Machine learning

17 March 2025 09:38

Machine learning

Computer -----> trained
How you make our computer trained

Prediction knowledge

How to predict something about future

	age	weight			age	weight
	20	64			26	73
2	22	67	3		22	67
2	24	70	3		20	64
2	26	73	3		24	70
2	28	76			28	76
					30	79
					25	71.5

We have multiple ways of making our machine trained

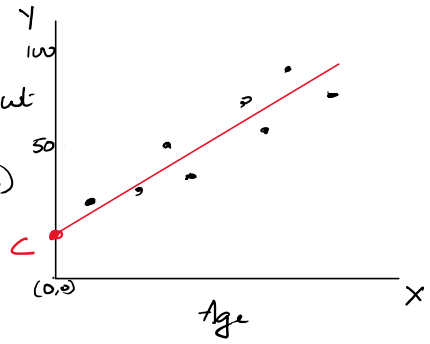
Scikit - learn ----> sklearn

Supervised learning methods
Unsupervised learning methods

Linear Regression:

Y = mx + C

Dependent variable ← y → weight
x → age (Independent vari)
m → slope
c → Intercept



Y = Dependent variable = Target variable = Output variable
X = independent variables = Input variables.

$y = mx + c$
$\hat{y} =$

y_i = Actual values $\hat{y}$ = predicted values

$y = mx + c$

↓

Model ← $y = \beta_0 + \beta_1 x$

β_0 = Bias = const

β_1 = Coefficient = Slope

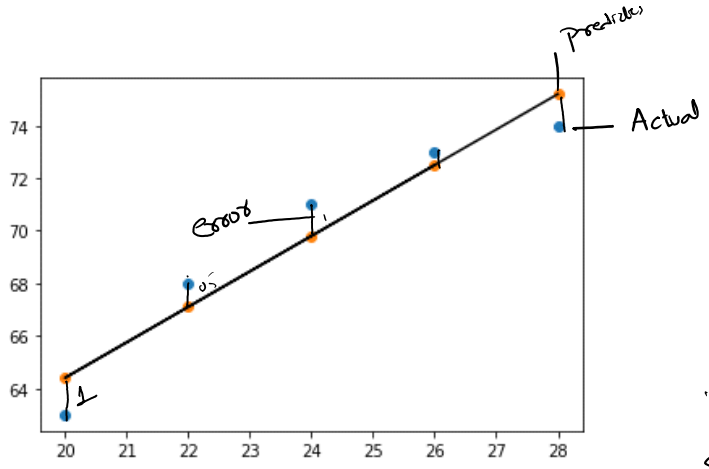
Simple Linear Regression:

β_1 = Coefficient = Slope

Simple Linear Regression:

We are trying to find a relationship in between given X and Y variables in the form of a mathematical model which is our simple linear regression concept.

$$y = 37.39 + (1.35 * Age) \longrightarrow \text{OLS Method}$$



$y_i - \hat{y}$ = Error

= Residual

$$\sum_{i=1}^{n} [y_i - \hat{y}]^2$$

Sum of Squared Errors

$$\frac{\sum [y_i - \hat{y}]^2}{n} \Rightarrow \text{Mean Squared Error}$$

$$\sqrt{\frac{\sum_{i=1}^{n} [y_i - \hat{y}]^2}{n}} \Rightarrow \text{Root Mean Squared Error}$$

We don't have any specific range to decide how much error can be acceptable, we always want to see error as least as possible

But we have an alternative measure to decide the models score or models performance.

R square:

Proportion of variation in Y "explained" by the regression on X

$$r^2 = \frac{\text{explained variation}}{\text{total variation}} = \frac{SSR}{SST} \qquad 0 \le r^2 \le 1$$

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

$\hat{y}$ = predicted values

$\bar{y}$ = Average of y values

$$R^2 = \frac{SSR}{SST} = \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

$\bar{y}$ = Average of y

y_i = Actual value

$\frac{\hat{y}_i}{y_i} = 100\%$

$y_i - \hat{y} = 0$

R^2 > 90% → excellent

> 80% → Good

> 70% → Average

< 70 → poor model

If my model contains lowest R square and highest MSE, then what shall we do
Add few more X variables which are related to your Y variables

Select the variables in a such way that which gives improvement of Rsquare and reduce the mse

If we are using more than one x variable, model becomes multiple linear regression

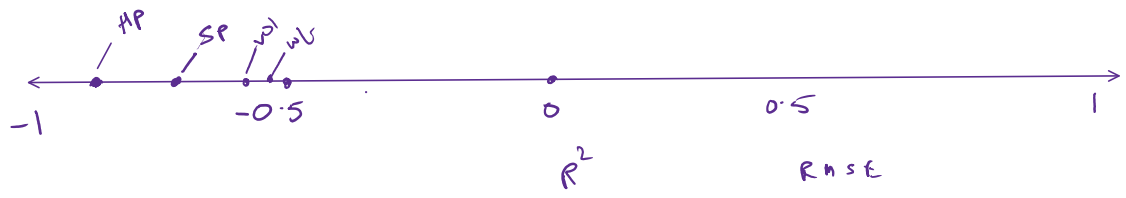
Multi linear regression

Weight: Age, height, Gender

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots \beta_K x_K$$

Select the variables through correlation is the best method to identify that which are important

```
In [32]: df.corr()
Out[32]:
            HP       MPG       VOL        SP        WT
HP    1.000000 -0.725038  0.077459  0.973848  0.076513
MPG  -0.725038  1.000000 -0.529057 -0.687125 -0.526759
VOL   0.077459 -0.529057  1.000000  0.102170  0.999203
SP    0.973848 -0.687125  0.102170  1.000000  0.102439
WT    0.076513 -0.526759  0.999203  0.102439  1.000000
```



1. Bo + B1 Hp
2. Bo + B1 Hp + B2 SP
3. Bo + B1 Hp + B2 SP + B3 VOL

4. Bo + B1 Hp + B2 SP + B3 VOL + B4 WT

==========

Fit all the above models and calculate the R square and RMSE and identify the best fit model from above list.

	R^2	RMSE
1. Bo + B1 Hp	0.53	6.25
2. Bo + B1 Hp + B2 SP	0.53	6.2
3. Bo + B1 Hp + B2 SP + B3 VOL	0.77	4.35
4. Bo + B1 Hp + B2 SP + B3 VOL + B4 WT	0.77	4.35
Hp + Vol	0.75	4.53

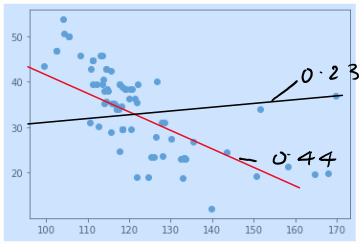
==========

Multicollinearity:

If two X variables are highly correlated among them rather than with Y variable, then multicollinearity issues exists

---> Bo + B1 Hp + B2 SP

Slope Slope



(Hp & SP) = 0.97

Hp & MPG = 0.72

SP & MPG = 0.68

```
In [32]: df.corr()
Out[32]:
```

	HP	MPG	VOL	SP	WT	Y_pred
HP	1.000000	-0.725038	0.077459	0.973848	0.076513	-1.000000
MPG	-0.725038	1.000000	-0.529057	-0.687125	-0.526759	0.725038
VOL	0.077459	-0.529057	1.000000	0.102170	0.999203	-0.077459
SP	0.973848	-0.687125	0.102170	1.000000	0.102439	-0.973848
WT	0.076513	-0.526759	0.999203	0.102439	1.000000	-0.076513
Y_pred	-1.000000	0.725038	-0.077459	-0.973848	-0.076513	1.000000

Finally, we have select that a model should have least RMSE and highest R square with no multicollinearity issues with minimum number of variables

Multicollinearity is very common issues will be there any kind of continuous variables in any types of datasets.

Do we have any specific range to decide that how much of multicollinearity can be accepted.

We have some other measure called VIF

Variance inflation factor (VIF): $\frac{1}{1 - R^2}$

VIF < 5 ---> taking up to those variables for model development is absolutely fine
VIF > 5 and below 10 ---> taking up to those variables for model development is some kind of warning , but can be acceptable
VIF > 10 ---> taking up to those variables for model development is definitely rejected.

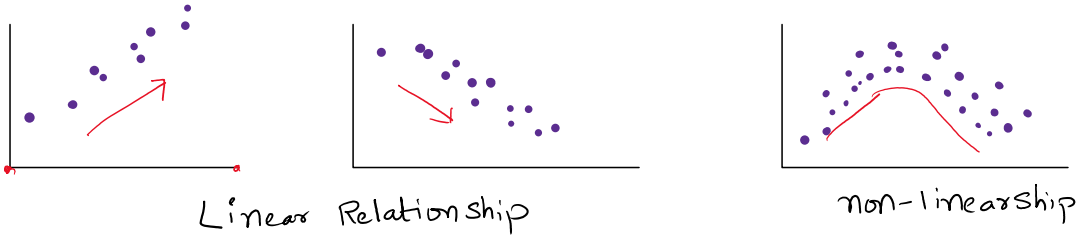
R square ---> the model that we fitted between those two variables R square will be consider

==========

Assumptions for the Linear Regression:

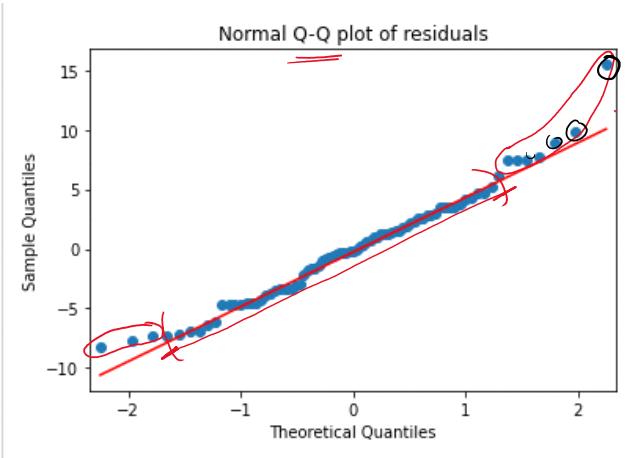
1. When our Target variable is continuous data type

2. the relationship between X and Y should be linear relationship

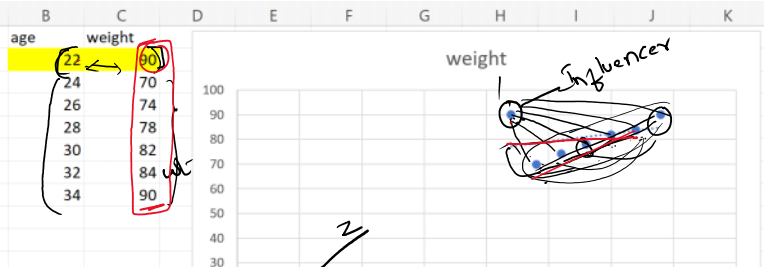


3. No multicollinearity issues

4. Data should follow Normal distribution, specifically errors (y actual - y predictions)



Model deletion diagnostics
Highly influencer points ---->

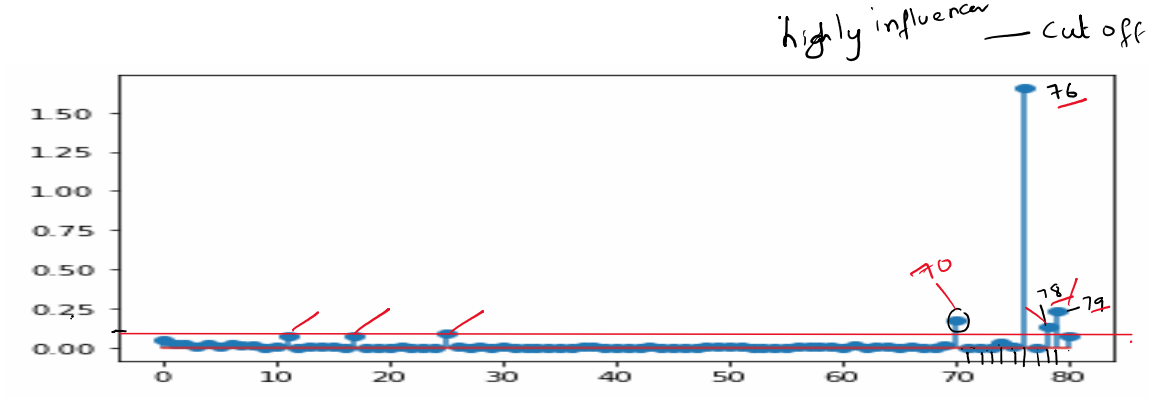


HP vol → MPG (81)

How can we find the highly influence points?

Cook's distance: will calculates the distance of every data point to every datapoint by using formulae.

When we use the python code, it will calculates the distance for every data point. Through that we can understand that which data point are highly influencer.




```
In [19]: df[df["cooks"]>leverage_cutoff]
Out[19]:
     HP        MPG  VOL          SP         WT     cooks
70  280  19.678507   50  164.598513  15.823060  0.178098
76  322  36.900000   50  169.598513  16.132947  1.654415
78  263  34.000000   50  151.598513  15.769625  0.137228
79  295  19.833733  119  167.944460  39.423099  0.230841
```

Box-plot – 1D

After removing the influencer points, we have seen a significant improvement of R square from 75% to 84%

Note that, it is for structured data ---> Rows and columns

1. Framing the problem
2. Collect the data from data base: SQL
3. Exploratory data analysis (Descriptive statistics)

   Central tendency, spread, shape(histogram, bar-graph, pie chart, boxplot, scatterplot)

4. Data cleaning: removing outliers, replacing, null values, any other technical

5. Data transformation: Label encoding

6. Data partition : X and Y

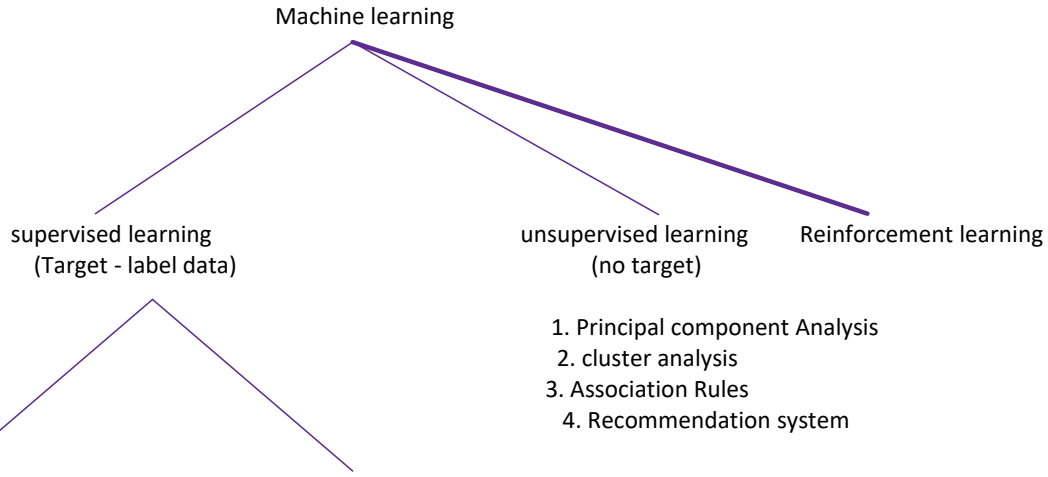
7. selection model

8. Cross validation

9 . metrics

10. deployment

==========



1. Linear Regression
(SLR , MLR)

2. Decision tree
(Decision tree regressor)

3. Support vector machine
(Support vector regressor)

1. Logistic regression (Binary classes)

2. Decision tree
(Decision tree classifier)

3. Support vector machine
(Support vector classifier)

4. KNN classifier

5. Naive Bayes classifier

Metrics:

	R square (0 to 1)	Accuracy score
Loss function:	MSE ()	log loss

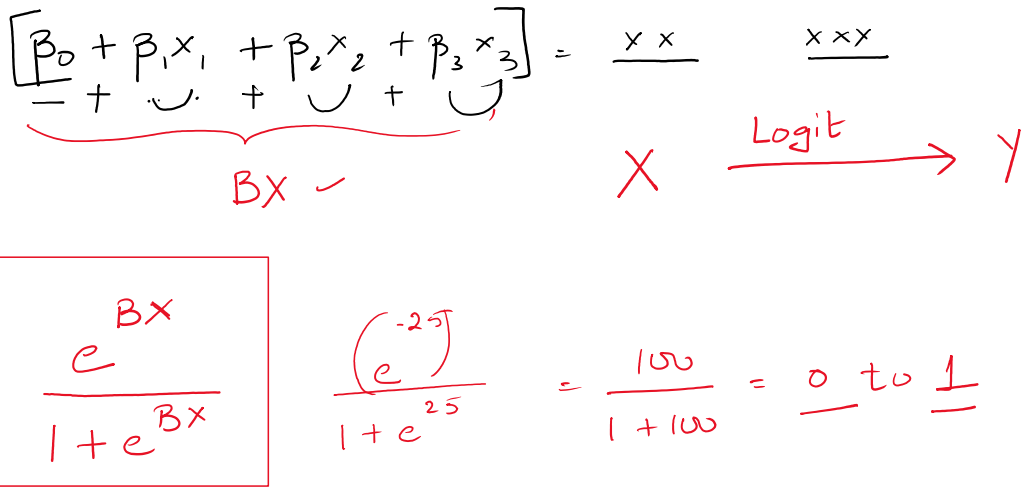
Cross validation
Ensemble methods: bagging, random forests, gradient boosting, Ada boost, XG boost.

==========

Logistic regression:

When our Target variable is binary class:

Gender, Diab:(yes/no), cancer: (yes / no), covid (pos/neg), loan(approved/rej), emi(yes/no), insurance(cliam/not), Purchase: (success/failure), ticket(travelled/ not), purchased(liked/disliked)……..



We are constructing a regression model using logit function , hence we will call it as Logistic regression

F	G	H
Y		Y_pred
1	250	1
0	25	1
1	5	0.993
0	1.2	0.769
1	-5	0.007
0		

==========

Confusion matrix:

2 x 2 table

confusion matrix			y pred		
		TN	0	1	FP
	y actual	0	4	1	
		1	1	4	
		FN			TP
Type II Error					

Type I Error

Accu

1.

$$\text{Accuracy score:} \quad \frac{TN + TP}{N}$$

==========

2. Sensitivity / Recall:

From the overall **Actual positives** how much our model successfully predicted them as **positives**.
Percentage of positives that are successfully classified as positives.

True positive Rate
TPR = TP/(TP + FN)

If FN decreases Sensitivity increases, If FN increases, Sensitivity Decreases

==========

3.Specificity:

From the overall **Actual Negatives** how much our model successfully predicted them as **Negatives**.

Percentage of negatives that are successfully classified as Negatives.

True negative Rate
TNR = TN/(TN + FP)
If FP decreases Specificity increases, If FP increases, Specificity Decreases
==========

4. Precision:

from the model predicted positives what Percentage of people are real positives.
The approach here is to find what percentage of the model's positive (1's) predictions are accurate.
Precision is calculated as the number of correct positive predictions (TP) divided by the total number of positive predictions

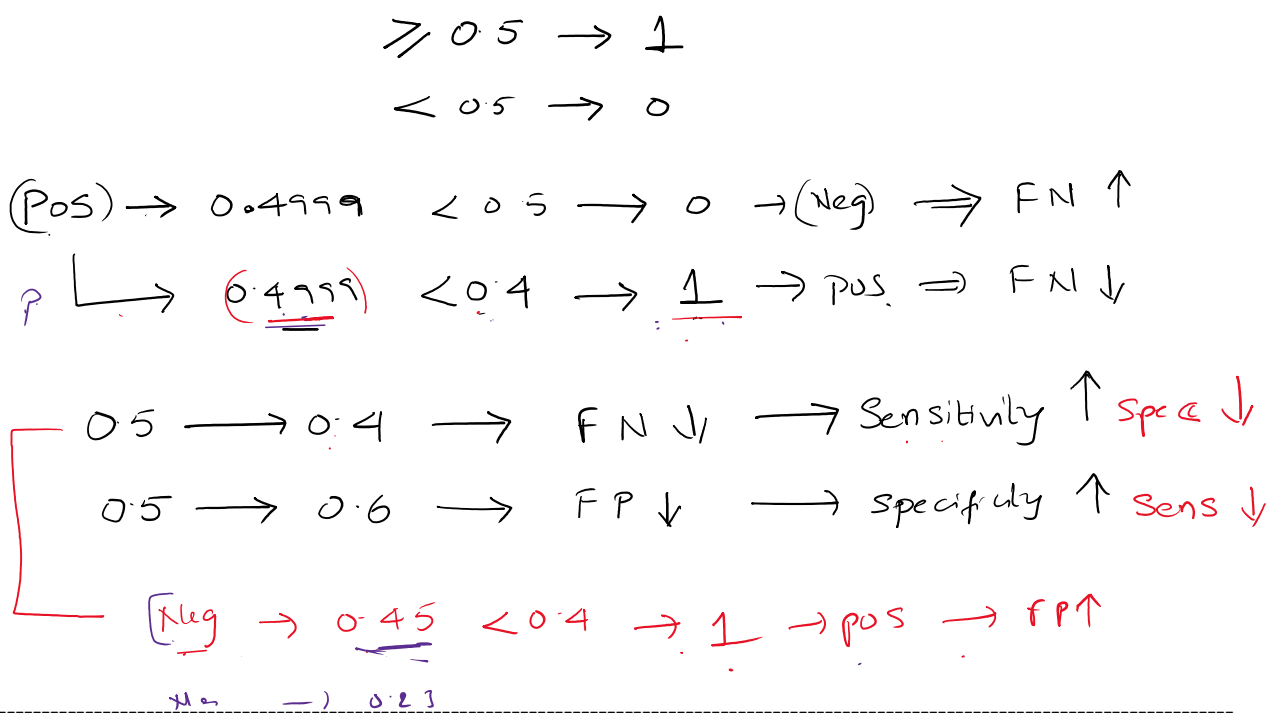
(TP) / (TP + FP)

5. F1. score: A good model should have a good **precision** as well as a high r**ecall\sensitivity**.
So ideally, I want to have a measure that combines **both these aspects** in **one single metri**c –the F1 Score.

F1 Score = (2 * Precision * Recall) / (Precision + Recall)

==========

Can we change the cut off ?



==========

ROC CURVE:

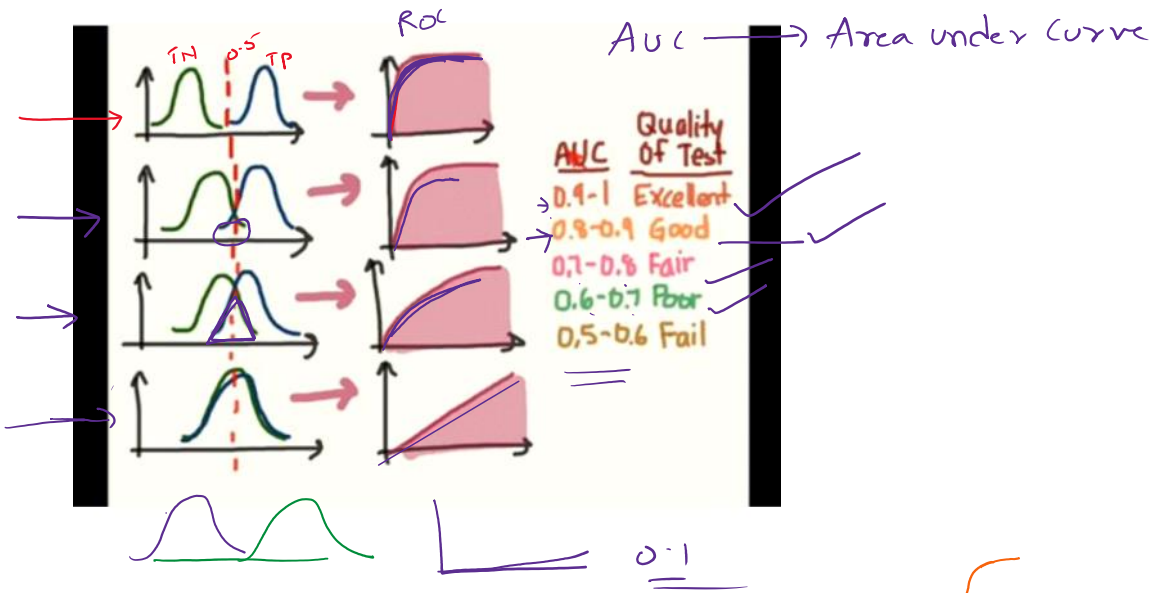
- Receiver operating characteristic (ROC) curve can be used to understand the overall performance (worth) of a logistic regression model.

  Extending the above two-by-two table idea, rather than selecting a single cutoff, we can examine the full range of cutoff values from 0 to 1. For each possible cutoff value, we can form a two-by-two table.

- TNR = specificity

- 1 - TNR = FPR
- Plotting the pairs of sensitivity versus one minus specificity (True positive vs false positive) on a scatter plot provides an ROC (Receiver Operating Characteristic) curve.

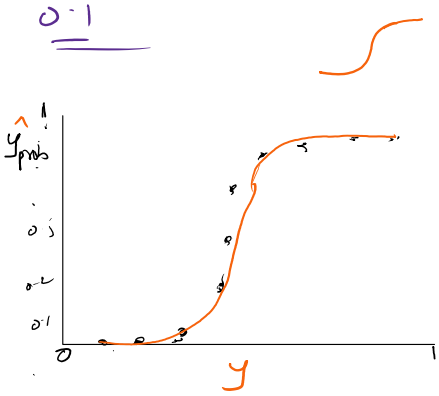
  This will calculates AUC value, Area Under Curve



Log loss for binary classifier:

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} * \ log(p_{ij})$$

y	$\hat{y}$	y
0	0.23	1
1	0.38	0
0	0.13	0
1	0.91	1



==========

Data Transformations:

To make our data in to our convenience

We have several types of transformation

1. Standardization → −3 to +3
2. Scaling / Normalization → 0 to 1
3. One hot encoding
4. Label encoding.

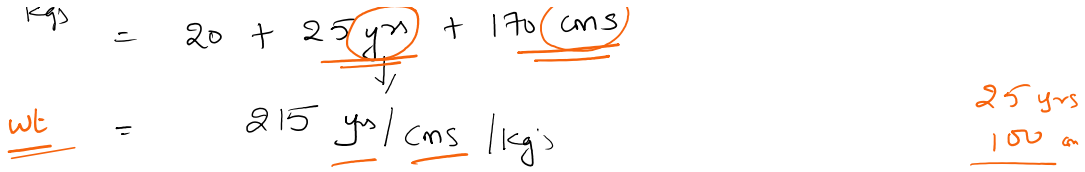
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

(Age) (Ht) (G)

wt =

$$75\ kgs = 20 + (1 \times 25\ yrs) + (1 \times 170\ cms) + (1 \times M)$$

$$= 20 + 25\ yrs + 170\ cms$$



We have to remove the units from the variables, such that it becomes unit free variables.
If we apply standardization on the X variables, then it becomes unit free

We have an alternative formulae will removes the units from the data and re-distributed whole data in between -3 to +3 only.

Z distribution. Transforming from the its original location to somewhere in between -3 to +3 sigma, is called Standardization

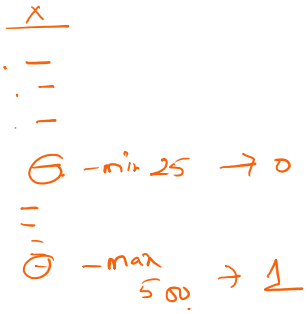
$$Z = \frac{X - \mu}{\sigma} \qquad \mu = \text{Mean}, \quad \sigma = \text{SD}$$

Sklearn --> standard scaler

2. Scaling / Normalization:

It will also removes the units and adjust the data in between 0 to 1.

$$\frac{x_i - x_{min}}{x_{max} - x_{min}} \Rightarrow 0 \text{ to } 1$$



Note that,
1. these two techniques are applicable for continuous variables only.
2. Any one technique can be applied during the machine learning training process.

How about for categorical variables

Usually we may have too few or too many categories.

When we have few categories

Ex: Gender

If we have apply one hot encoding variables

B	C	D	E	F	G	H	I
gender	G_female	G_male		education	BCA	Btech	MCA
male	0	1		Btech	0	1	0
Female	1	0		BCA	1	0	0
male	0	1		MCA	0	0	1
Female	1	0		Btech	0	1	0
male	0	1		BCA	1	0	0
Female	1	0		MCA	0	0	1
Female	1	0		BCA	1	0	0
male	0	1		MCA	0	0	1
male	0	1		MCA	0	0	1

Label encoding:

C	D
education	LE
Btech	1
BCA	0
MCA	2
Btech	1
BCA	0
MCA	2
BCA	0
MCA	2
MCA	2

Apart from these we have other types of transformations
Ex: log transformation, square transformation, square root transformation….

==========

Our whole data as already divided in two categories as x and y
Totally we will 4 data sets

X Y

Ex: (1000, 10) ----> (1000, 9) (1000, 1)

70% training data and 30% of test data 60:40, 65:35, 75:25, 80:20

Training data (700,9) --> X_train (700, 1) ---> (Y_train)
Test data (300,9) ---> X_test (300 , 1) --> Y_test

==========

Cross validation:

Validating our model not just for one time, we are repeating the same process of calculating of both training and testing results with different set of values under random states, so that we can come to know which one accuracies are highly repeated. So that we can finalized at some values.

Shuffle split cross validation:

Fix = 100

Repeat the model fitting for 100 times and calculate the training and testing for 100 times, and take average result from them.

When our data is in small or medium sized data sets, the above method is suitable.

K - fold Cross validation:
Which is suitable for larger datasets.

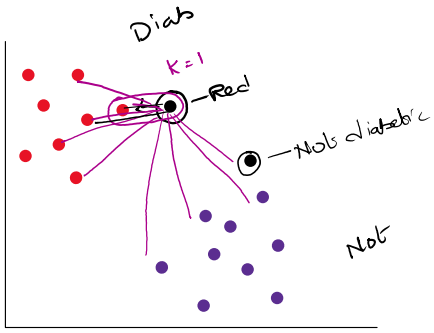
K = 5

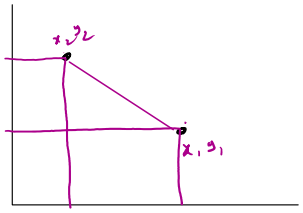
(1000,10)
1000/5 = 200

A	B	C	D	E	F	G	H
1003		k =5					
		1	2	3	4	5	
	201	Test	Train	Train	Train	Train	
	201	Train	Test	Train	Train	Train	
	201	Train	Train	Test	Train	Train	
	201	Train	Train	Train	Test	Train	
	201	Train	Train	Train	Train	Test	
	800	train_acc	train_acc	train_acc	train_acc	train_acc	avg(train)
	200	test_acc	test_acc	test_acc	test_acc	test_acc	avg(test)

K - Nearest Neighbourhood classifier

It will be identified as which nearest data point to its position accordingly it will predicts them.





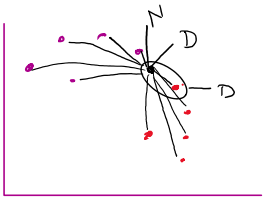
How our machine has been trained here?

It will calculates the distance between every data point to every data point.

Euclidian distance: $\sqrt{[y_2 - y_1]^2 + [x_2 - x_1]^2}$

K = 1

We will prefer who is the first nearest data point



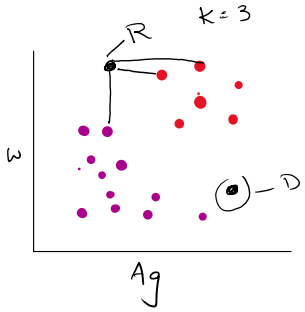
K = 2

First two nearest data points

If both the data points are belongs to same group, then it will predict the same color accordingly.
If one of them is nearer to red and one of them is nearer to blue , random choice

K = 3,
First three nearest data points
If all the 3 data points are belongs to same group, then it will predict the same color accordingly.
If one of them is red and other two are blue ---> it will predicts as blue
If one of them is blue and other two are red ---> it will predicts as red

If all the 3 data points are belongs to same group, then it will predict the same color accordingly.
If one of them is red and other two are blue ---> it will predicts as blue
If one of them is blue and other two are red ---> it will predicts as red



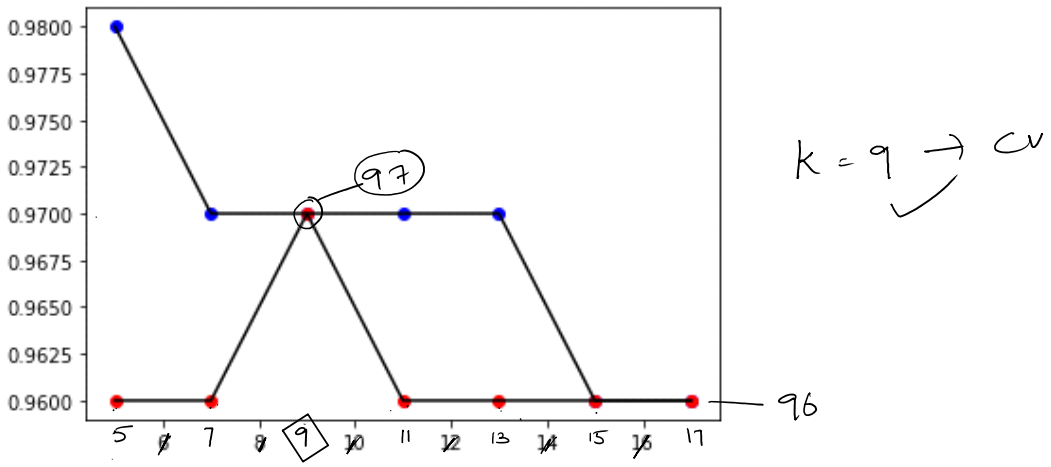
K = 4 --->

Avoid always even numbers for K values
Try with k = 5, 7, 9, 11, 13, 15, 17…

Fit our model with every k value and see the results of both training and testing using cross validation

For what k value we are getting highest test accuracy those k value is final.

	age	weight			1	2	3	4	5	6	7	8	9	10
1	24	68	No	1	0									
2	28	74	No	2	2.6	0								
3	32	78	No	3	4.2	1.4	0							
4	24	76	No	4	4.3	5.8	2.6	0						
5	30	70	No	5	7.8	4.9	5.6	4.7	0					
6	48	90	Yes	6	1.5	1.6	4.3	2.6	4.8	0				
7	58	95	Yes	7	8.9	4.5	4.6	7.6	4.9	1.5	0			
8	44	96	Yes	8	4.7	5.9	8.6	8.4	2.9	1.6	4.6	0		
9	56	88	Yes	9	4.7	6.9	5.9	4.7	5.9	6.3	5.9	5.7	0	
10	60	92	Yes	10	1.3	2.8	6.8	4.9	6.7	5.9	9.3	9.4	4.7	0
			?											
11	30	80	No	11	8.9	4.51	4.6	7.6	4.7	6.9	5.9	4.58	5.9	6.7



---

	metrics
Target variable ---> continuous --> regression model	R square (scores) and RMSE (loss function)
Target variable ---> categorical --> classification model	accuracy score (scores) and log loss (loss function)

Cross validation is common for both methods

While applying cross validations either we can use any of the metrics from R square or RMSE for regression models
Similarly, either we can use any of the metrics from accuracy score or log loss for classification models

---

If we are choosing scores (regression/classification) ---> Training accuracy and Test accuracy

If we are choosing loss functions (regression/classification) ---> Training error and Test error

---

If we received for a model where it has training accuracy = 60% and test accuracy = 60% ---> model will be called as "**Under fit model**"
If we received for a model where it has training accuracy = 95% and test accuracy = 91% ---> model will be called as "**Best fit model**"
If we received for a model where it has training accuracy = 98% and test accuracy = 68% ---> model will be called as "**over fit model**"

Ex: weight
If we received for a model where it has training error = 30 and test error = 30 ---> model will be called as "**Under fit model**"
If we received for a model where it has training error = 3 and test error = 3 ---> model will be called as "**Best fit model**"
If we received for a model where it has training error = 0 and test error = 10 ---> model will be called as "**over fit model**"

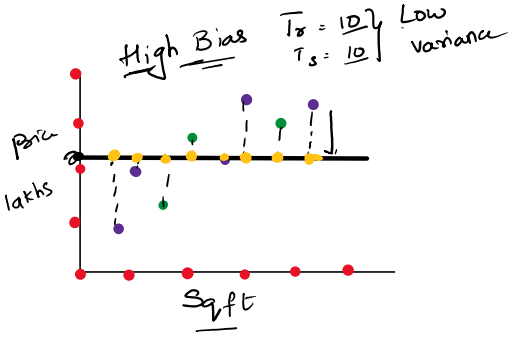
If model is under fit --> still need improvement --> add some more samples and variables to improve the performance

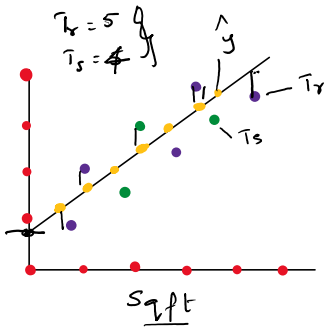
If model is best fit --> still need improvement --> no changes

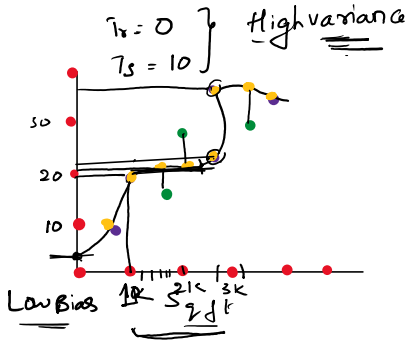
If model is over fit --> still need improvement -->
Reasons behind the model becomes overfit

1. **Data may have outliers still left**
2. **Too many variables are added for model fitting, some of them could be useless also, less significance variables also might be added.**

What happens if I accepts the overfit model?







$y = \beta_0 + \beta_1 x$

$y = \bar{y}$ ($\beta_1 = 0$) — Bias

Null Model

Underfit model

Min. Bias

Min variance

Best fit

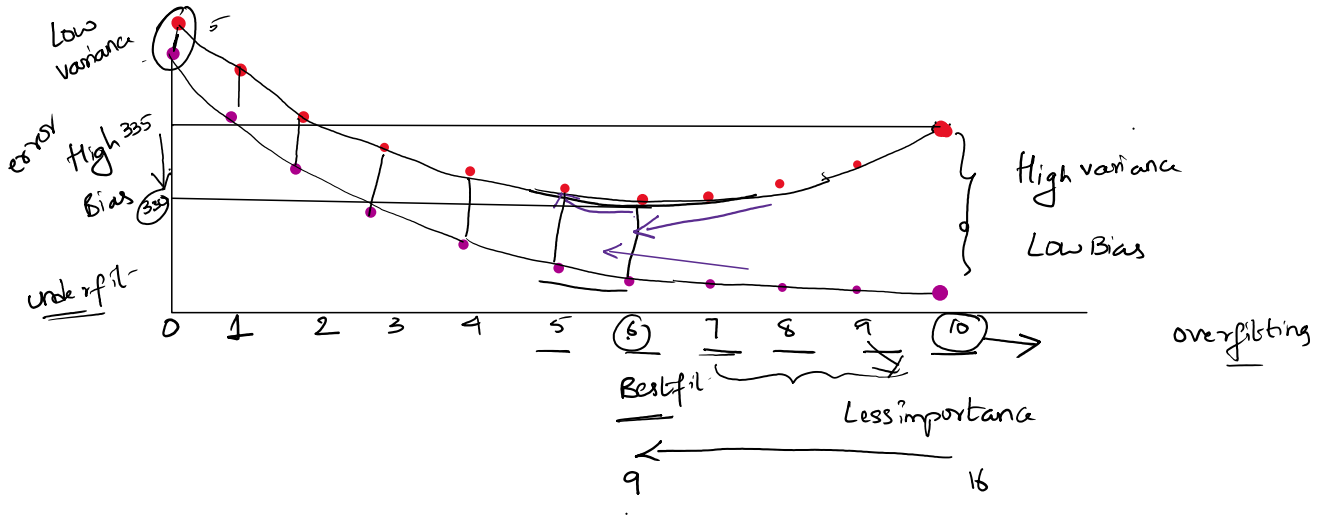
How to control the overfitting?

We have to identify the unnecessary variables and remove from the model fitting

If we have less variables such as below 10 X variables , we can use correlation method

If we have more X variables such as 20 to 40 --> Regularization (Ridge regression and Lasso regression)

If we have more X variables such as greater than 100 --> principal component Analysis (unsupervised learning)



Regularization (Ridge regression and Lasso regression)

It will identifies those less significant variables from the overfitted models and gives us hints which are the variables can be removed.

Ridge regression:

$$M.S.E = \underbrace{\frac{\sum_{i=1}^{n}[y_i - \hat{y}]^2}{n}}_{\text{error}} + \text{Bias (little)} + \text{——} \longrightarrow \text{How much of Bias ?}$$

$$+ \text{Sum of Square Coefficients}$$

$$+ \lambda [\ \beta_1^2 + \beta_2^2 + \beta_3^2 + \dots\ ]$$

$\beta_1 \downarrow$ $\beta_2 \downarrow$ $\beta_3 \downarrow$

Ex:

$+ 0.1\ [100] = 10$

Bias = 10

$\lambda = 0.1$ ✓

$[\lambda = 0.1, 10]$

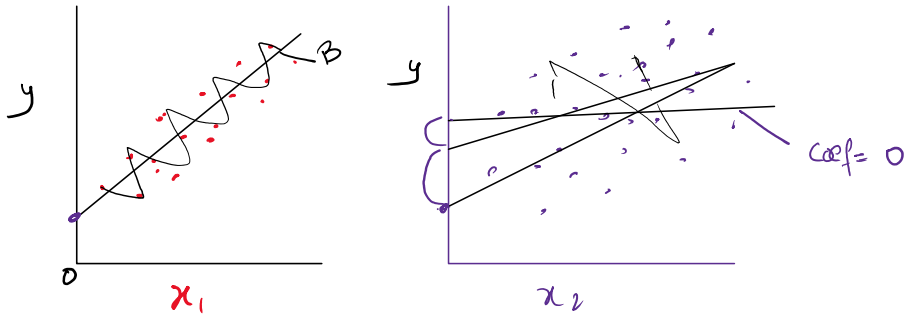
Bias = 10

$+ 10$ ✓ $[1] = 10$

When we apply the ridge regression by using some lambda values, some of the coefficients will be starts shrinking towards zero's
That means beta values will be start converting towards zero's, indirectly those variables are becoming weak, which variables are becoming weak those variables can be removed from the model fitting.
Note that, ridge regression coefficients shrink towards zero's but never becomes 100% zeros.

Whereas same method will be applied for even for Lasso regression as well, with lambda as a "**tuning parameter**"
But under the lasso coefficients will becomes 100% zero's

So using the above any one method, we can start removing the variables which are shrinks towards zero's



overfit underfit

307 = 307

B	C	D	E	F	G
		mod1		mod2	mod3
Linear regression		16X		9X	8x
Training error		300.42		307.74	316.86
Test erro		335.61		330.32	337.92

outliers → EDA → Removed outliers

Lasso regression is helping us to identify the useless variables, from the 16X variables we identified the 7 X variables which are less contributively and we removed from the model fitting.

Principal component Analysis (PCA):

Which is topic of unsupervised learning and comes under "Dimensional reduction techniques"

Dimensions --> variables

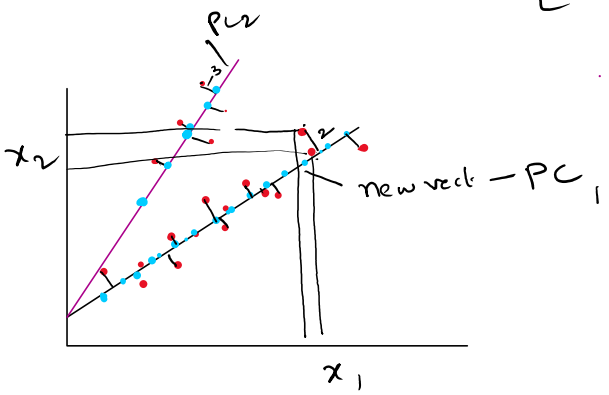
It is transformation technique will be applied on all X variables where entire data will be saved in a new dimensional data set

$A = 6 \times 5$

$B = 5 \times 5$

$A \cdot B = [6 \times 5] \cdot [5 \times 5]$

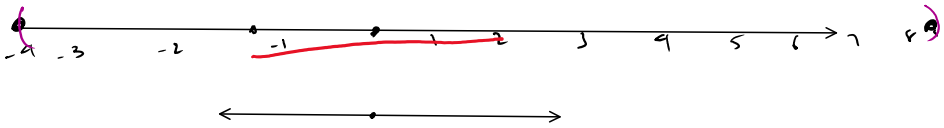
$= [6 \times 5]$ final



useless

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
id	age	heigl	weig	bp	cho		eigen vectors						PC_1	PC_2	PC_3	PC_4	PC_5
1	23	176	68	100	120		1	2	3	3	4		24		10		0
2							3	4	1	2	1		15		15		1
3							4	5	3	2	6		35		12		0

1	23	176	68	100	120	1	2	3	3	4	24	10	0
2						3	4	1	2	1	15	15	1
3						4	5	3	2	6	35	12	0
4						2	3	4	5	6	12	11	1
5						3	4	5	2	3	20	10	0
6											5	11	1
			6 x5					5x5				6x5	
											108	3.5	0.3



B	C	D	E	F	G	H	I
				Lasso Reg			
		mod1		mod2	mod3		
Linear regression		16X		9X	8x		
Training error		300.42		307.74	316.86		
Test erro		335.61	35.19	330.32	337.92		
PCA results							
		6X		5x	7x	4	8
Training error		324.92		327.04	324.74	329.65	324.43
Test erro		339.96	15.04	340.91	341.36	341.62	342.97

Advantages:

1. Dimension can be reduced.
2. Multicollinearity issues will be resolved when we are using PC's
3. Zero correlation will exists in PC's

Apply the PCA method on KNN data set (breast-cancer-wisconsin-data.csv) and see the results
With first 6 components only 96% accuracy.

Note: above all methods are applicable and gives best results only when X variables are continuous

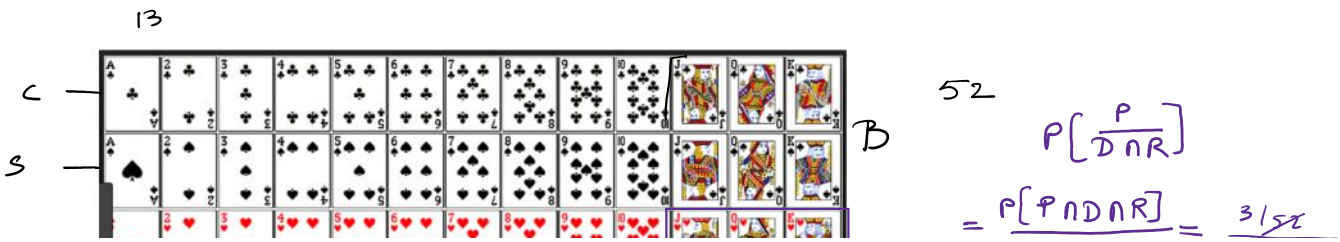
---

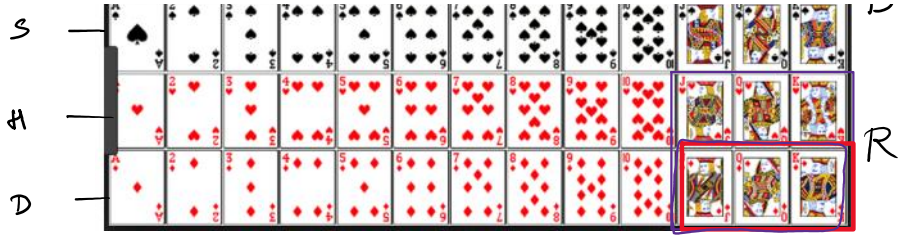
What happens if we have X variables are categorical only or more number of categorical variables if we see in our data

Naive bayes classifier:

Y is discrete: X variable is also discrete

Probability:





$= \frac{P[P \cap D \cap R]}{P[D \cap R]} = \frac{3/52}{13/52}$

$(13 + 12) - 3$

$P[\text{one}] = \frac{1}{52}$, $P[R] = \frac{26}{52}$, $P[P_{ic}] = \frac{12}{52}$

$P[R \underset{\text{and}}{\cap} P] = \frac{6}{52}$, $P[D \underset{\text{and}}{\cap} P] = \frac{3}{52}$, $P[D \underset{\text{or}}{\cup} P] = \frac{22}{52}$

Conditional probability :-

$$P\left[\frac{A}{B}\right] = \frac{P[A \cap B]}{P[B]}$$

① $P\left[\frac{D}{R}\right] = \frac{P[D \cap R]}{P[R]} = \frac{13/52}{26/52} = 13/26 \rightarrow 50\%$

$\rightarrow$ ① $R \xrightarrow{50\%} D$

② $P\left[\frac{R}{D}\right] = \frac{P[D \cap R]}{13/52} = \frac{13/52}{13/52} = 1 \Rightarrow 100\%$

② $D \xrightarrow{100\%} R$

$P[A/B] \times \frac{P[A \cap B]}{P[B]}$; $P[B/A] \times \frac{P[A \cap B]}{P[A]}$

$P[A/B] \times P[B] = P(A \cap B)$; $P[B/A] \times P[A] = P[A \cap B]$

$P[A/B] \times P[B] = P[B/A] \times P[A]$

$$P[A/B] = \frac{P[B/A] \times P[A]}{P[B]}$$

$B_i = 1, 2, 3, \ldots\ldots K$

$$P\left[\frac{A}{B_i}\right] = \frac{P[B_i/A] \times P[A]}{P(B_i)}$$ → Bayes theorem

A	B	C	D	E
1	mobile	charger	earphone	back cover
2				
3	mobile	charger	earphone	
4		charger		
5	mobile	charger		
6		charger		

$P[M] = 3/6$ $\quad P[C] = 5/6$

$P[M \cap C] = 3/6$ $\quad P(M \cap C \cap E) = 2/6$

$P\left[\frac{M}{C}\right] = \frac{P(M \cap C)}{P(C)} = \frac{3/6}{5/6} = 3/5 \Rightarrow 60\%$

$P(C)$ 60% M

$P\left[\frac{E}{M \cap C}\right] = \frac{P(M \cap C \cap E)}{P(M \cap C)}$ $\quad P\left[\frac{C}{M}\right] = \frac{}{P(M)} = \frac{3/6}{3/6}$ (1) $\Rightarrow$ 100%

Y 100 N 0

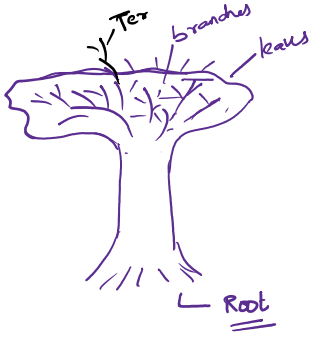
$= \frac{2/6}{3/6} = 2/3 = 66\%$

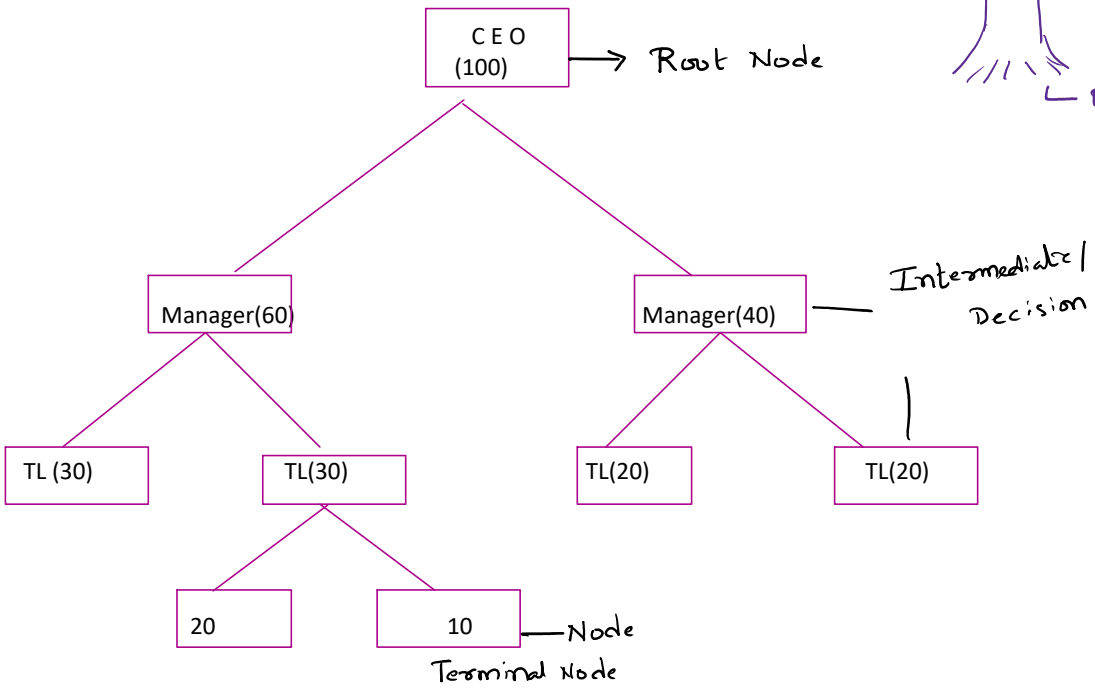
M $\xrightarrow{100\%}$ C

Decision Trees:

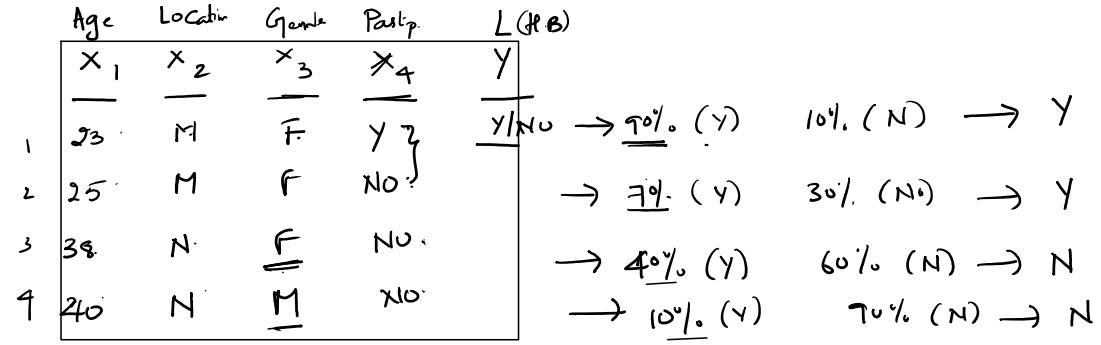
It is works for both classification and regression problems.
It also works for both linear and non-linear data sets.

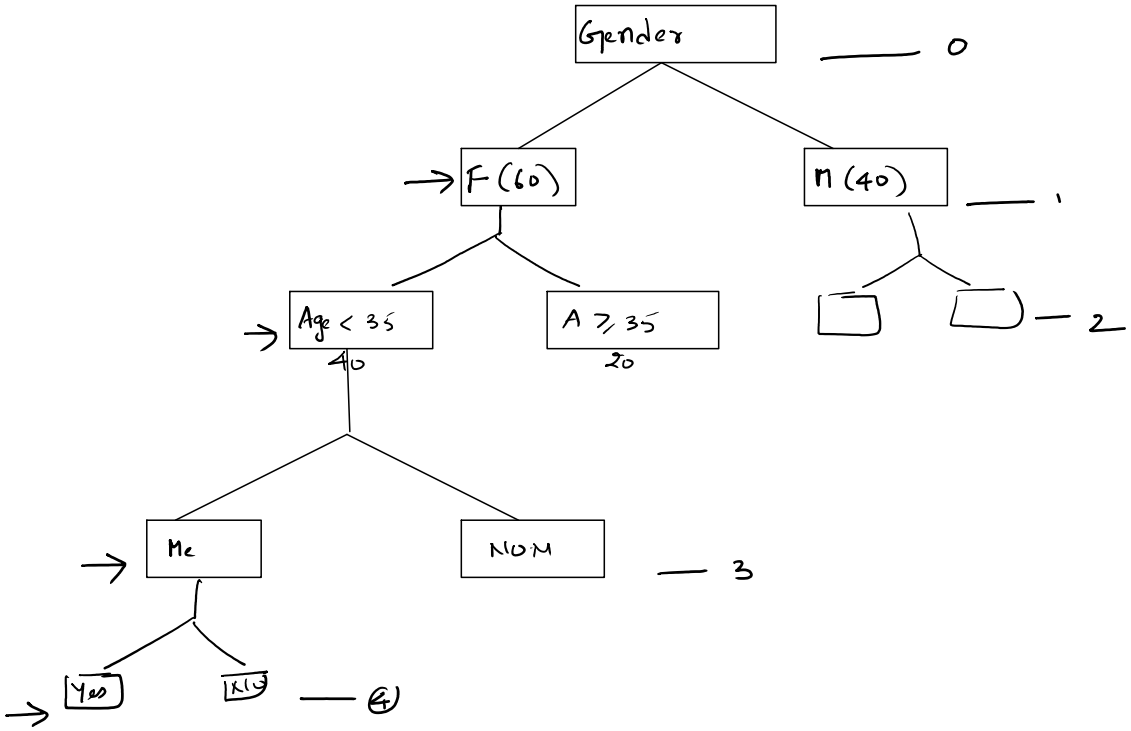
Tree





Age	Location	Gender	Partp	L (H B)
X_1	X_2	X_3	X_4	Y





$$P\left[\frac{P=Y}{P[M]}\right] = 70\% \qquad P[P=N] = 30\%$$

$$\frac{}{P[Ag<35]}$$

$$\frac{}{P[G=F]}$$

Decision tree has contains an inbuilt algorithms are their inside for variable selections

1. Gini Index
2. Information gain or entropy.

These mathematical methods will chose the most important variables from the overall data and fits their positions in the decision tree diagram above.

Sklearn.tree dtclassifier

Dtclassifier(Gini or entropy) ----> variable selections just like our correlation and gives them priority and arranged their order accordingly.

There is higher chances of model becomes overfitted.

Cross validation: Training accuracy score: 1.0
Cross validation: Test accuracy score: 0.73
Cross validation: trianing loss: 17.41
Cross validation: test loss: 17.0

Pruning:

It is a technique where it is controlling the max depth parameter and observing the test accuracy how much can be improved. This method is also controls overfitting.

We can try with both Gini and entropy to see that where we get the highest test accuracy.

```
for i in range(1, 501):
    X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=0.20, random_state=i, stratify=Y)
    model = DecisionTreeClassifier(criterion='gini',max_depth=7) # 'entropy'
    model.fit(X_train,Y_train)
    y_pred_train = model.predict(X_train)
    y_pred_test  = model.predict(X_test)
    training_accracy.append(accuracy_score(Y_train,y_pred_train))
    test_accuracy.append(accuracy_score(Y_test,y_pred_test))
    training_loss.append(log_loss(Y_train,Y_pred_train))
    test_loss.append(log_loss(Y_test,Y_pred_test))

print("Cross validation: Training accuracy score:", np.round(np.mean(training_accracy),2))
print("Cross validation: Test accuracy score:", np.round(np.mean(test_accuracy),2))
print("Cross validation: trianing loss:", np.round(np.mean(training_loss),2))
print("Cross validation: test loss:", np.round(np.mean(test_loss),2))
```

```
Cross validation: Training accuracy score: 0.95
Cross validation: Test accuracy score: 0.73
Cross validation: trianing loss: 17.56
Cross validation: test loss: 17.71
```

Ensemble methods:

Ensemble: grouping

Grouping the many models together

It has two types:

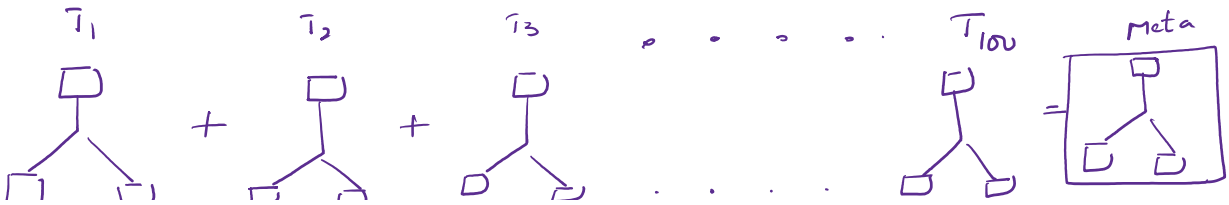
Parallel ensemble methods (Bagging)
1. Bagging 2. Random forests

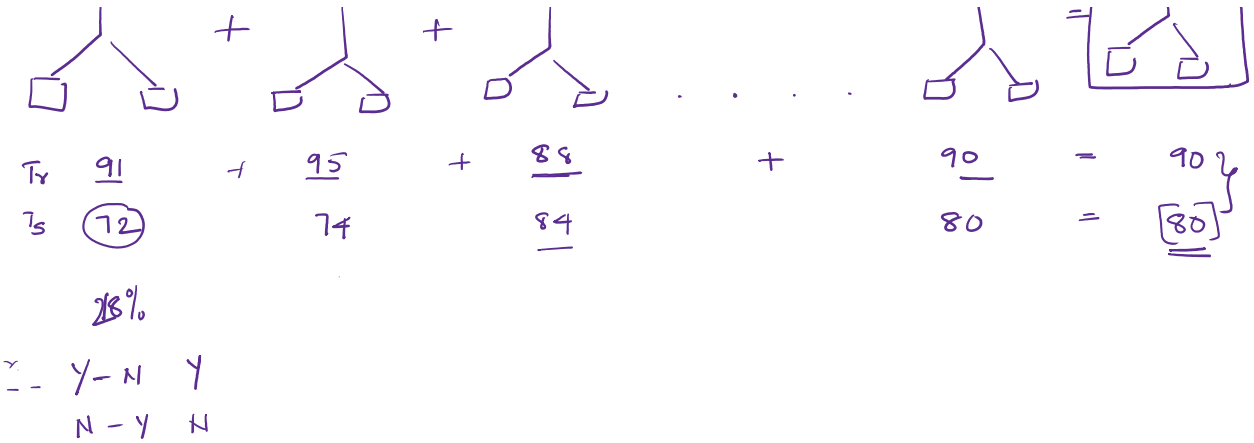
Sequential ensemble methods (Boosting)

1. Gradient Boosting 2. Ada Boost 3. Extreme gradient boosting.

Ensemble methods are a set of techniques in machine learning that combine multiple models to improve predictive performance.

The core idea behind ensemble learning is that a group of weak learners can come together to form a strong learner, reducing variance, bias, and improving accuracy.





- **Improved Accuracy:**
- Combining multiple models reduces the chance of making an incorrect prediction.
- **Reduced Overfitting:** Averaging multiple models prevents a single model from capturing noise in the training data.
- **Reduced Variance:** Multiple models balance each other's errors, leading to a more stable prediction.
- **Handles Complex Problems:** It works well for non-linear and complex decision boundaries.

Parallel methods:

Every tree will grow independently, it depends up on models that we are going to initiate

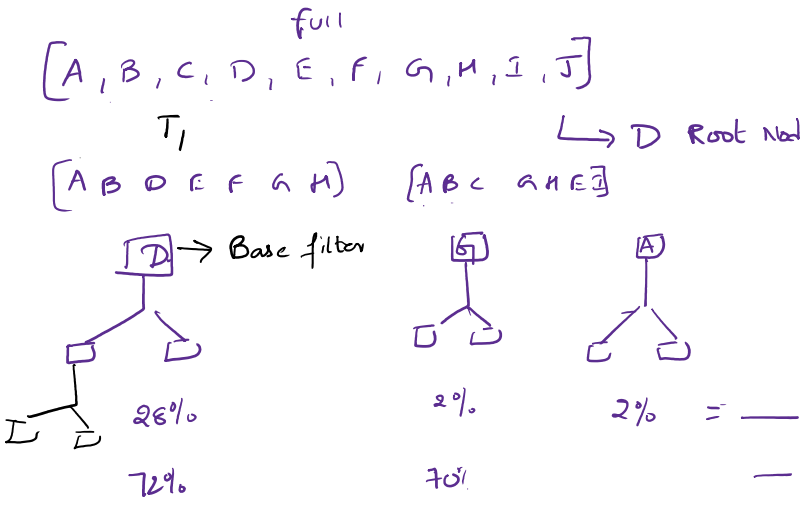
Bagging:

Base estimator: Decision tree(gini)

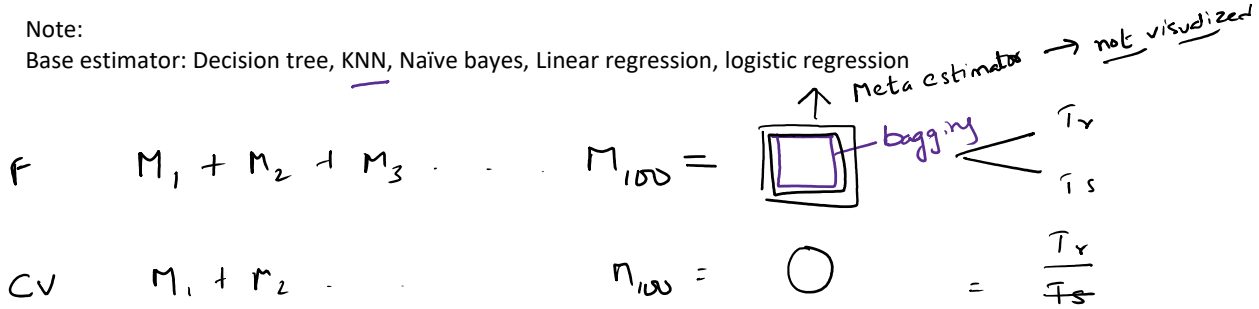
N estimators = 100

Ex: (1000, 10)
Max samples = 0.6 --> (600, ..)

Max features = 0.7 ---> (...,7)

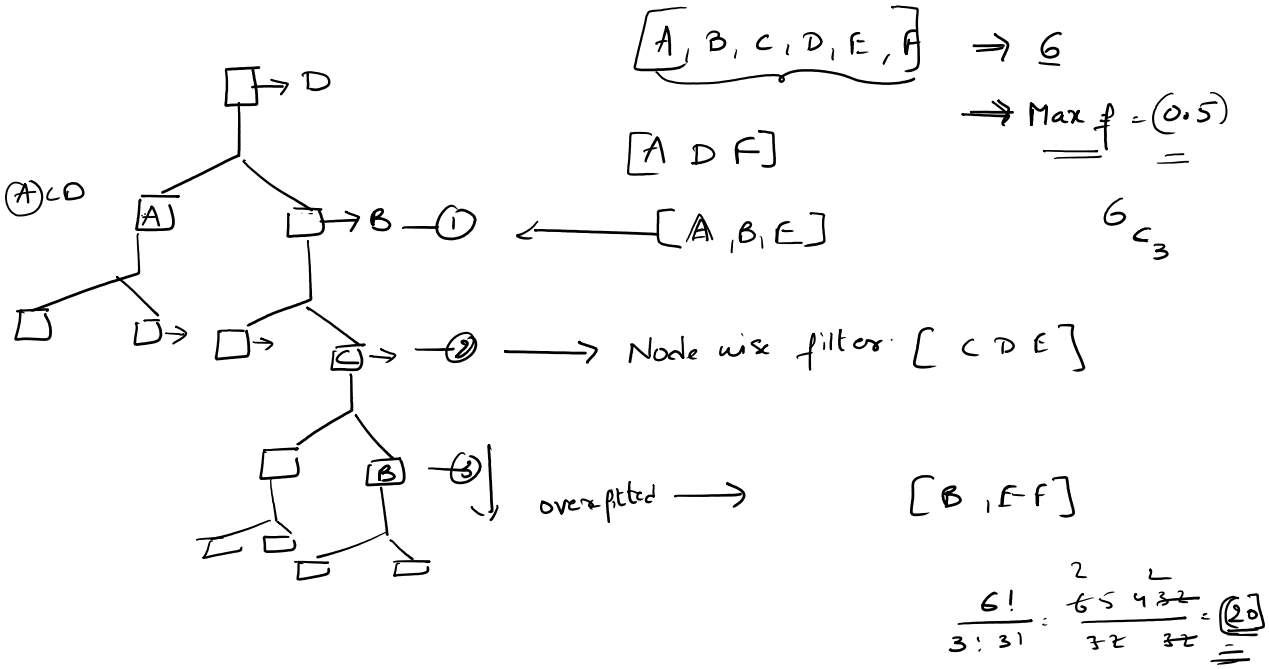


Note:
Base estimator: Decision tree, KNN, Naïve bayes, Linear regression, logistic regression



Random forests:
It is one of the special feature from the bagging only.

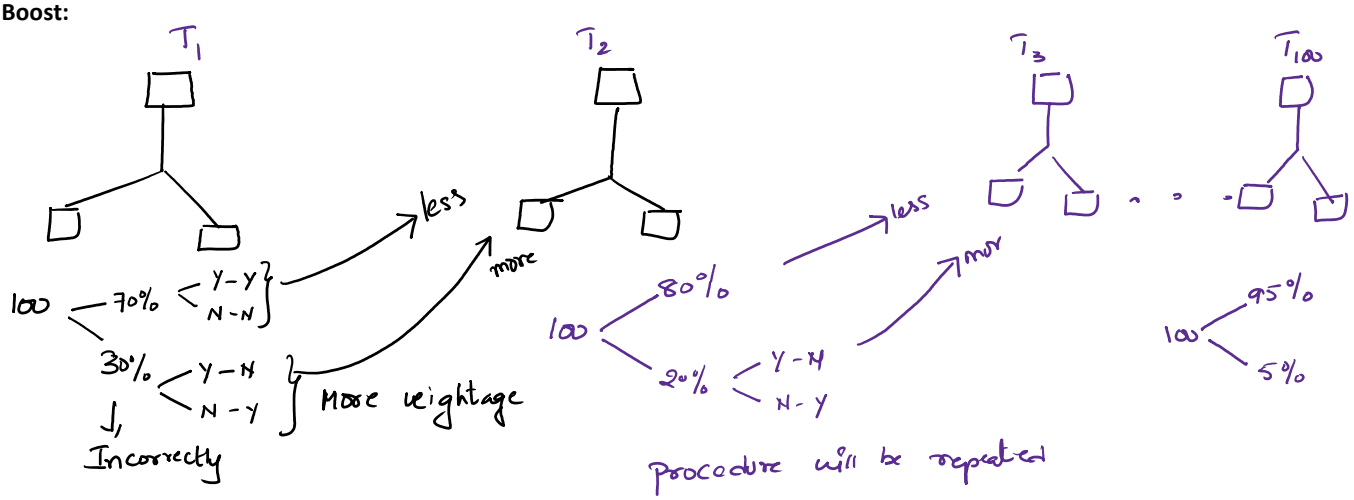
Random forests will works only for Decision trees, it means we cannot change the estimator name.
The biggest difference in between bagging (using DT as inbuilt) and Random forests(DT) is development of tree will be not exactly same.



==========

Sequential ensemble methods: Boosting algorithms

Boost:



Every tree will be works on the errors of the previous tree predictions

Boosting concepts contains many methods:

1. Gradient Boosting
2. Ada Boost
3. Extreme Gradient Boosting.

==========

Gradient Boosting:

Steps to fit a Gradient Boosting model

1. Fit a simple linear regressor or decision tree on data ***[call x as input and y as output]***
2. Calculate error residuals. Actual target value, minus predicted target value ***[e1= y - y_predicted1]***
3. Fit a new model on error residuals as target variable with same input variables ***[call it e1 _predicted]***
4. Add the predicted residuals to the previous predictions
   [y_predicted2 = y_predicted1 + e1_predicted]
5. Fit another model on residuals that is still left. i.e. ***[e2 = y - y_predicted2]*** and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant. Overfitting can be controlled by consistently checking accuracy on validation data

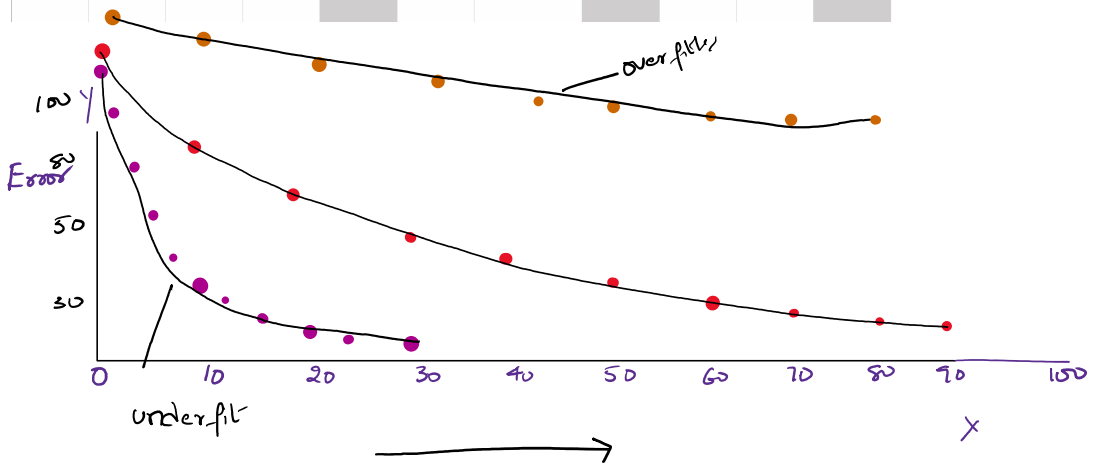
Mean square error

$$\frac{\Sigma [y_i - \hat{y}]^2}{n}$$

Mean absolute error

$$\frac{\Sigma |y_i - \hat{y}|}{n}$$

A	B	C	D	E	F	G	H	I	J	K
id	age	weight	F = Ypre	e1	e1_pred	Final=D + F	e2	e2_pred	Final=G+	e3
1	22	66	68	-2	2	70	-4	-2	68	-2
2	26	76	80	-4	3	83	-7	-5	78	-2
3	32	74	79	-5	-2	77	-3	1	78	-4
4	34	78	85	-7	-4	81	-3	1	82	-4
5	40	86	80	6	3	83	3	2	85	1
				24			20			13

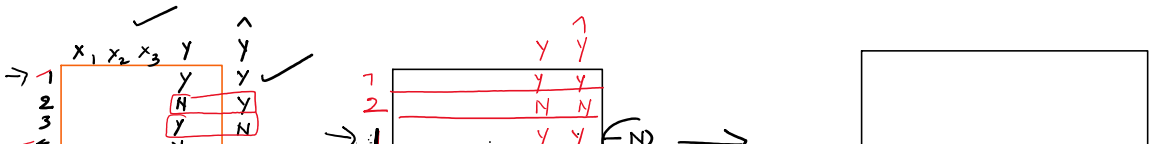


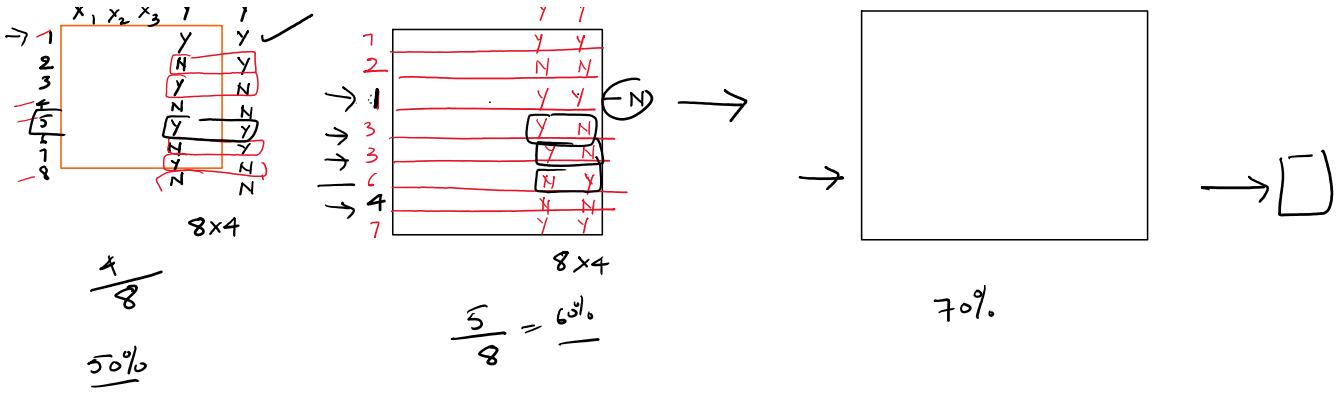
Learning rate:

Using up this learning rate we can make control the either at underfitting model and overfitting model , using learning rate we can find the best fit model

---

Adaptive Boosting algorithm:





Note:
1. Bagging and adaboost can be used for either classifiers/regressor not only for decision tree but also applicable for other machine learning models.

2. Random forest and gradient boosting can be worked only for decision trees

---

Extreme gradient boosting (XG Boost):

STEPS FOR XGBOOST:

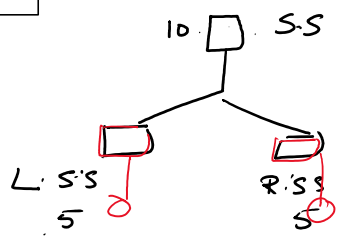
$y = 1 - \hat{y}$
$0 - \hat{y}$

Step1. Consider initial prediction 0.5 $\hat{y}$
Step2. Split the tree based on independent features just like our regular decision tree
Step3. calculate similarity weigh for each leave **similarity score** = (Σ Residuals)2/(Total residuals+ λ)
if you are using classification then
→ **similarity score** = (Σ Residuals)2/([Previous Probability * (1 - previous probability)]+ λ)
Step4. calculate **Gain** = Left Similarity score + Right Similarity score - Root Similarity.
Step5. Based on the highest Gain value tree will start expanding itself.
Step6. For Pruning our tree we will introduced Gamma (eta) values.
Step7. Calculate output value = sum of residuals / (Total residuals + λ)
Step8. New predictive value = Initial value (Y_pred (0.5)) + **Learning Rate** * Output value of node

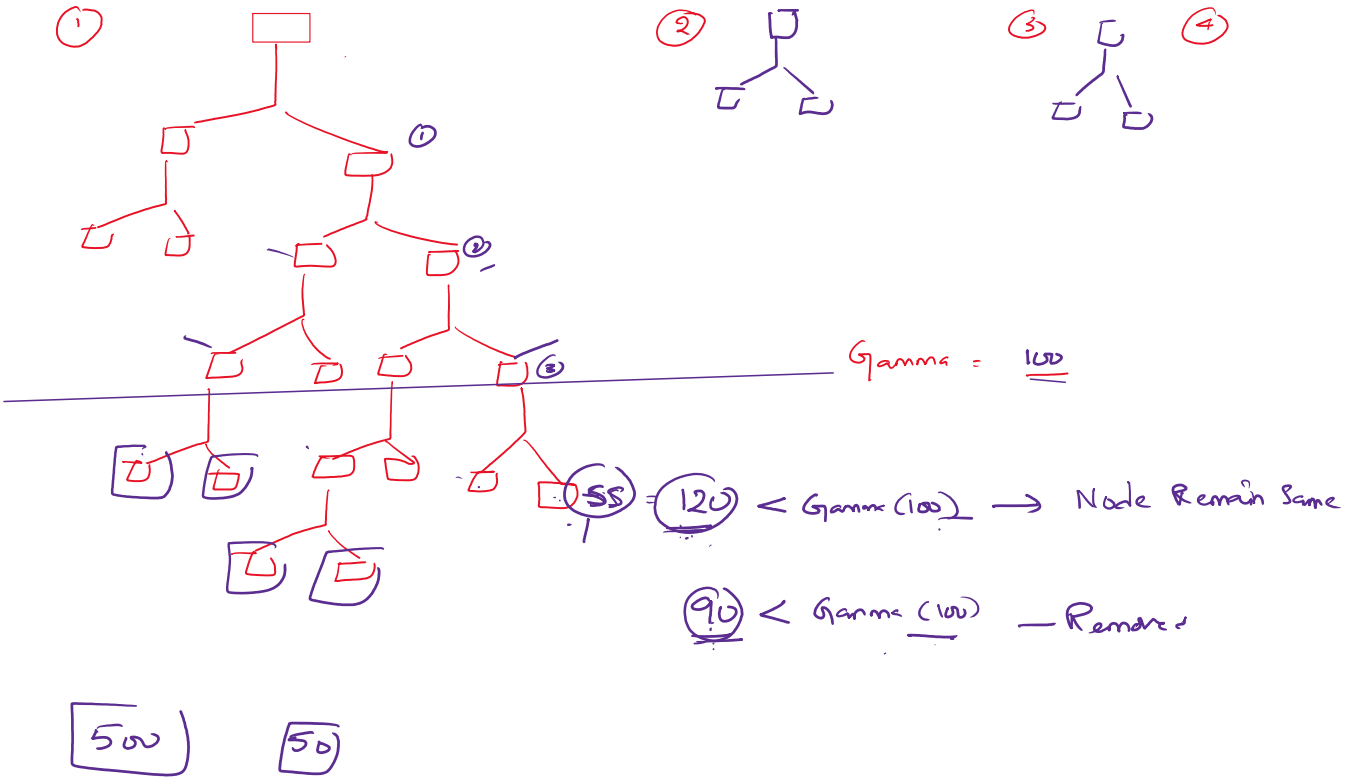
1) λ 2) Gamma 3) LR 4) N_estimators

$x_1\ x_2\ x_3\ y\ \hat{y}$ — 0.5, 0.5, 0.5, 0.5

$R \left[\frac{\Sigma [y_i - \hat{y}]^2}{10 + \lambda}\right]$ $\lambda = 1$



Gain = (L + R) − R



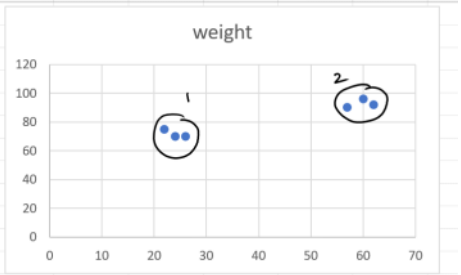
When we are not able to decide the proper gamma values, change the similarity score, how we can change it, it we change the lambda value SS will gets affect

==========

Cluster analysis:
Its comes under unsupervised learning, where there is no Target variable.

Cluster: it's a group,
Data points which are having similar interests are making them in to one cluster. It means data points to be nearest to each other

id	age	weight	cluster		Y
1	22	75	1	Juniors	0
2	57	90	2	Seniors	1
3	24	70	1	Juniors	0
4	62	92	2	Seniors	1
5	26	70	1	Juniors	0
6	60	96	2	Seniors	1
7	28	76		Junior	

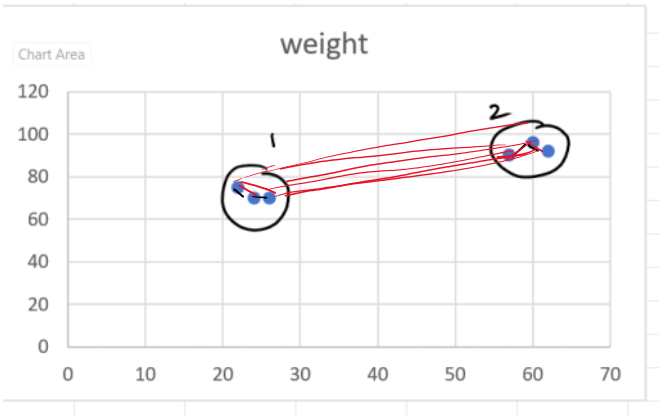


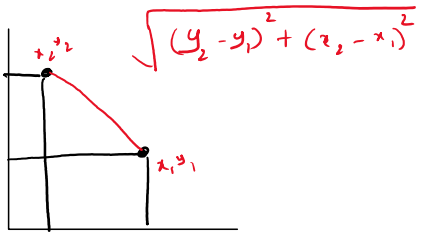
Cluster analysis contains two types:

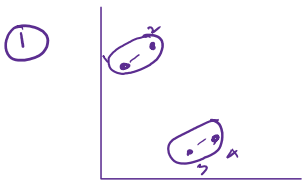
1. Hierarchal clustering (Agglomerative clustering)
2. Non Hierarchal clustering (K-Means clustering)
3. DBSCAN

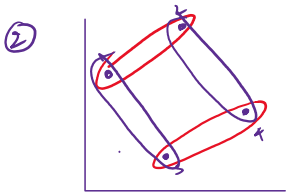
==========

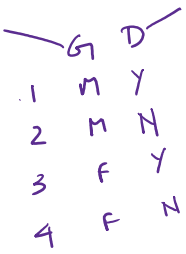
Hierarchal clustering (Agglomerative clustering)



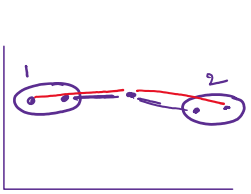








Linkage methods:



1. Single linkage method
2. Complete linkage method
3. Average linkage method
4. Ward linkage method

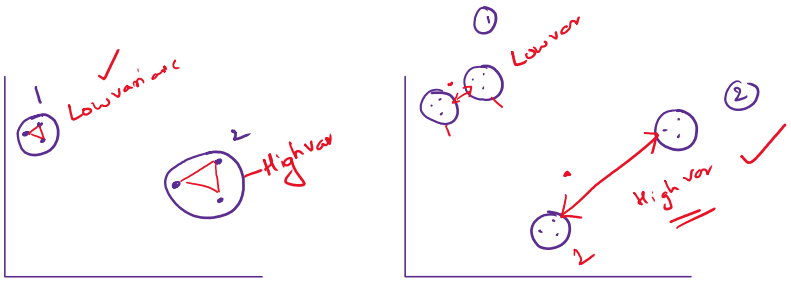
Our job is to identify that for the given data set, which linkage method is making appropriate cluster formation.

How can we recognize that which linkage cluster formation making best decision making skills.

Silhouette score:

How does silhouette score calculates:

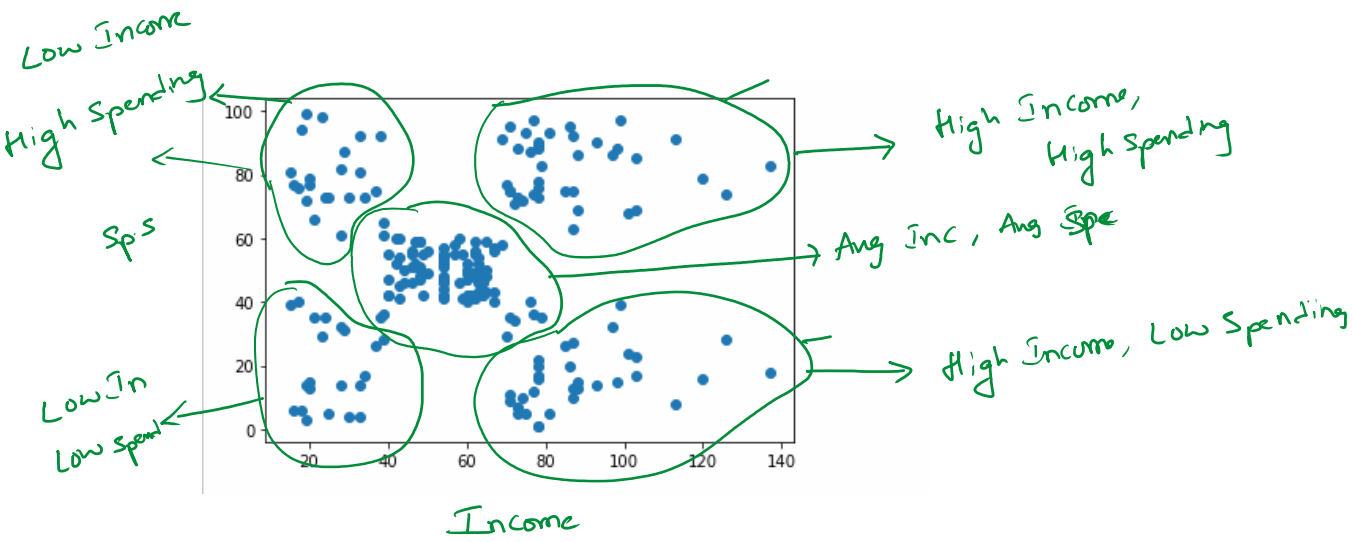
1. It will calculates how much variance is existed with in the cluster.
2. It also calculates how much variance is existed between the clusters.

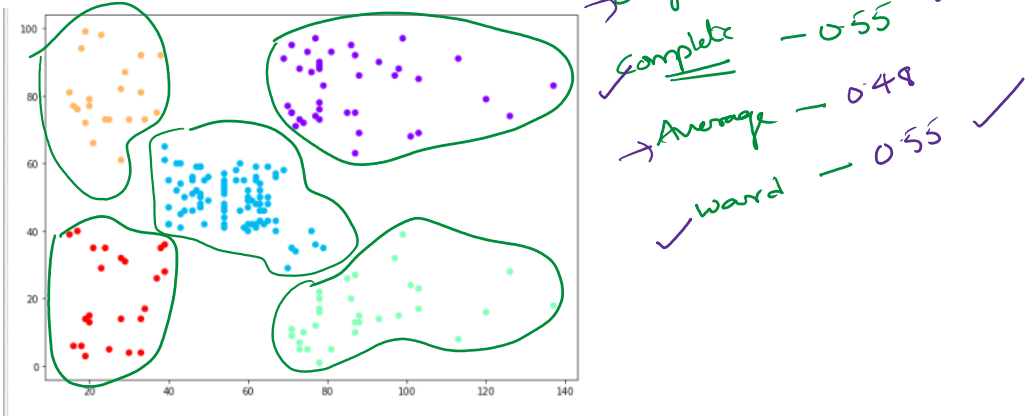


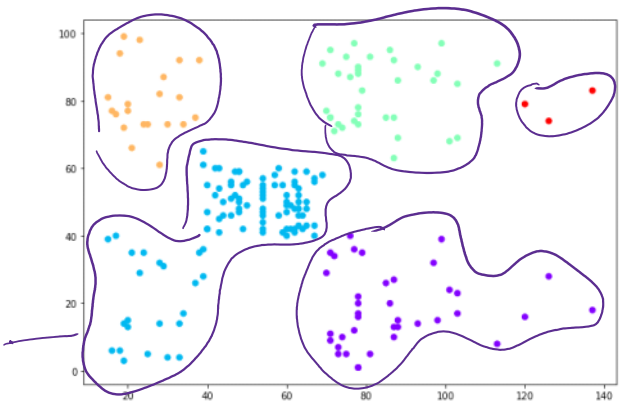
Our silhouette score measures the above two steps process for all the linkage methods above and gives us a value in between 0 to 1.

Which linkage method scores the highest value , those linkage method is making best cluster formation for our data set.

How to decide how many cluster are can be fixed.
Based on the domain knowledge and how points are plotted, we can decide the number of clusters from the given data.



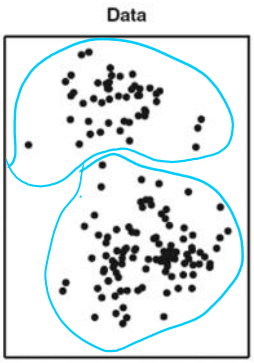


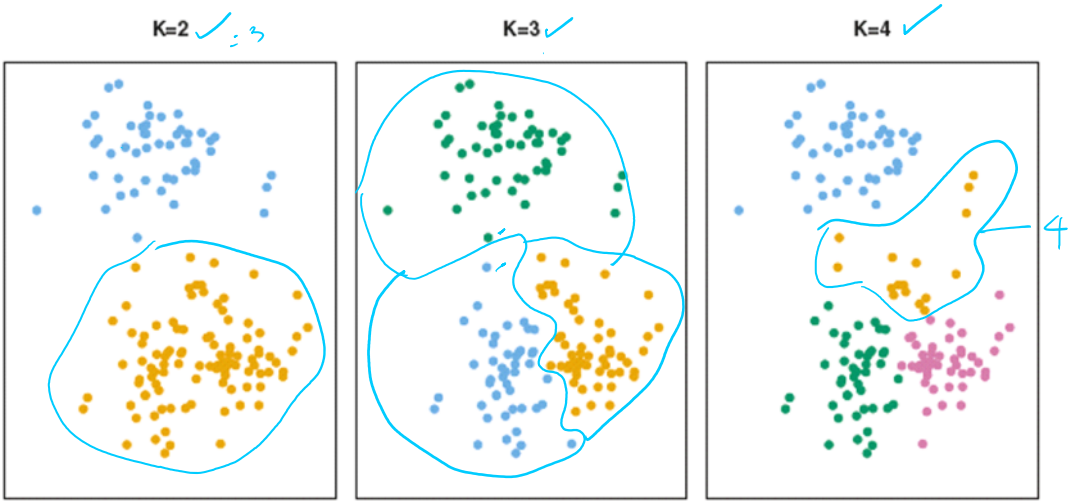


 # K - Means clustering:

Mean ---> central position

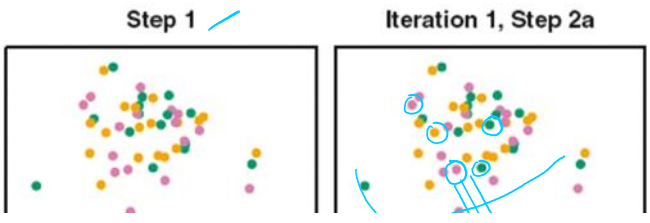
K --> number of cluster

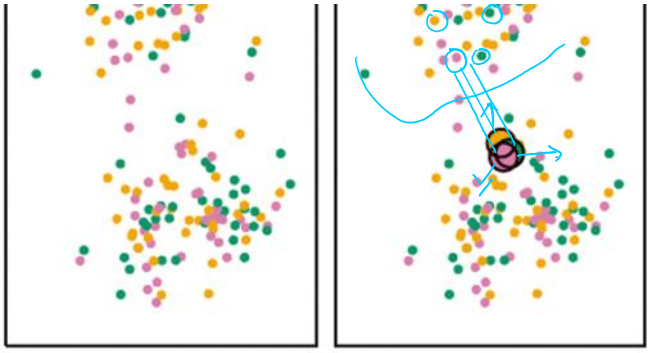




K =3 ---> 1 (yellow), 2(pink), 3 (green)

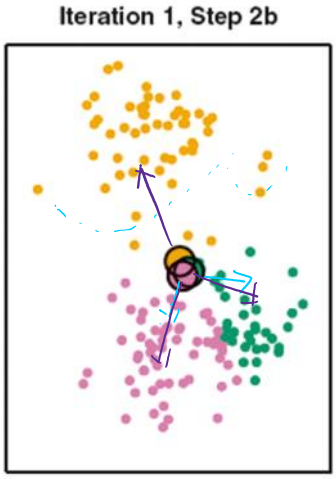
Step1: Initially, every data point will assigns a cluster number randomly.





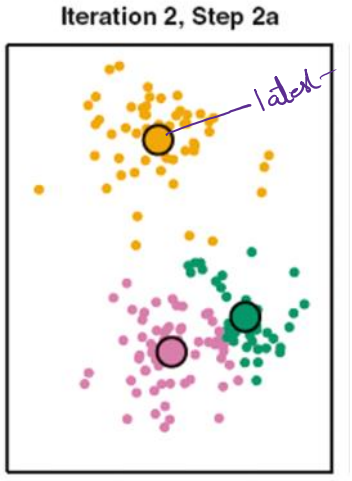
For each color/cluster , k means will calculates the centroid location as above

Step2: every data point will calculates the distance with every centroid location and recognizes the which one is nearest centroid location.

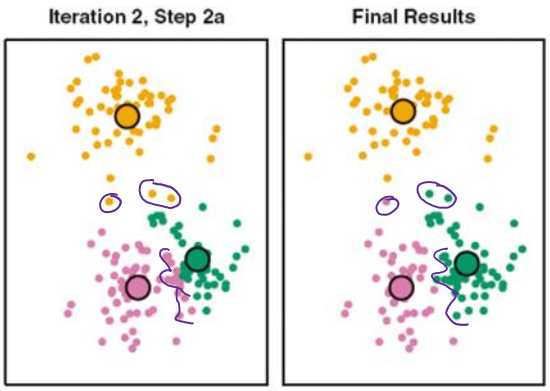


Iteration 2

1. Calculates the latest centroid location from the latest cluster assignments.



Step2: every data point will calculates the distance with every centroid location once again and recognizes the which one is nearest centroid location.



Previous centroid location = latest centroid location ---> assigning the cluster number will become stabled

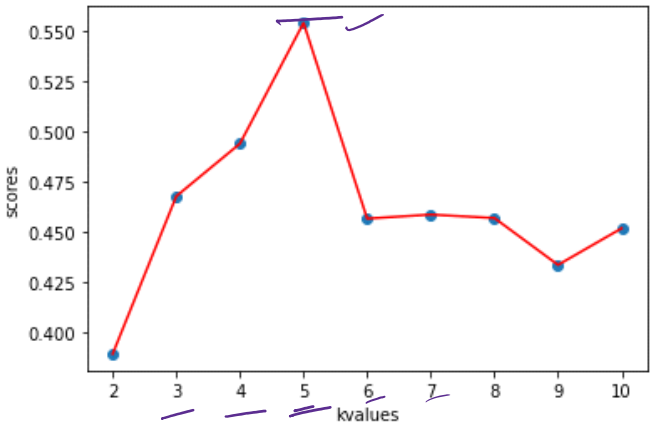
Sk-learn --> min = 20

How do we know which K value is to used?

Min k = 2, 3,4,5…….

We can verify that for each k value what is its silhouette score, where do we get highest score those k value is final.

K means (k =5) --> 0.55 score



① Ag ⇒ 5

② Kmeans → 5

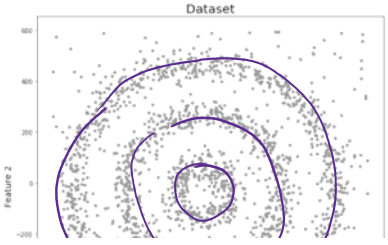
==========

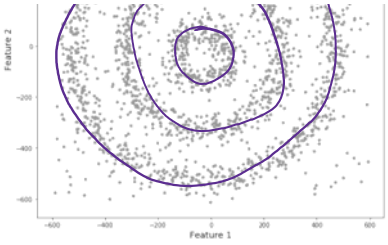
DBSCAN:

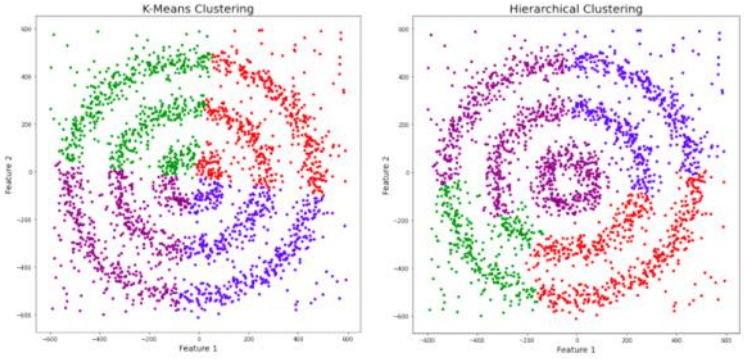
Density based spatial clustering applications with Noise:

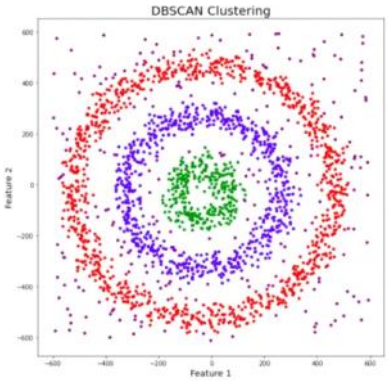
Noise: outliers

It will forms the cluster as well and at the same time it can identifies outliers as well.







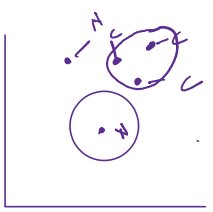


① Min_points = 3

② epsilon [radius] = 0.1, 0.5, 1, 1.5, 2 ...

Based on the above parameters, every data point will be identified as three categories

1. Core point
2. Border point
3. Noise point



Min = 3
ep: 0.5

If the centralized point along with another 2 points are there with in given radios, then centralized point will called as "**core point**"

If the centralized point along less than 2 points are there with in given radios, then centralized point will called as "**Border point**"

If the centralized point alone with no other points within given radios, then centralized point will called as "**Noise point**"

To form as one cluster, every data point should fall either it as core point or border point

Ex: increased the min points

Core point ---> border point

Border points increases

Border points remains same

Epsilon --> increase
Core point increases
Border points --> decreases
Noise points --> border points

Noise points will be predicted as -1
First cluster --> 0
second cluster --> 1

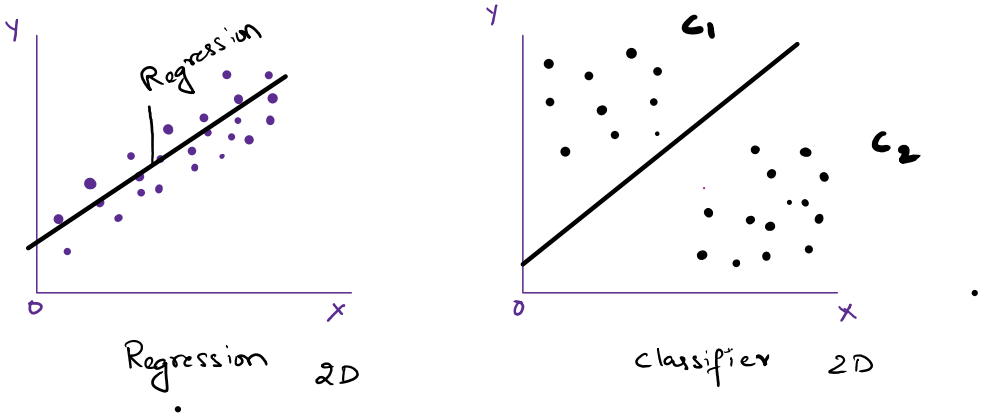
==========

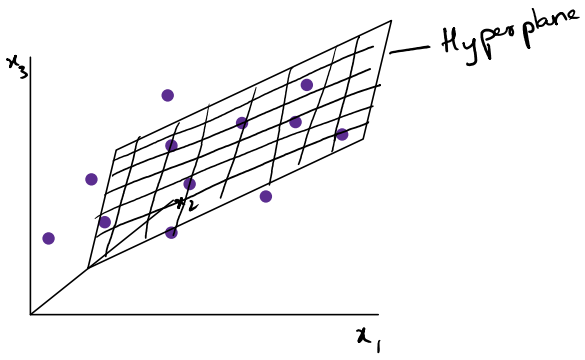
Support vector machine:

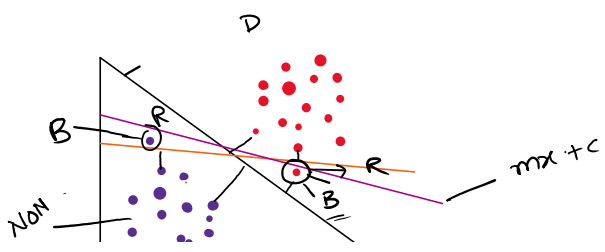
1. It works for both classification and regression models but highest usage is for classification only.
2. It also works for both linear and non-linear data sets.

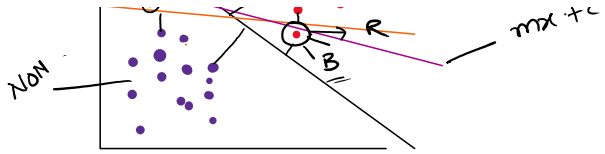
Suggested to choose when X variables continuous and that too non-linear ship existed in between X and Y variables.

1. Hyper plane
2. What is maximal margin plane
3. What are support vectors
4. Support vector classifier
5. Kernel functions.







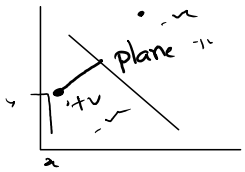


The line which is maintaining the maximum distance from both the groups is required to be choose by us

How the machine will recognize the best linear classifier?

It will calculates the distance between every data point to every possible equation.

Perpendicular distance



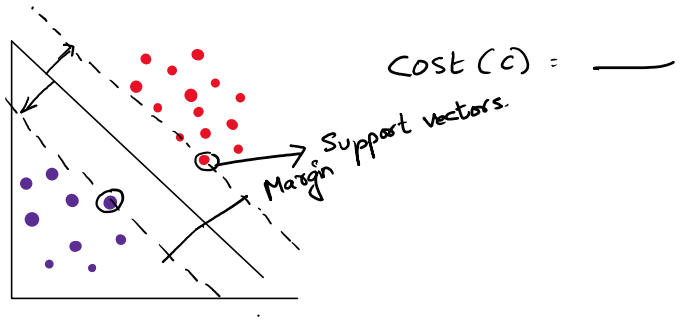
$$\pm d = \frac{ax_1 + by_1 + c}{\sqrt{a^2 + b^2}}$$

$$Blue = [d_1 + d_2 + d_3 + \dots d_k] = w_1$$

$$Red = [d_1 + d_2 + d_3 \dots d_k] = w_2$$

$$Bl = [d_1 + d_2 + d_3 \dots d_{16}] = w_3$$

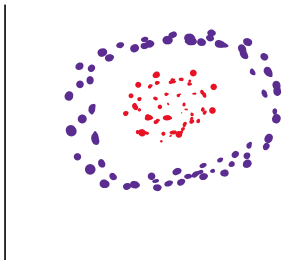
$$Max [w_1, w_2, w_3 \dots w_k]$$

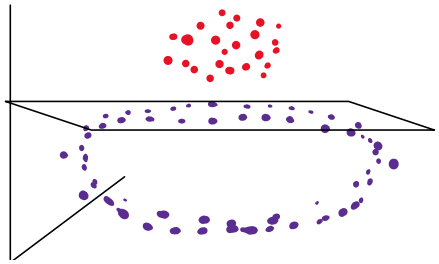


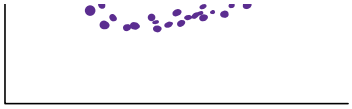
With the help of support vectors we are taking best linear classifier which has maximum margin. So, this maximum margin classifier is been taken support of those vectors, Hence we will called it as "**Support vector classifier**"

So, what parameter of value C If I passed, we are getting highest test accuracy. Those model is called best linear classifier.

Note: the above example is for when the data points are linearly separable







2D

3D

by applying some transformation technique
if we are able to convert them in to higher dimension then there is a possibility

Kernl functions:

1. Polynomial
2. Radial Basis function (RBF)

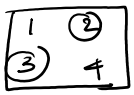
The above two formulae will take advantage of mathematical techniques and applies a new dimension to the existing data

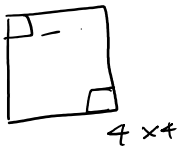
Polynomial

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

x_{ij} ⇒ i^{th} row, j^{th} column

$x_{1,j}$ →





d → degree d = 2, 3, 4, ...

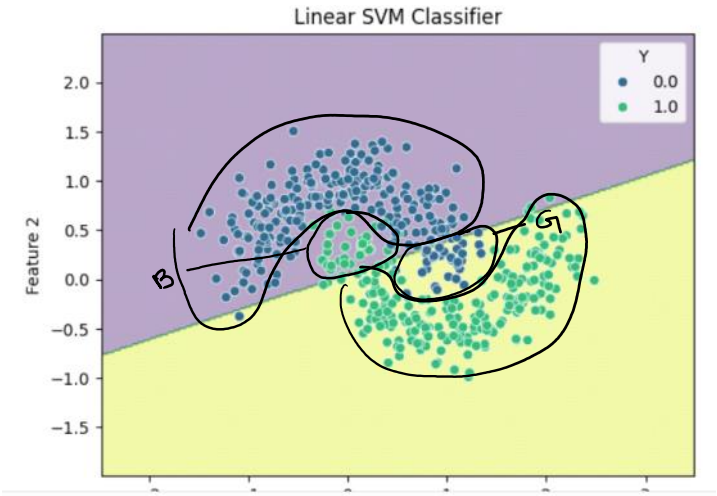
for what degree and cost parameter
we are getting highest accuracy

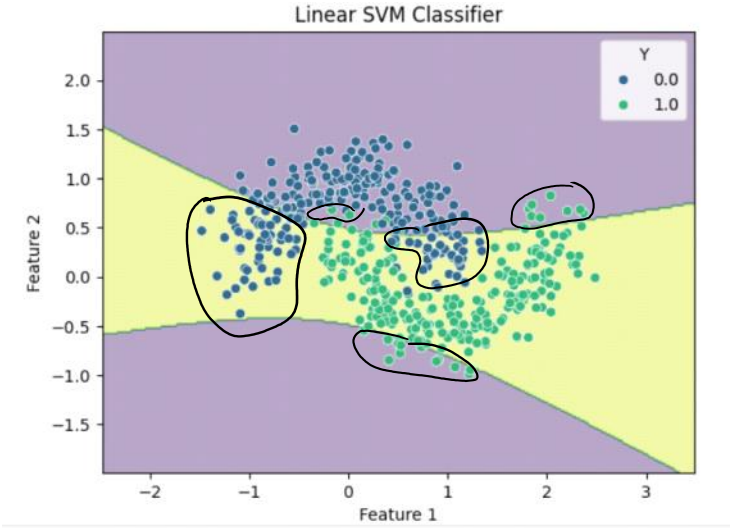
RBF:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

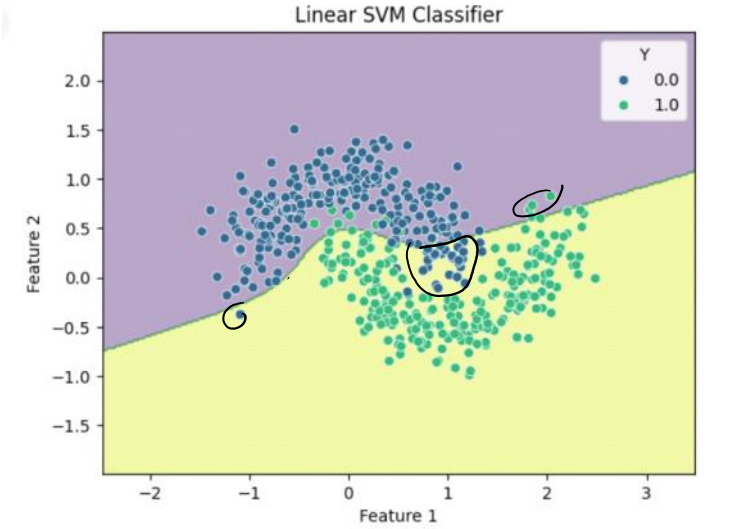
Gamma is the parameter

Linear classifier:
Training score: 0.85 Test score: 0.87

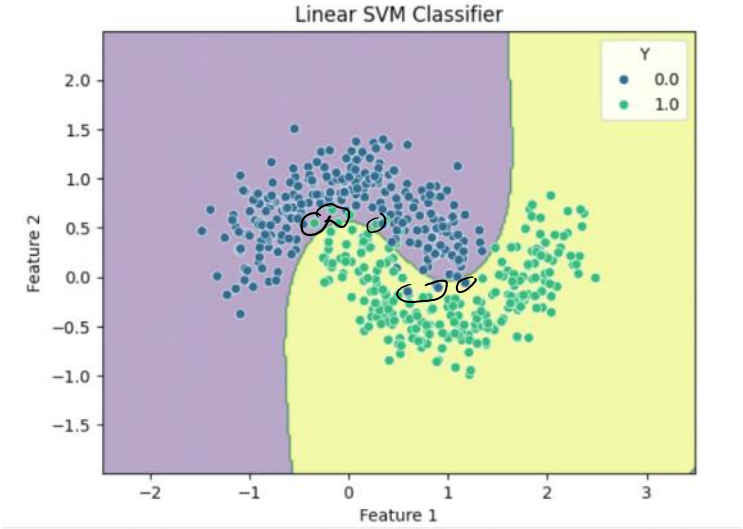


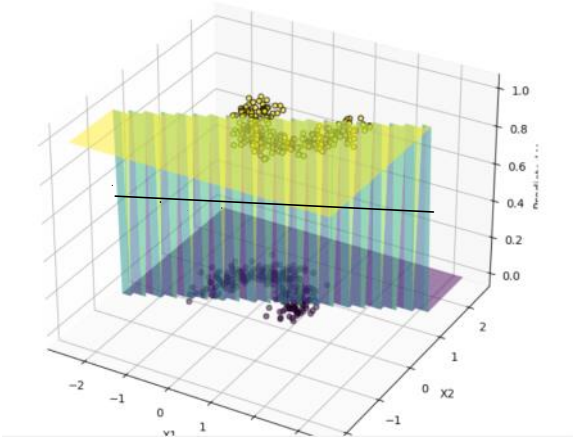


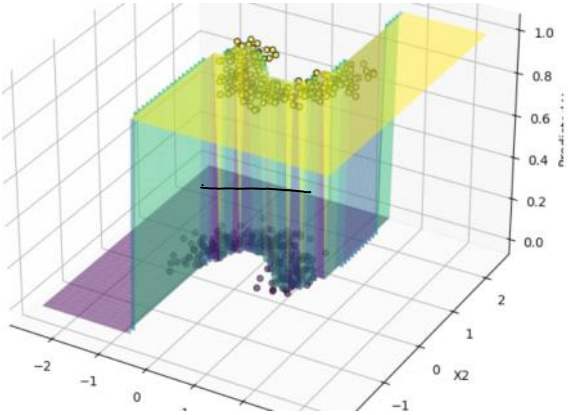
raining score: 0.91 Test score: 0.91



Training score: 0.98 Test score: 0.97







Association Rules:

Identifying the patterns , when the data is categorical
This analysis is specifically developed on the transactions of the customers

Example:

Market: merchants, flipcart, amazon, book my show, ticket, zomato, parkings, shopping in malls, departments, pharma, restaurant, supermarkets ….

Based on your transactions: what kind of person (hobbies, interests, activities, …)

Combination of items that you purchased in the past on those basis, we are trying to recommend new item/product to you.
This analysis we are doing on the market data , so it is also called '**Market basket analysis**'

item A	ItemB	ItemC	ItemD	item E
Bread	Milk	Jam	Soap	shampoo

	Bread	Milk	Jam	Soap	shampoo
A	Yes	Yes	Yes		
B	Yes	Rec	Rec		
C				yes	shampoo
D				Yes	Rec

Bread, Milk

Bread, Jam

When we are making an association based on the patterns and calculating some metrics is called "Rules", Association rules

We have three measuring rules:

1. Support
2. Confidence
3. Lift

TRANSACTION ID	ITEMS PRUCHASED
1	{Flowers, Greeting card, Glucose}
2	{Toys, Flowers, Balloons, Candy bar}
3	{Greeting card, Candy bar, Flowers}
4	{Toys, Balloons , Glucose}
5	{Flowers, Greeting cards, Glucose}

$6C_2 = \frac{6!}{4!\,2!}$

$6C_3 = \frac{3\,6\,5\,\cancel{4\,3\,2\,1}}{\cancel{4\,3\,2\,1}\;2\times} = 15$

Support:
How much of support is giving by each item?

$P[G] = \frac{3}{5} \Rightarrow 60\% \longrightarrow$ Greeting Card

$P[F] = 4/5 \Rightarrow 80\% \longrightarrow$ Flowers

How much of support is giving by together of two items?

$\rightarrow P[G \cap F] = \frac{3}{5} \Rightarrow 60\% \longrightarrow G \cap F$

$\rightarrow P[G \cap F \cap G] = \frac{2}{5} \Rightarrow 0$ to 1

Confidence:
How much of confidence is existing in between the two items:

Conditional probability

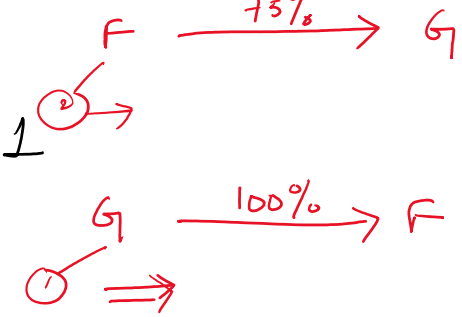
$P\left[\frac{A}{B}\right] = \frac{P[A \cap B]}{\quad}$

Conditional probability

$$P\left[\frac{A}{B}\right] = \frac{P[A \cap B]}{P[B]}$$

$$P\left[\frac{G}{F}\right] = \frac{P[G \cap F]}{P[F]} = \frac{3/5}{4/5} = 3/4 = 75\%$$

$$P\left[\frac{F}{G}\right] = \frac{P[G \cap F]}{P[G]} = \frac{3/5}{3/5} = 1$$



Lift:

$$\frac{\text{Confidence } [x \rightarrow y]}{\text{Support } (y)}$$

How many times product X is lifting sales of product Y

$y = 2x \Rightarrow 2y = 4x$ ↑

$y = 3x$

Apriori:

Min,max

```
In [16]: reports[0]
Out[16]: RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}),
confidence=0.29059829059829057, lift=4.84395061728395)])
```

$P\left[\frac{C}{Lig}\right] = 0.29$ high → C = 48#

	B	A	S	C	L
1					
2					
3					
4					
5					

```
In [30]: df_new.columns = ['Base','add','support', 'conf','lift']
```

```
In [30]: df_new.columns = ['Base','add','support', 'conf','lift']
    ...: df_new
Out[30]:
                  Base          add   support      conf      lift
0          light cream      chicken  0.004533  0.290598  4.843951
1  mushroom cream sauce     escalope  0.005733  0.300699  3.790833
2                pasta      escalope  0.005866  0.372881  4.700812
3        fromage blanc         honey  0.003333  0.245098  5.164271
4        herb & pepper  ground beef  0.015998  0.323450  3.291994
5       tomato sauce    ground beef  0.005333  0.377358  3.840659
6          light cream    olive oil  0.003200  0.205128  3.114710
7    whole wheat pasta    olive oil  0.007999  0.271493  4.122410
8                pasta       shrimp  0.005066  0.322034  4.506672
```

Recommendation system:

Data may be receives not an organized way,

But we have to prepare it in as per our convenience,

Username vs item names

Cosine-based similarity

$Cos(\mathbf{A},\mathbf{B}) = \mathbf{A}.\mathbf{B} / |\mathbf{A}| * |\mathbf{B}|$ $\quad sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$

- **A**: $(a_1, a_2, ..., a_N)$
- **B**: $(b_1, b_2, ..., b_N)$
- **A.B**: $a_1.b_1+a_2.b_2+...+a_N.b_N$
- $|\mathbf{A}|$: $(a_1^2+a_2^2+...+a_N^2)^{1/2}$; $|\mathbf{B}|$: $(b_1^2+b_2^2+...+b_N^2)^{1/2}$

Example:

	P1	P2	P3	P4
A	3	5	0	1
B	1	4	0	0

Cos(**A**,**B**)=
(3*1+5*4+0*0+1*0)/((3²+5²+0²+1²)^1/2*(1²+4²+0²+0²)^1/2)=0.94

	A	B	C	D
A	1	0.94	0.7	0.8
B	0.94	1	0.9	0.5
C	0.7	0.9	1	0.6
D	0.8	0.5	0.6	1

A ⟹ B C ⟹ B

B ⟹ A B ⟹ C

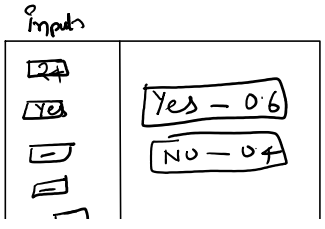
D ⟹ A

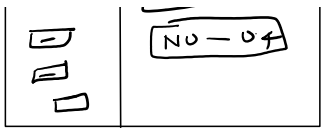
A ⟹ D

==========

Deployment:

FSD





1. Streamlit ✓
2. Flask
3. Django