Artificial Intelligence

① → Robotics ~ Mech | Electrical

→ Machine learning { Supervised / Unsupervised

→ Deep learning

Reinforcement learning

Real Time Processing → Kafka

Alexa

IOT { Smart work / Smart watches / Smart cards

Computer Vision

→ face detection
↳ image
↳ pixel

Machine learning

Supervised { Regression / classification (0, 1) }

Unsupervised

X

X → Y → target var.

② Machine Learning

x^2

x → [f(x) Model] → y

Predictor Independent

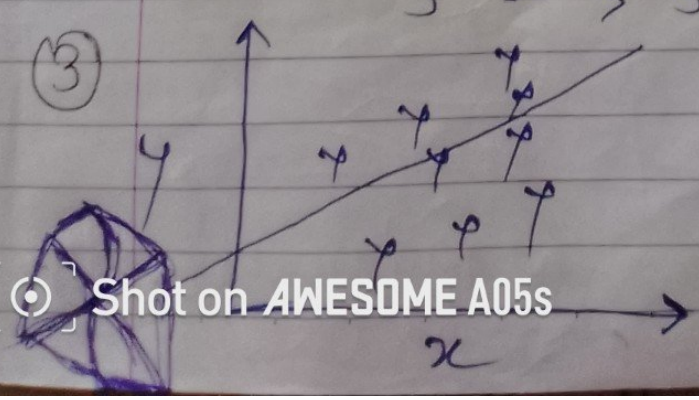
Target Dependent

1 → x^2 → 1

2 → 2^2 → 4

5 → 5^2 → 25

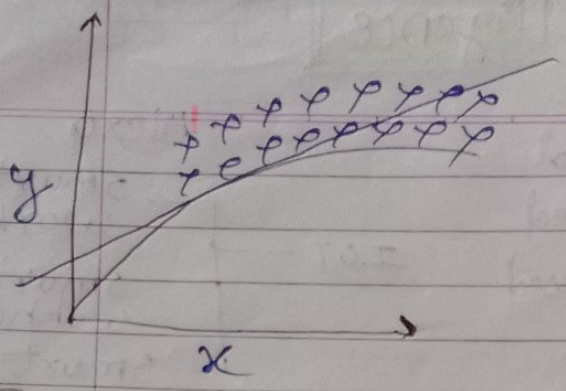
③

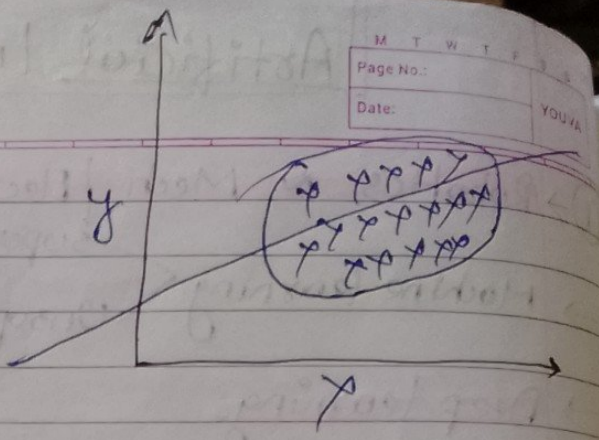


Linear Model

x → [f(x)] → y

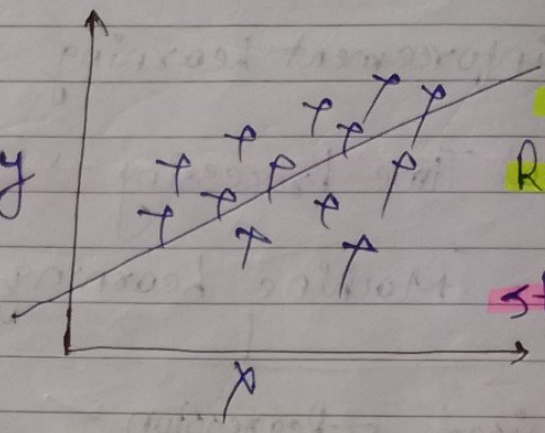
Black Box





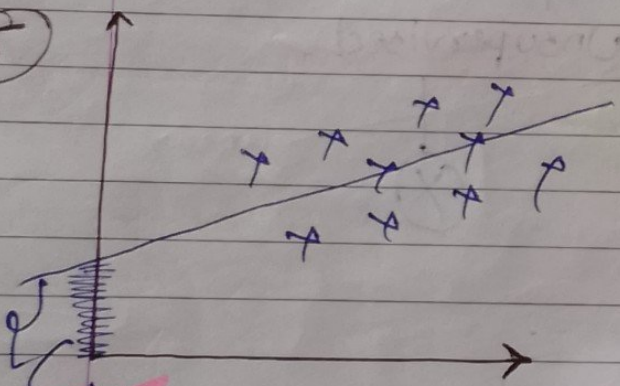
(4)

X	Y



Linear Regression → st. line

(5)

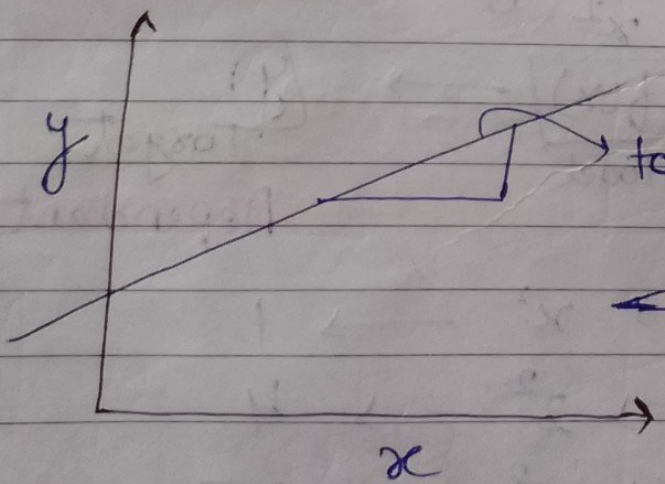


linear Regression → straight line

$y = mx + c$ St. line eq^n

c = intercept

m = slope



$\tan\theta = P/b$

rise

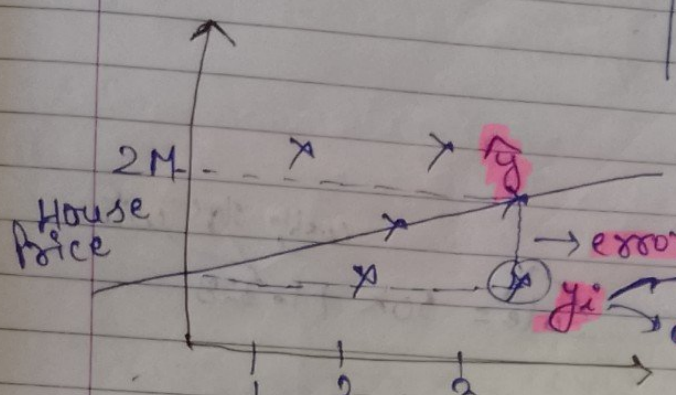
run

⑥ Deterministic Model → $y = mx + c$

Machine Learning

$\hat{y} = \beta_0 + \beta_1 x$

$\hat{y}$ = predicted value of y
β_0 = intercept
β_1 = slope



$y = \beta_0 + \beta_1 x + e$ → actual value

$y = y_i$ → actual value

$error = y_i - \hat{y}$ *

$y = mx + c$ → Coordinate Geometry equations.

$\hat{y} = \beta_0 + \beta_1 x$ → machine Learning equations.

⑦

You tube spent Lakhs (x_1)	TV add spent lakhs (x_2)	Sales in Crores (y)
35	15	142
40	20	160
45	21	150

Sales = 110 + 0.98 yt + 0.32 TV
(0.98 yt → $\beta_1 x_1$; 0.32 TV → $\beta_2 x_2$)

Sales → pred (y)

110 → intercept

Multiple Linear Regression = $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$

Sales = 110 + 0.98 YT + 0.32 TV

110 Cr

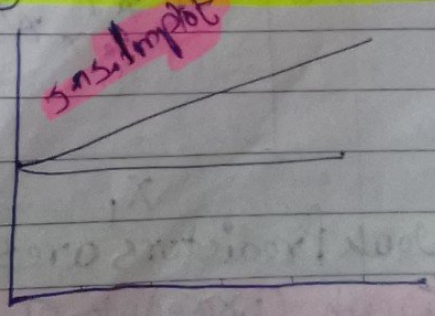
$yT = 0.98 \times 100000 = 980000$

TV = 0.32 → coeff

120 Cr

$\hat{y} = \beta_0 = \bar{y}$

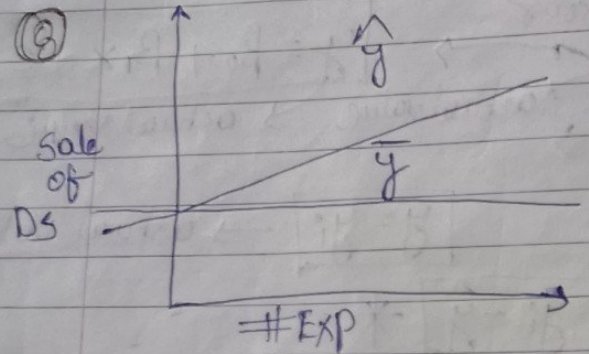
NULL MODEL



~~Distance Matrix~~

If my slope of the equation becomes zero then my average sale in this problem B_0 (170)

(8)



coeff. slope wt

Sale = 50k + 1.328

#Exp	#Sale
2	50k
6	100k
	90k

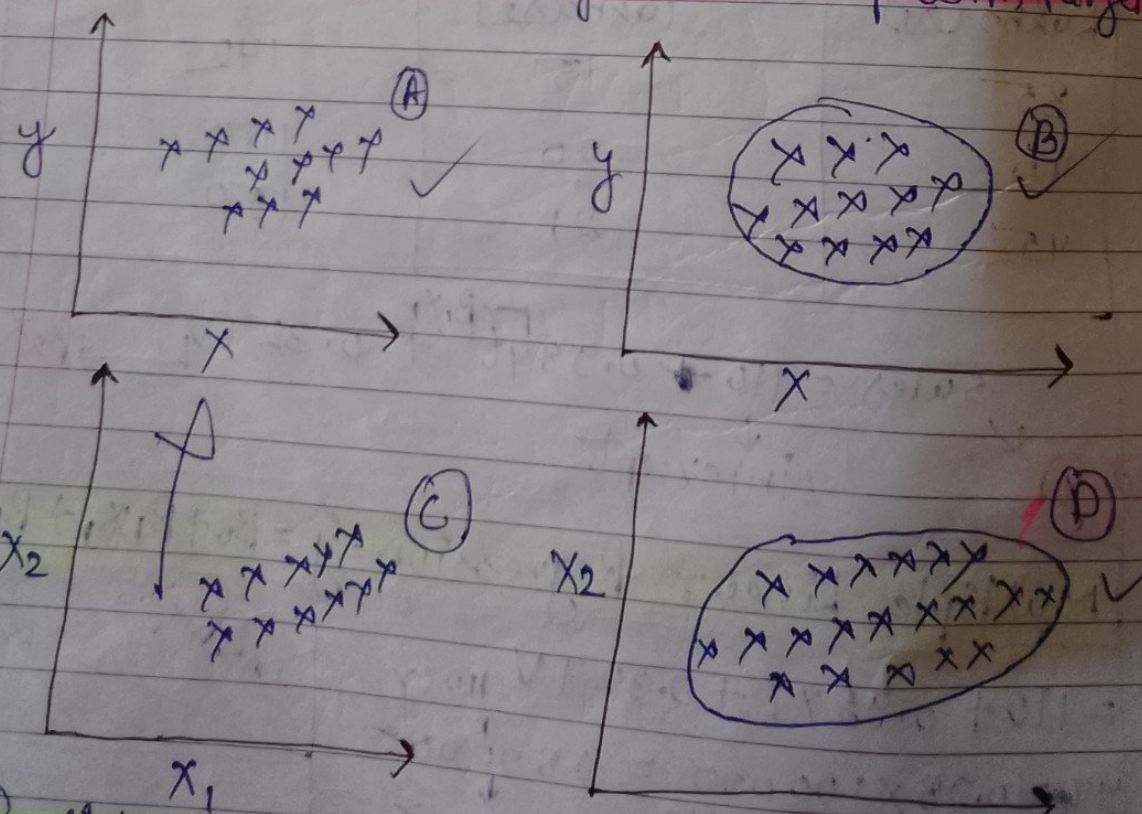
Sale ⇒ ȳ (intercept)

(*) Base Model is always Model → is always worst model.

(9)

→ Two predictors should not have a strong relation.

→ Predictors should have strong relationship with target var.



(*) Weak Predictors are strongly related ⇒ Multicolinearity

B/w predictors is ok

(10)

$\hat{y} = f(x$

(11)

$B_1 =$

(12)

(10) $\hat{y} = f(\text{youtube}, \text{TV add})$

~~$\hat{y} = f(\text{youtube}, \text{TV add})$~~

youtube TV add = 85%.

Youtube V/s Sales ~ 91%.

TV add V/s Sales ~ 89%.

(11) $$\text{Covariance} = \frac{\sum (x_i - \bar{x}) \cdot (y_i - \bar{y})}{n-1}$$

$$\text{Correlation} = \frac{\text{cov}(x,y)}{\sigma_x \cdot \sigma_y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2 \cdot (y_i - \bar{y})^2}$$

Slope (B_1)

$$B_1 = \frac{\text{cov}(x,y)}{\text{Var}(x)} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

$$\text{Intercept} = \bar{y} - B \cdot \bar{x}$$

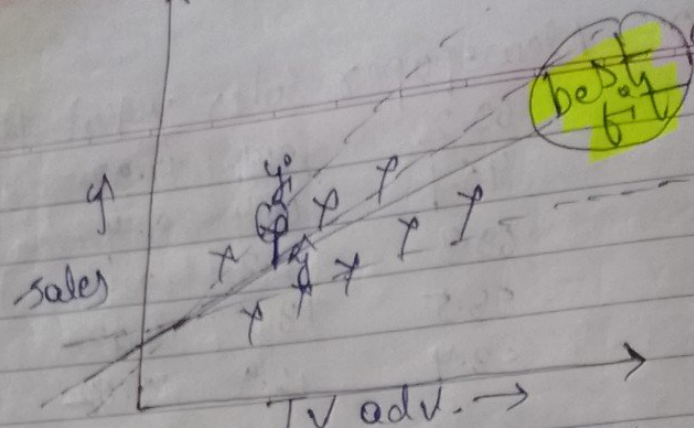
```
advertising = pd.read_csv("~/Downloads/Advertising.csv")
advertising.head()
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

~~#-#-#-#~~ About Linear Regression

(1) If I use only only *1 Predictor ** and Target to build the Linear Regression Model, then such a model is known as Simple Linear Regression

(13)



Gradient descent

PCA - Perpendicular Distance

OLS - error is least, Vertical Distance.

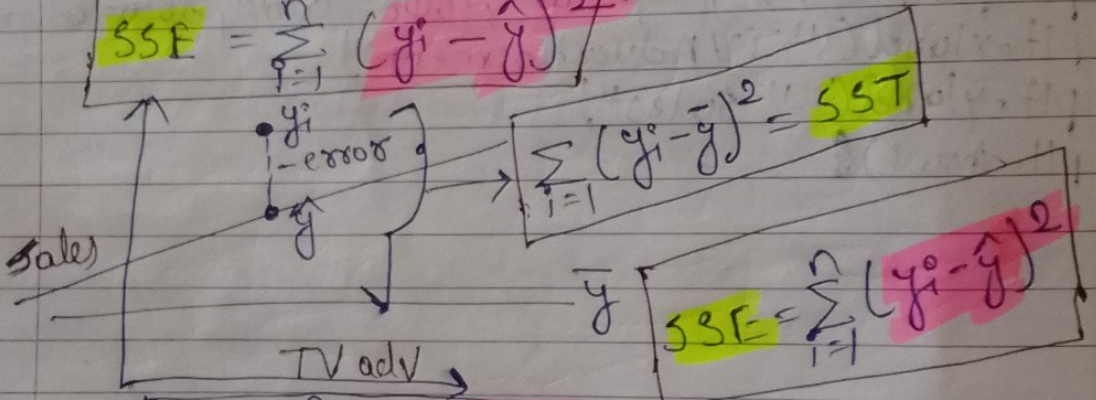
Ordinary least square Method

(14)

SSE → Error, SSR → Regression, SST → Total

SSE, SSR, SST

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2$$



$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = SST$$

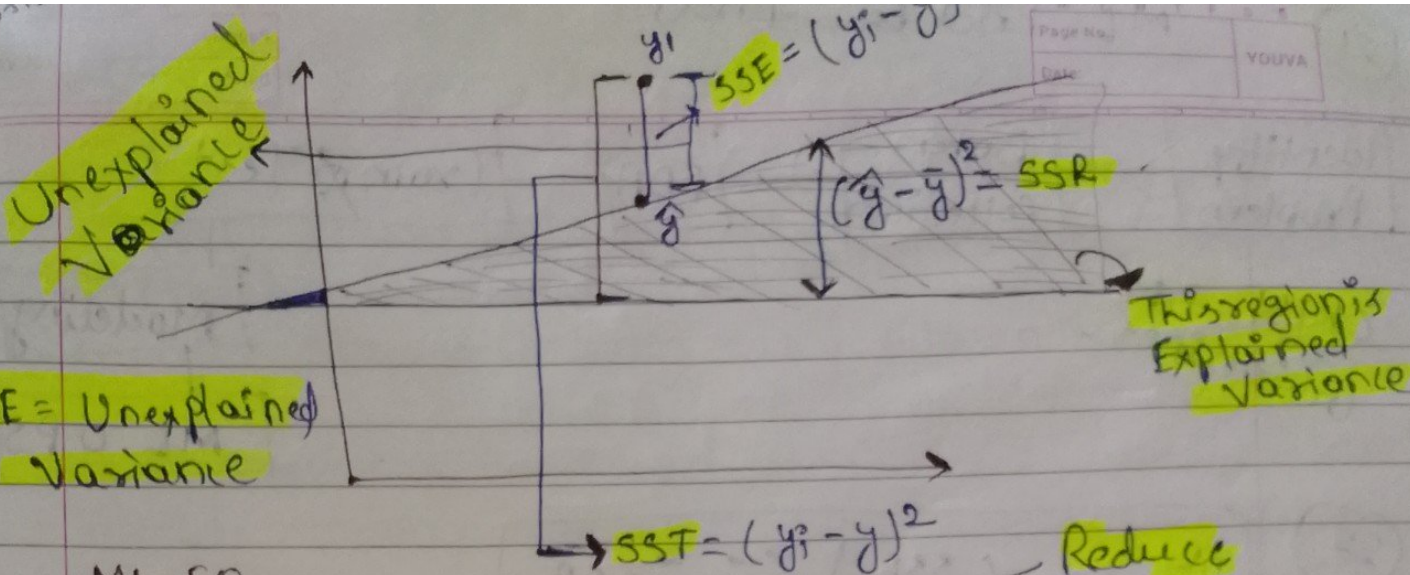
$$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

$$SSR = \sum_{i=1}^{n} (\hat{y} - \bar{y})^2$$

Note: SSR is sum of SQ Regression

$$R^2 = 1 - \frac{SSE}{SST} \quad OR \quad R^2 = SSR/SST$$

OLS - Fit the line in such a way that the errors are the least

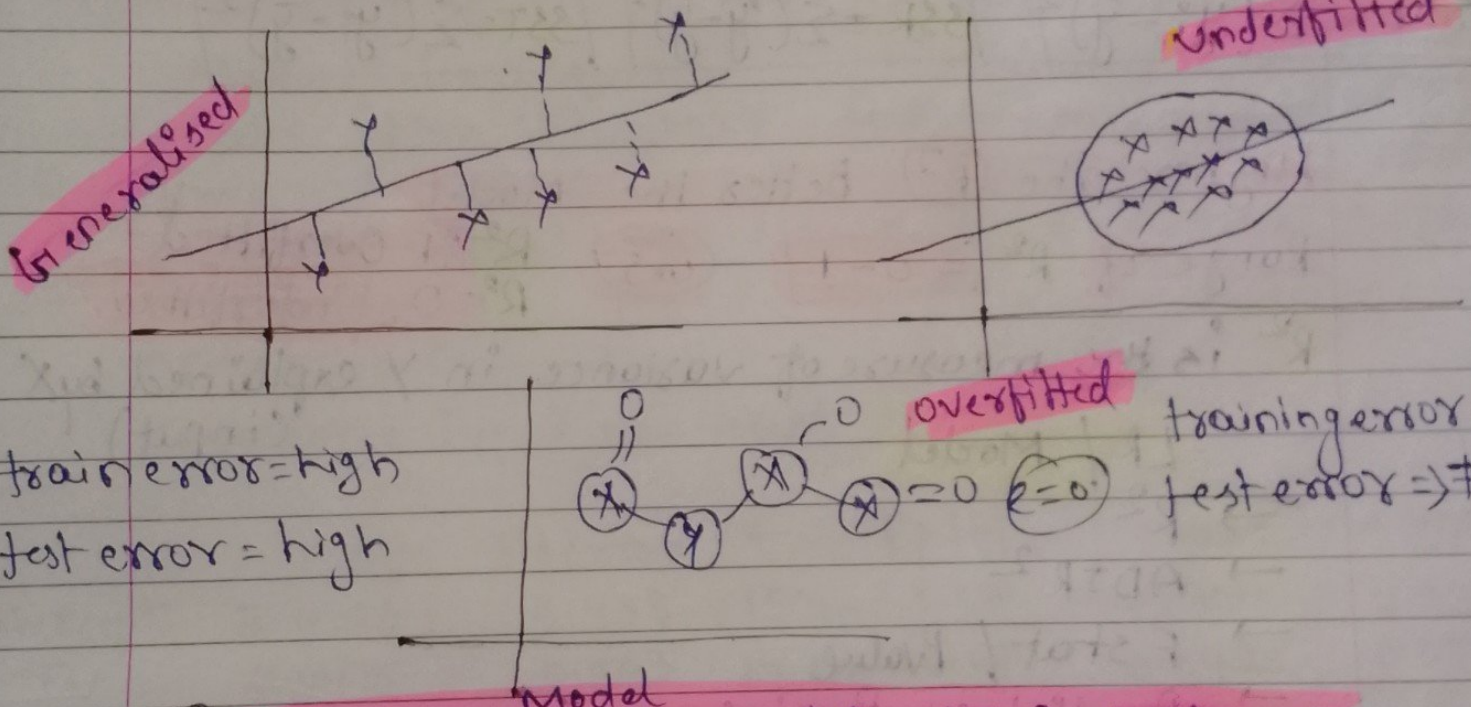


ML EQ

$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + e$

Reduce

Irreduce



(*) Tree based a lot used in IT industry Model

(1) Train error = 0
Test error = 12
M_1
Overfitting

(2) Train error = 15
Test error = 15.83
M_2
Underfitting

(3) Train error = 8
Test error = 7.65
M_3
Generalised

(*) Test error जहां कम होगा वही Best Model होगा

(21) ML End to End

Identify Problem → Data collection → EDA → Train & Test → Modeling → ML OPS

CRISP-DM Guidelines.

(22) $R^2 = 1 - \left(\frac{SSE}{SST}\right)$ or $\frac{SSR}{SST}$

$SSE = \Sigma(y_i - \hat{y})^2$ $SSR = \Sigma(\hat{y} - \bar{y})^2$ $SST = \Sigma(y_i - \bar{y})^2$

Higher the R^2 better the Model

Range of R^2 = 0-1 (0.5) $R^2 = 1$ Overfitted

$R^2 = 0$ Underfitted

R^2 is the measure of variance in Y explained by X (input)

LR / Model

- → R^2
- → ADJ R^2
- → F stat / Pvalue
- → Predictor Pvalue
- → Durbin watson
- → condition No.
- → JARQUE BERA

} — Assumptions.

Note: while adding new features / new column in the data to check ~~as~~ k liye ki column Hamare model k liye sahi hai ya nahi, and Toh R^2 hamesha increase hoga new columns / new features add krne pr but agar adj R^2 increase hoga tbhi ye prove hoga ki wo new column / new features hamara model acha hai

① R^2 = Higher the R^2 better the model ✓

② Adj R^2 = Adj R^2 ↑, R^2 ↑ Good model ✓ Predict power ↑

= Adj R^2 ↓, R^2 ↑ Bad model ✓

③ Predictor Pvalue < Alpha (0.5) Significant Predictor

Predictor Pvalue > Alpha (0.5) Not Significant Predictor

④ Fstat Pvalue < alpha (0.5) Significant Model

Fstat Pvalue > Alpha (0.5) Not Significant Model

If all conditions Met होगी तभी Then we will called Our Model is good Model

R Squared

residual ↑

```
#SSE
SSE = np.sum(advertising[residual]**2) #yi-yhat

# SST
SST = np.sum((advertising["Sales"]-advertising["Sales"].mean())
                **2)

#SSR = SST-SSE
rsquared = 1-(SSE/SST)
print("R Squared:",rsquared)
```

R Squared: 0.611875050850771

Adjusted R Sq $= 1-(1-R_2)*(n-1)/(n-k-1)$

```
n = advertising.shape[0] # No. of Rows in the Data

k=1 #Since it is Simple Linear Regression

num = 1-((1-rsquared)*(n-1)
deno = n-k-1
adj-r2 = 1-(num/deno)
print("Adjusted R2:",0.6099148238341623
                ,adj-r2)
```

F Stat & P Value

24

$$F\ stat = \frac{MSR}{MSE}$$

k=1 → dfn

$df = n-k-1$ → dfd

$$\frac{SSR}{k} = MSR$$

$$MSE = \frac{SSE}{df}$$

$$F\ Test = \frac{MSB}{MSW}$$

$df = n-k-1$

G1	G2	G3
1	2	5
2	3	6
3	4	7

n-1 + n-1 + n-1

3-1 + 3-1 + 3-1

6

RIGHT ANSWER

$SSR = SST - SSE$

$MSR = SSR / k$

$MSE = SSE / (n-k-1)$

fstats = MSR / MSE

print("F stats:", fstats)

fstats: 312.1449943727 13

P Value

import scipy.stats as stats

1 - stats.f.cdf(fstats, k, n-k-1)

1.1102230246551565e-16

Ho: The Regression model is not a significant model

Ha: The Regression model is a significant model.

Day 2 session 3

T Test and PValue

- Ho : There is No Relationship b/w the X and Y
- Ha : There is a Relationship b/w the X and Y

** tstats = coefficient / standard_error

tstats = 0.0475 / 0.003
15.833333333334

model.params 7.032594
Intercept 0.047537
TV
dtype: float64

model.params represent the slope and intercept

model.params[1]

Standard Error
model.bse
Intercept 0.457843
TV 0.002691
dtype: float64

model.bse = standard error

print("T TestStats:", model.params[1]/model.bse[1])
T TestStats: 17.667625600875555

# PValue * 2
df = n-k-1
pvalue = (1 - stats.t.cdf(tstats, df)) * 2
print("PValue:", pvalue)
T TestStats: 17.667625600875555
PValue : 0.0

Model.params = coefficient
Model.bSE = S.E.

Standard Error

$$SE = \sqrt{\frac{\sum (y_i - \hat{y})^2}{n-2}} \rightarrow SSE$$

$$\frac{}{\sqrt{\sum (x_i - \bar{x})^2}}$$

SSE .·. num = $\sqrt{\frac{SSE}{n-2}}$

Standard Error calculation

STANDARD ERROR

```
num = np.sqrt(SSE/(n-2))

xi = advertising["TV"]
xbar = advertising["TV"].mean()

deno = np.sqrt(np.sum((xi-xbar)**2))

se = num/deno
print("Standard Error:", se)
print("SE from Model:", model.bse[1])
```

Standard Error : 0.0026906071877968703
SE from Model : 0.0026906071877968703.

Linear Regression
Degree of freedom = n-k-1

Confidence Interval

$CI = \bar{x} \pm Z/t \times \frac{s/\sigma}{\sqrt{n}}$ → SE

$\bar{x}$ ↓ slope

Confidence Interval of slope

```
stats.t.interval(0.95, loc=model.params[1], scale=
                 model.bse[1], df = df)
(0.042230716032698 82, 0.05284256483334068)
```

Confidence Interval of Intercept

```
stats.t.interval(0.95, loc=model.params[0],
scale=model.bse[0], df=df)
(6.1297192688055 35, 7.093546782944 9835)
```

Revision All Formulas Till the Date

① $\text{slope} = \frac{Cov(x,y)}{Var(x)} = \frac{\Sigma (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\Sigma (x_i - \bar{x})^2}$

② $\text{Intercept} = \bar{y} - \beta_1 \cdot \bar{x}$

③ $R\text{-Squared} = 1 - \left(\frac{SSE}{SST}\right) OR \left(\frac{SSR}{SST}\right)$

$SSE = \Sigma (y_i - \hat{y})^2$

$SST = \Sigma (y_i - \bar{y})$

$SSR = \Sigma (\hat{y} - \bar{y})$

④ $\text{Adjusted } R^2 = \frac{1 - (1 - R^2) \times (n-1)}{(n-k-1)}$ Note: $df_{Reg.} = n-k-1$

⑤ $F\text{test} = \frac{MSR}{MSE}$ $MSR = \frac{SSR}{K}$

$MSE = SSE/df$

⑥ $T\text{ test statistic} = \frac{\text{Coefficient}}{SE}$

Linear Regression Assumptions

(1) Linearity → Rainbow Test

For normality
skew = 0
kurt = 3

(2) Normality: Residuals should be normal
→ JARQUE BERA Test of normality

(3) Multicollinearity: Predictors should not be strongly correlated.
→ ✱ Correlation (Heat map)
→ ✱ Condition No (100 – 1000) ← MILD COLLINEARITY
(range)
→ ✱ VIF

(4) HETERO SKEDASTICITY
Breusch Pagan Test.

(5) Auto Correlation of Errors DW – Test

Linear Regression Assumptions

* **Linearity** : Stastical test for testing Linearity is **Rainbow Test**

* **Normality** : Statistical test for testing Normality is **Jarque Bera Test**

* **Multicollinearity**: Statistical test for testing Multicollinearity is Variance inflation factor. This is in addition to **Correlation Plot and Condition Number**

* **Hetroskedasticity**: Unequal Variance, so the assumptions is that there should be equal variance and if that is not found then the Regression is not a good model. The statistical test is **Breusch Pagan Test**. There is another test to check the Hetroskedasticity (Equal variance) known as **GoldFend Quandt Test**

* **AutoCorrelation of errors**: The errors should be correlated. The test to verify is **Durbin Waston Test**

(3) assumptions check krne pdte one by one check

Linearity CHECK. (1)

Checking Our Model is linear or not

Ho: Data/Model is Linear
H1: Ho is False

```
import statsmodels.api as sma

teststats, pvalue = sma.stats.linear_rainbow(model
print("Pvalue:", pvalue)
```

Pvalue: 0.7185004164833 91

Conclusion Since the Pvalue > 0.05 we Fail to Reject
Meaning that the Linearity is Present

Normality check (2)

Normality check Homesha Residuals check kiyo jata hai

Ho: Residuals are normal
Ha: Residuals are not normal

```
stats.jarque_bera(model.resid)
```

Conclusion: Reject the Ho. It means that the Residuals are not Normal

nificanceResult(statistic=151.2414204, pvalue=1.439934-33)
0.0

Pvalue = 0.0
∴ Residuals are not normal

Residual

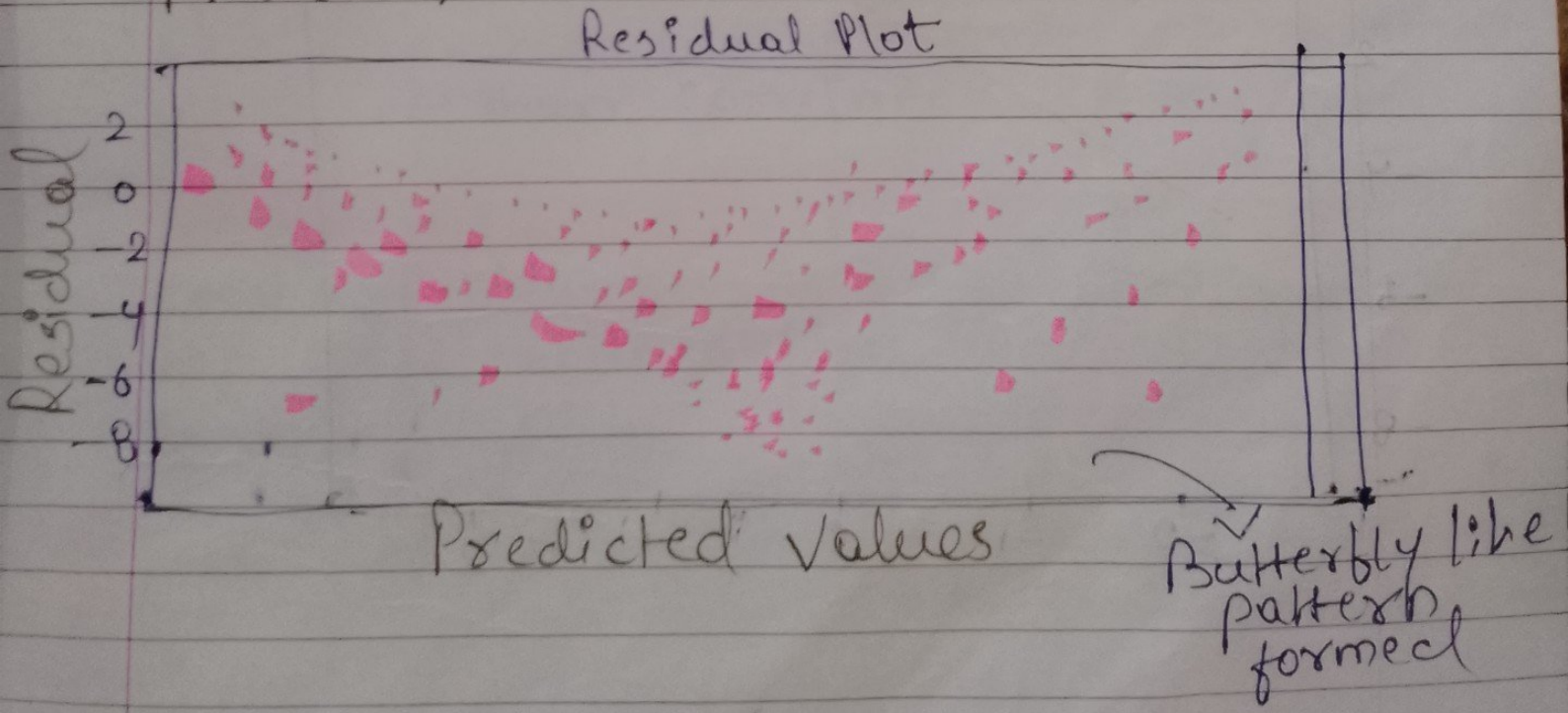
(*) AR
Fit
R
T

Residual Plot

कोई pattern नहीं होना चाहिए Bilkul randomly spread होना चाहिए

* A Residual Plot is a plot of Models Fitted Values (y_hat) and Models Residual Values
* The important aspect of a Residual Plot is that it should be absolutely random and should not exhibit any shape.....
* If there is some pattern in the Residual Plot then it means that most likely the errors are related and Regression is not a good fit.

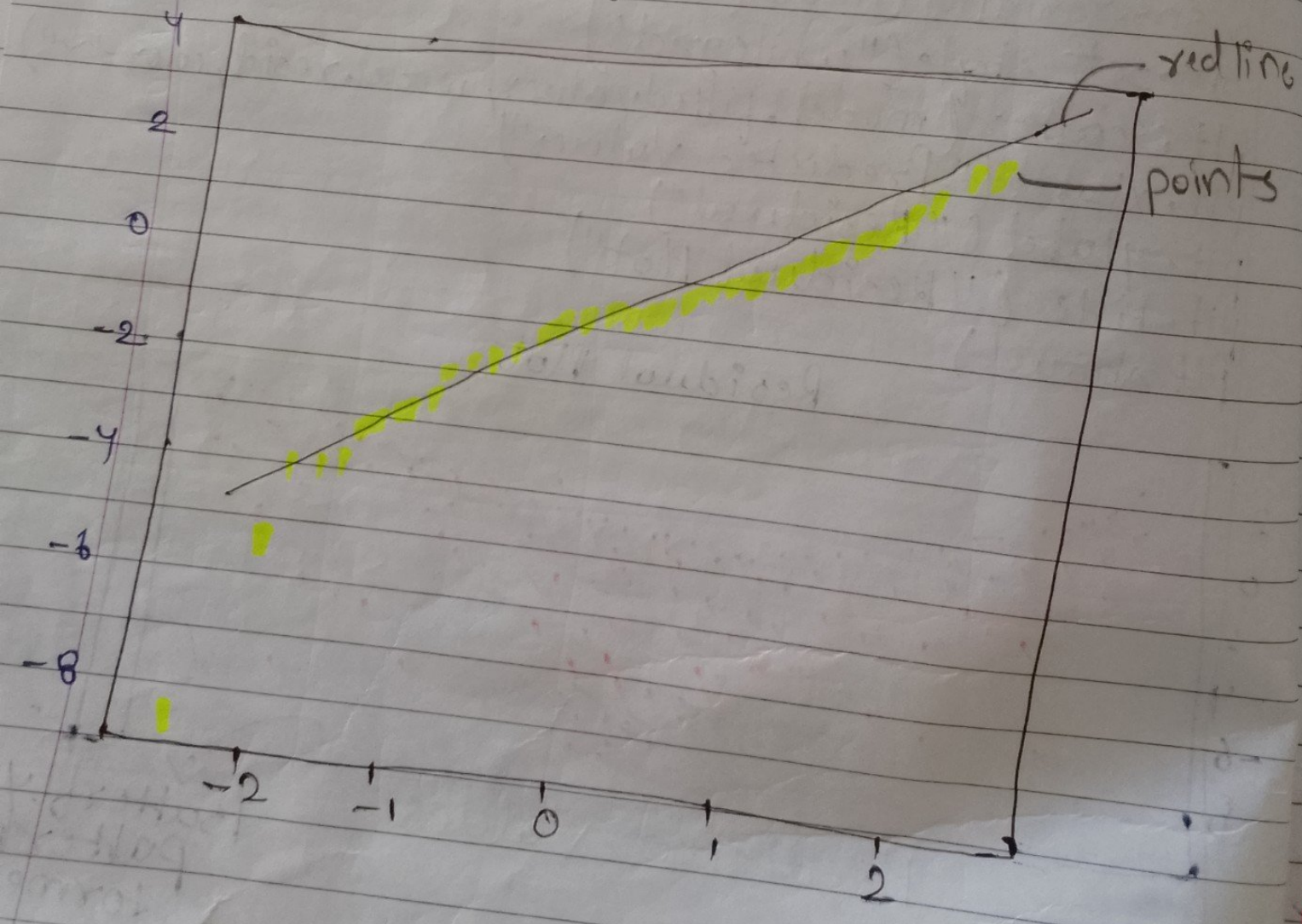
```
sns.residplot(model.fittedvalues, model.resid, color="m")
sns.set_style("whitegrid")
plt.scatter(model.fittedvalues, model.resid, color="r")
plt.xlabel("Predicted Values")
plt.ylabel("Residual")
plt.title("Residual Plot")
plt.show()
```



ProbPlot also known as Q-Q Plot

```
stats.probplot(model.resid, plot=plt)
plt.show()
```

The points should stick to the red line and if that happens
then we assume normality. However, if the Residual
are not normal
then the data points will deviate from the red line
Larger the Deviation, highly likely it is that the data is not Normal



Multicollinearity (3)

Predictor Variables should not be highly correlated amongst each other

If the prediction are Highly Correlated then is this a problem.

Yes, it is a problem because the predictors have the same prediction power and it would become difficult to identify which of those predictors are responsible in generating the prediction

The ideal hypothesis is that the predictors should be the of independent of Nature

CORRELATION PHOT

if two predictors are highly correlated amongst each other then we will drop that predictor which less ~~than~~ correlation with the target

Condition Number

Condition Number is another way to find the the Multicollinearity in the variables.

if the

- CN Number is <100 : No Collinearity whereas if
- CN Number is between **100-1000 : Mild Collinearity
- >1000 : severe Collinearity

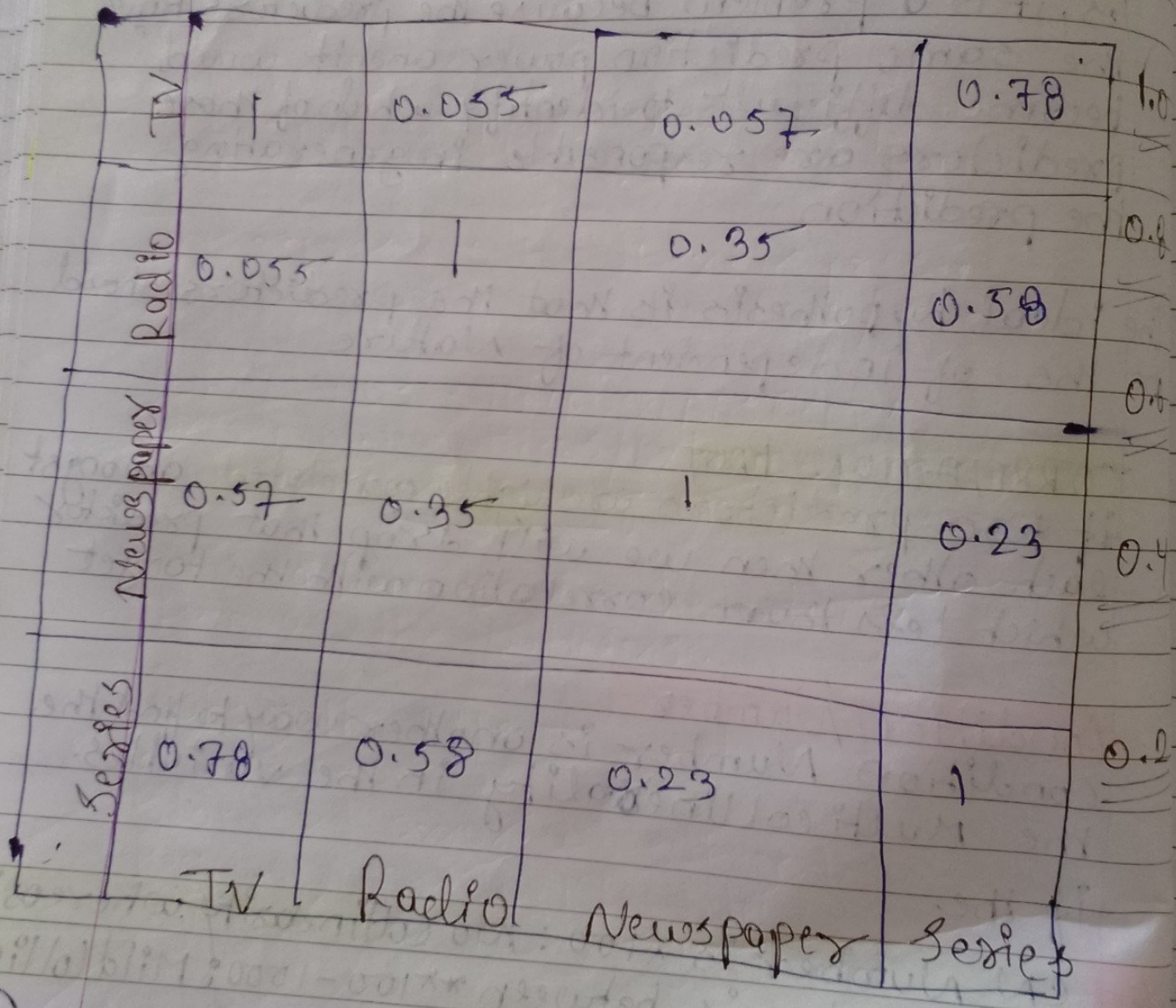
Variance Inflation Factor:

So in this deployement, it takes all the predictions and removes the target

make one predictor as target and using remaining predictors, it will build

VIF Value = 1 : No collinearity VIF value 1 to 5 : Mild collinearity ** VIF Value > 5 : Severe collinearity

```
sns.heatmap(advertising.drop(["y-hat", "residuals"], axis=1).corr(), annot=True)
```



Q. What is difference between Condition No. and VIF.

VIF

```
from statsmodels.stats.outliers_influence
import variance_inflation_factor
```

Drop the Target

```
X = advertising.drop(["sales", "y-hat", "residual"],
                     axis=1)
```

make a list and store the values of VIF

```
vif_list = [ ]
for i in range (X.shape[1]):
    vif_list.append(variance_inflation_factor(X.
                    values, i))

pd.DataFrame({"Features":X.columns,"VIF":vif_list})
```

	Features	VIF
0	TV	2.486772
1	Radio	3.285462
2	Newspaper	3.055246.

CONCLUSION Since the VIF value is less than 5, these predictors are good to go, we can use all the predictors are good to go. we can use predictors to build the model.

However, if any of the predictor(s) had a high VIF value, we would have concluded. that there is presence of High collinearity in that variable and that feature should be excluded from the model building

HETEROSKEDASTICITY

* The assumption under Heteroskedasticity is Equal Variance of Residuals. In other words. it tells us that the sample drawn from the population has equal variance or not

* The shape of Heteroskedastic pattern is either a open tunnel or closed tunnel.

This represents that there is presence of unequal variance in the Data/Model

WHAT CAUSES HETEROSKEDASTICITY

presence of outliers/Extreme points in the data is very case of unequal variance

Way to Remove Heteroskedasticity

Remove the Outliers.

Whats the Effect

Standard error will either Increase of Decrease. This will lead to Predictor being Significant or Not Significant

Confidence Interval Region will also narrow down or become wide. Net, the model will loose reliability

(5) AUTOCORRELATION OF ERRORS

DURBIN WATSON TEST

* If there is Autocorrelation of errors, then the Metric that will be severly impacted is the Standard Error
* If DW Test statistic = 2 No Autocorrelation
* If DW Test statistic is between 0 to 2 Positive Auto Correlation
* If DW Test Statistic is between 2 to 4 Negative Auto Correlation

Note: The Rule of Thumb is that DW value between 1.5 to 2.5 is acceptable

```
from statsmodels.stats.stattools import durbin_watson
durbin_watson(model.resid)
```

NO AUTOCORRELATION

2.0836480529 4407

Day 41 Session 3

FEATURE SELECTION METHODS

(*) when we build a model, we include all the features to build our model

(*) One way to find if the feature is significant or not is by applying Statistical Tests during the EDA step

(*) The Machine learning way is to use a wrapper Method known as Sequential Feature Selector coming from MLxtend Library

How SFS works?

THE CONCEPT IS CALLED FORWARD SELECTION OR BACKWARD SELECTION TECHNIQUE

Note: There is another approach known as Step wise Regression Approach

How SFS works?

(*) Forward selection Approach: It will start with 1 predictor and build the model. It is a blank slate initially and will start with 1 predictor and capture the Adjusted R2

(*) Then, it will add another feature, build the model and measure the Adjusted R square. If the Adj R2 score has increased, it will keep the predictor else remove the predictor and try the remaining predictors.

(*) This will process will keep on going till all the predictors are not exhausted. In the end, we will get the Final List of the predictors.

x1	x2	x3	x4	y
=	=	=	=	=
=	=	=	=	=
=	=	=	=	=

① Build the Model using all features
$y = f(x_1, x_2, x_3, x_4)$
↳ Pvalue.

Statistical Approach

(I) [x1 | Y] → LR Model → R^2 → 0.65

(II) [x1 | x3 | Y] → LR Model → R^2 → 0.78

[x1 | x3 | X | Y] → $R^2 = 0.65$

② The similar Approach works for ** Backward Elimination Technique ** where the model will be build using all the predictors and then the predictors will be removed recursively to find the impact on Adjusted R2. The final list of the important features will be generated in the end.

अब हमें यहाँ feature selection krna Hai

```
# ! pip install mlxtend
from mlxtend.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression
# FORWARD SELECTION APPROACH
sfs = SequentialFeatureSelector()
      (estimator=lr, forward=True, scoring="r2",
       cv=5,
       k_features="best")
```

→ यहाँ पे तो नाम जहाँ से जिस model में feature selection करना हो।

cv=5 पाँच बार चलेगा ये

(forward selection approach k liye likha Hai ye)

```
# It will (sfs) will be applied on X and Y
sfs.fit(X, y)
# Extract the Best features.....
print(sfs.k_feature_names_)
```

('...', '...s', 'T4', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_out', 'Windspeed')

Backward El...

sfs = SequentialFeatureSelector(estimator = lr, forward = False, scoring = "r2", cv = 5, k_features = "best")

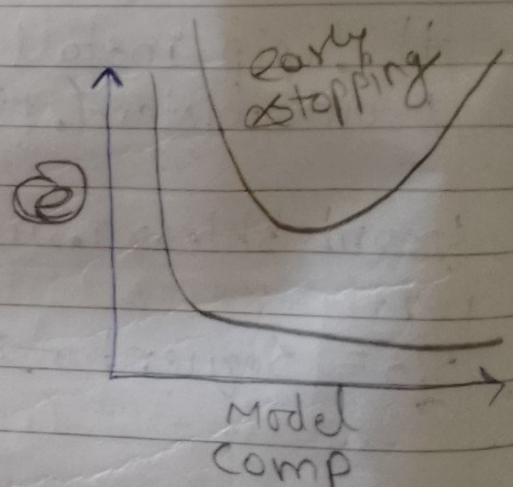
sfs.fit(X, y)
print(sfs.feature_names)

['T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility']

Cross Validation

Problem statement
↓
Data
↓
EDA
↓
Modelling → Train < Train, Validation
Modelling → Test

Train / Validation set	— ①
Test set ~ Predict	— ②

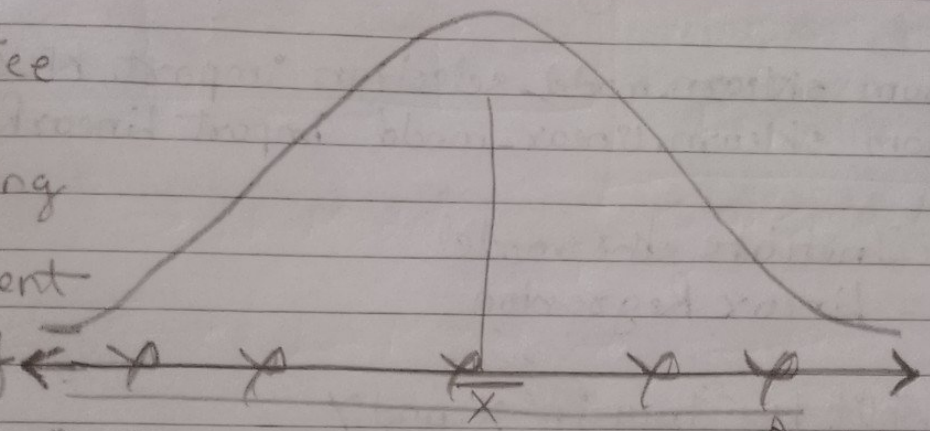


Cross Validation → cross validation means machine को ko baar ~~...~~ karana and. ऐसा करने se error कम करें तो कम होगा but इसे Time काफ़ी increase हो जाएगा जैसे ऊपर graph Mein दिखाया गया है।

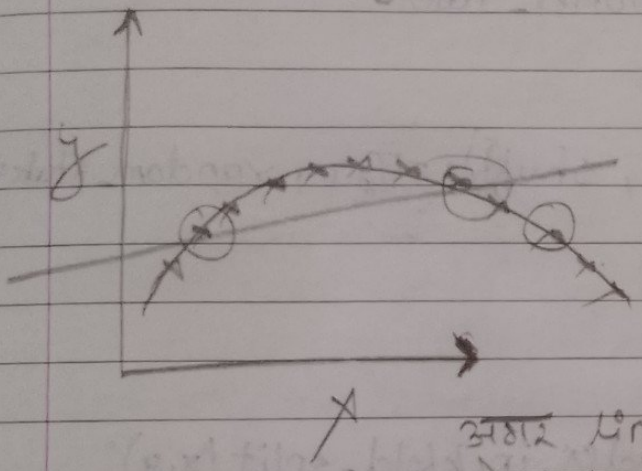
Shot on AWESOME A05s

Day 3 Session 1

→ D-Tree
→ RF
→ Bagging
→ Gradient
→ Boosting



→ Overfitting ∧ High Variance Problem



Linear Regression Model एसा Underfitted Model होता

$y = mx + c$

$x^{1.2}$ / $x^{1.5}$ / x^2

अगर Linear Regression model में error और अगर Dot से Dot match करना हो तो predictors का power बड़ा दो and means $y = mx + c$ में x की power बड़ा दो, जिससे वह curved होने लगा जाएगा।

35 Data

Xtrain	Xtest
Ytrain	Ytest

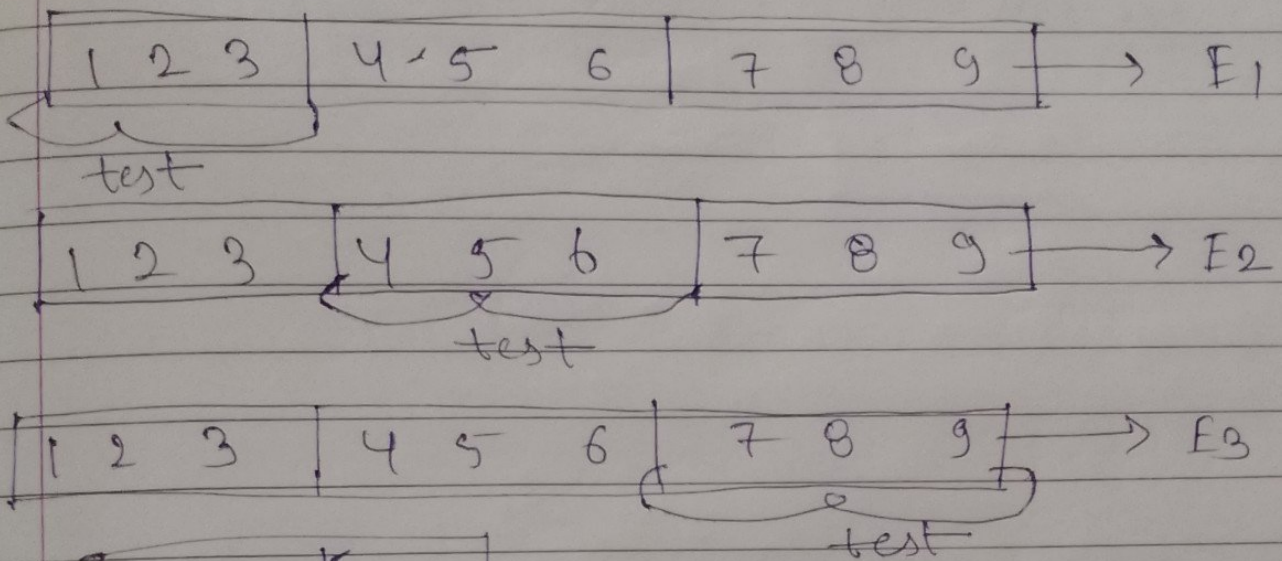
80-20 → Cross Validation

k fold CV

epoch-1	Tr	Tr	Tr	Tr	Test
epoch-2	Tr	Test	Tr	Tr	Tr
epoch	Test	Tr	Tr	Tr	Tr

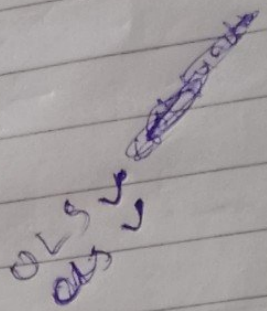
Cross validation

3-Fold Cross Validation



$$E = \frac{1}{k}\sum_{i=1}^{k} E_i$$

cross validation solves the problem of test overlapping.

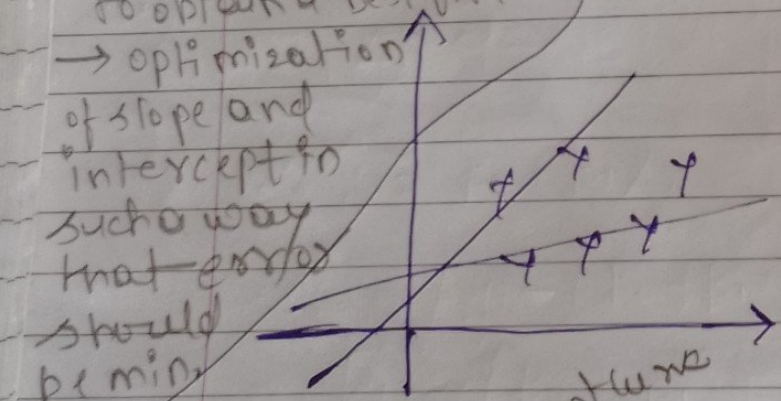


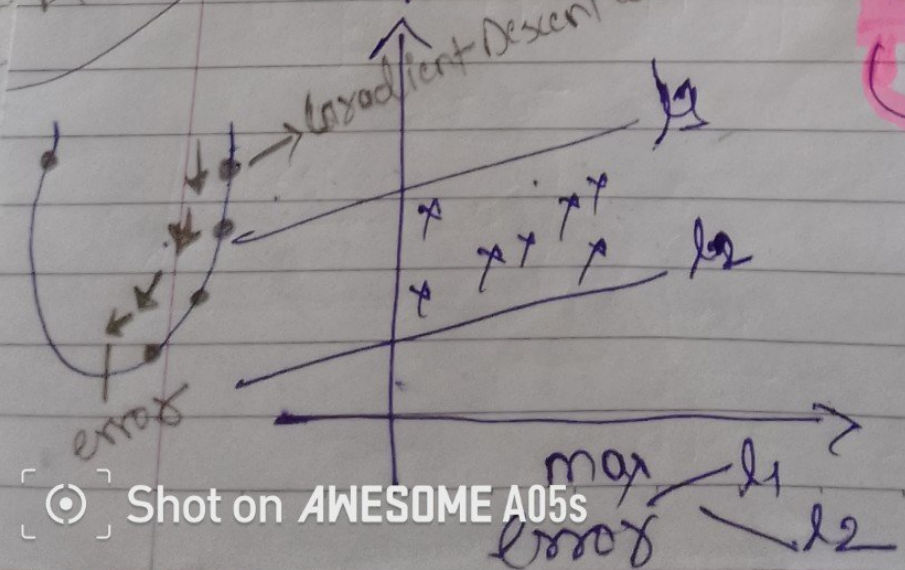
Week ② Day ① Session ①

Gradient Descent

→ Job of gradient descent is to reduce the error in such a way that we are able to obtain a best fit line

→ optimization of slope and intercept in such a way that error should be min.





OLS — stat model
sklearn

OLS("y ~ x", data).fit()

$$\beta_1 = \frac{Cov(x,y)}{Var(x)}$$

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

$$\hat{y} = \beta_0 + \beta_1 x_1$$

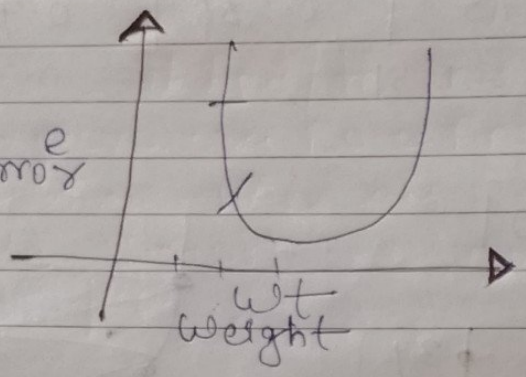
↳ predicted ⇒ Residual

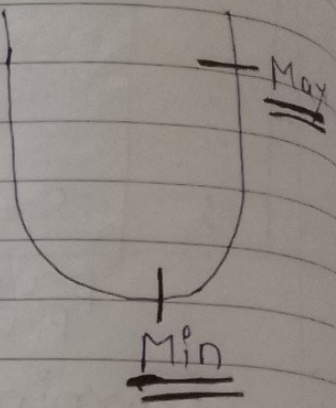
$$SSE = \sum (y_i - \hat{y})^2$$

(53) . Convergence

→ when in 2 or more than 2 iterations, the error reduction is Minimal, that is the point of convergence

So, goal is to reach from maxima to minima





(54)

Gradient Descent

- Batch Gradient Descent (Underfitted)
- Stochastic Gradient Descent (Overfitted) (row by row)
- Mini Batch Gradient Descent ↳ Best of three

Batch ⇨ 1000 Rows. → 100 Rows, 100 Batches

(55) Sk learn → Grid Search OR Random Search

{ lr-rate = [0.01, 0.001, 0.1, 1]
CV = 5 }

SSE 0.01 = ?
SSE 0.001 = ? min.
SSE 0.1 = ?
SSE 1 = ?

Grid Search (estimator = param_grid = params

LR = 0.001

(56) Auto sales of car

Data

	X	Y
MT	0	8
AT	1	16

(57)

$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2$

x_1 — {β_0, β_1} w_{t_1}

EDA → x_2 — w_{t_2} — $f(x)$ pred → $\hat{y}$

x_3 — w_{t_3}

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

cost fun

$MSE = \frac{1}{20} \sum_{i=1}^{n} (y_i - \hat{y})^2$

$RMSE = \sqrt{\frac{1}{20} \sum_{i=1}^{n} (y_i - \hat{y})^2}$

Day 2 Session 2

Regularization — overfit

lasso

Elastic Net

Ridge

$SSE = \Sigma (y_i - \hat{y})^2 + \alpha \cdot \sum_{i=1}^{f} (\beta_i)^2$

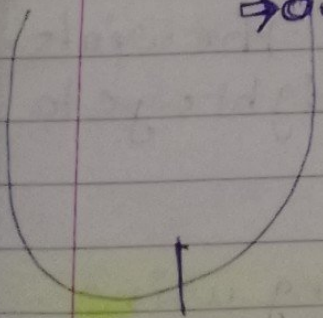
50 features

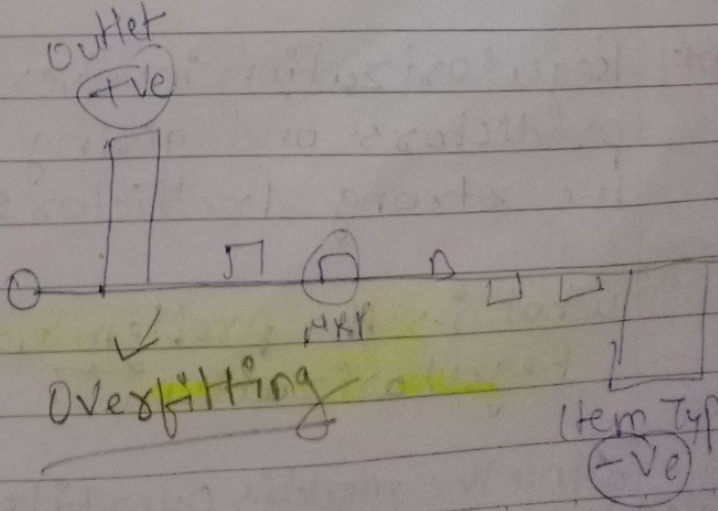
$SSE = 0.14$ $\alpha = 0.05$ $\sum_{i=1}^{p=50} \beta_i = 0.25$

$0.14 + 0.05 * 0.25$

$\rightarrow 0.1525$

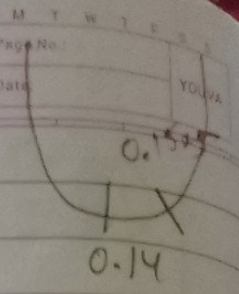
→ RMSE = 0.14





1150

1100 | 1200



Ridge

$SSE = \sum (y_i - \hat{y})^2$ Ridge Regularization.

$\sum_{i=1}^{n} (y_i - \hat{y})^2 + \alpha \cdot \sum_{i=1}^{p} (\beta_i)^2 \Rightarrow$

SSE = 0.14

$\alpha = 0.05$

$\sum_{i=1}^{p=50} \beta_i = 0.25$

Regularization Parameter

$0.14 + 0.05 \times 0.25 = 0.1525$

Regularization Parameter = Bias

(62)

Outlet ID = 10

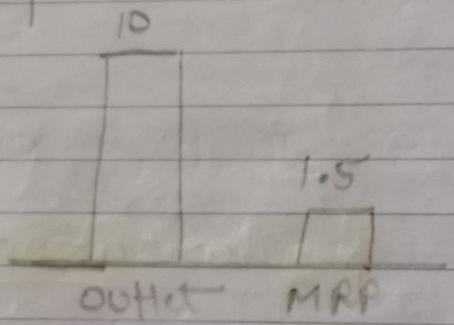
MRP = 1.5

$\hat{y} = \beta_0 + 1.5 \times MRP + 10 \times$ Outlet ID

$SSE + \lambda \Sigma (\beta_i)^2$

$\beta_1 = 10 \sim 5$

$\beta_2 = 1.5$



Q what is Regularization Parameter?

~~Deal ID~~

~~Real ID~~ ~~Soil ID~~

Regularization → penalizing the weak predictor and giving the weightage to strong predictors.

 # Regularization

* Regularization involves penalizing the weak predictors and giving correct weightage to the strong Predictors.

** **What is the problem we are solving using Regularization ****

Since the model is overfitted and gives weightage to the weak predictors. Thus we introdu

How it works

It will introduce a Bias in the model and the equation for the said Model under Ridge Regression would be SSE + alpha * sum of (coefficient_weights)^2

WHAT REALLY HAPPENS IS WE INTRODUCE BIAS TO CONTROL THE VARIANCE

L2

Ridge = $\sum_{i=1}^{n} (y_i - \hat{y})^2 + \alpha \cdot \sum_{i=1}^{p} (\beta_i)^2$

10 columns lesser Data small

$\beta_1^2 + \beta_2^2 + \dots \beta_n^2$

weak predictors ko kam weightage karega

L1

Lasso = $\sum_{i=1}^{n} (y_i - \hat{y})^2 + \alpha \sum_{i=1}^{p} ||\beta_i||$

when you have more than 100 features

weak predictors ko high weightage kr dega

Elastic Net — Lasso / Ridge

$a L_1 + b L_2$

$a + b = 1$

$L_1\text{-ratio} = \frac{a}{a+b}$

$a L_1 + b L_2$ → a → 0, b → 1

If $L_1\text{-ratio} = 0$

$\frac{a}{a+b} = 0 \quad a = 0$ | $a + b = 1 \Rightarrow b = 1$

$a = 0$ & $b = 1$

Q. When L1 is 0, then कौन सा Model बनेगा? Ridge या?

$L_1\text{-ratio} = 0$

$\therefore L_1\text{-ratio} = \frac{a}{a+b}$

$\frac{a}{a+b} = 0$

$a = 0$

$a L_1 + b L_2 \Rightarrow b = 1$

$L_1\text{-Ratio} = 1$

$\frac{a}{a+b} = 1$

$a = a + b$

$b = 0$

$a + b = 1$

$a = 1$

$a L_1 + b L_2$

L_1

$L_1\text{-ratio} = 0$

$L_1 \rightarrow L_2$

else → L_1

As per Regularization Model, we are increasing Bias to Reduce the Variance (Overfitting)

So, when in a model we increase Bias to reduce variance or increase variance to reduce bias, such a relationship is called Bias Variance Trade off.

Ridge Regression

```
ridge = Ridge(alpha=10)
for
  xtrain
  ytrain
  xtest
  ytest
  yhat = ridge.fit(xtrain, ytrain).predict(xtest)
  print("RMSE:", rmse(ytest, yhat))
```

RMSE: 1021.20594
RMSE: 973.50693
RMSE: 931.671122
RMSE: 1017.5012
RMSE: 1000.1234

Lasso Regression

```
lasso = lasso(alpha=50)
for
  xtrain =
  ytrain =
  xtest =
  yhat =
```


.train).predict(xtest)

```
from sklearn.model_selection import GridSearchCV
params = {"l1_ratio": [0.1, 0.12, 0.2, 0.5, 0.65, 0.7, 0.75, 0.33], "alpha"
grid = GridSearchCV(                                    [1, 2, 3, 4, 5] 3.
    estimator = cnet, param_grid = params, scoring =
    rmse , cv = 5)
```

grid.fit(X, y) # Fitted the Model

> GridSearchCV
> estimator : ElasticNet
:> ElasticNet

grid.best_params_
{'l1_ratio': 0.1}

Cross validation iteratively splits the data set into two portions : a test and a training s
The prediction errors from each of the test sets are the averaged to determine the expected prediction error for the whole model

K- Fold - Cross - Validation

- Data is divided into 'k' folds or subsets
- The model is trained 'k' times, each time using k-folds for training and remaining fold for validati
- Average performance across all folds is often used asses the models generalization
- row by row

Leave-one-out Cross validation

- 'k' is set equal to the number of data points.
- Each data point serves as a validation set exactly one, while the rest are used for training
- Provides a more thorough assessment but can be computationally expensive for large datasets.

k folds splits the data into k ... and validation, whereas leave-one-out validation

whereas leave-one-out validates the model iteratively, leaving out one data point at a time, k-Fold is computationally more efficient while LOOCV provides a more exhaustive evaluation at a cost of increased computation.

Leave one out

This will give 100% efficiency so this we have never used for modelling

Rfe No. of features are best.
We have option of providing best no. of features

Rfe min best nhi ho kuch nahi hota, hume manually karna hoga

GridSearch CV

1. set the learning rate range ✓
2. train model with each learning rate ✓
3. evaluate the performance ✓
4. choose the learning rate that gives best result. ✓

GridSearch CV is a technique for finding the optimal parameter values from a given set of parameters in a grid

Ay