⇒ Machine learning Introduction

Part 01

Que :- What is Machine Learning ?

Ans :- Machine learning is The SubField of computer Science that gives " Computer Ability to learn without being Explitcitly Programmed "

Example :-

(developer) Traditional Programming	Machine learning
a = 3 b = 5 → input c = a+b → code ↓ c = 8 → output input → code → output user → computer	a = 3 b = 5 c = a+b input (user) c = 8 Machine gives relation between a, b and "c"

input	output
X	Y
1	10
2	20
3	30
4	40
5	50
6	?

→ 6 (Prediction)
"$y = 10x$"

input Have more columns

x_1	x_2	y

→ $y = f[x_1, x_2]$
$ax_1 + bx_2 = y$

MODELLING

Q. How I can say it is 6 ?

Ans :- * Identifying The between "x" and "y"

$\therefore y = 10x \Rightarrow$ MODEL

* Substitute "6" in "x" value

$y = 10[6] = 60$

$\therefore y = 60.$

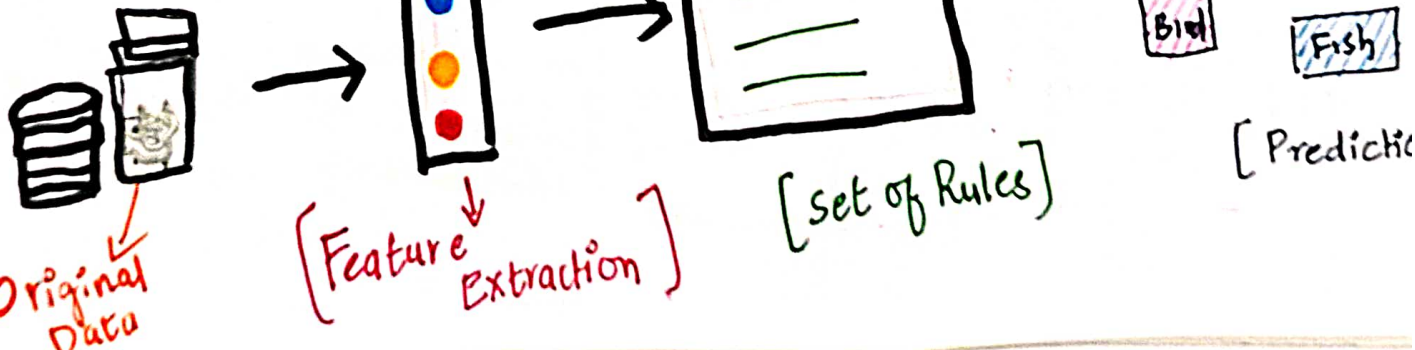
PREDICTIO

Machine learning

* Machine learn The given Data.
* Identifying The relation between Input and Output is called Machine Learning.

* Prediction

*ML Works :-



⇒ Major Machine Learning Techniques

* **Regression / Estimation**
  - Predicting **Continous Values**

* **Classification**
  - Predicting **Discrete Values** / **Item class** / **category of a case**

* **clustering**
  (No prediction)
  - **Finding** the **structure of data**; **Summarization**

* **Associations**
  - Associating Frequent **Co-occurring items / Events**
    Ex: Buying Brush/paste.

* **Anomaly detection**
  - Discovering **abnormal** & **Unsual Cases**

* **Sequence Minning**
  - **Predicting next Events**; **Click stream** (Markov Model, HMM)

* Dimensional Reduction
  - Reducing The size of data [PCA]
* Recommendation systems
  - Recommending Items.

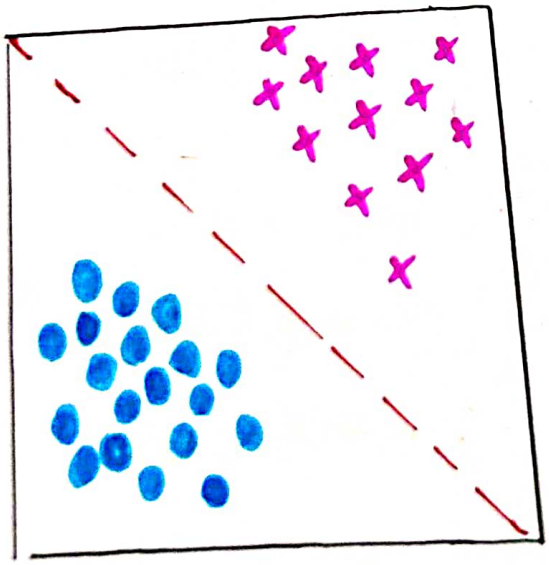
Que What is Supervised Learning ?

Ans:- We "Teach The model" then with that knowledge, it can predict Unknown (or) Furture instances.

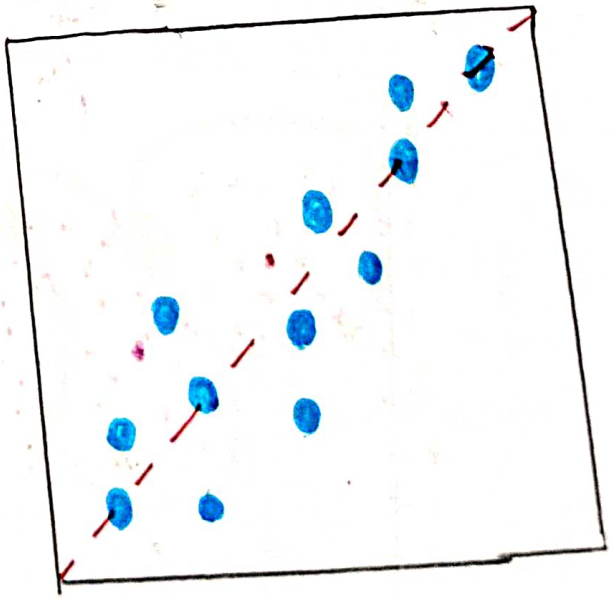
* Where Ever The Output Variable is Available is called as "Labeled data"

"In Supervised Learning Model", "The Algorithm Learns on a Label dataset, To generate Reasonable predictions For The response to new data.

1. Regression [Continuous Data]
2. Classification [Discrete Data]



* classification



* Regression

Que :- What is UnSupervised Learning ?

Ans :- UnSupervised Learning, We have **Unlabed Data**

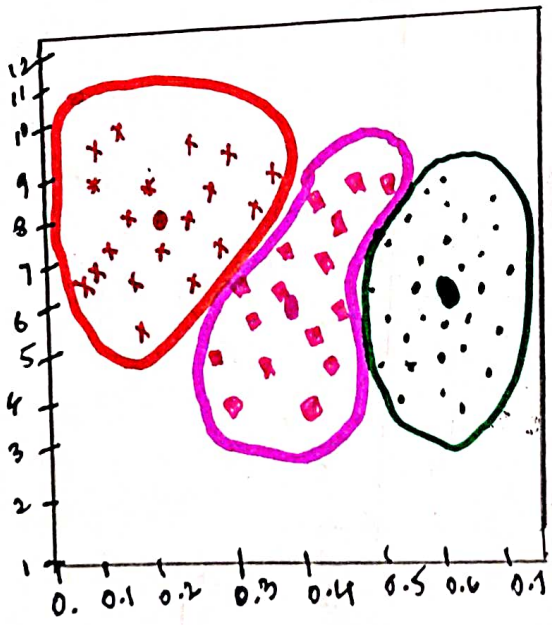
ie. **No Output Feature Available.**

"UnLabed Data" that the algorithm tries to make Sense of by "Extracting Features, Co-occurence and **Underlying patterns on its Own.**

1. Clustering
2. Anomaly Detection
3. Association
4. Auto Encoders

Que : What is clustering ?

Ans: "clustering" is **grouping of data points** (or) **Objects that are Somehow Similar by**



* clustering

* Supervised Learning	* Unsupervised Learning
* Regression - Classifies labeled data * classification - Predicts trends using previous labeled data * Has more Evaluation methods than The Unsupervised Learning. * Controlled Environment	* clustering - finds pattern and grouping From Unlabeled data. * has Fewer Evaluation Methods Than Supervised Learning * Less Controlled Environment

Dt:- 6/4/22
10:30pm

6/4/22
3:30 pm

Sklearn
↓
Scikit Learn

Feature Scaling

From Sklearn.preprocessing import StandardScaler

df['Age_sc'] = sc.fit_transform(df[['Age']])

Using Pandas

input, we giving as Data Frame

Out :- We get DataFrame.

missing Values

From sklearn.impute import SimpleImputer

df["CLMAGE"] = pd.DataFrame(mean_imputer.fit_transform(df[["CLMAGE"]]))

Using Simple Imputer Pandas.

Data Frame

input, we giving Same

Out :- We get array.

Example :- User Defined Functions

```
def add(a,b)
    c = a+b
    return float(c)
```

→ dd (3,4)

Out : 7.0 → Float value

```
def add (a,b)
    c = a+b
    return c
```

→ add (3,4)

Out : 7

```
def maths (a,b)
    c = a+b
    d = a*b
    e = a%b
    f = a-b
Print ("sum":, c)
Print ("mult":, d)
Print ("div":, e)
Print ("sub": f)
```

→ Saving in "Jack.py"

```
From Jack import maths
```

Jack → module; maths → Function

```
maths(3,4)
Out:
Sum : 7
Sub = -1
Mul = 12
div = 0.75
```

* Because, Function is Created in a such a way That answer should be Float. / But Both The Functions are same Input

Only output is different.

* Same, Simple Imputer, Standard Scaler and More Functions From sk.learn are InBuilt function.

* We Can't Remember Every Function. So, We Take The Help function and see The Output Variable Type.

* If Out is Array. We change (or) Convert into Data Frame, Just writting [Pd.DataFrame] before The Function.

⇒ Estimator API

Modelling

import 'ML' Algorithm

model = MLAlgo ( )

model.fit [x_train, y_train)

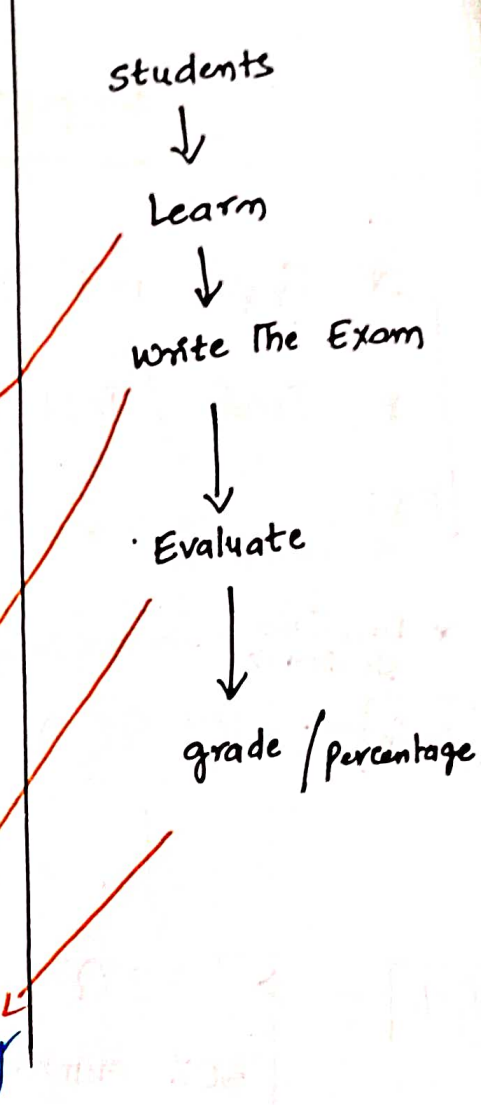
Prediction

ypred = model.predict (x_test)

Evaluation

Comparsion (y pred, y test)

Accuracy



Ex:-

X	Y
1	10
2	20
3	30
4	40
5	50
6	?

Relation : y = 10(x)

ML. Algorithm [model]

model = Algorithm [y = 10x]

ypred # model. predict [x_test]

6

y = 10x [x = 6]

y = 10 x 6

Output = 60

* MODEL Evaluation * → Train_test_Split

* Train and Test on Same Dataset
* Train / Test split.

Ex :-

Case 1

* In a class sir given → * Interview Question

Train → 100 Q & A

↓

Test → 30 Q

80% marks (%)

Your a good student, But Only for this class Only.

* If what ever The Data I have given [100 Q/A]. if we ask 30 Q/A From The same dataset. it gives better performance, But. Only For that dataset it works, not for any new dataset (Q/A)

Case 2

100 Q/A

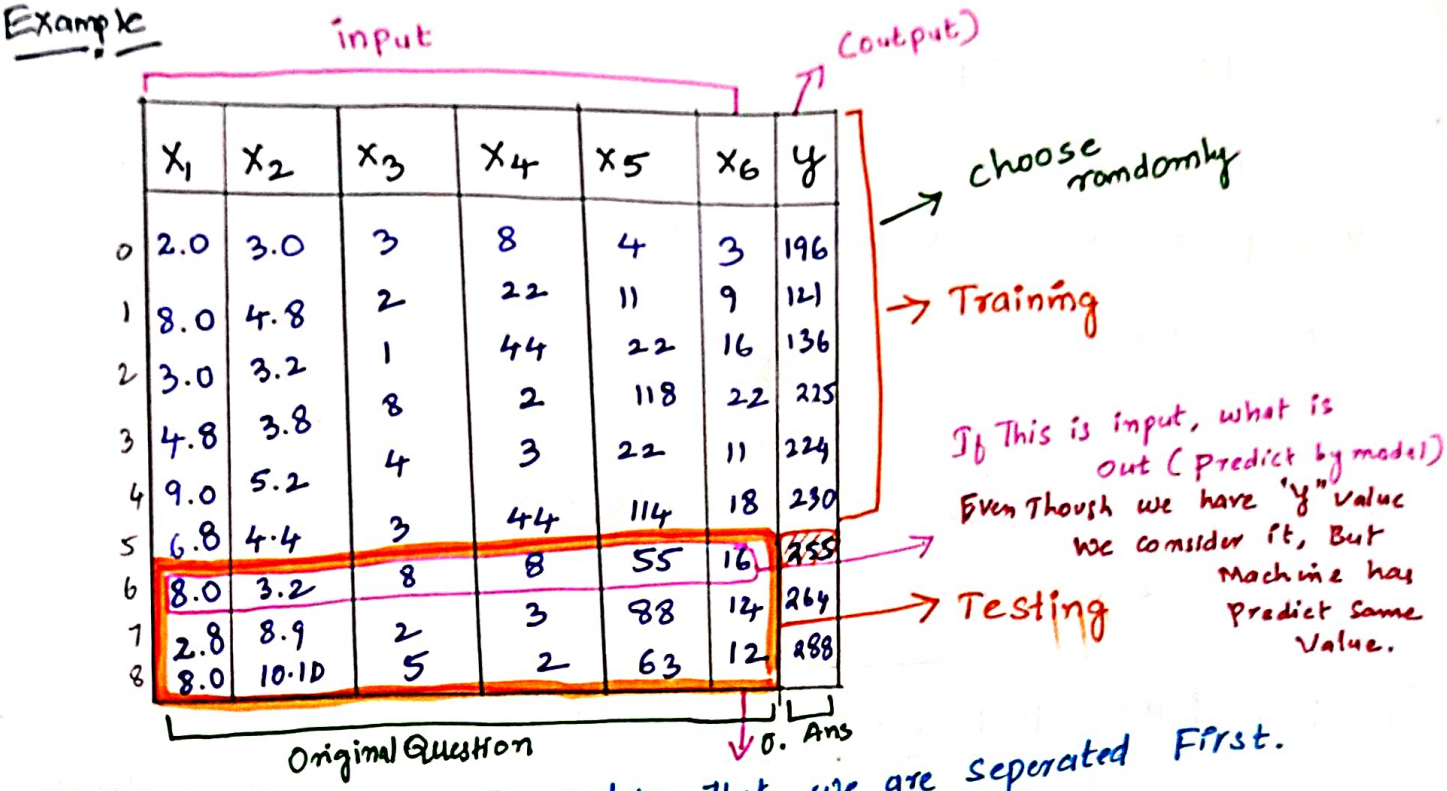
70 [Traning] 30 [Testing]

↓

For These 70 (Q/A) Given To Machine. not to understand, But given Only For Understand Concepts Only. not For reading data

Now, we ask new 30 (Q/A), not From 70 (Q/A) that we have given. and Evaluate Accuracy For Traning and For Testing, Seperately.

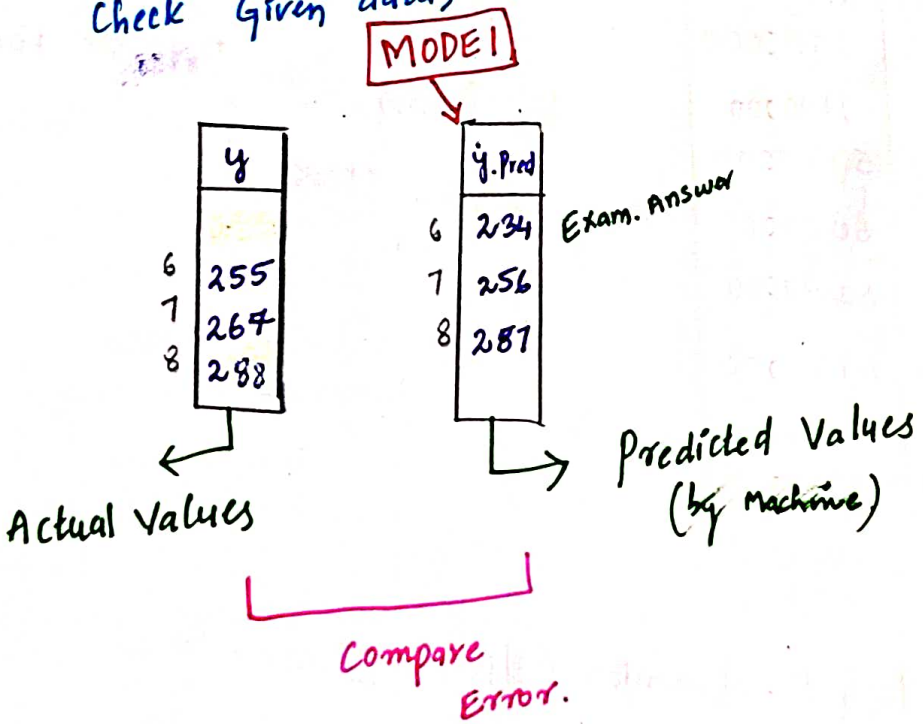
Example



	X_1	X_2	X_3	X_4	X_5	X_6	y
0	2.0	3.0	3	8	4	3	196
1	8.0	4.8	2	22	11	9	121
2	3.0	3.2	1	44	22	16	136
3	4.8	3.8	8	2	118	22	225
4	9.0	5.2	4	3	22	11	224
5	6.8	4.4	3	44	114	18	230
6	8.0	3.2	8	8	55	16	255
7	2.8	8.9	2	3	88	14	264
8	8.0	10.10	5	2	63	12	288

* In Testing, we give input data That we are seperated First.
and check with Accuracy of "y" value (output)

* In Testing, we don't take "y" Variable (output) because To Check Given data, Model is Working (or) not.



Code :-

```
# Import numpy as np
# Import pandas as pd
# Import matplotlib.pyplot as plt
%. matplotlib inline

# df = pd.read_csv ("homeprices.csv")
df
```

Out :-

	town	area	Price
0	Chennai	2600	5500000
1	Chennai	3000	5650000
2	Chennai	3200	6100000
3	Chennai	3600	6800000
4	Bangalore	2800	5850000
5	Bangalore	3300	6150000
6	Bangalore	2600	6500000
7	Bangalore	3600	7100000
8	Hyderabad	2600	5750000
9	Hyderabad	2900	6000000
10	Hyderabad	3100	6200000
11	Hyderabad	3600	6950000

Here we use Same dataset of Encoding. and we divided Them into train / test. it will Easy to understand by Taking Same Dataset For Further steps.

creating dummies

```
#   df_dum = pd.get_dummies (df, dropfirst = True)
#   df_dum
```

Out

Area	Price	town_Chennai	town_Hydrabad
2600	5500000	1	0
3000	5650000	1	0
3200	6100000	1	0
3600	6900000	1	0
2600	5850000	0	0
2900	6150000	0	0
3300	7100000	0	0
3600	5750000	0	0
2600	6000000	0	1
2900	6500000	0	1
3100	6200000	0	1
3600	6950000	0	1

Now, this Changed data set, is we Consider Further to Train/test

square Foot

Here Based on Area (sft) and town, we want predict The price

Independent Variables

So..., area, town-chennai, town Hyderabad are → Input variable

price is Output Variable (y)

Dependent Variable

Because, it is Depending on "x" values To give Output.

option To Consider x & y Variable

```
x = df_dum.drop("price", axis = "columns")
y = df_dum.price
```

#Here we split The data into Train/test.

stores in "x" and "y".

check "x" → Input Variables (or) Independent Variables

```
x
```

Out:

	area	town_chennai	town_Hydrabad
0	2600	1	0
1	3000	1	0
2	3200	1	0
3	2600	1	0
4	2800	0	0
5	3300	0	0
6	3600	0	0
7	2800	0	0
8	2600	0	1
9	2900	0	1
10	3100	0	1
11	3600	0	1

check "y" → Output Variables (or) Dependent Variables

```
y
```

Out:

	price
0	5500000
1	5650000
2	6100000
3	6800000
4	5850000
5	6150000
6	7100000
7	5750000
8	6000000
9	6500000
10	6200000
11	6950000

from Sklearn . model_Selection import train_test_split

Liberary — Module — (Function)

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size = 0.3, Random_state = 29)

y → train y_train, Test y_test

Here x, y → Divided into two parts → are stored in x_train, x_test, y_train, y_test.

We Test on this → x

Spliting into two parts

Fixing The Same data point state. (To make data Constant)

Train 70% → x_train → 12 X 0.7 = 8.4 records (8 records)

Test 30% → x_test → 12 X 0.3 = 3.6 records (4 records)

Here 0.3 indicates 30%, we are giving To Test The data

"ML" Algorithm 1	"ML" Algorithm 2
* Same Data	* Same Data

In order To Compare which algorithm is working well and giving high Accuracy.

We Have To Same data For Every algorithm. and Compare with all algorithm

code :

```
# X_train, Xtest, y_train, ytest = train_test_split = (x, y, test_size=0.3,
                                                       Random_state = 29)
```

X_train

Out :-

	area	town_chennai	town_hyderabad
7	3600	0	0
6	3300	0	0
1	3000	1	0
0	2600	1	0
8	2800	0	1
2	3200	1	0
3	3600	1	0
5	2800	0	0

X_test

Out :-

	area	town_chennai	town_hyderabad
9	29	0	1
4	26	0	0
10	31	0	1
11	36	0	1

y_train

Out :-

	Price
7	7100000
6	6500000
1	5650000
0	5500000
8	5750000
2	6100000
3	6800000
5	6150000

→ This Heading is not mentioned in my pe

→ Only These Values.

y_test

Out :-

	Price
9	6000000
4	5850000
10	6200000
11	69500000

Dt:- 7/4/22
12:30pm

Example :- # past Data

Input ← X | Y → Output

	X	Y
0	1	10
1	5	15
2	9	36
3	14	45
4	12	12
5	4	88
6	78	56
7	2	44
8	14	68
9	36	52

Based on the data model is formed (Equ)

If we change some data (1,2 records) our model also changes.

Step1 :- Split The Data For Train/test

	X_train		y_train
0	1	10	
1	5	15	
2	9	36	
9	36	52	
4	12	12	
5	4	88	
8	14	68	

	X_test		y_test
7	2	44	
3	14	45	
6	78	56	

Train 70% — Selected Randomly

Test 30%

Step2 :- Modelling

MODEL = 10x + 5

Step3 :- Prediction.

Train

MODEL = 10x + 5 ← Substituting x values in

X	Y	y_Pred_train
1		15
5		55
9		95
36		365
12		125
4		45
14		145

⇒ 10(1) + 5 = 15
⇒ 10(5) + 5 = 55
⇒ 10(9) + 5 = 95
⇒ 10(36) + 5 = 365
⇒ 10(12) + 5 = 125
⇒ 10(4) + 5 = 45
⇒ 10(14) + 5 = 145

Test

X	Y	y_Pred_test
2		25
14		145
78		785

⇒ 10(2) + 5 = 25
⇒ 10(14) + 5 = 145
⇒ 10(78) + 5 = 785

Step4 : Evaluation

Train

Y	y_Pred_train
10	15
15	55
36	95
52	365
12	125
88	45
68	145

difference blw Actual "Y" - Predicted "Y" = Error (anything)

Ex:- 10 − 15 = −5
15 − 55 = 40
36 − 95 = 59
⋮

Train_Error

Test

Y	y_Pred_test
44	25
45	145
56	785

Test_Error

* Machine Learning Algo steps

⇒ MODELLING

1. Finding the **relation between X & Y** of given data

X_train , Y_train

#relation → "MODEL"

Ex: $y = f(x)$
$y = ax^2+bx+c$
$y = mx+c$
$y = 10x+5$ (Equation)

⇒ PREDICTION

Substuting the **Values of "x"** in the given Equation and MODEL

MODEL & Calculating **"y" Values.**

⇒ Evaluation

Calculating the **error between** Predicted Values and Actual Values.

Good Model

#Train_Error = 5 % Accuracy = 95 %

#Test_Error = 5 % Accuracy = 95 %

[±5] with difference

⇒ train accuracy ≅ Test accuracy # Good Model.

with deviation of ±5 %.

Overfitting

→ train Accuracy = 90 %.
→ Train Error = 10%.

→ test Accuracy = 60 %.
→ Test Error = 40%.

[Model Performs good in training But, Gives Bad, in testing results]

train Accuracy > test accuracy.

Underfitting

→ train accuracy = 60 %.
→ Train Error = 40%.

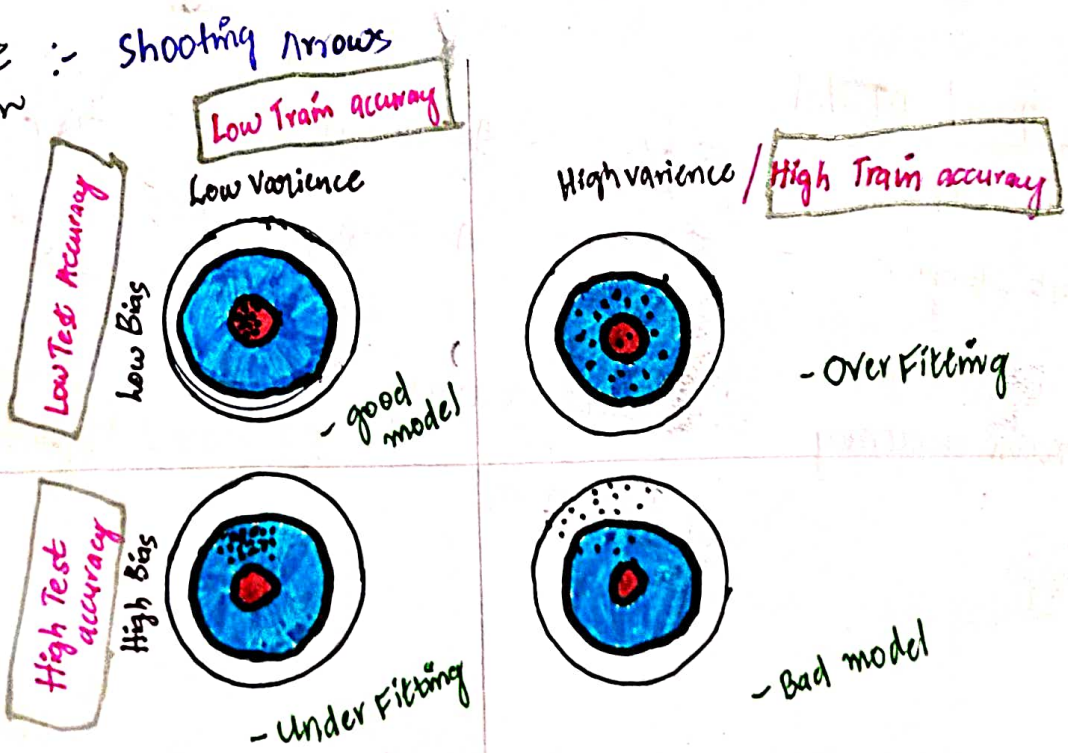
→ test accuracy = 90 %.
→ Test Error = 10 %.

[Model Performs bad in training Data. But Performs Good in Testing data]

train accuracy < test accuracy.

⇒ Bias / variance Trade-off → difference

Example :- Shooting Arrows



Variance = Distance b/w Data points / Train Error

Bias = close To Target / Test Error

***

Varience = train
Bias = test

Overfitting

→ High varience = High train Accuracy

→ Low bias = Low test Accuracy

Underfitting

→ High bias = High test accuracy

→ Low Variance = Low train accuracy

good model

→ Low Bias = Low Test accuracy

→ Low Variance = Low Train accuracy

1/4/22
5:00 pm

Q/A.

Over View o
Machine lea

1. Linear Regression

It is Used to Predict real Values, (or) Predicting Output Variable is "Continous Data" a relationship is established between independent and dependent Variable by fitting a best fit line. (or) Assuming relation between x & y is "Linear". The best fit Line is known as "regression line". represented as

$$y = mx + b$$

- m → slope
- b → intercept
- y → dependent variable
- x → independent Variable

2. Logistic Regression :

it is a classification algorithm used to estimate "Discrete Data Values". based on given set of independent Variables. its Output Value lies between [0 and 1] or [Yes / No]. Uses The sigmoid function. Probability

$$\frac{e^y}{1+e^y}$$

3. Decision tree

It is Supervised learning algorithm Used for classification problems. which Works For both Categorical and Continous dependent Variables.

In this algorithm "population is Split" into two (or) More homogenous sets. according to our Given Conditions.

it Uses different Techinques Like

* Gini impurity
* Information Gain
* Chi-Square Entropy.

4. Random Forest

A Collection of Decision trees is called as "random Forest." To clasify a new object based on its attributes, Each Tree is classified and Take majority of votes for That class. and classifies them. (Bagging Technique)

applicable for both Classification & Regression

5. K-Nearest Neighbors

K-Nearest Neighbors is a Simple algorithm that stores all available cases and classifies New Cases by a majority Vote of its "K-Neighbors".

it is Measured by distance Function

* Manhattan Distance $|x_2-x_1| + |y_2-y_1|$
* Euclidean Distance $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

6. Naive Bayes Theorm (algorithm)

Conditional Probability

it is Supervised learning algorithm. Which is based on "Bayes theorm". and used for Solving classification problems.

Bayes theorm is used to determine Probability of a hypothesis with Prior knowledge. and Likelihood.

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

* Naive :- it assumes that Each input variable is independent.

7. SVM (Support Vector Machine)

it is Supervised Machine learning algorithm Used for both classification & Regression. Main objective of SVM to find a hyplane in a "N-dimensional space that distinctly classifies data points

(or)

SVM to find best line in two dimensions (or) The best hyperplane in more than two dimension in order to help us seperate our space into classes.

* Linear Seperable
* Non-Linear Seperable (or) Kernal SVM

8. Boosting Techniques

Ensemble learning (or) boosting Techniques it is combination of "Weak Learners". that implies Combination of estimators (models) with an applied coefficient could act as effective Ensemble Estimator.

Boosting Formula :

$$F_T(x) = \sum_{t=1}^{T} f(x) = \alpha_t \, h(x)$$

Sumation of t=1 to "T"

Each model

α_t → learning rate

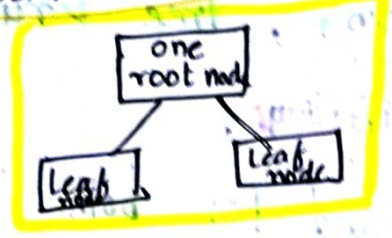
9. AdaBoost Algorithm

AdaBoost (adaptive Boosting) Ensemble method.

it assigns weights to observations which are incorrectly predicted. Next model works to predict these values correctly which are wrongly predicted before.

* Each "Weak Learner" is called as "stump"

- Depth = 1



consists of
- 1 root node
- 2 leaf nodes

10. Gradient Boost

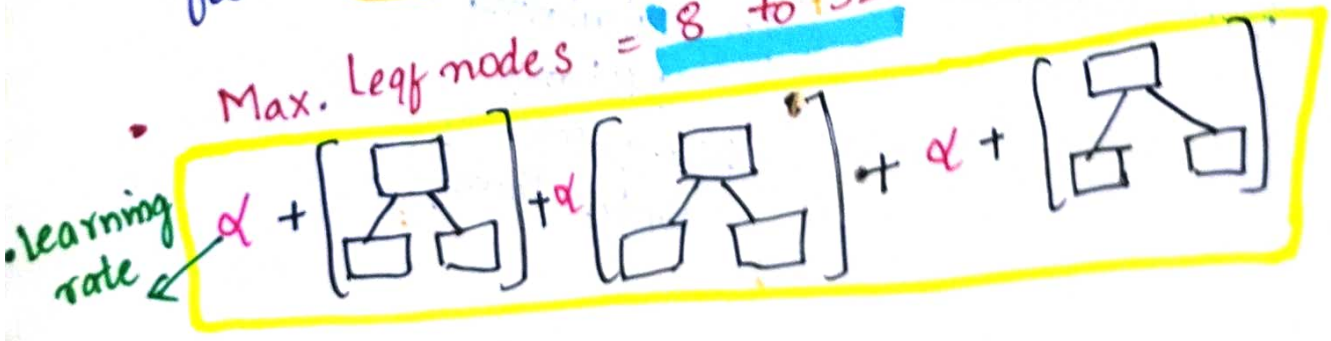
Gradient boost is used to predict both Continuous and categorical variables. it is same as boosting techinques. Here we calculate The Errors and minimize "loss function" (single training Error) in a sequential order.

↓

(Sum of error) min

and Trees are adjusted after multiple times.

- Max. Leaf nodes = 8 to 32



11. XGBoost

XGBoost also called as "Regularized Boosting" Like L_1 & L_2 regularization. it is "10" times faster Than Gradient boost. it implement "parallel processing" it reduces The overfitting, Handle Missing Data, Inbuilt "Cross validation", Tree pruning using depth approach; High Flexibility.

it Can Used For both "classification" and Regression problems".

12. K-Means algorithm

it is "Un Supervised" learing algorithm. (or) Unlabel data. K-means algorithm "identifies 'k' no. Of Centroids, and then allocates every data point to nearest Cluster, while keeping Centroids small as possible, it uses "Distance formula" same as K-nearest Neighbours.

Based on Assumption.

↓

For calculating Distance blw points & cluster them.

13. **Hierarchial clustering**

it combines Data points which are very close by distance. That Groups Similar objects into Groups called clusters. This clustering involves creating clusters that are predetermined Ordering from Top to bottom.

These are two of hierarchial clustering

* Agglomerative (From Top to bottom)
* Divisive. (From bottom to Top)

14. **DB-Scan clustering**

DB-scan (Density Based Spatial Clustering of application with noice")

DB-scan clustering Locates regions of "high densi and Seperate Outliers.

(or)

DB Scan is to find the neighborhoods of data Points Exceeds Certain Density threshold.

* two parameters.
* R (radius of neighborhood)
* m (min. no. of neighbours)

15. **Gradient descent algorithm**

Gradient Descent is an "optimization" algorithm for Finding a Local minima (minimum Point of a differentiable function.) (or) it Gradually reducing Weights and making Errors as Minimum Value.

(or) Ex:- By applying diffirent "x" values and identifying The perfect "x" value, till we get error as "0" Zero is called gradient "descent" minize Cost Functi

Update rule formula

$$y = m \times b$$

16. **Stochastic Gradient Descent**

it has both * "Local Minima" and "Global minima".

Entire data We divide into parts (or) batches Each part we are identifying The "local minima" and from that Over all identify The "Global minima" is called "stochastic Gradient Descent"

Ex:- State ranker (local Minima)
All india ranker (Global Minima)

7/4/22
9:30pm

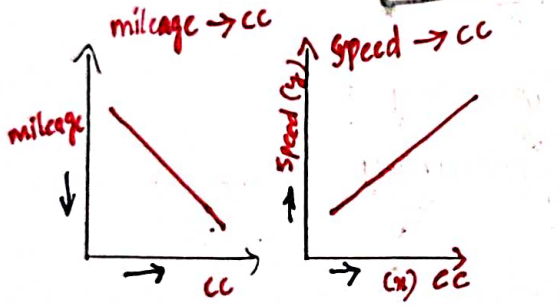
— Regression

Predicting Output which is Continous

- Regression
  - Linear Regression
  - Non-Linear Regression

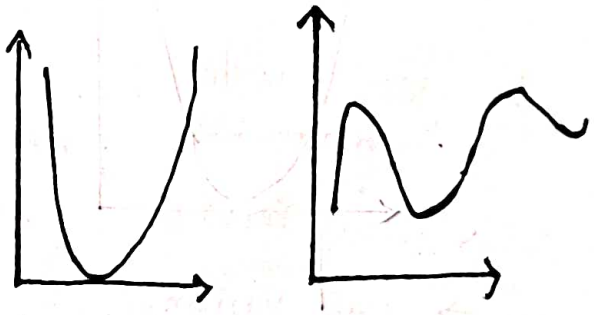
("Assuming the relation between x & y is Linear")

Ex:- In The Form of $(y = mx + c)$

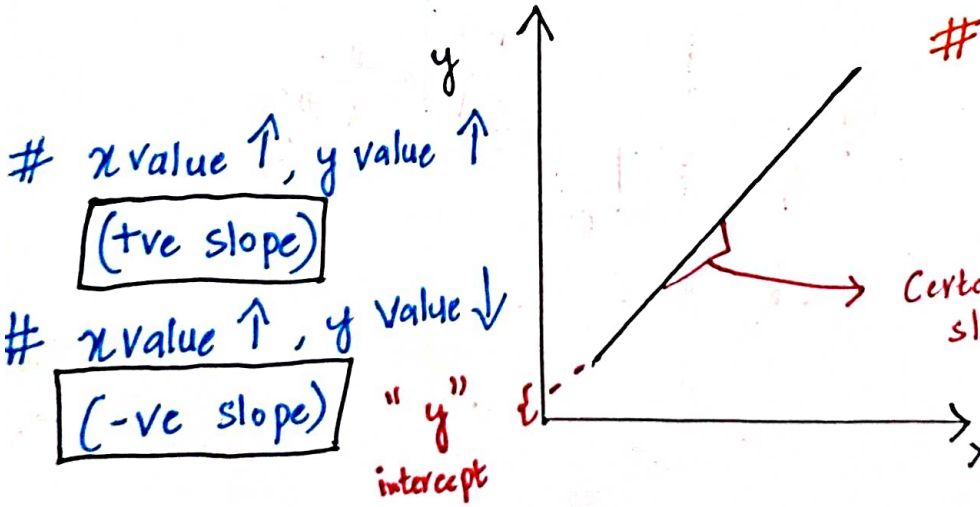


* mileage decreases ↓ cc increases ↑
* cc increases ↑ Speed increases ↑

Any line other than straight line is Non-Linear



Quadratic equation



x value ↑, y value ↑ (+ve slope)

x value ↑, y value ↓ (-ve slope)

Any relation in terms of Straight Line, it is called as "Linear Regression"

Ex:-

X	Y
1	15
2	25
3	35
4	45
5	55
6	? 35
7	35
8	35.

In Early Ages, They took "Avg" values.

Like

$$\frac{15+25+35+45+55}{5} = 35$$

→ They say, Avg Line is Best Fit Line.

* Best rants

tips (waiter)

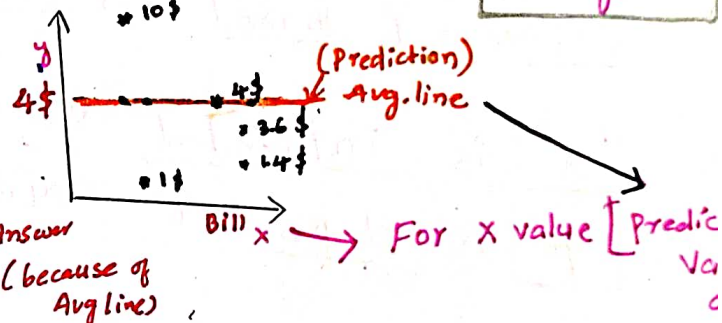
Bill amount

1. 10 $ – (not given)
2. 1 $ –
3. 4 $ –
4. 3.6 $ –
5. 1.4 $ –
6. → Avg 4$
7. → 4$ Every time Answer is 4$ (because of Avg line)

so, For only One Variable / column – (Avg)

For Single Variable

Best Fit line is Average line

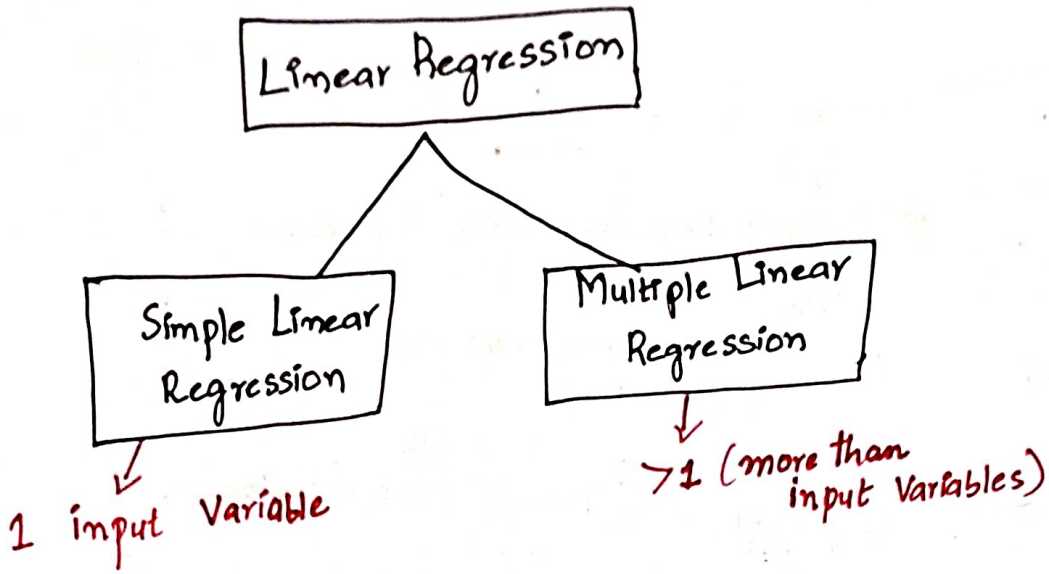


→ For X value [Predic Va

H₀ : Average line is best fit line

H₁ : Regression line > Avg. line.

Y	avg. line Y – Prediction	Error	error²
10	4	6	36
1	4	-3	9
4	4	0	0
3.6	4	-0.4	0.16
1.4	4	-2.6	6.76
		0	51.92



Simple Linear Regression :

→ In Simple Linear regression, We predict the value of One Variable "Y" based on other Variable "X"

→ "X" is independent Variable (or) Input variable (or) Exploratory variable

"y" is dependent Variable (or) output variable (or) response variable

Ex:-

Weight	CC	Speed	Mileage → O/P
indepen dent	in. var	in. var	(it depends on, weight, cc, speed)

Que :- Why it is Simple ?

Ans :- Because, it Examines Relation ship between two Variables Only.

X	Y

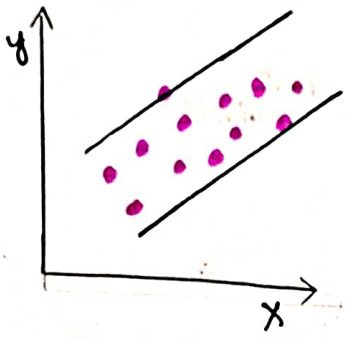
$y = x$

Que :- Why linear ?

Ans :- When the independent Variable increases (or Decreases) The dependent Variable increases (or Decreases) in a linear Fashion.

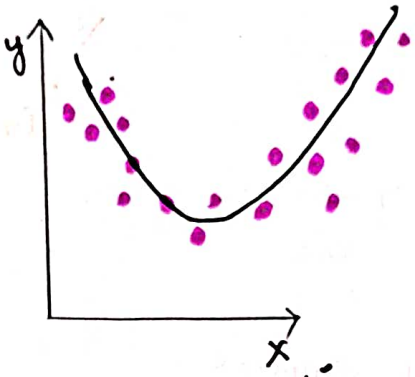
Que: How to check Linear Regression ?
(or) not

Ans:- By scatter plot ← Linearity, direction

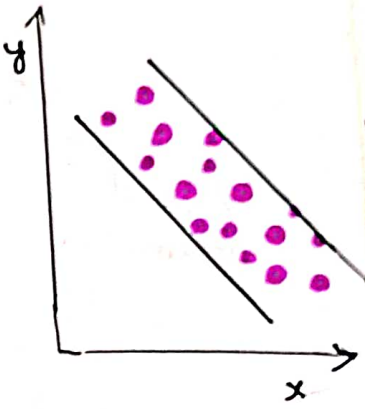


Linear

As "x" value increasing
"y" value also increasing
"Positive Direction"



Quadratic line

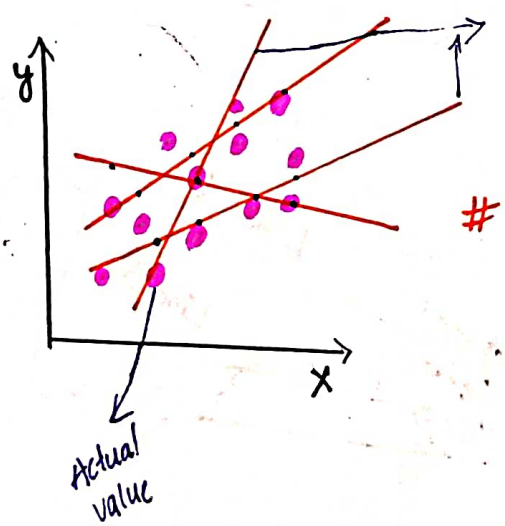


Linear But downward direction

As "x" value increasing
"y" value also decreases
"Negative direction"

Que:- Which Line to be Considered ?

Ans:-



Taking Relation → linear

All lines are straight line

Que:- Different notations ? But Same Concept .?

Ans:-

$y = mx + c$ → slope, "y intercept

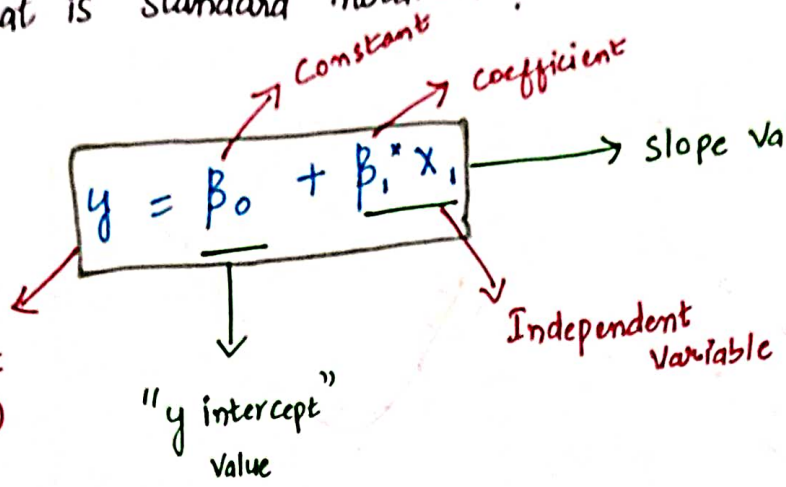
$y = ax + b$ → slope, "y" intercept

$y = b_0 + b_1 x$ → y intercept, slope

$y = \beta_0 + \beta_1 x$ → "y" intercept, slope

Que:- What is Standard notation ?

Ans :-



Ordinary Least squares :

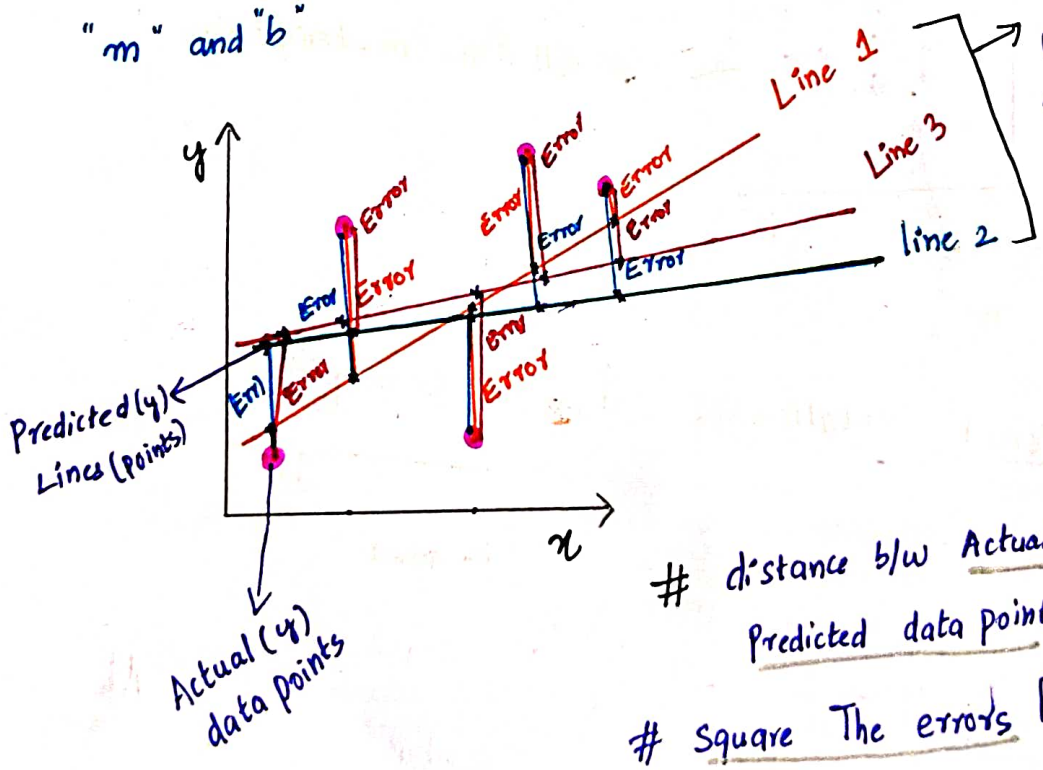
→ Least squares fitting is a way to find The best fit Curve (or) Line for a set of points

→ Sum of Squares of Residuals (Errors) are used to estimate the best fit curve (or) Line

→ Least squares method is used to obtain The coefficient of "m" and "b"

#best fit line ?

Where The Sum of squares of Error is Minimum



distance b/w Actual data point - Predicted data point [Error]

Square The errors [each error do square]

after Squaring Error, Than Sum Error

Que:- What is SSE ?

Ans:- SSE ⇒ Some of squares of Error

Denoted by $\boxed{\Sigma (y-\hat{y})^2}$ → squaring them = SSEmin

Sum, Actual Value, Predicted Value

$\Sigma(y-\hat{y})^2_{min}$ → This value Should be Minimum $\boxed{\Sigma(y-\hat{y})^2_{min}}$

* you Have identify the line Such a way

Que :- How to identify The Exact line ?

Ans :- Explanation :- of Calcus

Minimum point : $y = x^2$ (what is Least "y" value) = 0

if 0.00001 also it get squared

so, "0" is Least y value

how it is identified $y = 0$

$\frac{d(y)}{dx}$ at $x=0$ ⇒ $\frac{dy}{dx}$ at $x=0$ ⇒ $\frac{d(x^2)}{dx}$ at $x=0$ ⇒ $2x \Rightarrow 2(0) = 0$

Ex :- $y = 4x^2+5$ → $4 \times 2 = 8$

$\frac{dy}{dx}$ at $x=0$ ⇒ $8x + 0 \Rightarrow 8(0)+0 \Rightarrow 0$

* $\frac{d}{dx}(4x^2) = 2 \times 4 = [8x]$

* $\frac{d}{dx}(5) = 0$

Mimimum Value $= \frac{dy}{dx}$ at $x = 0$

$SSE_{min} = \varepsilon[y - \hat{y}]^2$

So, in order to get minimum value For S.S.E

Calculate $\frac{d[\varepsilon[y - \hat{y}]^2]}{dx}$ at $x = 0$ → $\hat{y} = \beta_0 + \beta_1 x$

$\therefore \frac{d}{dx} \varepsilon (y - \beta_0 + \beta_1 x)$

Expand This Equation

$$\beta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

"y" Mean, "x" Mean

H_0 : Avg. line (Errors) $\leq$ Regression Line

H_1 : Regression line $<$ Avg. line

Example :

X	Y
4	11
6	4
7	6
8	5
12	8

identifying best fit line for These values

By using $y = \beta_0 + \beta_1 x$ [Linear line] ✓

not in $y = \log x$, e^x, ax^2, $\beta x^{-2} x$

For Average Fit Line

X	Y	y-Pred avg	Error	Error^2
4	11	6.8	4.2	17.64
6	4	6.8	-2.8	7.84
7	6	6.8	-0.8	0.64
8	5	6.8	-1.8	3.24
12	8	6.8	1.2	1.44
				30.8

Avg. line

For Random value $2+5x$ (Eqn)

SSE

For $2+5x$ Equation

X	Y	y-Pred	Error	Error^2
4	11	22	-11	121
6	4	32	-28	784
7	6	37	-31	961
8	5	42	-37	1369
12	8	62	-54	2916
				6151

For $5+4x$ Equation

X	Y	y-Pred	Error	Error^2
4	11	21	-10	100
6	4	29	-25	625
7	6	33	-27	729
8	5	37	-32	1024
12	8	53	-45	2025
				4503

SSE

* Avg $= \frac{11+4+6+5+8}{5} = 6.8$

* Error $\Rightarrow$ (Actual - Predicted Answer) = y - $\hat{y}$

$11 - 6.8 = 4.2$

$4 - 6.8 = -2.8$

$6 - 6.8 = -0.8$

$5 - 6.8 = -1.8$

$8 - 6.8 = 1.2$

* Error^2 $= (4.2)^2 = 17.64$

$(2.8)^2 = 7.84$

$(0.8)^2 = 0.64$

$(1.8)^2 = 3.24$

$(1.2)^2 = 1.44$

* Some of square of Error $= 17 + 7.84 + 0.64 + 3.24 + 1.44 = 30.8$

For, $2+5x$ SSE

$\Rightarrow$ y pred $\Rightarrow$ $2+5(4) = 22$

$2+5(6) = 32$

$2+5(7) = 37$

$2+5(8) = 42$

$2+12(12) = 62$

$\Rightarrow$ Error $(y - \hat{y}) = 11 - 22 = -11$

$4 - 32 = -28$

$6 - 37 = -31$

$5 - 42 = -37$

$8 - 62 = -54$

$\Rightarrow$ Error^2 $= (-11)^2 = 121$

$(-28)^2 = 784$

$(-31)^2 = 961$

$(-37)^2 = 1369$

$(-54)^2 = 2916$

$\Rightarrow$ SSE $= 121 + 784 + 961 + 1369 + 2916$

$= 6151$

For $5+4x$ SSE

* y pred $\Rightarrow$ $5+4(4) = 2$

$5+4(6) = 29$

$5+4(7) = 33$

$5+4(8) = 37$

$5+4(12) = 53$

* Error $y - \hat{y}$ pred $= 11 - 21$

$4 - 29$

$6 - 33$

$5 - 37$

$8 - 53$

* Error^2 $= (-10)^2 = 100$

$(25)^2 = 62$

$(27)^2 = 72$

$(32)^2 = 10$

$(45)^2 = 20$

SSE $= 100 + 625 + 729 + 2025 =$

For Regression line

$y = \beta_0 + \beta_1 x$

$\therefore \beta_0 = \bar{y} - \beta_1(\bar{x})$

$\beta_1 = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\Sigma(x-\bar{x})^2}$

X	Y	$x-\bar{x}_{mean}$	$y-\bar{y}$	$\bar{x}*\bar{y}$	$(x-\bar{x})^2$
4	11	-3.4	4.2	-14.28	11.56
6	4	-1.4	-2.8	3.92	1.96
7	6	-0.4	-0.8	0.32	0.16
8	5	0.6	-1.8	-1.08	0.36
12	8	4.6	1.2	5.52	21.16
				Σ -5.6	Σ 35.2

X Avg: 7.4

Y Avg. 6.8

* Avg. of $X = \frac{4+6+7+8+12}{5} = 7.4$

* Avg. of $Y = \frac{11+4+6+5+8}{5} = 6.8$

* $X-\bar{X}_{(mean)}$ = $4-7.4 = -3.4$, $6-7.4 = -1.4$, $7-7.4 = -0.4$, $8-7.4 = 0.6$, $12-7.4 = 4.6$

* $Y-\bar{Y}_{(mean)}$ = $11-6.8 = 4.2$, $4-6.8 = -2.8$, $6-6.8 = -0.8$, $5-6.8 = -1.8$, $8-6.8 = 1.2$

* $\bar{x}_{mean} * \bar{y}_{mean}$ =

$-3.4 \times 4.2 = -14.28$

$-1.4 \times -2.8 = 3.92$

$-0.4 \times -0.8 = 0.32$

$+0.6 \times -1.8 = -1.08$

$4.6 \times 1.2 = 5.52$

* $\Sigma \bar{x}_{mean} * \bar{y}_{mean}$

$= -14.28 + 3.92 + 0.32 - 1.08 + 5.52 = -5.6$

* $(x-\bar{x})^2$

$= (3.4)^2 = 11.56$

$(-1.4)^2 = 1.96$

$(-0.4)^2 = 0.16$

$(0.6)^2 = 0.36$

$(4.6)^2 = 21.16$

* $\Sigma[x-\bar{x}]^2$

$= 11.56 + 1.96 + 0.16 + 0.36 + 21.16 = 35.2$

$\beta_1 = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\Sigma(x-\bar{x})^2} \Rightarrow \frac{-5.6}{35.2} = 0.15909$ → Slope

$\beta_0 = \bar{y} - \beta_1(\bar{x}) = 6.8 - (0.159)(7.4)$

$6.8 - (-0.159) = 7.9771$ → y intercept

Now, The equation is

$y = \beta_0 + \beta_1 x$

$y = 7.977 - 0.15909\,[x]$

Here $y = 7.977273 - 0.159[x]$ Equation.

X	Y	ypred	Err	Error^2
4	11	7.34	3.66	13.39
6	4	7.02	-3.02	9.12
7	6	6.86	-0.86	0.73
8	5	6.70	-1.7	2.89
12	8	6.06	1.94	3.76
				29.89 → SSE

Equation: $7.977 - 0.159(x)$

$7.977 - 0.159(4) = 7.341$
$7.977 - 0.159(6) = 7.023$
$7.977 - 0.159(7) = 6.864$
$7.977 - 0.159(8) = 6.705$
$7.977 - 0.159(12) = 6.069$

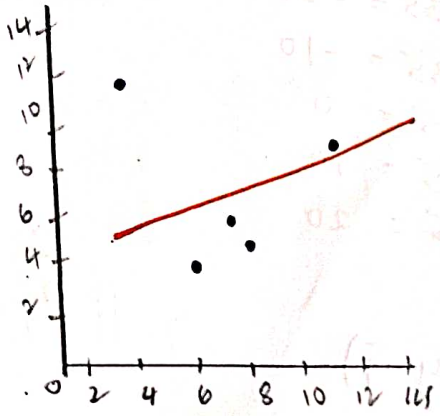
Error
$11 - 7.34 = 3.66$
$4 - 7.02 = -3.02$
$6 - 6.86 = -0.86$
$5 - 6.70 = -1.7$
$8 - 6.06 = 1.94$

Error^2
$= (3.66)^2 = 13.39$
$(-3.02)^2 = 9.12$
$(-0.86)^2 = 0.73$
$(-1.7)^2 = 2.89$
$(1.94) = 3.76$

$SSE = 13.39 + 9.12 + 0.73 + 2.89 + 3.76$
$= 29.89$

Comparing with Avg. line and Other Equations

$y = 7.977 - 0.159[x]$
Best fit line.
$[SSE]_{min}$



For, Average →

For, 10x+5

X	Y	$\bar{y}$	Error $y-\bar{y}$	error^2
1	15	35	+20	400
2	25	35	+10	100
3	35	35	0	0
4	45	35	-10	100
5	55	35	-20	400

SST = 1000

X	Y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x}*y-\bar{y})$	$(x-\bar{x})^2$	$\hat{y}$	Error	Error^2
1	15	-2	-20	40	-4	15	0	0
2	25	-1	-10	10	-1	25	0	0
3	35	0	0	0	0	35	0	0
4	45	1	10	10	1	45	0	0
5	55	2	20	40	4	55	0	0
				100	0			SSE=0

$\bar{x}=3$ $\bar{y}=35$

X Avg $= \frac{1+2+3+4+5}{5} = 3$

y Avg $= \frac{15+25+35+45+55}{5} = 35$

* $x-\bar{x} = 1-3 = -2$

$2-3 = -1$
$3-3 = 0$
$4-3 = 1$
$5-3 = 2$

* $y-\bar{y} = 15-35 = -20$

$25-35 = -10$
$35-35 = 0$
$45-35 = 10$
$55-35 = 20$

* $(x-\bar{x}) \times (y-\bar{y})$

$= -2 \times (-20) = 40$
$-1 \times (-10) = 10$
$1 \times 10 = 10$
$2 \times 20 = 40$

* $\Sigma(x-\bar{x} \times y-\bar{y})$

$= 40+10+0+10+40 = 100$

* $(x-\bar{x})^2$ # For squaring don't consider (+,-)

$= (-2)^2 = +4$
$(-1)^2 = +1$
$(1)^2 = 1$
$(2)^2 = 4$

* $\Sigma[x-\bar{x}]^2 = -4-1+1+4 = 10$

$\beta_0 = \bar{y} - \beta_1 \bar{x}$

$\beta_1 = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\Sigma(x-\bar{x})^2}$

$\beta_1 = \frac{100}{10} = 10$ → slope

$\beta_0 = 35 - 10(3) = 5$ → y intercept

Equation is

$y = \beta_0 + \beta_1 x$

$y = 5+10x$ ✓

*** Ex [For Interv

$\therefore R^2 = 1 - \frac{0_{avg}}{1000_{avg}}$

$R^2 = 1$

Maximum value

8/4/22
3:30am

8/4/22
1:00 pm

* BIVARIATE STATISTICS *

X	Y

⇒ Simple linear regression :-

it is **2 Sample "T" test**

Ho : Avg. line ≥ Regression line

H1 : Regression line > Avg. line.

Q :- How can This can be proved by statistically

A :- Where $(SSE)_{Regression\ line} < (SSE)_{Average\ line}$

Ex :- 120 < 30

⇒ Multiple linear regression :-

X_1	X_2	X_3	X_4	Y

Where we have multiple "x" values and only single "y". We use **"ANOVA"** ↓ applied when **Statistical Test**. Output variable is **"continous"**.

Q :- What is the hypothesis Test apply For Regression ?

A :- **Anova Test** (> 2 samples) Apply For More Than 2 Samples

if we apply Only For 2 samples it is **"2 sample T Te**

* **"correlation"** is high gives "good" results → Linea Regression

"Relation between two variables"

Modulus of 'r' ← $|r| = \pm$ any value

→ $|r| \geq 0.8$ (strong) correlation

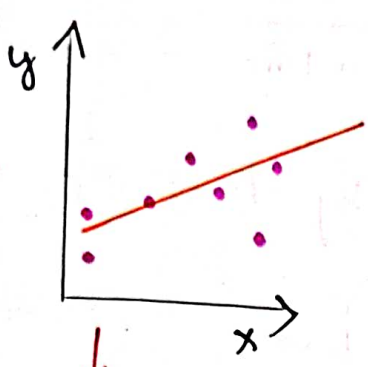
$0.5 < |r| < 0.8$ (moderate)

$|r| < 0.5$ (weak)

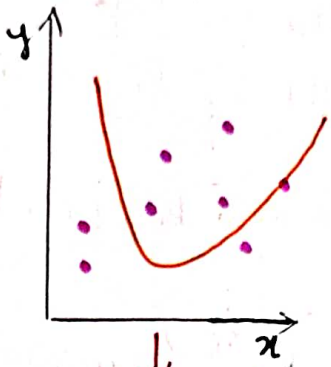
The Value of "One Variable", is a function of "Other Variable"

The Value of y, is a Function of x;

$$y = f(x)$$



$\therefore y = mx + c$



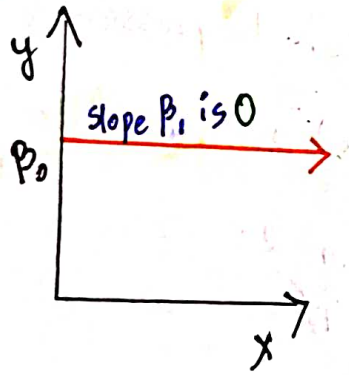
$y = ax^2 + bx + c$

The Value of dependent Variable, is function of Independent Variable

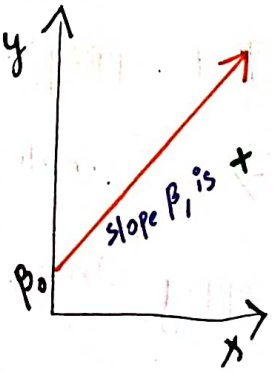
$\Rightarrow E(y) = \beta_0 + \beta_1 x$ → "slope"

→ "y" intercept

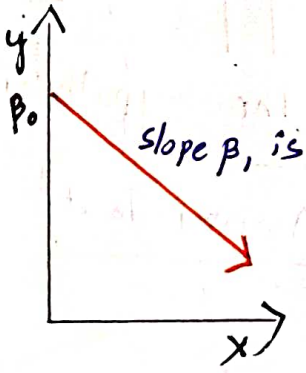
Estimated value



$E(y) = \beta_0 + 0(x)$



$E(y) = \beta_0 + \beta_1 x$



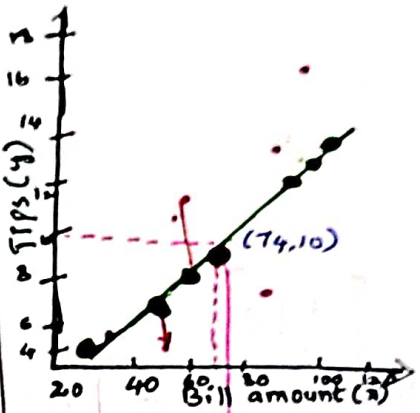
$E(y) = \beta_0 - \beta_1 x$

* Calculations

$y = 0.1462[x] - 0.8188$ Equation

X	Y	$X-\bar{X}$ (Deviations)	$Y-\bar{Y}$ (Deviations)	$(x-\bar{x})[y-\bar{y}]$	$(x-\bar{x})^2$	$y = 0.1462x - 0.8188$	$\hat{y}$
34	5	-40	-5	200	1600	0.1462[34] - 0.8188	4.1505
108	17	34	7	238	1156	0.1462[108] - 0.8188	14.9693
64	11	-10	1	-10	100	0.1462[64] - 0.8188	8.5365
88	8	14	-2	-28	196	0.1462[88] - 0.8188	12.0453
99	14	25	4	100	625	0.1462[99] - 0.8188	13.6535
51	5	-23	-5	115	529	0.1462[51] - 0.8188	6.6359
				$\Sigma = 615$	$\Sigma = 4206$		

$\bar{x} = 74$, $\bar{y} = 10$



Centroid

The line should pass through centroid.

Avg. of x :- $\frac{34+108+64+88+99+51}{6} = \bar{x} = 74$

Avg of y :- $\frac{5+17+11+8+14+5}{6} = \bar{y} = 10$

* Descriptive statistics / Centroid

$\bar{x}, \bar{y} = 74, 10$

* line $[\hat{y} = b_0 + b_1 x]$ min

$b_0 [\bar{y} - b_1 \bar{x}]$

$$b_1 = \frac{\sum [x_i - \bar{x}][y_i - \bar{y}]}{\sum [x_i - \bar{x}]^2}$$

Slope $= \frac{\sum [x-\bar{x}][y-\bar{y}]}{\sum [x-\bar{x}]^2}$

How This Equation is Formed ?

** Variance $(x) = \frac{\Sigma (x-\bar{x})^2}{(n-1)}$ → s

Variance(x) can be written as.

$$\frac{\Sigma (x-\bar{x})(x-\bar{x})}{(n-1)}$$

*** Covariance (x,y) written as

$$\frac{\Sigma (x-\bar{x})(y-\bar{y})}{(n-1)}$$

Substuting & Equating

$$\frac{Cov(x,y)}{V(x)} = \frac{\frac{\sum [x-\bar{x}][y-\bar{y}]}{n-1}}{\frac{\sum [x-\bar{x}]^2}{n-1}}$$

(S.S.E)

* Sum of Squares of Error $= \varepsilon [y - \hat{y}]^2$

$= \varepsilon (y - [\beta_0 + \beta_1 x])^2$

$= \varepsilon [y - \beta_0 - \beta_1 x]^2$

↓ should be minimum

Slope

$$b_1 = \frac{\sum [x - \bar{x}][Y - \bar{Y}]}{\sum [x - \bar{x}]^2}$$

$b_1 = \frac{615}{4206}$

$b_1 = 0.1462$

y intercept $\Rightarrow b_0 = \bar{y} - b_1 \bar{x}$

$b_0 = 10 - 0.1462 [74]$

$b_0 = -0.8188$

Final Equation $\Rightarrow$ $y = b_0 + b_1 x$

$y = -0.8188 + 0.1462 x$

$y = \beta_0 + \beta_1 x$

(or)

$y = \underset{mx}{0.1462 x} \overset{+}{} \underset{c}{0.8188}$

*** "Accuracy" called as "Coeffient of Determination"

$R^2 = 1 - \frac{SSE_{(reg)}}{SST_{(Avg)}}$

"Determing The variation of "y" with respect To variation of "x"

(or)

→ As "x" value changes, how "y" value changes.

(or)

"y" variable Explained by "x" variable.

Que:- What is Max. value of R^2

Ans: $R^2 = 1$ [Max. value]

$(SSE)_{reg} = 0$

$R^2 = -$Infinite

$\therefore R^2 < 0$ (Worst model)

***** Range of R^2.

$[\alpha, 1]$

Infinite

"Regression Squared Error"

$\hat{y}$	Error $y - \hat{y}$	Square Error $(y-\hat{y})^2$
4.1505	(5 - 4.1505) 0.8495	(0.8495)² 0.7217
14.9693	2.0307	4.1237
8.5365	2.4635	6.0688
12.0453	-4.0453	16.3645
13.6535	0.3465	0.1201
6.6359	-1.6359	2.6762

$SSE \Rightarrow \sum = 30.075$

* Avg. line = 120
* Regression line = 30.075

→ difference between these are $\frac{30}{120} \Rightarrow 75\%$.

Co-efficient of Determination.

*** (or)

$$R^2 = 1 - \frac{SSE_{(Reg)}}{SST_{(Avg)}} \rightarrow \frac{SSE\ [\text{Sum of squares of Error [Regress line}}{SST\ [\text{sum of squares of Error [Average line}}$$

$$\sum [y - \hat{y}]^2$$

$$\sum [y - \bar{y}]^2$$

***

$$R^2 = 1 - \frac{\sum [y - \hat{y}]^2_{reg}}{\sum [y - \bar{y}]^2_{Avg}}$$

$$R^2 = 1 - \frac{30}{120} \quad (30 \to 1,\ 120 \to 4)$$

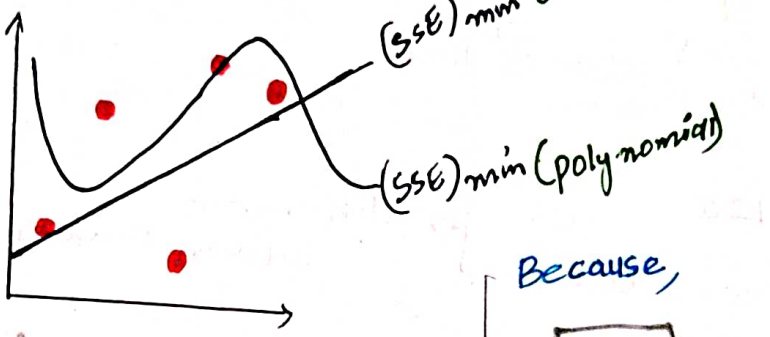
$$R^2 = \frac{4-1}{4} = \frac{3}{4} \Rightarrow 0.75 \Rightarrow \text{75 \% accuracy}$$

8/4/22 4:30 PM

Dt:- 9/4/22
3:00 pm

Evaluation Metrics in Regression

After Model Fitting, we would like to assess the performance of model by **Comparing model predictions** to actual **[True] data.**



1. **MAE** [Mean Absolute Error] → $\frac{\sum |y - \hat{y}|}{n}$

Bad → differentiation

squaring of units

2. **MSE** [Mean Squared Error] → $\frac{\sum [y - \hat{y}]^2}{n}$

Because,

MAE → $\frac{d[|x|]}{dx}$

$\frac{d[|x|]}{dx}$ at $x=0$ → $\frac{x}{|x|} = \frac{0}{|0|} \Rightarrow \alpha$ (Infinite)

MSE → $\frac{dx^2}{dx}$

$\frac{dx^2}{dx} = 2x$ at $x=0$

**** Only way to make Error Minimum * *

$(error)_{min}$ is

$$\frac{d[\,]}{dx} \text{ at } x = 0$$

x is only notation it's 2 after d( )/dy

Example — milage of a car

Y	$\hat{Y}$	$y-\hat{y}$
182	189	0.7

Here 18.2 Kmph → 18.9 Kmph

Error = $(0.7)^2 = 0.49 \frac{km^2}{h^2}$

If we square it

Problem is Here it's also squaring

Que:- What are Evaluation Metrics used for Regression?

Ans:- We have 1. Mean Absolute Error (MAE)
and 2. Mean Square error [MSE].

Que ⇒ **Mean absolute Error (MAE)** Where, we are going To use?

Ans:- MAE is "Absolute Values of Mean" is called mean absolute Error.

But, Problem with $\frac{d}{dx}$ of (Mod) $|x|$ is $\frac{x}{|x|}$ and at $x=0$, The answer is infinite

So, we are not going to Consider "MAE"

Que ⇒ **Mean Square Error (MSE)**

Ans:- Problem with MSE is with calculating of $\frac{d}{dx}$ of x^2 we will get Some Value of $2x$. But, by Squaring the Error. Units are also squaring up it gives wrong prediction. So, we not going to Consider "MSE".

And Solution For This is :-

$$RMSE = \sqrt{\frac{\Sigma(y-\hat{y})^2}{N}}$$

Same as in Variance

we Take Standard deviation

"Root Mean Square Error"

Que :- What is "Root Mean Square Error" RMSE ?

A :- "RMSE" represents Standard deviation of residuals [Errors] difference between model predictions and True Values (Training data)

* RMSE can be easily intrepreted Compared To MSE because RMSE Units match units of Output.

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

(or)

$$\sqrt{\frac{\sum[y-\hat{y}]^2}{n}}$$

Example

squaring of Errors

Converting negetaive Values To positive Values by adding | |

X	$\hat{y}$	Error	MAE	MSE
1	1.4	0.4	0.4	0.16
2	2.2	0.2	0.2	0.04
3	3.8	0.8	0.8	0.64
4	4.1	0.1	0.1	0.01
5	5.6	0.6	0.6	0.36

$\frac{\Sigma}{n} = 0.42$ $\frac{\Sigma}{n} = 0.242$

Root Mean Squar Error $= \sqrt{0.242}$

⇒ **Mean percentage Error [MPE]**

$$MPE = \frac{100\%}{n} \sum_{i=1}^{n} (y - \hat{y}] / y_i$$

In this we calculate Percentage of Error. But We don't consider "positive" or "Negative" value, without absolute operation.

⇒ **Mean absolute Percentage Error [MAPE]**

In This "MAPE" we convert Every value To positive by applying "Absolute" to the Equation.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} |y - \hat{y}| / |y_i|$$

9/4/22 5:10pm

9/9/22
9:30 pm

"Simple Linear Regression"

⇒ Data Science Project life cycle (crisp-dm) → Data mining.

Cross-Industry Standard Process For

crisp-dm → Methodology

1. Bussiness problem Understanding
2. Data Understanding
   - Data collection [D...]
   - Data variables (research - domain Expert)
   - Dataset understandring
3. Data Preprocessing
   - EDA
   - Data cleaning
   - Data Wrangling
   - Train / test split
4. Modelling → Apply Different Algorithms
5. Evaluation → Accuray of The Model checking (or) If not go back and start again
6. Presentation → Visulazation (Tableau, Power BI)

⇒ Basic Liberaries For all Machine Learning

```
# Import numpy as np
# Import Pandas as pd
# Import matplotlib.pyplot as plt
  % matplotlib inline
# Import Seaborn as sns
```

Step 2 : Business Problem Understanding

- Is there a relationship between total advertising Spend and Sales ?
- Our next ad Campaign will have a total spend of $200, how many units do we expect to sell as a result of this ?

Step 2.1 Data collection

, Comma Seperated values

```
# df = pd.read_csv("Advertisimg.csv")
# df.head()
```

Out

Tv	Radio	newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	59.5	18.5
180.8	10.8	58.4	12.9

Step 2.2 Data Understanding

The sample data displays Sales (in thousands of units) for a particular product as a function of advertising budgets (in thousand of dollars) for Tv, radio and Newspaper media

Independent Variables

→ Tv : Advertising dollars spent on Tv for a Single product in a Given market (in Thousands of dollars) → Ex: 230.1 x 1000 already done scaling

→ Radio :- Advertising dollars Spent on Radio

→ Newspaper : Advertising dollars Spent on news paper

Target Variable

→ Sales : Sales of a Single product in a given market (in thousands of widgets) → Ex:- 22.1 x 1000 = 22100 products sold

Step - 2.3 Dataset Understanding

```
# df.info()
```

Out

	column	Non Null Count	Dtype
0	Tv	200 non null	float 64
1	Radio	200 non null	float 64
2	Newspaper	200 non null	float 64
3	Sales	200 non null	float 64

(200, 4)

Que :- If Some one was to spend a total of $200, what would The expected Sales be ?

A :- We have Simplified This quite a bit by Combining all Features into "total spend"

We add new column, total of 3 column Creating new one

```
# df["total_spend"] = df["TV"] + df["radio"] + df["newspaper"]

# df.head()
```

Out

	TV	Radio	newspaper	Sales	total_spend
0	230.1	37.8	69.2	22.1	337.1
1	44.5	39.3	45.1	10.4	128.9
2	17.2	45.9	69.3	9.3	132.4
3	151.5	41.3	58.5	18.5	251.3
4	180.8	10.8	58.4	12.9	250.0

We drop 3 columns, Consider only two columns For Calculating

```
#df.drop(columns = ["TV", "radio", "newspaper"], inplace=True)

# df.head()
```


Out

Sales	Total-Spend
22.1	337.1
10.4	128.9
9.3	132.4
18.5	251.3
12.9	250.3

Step-3.1 Exploratory Data Analysis [EDA]

On The basis of This data. How should you spend advertising money in future? These general questions might lead you to more specific questions:

1. Is there a relation ship between ads and Sales?
2. How strong is that correlation?
3. Given ad spending, can Sales be predicted?

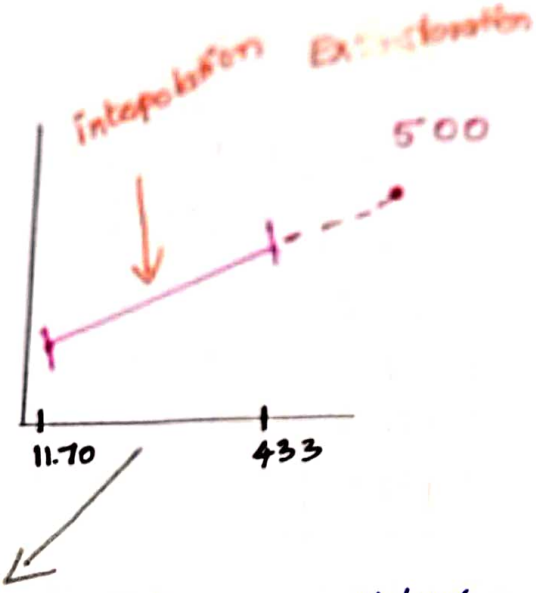
df.describe()

Out :-

	Sales	total_spend
count	200.000000	200.000000
Mean	14.022500	200.860500
std	5.217457	92.985181
min	1.600000	11.700000
25%	10.375000	123.550000
50%	12.900000	207.350000
75%	17.400000	281.125000
max	27.000000	433.600000

How close The values in dataset ← To The mean

Mean and median are close. They normal distribution

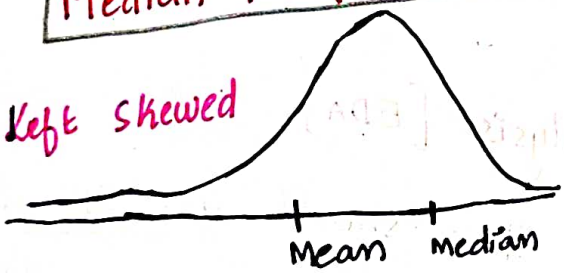


* **Interpolation** :- Estimation of Value with in two known Values in sequence of Values.

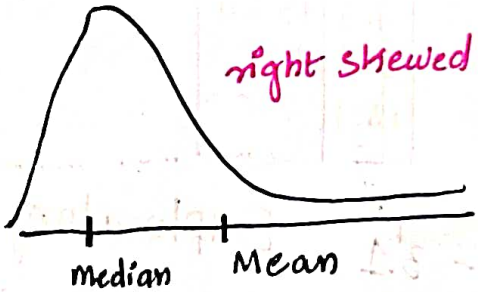
* **Extrapolation** :- Prediction The Value, Outside of known Values it is called Extrapolation.

***

⇒ Median > (greater Than) Mean



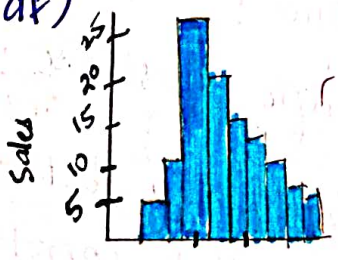
⇒ Median < [Less Than] Mean

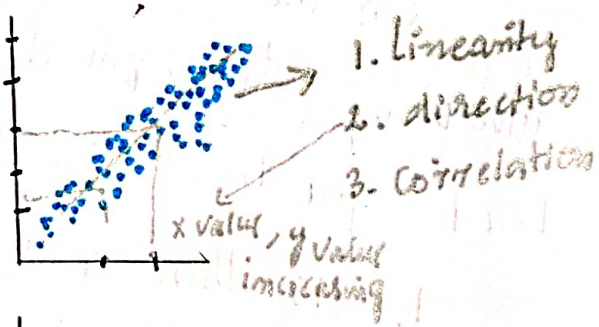


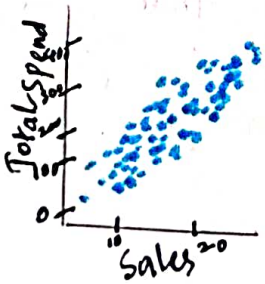
sns.pairplot(df)

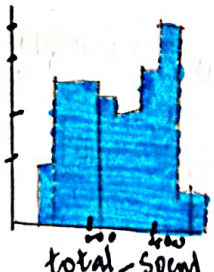
plt.show()

Out :









df.corr()

Out

	Sales	total_spend
Sales	1.000000	0.867712
total_spend	0.867712	1.000000

Step 3.2 Data cleaning

df.isnull().sum()

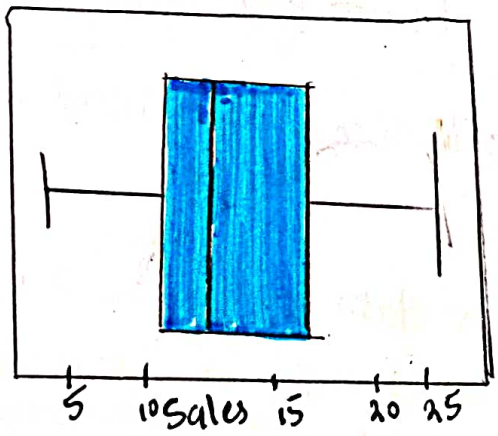
Out

sales 0
total_spend 0

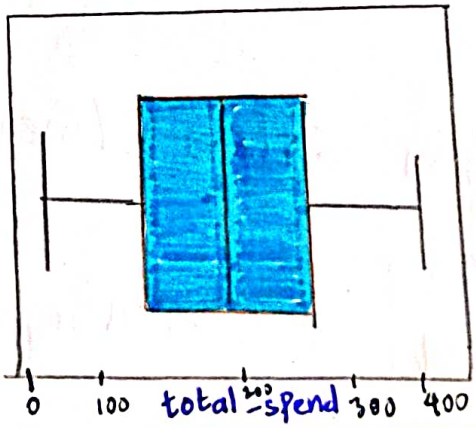
Step 3.3 Data Wrangling

* Outliers

sns.boxplot(df.sales)
plt.show()



sns.boxplot(df.total_spend)
plt.show()



* Feature Transformation

```
# df.sales.skew()
```

Out :- 0.407

```
# df.total_spend.skew()
```

Out : 0.049

STep 3.. Train Test split

```
# X = df.drop(columns = "sale")
# Y = df["sales"]

from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(x, y,
                                     test_size = 0.3, random_state= )
```

STEP-4 Modelling :- If doing with data preprocessing. directly working on modelling is "Base line model (or) raw model.

```
# from sklearn.linear_model import LinearRegression
  storing
# model = LinearRegression()
```

fitting the data

```
# model.fit(x_train, y_train)
```

Out :- LinearRegression()

Attributes (help)

→ y intercept

$\beta_0 + \beta x$

↓ coefficient value

```
# Print("Model Intercept : ", model.intercept_)
  Print("Model Coefficient : ", model.coef_)
```

Out :- Model Intercept : 4.33176

Model Coefficient : 0.0480

$\hat{y} = 4.33176 + 0.0480(x)$

⇒ Predictions

```
# spend = 200
# predicted_sales = 4.33176 + 0.0480 * spend
# predicted_sales
```

Out :- 13.93860

If we have multiple equations, we can write this

```
# model.predict([[200]])
```

→ input should given in 2 dimensional

Out array([13.93860])

Predicting on x_train, x_test

	x	y	ŷ
x Train	1	10	14
	2	15	12
	3	20	13
	4	30	14
		40	33
x Test	5	50	42

→ Train accuracy

→ Test accuracy

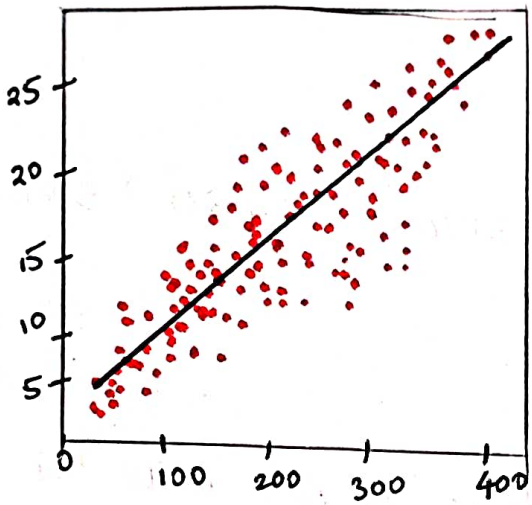
```
# train_predictions = model.predict(x_train)
# test_predictions = model.predict(x_test)
```

Only stores, No output

Regression line

⇒ plotting the Least squares line

```
# plt.scatter (x_train , y_train , color = "red")
# train_predictions = model.predict (x_train)
# plt.plot (x_train, train_predictions , color = "Black")
# plt.show()
```



Steps

⇒ Evaluation metrics

```
from sklearn.metrics Import mean_absolute_error

# print ("MAE For Test data :", mean_absolute_error (y_test,
                                                     test_predictions))

# print ("MAE For Train data :", mean_absolute_error (
                                  y_train , train_predictions))
```

Out: . MAE For Train data : 1.97768
MAE For Test data : 1.90653

```
from sklearn.metrics import mean_squared_error

# print ("MSE for test data : "mean_squared_error (y_test,
                                                    test_predictions))

# print ("MSE for train data : mean_squared_error (y_train,
                                                    train_predictions))
```

Out : MSE for Test data : 6.82220
MSE for Train data : 6.40720

RMSE (Just write np.sqrt).

```
# print ("RMSE for test data", np.sqrt(mean_squared_error
                                        (y_test, test_predictions)))

# print ("RMSE for train data", np.sqrt (mean_squared_error
                                          (y_train, train_predictions)))
```

np.sqrt

Out : RMSE for test data : 2.611935
RMSE for train data : 2.53124

```
from sklearn.metrics import r2_score

# print ("R2 for test data", r2_score (y_test, test_
                                        Predictions))

# print ("R2 for train data", r2_score (y_train, train_
                                         Predictions))
```

Out R2 for test data : 0.74001
R2 for train data : 0.76534

(24)

Another way for R²

```
# model.score(x_train, y_train)
```

Out : 0.76534

```
# model.score(x_test, y_test)
```

Out : 0.740017

Cross_validation ⇒ K-Fold Cross Validation

```
from sklearn.model_selection import cross_val_score
# scores = cross_val_score(model, x, y, cv=5)
```

Here K = 5

```
# print(scores)
# scores.mean()
```

Out : [0.74964192, 0.79455226, 0.76417134, 0.74872042, 0.65980565]

⇒ 0.74337 → should be Equal to Test square
Than it is good model.

Linear Regression assumptions

Errors

L → Linearity
I → Independent
N → Normality
E → Equal variance

10/4/22
3:30 AM

⇒ CHECK LIST ⇐

Assumptions of linear regression

1. check whether model has overfitting (or) underfitting problem
2. Is test Accuracy = Cross Validation Score
3. check assumptions (if it is linear Regression)
4. check model meets bussiness problem requirements
5. Finally, save the model and share to deployment Team.

Purely Assumptions of The Researcher

1 Que :- Is model has overfitting (or) underfitting problem ?

Ans :- it's good model.

2 Que Is test accuracy = Cross Validation Score ?

Ans :- Applied K-Fold cross validation and it is Equal in test accuracy and CV score.

3 Que :- check Assumptions (if it is linear regression)

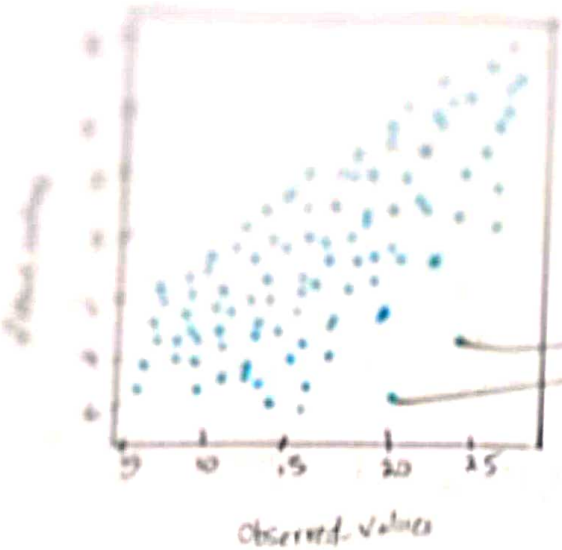
Ans :- Line Assumptions.

1 Linearity of Errors

```
# test_res = y_test - test_predictions.
```

↑ Saved in TRE.

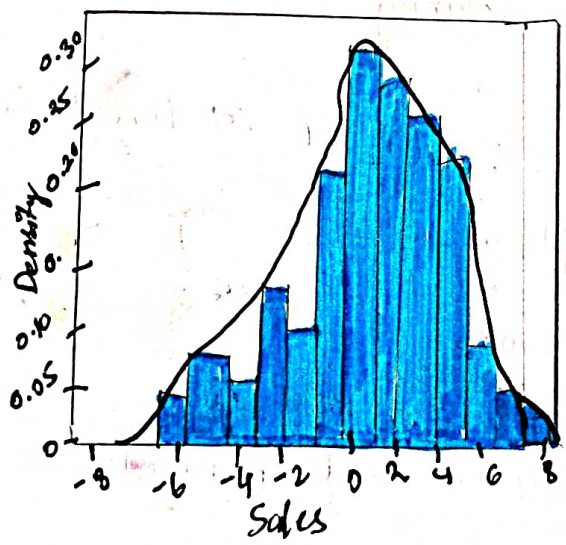
```
# plt.scatter (y_test, test_res)
  plt.xLabel ("Observed_values")
  plt.YLabel ("fitted_values")
  plt.show()
```



2. Normality of Errors *

```
# sns.distplot(test_res, bins = 15, kde=True)
  plt.show()
```

Out



Normal skewed → "Assumption meet"

3. Equal variance of Error (Homoscadesicity)

```
# plt.scatter(test_predictions, test_res, c="r")
# plt.axhline(y=0, color="blue")
```

ax = axis → h = horizontal line

% of Positive Errors
% of negative Errors

color = red

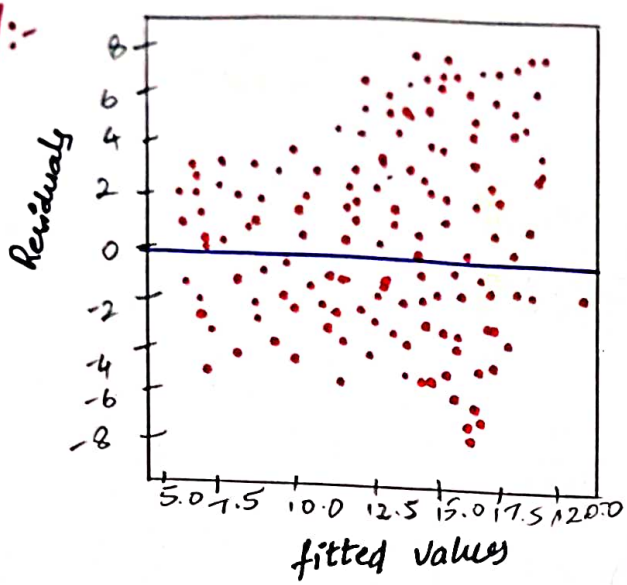
```
# plt.xlabel ("fitted values")
# plt.ylabel ("Residuals")
# plt.show()
```

For model

Ho :- Avg line best fit line
H1 : regression line best line

Anova Test
P: $\leq \alpha$
P $\geq \alpha$

Out :-



1 sample two Tail test

Variable → Significant

Ho :- $\beta_1 = 0$
H :- $\beta_1 \neq 0$

Variable insignificant → don't ⇒ Consider

4. Variable → significance

```
# import statsmodels.formula.api as smf
# m = smf.ols("y~x", data=df).fit()
# m.summary()
```

"y" upon "x"
y multiple column
ordinary least square (SSE) min

Out

OLS Regression Results.

Train

Dep. variable : Y
Model : OLS
method : Least squares

R-squared : 0.753
Adj. R-Squared : 0.752
F. static : 603.4
prob.(F.static) : 5.06×10^{-62}

P = 5.06^{-62}
P < 0.05
Reject Ho

	coeff	std.err	t	P>|t|	[0.025	0.975]
intercept	4.2730	0.499	9.676	0.000	3.378	5.508
x	0.0487	0.002	24.564	0.000	0.045	0.053

Confidence Interval
Std.Error = $\sigma/\sqrt{n}$
$\bar{X} \pm Z \frac{\sigma}{\sqrt{n}}$
Central limit theorem

One sample
Test applied
1 Sample Test
Two Tail
95%
2.5
2.5
Error

Ans: When Average line is going to fail in Regression?

When Sum of Square (SSE) of Average line is Less than the Sum of Squares of regression line $(SSE)_{reg}$ it going To fail in this Situation.

$$(SST)_{avg} < (SSE)_{reg}$$

(Errors)

Step 6 Final Inferences

```
# model.predict([[200]])
```

Out array([13.9898])

⇒ Save a Model

```
from joblib import dump
# dump(model, "Sales_model.joblib")
```

Out: ["Sales_model.joblib"] → Saves in working directory.

⇒ Load a Model → From client side

```
# from joblib import load
```

```
# loaded_model = load ("sales_model.joblib")

# loaded_model.predict ([[200]])

Out  array ([13.989 ])

# loaded_model.predict ([[500]]) # Check with multiple values.

Out :: array ([ [28.3488] ])
```

with 74% Accuracy.

19/4/22
5:00pm

Date :- 11/04/22
10:30 pm

Multiple Linear Regression

#Multiple Linear Regression :-

Examines Relationship between [two] (or) [more Than two] Variables

* Each [independent Variable] has its own corresponding coefficient.

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n x_n$$

dependent Variable

Independent Variables.

⇒ Example :

TV	Radio	np	Sales

TV, Radio, news paper, Sales

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta x_3$$

TV, News paper, newspaper

y- intercept :- $\hat{y}$

Coefficient :- $\beta_1, \beta_2, \beta_3$

Same Example : But different method

Step-1 Bussiness Problem Understanding.

* What is The relation b/w each adversting channel [TV, Radio, Newspaper] and Sales ? $y = f[x_1, x_2, x_3]$
* Previously, we Explored is There a relationship between Total adversting Spend and Sales ? as well as predicting the total Sales For Some Value of Total spend.

Step: 2.1 Data collection

```
# df = pd.read_csv("Advertising.csv")
# df.head()
```

Out

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Step 2.2 Data Understanding

Same as Simple linear Regression Dataset.

Step 2.3 Data Set Understanding

```
# df.info()
```

Step-3.1

Exploratory Data Analysis [EDA]

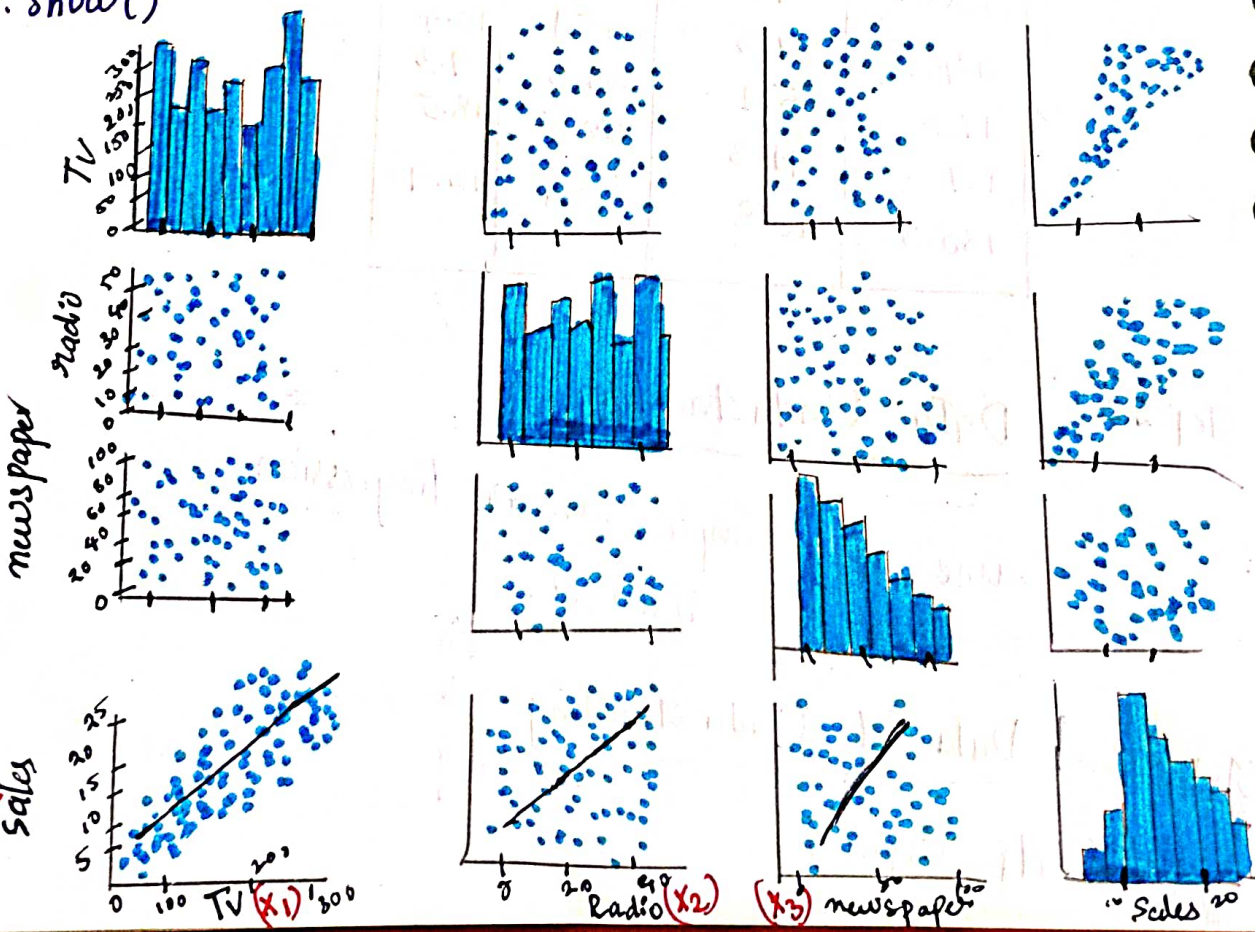
df.describe()

Out:

	TV	Radio	newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
Mean	147.042500	23.264000	30.554000	14.022500
Std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max%	296.40000	49.600000	114.000000	27.000000

sns.pairplot(df)

plt.show()



⇒ By observing the scatter plot, we made an assumption of relation between [y] and [$x_1 + x_2 + x_3$] is Linear

(symmertical Matrix)

df.corr() →

	TV	Radio	News paper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
News paper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

* Where upper Triangle, lower Triangle is same
The diagonal elements = 1

i) y (vs) x_1
ii) y [vs] x_2
iii) y [vs] x_3
→ relation strong → High Accuracy

x_1 [vs] x_2
x_1 (vs) x_3
x_2 [vs] x_3
→ Low ↓
If strong
collinearity problem

⇒ The relation between "y" and "x" be high ...
dependent
Independent variable.

⇒ The higher the value - stronger the correlation

⇒ The relation between any two independent Variables should be "Low"

**** ⇒ if the correlation between any two (2) independent Variables is strong, then it is called "collinearity problem".

Should be "weak"

Should be weak

* idenpendent variable [student 2]

Should be strong

* idenpendent Variable [students]
* dependent Variable [Teacher]

Example

12/4/22
3:30 pm

Dt: 12/4/2
1:00pm

STEP: 3.2 Data cleaning

Same as Simple Linear

STEP 3.3 Data Wrangling

not Same •

STEP 3.4 Train_Test split

```
# X = df.drop(columns = "sales")
# y = df["sales"]

from sklearn.model_selection import train_test_split

# X_train, Xtest, y_train, y_test = train_test_split(x, y,
                                    test_size = 0.3, Random_
                                    state = 29)
```

STEP-4 Modelling

Here, $y = \beta_0 + \beta_x \times [TV] + \beta_2 \times [radio] + \beta_3 \times [Newspaper]$

```
from sklearn.linear_model import LinearRegression
                                  ↑ capital
# model = LinearRegression()
# model.fit(x_train, y_train)
```

Out: LinearRegression()

model.coef_

Out : array([0.04422917, 0.18781641, 0.00758744])
β_1 β_2 β_2

model.intercept_

Out : 2.974097357
β_0

⇒ Predictions → Predicting on "X"

train_predictions = model.predict(x_train)

test_predictions = model.predict(x_test)

STEP: 5 Evaluations → on "y"

from sklearn.metrics import mean_squared_error

test_RMSE = np.sqrt(mean_squared_error(y_test, test_predictions))

train_RMSE = np.sqrt(mean_squared_error(y_train, train_predictions))

print(train_RMSE, test_RMSE)

Out : 1.64082 , 1.7805

model.score (xtrain, y_train) (train R^2)

Out: 0.888817

model.score (x_test, y_test) (test R^2)

Out: 0.905258

⇒ checklist

Que: Is model has Underfitting (or) Overfitting problem?

Ans:- Good model

Que: Is Test Accuracy = Cross Validation Score

Ans:

```
from sklearn.model_selection import Cross_Val_Score
# scores = cross_val_score (model, X, y, cv=5)
# print (scores)
# scores.mean()
```

Out [0.87865198, 0.9176312, 0.92933, 0.81443, 0.895...

→ 0.887106349, compare with xtest, ytest Accuracy.

3. Check Assumptions

* Linearity of errors

test_res = y_test - test_predictions (Xtest)

↓ Residuals

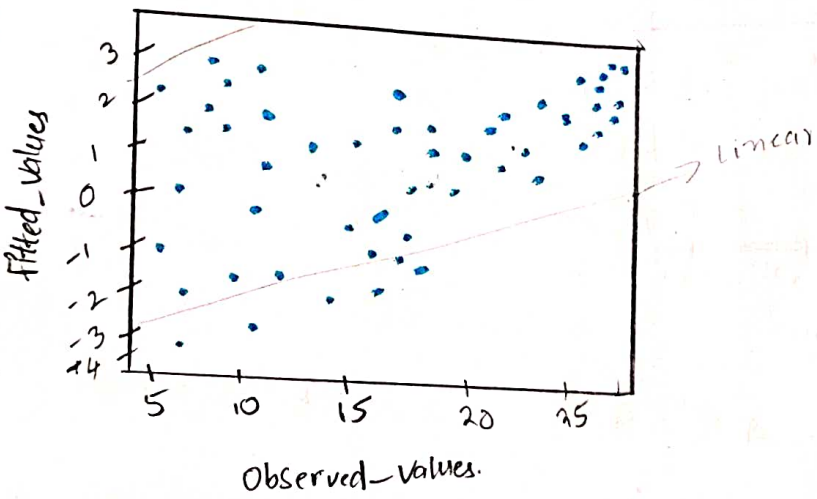
plt.scatter (y_test, test_res)

plt.xlabel ("Observed_values")

plt.ylabel ("fitted_values")

plt.show()

Out:



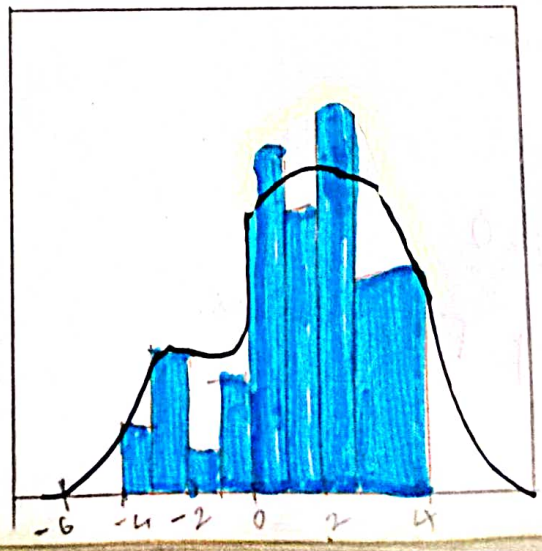
* Normality of errors

sns.distplot (test_res, bins = 15, kde = True)

plt.show()

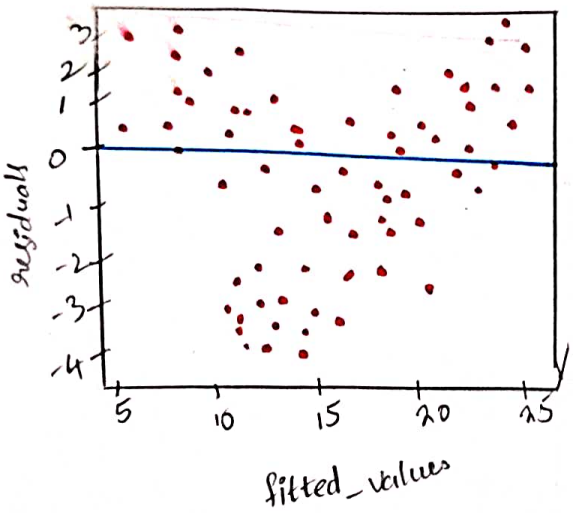
Out:

Left skewed →



3. Equal Variance of Errors (Homoscadesicity) ✗

```
# plt.scatter(test_predictions, test_res, c="r")
# plt.axhline(y=0, color="blue")
# plt.xlabel("fitted_values")
# plt.ylabel("residuals")
# plt.show()
```



* Assumptions Failed, $R^2 = 86\%$ [We reject the model]

⇒ Every condition in checklist should satisfy Than, Only we consider the model

4. Variable Significance *

$H_0 : \beta_1 = 0$

$H_1 : \beta_1 \neq 0$

⇒ Hypothesis Testing For Variables.

import Statsmodels.formula.api smf

```
# model 1 = smf.ols("y ~ x", data = df).fit()
# model 1.summary()
```

new

Out: OLS Regression Results.

Dep variable : Y
Model : OLS
method : least Squars

R-squared : 0.897
Adj. R-squared : 0.896
F statistic : 570.3
Prob (F-statistic) : $1.58e^{-96}$

For Model: Accept H_0 :- 1.58×10^{-96} - $P < 0.05$
↳ Reject H_0

	Coeff	Std err	t	P>|t|	[0.025	0.975]
intercept	2.9389	0.312	9.422	0.000	2.324	3.554
β_1 X(0)	≠0 0.2458	0.001	32.809	0.000	0.043	0.049
β_2 X(1)	≠0 0.1885	0.009	21.893	0.000	0.472	0.206
β_3 X(2)	-0.0010 =0	0.006	-0.177	0.860	-0.013	0.011

Variable wise

Variable 1
H_0 : $\beta_1 = 0$ ✗
H_1 : $\beta_1 \neq 0$ ✓

Variable 2
$\beta_2 = 0$ ✗
$\beta_2 \neq 0$ ✓

variable 3
H_0 : $\beta_3 = 0$ ✓
H_1 $\beta_3 \neq 0$ ✗

$P < 0.05$ → P. low Null. go → Reject H_0

$P > 0.05$

⇒ checking whether data has any influence value using influence index plots

1) * Influence Index plots, For Making "p" value equal < 0.05)

```
import statsmodels.api as sm
# sm.graphics.influence_plot(model1)
```

Why?

$\beta_3 = 0$,

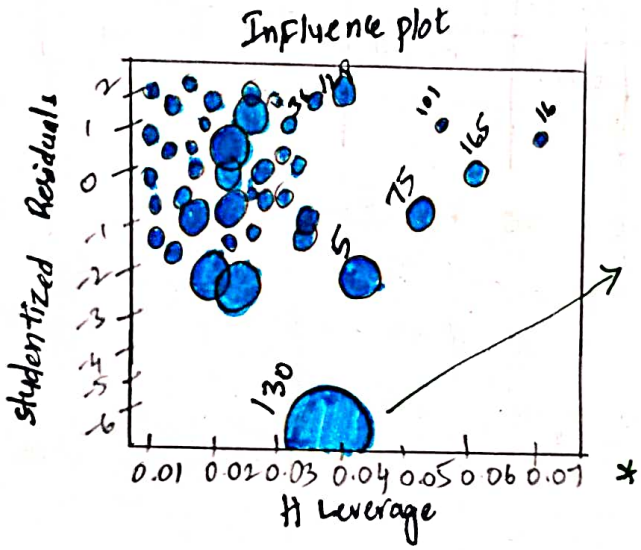
$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x_2 + \beta_3 x_3$

By dropping, we are losing 1/3 of data = 33% data

In That Case, we drop some records, That are influencing More.

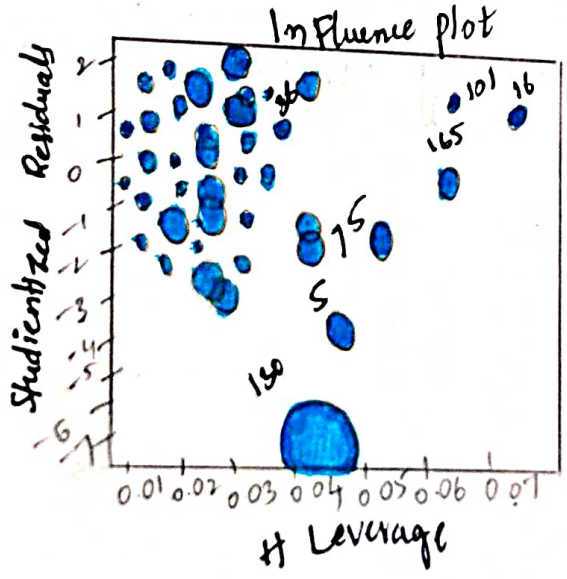
Ex:- 1/200 = 0.05%.

Out:



Index 130 is showing high influence, So, we exclude that entire row.

* Studentized Residuals = Residuals / standard deviation of residuals.



row

```
# df_new = df.drop(df.index[[130]], axis=0)
# df_new.
```

Out

	Tv	Radio	News paper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
⋮	⋮	⋮	⋮	⋮
199	232.1	8.6	8.7	13.4

199 x 4 columns

Once again Rebuild model

```
# model 2 = smf.ols(formula = "Sales ~ Tv + radio + newspaper",
                    data = df_new).fit()

# model2.summary()
```

Out : OLS Regression MODEL

	Coeff	Std.err	t	P>|t|
Intercept	3.0931	0.290	10.654	0.000
Tv X(0)	0.0448	0.001	34.425	0.000
radio X(1)	0.1939	0.009	24.130	0.000
news paper X(2)	-0.043	0.005	-0.777	0.438

→ This is one method of reducing the P value.

→ Reduced From 0.86 to 0.43 By Reducing.

2)

⇒ Variance inflation Factor [VIF]

Variance inflation Factor [VIF] measures ratio between the variance for a given regression coefficient with only that variable in the model versus the variance for given regression coefficient with all variables in the model.

1 independent variable influence on other independent variable is called as ("VIF")

affecting

X_1 X_2 X_3

TV → radio → news

TV ← Radio → news

TV ← Radio ← news

$$V.I.F = \frac{1}{1-R^2}$$

```
# rsq_Tv = smf.ols ("Tv ~ radio + newspaper", data = df)
                                              .fit()

# rsq_Tv.summary()
```

Out : OLS Regression Results

R-squared = 0.005

$$V.IF = \frac{1}{1-(0.05)^2}$$

V.IF = 1.0000

Calculating VIF's values of independent variables

```
# rsq_Tv = smf.ols ("Tv ~ radio + newspaper", data = df).
                                    .fit().rsquared

# Vif_Tv = 1/(1 - rsq_Tv)
```

```
# rsq_radio = smf.ols("radio ~ TV + newspaper", data=df)
                .fit().rsquared

# vif_radio = 1/(1-rsq_radio)

.....

# rsq_newspaper = smf.ols("newspaper ~ radio + TV",
                  data = df).fit().rsquared

# vif_newspaper = 1/(1-rsq_newspaper)

.....

Storing VIF values in a dataFrame

# d1 = {"variables": ["TV", "radio", "newspaper"],
        "VIF": [vif_TV, vif_radio, vif_newspaper]}

# vif_frame = pd.DataFrame(d1)

# vif_frame
```

2nd way : VIF

Out

	variable	VIF
0	TV	1.004611
1	Radio	1.144952
2	newspaper	1.145187

If the VIF model > (greater) 4. For any independent variable, drop that particular variable

Rishu
12/04/22
4:40 pm

3) Dt: 16/9/22

$[SSE)_{min}$

$[E(y-\hat{y})^2]_{min}$

$\Rightarrow$ AV plot [Added Variable plot)

$\frac{d(o)}{dx}$ at $x=0$

Partial differention instead of normal differentition. → Apply $\frac{dou}{dau(x}$ $\Rightarrow$ $\frac{\partial}{\partial x}$

#

What is partial differention ?

$\sum [y-\hat{y}]^2$ → it should be Minimum

$\{[y-[\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3]]^2$ minimum

$\Sigma[y-\beta_0-\beta_1 x-\beta_2 x_2+\beta_3 x_3]]^2$ minimum

In Simple linear Regression ?

$\frac{d}{dx}[E[y-\beta_0-\beta_1 x_1)^2]$ at $x=0$ → He we have only one variable.

In Multiple linear Regression ?

Constant Calculate Constant

$\frac{\partial}{\partial x}\Sigma[y-\beta_0-\underline{\beta_1 x}-\underline{\beta_2 x_2}-\underline{\beta_3 x_3}]]^2$

calculate Constant constant

↓

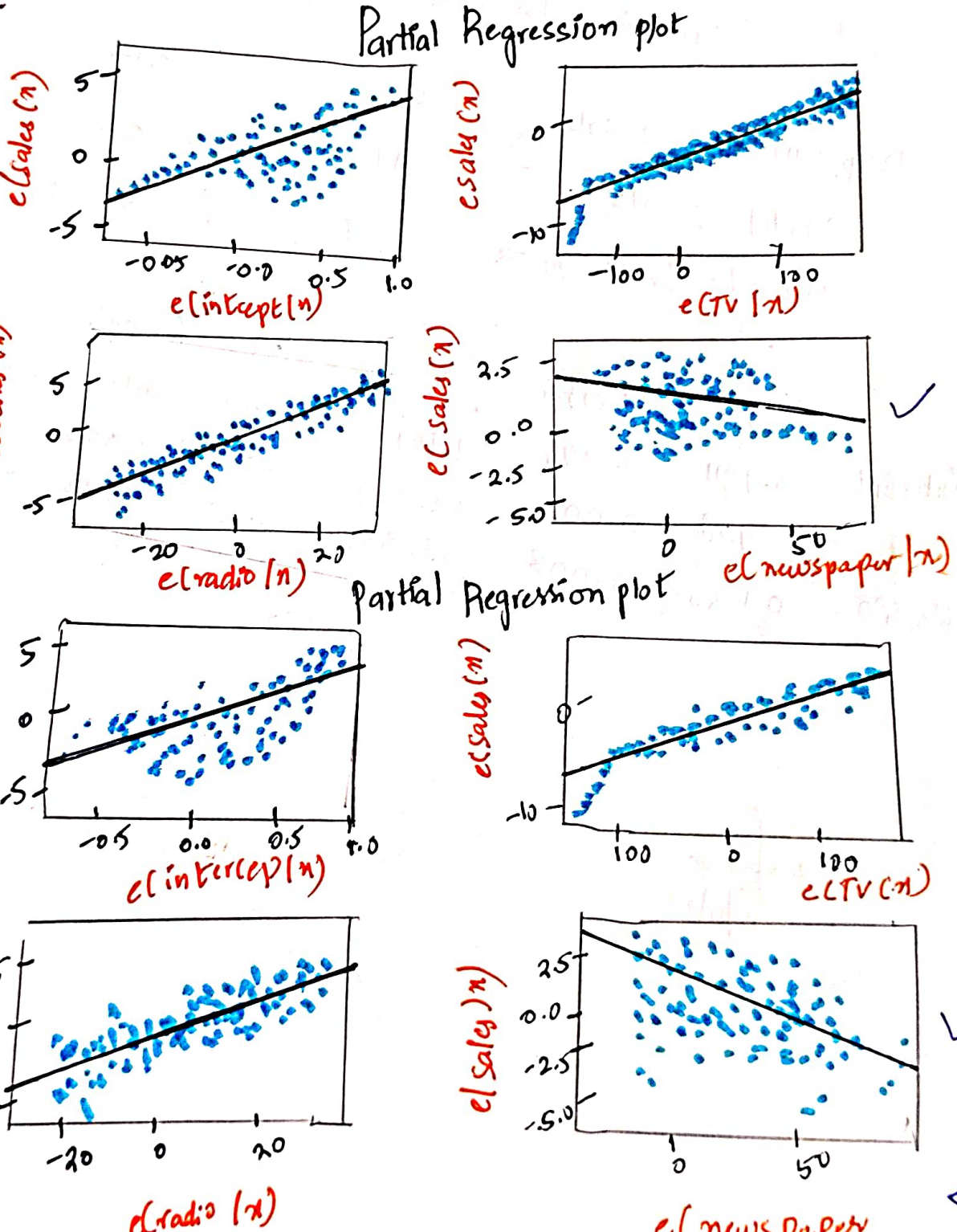
when we calculate with $\frac{dou}{dau(x)}$ $\frac{\partial}{\partial x}$.

When we calculate with one variable other variable will be Constant.

⇒ it is apply Simple Linear Regression on x_1 and x_2 Simple Linear Reg on x_3. it is Applying individually S.L.R on Each and Every individual. variable.

sm.graphics.plot_(partregress)- grid (lm)
Partregress

Out:



Added variable plot is not showing any significance For

⇒ Final model including "TV" and "Radio" Only

```
# final_model = smf.ols(formula = "Sales ~TV + radio",
                        data = df).fit()

# final_model.summary()
```

Out

OLS Regression Results

Dep. Variable : Sales
Model : OLS
Method : Least Squares

R-squared : 0.897 = 90%
Adj. R-squared : 0.896
F-statics = 859.6

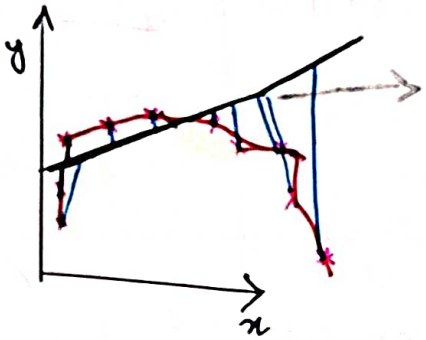
	Coeff	Std. error	t	P(>|t|)	(0.025)	(0.975)
Intercept	2.9211	0.294	9.919	0.000	2.340	3.502
TV	0.0458	0.001	32.909	0.000	0.043	0.048
Radio	0.1880	0.008	23.382	0.00	0.172	0.204

16/4/22

Dt: 16/04/23
1:00pm

Polynomial Regression

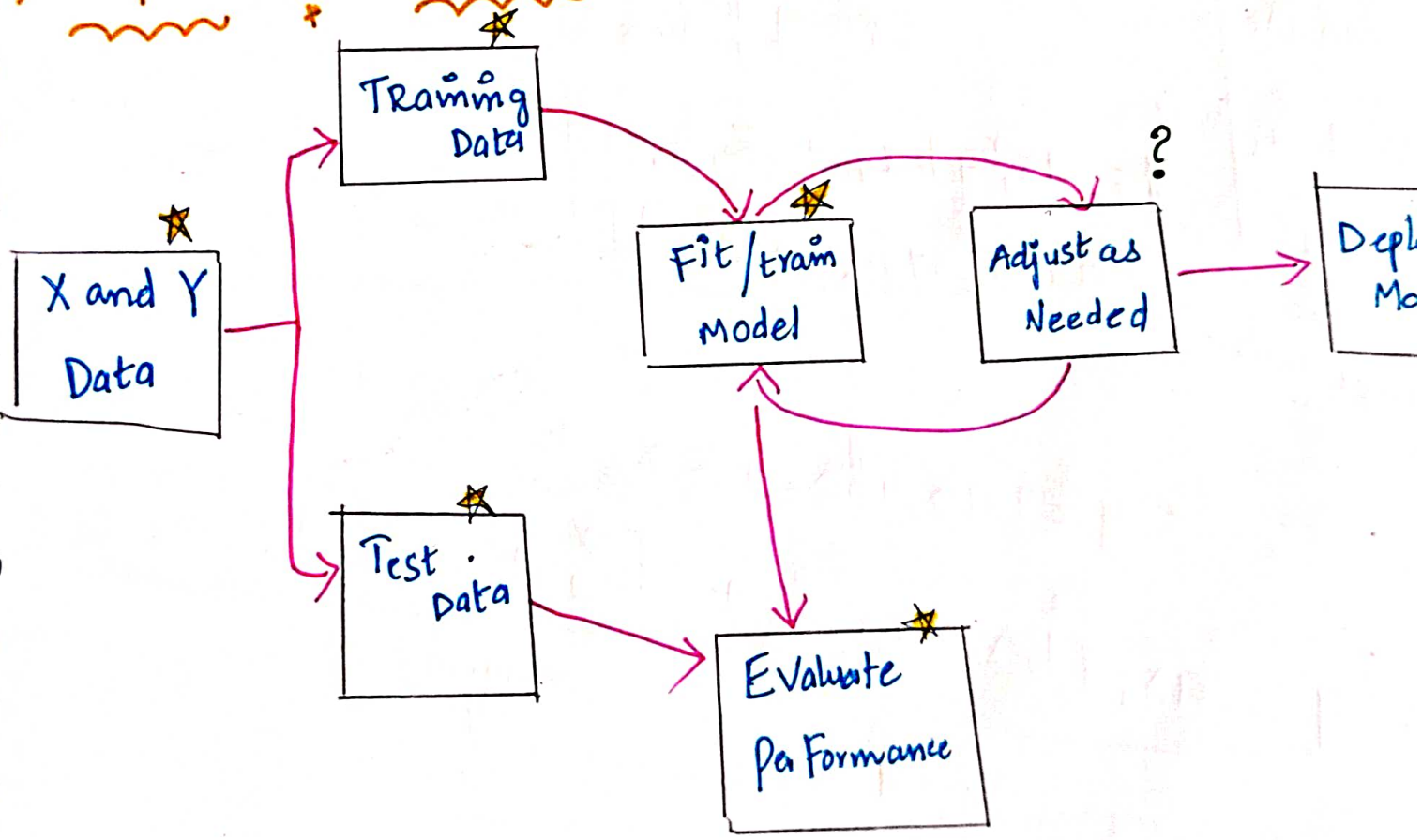
⇒ Any Regression problem. $[SSE]_{min}$ ⇒ $[\varepsilon[y-\hat{y}]^2]_{min}$



straight line (Linear line)

$[\varepsilon(y-\hat{y})^2]$

⇒ Supervised Machine learning process



S.L.R → 1 Variable

$y = ax + b$

Linear

M.L.R → 2 Variables

$y = ax + b + cx + d$

$y = ax_1 + bx_2 + c$

it is came from joining two equations

$= ax_1 + b + cx_2 + d$ (b, d: intercepts)

$y = ax_1 + cx_2 + e$ (always intercept have one value only)

3 Variable

$y = ax_1 + bx_2 + cx_3 + d$

it is came from joining three equations

$y = ax_1 + b$

$y = cx_2 + d$

$y = ex_3 + f$

Quadratic

1 Variable

$y = ax^2 + bx + c$

Here we write intercept value as "Z"

$y = ax_1^2 + bx_1 + dx_2^2 + ex_2 + z$

it is came from joining two equations

$y = ax_1^2 + bx_1 + z_1 c_1 + dx_2^2 + ex_2 + z_2 f_2$

* $y = ax_1 + bx_2 + cx_3 + z$

* $y = ax_1 + bx_2 + cx_3 + dx_1^2 + ex_2^2 + fx_3^3 + z$

(squaring of)

$+ gx_1x_2 + hx_2x_3 + ix_3x_1$

Combinations of variable [iteration terms]

original data | Squaring original Data | Combination (or) Iteration terms

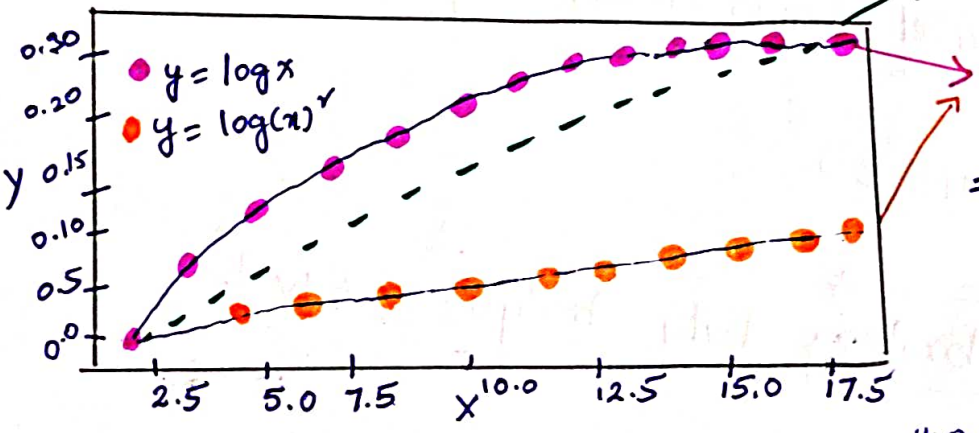
TV	Radio	newspaper	Sales	TV^2	$Radio^2$	$Newspaper^2$	TV*Radio	Radio*Newspaper	Newspaper*TV
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—

* Quadratic equation : $x_1' * x_2' = x_1' + x_2'$?

multiply → degree 2 ; if we add two variables → degree 1

$a^m \times a^n = a^{m+n}$

⇒ Polynomial Regression



#quadratic – ax^2+bx+c

⇒ If we want to calculate Combinations with Squaring.

* Degree 2 = TV^2 (01) $x_1^2+x_2^2+x_3^2$ + $x_1x_2+x_2x_3+x_3x_1$ + $x_1+x_2+x_3$ (interaction terms)

* Degree 3 = $x_1^3+x_2^3+x_3^3$ + $x_1x_2^2+x_2x_3^2+x_3x_1^2$ + $x_1^2x_2+x_2^2x_3+x_3^2x_2$ + $x_1^2+x_2^2+x_3^2$ + $x_1+x_2+x_3$

cubic Equation : ax^3+bx^2+cx+d

Step 1 — Problem Understanding

Step 2 — Data Collection

```
# df = pd.read_csv("Advertising.csv")
# df.head()
```

Step 3 — Data Understanding

```
# df.shape
```

Step 4 — Dataset Understanding

```
# df.info()
```

Step 5 — Exploratory Data Analysis [EDA]

```
# df.describe()
# sns.pairplot(df)
```

Step 6 — Data cleaning

```
# df.isnull().sum()
```

Step 7 — Data wrangling

```
# X = df.drop("sales", axis=1)
# y = df["sales"]
```

Adding new Features to original Data

Polynomial Regression with Sci-kit Learn

```
from sklearn.preprocessing import PolynomialFeatures
# polynomial_converter = PolynomialFeatures(degree=2, include_bias=False)
```

include_bias = False → don't include "z" value

$ax_1^2 + bx_2^2 + cx_3^2$
$dx_1x_2 + ex_2x_3 + fx_3x_1$
$gx_1 + hx_2 + ix_3$

```
# X_poly = polynomial_converter.fit_transform(X)
```

fit_transform: Calculate, Convert

```
# X_poly.shape
```

Out (200, 9) — changed 3 to 9

rows = 200, columns

STEP

```
from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.3, Random_state=29)
```

200 → 140 × 9 = train, 60 × 9 = test

MODEL Fitting On Polynomial Data.

```
from sklearn.linear_model import LinearRegression

# model = LinearRegression()
# model.fit(x_train, y_train)
```

Predictions

```
# train_pred = model.Predict(x_train)
# test_pred = model.predict(X_test)
```

Evaluation

```
# model.score(x_train, y_train)   [# Train R^2]
```

Out 0.986

```
# model.score(x_test, y_test)   [# Test R^2]
```

Out 0.980

Cross - Validation

```
from sklearn.model_selection import cross_val_score

# scores = cross_val_score (model, X_poly, y, cv=5)

# print (scores)

# scores.mean()
```

Out [0.987 , 0.989, 0.991, 0.958, 0.993]

mean : 0.984

RMSE

```
from sklearn.metrics import mean_squared_error

# test_RMSE = np.sqrt (mean_squared_error (y_test, test_pred))

# train_RMSE = np.sqrt (mean_squared_error (y_train, train_pred))

# print (train_RMSE, test RMSE)
```

Out 0.5950 , 0.7233

Earlier,

Multiple Linear Regression
* RMSE - 1.94

Polynomial Regression
* RMSE - 0.72

Applying loop for knowing better Accuracy From First to last.

- from sklearn.preprocessing import PolynomialFeatures
- from sklearn.model_selection import train_test_split
- from sklearn.linear_model import LinearRegression.

```
# train_rmse_errors = [ ]
# test_rmse_errors = [ ]

for d in range [1,10):
# polynomial_converter = PolynomialFeatures (degree = d, include_bias = False)
# X_poly = polynomial_converter.fit_transform(x)
# X_train, X_test, y_train, y_test = train_test_split (X_poly, y, test_size = 0.3, Random_state = 29)
# model = LinearRegression()
# model.fit (x_train, y_train)
# train_pred = model.predict (x_train)
# test_pred = model.predict (X_test)
# train_RMSE = np.sqrt (mean_squared_error (y_train, train_pred)
# train_rmse_errors.append (train_RMSE)
# test_RMSE = np.sqrt (mean_squared_error (y_test, test_Pred)
# test_rmse_errors.append (test_RMSE)
```

train_rmse_errors

Out : 0 [1.7345 d=1 ← degree
1 0.5879 d=2
2 0.4339 d=3
3 0.3517 d=4
4 0.2509 d=5
5 0.19704 d=6
6 5.4214 d=7
7 0.14180 d=8
8 0.16654 d=9

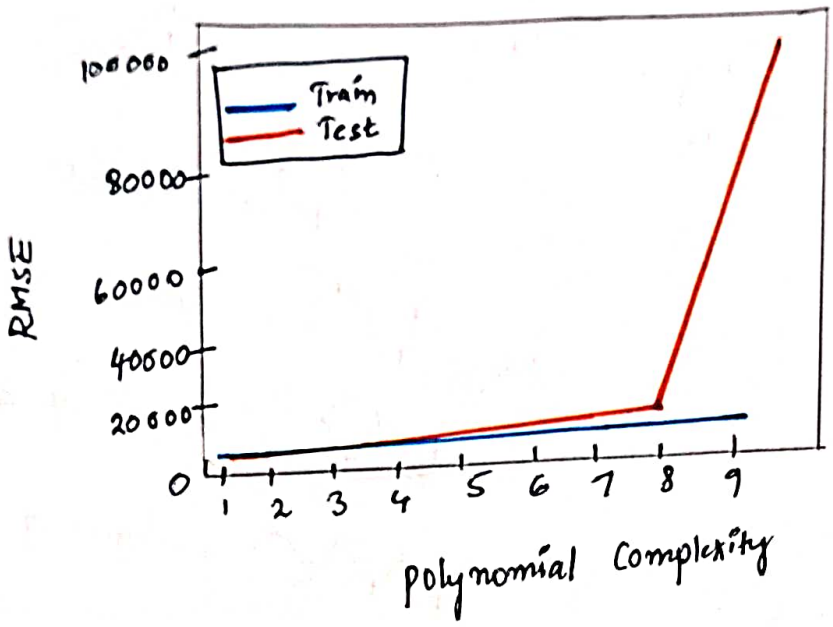
No. of bends
↓
∪ = Quadratic
∩∪ = Cubic
∪∩∪ = degree of 4
∪∩∪∩∪∩∪ = degree of 7

test_rmse_errors

Out : 0 1.5161 d=1 ↓
1 0.6646 d=2
2 0.5803 d=3
3 0.5077 d=4
4 2.5758 d=5 ↑
5 4.4926 d=6
6 1381.404 d=7
7 4449.599 d=8
8 9591.24 d=9

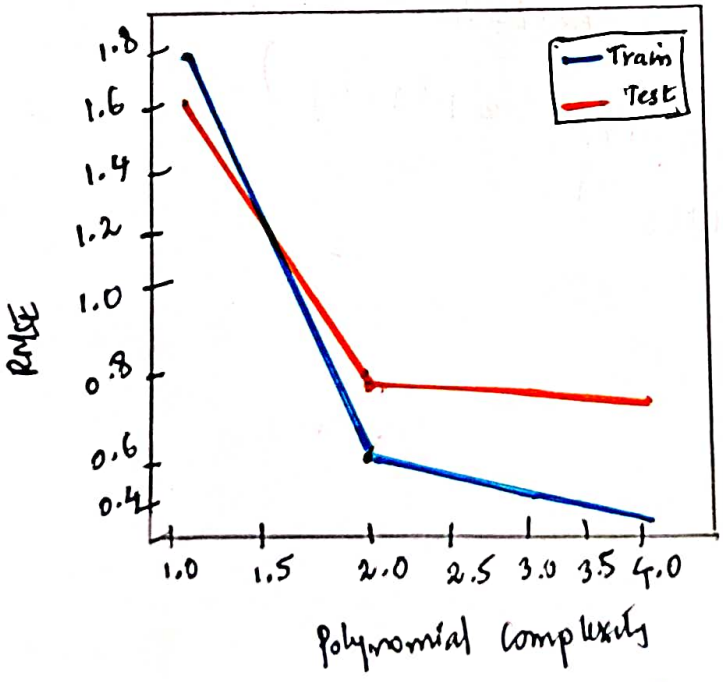
plt.plot (range (1,10), train_rmse_errors, Label = "TRAIN")
plt.plot (range (1,10), test_rmse_errors, Label = "TEST")
plt.xlabel ("Polynomial complexity")
plt.ylabel ("RMSE")
plt.legend()
plt.show()

Out :




```
# plt.plot(range(1,5), train_rmse_errors[:4], label = "TRAIN")
# plt.plot(range(1,5), test_rmse_errors[:4], label = "TEST")
# plt.xlabel("Polynomial Complexity")
# plt.ylabel("RMSE")
# plt.legend()
# plt.show()
```

Out :



S.No.5 Finalizing Model choice

```
# final_poly_converter = polynomial Features (degree = 2,
                                              include_bias = False)

# final_model = Linear Regression()
# final_model.fit (final_poly_converter.fit_transform(x), y)
```

Out: Linear Regression()

* Saving MODEL and CONVERTER *

```
from joblib import dump
# dump (final_model, "Sales_poly_model.joblib")
```

Out ["Sales_poly_model.joblib"]

```
# dump (final_poly_converter, "poly_converter.joblib")
```

Out: ["poly_converter.joblib"]

Deployment & predictions :

Client wants to spend 149k on "TV", 22k on "radio" 12k on "Newspaper" Ads. How many units could we expect to sell as a result ?

```
from joblib import load
```

```
# Loaded_poly = load ("poly_converter.joblib")
# loaded_model = load ("Sales_poly_model.joblib")

-
# Campaign_poly = loaded_poly.transform ([[149, 22, 12]])

-
# final_model.predict (campaign_poly)
```

Out: array ([14.51114])

16/04/22
9:25 pm

* **DBScan Clustering**

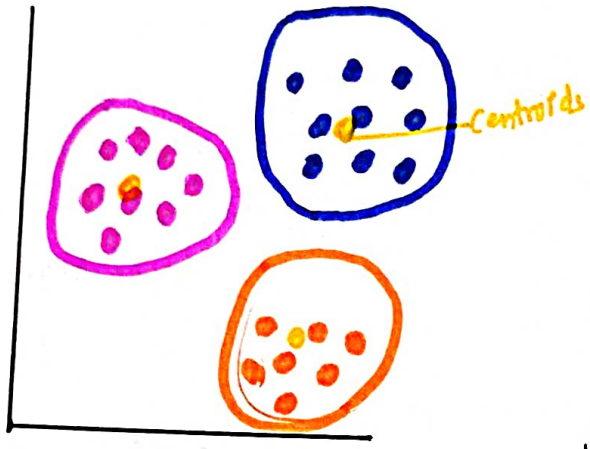
↓

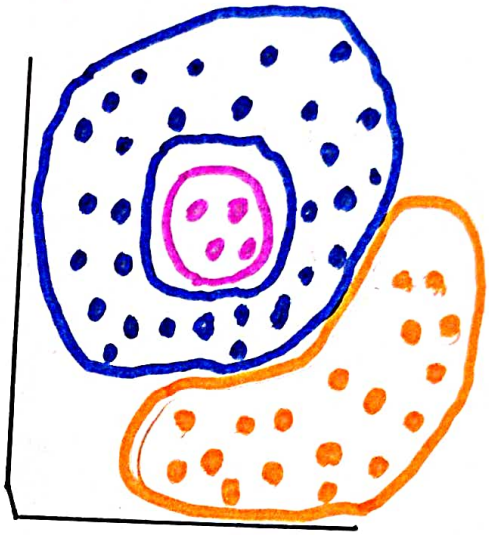
Density Based Spatial Clustering of application with noise.

Density-Based clustering

* Spherical-shape clusters

* Arbitary shape clusters





- K-means assigns all points To a cluster Even if they do not belong in any

- Density based clustering Locates regions of "high density", and seperates outliers.

*** Based on Given density of values it is identifying the clusters

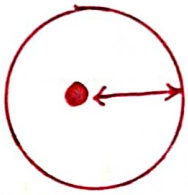
* What is DBSCAN?

* DBSCAN

- it is one of the most common clustering algorithms
- Works based on density of objects.

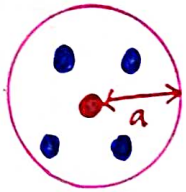
* R (radius of neighborhood)

- Radius (R) that if includes enough number of points within, we call it a dense area.

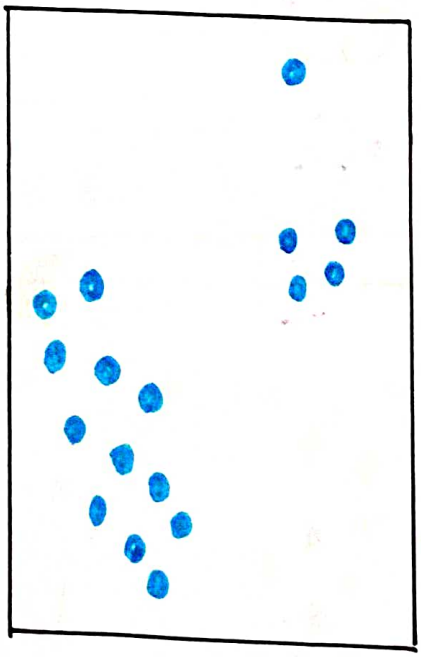


* M (min number of neighbors)

- The minimum number of data points we want in a neighborhood to define a cluster.



Ex:-



Each point is either:

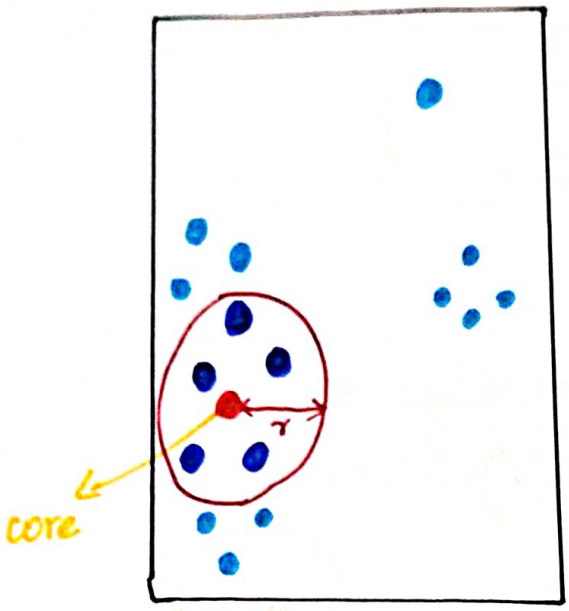
- core Point
- border point
- Outlier point

$\therefore R = 2 unit, M = 6$

* DBSCan algorithm :: Core point

↓

which is satisfy minimum no. of neighbours.



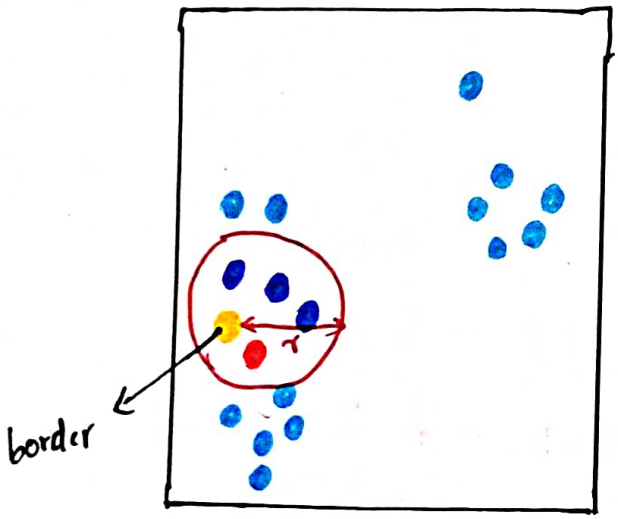
R = 2 unit, M = 6

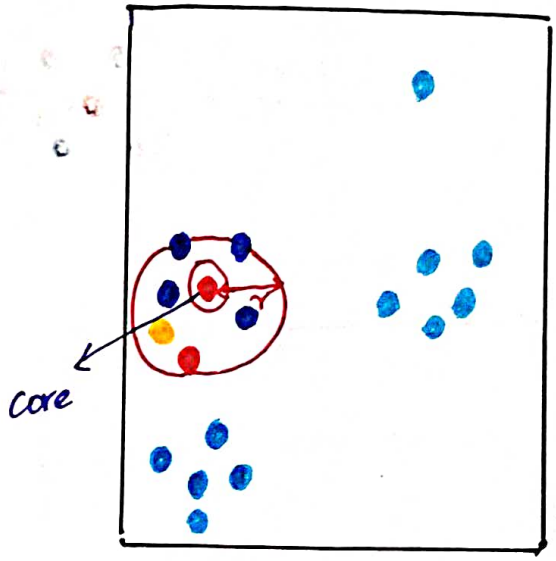
↓

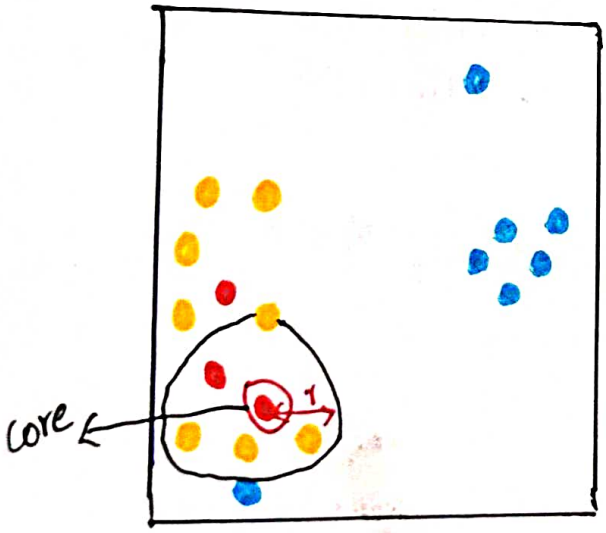
minimum.

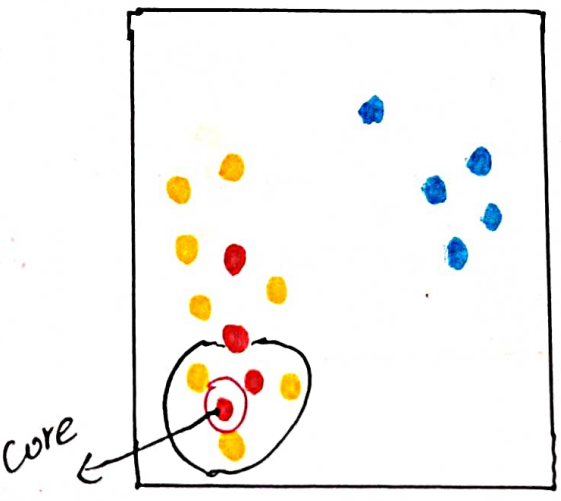
* DBSCan algorithm : border point

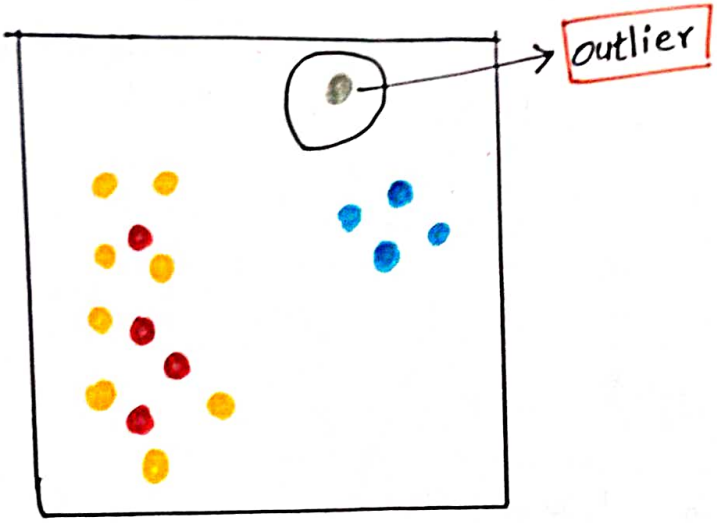
R = 2 unit , M = 6

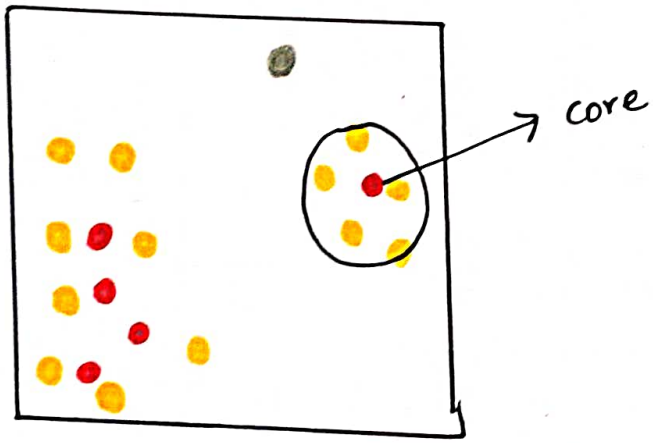


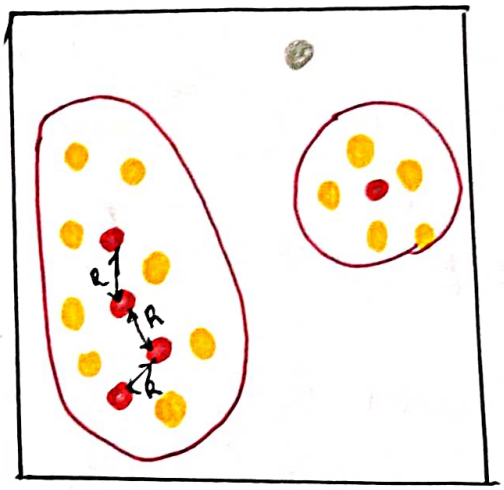












R = 2unit , M = 6

CODE : Same data & Same steps till modelling

MODEL: DBSCAN

```
from sklearn.cluster import DBSCAN
# dbs = DBSCAN (eps = 5 , min_samples = 5)
                ↳ Radius.
```

Predict

```
# y_dbs = dbs.fit_Predict (x)
```

y-dbs

Out : array([-1, 0, -1, 0, -1, 0, -1, -1, 0, 0, -1
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - -]).

np. unique (y-dps)

Out: array ([-1, 0, 1, 2, 3, 4]).

Visualising the clusters

cluster 1

# plt. scatter (x [y-dbs == -1, 0], x [y-dbs == -1, 1],
s=100, c = "red", label= "cluster 1")

cluster 5

# plt. scatter (x [y-dbs == 0, 0], x [y-dbs == 0, 1],
s = 100, c = "magenta", label= "cluster 5")

cluster 2

# plt. scatter (x [y-dbs == 1, 0], x [y-dbs == 1, 1],
s= 100, c = "blue", label= "cluster 2")

cluster 3

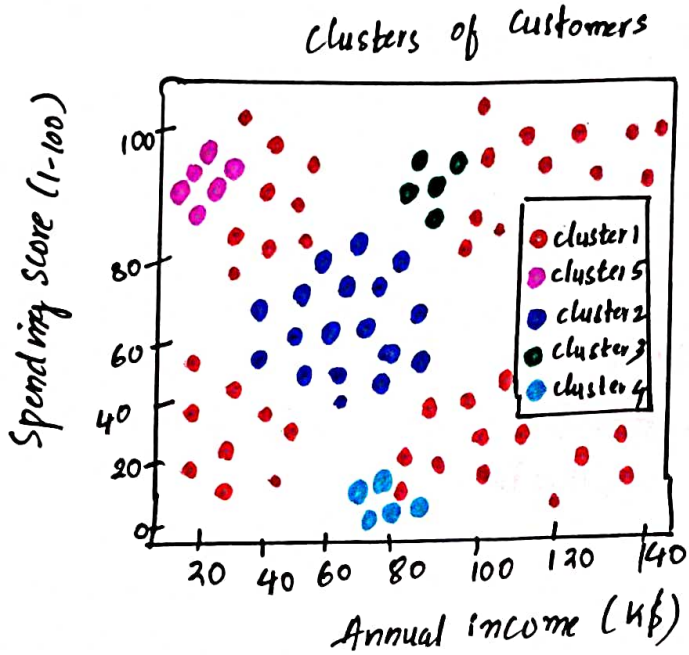
# plt. scatter (x [y-dbs == 2, 0], x [y-dbs == 2, 1],
s=100, c = "green", label = "cluster 3")

cluster 4

# plt. scatter (x [y-dbs == 3, 0], x [y-dbs == 3, 1],
s=100, c = "cyan", label = "cluster 4")

```
#plt.title("clusters of customers")
# plt.xlabel("Annual income (k$)")
# plt.ylabel("spending score (1-100)")
#plt.legend()
# plt.show()
```

Out:

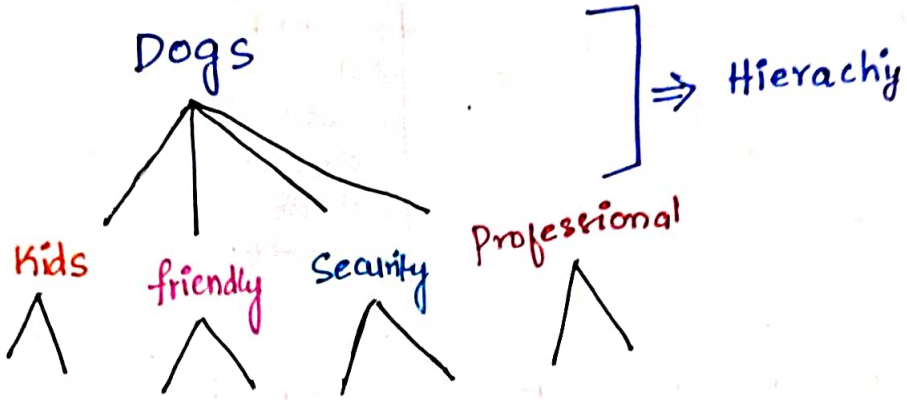


04/05/22
5:00pm.

* Hierarchical Clustering [HC]

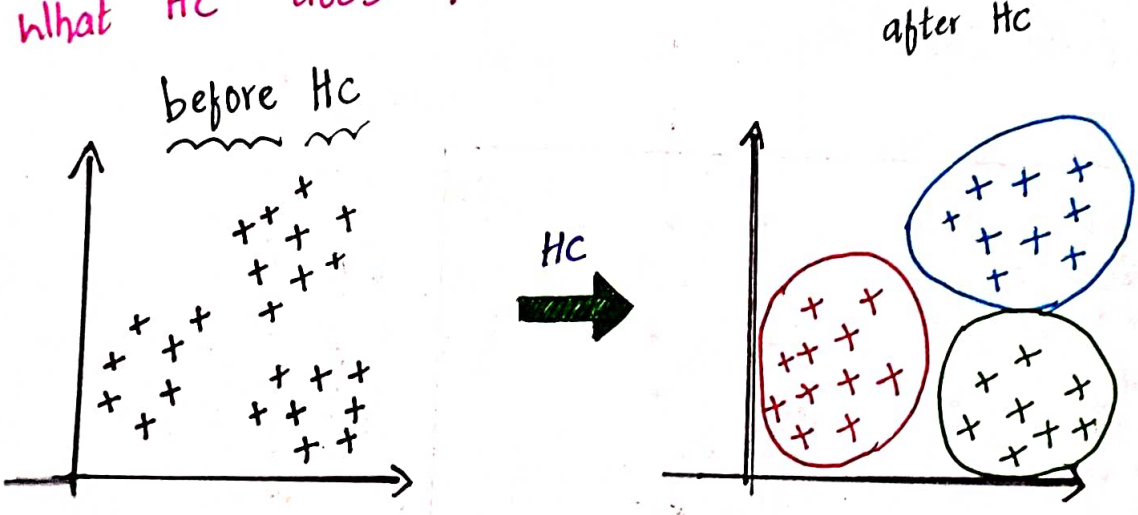
↓ Combine data points which are very close by distance

Ex:-



* Hierarchical clustering algorithms build a hierarchy of clusters where "Each node is a cluster" consists of clusters of its daughter nodes.

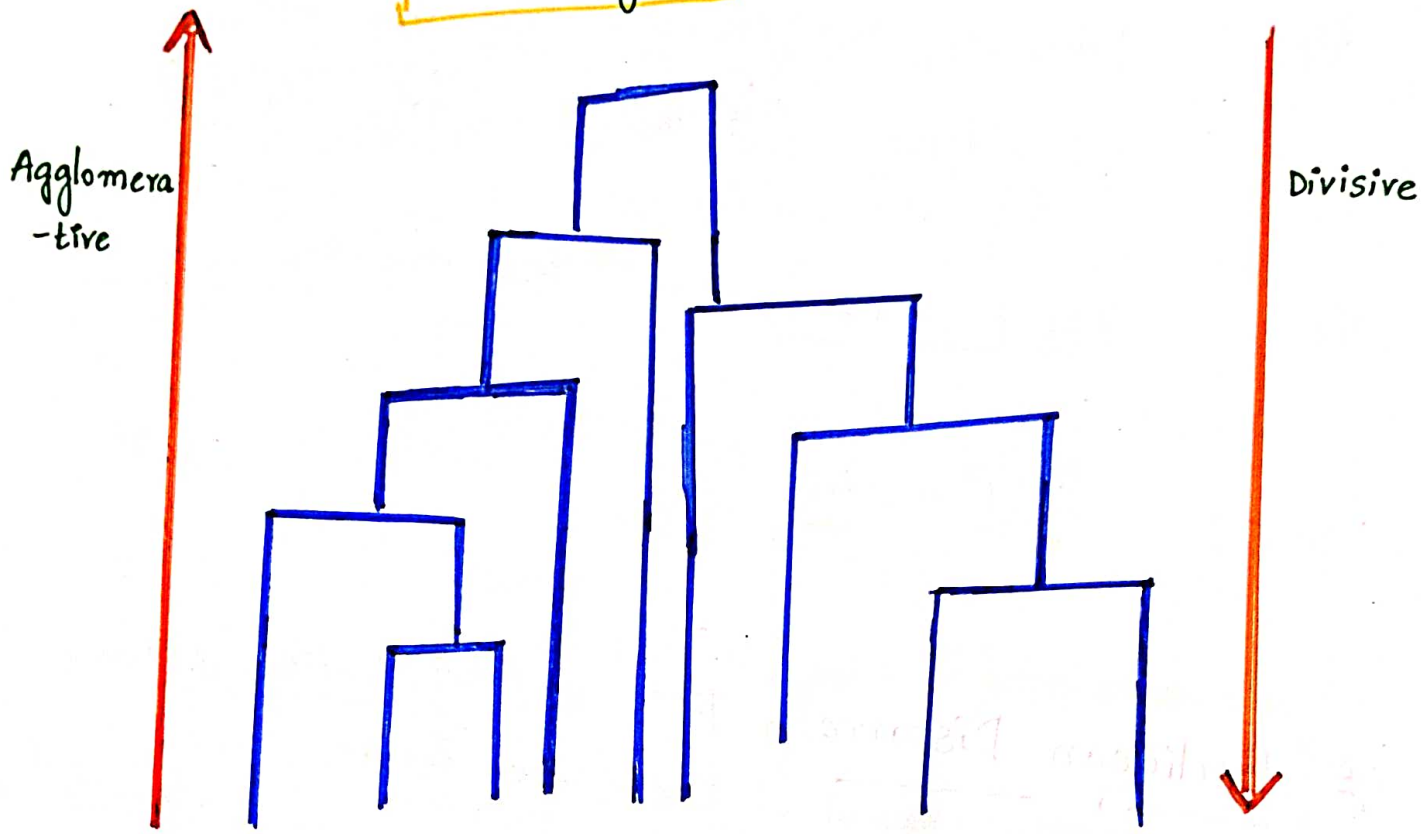
What "Hc" does ?



same as k-means but different process.

* Hierarchical clustering

Dendogram



* They are two types clussering.
  * Algomerative (From bottom to top)
  * Divisive (From Top to bottom)

Most of The Time, we use, This Techinque.

⇒ Steps of Hierachical clustering of (Algomerative)

STep : 1 Make Each data point a single point cluster (# That forms N clusters)

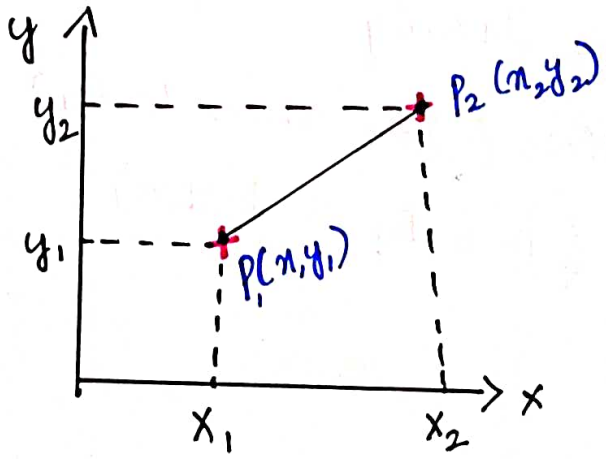
Step:2 Take the two closest data points and Make them one cluster (# that forms N-1 clusters)

Step:3 Take the two closest clusters and make them one cluster (# that forms N-2 clusters)

Step:4 Repeat Step 3, until there is only one cluster

Step 5: Final model.

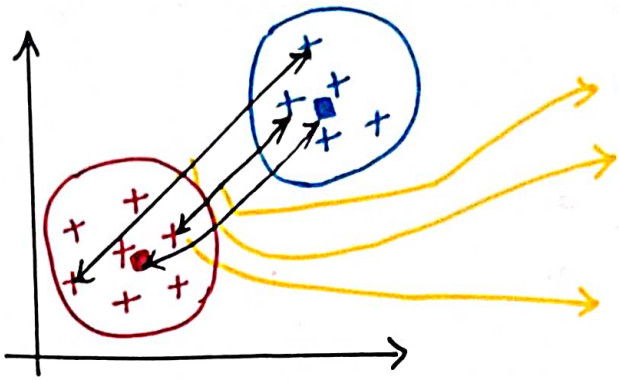
Euclidean Distance :- it is used to identify distance between two points.



*Euclidean distance between P_1 and P_2

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

⇒ Distance between clusters

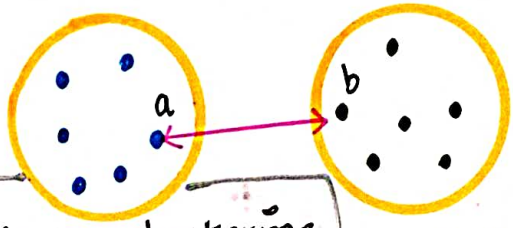


Distance b/w two clusters.

- option 1 : closest Points
- option 2 : furthest Points
- option 3 : Average Distance
- option 4 : Distance between Centroids

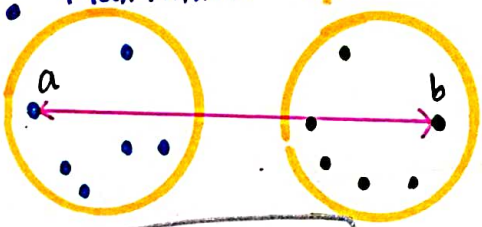
* **Single-linkage clustering**

- Minimum distance between clusters.



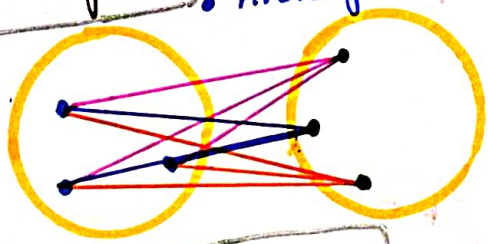
* **Complete-linkage clustering**

- Maximum distance between clusters



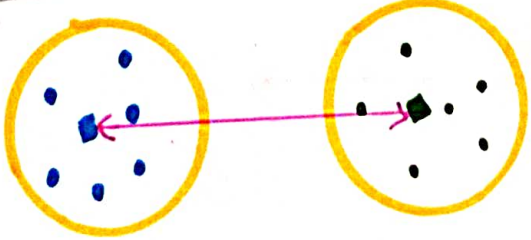
* **Average linkage clustering**

- Average distance between clusters



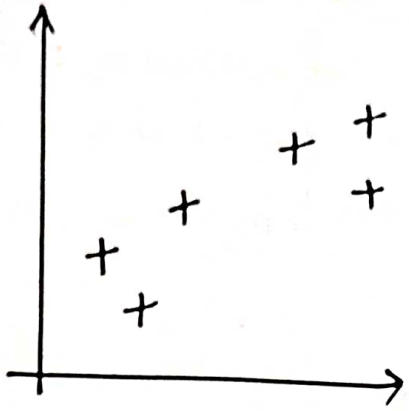
* **Centroid linkage clustering**

- distance between cluster Centroids

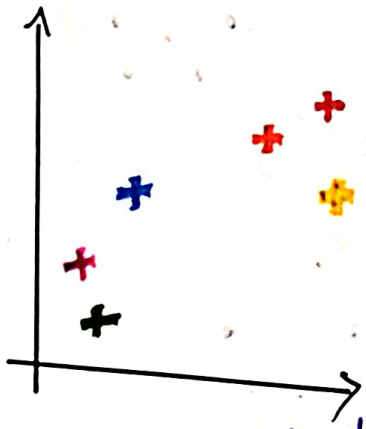


Agglomerative HC

Consider the following dataset of N=6 data points

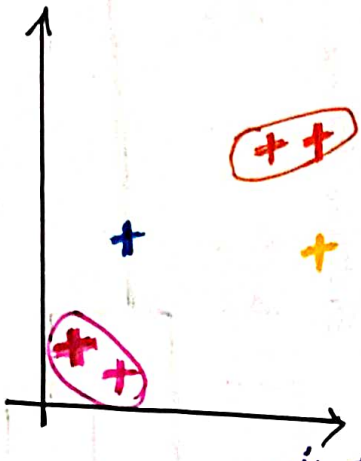


STep : 1 (Make Each data point a single point cluster)
that forms 6 clusters)

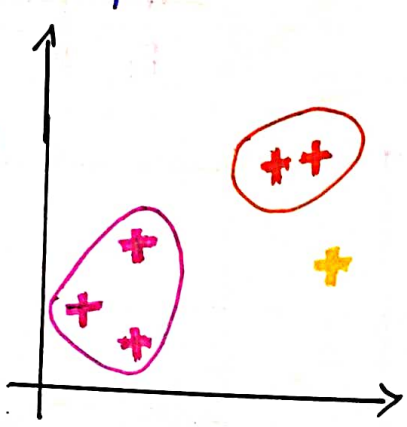


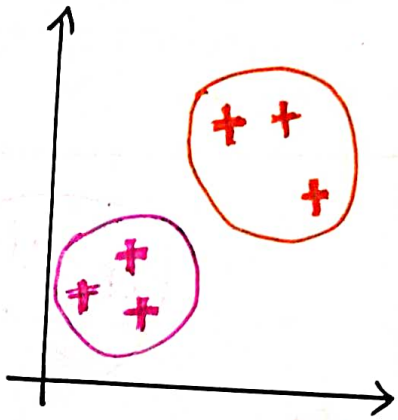
STep : 2 (Take the two closest data points and make them one cluster → that forms 5 clusters)
(n-1)

Step 3 : Take the two closest clusters and make them one cluster → that forms 4 clusters (n-2)

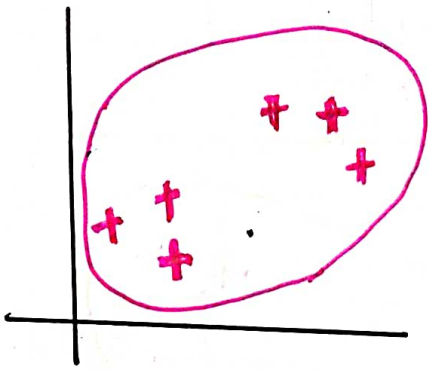


Step 4 : Repeat step 3. until there is only one cluster.

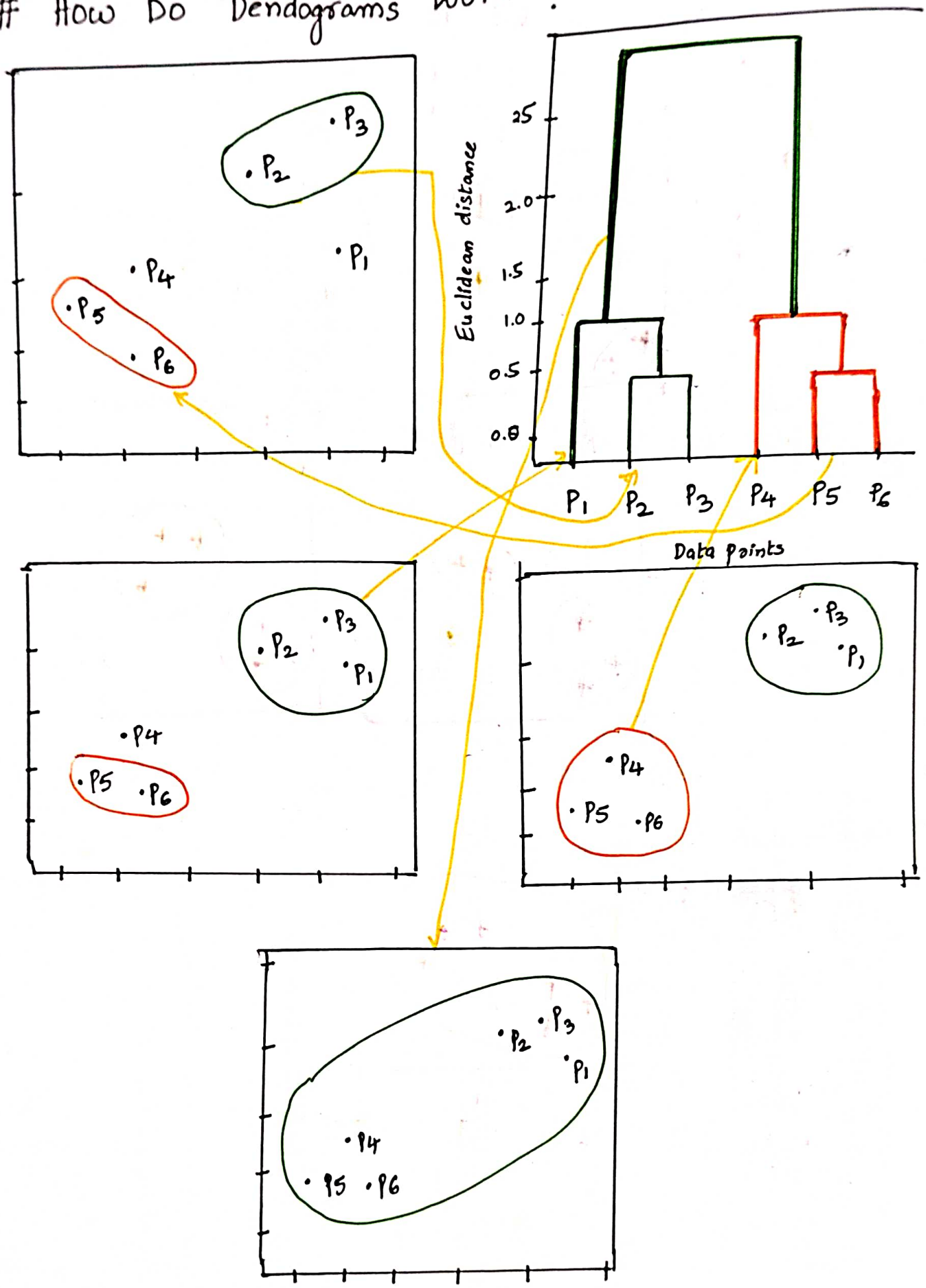




Final :

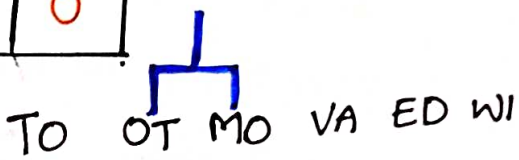


How Do Dendograms Work ?



Example

	TO	OT	VA	MO	WI	ED
TO	0	351	3363	505	1510	2699
OT		0	3543	167	1676	2840
VA			0	3690	1867	819
MO				0	1824	2976
WI					0	1195
ED						0



OT and MO (combined)

Distances are recalculated again.

	TO	OT/MO	VA	WI	ED
TO		351	3363	1510	2699
OT/MO			3543	1676	2840
VA				1867	819
WI					1195
ED					

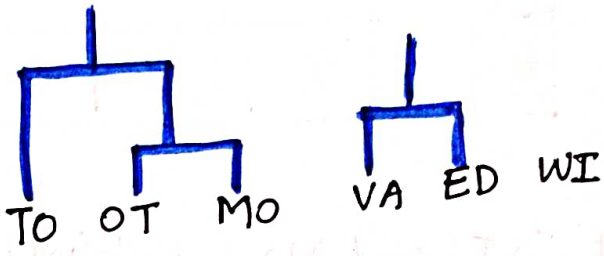
OT/MO and TO (combined)

Distances are recalculated again, to make as one cluster.

	TO/OT/MO	VA	WI	ED
TO/OT/MO		3543	1676	2840
VA			1867	819
WI				1195
ED				

ED and VA (combined)

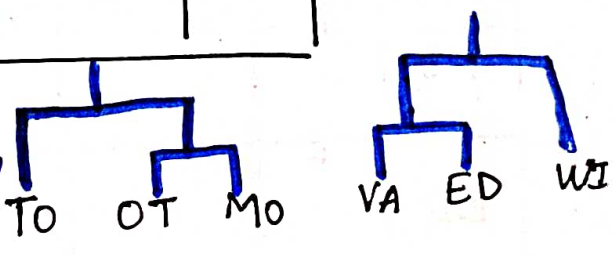
Distances are recalculated again



	TO/OT/MO	VA/ED	WI
TO/OT/MO		2840	1676
VA/ED			1667 # value wrong.

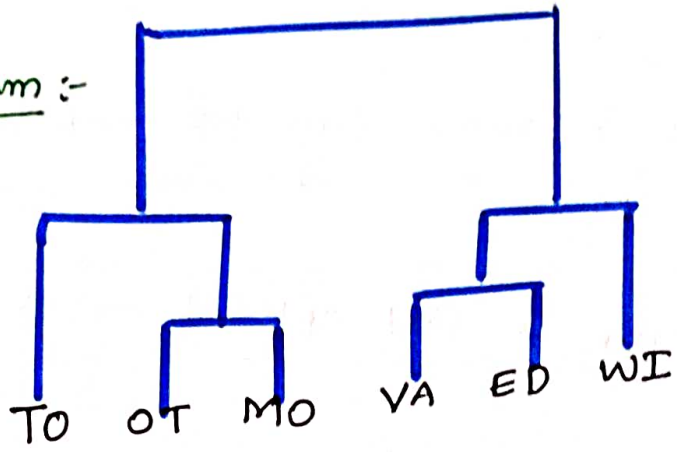
WI and VA/ED (combined)

Distances are recalculated again

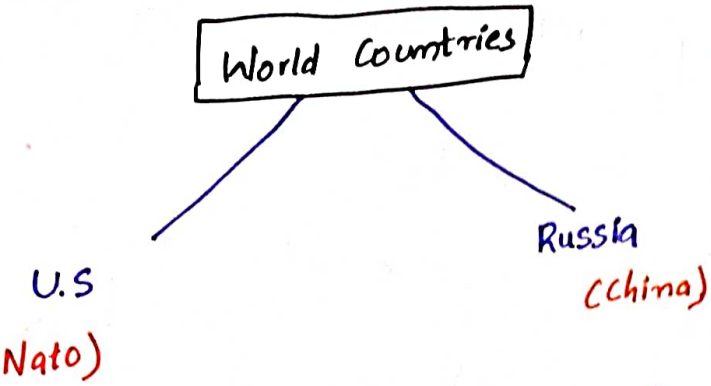


	TO/OT/MO	VA/ED/WI
TO/OT/MO		1676
VA/ED/WI		

Dendogram :-



EX:-



if U.S attack russia. it is going to identify which country is going to support russia.

which country is directly participate in war.

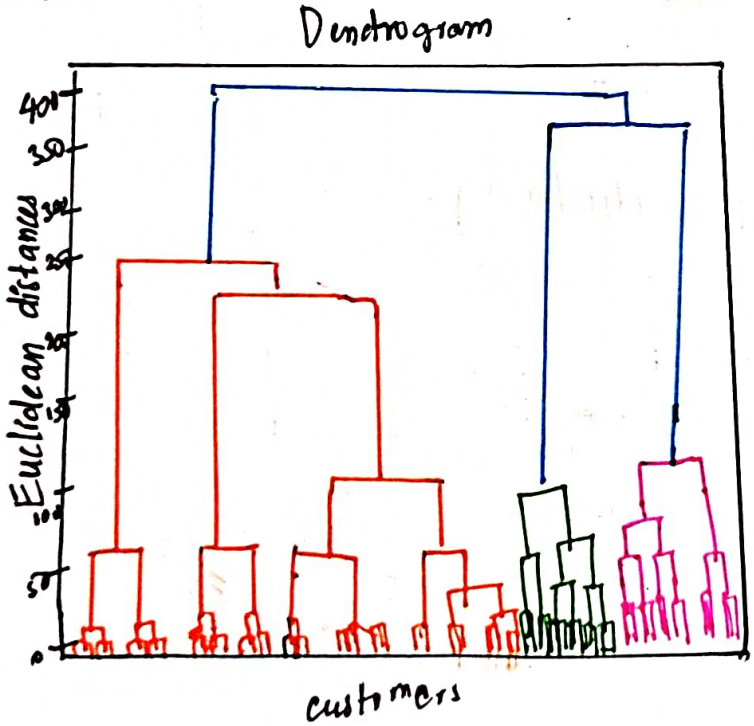
which are very close to Each other. it is going to identify.

this is Hierarchical clustering.

Code :- Same data & Same steps till modelling

Using the dendrogram to find optimal no. of clusters.

```
import scipy.cluster.hierarchy as sch
# dendrogram = sch.dendrogram(sch.linkage(x, method = "ward"))
                                                  ↳ Based on Centroids
# plt.title("Dendrogram")
# plt.xlabel("Customers")
# plt.ylabel("Euclidean distances")
# plt.show()
```



Q:-How to identify optimal clusters during dendrogram?

A:- biggest disadvantage with "HC" is we can't identify "optimal no. of clusters". only in K-means we identify.

Hierarchical clustering model :

```
from sklearn.cluster import AgglomerativeClustering.
```

(small) (capital) (no gap)

```
# hc = AgglomerativeClustering (n_clusters = 5, affinity =
                                "Euclidean", linkage = "ward")
```

Predict

```
# y_hc = hc.fit_predict (x)
```

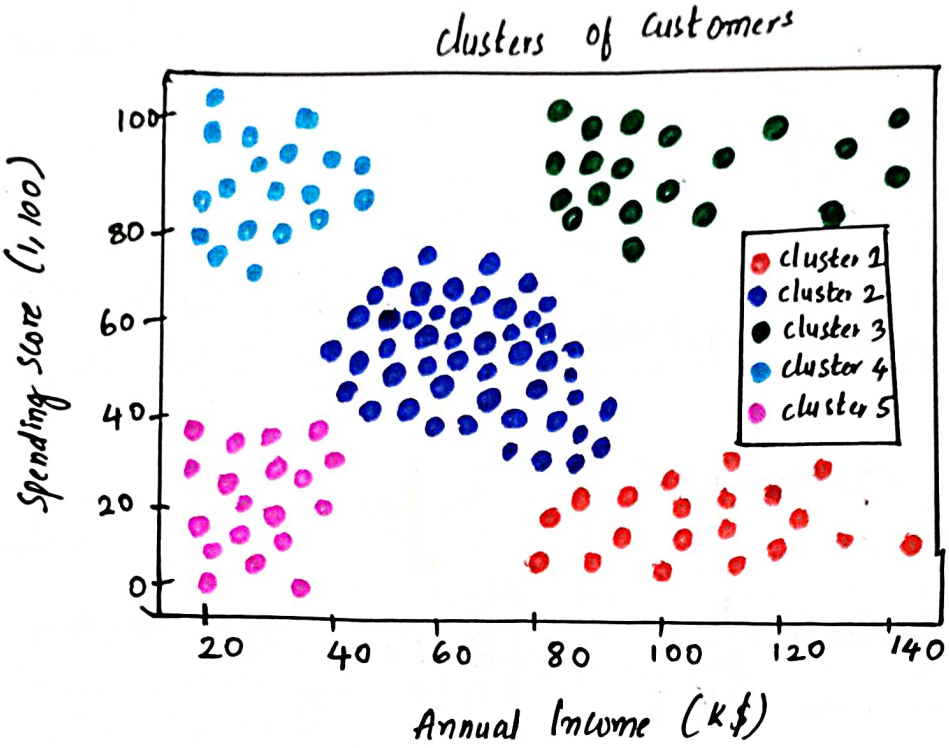
```
# y_hc
```

Out: array ([4, 3, 4, 3, 4, 3, ...
. . . .
. 1,1,1,1,1,1 ---
-- --- - - - - -
0, 2])

Visualising clusters

```
# cluster 1
# plt.scatter(x[y_hc==0,0], x[y_hc==0,1], s=100,
              c="red", label="cluster 1")
# cluster 2
# plt.scatter(x[y_hc==1,0], x[y_hc==1,1], s=100,
              c="blue", label="cluster 2")
# cluster 3
# plt.scatter(x[y_hc==2,0], x[y_hc==2,1], s=100,
              c="green", label="cluster 3")
# cluster 4
# plt.scatter(x[y_hc==3,0], x[y_hc==3,1], s=100,
              c="cyan", label="cluster 4")
# cluster 5
# plt.scatter(x[y_hc==4,0], x[y_hc==4,1], s=100,
              c="magenta", label="cluster 5")

# plt.title("clusters of customers")
# plt.xlabel("Annual Income (k$)")
# plt.ylabel("spending score (1-100)")
# plt.show()
# plt.legend()
```



04/05/22

5:00 Am.

Dt: 3/05/22
3:30 Am

* Un Supervised Machine Learning :

We don't have any output columns, variable /, Feature

↓ (We Group them)

Clustering :-

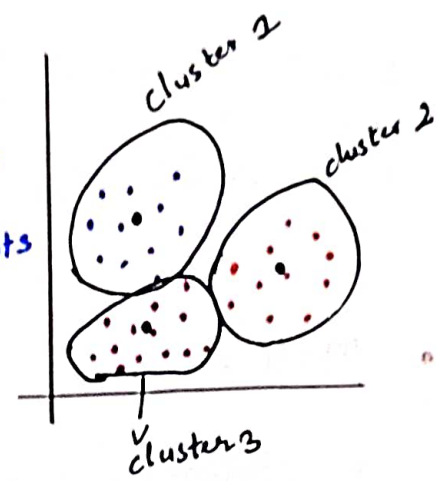
Grouping of Similar objects

1. K-means Clustering
2. Hierachical clustering
3. DB Scan clustering

Un label data

1. What is a cluster ?

A:- A group of objects that are "similar to other objects" (records.) in the cluster, and dissimilar to data points in other clusters.



groups :- segments

2. Clustering applications ?

- Retail / marketing :
  - * Identifying buying patterns of Customers
  - * Recommending new books (or) movies to new Customers.

- Banking :
  - * Fraud detection in credit card use.
  - * Identifying clusters of customers (eg. loyal)

- Insurance :
  - * Fraud detection in claims analysis
  - * Isurance risk of customers.

- Publication :
  - * auto - categorizing news based on their content
  - * Recommending similar news articles

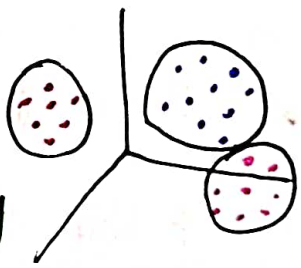
- Medicine :
  - * characterizing patient behaviour

- Biology :
  - * clustering genetic markers to identify family ties.

3. Clustering algorithms

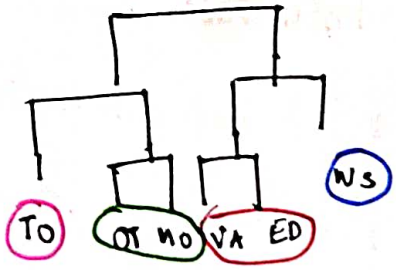
* Partitioned - based clustering
  - Relatively Efficient

Eg:- K-means, K-median, Fuzzy C-means.



* Hierarchical clustering
  - Produces trees of clusters

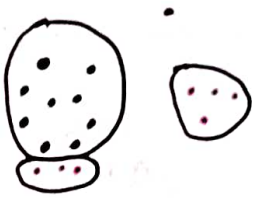
Eg:- Agglomerative, Divisive



* Density - based clustering

- Produces arbitary shaped clusters

eg:- DBSCAN



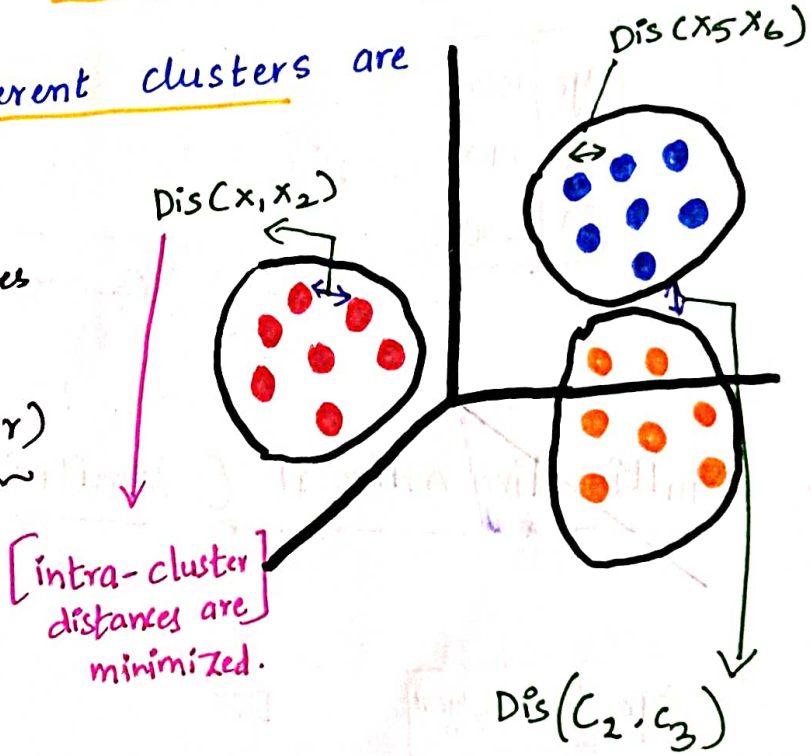
1. K-Means.

* Partitoning clustering
* K-means divides the data into non-Overlapping subsets (clusters) without any cluster internal structure
* records within a cluster are very similar
* records across different clusters are very different

Ex :- covid zones

*Determine the similarity (or) dissimilarity

- intra - within
- inter - outside



⇒Aim:-

* Between two clusters, the distance should be maximum - Better the model
* within cluster, the data point should be closely to each other - Better the model

* One-dimensionality (similarity / distance)

	Age
1	54
2	50

We, calculate Distance.

$$\sqrt{(54-50)^2} = 4.$$

$$Dis(X, X_2) = \sqrt{\sum_{i=0}^{n}(x_{1i}-x_{2i})^2}$$ distance formula.

#3 dimensional - 3 columns

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+z_2-z_1)^2}$$ (Euclidean)

* two-dimensionality (similarity / distance)

Age	Income
54	190
50	200

$$Dis(X, X_2 = \sqrt{(x_1-x_2)^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2}$$

$$= 10.77$$

* Multi-dimentional (similarity / distance)

Age	income	Education.
54	190	3
50	200	8

$$Dis(X, X_2) = \sqrt{(x_{1i}-x_{2i})^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2+(3-8)^2}$$

$$= 11.87$$

Q:- what is the difference b/w "K means" & KNN ?

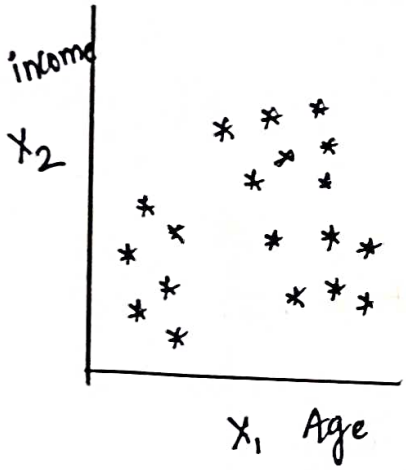
A: Both use Distance Formulas to group them.

but, K-means is clustering, KNN- classification.

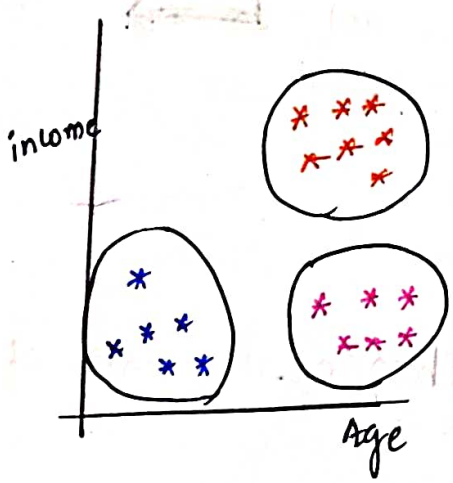
Non supervised (only group)

supervised (o/p) predict

* K-means



(normal Scatterplot)



(# K-means) Clustering.

* Steps to K-means :

STep : 1 (Choose The no. of "K" of clusters)

↓

STep : 2 (Select at random "K" points, The Centroids) (not necessarily from Your dataset)

↓

STep : 3 (assign Each data point to closest Centroid → (that forms "K" clusters)

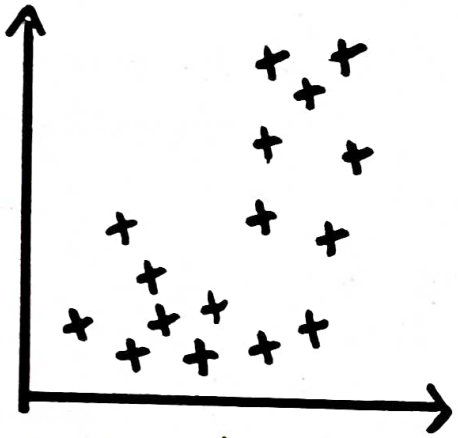
STep: 4 (Compute & place new Centroid of Each cluster)

STep: 5 (Re assign Each data point to the new closet Centroid. if any reassignment took place. go to Step 4. Otherwise go to FIN

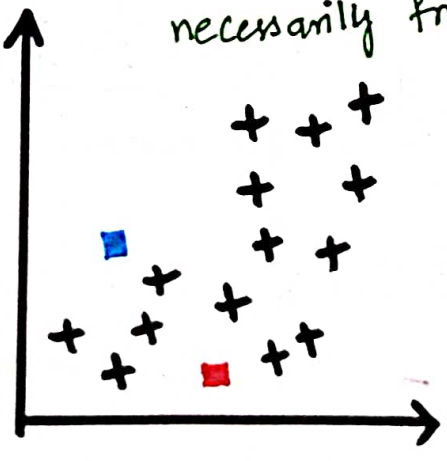
* K-value is not Fixed...

→ Your Model is Ready

⇒ STep :1 :- Choose the no. of K-clusters : 2

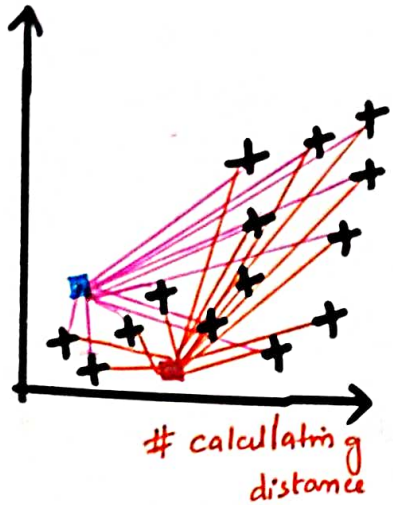


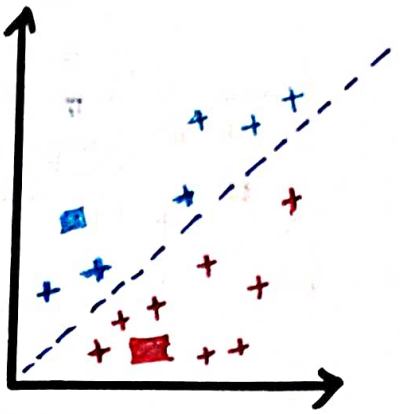
⇒ Step: 2 Select at random K-points, The Centroids (not necessarily from your dataset)



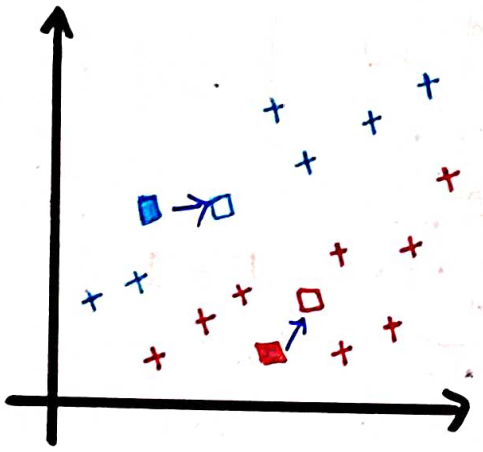
Centroid. Based on #assumptions

STep:-3 (assign Each data point to the closest Centroid that forms K-clusters.)





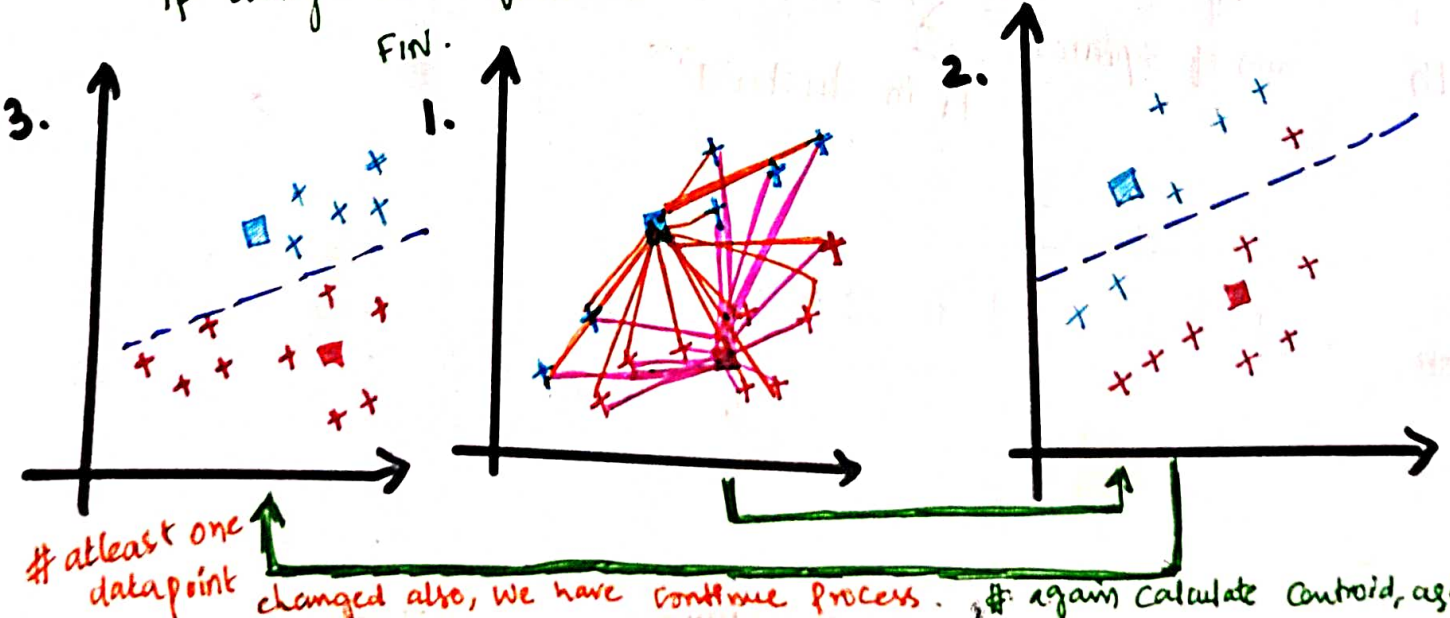
STep:4 Compute and replace the new centroid of Each other.



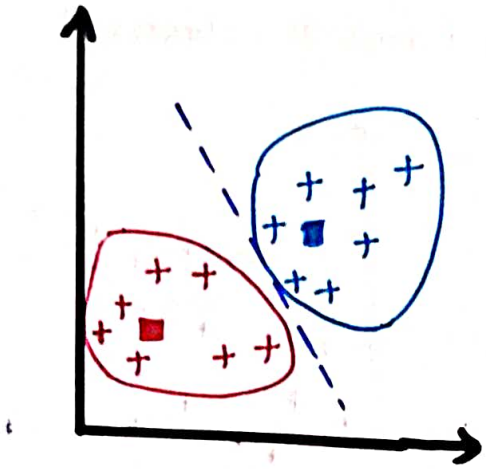
Calculate the original Centroid
Avg = $\bar{x}$ $\bar{y}$
# replace with avg.

STep:5 Reassign each data point to the new closest Centroid.
if any reassignment took place, go to step 4. Orther wise got

FIN.

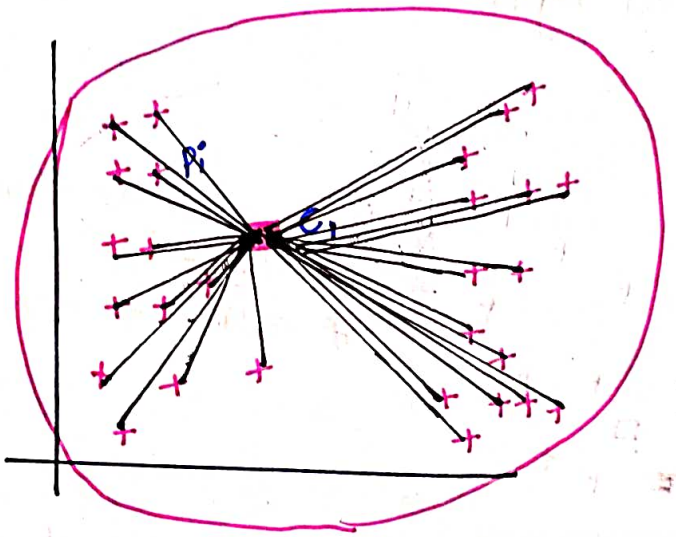


FINAL : Model is Ready.



Que :- How to choose right number of clusters ?

A :-



cluster 1

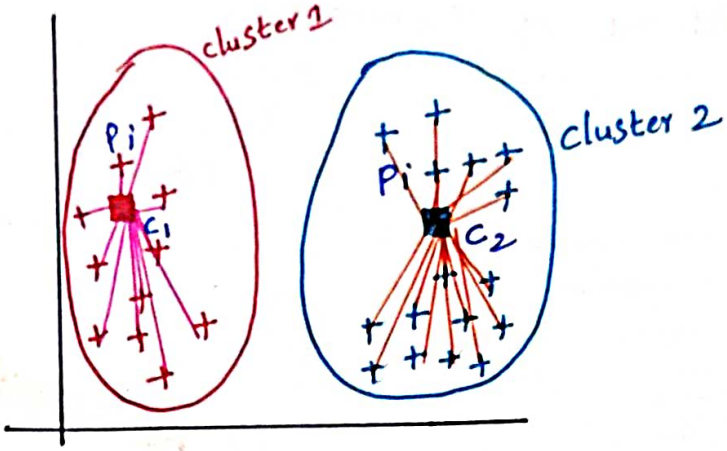
$K = 1$

* Wcss (cluster) ⇒ (with cluster Sum of Squares)

with — Sum of squares $= \sum_{P_i \text{ in cluster } 1} \text{distance}(P_i, C_1)^2$

$$* \text{Wcss} = \text{distance}(P_i, C_1)^2$$

$K = 1$

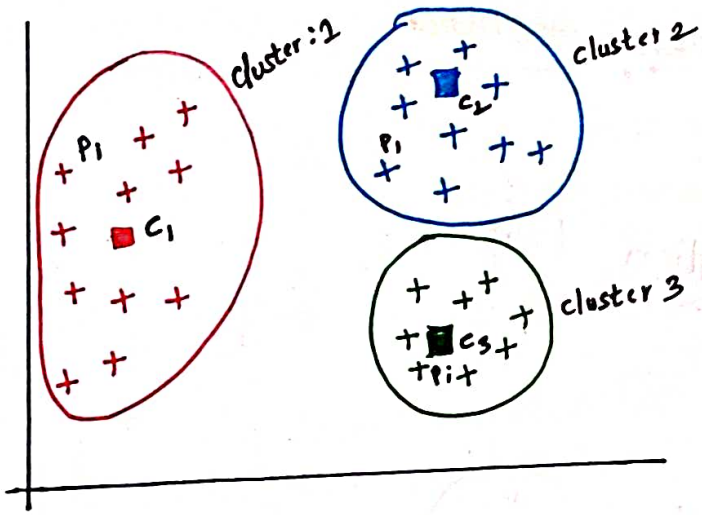


#Calculate SSE

K = 2

* $Wcss = \sum_{P \text{ in cluster } 1} distance(P_i, C_i)^2 + \sum_{P_i \text{ in cluster } 2} distance(P_i, C_2)^2$

(SSE) + ... (SSE)



K = 3

Apply loop.

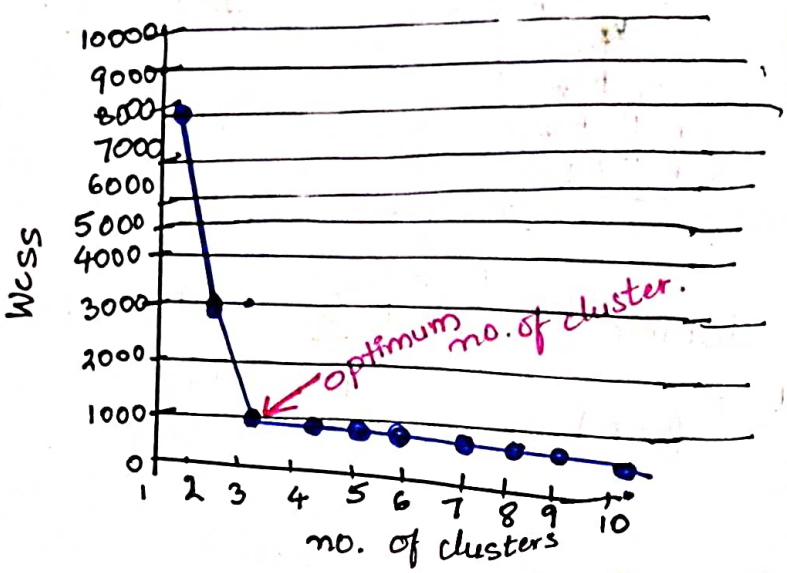
$Wcss = \sum_{P \text{ in cluster } 1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in cluster } 2} distance(P_i, C_2)^2 + \sum_{P \text{ in cluster } 3} distance(P, C_3)^2$

03/05/22

6:00 Am.

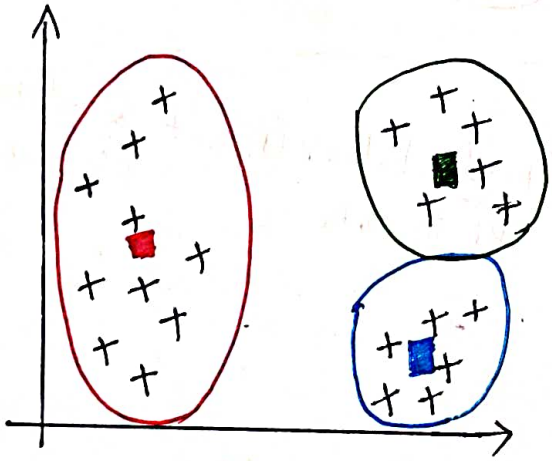
* choosing The right no. of clusters

* Elbow Technique



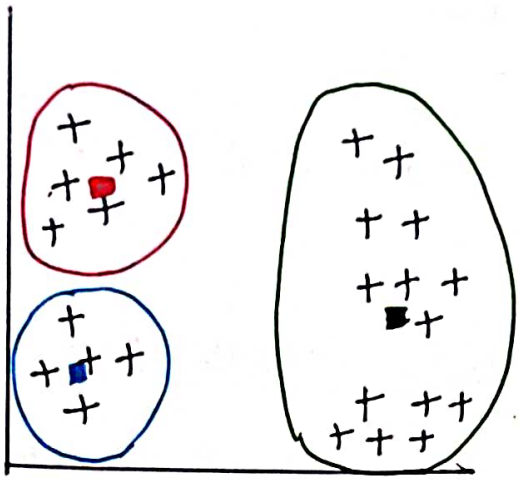
* Elbow method

* Random initialization trap



randomly

Que :- What would happen if we had a bad random initialisation ?



different types of Clusters But, Data set is common.

Based on random initalization. it is going to change clusters

If you choose wrong initalization. Total problem Gets wrong. it is the problem in K-means.

A:- Solution for this is K-means ++

⇒ The Random initialisation Trap

One major drawback of K-means clustering is Random initialisation of Centroids. The formation of clusters is closely bound by the initial position of a centroid. The random positioning of Centroids can Completely alter clusters and can result in random formation.

* The Solution is K-means ++
* K-means ++ is an algorithm that is used to initialise The K-means algorithm. *** instead of "arthematic mean" it Take "Weighted mean". For Calculating Centroid.

. K-means ++

1. Choose one Centroid uniformly at random from among the data points.
2. For each data point say x, Compute $D(x)$, which is distance between x and nearest Centroid that has already been choosen.
3. Choose one new data point at random as a new Centroid, using weighted probability distribution where a point "x" is chosen with probability proportional to $D(x)^2$.
4. Repeat steps 2 and 3 until K Centres have been choosen.
5. Proceed with Standard K-means clustering.

Ex:-

Age	Edu	Exp
21	B.Tech	0
24	B.Tech	2
22	B.Tech	0
28	MTech	3

group

→ grouping same Element (or) similar

Multiple feature over all

(calculate distance formula)

Encoding

CODE :-

```
# df = pd.read_csv("Mall_Customers.csv")
# df.head()
```

Out :-

Customer ID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	21	17	40


```
# df.shape
```

Out: (200, 5)

```
# df.info()
```

Out:

200 entries, 0 to 199

Column	Non null Count	Dtype
CustomerID	200 non null	int64
Genre	200 non null	object
Age	"	int64
Annual Income	"	"
Spending Score	"	"

```
# df.isnull().sum()
```

Out 0.

X # to Understanding, we take only two columns.

```
# x = df.iloc[:,[3,4]].values
```

MODEL

```
from sklearn.cluster import KMeans.

# using the elbow method to find the optimal no. of clusters.

# wcss = []

for k in range (1, 21):

    kmeans = KMeans (n_clusters = k, init = "k-mean++")

    k means.fit (x)

    wcss.append (kmeans.inertia_)
```

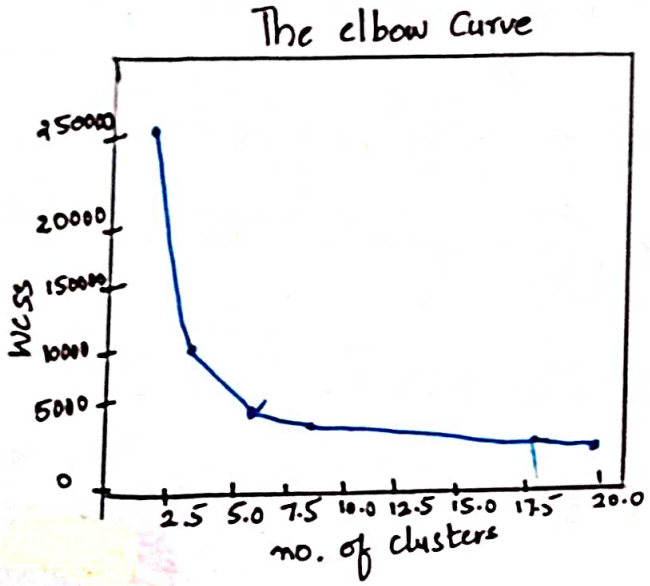
- #WCSS values is called inertia

```
# WCSS

out :: [269981.2800 ... # k=1
        1811363.5959 , # k=2
        106348.373062,
        73679.789039
        .
        .
        .
        8504.24137246]
```

Elbow curve

```
# plt.plot(range(1,21), wcss)
# plt.title('The elbow method")
# plt.xlabel("no. of clusters")
# plt.ylabel("wcss")
# plt.show()
```



fit The K-means Model on the data.

```
# Kmeans = KMeans(n_clusters = 5, init = "K-means++",
                  random_state = 42)
```

Predict

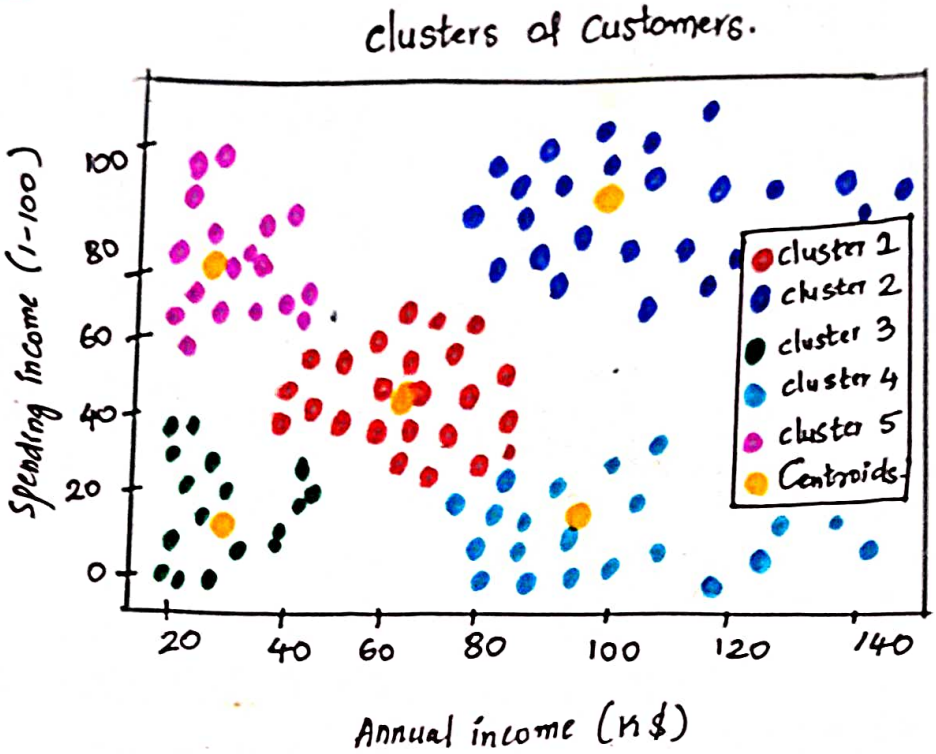
```
# y_Kmeans = Kmeans.fit_Predict(x)
```

Visualising The clusters

```
# cluster 1
# plt.scatter(x[y_kmeans == 0,0], x[y_kmeans == 0,1],
                s=100, c="red", label="cluster 1")
# cluster 2
# plt.scatter(x[y_kmeans == 1,0], x[y_kmeans == 1,1],
                s=100, c="blue", label="cluster 2")
# cluster 3
# plt.scatter(x[y_kmeans == 2,0], x[y_kmeans == 2,1],
                s=100, c="green", label="cluster 3")
# cluster 4
# plt.scatter(x[y_kmeans == 3,0], x[y_kmeans == 3,1],
                s=100, c="cyan", label="cluster 4")
# cluster 5
# plt.scatter(x[y_kmeans == 4,0], x[y_kmeans == 4,1],
                s=100, c="magenta", label="cluster 5")
# Centroid
# plt.scatter(kmeans.cluster_centers_[:,0],
                kmeans.cluster_centers_[:,1],
                s=100, c="yellow", label="centroids")

# plt.title("clusters of customers")
# plt.xlabel("Annual Income ('k $)")
# plt.ylabel("spending score (1-100)")
# plt.legend()
# plt.show()
```

Out :-



#orginal data

```
# a = df.iloc[:,3]
# b = df.iloc[:,4]
# plt.scatter(a,b)
```


Out :