Classification Tree

Classification:

Classification is the action of classifying something according to shared qualities or charateristics.

In classification World....

→ To explain is called as profiling.

→ To predict is called as classifying.

Exp	state	Qualification	Hire
5	TN	B.Tech	Yes
5	AP	B.Tech	Yes
10	UP	Metric	No
5	UP	PhD	No
7	TN	B.Teh	Yes

→ profiling and Training

New Incoming data

Exp	state	Qualification	Prediction
5	TN	B.Teh	Yes
8	UP	Metric	No

→ classifying and prediction

{Independent variable} → {Predictor, dimensions, columns, features, Input, $x_1 - x_2 - x_3 - x_4 - \ldots$}

{Dependent variable} → {Predected, output, Target, Y}

Popular classification Techniques:

* CHAID → CHi-squared Automatic Interaction Detector. CHAID is not a binary classification tree. It is used for categorical variables.

* CART → Classification And Regression Tree. CART is a binary decision tree.

* C4.5 → It uses the concept of information entropy.

→ Both CART and C4.5 can handle categorical and numerical data. CART is most widely used algorithm.

CART: - (classification and regression tree):

CART is a classification algorithm that uses GINI as the measure of impurity.

GINI Index is the measure of impurity

Gini Impurity / Gini index: It tells us the probability of mis-classification of an observation.

Gini at a node:

$$Gini(t) = 1 - \sum_j [P(j|t)]^2$$

{P(j|t): is the relative frequency of class j at node t.}

Gini of a split:

$$Gini_{(split)} = \sum_{i=1}^{k} \frac{n_i}{n} \cdot Gini(i)$$

{n_i - no. of records at child i, n - Total no. of records in parent node}

Gini gain = Gini(t) − Gini(split)

(x) → Lower the Gini (split) better the split will be. This means lower the chances of misclassification.

(x) → We can also say higher the gini gain greater the split priority will be.

Calculation:

Gender	Occupation	Age	target
M	salary	22	1
M	salary	22	0
M	self-employed	23	1
M	self-employed	23	0
M	self-employed	24	1
M	self-employed	24	0
F	salary	25	1
F	salary	25	0
F	salary	26	0
F	self-employed	26	0

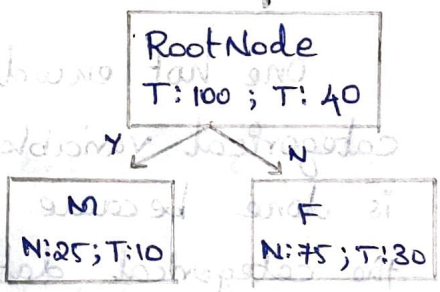
Node Gender	Gini computation Formula	Gini Index
overall	$1 - \left[(4/10)^2 + (6/10)^2\right]$	0.48
Gender (M)	$1 - \left[(3/6)^2 + (3/6)^2\right]$	0.5
Gender (F)	$1 - \left[(1/4)^2 + (3/4)^2\right]$	0.375
Gender (split)	$\left[(6/10) \times 0.5\right] + \left[(4/10) \times 0.375\right]$	0.45
Gini Gain	0.48 − 0.45	0.03

Node occupation	Gini computation Formula	Gini Index
overall	$1-\left[(4/10)^2+(6/10)^2\right]$	0.48
occupation (salary)	$1-\left[(2/5)^2+(3/5)^2\right]$	0.48
occupation (self-employed)	$1-\left[(2/5)^2+(3/5)^2\right]$	0.48
occupation (split)	$\left[(5/10)\times 0.48\right]+\left[(5/10)\times 0.48\right]$	0.48
Gini Gain	$0.48-0.48$	0.00

Since the gender has high gini gain and low gini split when compared to occupation, Gender has the high priority and low mis-classification and it takes the first split (root node).

Problem:

{Find the gini gain for the given tree split} ⇒



overall : $1-\left[(60/100)^2+(40/100)^2\right] \Rightarrow 0.48$

{Gender (Male)} : $1-\left[(10/25)^2+(15/25)^2\right] \Rightarrow 0.48$

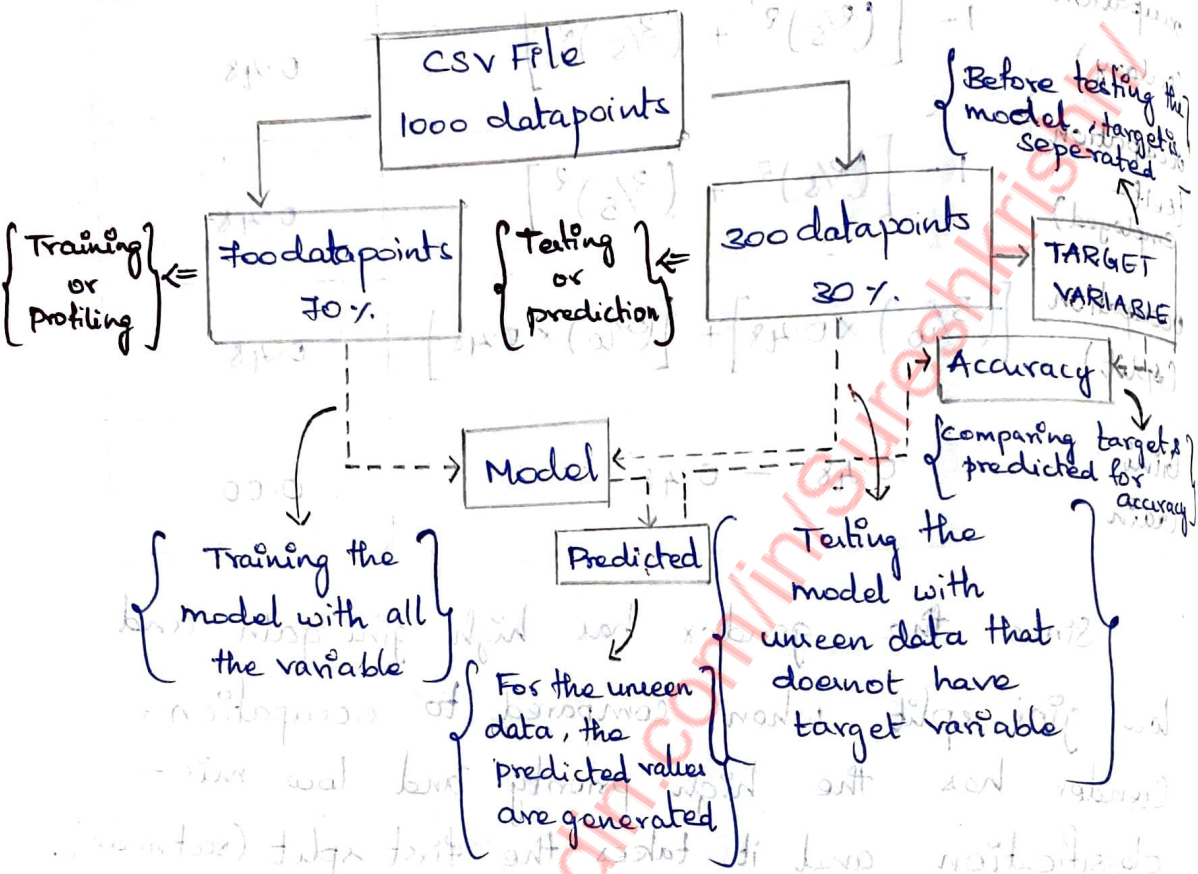
{Gender (Female)} : $1-\left[(30/75)^2+(45/75)^2\right] \Rightarrow 0.48$

{Gender (split)} : $\left[(25/100)\times 0.48\right]+\left[(75/100)\times 0.48\right] \Rightarrow 0.48$

{Gini Gain} : $0.48-0.48 \Rightarrow 0.00$

Training and Testing:

Training and testing in CART is also called as profiling and prediction.



Encoding: (one hot encoding)

One hot encoding is used to convert the categorical variable into quantitative variable. This is done because the ML model can understand the categorical data.

Gender
Male
Female
Female
Female
Male
Female
Male
Male

(one hot encoding) →

Gender
1
0
0
0
1
0
1
1

Target
No
Yes
Yes
No
Yes
No
No
Yes
No

(one-hot encoding) →

Target
0
1
1
0
1
0
0
1
0

Cust.	Occupation
C_1	self
C_2	Non-self
C_3	Govt
C_4	self
C_5	private
C_6	self

one-hot encoding →

cust	self	Non self	Govt	Private
C_1	1	0	0	0
C_2	0	1	0	0
C_3	0	0	1	0
C_4	1	0	0	0
C_5	0	0	0	1
C_6	1	0	0	0

{one-hot encoding} : converts categorical variable into 0's and 1's.

{Label encoding} : Assigns numerical values (0, 1, 2, ...) to the categorical variable. By this the converted values are considered as weights. (ie. $0 < 1 < 2 < 3 < \dots$). So it is not a effective method.

When to we one-hot encoding and when to we label encoding ?

One hot encoding when:

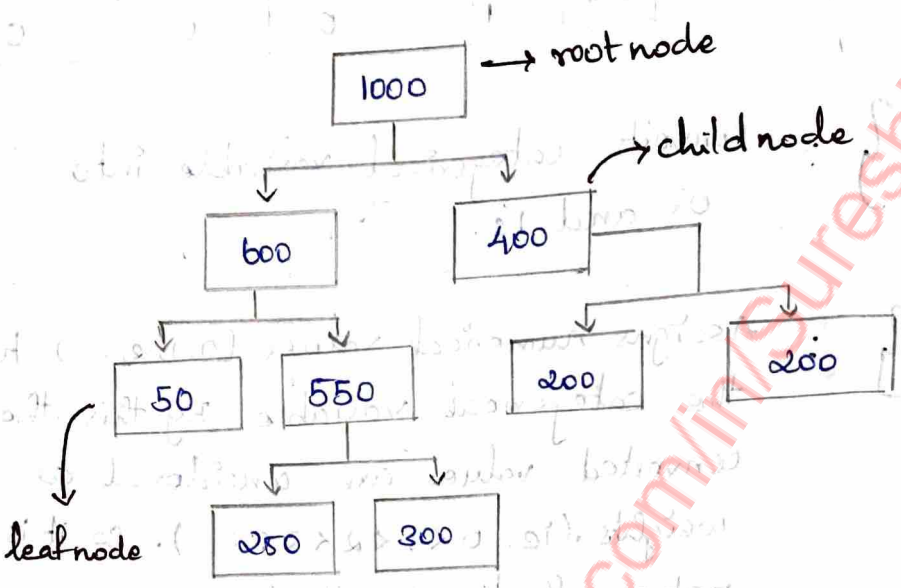
* The categorical feature is not ordinal. (eg: countries, color, job occupation etc..)
* When the no. of. categorical feature is low.

Label encoding when:

* The categorical feature is ordinal (eg: rank, class, job type etc..,)
* When the number of categories is quite large.

Code to build CART tree:

```
from sklearn.tree import DecisionTreeClassifier
model_df = DecisionTreeClassifier(max_depth = 8, criterion =
    'gini', min_sample_split = 100,
    min_sample_leaf = 10):
```



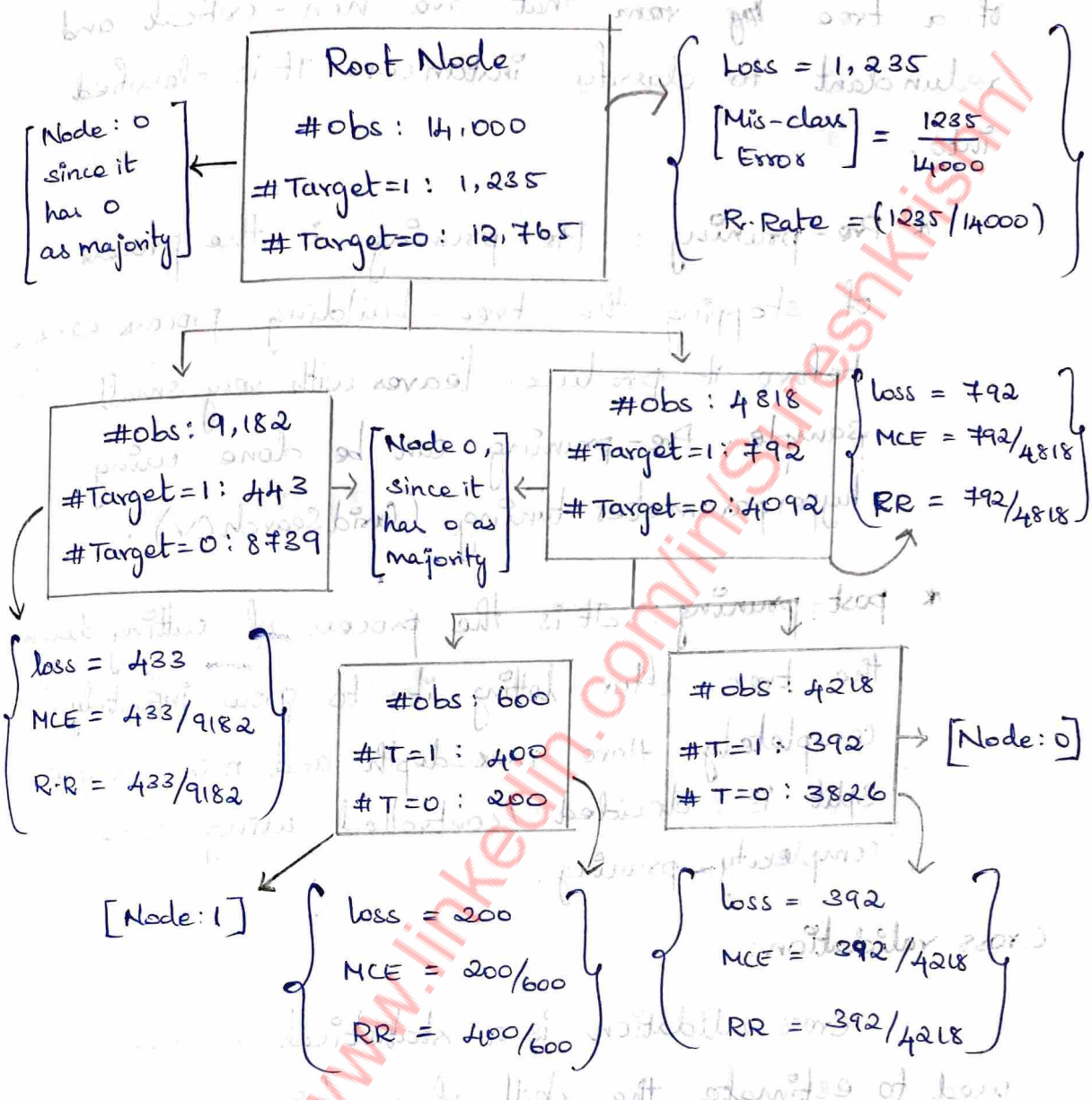
* Min_sample_split: The minimum number of observation that should present in a node in order to split any further.

* min_sample_leaf: The minimum number of observation in any terminal leaf node (node that will be terminated if it contains such observations). If min_sample_split is only specified then min_sample_leaf will be min_sample_split / 3. If min_sample_leaf is specified then min_sample_split will be min_sample_leaf x 3.

* max_depth: The maximum depth of a tree. If none is specified, the tree grows till all leaves are pure or untill all leaves contains less than min_sample_split.

* Criterion - 'gini' for gini impurity and 'entropy' for information gain.

Loss, Mis-classification error and response rate :



Loss : Number of cases mis-classified in the given node.

Mis-classification error : Ratio of total number of cases misclassified to the total number of observations at the node. (have to find for full tree)

Response Rate : Ratio of total number of responders (Target=1) to the total number of observation at node.

Nodes with high response rate are always best and that what we are interested in.

Pruning:

Pruning is the process of reducing the size of decision tree by removing the sections of a tree that are non-critical and redundant to classify instances. It is classified into,

* Pre-pruning: Pre-pruning is the process of stopping the tree-building process early, before it produces leaves with very small sample. Pre-pruning can be done using hyperparameter tuning (GridSearchCV).

* post-pruning: It is the process of cutting down the tree after letting it to grow infinitely/completely. Here max_depth and min_sample_split is decided/controlled using cost-complexity-pruning.

Cross validation:

Cross validation is a statistical method used to estimate the skill of machine learning model on new (unseen) data. Common methods of cross validations are,

* K-Fold cross validation.
* stratified cross validation.

K-fold cv uses random sampling so the chances of bias in sampling is possible. stratified cv divides based on strata and does random sampling on each strata.

Typical value of k-fold cross validation is set to 10. This number represents the number of folds.

k Fold CV	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Fold-1	Train	Train	Train	Train	Train	Train	Train	Train	Test	Test
Fold-2	Train	Train	Train	Train	Train	Train	Test	Test	Train	Train
Fold-3	Train	Train	Train	Train	Test	Test	Train	Train	Train	Train
Fold-4	Train	Train	Test	Test	Train	Train	Train	Train	Train	Train
Fold-5	Test	Test	Train	Train	Train	Train	Train	Train	Train	Train

obs = 10
k = 5
Test per fold is equal to obs/k
obs/k = 10/5 = 2

Accuracy metrices:

* Confusion matrix - Precision, Recall & F1 score

		Actual			
		Positive	Negative		
Predicted	Positive	A (TP)	B (FP)	Predicted +ive rate (PPR)	$\frac{A}{(A+B)}$
	Negative	C (FN)	D (TN)	Predicted -ive rate (PNR)	$\frac{D}{(C+D)}$
		sensitivity	specificity		
		$\frac{A}{(A+C)}$	$\frac{D}{(B+D)}$		

(Precision)

(Recall) ⇒ True positive Rate

$$\text{Accuracy} = \frac{A+D}{(A+B+C+D)} \quad (\text{or}) \quad \frac{(TP+TN)}{(TP+FP+FN+TN)}$$

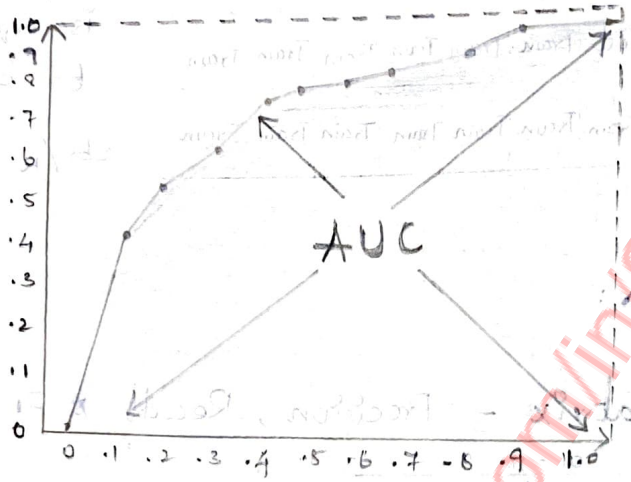
$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

False positive Rate = 1 - specificity.

* Area under the curve: (AUC)

Area under the curve shows how many mistakes (wrong prediction) are made by the model before it makes correct prediction.

For AUC, False positive rate is plotted against the true positive rate.



Random Forest

Random forest is a supervised learning, classification algorithm. The forest that it builds is an ensemble of decision tree, usually trained with a bagging method. Feature importance can be easily measured using random forest.

Ensemble: Ensemble technique uses multiple learning algorithms to obtain better predective performance.

eg: Random forest is built using multiple decision trees.

Bootstrap Aggegating :- (Bagging):

Bagging is the process of creating multiple decision tree from the given training set. For each of the decision tree the observation are picked by doing random sampling with replacement.

* In decision tree algorithm he have fixed proportion of target variable. But in random forest due to bagging each decision tree has a varying proportion.

* The models are fitted using the bootstrap samples (decision tree) and combined by averaging the voting. Eg: For a test datapoint if the output proportion is predicted as 1 in 700 trees out of 1000 trees, then the final predicted output will be 1.

* Individual trees in the random forest may have high over-fitting but averaging the output removes the bias. Hence randomforest helps reduce over-fitting.

RandomForest is a blackbox Technique

Inside the RandomForest blackbox:

Step-1: * Random sampling with Replacement.
Here we create "n" number of trees by using sampling with replacement.

ID	X_1	X_2	X_3	X_4	...	Y
R_1						
R_2						
R_3						
.						
.						
.						
R_n						

⇒ Training set

D_1

ID	X_1	X_2	X_3	X_4	...	Y
R_{10}						
R_3						
R_{17}						
.						
.						

D_n

ID	X_1	X_2	X_3	X_4	...	Y
R_1						
R_1						
R_{12}						
.						
.						

D_2

ID	X_1	X_2	X_3	X_4	...	Y
R_1						
R_1						
R_{12}						
.						
.						

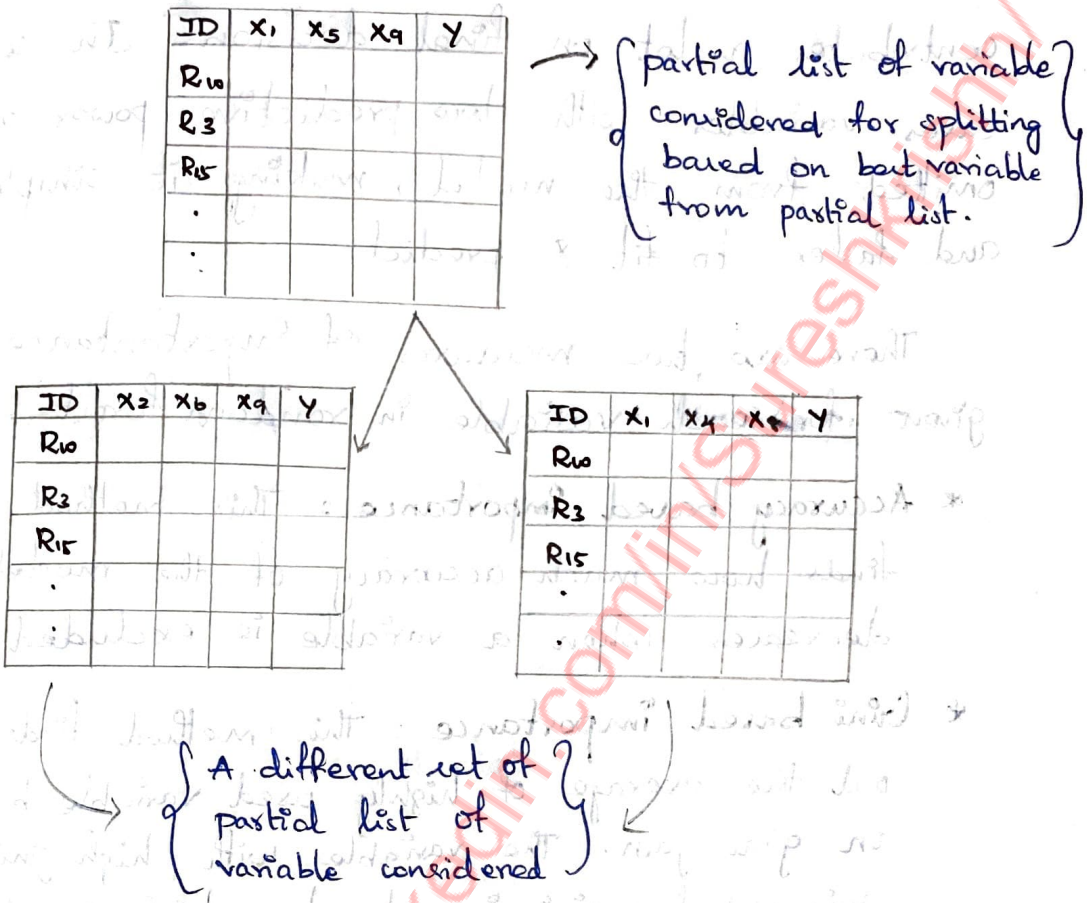
$D_...$

ID	X_1	X_2	X_3	X_4	...	Y
R_1						
R_1						
R_{12}						
.						
.						

$D_1, D_2, \cdots, D_n$ ⇒ Decision tree created by bagging.

step-2 : * Building the tree for each sample with only partial set of 'm' variables (predictors) being considered at each node.

* $m << M$, where M = total # of predictor variable



step-3 : * classifying.

→ Based on 'n' samples, 'n' trees are built.

→ Each records is classified based on 'n' tree. Final class record is decided based on voting.

We don't have pruning in Random Forest

Out of Bag error rate :

Out of bag error rate (oob) is computed from averaging the loss on each decision tree.

{ Number of mis-classification at each node of tree }

Based on OOB, the number of trees to build can be decided.

Variable Importance:

Decisions on the output of random forest is purely driven by the predictive power of the variable. The variable with high predictive power contributes a lot on final decision. In some cases variables with low predictive power are omitted from the model, making it simpler and faster to fit & predict.

There are two measures of importance given for each variable in random forest,

* **Accuracy based importance:** This method finds how much accuracy of the model decreases when a variable is excluded.
* **Gini based importance:** This method finds out the average of highly used variable based on gini gain. The variable with high gini gain (or) low gini impurity has high importance.

Code to build RandomForest:

```
from sklearn.ensemble import RandomForestClassifier
RandomForestClassifier(bootstrap = True, criterion = 'gini'
{number of tree to build (decided by oob)} <- N_estimator = 100,
{Number of variables randomly sampled} <- Max_features = auto,
{minimum # of records in terminal node} <- Max_leaf_node = 10,
                                           min_sample_split = True)
```

Which one is best? RF or decision Tree?

No algo is superior or inferior. The usage depends on domain. Mostly in banking and health RF is not used due to no clarity in structure. RF is prefered when there is more predictor variables.

Linear Regression

⇒ Linear regression is a machine learning algorithm that finds the best linear fit relationship on any given data, between independent and dependent variable.

⇒ The target or dependent variable in case of linear regression will be a continuous variable.

⇒ Linear regression is also called as multiple regression, multivariate regression, ordinary least square (OLS) regression.

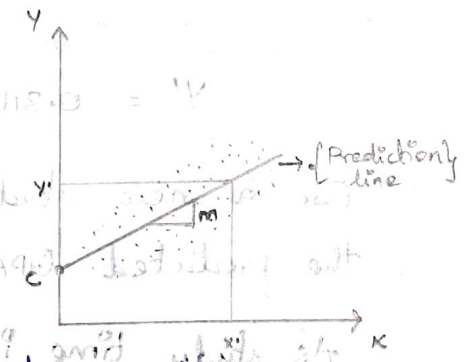
⇒ In linear regression, the regression line predicts the patter of data.

> PREDICTS THE CHANGE IN "Y" WHEN "X" INCREASES BY ONE UNIT
>
> (CHANGE → Increase / decrease)

Regression line equation:

$$y' = mx + c$$

- y' → {predicted value of y}
- m → {slope}
- x → {any value of x}
- c → {y intercept}



From the above equation:

$$c = -m\bar{x} + \bar{y} \quad \text{(or)}$$

$$c = \bar{y} - m\bar{x}$$

and

$$m = r\left(s_y / s_x\right)$$

r - correlation coeff.

s_x, s_y - slope standard deviation of x & y.

Example:

The table represents the study time and GPA of students. Predict the student GPA from amount of time they study each week. $r = 0.9$

Hours (x)	GPA (y)
1	2
2	1.5
3	2.5
5	3.5
6	3.0
8	4.0
10	4.5
$\bar{x} = 5$	$\bar{y} = 3$
$S_x = 3.26$	$S_y = 1.08$

We know that, $y' = mx' + c$ — ①

$$c = \bar{y} - m\bar{x}$$

$$m = r\left(S_y / S_x\right) = 0.94\left(1.08 / 3.26\right) = 0.311 \text{ — ②}$$

$$c = (3) - (0.311 \times 5)$$

$$c = 1.45 \text{ — ③}$$

Substitute ② and ③ in ①, we get

$$Y' = 0.311(x') + 1.45$$

For a new student with study time 7 hours the predicted GPA will be $Y' = 0.311(7) + 1.45$

As study time increases by one hour, we predict the GPA increases by 0.311

The example is just a numerical representation with less complex line fitting. When we have more number of datapoints with similar x value but different y value, we choose to find RSS for best line of fit.

RSS - Residual sum of Squares.

$$RSS = \sum_{i=1}^{n} : [(Actual - predicted)^2]$$

RSS is squared to prevent the cancellation of lower order values by higher ones. [The line with least RSS value is considered as the best fit line.]

Accuracy of linear regression :

The accuracy of a linear regression model can be evaluated by a term called R^2.

- R^2 - It represents the percentage variation in the data that the model can understand / capture. Higher the value of R^2 better the prediction will be.

- {Adjuted R^2} - Adjuted R^2 gives the value of per unit change in R^2 value when you are removing or adding a particular variable in the model.

- One big disadvantage of R^2 is that it will keep on increases when a variable is added to a model, even if the variable is not significant. In the other hand the adjuted R^2 decreases if the variable is not significant. In this way Adjuted R^2 is superior than the R^2.

Code for checking statistical significance:

```
import statsmodel.api as sm
mod = sm.OLS (y, x)
res = mod.fit()
print (res.summary())
```

dependent variable, R^2, Adjusted R^2, P-values can be found using this

Code for Linear Regression model:

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
model = linreg.fit(x_train, y_train)
```

Code for finding model accuracy (R^2):

```
print (model.score (x_train, y_train))
print (model.score (x_test, y_test))
```

Clustering

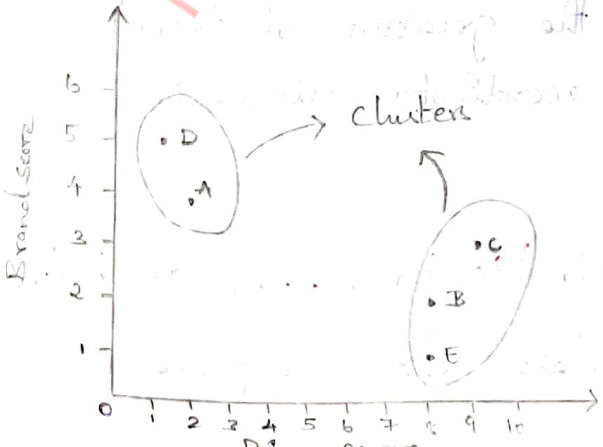
* Clustering is a type of unsupervised learning algorithm. It is a technique of finding similar group of datapoints in the data, called as clusters.

* It groups data instances that are similar to each other in one cluster and data instances that are very different from each other into different clusters.

* Cluster is a collection of object that are homogeneous between them and heterogeneous to objects of other clusters.

Example:

Individual	Price Score	Brand Score
A	2	4
B	8	2
C	9	3
D	1	5
E	8	1



Insights:

From scatter plot, we see that,

* customes D & A belong to the cluster of very high brand score
* customer C & B & E belongs to the cluster of very high price score

Distance computation:

In any of the clustering algorithm, distance computation plays a major role in decision making. Most popular distance metrices are,

* Euclidean distance.
* Chebyshev distance.
* Manhattan distance.

Euclidean distance:

Euclidean distance is measured by drawing a shortest line between two points and pluging the values into the formula

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \ldots + (n_2 - n_1)^2}$$

Euclidean distance is the extension of Pythagoras theorem.

Chebyshev distance:

In chebyshev distance, the distance between two vectors is the greatest of their differences among all coordinate dimensions. The formula is,

$$d = \max(|a_1 - b_1|, |a_2 - b_2|, \ldots, |a_n - b_n|)$$

This is used for the data where response is ordinal in nature.

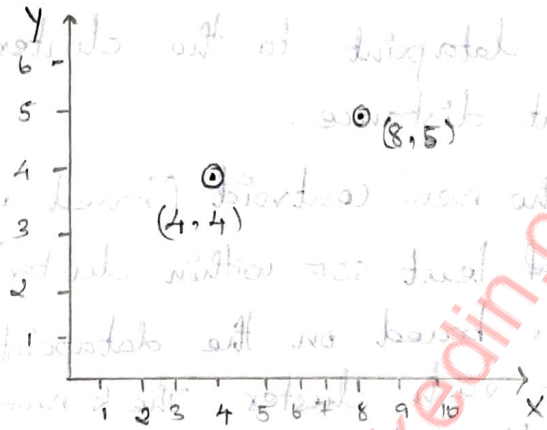
Manhattan distance:

Manhattan distance is called as the city block distance. It is the sum of perpendicular distances. Formula is,

$$d = |a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + \ldots |a_n - b_n|$$

Example:

Formulate Euclidean, chebyshev and manhattan distance for the given two points.



$$\text{Euclidean} = \sqrt{(8-4)^2 + (5-4)^2}$$

$$= \sqrt{16+1}$$

$$= 4.123$$

$$\text{chebyshev} = \max(|8-4|, |5-4|)$$

$$= \max(4, 1)$$

$$= 4$$

$$\text{manhattan} = |8-4| + |5-4|$$

$$= 4 + 1$$

$$= 5$$

K-means clustering:

K-means clustering is a unsupervised, non-hierarchical clustering algorithm. It is based on within cluster variation. (squared distance from the centre of cluster).

steps in K-means algorithm: (Inside blackbox)

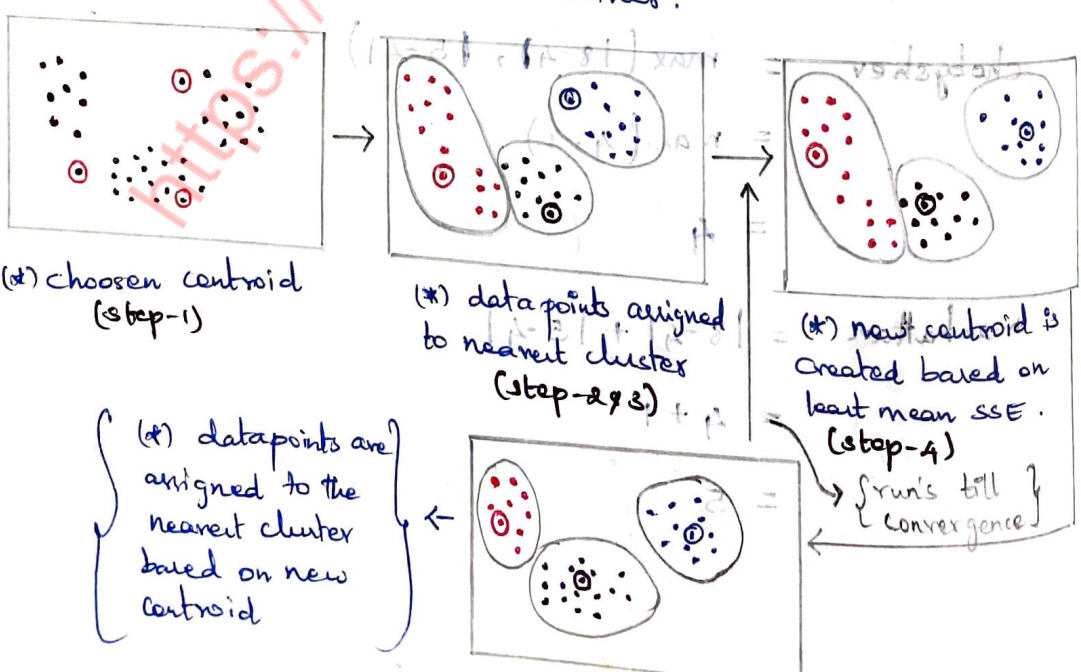
step-1: Assume K-centroids (for K-clusters)

step-2: compute Euclidean distance of each datapoint with these centroids.

step-3: Assign the datapoint to the cluster within nearest distance.

step-4: compute the new centroid (based on the mean of least SSE within cluster) of each cluster based on the datapoints assigned to each cluster. The K number of means obtained will became the new centroid for each cluster.

step-5: Repeat steps 2-4 till convergence. That is defined by no movement of datapoints from one cluster to another.



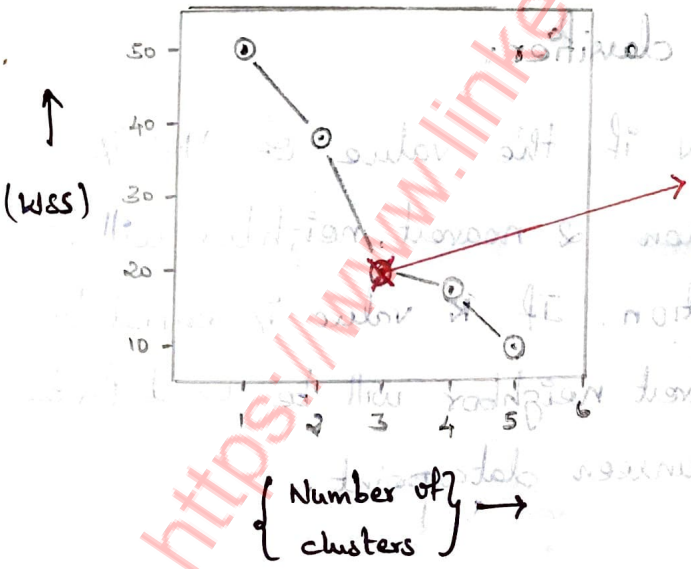
centroids are the existing datapoints, they are not created as new data point

choosing optimum number of clusters:

The one of the biggest challenge of K-means clustering is pre-defining the number of clusters (step-1). To pre-define the optimum number of clusters we have to find it using elbow graph, that is popularly called as WSS plot.

WSS plot:

Within Sum of Square error (WSS) plot is used to find the optimum number of cluster. This plots the number of clusters and the within group sum of square error of each cluster.



K=3 is the optimum number of cluster because that is where the elbow is formed. After this point it is considered as overfitting

Standard Scalar / preprocessing.scale() :

It converts normal distribution to standard normal distribution where $\mu = 0, \sigma = 1$ always. This makes all the feature in same range.

Required In - Regression, K-means, KNN, SVM

Not required In - Decision tree, Random forest (As it uses GINI)

K - Nearest Neighbor classification (KNN)

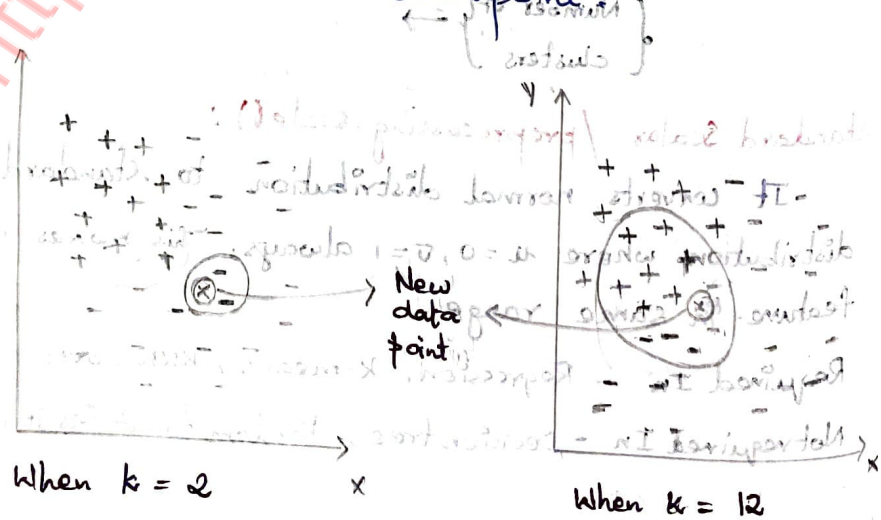
KNN is a classification type, supervised learning algorithm. It is an instance based classifier, that stores the training record and uses training records to predict the class label of unseen data / classes.

Example of instance based classifier:

* Rote learner: Memorizes entire training data and perform classification only if attributes of record matches the training example.
* Nearest Neighbor: Uses 'k' closest point that is called as nearest neighbor for performing classification.

Nearest Neighbor classifier:

* In KNN if the value of 'K' is equal to 2, then 2 nearest neighbor will be used for prediction. If K value is equal to 12 then, 12 nearest neighbor will be used for the prediction of unseen datapoint.



* The distance between the nearest neighbor and the unseen datapoint is calculated using Euclidean distance.

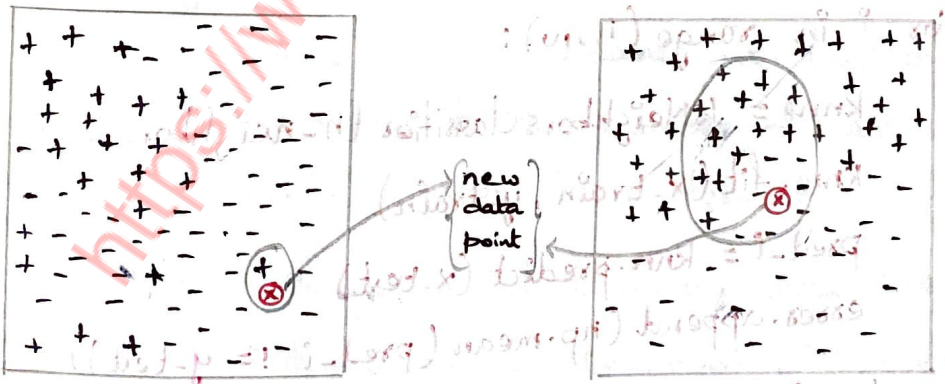
$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

$$\text{similarity} = 1/d$$

$$s \propto 1/d \Rightarrow \{\text{When the distance between the points increases, the similarity between them decreases \& viceversa.}\}$$

Choosing the value of 'K':

* The value of 'K' should be choosen as odd number to make the correct decision. If it is even number, there is 50-50 chance of both classes and the prediction can fall in any of those classes.
* If the value of 'K' is too small, sensitive to noise points.
* If the value of 'K' is too large, neighborhood may include points from other class.



(*) When 'K' value is small. Actually the new data point has to be (-) but due to low 'K' value it is (+)

(*) When 'K' value is large. The new data point has to be (-), but due to high 'K' value it is predicted as (+) by taking points from other class.

* The optimum value of 'k' has to be choosen by running a loop with a range of 'k' value and plotting it with graph. The value with optimum and less number of error rate (misclassification) has to be choosen.

code for kNeighborsClassifier:

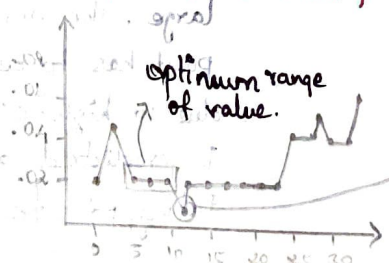
```
from sklearn.neighbors import kNeighborsClassifier
classifier = kNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)
```

code for prediction & Evaluation:

```
y_pred = classifier.predict(x_test)
from sklearn.metrics import classification_report,
                            confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

code for optimum k-value detection:

```
error = []
for i in range(1,40):
    knn = kNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))
plt.plot(range(1,40), error)
```



* KNN is a lazy learner. It doesnot build model explicitly unlike eager learners such as decision tree & random forest. So, this is computationally expensive algorithm.

Difference between clustering and classification?

(*) clustering is unsupervised ML algorithm, the clusters are homogeneous within themselves and hetrogeneous to each other.

(*) classification is supervised ML algorithm, the classifiers can be hetrogeneous or homogeneous to each other.

Difference between KNN and K-means?

(*) KNN is a classification type supervised learning algorithm. The 'K' value has to be predicted by using misclassification rate (error) by looping over a range of value and the value of 'K' represents the number of neighbors to select near the point.

(*) K-means is a clustering type unsupervised learning algorithm. The value of 'K' in K-means is found using within sum of square error (WSS) plot. The value of 'K' represents the number of centroids and the number of clusters to be formed for clustering.

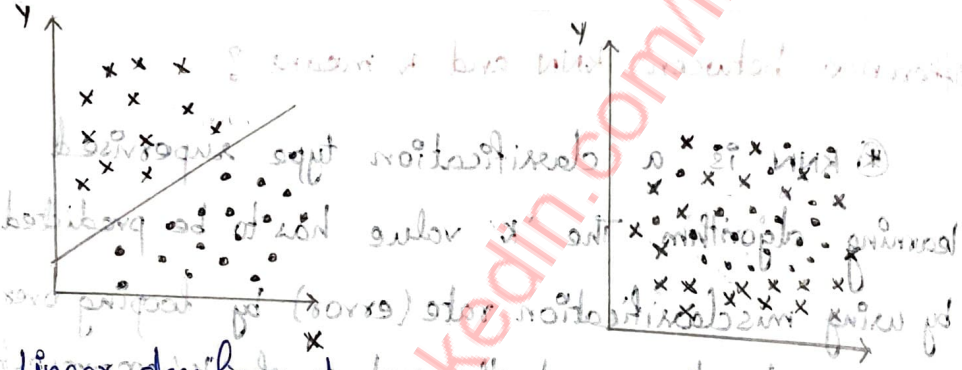
similarity: Both KNN and K-means uses the distance metric (Euclidean distance) to calculate the nearest points.

Support Vector Machines (SVM)

* SVM is a classification type, supervised learning algorithm. It is related to statistical learning theory developed in 1992. It has a greter success in handwritten digit recoganisation.

* It is a example of kernel learning method / kernal method. It is one of the key area in machine learning.

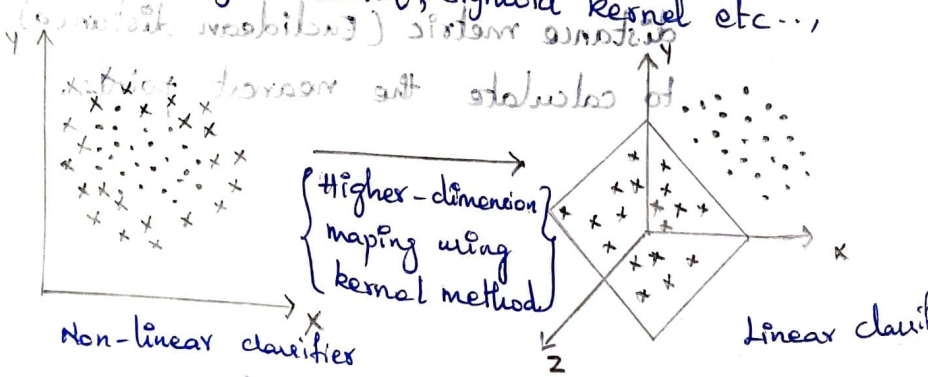
Types of classifier: Linear & non-linear classifiers.



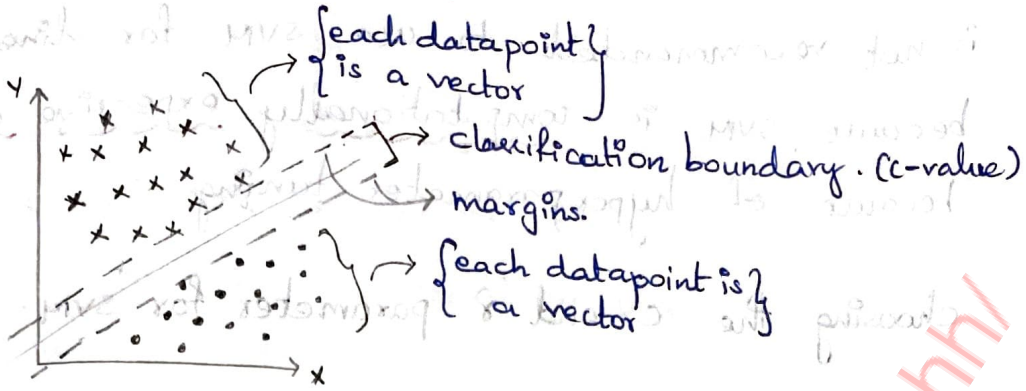
Linear classifier, where the data points can be seperated by simple line.

Non-linear classifier, where the datapoint can't be seperated by simple line.

SVM is capable of handling both linear and non linear data by using its kernel method. Few of the SVM kernels are, polynomical kernel, Gaussian kernel, Gaussian radial basis function, Laplace RBF, Hyperbolic tangent kernel, sigmoid kernel etc..,



How SVM algorithm works?



* SVM creates the classification boundary as a threshold that touches the extreme datapoints in both the cases. Below or above which the classes are predicted by the threshold.

* The datapoints are called as the vectors. The margin of boundary (classification boundary) is supported by these vectors and so it's called as SVM.

Choosing the kernel function:

i) Polynomial kernel. → For non-linear data.
ii) Radial or RBF kernel. → For non-linear data.
iii) Linear kernel. → For linear data.

How to find the data is linear / non-linear ?

* We cant simply say by seeing the data, As there is no way of visually finding the linearity of data. As in real time it has many dimensions. Number of dimensions will be the number of features so it is not possible to find visually by seeing data.

* When the basic classification algorithm gives low accuracy even after all EDA (65% - 75%), then the data may be non-linear type. Then try SVM and if accuracy increases then its non-linear data.

SVM can also handle the linear data. But it is not recommended to use SVM for linear data because SVM is computationally expensive algorithm because of hyperparameter tuning.

Choosing the C and γ parameter for SVM:

C parameter: It is a regularisation parameter that controls the tradeoff between achieving a low training error and low testing error. It is a ability to generalise the classifier to unseen data.

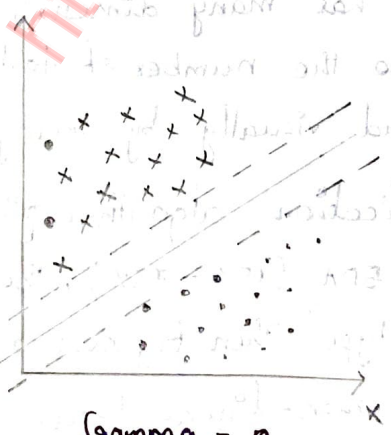
C represents the width of the margin for classification purpose.

Higher C = less width ⇒ not a good model

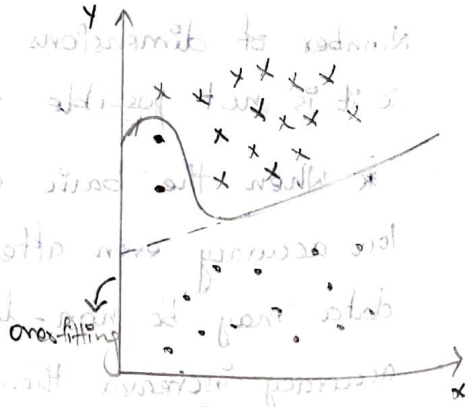
Lower C = Higher width ⇒ good model

Optimum value of C has to be calculated based on model accuracy.

Gamma parameter: It defines how far the influence of single training example reaches. Low value means far and high value means close. Optimum value of γ has to be choosen otherwise model will run into overfiting.



Gamma = 0



Gamma = 100

How to choose the optimum value for c, γ and the kernel method?

The optimum value of c, gamma and the right kernel can be choosen with the help of GridSearchCV. The GridSearchCV was designed orginally for SVM because the hyperparameter tuning is difficult in SVM.

* SVM should not be used unless until the accuracy cant be increased and the data can't be linearly seperable. This is defined by the law of passimony, it states that "work of ~~sword~~ needle should not be done by sword". We have to try other non expensive algorithm for classification and prefer SVM and KNN at worst case.

* Random forest is a space consuming algorithm when we have a contrain of space we can use decision tree. Because random forest creates data withusing bagging, so it creates more trees.

When we know the data is non-linear which algorithm can be used?

SVM can be used because it can use kernel method to transform the non-linear data into linear data. In this way SVM is superior in handling non-linear data.

Logistic Regression

* Logistic Regression is a supervised learning, classification algorithm. It is used to predict binary outcome based on a set of independent variable. It is probabilistic model.

* like other classification algorithm, logistic regression can be used with independent variables that falls in following categories, and are called as,

i) Binary logistic Regression - spam / not-spam
ii) Multinomial logist Regression - veg / Non-veg / vegan
iii) ordinal logistic Regression - Movie rating (1 to 5)

Sigmoid Function:

sigmoid function can convert any number between $-\infty$ to $+\infty$ into the range of 0 to 1. The formula for sigmoid function is,

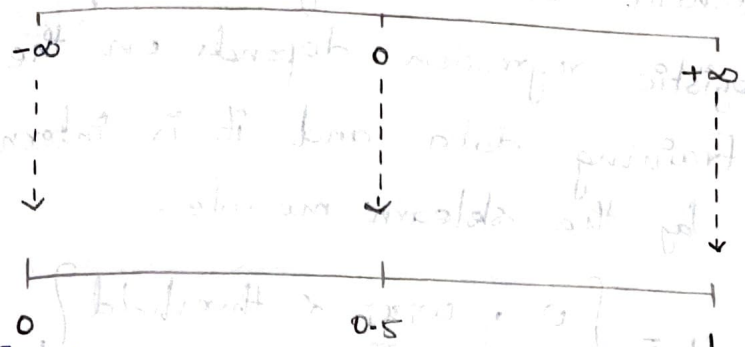
① $$S(x) = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-y}}$$

② $$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \ldots + w_nx_n$$

Substitute ② in ①, we get,

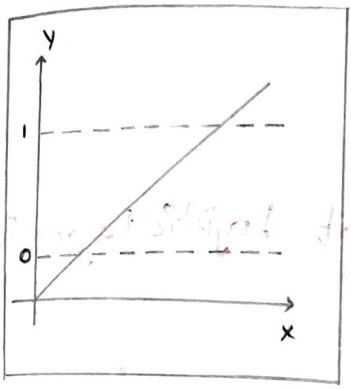
$$S(x) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + \ldots + w_nx_n)}}$$

→ Sigmoid function gives as a sigmoid curve that is also called as 's-curve'.



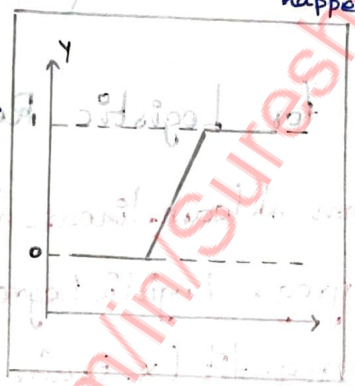
mapping the values between $-\infty$ to $+\infty$ into range of 0 to 1.

$[(val < 0.5) = 0 \quad , \quad (val > 0.5) = 1]$ → { This is how the classification happen in logreg }



{ Transforming a line into s-curve } →

$$s(x) = \frac{1}{1+e^{-x}}$$



Softmax function:

$$p = \frac{e^x}{1+e^x} = \frac{e^{(w_1x_1+w_2x_2+\ldots+w_nx_n)}}{1+e^{(w_1x_1+w_2x_2+\ldots+w_nx_n)}}$$

ODDS RATIO:

The weight factors $(w_1x_1 + w_2x_2 + \ldots)$, the value of w_n will be calculated by odds ratio in logistic regression. The odds ratio is defined as the ratio of odds of A in the presence of B, A in the absence of B or equivalenty odd of B in the presence of A, B in the absense of A.

It Quantifies the strength of association between two events A and B.

$$\text{odd} = \frac{\text{Probability}}{1 - \text{probability}}$$

Threshold for Logistic Regression:

Threshold for acheiving the highest accuracy in the logistic regression depends on the output of the training data and it is internally assigned by the sklearn module.

$$y = \begin{cases} 0, & w_i x_i < \text{threshold} \\ 1, & w_i x_i > \text{threshold} \end{cases}$$

code for Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_test)
```

code for accuracy and performance:

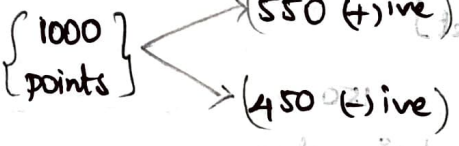
```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
print('Accuracy', metrics.accuracy_score(y_test, y_pred))
print('precision', metrics.precision_score(y_test, y_pred))
print('Recall', metrics.recall_score(y_test, y_pred))
```

How to handle a Imbalance dataset ?

Balanced dataset :

Balanced dataset is one which contains equal or approximately equal number of samples from the each available class.

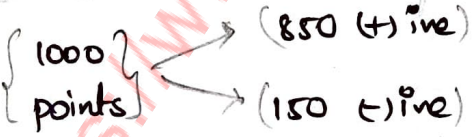
Ex : outof 1000 datapoints available, if 550 belongs to +ive class and 450 belongs to -ive class then it's balanced dataset.



Imbalance dataset :

Imbalance dataset is one which doesnot contain equal or approximately equal number of samples for each available class.

Ex : out of 1000 datapoints available, if 850 belong to (+)ive and 150 belong to (-)ive class then its imbalanced dataset.



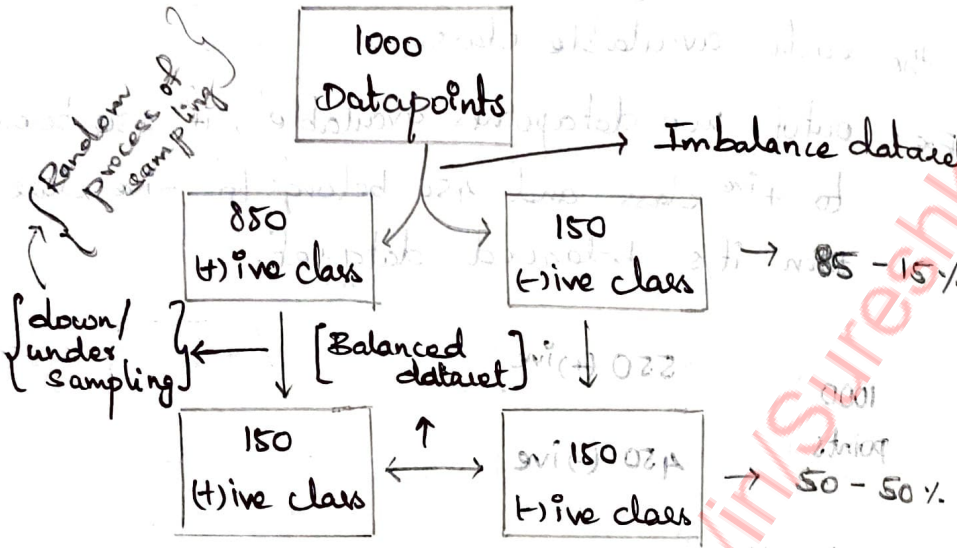
Ways to handle imbalance dataset :

Imbalance dataset may leads to low recall scores and larger misclassification in minority class. This can be prevented by converting imbalance dataset to balanced one. This is done by,

* Down sampling (undersampling)
* Up sampling (oversampling)

* Down sampling / under sampling:

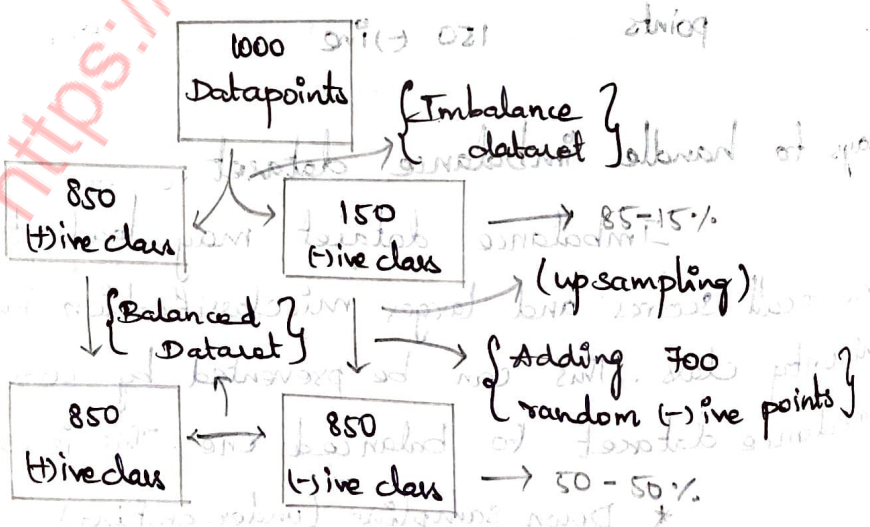
Under sampling simply means that delete samples from majority class untill the dataset becomes balance.



→ This leads to artificially reducing the size of dataset, by which some value information might be missed.

* upsampling / over sampling: (simple)

Over sampling simply means duplicating samples from minority class untill the dataset becomes balanced.

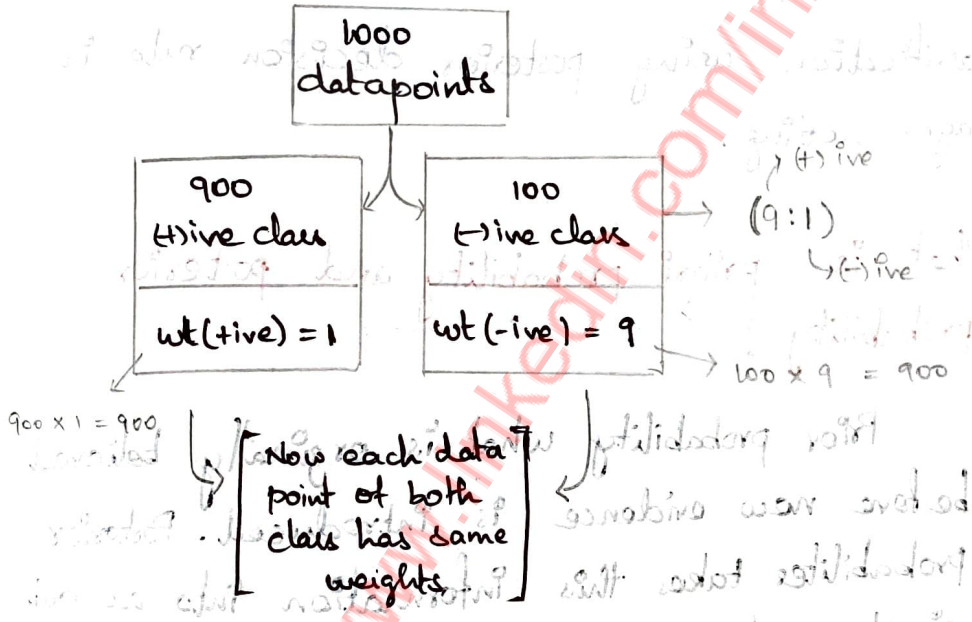


* upsampling / oversampling by extrapolation:

This is called as synthetic minority over-sampling technique (SMOTE). In this method of oversampling artificial datapoints / synthetic datapoints for minority class is generated till the dataset is balanced.

* upsampling / oversampling by class-weight:

In this method the weight of the minority class will be equal to the proportion of the majority class.



→ oversampling may leads to overfitting of the machine learning models, because it creates the exact copies / duplicates of the existing minority class data points.

Naive Bayes classification

It is a supervised learning, classification type probabilistic machine learning algorithm. It is based of Bayes theorem for calculating probabilites and conditional probabilities.

Bayes Rule:

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

It is a probabilistic model that makes classification using posterior decision rule in Bayes setting.

What is priori probability and posterior probability ?

Prior probability what is orginally believed before new evidence is introduced. Posterior probabilites takes this information into accoub. simply, whatever is known is called as prior probability (coin, dice, card, etc..). Whatever has to be calculated by data is called as posterior probability.

What are all the probabilistic machine learning models?

* Logistic Regression.
* Naive Bayes classification.

How Bayes classification work?

Age	Gender	Occupation	skill	Target
26	M	self	ML	0
25	F	Govt	C++	1
24	M	Govt	ML	0
22	M	Priv	ML	1
26	F	sal	C++	1
24	M	self	C++	0

Now a new datapoint arrives for prediction:

→ Age - 26, Gender - M, occupation - self, skill - ML

Now the prediction happens using Bayes rule, with available training data.

$P(Y=0) \Rightarrow P(Y=0 \mid Age) \times P(Y=0 \mid Gender) \times P(Y=0 \mid skill) \times P(Y=0 \mid occupation)$

$P(Y=1) \Rightarrow P(Y=1 \mid Age) \times P(Y=1 \mid Gender) \times P(Y=1 \mid skill) \times P(Y=1 \mid occupation)$

decision: If $P(Y=0)$ is (say - 0.88) greater than the $P(Y=1)$ (say - 0.12), then the new dataset will fall into '0' class.

If $P(Y=1)$ is (say - 0.7) greater than $P(Y=1)$ (say - 0.3), then the new dataset will fall into '1' class.

Find the probability of one given another :

A patient takes the lab test and results came as positive. The test returns correct positive results in 98% of cases in which the disease is actually present and correct negative results in 97% where disease is not present. Furthermore, 0.008 of entire population has cancer.

probabilities :

$P(\text{Cancer}) = 0.008$ $\quad$ $P(-\text{cancer}) = 0.992$

$P(+|\text{cancer}) = 0.98$ $\quad$ $P(-|\text{cancer}) = 0.02$

$P(+|-\text{cancer}) = 0.03$ $\quad$ $P(-|-\text{cancer}) = 0.97$

Why Bayes classifier is called as 'Naive' Bayes classifier ?

It is called as Naive because the character of one feature doesnot affect the character of other features. Each feature has a direct probabilistic relation with the target variable. It is not affected by mult colinearity, it doesnot requires scalling / Normalisation.

In simple its so Naive that it doesnot care about other variables.

Principal Component Analysis (PCA)

Principal component Analysis (PCA) is a feature engineering technique used for dimension reduction. It is invented by pearson and hotelling.

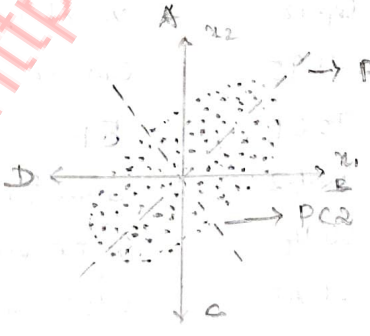
PCA takes a data matrix of n object by p variables, which may be correlated and summarizes it by uncorrelated axes (principle component / principal axes) that are linear combination of orginal p variables.

Example:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Age	Wt	Ht	skill	Qual	Exp	sal	Target
PC1				PC2			y

Now PC1, PC2 can be further used for upcoming process such as training & testing.

PC's has to be created in a way that PC1 should have high variance, PC2 should have next high variance and so on...



All PC's should be orthogonal to each other. Because of which there is no possibility of covariance between the PC's

The Algebra of PCA:

* PCA uses Euclidean distance calculated from the p variables as the measure of dissimilarity among the n objects. So it is mandatory to standardize / normalise the data before PCA.

* In practice nobody uses PCA with two variables. Only when the dataset has more number of features PCA can be used (say, 300 variables).

* Eigen value and Eigen vectors are used behind the PCA for creating PC's.

Eigen Value - It gives the percentage (%) variance covered in particular axis.

Eigen vector - It gives the direction of the particular PC.

PCA is a blackbox technique

Eigen Value			
Axis/ PC's	Eigen value	% of Variance	cummulative % of Variance
1	5.855	36.60	36.60
2	3.420	21.38	57.97
3	1.122	7.01	64.98
4	1.116	6.97	71.95
5	0.982	6.14	78.09
6	0.725	4.53	82.62
7	0.563	3.52	86.14
8	0.529	3.31	89.45
9	0.476	2.98	92.42
10	0.375	2.35	94.77
11	0.312	1.98	96.75

" Optimum number of PC's can be choosen based on cummulative % of variance. We can see that upto 5 PC's the cummulative sum increases drastically between the PC and after which it has decreased. So 5 PC's is considered as optimum. "