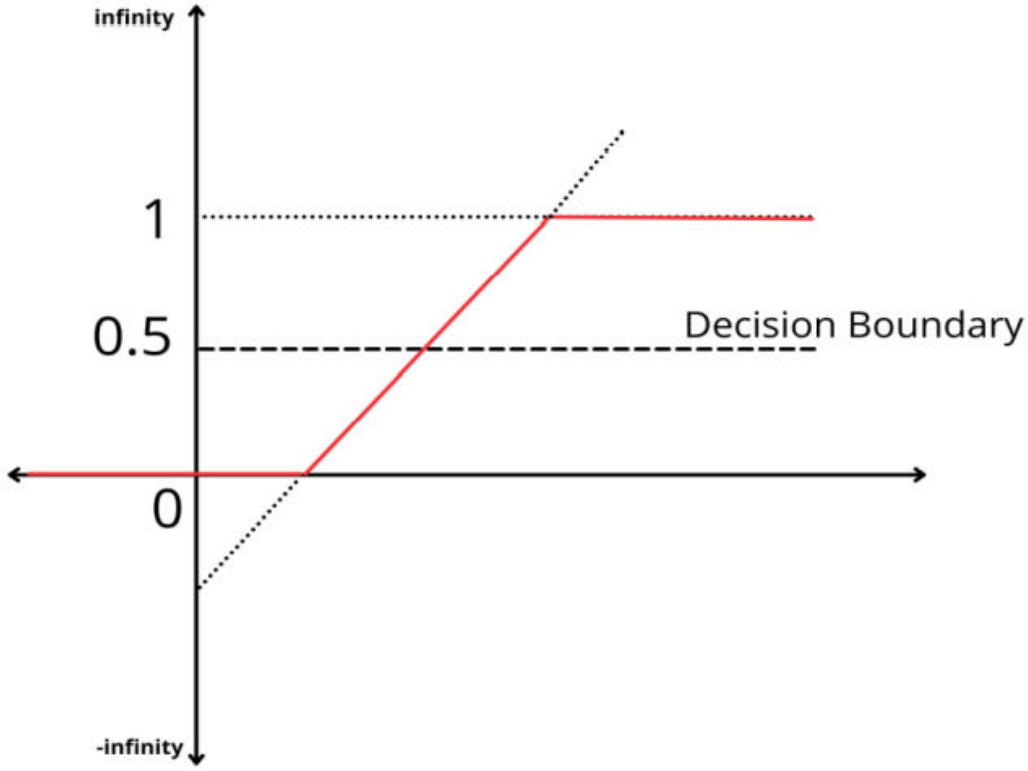
but it's not used to predict a continuous outcome. -Not for Regression.

Instead, it is a statistical BINARY classification model.

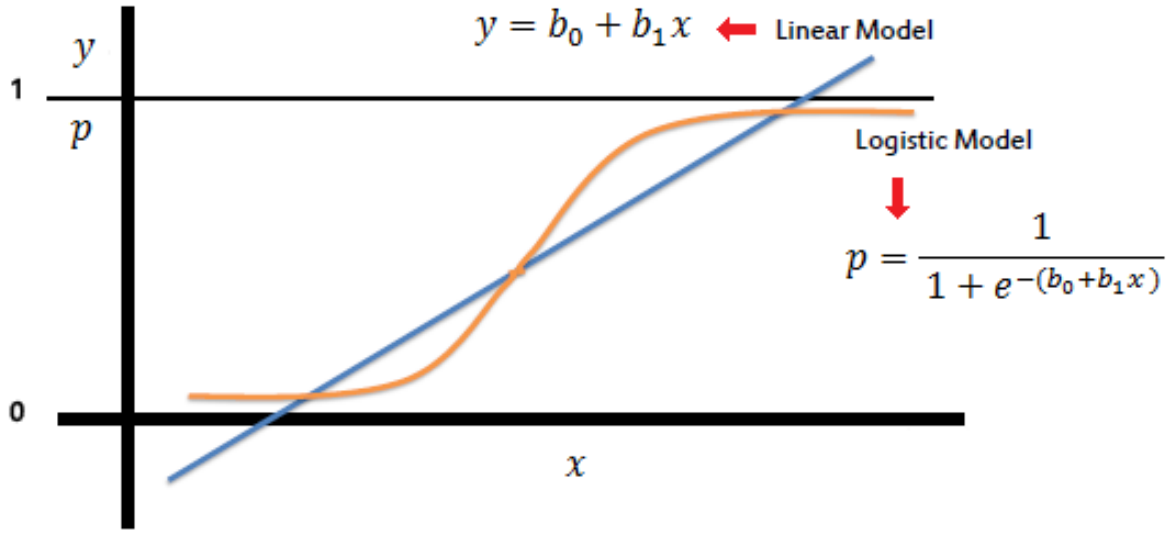
Binary classes defined as success and failure: -

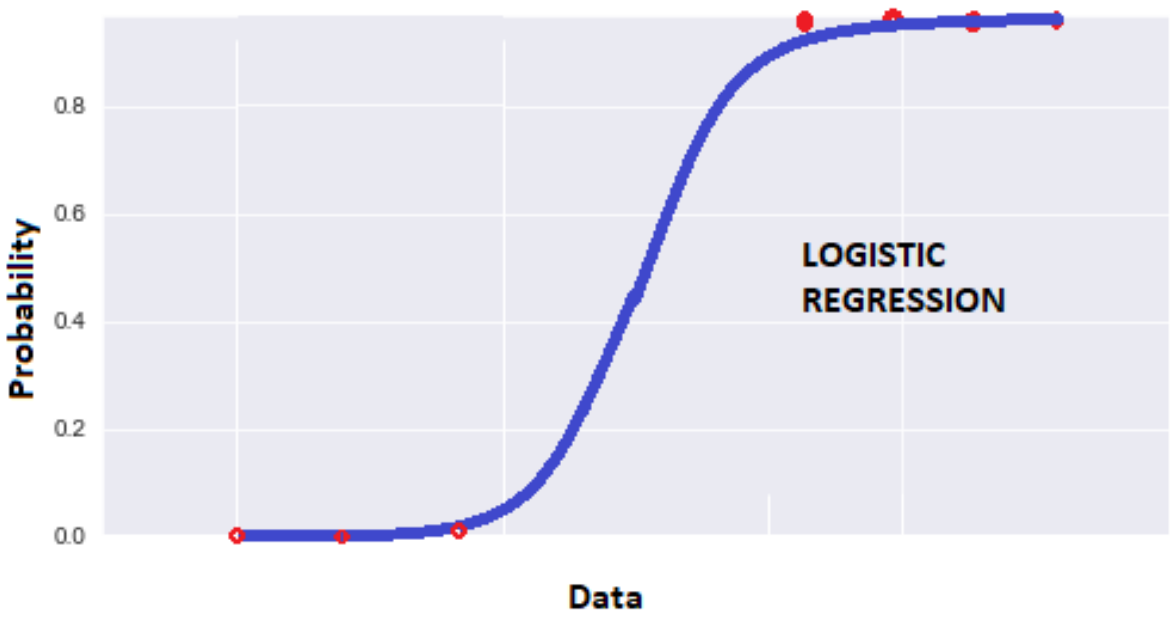
0 (Failure) and 1 (Success), Yes / No ,Selected / Rejected ,True/ False

Decision boundary:



The S-shaped curve is a sigmoid curve. Logistic regression function is also called sigmoid function.





p is a non-linear function, say, an exponential function.

Sigmoid Function: $\boldsymbol{p = \frac{1}{1 + e^{-y}}}$

For -∞

$$p = \frac{1}{1 + e^{-(-\infty)}}$$

$$p = \frac{1}{\infty}$$

Anything divided by ∞ is 0, Hence for -∞ it is **p = 0**

$$p = \frac{1}{1 + e^{-y}}$$

For ∞

$$p = \frac{1}{1 + e^{-(+\infty)}}$$

$$p = \frac{1}{1 + 0}$$

Hence for ∞ it **is p = 1**

logistic regression formula

Sigmoid function:-

Mathematical function having a characteristic "S"shaped curve or sigmoid curve.

$$p = \frac{1}{1 + e^{-y}}$$

Transformation of the sigmoid function

$$p(1 + e^{-y}) = 1$$

$$p + pe^{-y} = 1$$

$$pe^{-y} = 1 - p$$

$$e^{-y} = \frac{1 - p}{p}$$

$$log(e^{-y}) = log\left(\frac{1 - p}{p}\right)$$

$$-y = ln\left(\frac{1 - p}{p}\right)$$

$$y = ln\left(\frac{p}{1 - p}\right)$$

Hence we have converted the sigmoid function such that now it is expressed as providing the value of Y that is the target variable

$$y = ln\left(\frac{p}{1-p}\right)$$

Here (P/ (1-P)) is called "odds ratio".

Hence Y can be defined as the log of the odds ratio.

Y is also called as Log Odd Function or Logistic Function or Log it Function

That's why the name Logistic Regression and not Logistic Classification? Because Logistic Regression model outputs probabilities (or log odds ratios in the logit form) that have a linear relationship with the predictor variables.

Applying Linear Equation Formula

$$exp\left(log\left(\frac{p}{1-p}\right)\right) = exp\left(a + b_1 x_1\right)$$

$$\Rightarrow \frac{p}{1-p} = exp\left(a + b_1 x_1\right) \quad [Since\ exp(logx) = x]$$

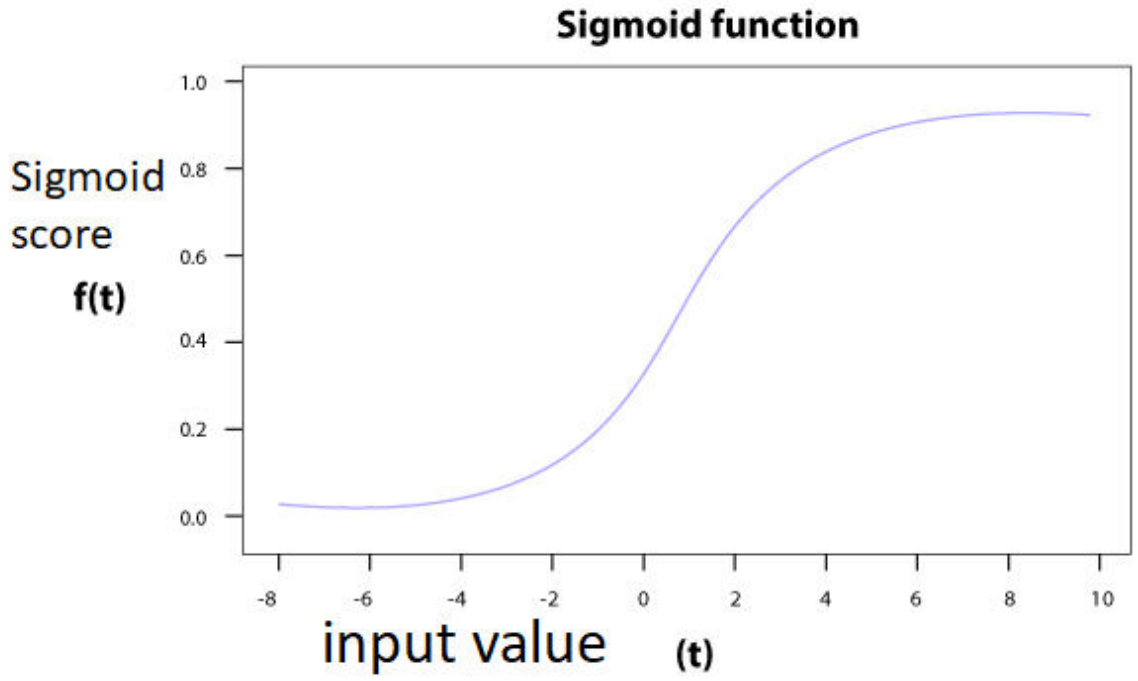
Solving the above equation for p, we get

$$p = exp\left(a + b_1 x_1\right) - p\ exp\left(a + b_1 x_1\right)$$

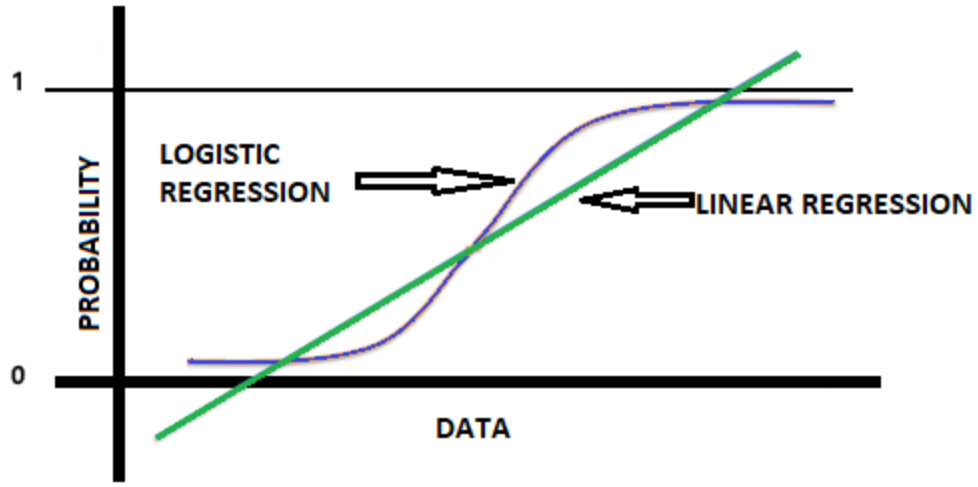
$$p = \frac{exp(a+b_1 x_1)}{1+exp(a+b_1 x_1)}$$

Finally, we will get the value of probability p in the range of 0 to 1.

Keeping Threshold =0.5
Above 0.5 output belongs to class 1
Below 0.5 output belongs to class 0



Why Logistic Regression over Linear Regression?



Difference between Logistic and Linear Regression

When it comes to binomial classification (0/1), we need to create a boundary between the values that are classified as 0 or 1. In linear regression, we know that the output is a continuous variable, so drawing a straight line to create this boundary seems infeasible as the values may go from $-\infty$ to $+\infty$.

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train1,Y_train)

Y_pred_lr_train=lr.predict(X_train1)
Y_pred_lr_test=lr.predict(X_test1)

print("LOGISTIC REGRESSION TRAINING ACCURACY ",accuracy_score(Y_train,Y_pred_lr_train))

print("######"*20)

print("LOGISTIC REGRESSION TESTING ACCURACY ",accuracy_score(Y_test,Y_pred_lr_test))
```

MATHEMATICS of Logistic Regression: -

Maximum Likelihood Estimation

logistic regression formula
Sigmoid function:-
Mathematical function having a characteristic "S"shaped curve or sigmoid curve.

$$p = \frac{1}{1 + e^{-y}}$$

The logistic regression function converts the values of **logits** also called **log-odds** that range from **−∞ to +∞** to a range between **0 and 1**.

Transformation of the sigmoid function

$$p(1 + e^{-y}) = 1$$

$$p + pe^{-y} = 1$$

$$pe^{-y} = 1 - p$$

$$e^{-y} = \frac{1-p}{p}$$

$$log(e^{-y}) = log\left(\frac{1-p}{p}\right)$$

$$-y = ln\left(\frac{1-p}{p}\right)$$

$$y = ln\left(\frac{p}{1-p}\right)$$

Odds is defined as the ratio of the probability of occurrence of a particular event to the probability of the event not occurring.

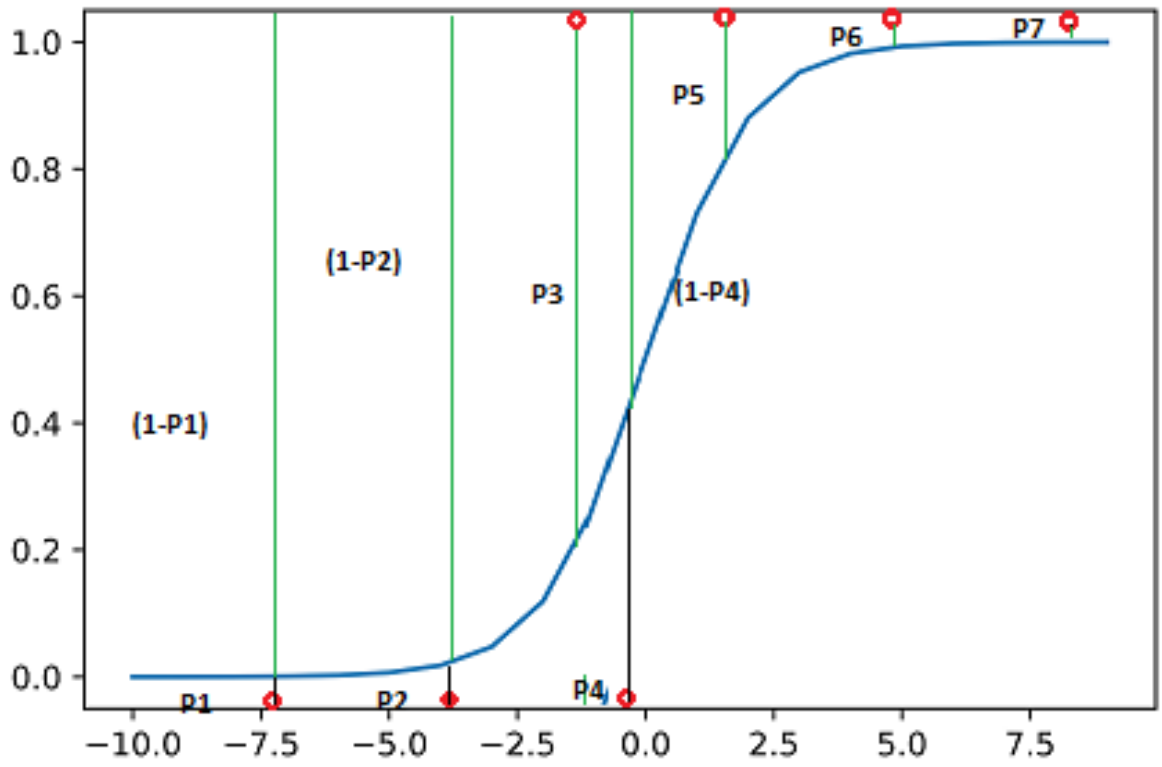
$$Odds = \frac{P}{1-P}$$

$$\ln(Odds) = \ln\left(\frac{P}{1-P}\right) = y$$

$$\ln\left(\frac{P}{1-P}\right) = \beta 0 + \beta 1x$$

Cost function tells how close the values are from actual. Such a cost function is called as **Maximum Likelihood Estimation (MLE)** function.

Maximum Likelihood Estimation (MLE) function is to find the best fit sigmoid curve, that gives the optimum values of beta co-efficient.



Sigmoid curve and probabilities

From the above figure, there are TRUE Value of 7 points and seven associated probabilities P1 to P7, classified as class 0 and 1.

Here True value of P1,P2,P4 are class 0. We need the probabilities of P1, P2 and P4 to be as minimum as possible w.r.to class 1.

and for points p3,p5,p6& p7 the TRUE value is class 1.We need the probabilities of P3, P5, P6 and P7 to be as high as possible w.r.to class 1 , for correct classification.

We can also say that (1-P1), (1-P2), P3, (1-P4), P5, P6 and P7 should be as high as possible.
The joint probability is nothing but the product of probabilities.

Joint Probability = **(1-P1)*(1-P2)* P3*(1-P4)*P5*P6*P7** should be maximum.

This joint probability function is our cost function or **the Maximum Likelihood Estimation (MLE)** function.

This Cost function should be maximized in order to get a best fit sigmoid curve. Or we can say predicted values to be close to the actual values.

EXAMPLE:-

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

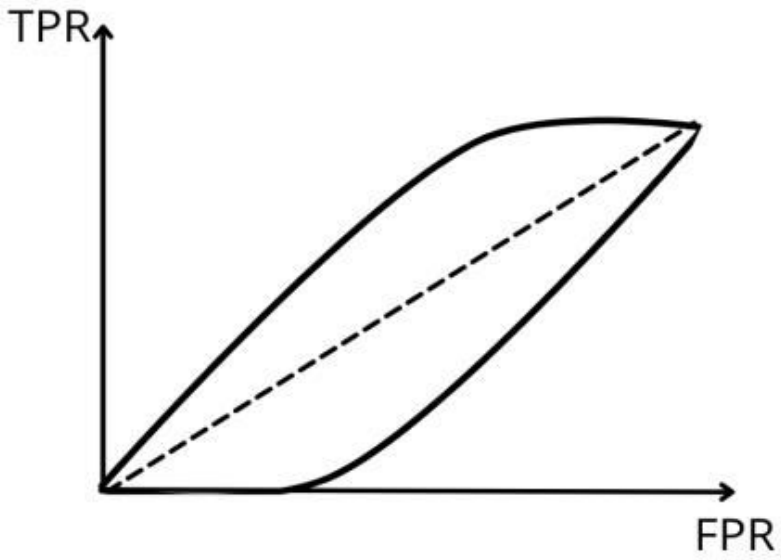
- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (10+5)/165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes = 100/105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no = 10/60 = 0.17
- **True Negative Rate:** When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"

- **Precision:** When it predicts yes, how often is it correct?
    - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
    - actual yes/total = 105/165 = 0.64

ROC AUC

ROC AUC Curve

- ROC means Receiver Operating Characteristics.
- This was initially used by operators of the military radar in 1941, World war II
- AUC taken from Economics world.



ROC curve is a graph plotted between TPR and FPR

ROC (Receiver operating characteristic) curve

Area under the curve

The Area under the Curve (AUC) is a summary of the ROC curve and is a measure of a classifier's ability to discriminate between classes.

The higher the area the better the performance of the model to distinguish between the positive and the negative cases.

The confusion matrix helps us visualize whether the model is "mistaken" in distinguishing between two classes.

Positive or **Negative** are the names of the predicted labels by the ML model. Whenever the prediction is wrong, the first word is **False**, and when the prediction is correct, the first word is **True**.

ROC curve is built upon two metrics that are arrived from the confusion matrix: true positives rate (**TPR**) and false positive rate (**FPR**). **TPR** is the same as recall. It is the ratio of the correctly predicted positive samples divided by all actual positive samples available from the dataset. TPR focuses on the actual positive class:

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

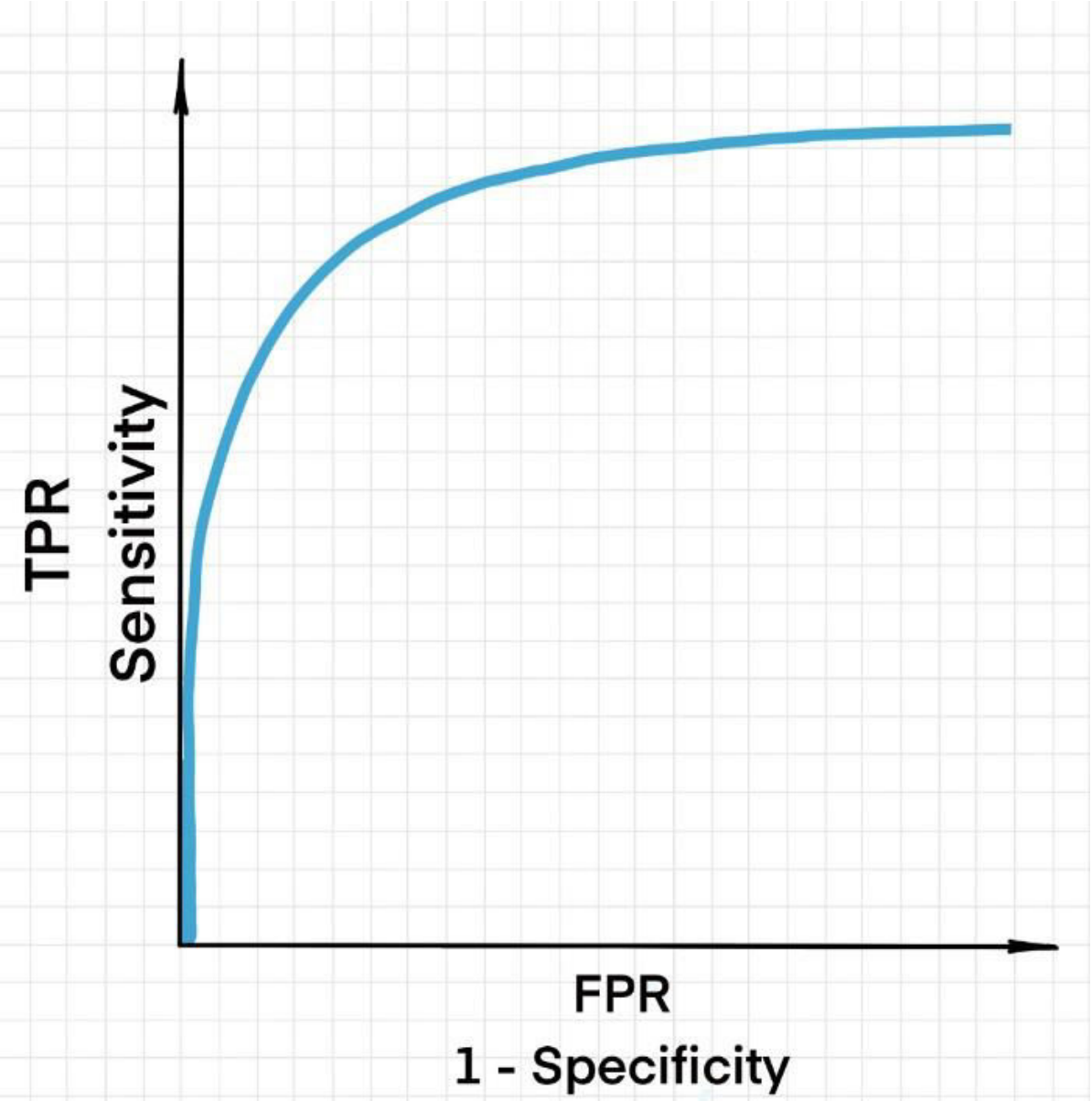
True Positive Rate formula

In turn, FPR is the ratio of false positive predictions to the total number of actual negative samples. It is the same as *1 — Specificity*:

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

FPR formula

The ROC Curve is plotted based on TPR and FPR.

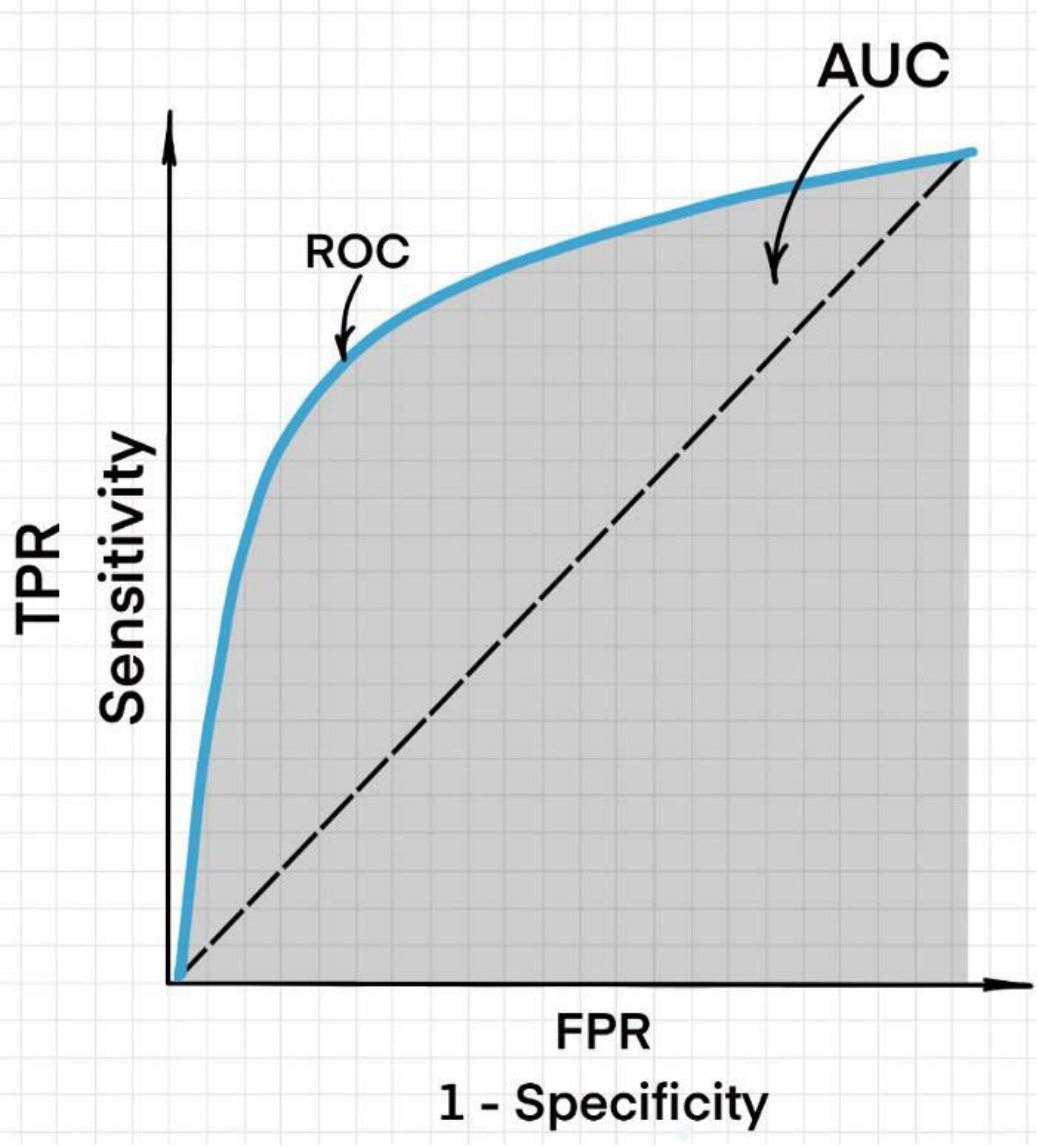


ROC curve example

By using TPR and FPR, the ROC Curve shows your classification model's performance at all classification thresholds.

By default, classification threshold is 0.5. Any probability above 0.5 will be classified as class 1 (positive) and below 0.5 will be classified as class 0 (negative).

So, ROC, in turn, tells us how much the ML model is capable of distinguishing between two classes at various thresholds.



Mithilesh Singh

KNN regression/Classifier

-non-parametric method

- approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighborhood*.

-The size of the neighborhood K value needs to be set by the analyst or can be chosen using cross-validation to select the size that minimizes the mean-squared error.

Parametric V/s Non-parametric method

Parametric Method-assume the data is of sufficient "Quality", exp :- Linear Regression, Logistic Regression.

- **The result can be misleading if assumptions are wrong.**
- **Quality is defined in terms of certain properties of the data, like Normally distributed ,Symmetrical linear distribution, homogeneity of variance etc.**

Non Parametric tests can be used when the data is not of sufficient quality to satisfied the assumptions of parametric test. Non parameter tests STILL have assumptions but are less stringent.

Non parameter tests can be applied to Normal distributed data but parametric tests have greater power IF assumptions met.

- Parametric tests are preferred when the assumptions are met because they are more sensitive.

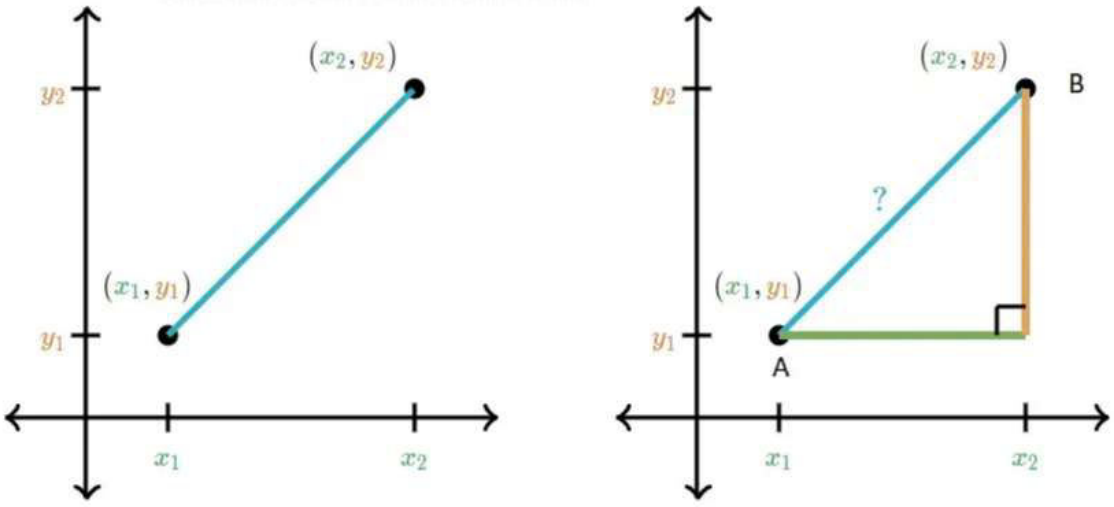
KNN model classifies the points based on proximity or distance.

Important Distance Metrics in Machine Learning

- **Euclidean Distance**
- **Manhattan Distance**
- **Minkowski distance**

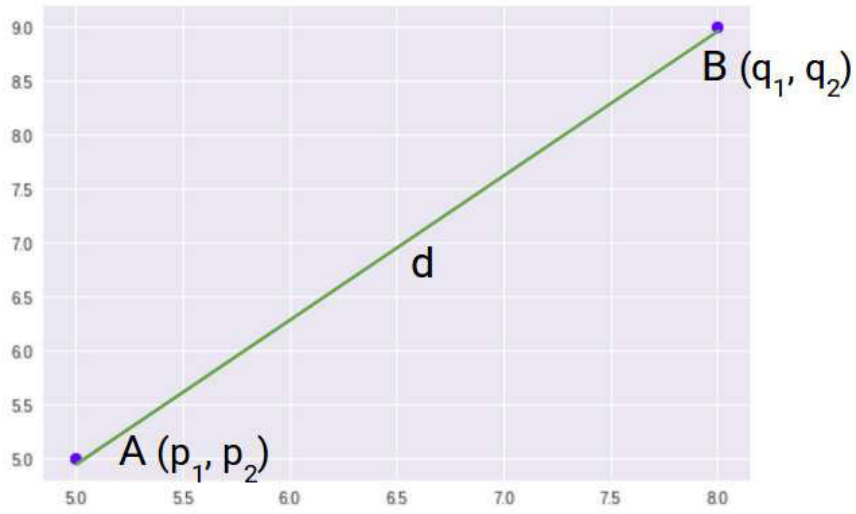
1. Euclidean Distance

Euclidean Distance represents the shortest distance between two points.



Pythagorean theorem

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



The 2 dimensions' formula for Euclidean Distance:

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

For n-dimensional space as:

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

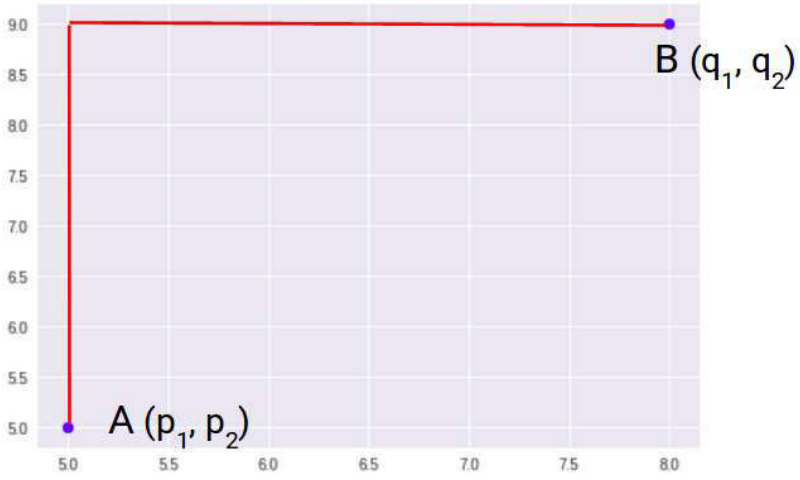
Where,

- n = number of dimensions
- pi, qi = data points

2. Manhattan Distance

Manhattan Distance is the sum of absolute differences between points across all the dimensions. Also called L1 Norm ,Taxi Cab Norm.

We can represent Manhattan Distance as:



Manhattan Distance, sum of absolute distances x and y directions.

In a 2-dimensional space is given as:

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

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

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

Where,

- n = number of dimensions
- pi, qi = data points

(3)Minkowski distance:

This distance measure is the generalized form of Euclidean and Manhattan distance metrics.

Euclidean distance is represented by this formula when p is equal to 2, and Manhattan distance is denoted with p equal to 1.

Minkowski distance=

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

→ KNN model used for classification (and regression).

→ → KNN uses distance metrics in order to find similarities or dissimilarities.

- working off the assumption that similar points can be found near one another.
- The distinction between these terminologies is that "majority voting"

Example of KNN- Euclidean distance in real life

Liability→ Person	a	b	c	d	Status (loan approved)
1	160	10	20	2000	No
2	180	20	30	5000	Yes
3	200	30	35	8000	Yes
4	150	20	25	4000	No
5	350	60	100	6500	Yes

--					
100	200	50	80	7500	Yes
A	175	35	30	4500	?

Euclidean distance: -

A-1→square-root(15 sq +25 sq+10 sq +2500 sq)

A-2→ square-root (5 sq+15 sq+ 0 sq+500 sq)

A-3→ square-root (25 sq+5 sq+5 sq+3500 sq)

A-4→ square-root (25 sq+15 sq +5 sq+ 500 sq)

A-5→ square-root (175 sq+25 sq+70 sq+2000 sq)

A-100→……

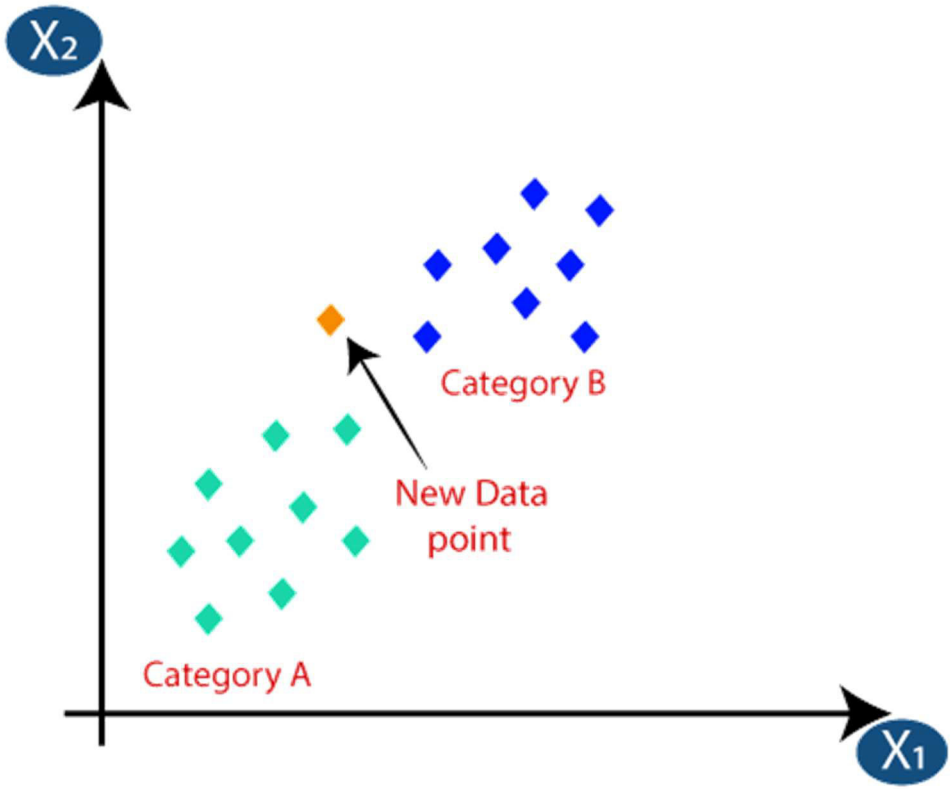
Nearest neighbor →lowest distance…

KNN method working concept:-

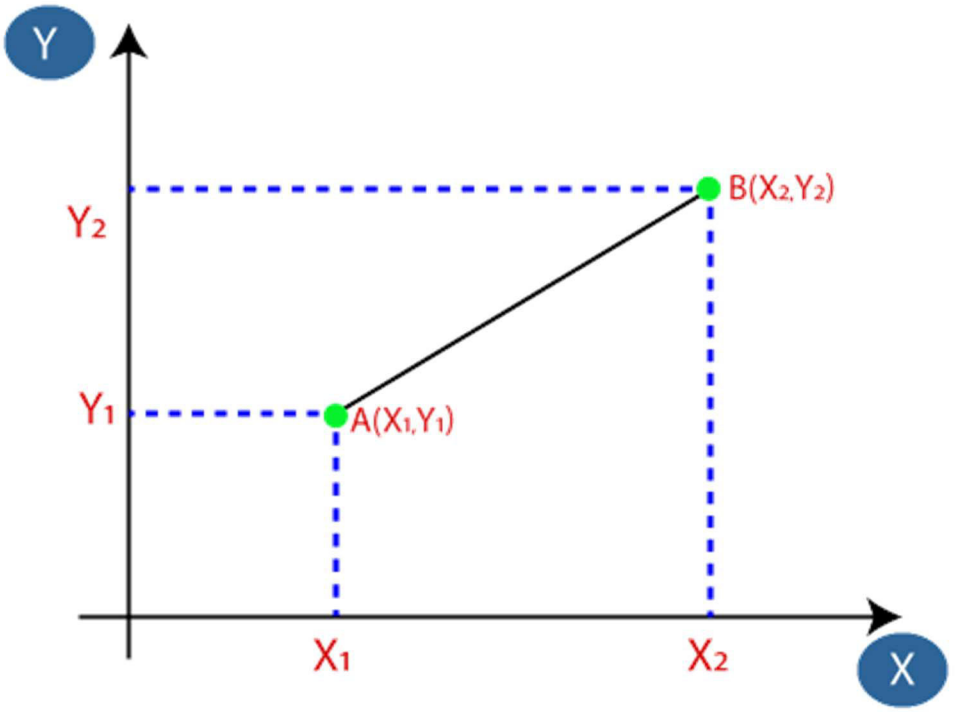
The K-NN working can be explained on the basis of the below algorithm:

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

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

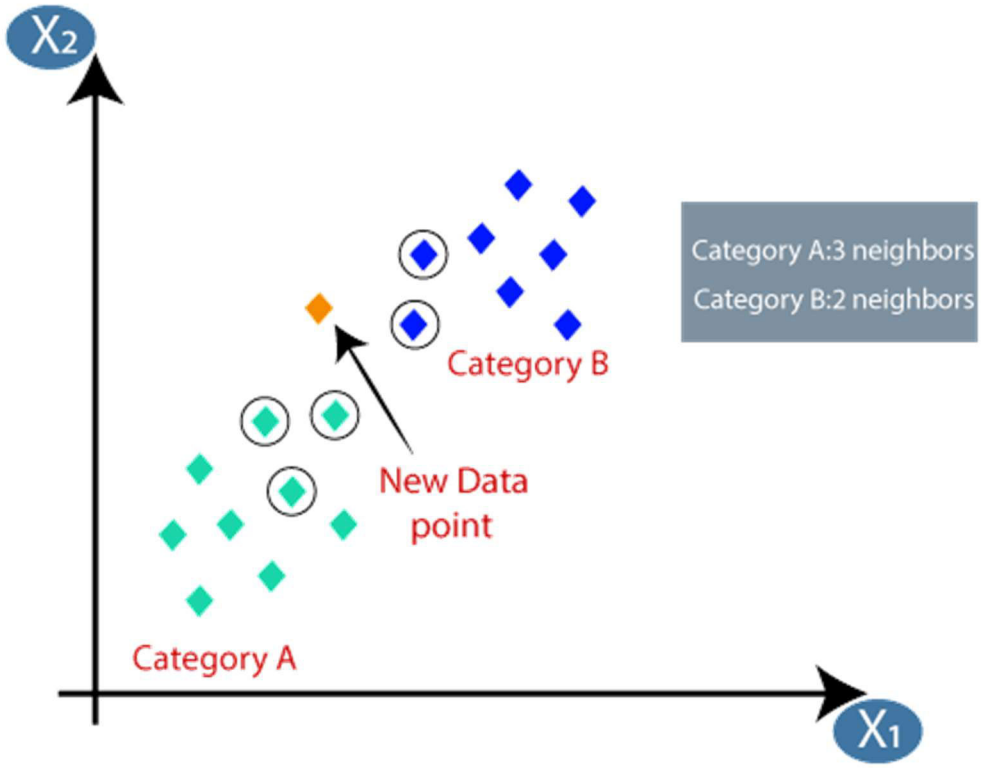


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



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

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



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

- KNN can be used in regression, the target continuous value is computed as the mean of the target value of k nearest neighbors.

For classification Problems:-

from sklearn.neighbors import KNeighborsClassifier

FOR Regression Problems: -

from sklearn.neighbors import KNeighborsRegressor

How to select the optimal value of k?

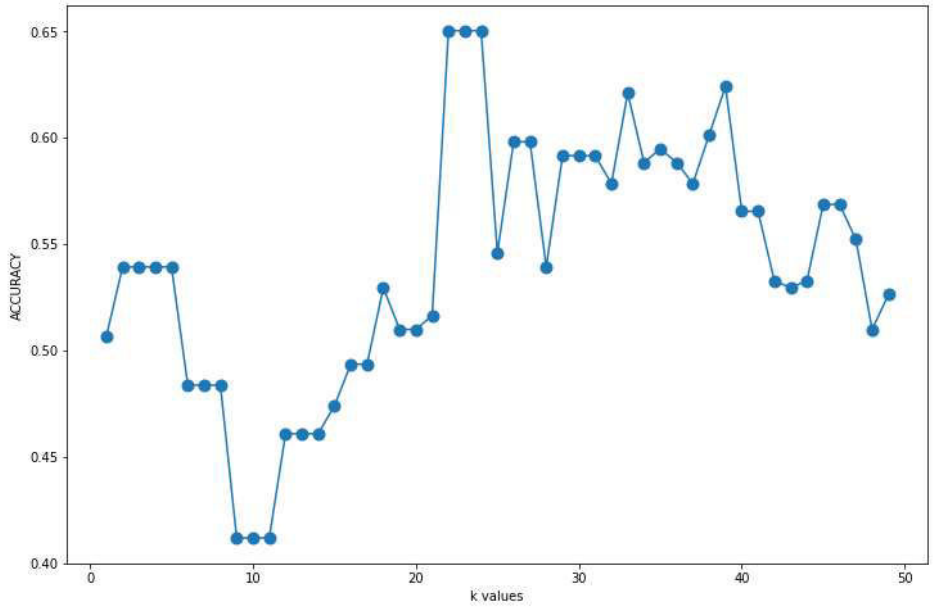
- Prefer odd k values, as there are chances of tie in even values.
- K should not be too small.
- Thumb rule: k should be generally sqrt(n), where n denotes the total number of data points
- Further, one can try different values of k, and then observe the evaluation metrics to decide upon the best value of k

Python code:-

```
from sklearn.neighbors import KNeighborsClassifier
model_name = ‘K-Nearest Neighbor Classifier’
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric =
‘minkowski’, p=2)
knn_model = Pipeline(steps=[(‘preprocessor’,
preprocessorForFeatures), (‘classifier’ , knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

 # find the more effective value of n_neighbors parameter k value:

```
accuracy_K = []
for k in range(1, 50):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train1, Y_train)
    Y_pred = knn.predict(X_test)
    accuracy =accuracy_score(Y_test, Y_pred)
    accuracy_K.append(accuracy)
plt.figure(figsize=(12,8))
plt.xlabel("k values")
plt.ylabel("ACCURACY")
plt.plot(range(1,50),accuracy_K, marker='o', markersize=9)
```



KNN is a Lazy Learner model

- Almost all the models get trained on the training dataset, but KNN does not get

trained on the training dataset.

- When we use knn.fit (X train, Y train), this model 'memorizes' the dataset. It does not understand it or tries to learn the underlying trend.
- Now when we ask the model to predict some value, then it takes a lot of time because now it actually will have to recall all the points and work around them so that it can help predict the correct value.
- Hence where most of the models, take time during training, this model does not take any time during training.
- Most of the models take no time in prediction, but the KNN model takes a lot of time during the prediction stage.

Important points

- **Since it is a distance-based model, feature scaling is a must for it.**
- **Besides logistic regression, the rest all the classification models can work on multi class classification.**

Advantages of KNN Algorithm:

- It is simple to implement.
- When the label data is too expensive or impossible to obtain.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- Entire dataset is processed for every prediction, not good for large dataset.

The computation cost is high because of calculating the distance between the data points for all the training samples. computational Time complexity for each prediction is equal to MNlog(k)

M is the dimension of data, N is the size of instance of training data and K is nearest point selected.

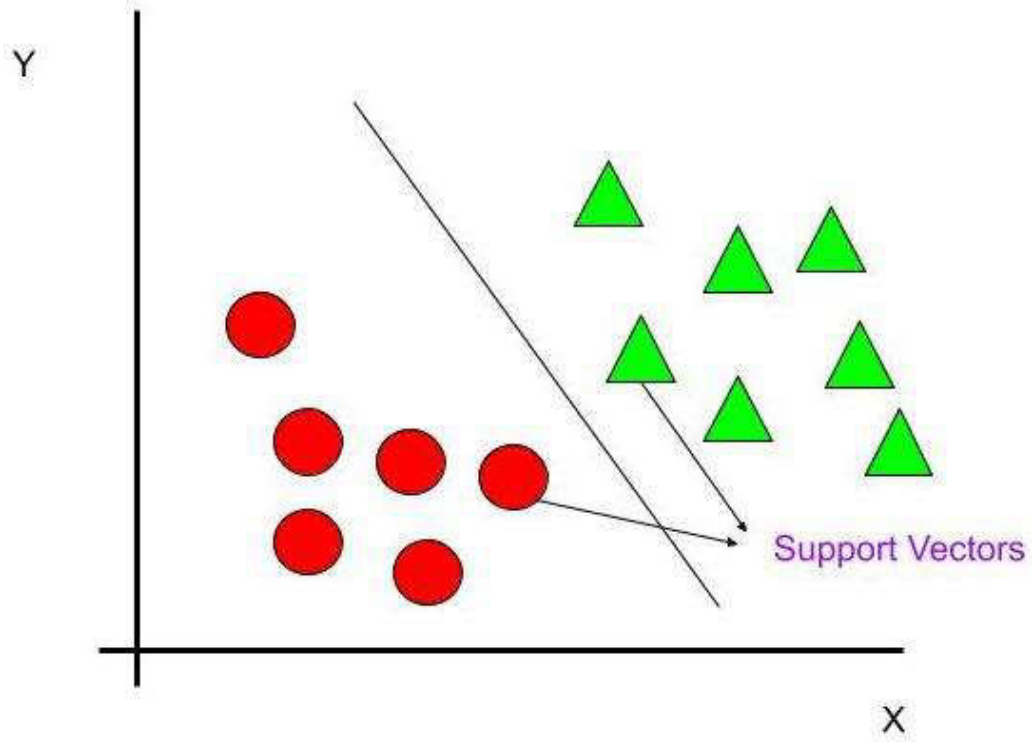
Support vector machines (SVM)

1990→AI called winter year. USA had invested huge amount in military but not successful due to failure of AI.

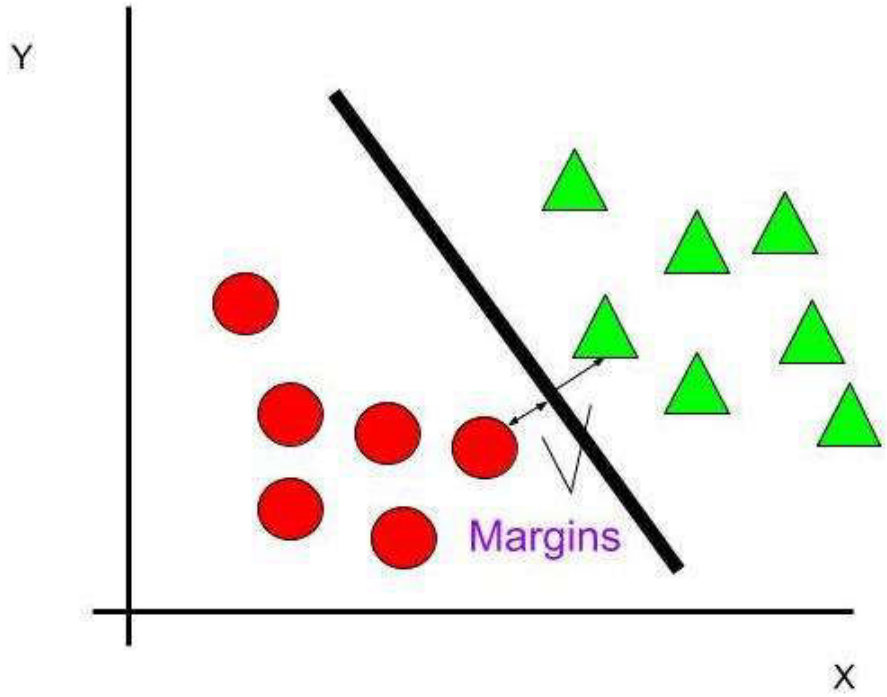
SVM →During that period SVM Invented →popular & successful More than 200 paper published.

→SVM Most popular, Versatile MLA Used in Regression & Classification

- Preferred for Medium and small sized dataset.
- It separates data in two components using hyper plane by maximizing the margin (Also called large marginal classifier).



- Support Vectors are simply the coordinates of individual observation.
- Dotted line is hyperplane, separating blue and pink classes balls.



Hyper-plane

It is plane that linearly divide the n-dimensional data points in two components.
In case of 2D, hyperplane is line,

in case of 3D it is plane.

It is also called as *n-dimensional line*.

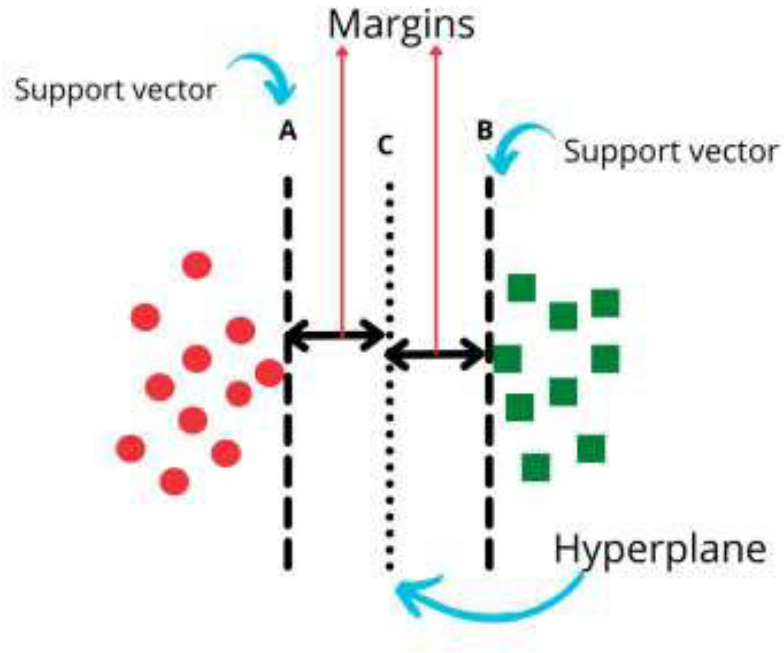
hyperplane line written as

$$y = a * x + b$$

$$a * x + b - y = 0$$

Let vector $X=(x,y)$ and $W=(a,-1)$ then in vector form hyperplane is

$$W . X + b = 0$$



Hyperplane: A-line classify with highest margin .

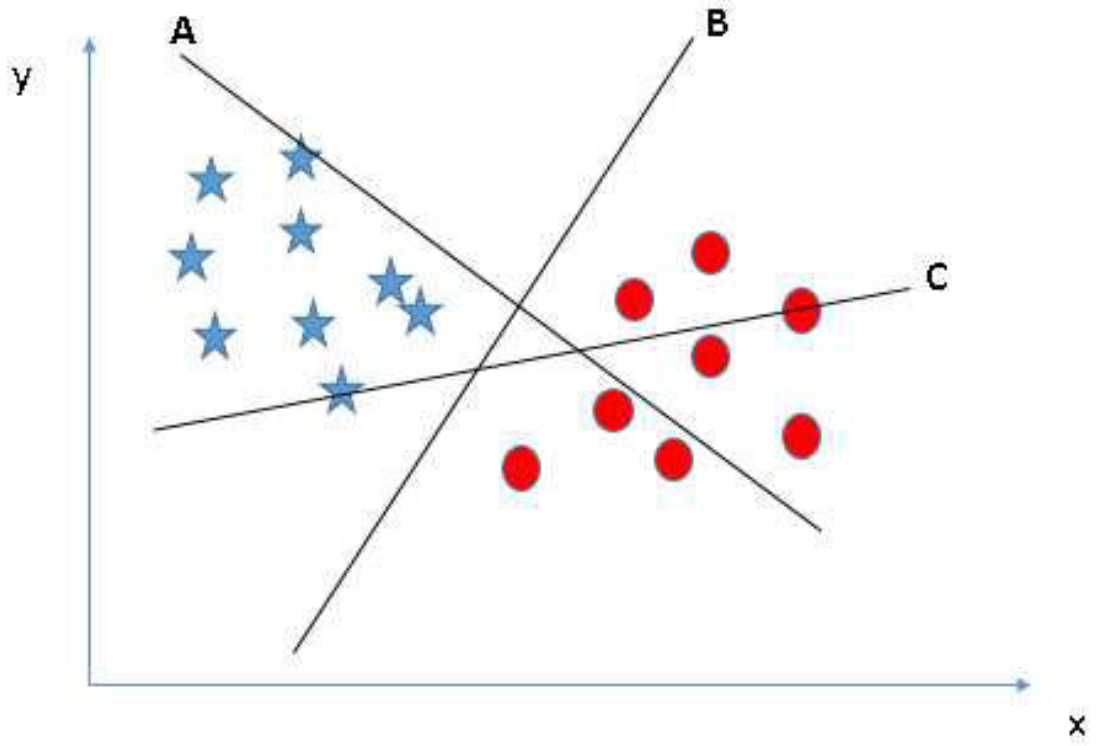
Kernel →means approach

There are different kernel functions available:

- linear
- Gaussian (RBF kernel- Radial Basis Function)
- Polynomial
- Sigmoid

(Scenario-1)

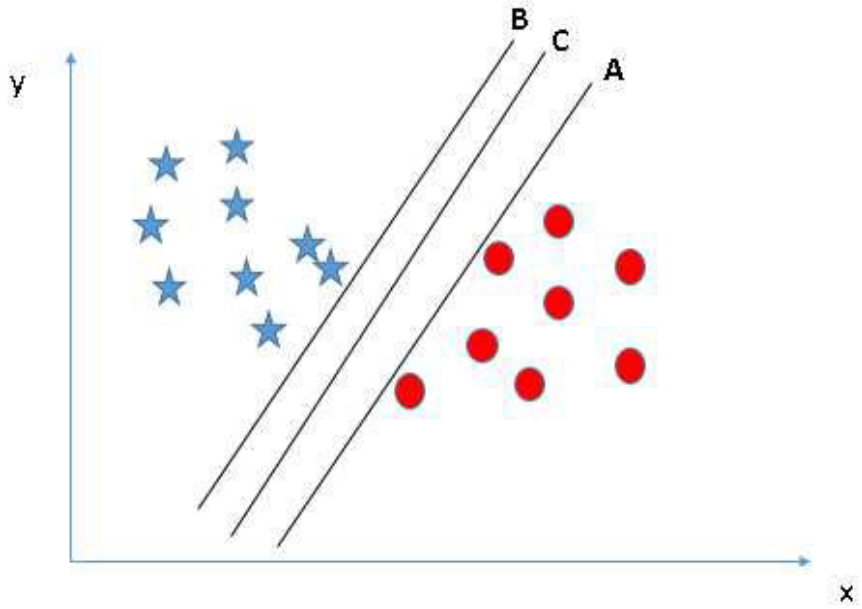
Identify the right hyper-plane: Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



:- “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

(Scenario-2)

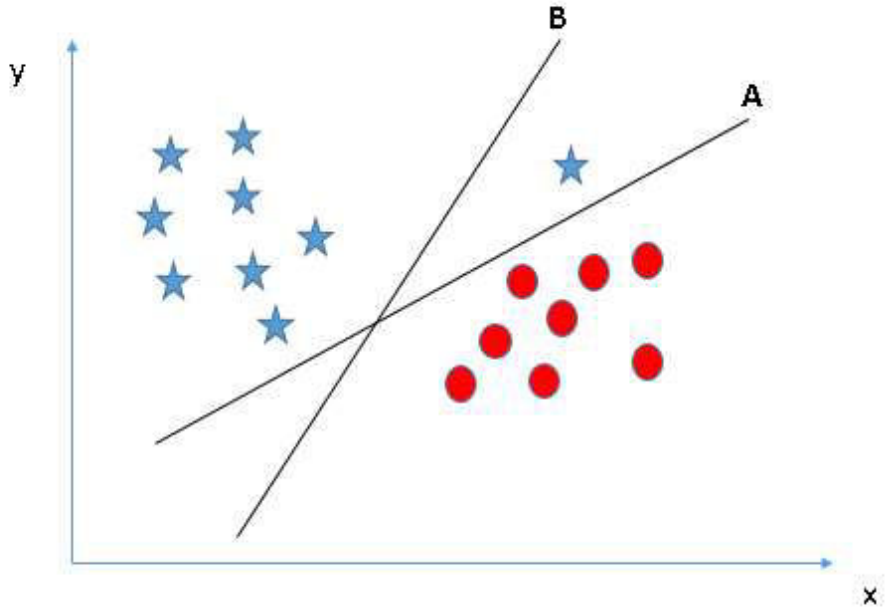
- **Identify the right hyper-plane:** Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



Here, maximizing the **Margin**. the right hyper-plane as C. If we select a hyper-plane having low margin then there is high chance of miss-classification.

(Scenario-3)

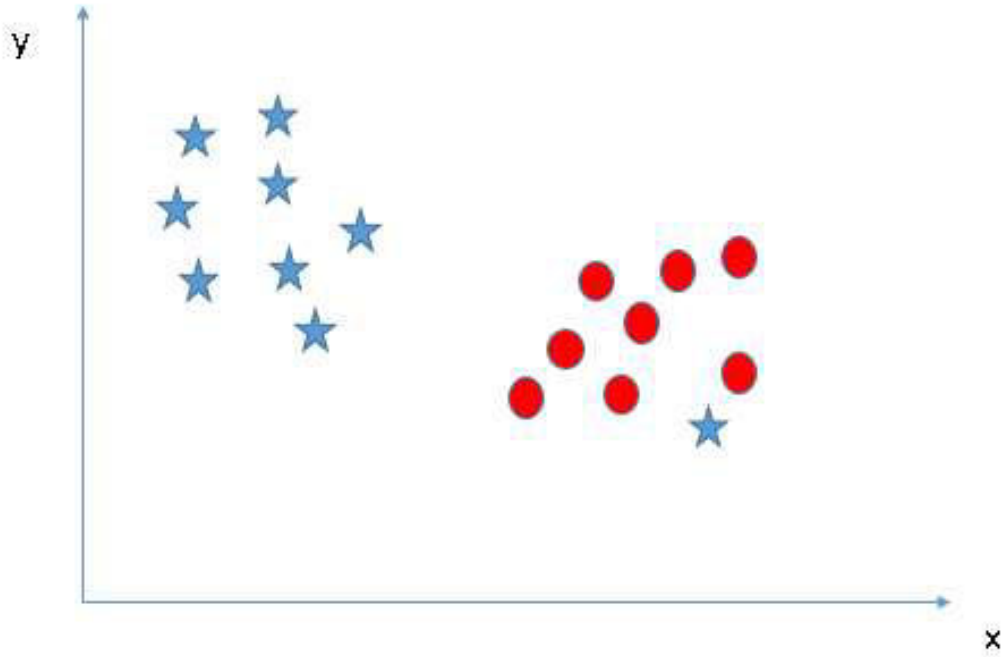
- **Identify the right hyper-plane:** Hint: Use the rules as discussed in previous section to identify the right hyper-plane



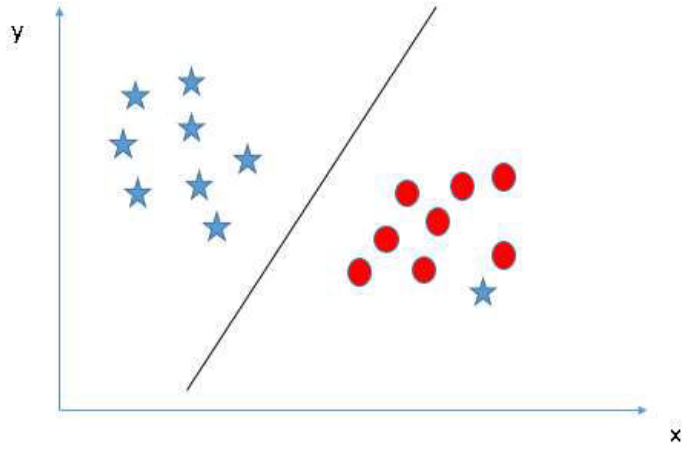
Note:- hyper-plane **B** as it has higher margin compared to **A.**

But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

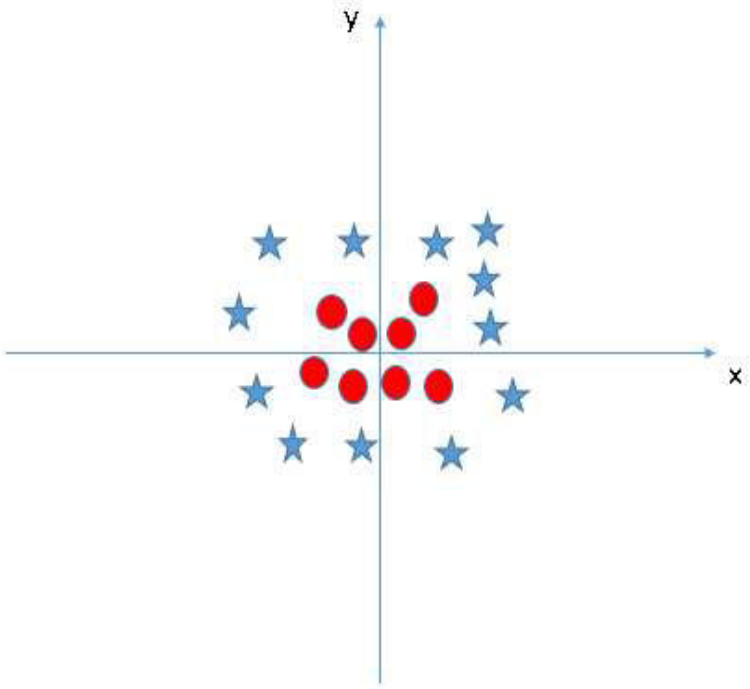
(Scenario-4)



The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, SVM classification is robust to outliers



(Scenario-5)
Find the hyper-plane to segregate to classes: In the scenario below, we can't have linear hyper-plane between the two classes.

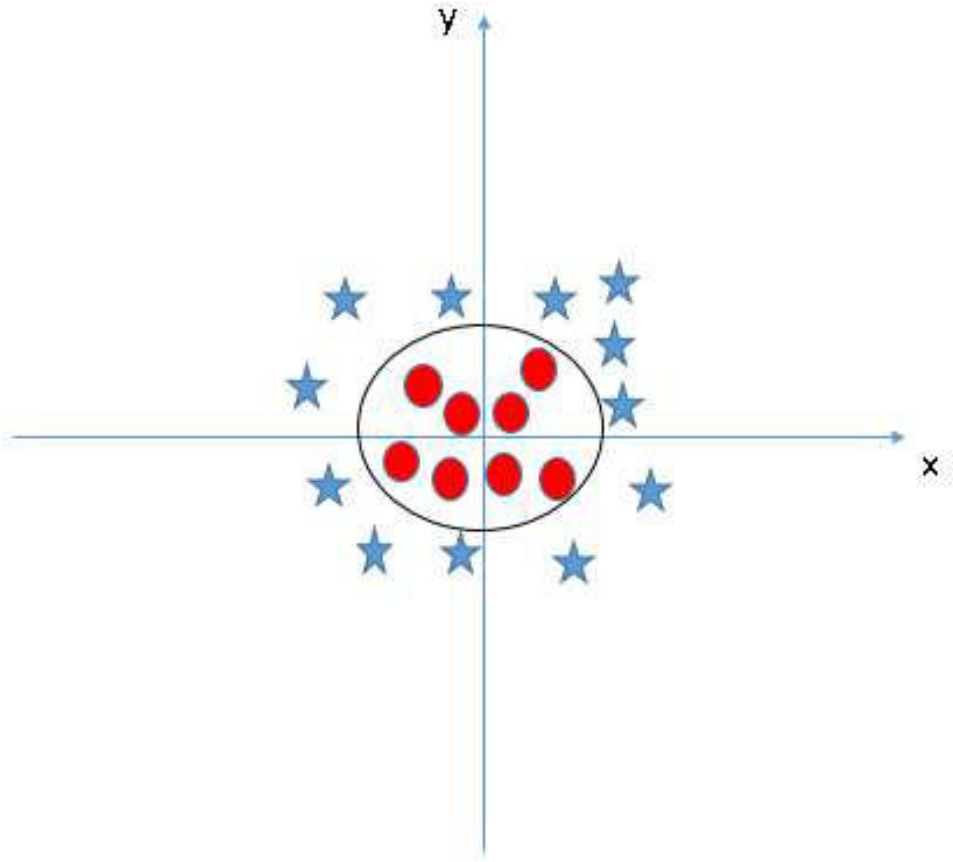


If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line..

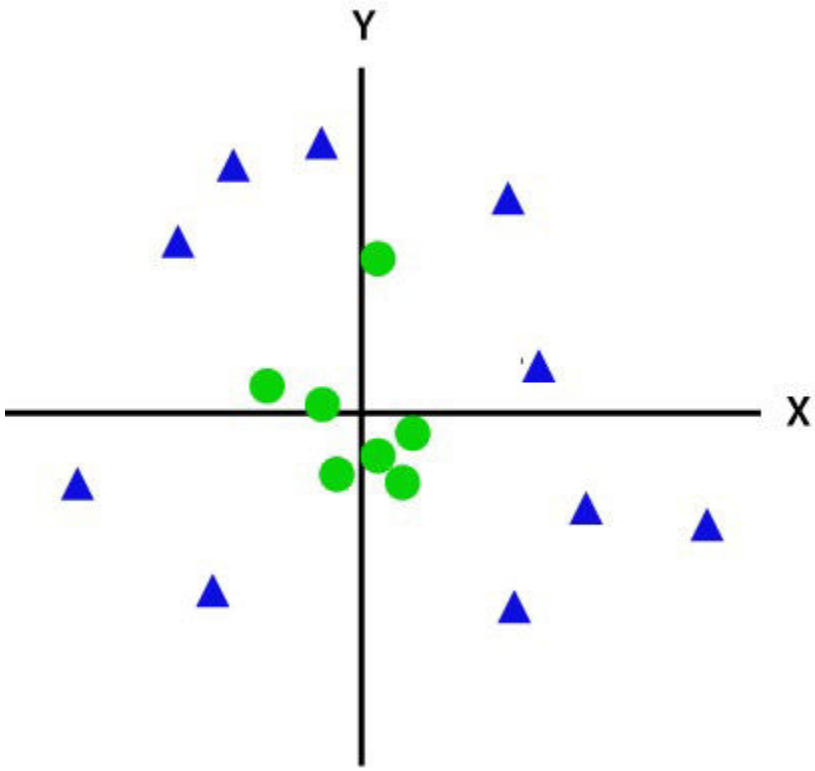
It is possible to solve in SVM *!!*. It will add one extra dimension to the data points to make it separable.

Kernel trick CONVERTING TO HIGH DIMENSIONALITY

It keep on increasing the dimensions unless the classes are separable.



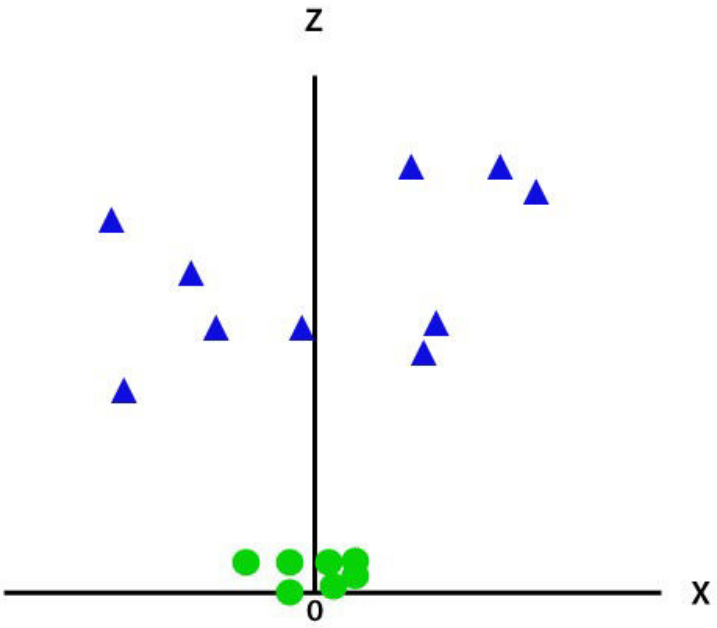
Concept Behind Non Linear Separation:

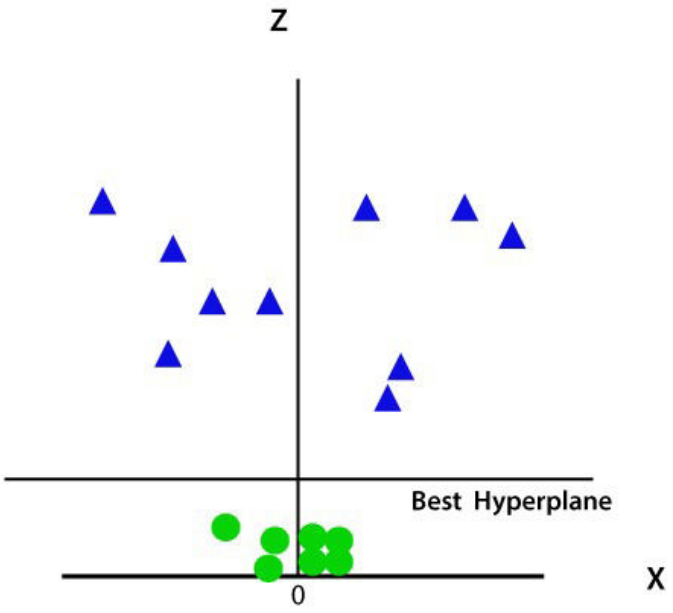


So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions' x and y, so for non-linear data, we will add a third dimension z. **It can be calculated as:**

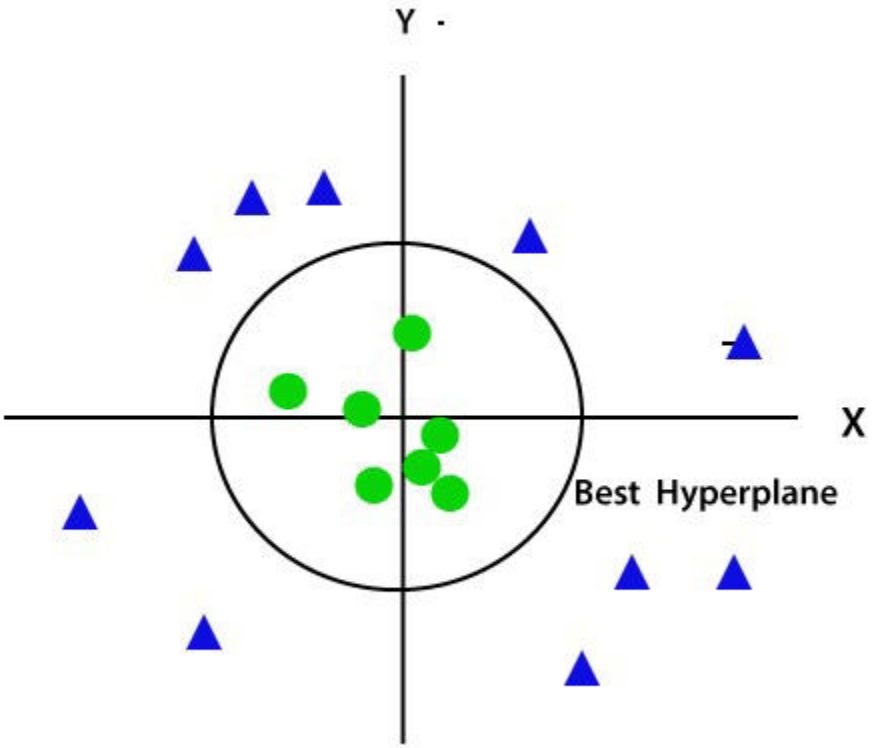
$$z=x^2 +y^2$$

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





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.

Kernel functions. - transform non-linear spaces into linear spaces. It transforms data into higher dimension so that the data can be classified.

There are various options associated with SVM training; like changing kernel, gamma and C value.

kernel: various options available - “linear”, “rbf”,”poly” and sigmoid (default is “rbf”).

Here “rbf” and “poly” are useful for non-linear hyper-plane.

gamma: Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem.

gamma different gamma values like 1, 10 or 100.

C: Penalty parameter C of the error term. It also controls the trade-off between smooth decision boundaries and classifying the training points correctly.

Pro & Cons associated with SVM: -

Advantage: -

- Works really well with margin of separation.
- Effective in high dimensional space
- Accurate results.
- Useful for both linearly separable and non-linearly separable data.

Disadvantages: -

- It doesn’t perform well when we have large dataset because the required training time is very high.

Applications of SVM

- **Sentiment analysis**
- **Spam Detection**
- **Handwritten digit recognition**
- **Image recognition**

```
from sklearn.svm import SVC #"Support vector classifier"
from sklearn.svm import SVR #"Support vector Regressor"

 # Building a Support Vector Machine on train data
svc_model = SVC(C=1, kernel='linear', gamma= 100)
svc_model.fit(X_train, Y_train)
 prediction = svc_model.predict(X_test)
```