MACHINE LEARNING

Machine learning → The subfield of computer science that gives computer the ability to learn without explicitly programmed.

↓

Patterns in the data

Machines learns pattern from the data and replicate the same in future

* Why ML is famous these days?
  - → Advance Processor
  - → Data is new oil

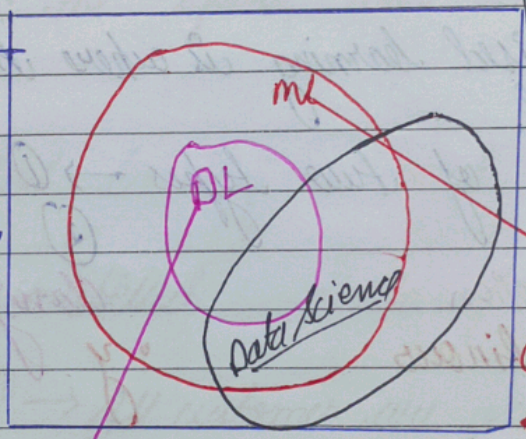
* Lot of data
  - ⇓
  - Process this data (advance processing)
  - ⇓
  - Valuable insights
  - ⇓
  - improve business performance

* Difference b/w AI, ML, DL

AI ←

↓

Describes how computer and technology mimics human intelligence.



ML focuses on creating algo and statistical model to let computers learn and make prediction without explicitly programmed.

DL is specialised ML algo. that mimics human brain
- → Multilayer Neural Network.

Types of machine learning

1→ Supervised
2→ Unsupervised
3→ Semi-Supervised
4→ Reinforcement learning

1→ Supervised ML →

Lets say we have a data set of houses contains
① area of house ② No of rooms ③ price of house
↓ ↓
Independent variable ↓ dependent/target

Now for new data we predict the target variable based on independent variable.

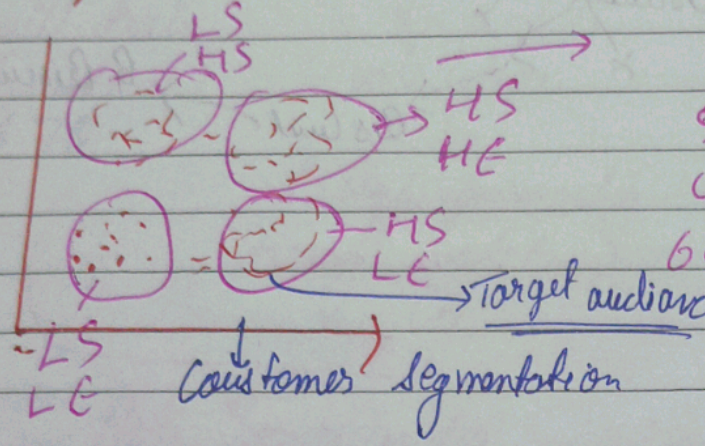
Here past data act as supervisor.

Supervised learning is where the data we fed is labeled.
it is of two types → ① Regression
② Classification

Regression	Classification
y is continous	y is discrete
eg → prices of house	eg → Pass/fail.

2) Unsupervised learning

No supervision → No 'y' → No target variable → that means no past data



Salary	expense
8000	4 Lac
4000	7 Lac
6000	3-5 Lac

* We try to find similar groups in data and classify the

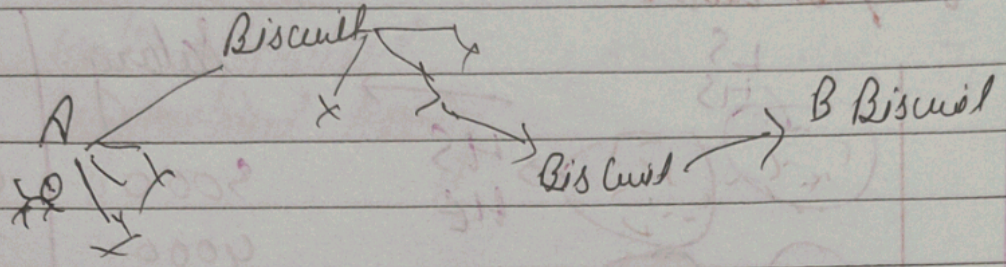
ex → in above ex created 4 groups (HS,LE)(LE,LS)..
① we give 20% discount to HS, LE persons so the spend more on expenditure.

4) Semi Supervised learning → ~~Comp~~ Combination of both Supervised & Unsupervised
↳ labelled + unlabelled

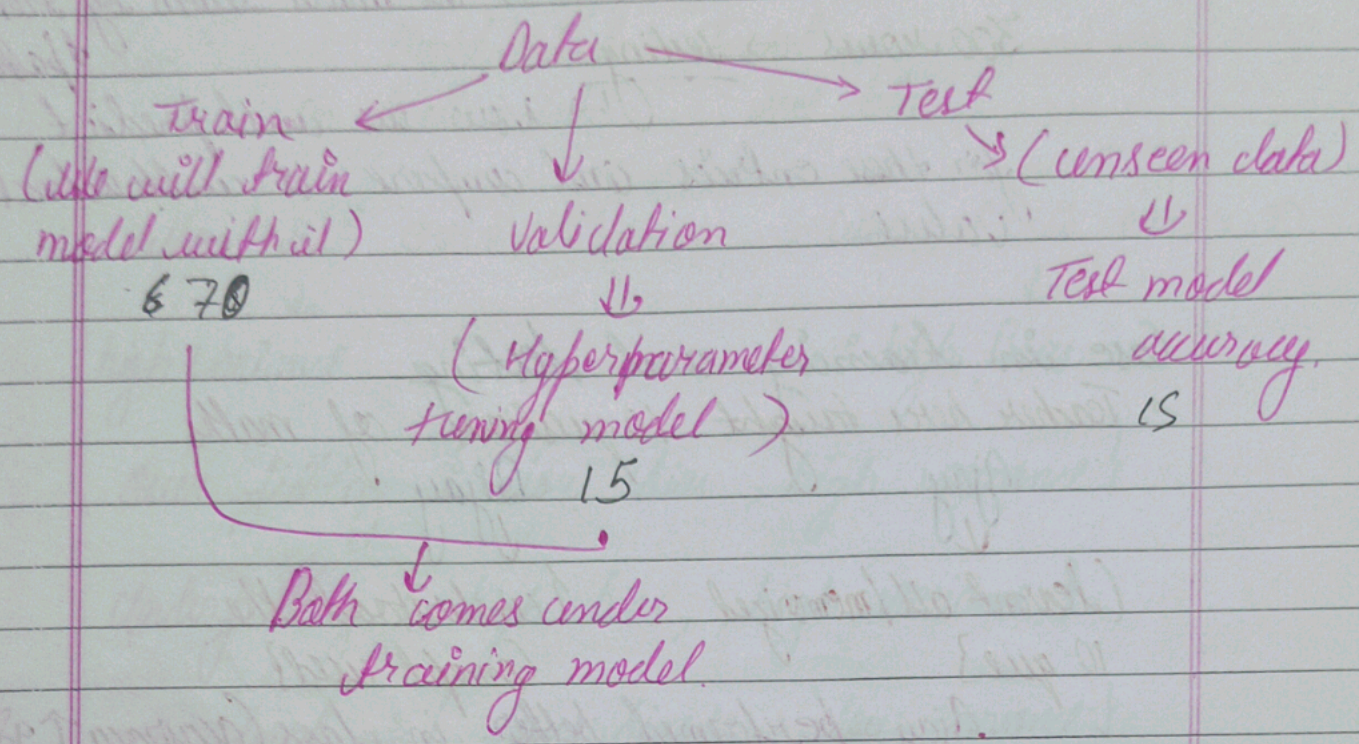
Netflix / prime → All customer are grouped / divided ~~into~~ into cluster (USL)
↳ The they suggest movies (SL)

Note → We convert Semi Supervised to either SL or USL

4⇒ Reinforcement Learning → concerned with how intelligent agent take action in an environment to maximize the award.



Train, Validation and test data



Model training → given the data estimate the prediction function by minimizing error.

area of house	Rooms	Price
2	4	3
3	6	4
4	8	5
5	10	6

→ This will generate a pattern, $y = f(x)$

with this our model will learn pattern, and predict the price based on different entries.

$y = mx + c$ { m is slope, c is intercept }

now lets say we have 1000 rows, so we will use 700 rows → training
↳ the model learn by trend & pattern
300 rows → testing
↳ Now we will predict for these entries and compare it with actual value

Case in training and testing
- Teacher here taught 10 question of maths

Ajay
↓
(learnt all/memorized 10 que)

Bijay
↓
(understood the concept used)

Ajay performed better in class (accuracy ↑ 98%)
10 new que came in board exam
→ Here ajay have memorized only these 10 que, but when new problem came he failed, scored (30%), while bijay scored (89%)
→ Bijay performed better (89%)

⟹ This is case of overfitting
↓
in machine learning, when our model score/gives high accuracy. (model perform well on train data, but failed in test data).

⟹ opposite of this is underfitting
Test accuracy → 40%, test accuracy - 35%
→ our model did not learnt any thing

⇒ Bias / variance

Data

Model trained on train data — Tested on test data

When test error ↑ → high Variance

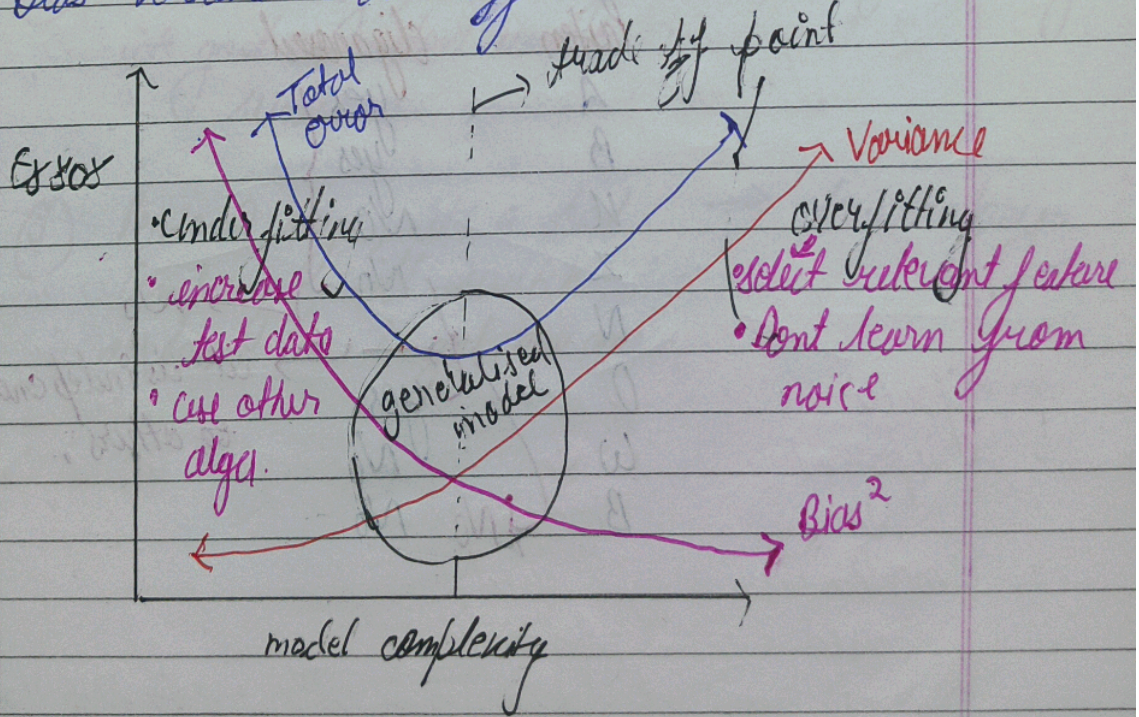
When train error ↑ → high Bias

over fitting { low bias, high variance }

underfitting { high bias, high variance }

generalised → { low bias, low variance }

⇒ Bias Variance trade off



- Missing Values and its treatement.

missing data → some data points are not filled/missed

Reason → Human error
data is corrupted.

missing data occurs when some info is not saved/stored in our data set.

1→ Missing Completely at Random (MCAR)

The data missing is independent of observed and unobserved data/missing data.

Survey → if person is diagonesed

Paiten	diganosed	
A	yes	
B	yes	
K	No	
Z	No	
N		→ No
O	yes	it is independent of to others.
W	No	
B	No	No

(2) Missing at Random (MAR) → missing data depend on observed data not on missing data itself

survey → To income of people.

family → income of all family members

lets say one people in family don't want to disclose the income due to x-y-z reason

(3) Missing data not at Random (MNAR) → missing value depends on missing data itself

Eg → Thought of employee / satisfied with their job ∝ Income.

(1) Person 1 say, i am satisfied its good and dont want to switch ⇒ high salary

(2) Person 2 says, its a bad company i'll switch as soon as i get new one ⇒ low salary

Missing Value Treatment

A rule says whenever you want to ~~tre~~ treat missing data, {always as business team}
↳ {I will treat it like this, it will affect this, should i go on.?}

Why missing value treatment :- For analysis & ML prediction it is important to deal with missing values

Note :- 1. if missing value is less then 1%, and data is huge in amount, drop the missing data

2. if its greater than >1%, impute / replace the missing data

imputation

- numerical
  - if outlier is present → median is used
  - if not → mean is used
- categorical → mode is used to impute

3. if any col has missing value greater than 40%. drop that column.

4) impute the missing value with a random number, or with a value which is not possible

Marks
100
50
60
←→ -1/Na
90
70
30

Note → whenever we fill missing value, try to plot a histogram and if the distribution is normal & no ~~one~~ many outliers were present, fill with mean in that case.

Imbalanced data.

When we have 1 target variable and it has 2 outcomes → binary classification problem.

eg →

Sugar level	Cholestrol level	diabetis (y/n) (y)
28	102	0
38	100	0
31	105	0
77	193	0
83	201	1

data says out of 90% of the y is 0 (non-diabetic) or not

and you haven't make any model and your predicted all values to 0 → got accuracy 90% → why? → because 90% is 0

→ This is called class imbalance

im + balance class

When one class has very high percentage as compared to other class,

→ in our case → class 1 → 0 (non-diabetic) → class 2 → 1 (diabetic)

• We don't want our model to learn the only data pattern that is in majority

↓

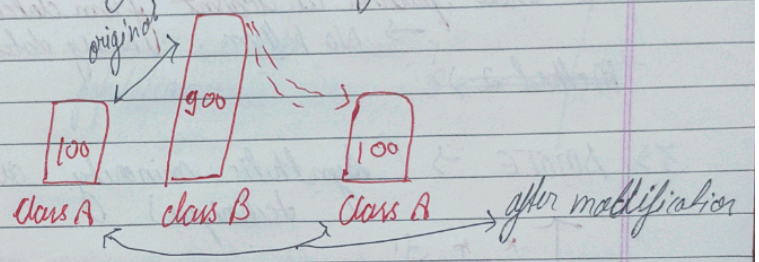
Thats why we need to deal with class imbalance

class/ imbalance → When the difference in the count/ percentage of both class is very huge.

90% → class A } → This is std. imbalance case
10% → class B

Solution for class Im-balance

① Under Sampling (Down sampling)



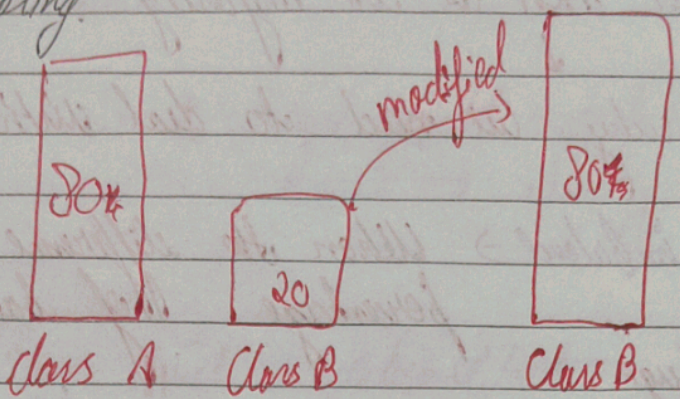
Hamne jo imbalance data mai se class B 90% tha use down sample karke 10% mai laye

Now class A → 100 values } overall data 50%
class B → 100 values

Disadvantage → A lot of data is lost

→ yaha pr hamara data waste hua jiske liye hamara bhut resources use hue the.

2-> Over Sampling



Method 1 → repeat class B to get in the level of class A

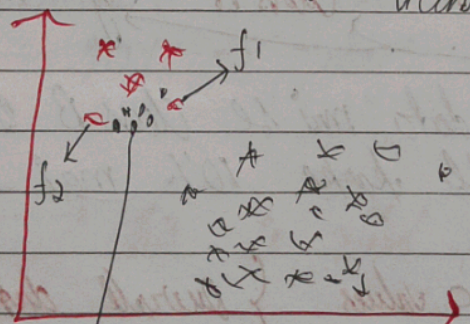
↓

Disadvantage → ML is about learning patterns but here data is repeated so no new pattern is learnt from data.

→ No pattern, Noisy data.

~~Method 2 →~~

3> SMOTE → (Synthetic - minority oversampling technique)



→ problem of No pattern is solved.

→ like datapoints will always be together, this is reasone Smote is effective.

Synthetic value

↓

get by avg, ~~mean~~ mode, knn etc

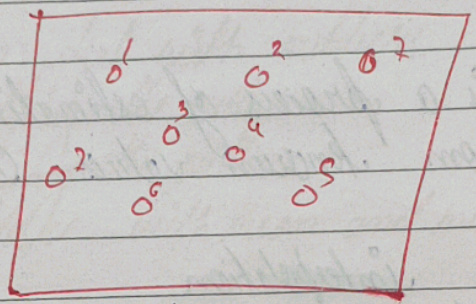
Use in industry

How to use Smote & down sampling in data

→ 100 → Smote → 800

900 → down sampling → 700

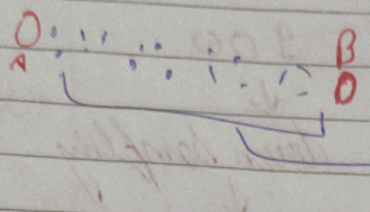
* In practical implementation



population

With replacement	Without
1	1
2	2
3	4
2	5
1	7
~~6~~	

Data Interpolation



- interpolation is imputing unknown values based on known values
- it is where we impute new data values b/w two known data points

interpolation → it is a process of estimating unknown values from known values

→ Types

1 → linear interpolation
2 → cubic interpolation
3 → polynomial interpolation

Use Cases → 1 → to fill missing values
2 → to impute new values

Outliers And its treatement

outliers → these are the extreme values

in outlier treatement
step 1 → check five point summary

To check outlier using plots
① → distplot, boxplot

How to deal with outliers
① dropping the outlier
② capping the outlier
③ replace with mean and median

① dropping
↳ drop all values ~~apart~~ (less than lower fence & greater than upper fence)

② imputing →
↳ replacing the outlier values with mean and median.

③ capping → capping the outliers with the nearest value which is not an outlier
↓
(lower fence, upper fence)

* Feature extraction and selection

Steps we have covered till now

- understanding problem statement
- Data ingestion
- Data preparation / preprocessing
  - → missing value treatment
  - → class Imbalance
  - → outlier treatement

next steps →

- Data encoding
- Feature extraction
  - Creating new features
  - modifying the exist feature
  - selecting riyht feature.

Feature extraction → process of selecting and extracting the relevant features from raw data.

CN → out of 100 → 60 are relevant

A question arises out of that → ~~tuhoo~~ How it can be a problem

↓

There comes a concept, → curse of dimensionality

Curse of dimensionality

↳ with increase in no. of features
① Model training becomes computationally expensive
② Model interpretation becomes complex

How to solve

① Create new features

↳ ex

Distance	Time
100	1
110	2
120	3
130	4
⋮	⋮

→ Speed = $\frac{D}{T}$

② Modifying existing features

① changing data types

Date → day / month / year

③ feature scaling (optional)

Area of house	rooms	cost
1500	2	60
2100	3	65
2500	4	100

feature scaling → it is the process to scale all the features to same scale

↓

why → • Many of the algorithms are distance based, that's why if high magnitude comes, it becomes computationally expensive.
• interpretation becomes easier.

Types of Scaling.

(1) Standardization

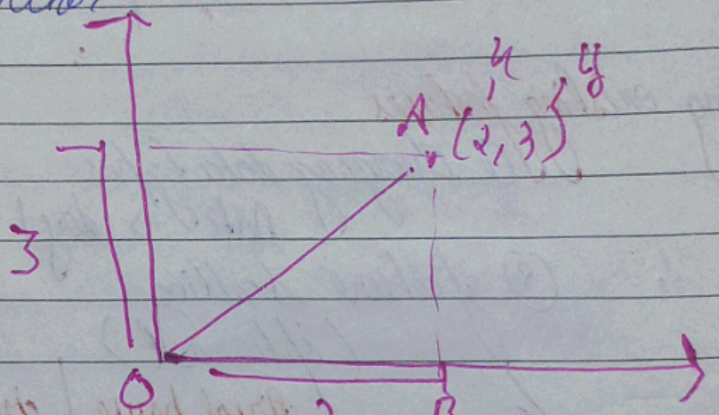
$$Z_{score} = \frac{x_i - \bar{x}}{\sigma}$$ for SND $\mu = 0, \sigma = 1$

(2) Normalization (min, max Scaller)

↳ reduced to [0 to 1]

$$x_{scaled} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

(3) Unit Vector



$2\hat{i} + 3\hat{j}$

$$OA^2 = OB^2 + AB^2$$

$$\overrightarrow{OA} = \sqrt{3^2 + 2^2}$$

$$\|OA\| = \sqrt{13}$$

$$\hat{u} = \frac{2}{\|OA\|}, \frac{3}{\|OA\|}$$

unit vector

(4) Selecting the right features

(1000) ⟶ (20)

methods

1) filter method
   ↳ calculate correlation with target variable features with larger value will be relevant one

2) Embedded method

3) Wrapper method

** PCA

df['total bill']

std

~~μ, σ~~ μ, σ

{Calculate {μ, σ}} : {apply the calculation (on dataset using μ, σ)}

$\frac{x_i - \mu}{\sigma}$

This is called as
- fit
- ↓
- train data

this is called as
- transform
- ↓
- test data.

* Scaling should be done after train-test split.

~~Scaling~~ Data encoding.

The machine only understand numerical no.

↓↓

So we need to convert object data to numerical data.

↓↓

for that we need encoding.

Types

1. Nominal / One hot encoding
2. Label and ordinal
3. Target guided ordinal encoding.

(1) Nominal / One hot encoding.

↳ converts categorical to Numerical

∴ No order in data.

Status
Single
Maried
engaged
Single
Single
maried

Single → 1 0 0 0 1

Maried → 0 0 1 0 0

engaged → 0 1 0 1 0

disadvantages of One hot encoding
→ A Column has many categories

→ The column we create are called as dummy variables

Note → n-1 dummy col. can explain n dummy col.

(2) Label and ordinal encoding
Label encoding →
→ assign numerical data to each category.

States	→	States
M		0
S		1
D		2
E		3
M		0
E		3

disadvantages
→ good for ordinal data, as for nominal data it can learn pattern.

Ordinal
→ High School = 1
College = 2
master = 3
Job = 4
→ used for ordinal data.

* Target Guided Ordinal Encoding

↳ does encoding based on their relationship with the target variable.

→ useful when we have large no of unique categories in categorical value.

→ categorical groups with meaningful of corresponding target variable.

Regression

Simple Linear Regression comes under supervised machine learning algorithms

Supervised
- Regression (continous)
- Classification (discrete)

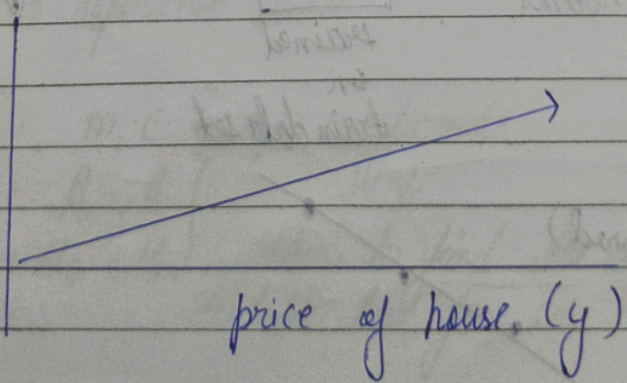
Regression → it is to establish relationship b/w two or more than two variables.

Linear → The relationship established is linear.

*** Simple linear regression attempts to determine the strength and characteristics of relationship b/w one independent variable (x) and another dependent variable (y). ***

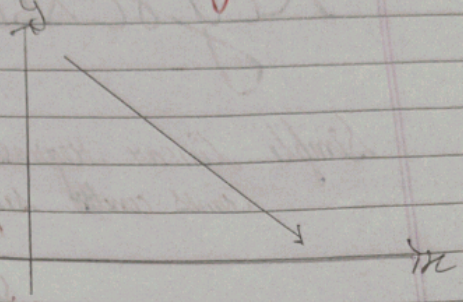
eg → predict house of price based on area

Area	Price
1100	50
1200	60
1300	70



eg -> Selling price of car w.r. to years used

x	y
age	price
1	11
2	10
3	8
4	6



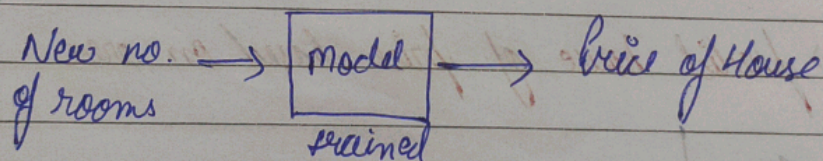
Simple → only one (x & y)

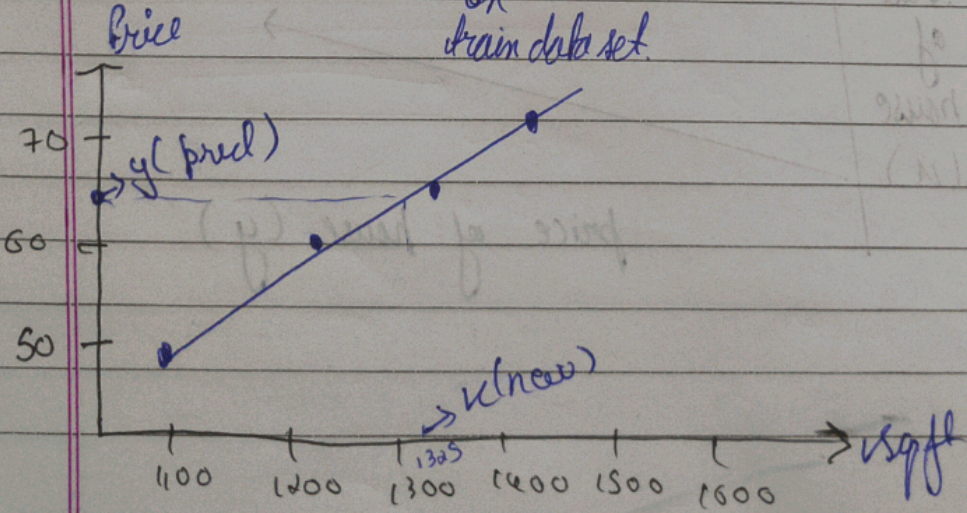
linear → captures linear relationship

Regression → understand relationship

→ There were only 4 data points thats why we understood the trend

→ if many data points exist we need to train our model





Here the line's eq = $y = mx + c$

$m \rightarrow$ slope $= \frac{y_2 - y_1}{x_2 - x_1}$

$c \rightarrow$ intercept ($\rightarrow$ where it cuts y axis at $x = 0$)

using $y = mx + c$ for new 'x' we will get 'y' correspond to it.

$\hat{y} = mx + c$

$\hat{y}$ estimator $\rightarrow$ you have predicted price of house based on m, x, c

$\rightarrow$ this also written as $\hat{y} = \beta_0 + \beta_1 x$ or $h_\theta(x) = \theta_0 + \theta_1 x$

Hypothesis representation

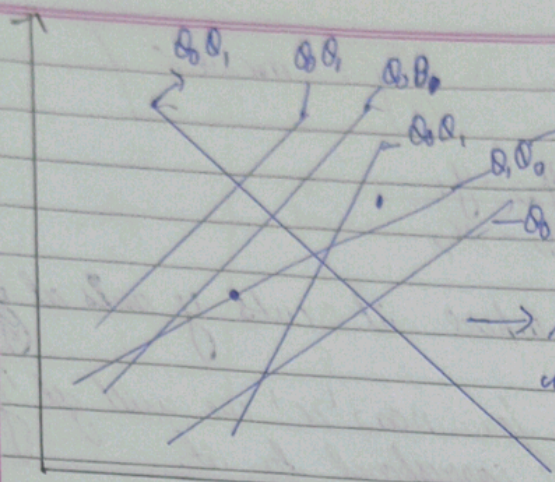
Here

$x \rightarrow$ independent variable

$y \rightarrow$ dependent variable

m, c / β_0, β_1 / θ_0, θ_1 $\rightarrow$ coefficients

How to find the coefficient suitable best for us



→ this have say error is least.

→ Among all these we want best (θ_0, θ_1)

We want that coefficient ($\theta_0\theta_1$ or $\beta_0\beta_1$) where error is least.

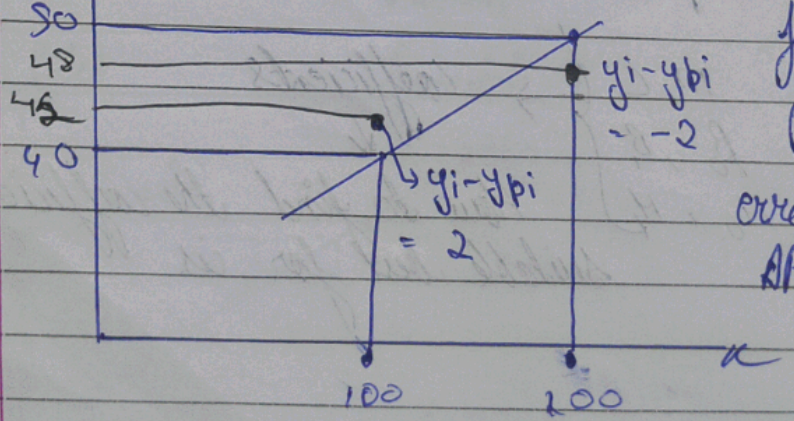
That coefficient also called as $\theta_0\theta_1$

for all lines we want → $y_{actual} - y_{pred.}$ = least error

for $y = mx + c$, (m, c) to be those best optimal $(\theta_0\theta_1)$

error = $y_{act} - y_{pred}$

$y_i - y_{pi}$



for DP_1 = +2 → error

for DP_2 = -2 → error

error for all DP_s → 2 + -2 = 0

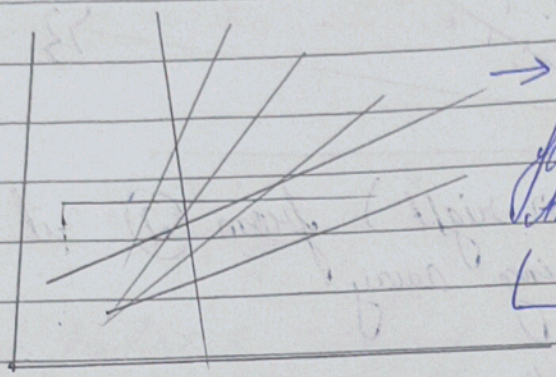
↓

is there no error ml,

To solve this problem instead of absolute error we measure squared error → $(y_{act} - y_{pred})^2$

error = $(dp_1)^2 + (dp_2)^2$

- $2^2 + (-2)^2$
- $4 + 4 = 8$



→ now we will calculate (error) for all the lines and choose the line with least (error)

(error → squared errors)

$\min (y_{act} - y_{pred})^2$

Best line ⟶ $\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ → $\hat{y} = mx + c$

↓

best least sq method

or

Ordinary least sq method

i = i^{th} line

n = no. of lines

$H_0(x) = H_0 x + H_0$

$\min \sum (y_i - H_0(x))^2$

$\min \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 x_i)^2$

* for best fit line we want MSE to be least.

Sum of Sq. error → $\min_{i=1}^{n} (y_i - \hat{y}_i)^2$

when we divide it by n

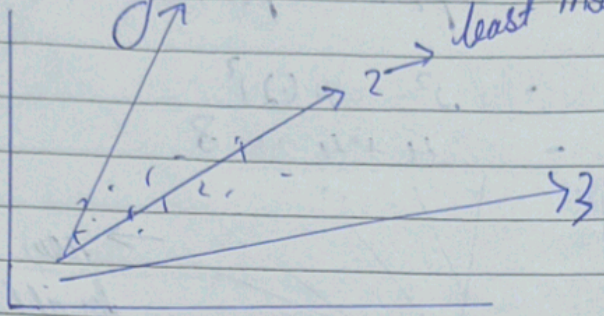
it becomes (MSE)

↓

mean sq. error = $\dfrac{\min_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$

⟹ MSE is used ~~to~~ as CF to get optimal (θ_0, θ_1)

Why CF is Penalty :



deviating (left or right) from (2) it is penalty as we are moving away

How to choose the best line

1→ Chose right combination of line

2→ Find MSE line

3→ Find Best line

⋆⋆⋆ The predicted line should be made in such a way that over all sum of error should be minimum or least

Sum of squared error

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

y_i → actual

$\hat{y}$ → y predicted

$\frac{\text{actual}}{n}$ → mean squared error

also called as cost function

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

Cost function = $J(\theta_0, \theta_1) \Rightarrow \frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{n}$ or $\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2$

n - no. of data point

y_i ⇒ actual value

$\hat{y}$ ⇒ y predicted value

$\hat{y} = mx + c$ / $\theta_0 + \theta_1 x$ / $\beta_0 + \beta_1 x$

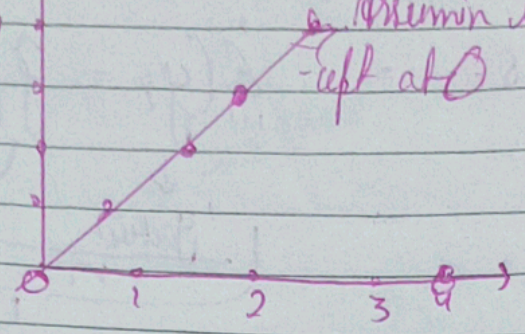
$$\min(J(\theta_0, \theta_1)) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2$$

Aim → minimize cost function to get optimal coefficient.

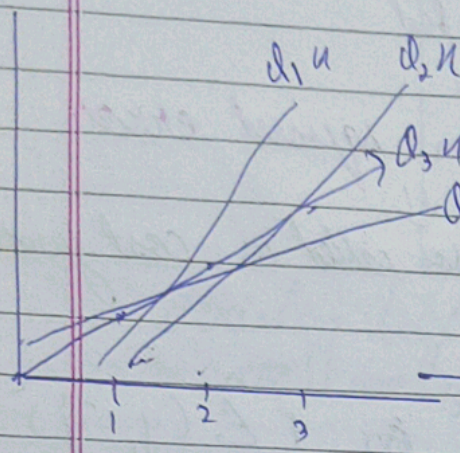
The data point were like

1	1
2	2
3	3

$\hat{y} = H_\theta(x) = \theta_0 + \theta_1 x$



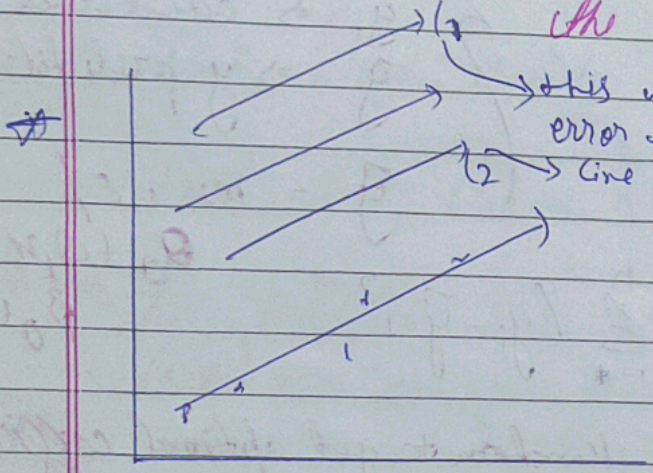
Assumin line inter-cept at 0



$\{ H_\theta(x) = \theta_1 x \text{ as } \theta_0 = 0 \}$

→ So How could we say any line is best fit line

↳ we will choose that one line with the least cost function.



this line has more error than this line

{ in short $mse(l_2) < mse(l_1)$
{ if at any point $(mse(l_2) > mse(l_1))$ then we going again far from best fit line

* Shape of cost function of n line will always be a parabola.

for the best fit line we have got an algorithm called as convergence algorithm.

- Convergence Algorithm
  - keep moving new best fit line in direction of all data point until error (mse / CF) is reduced as compared to previous line
  - Repeat until convergence

$$\begin{cases} m_{new} = m_{old} - \eta \dfrac{d(CF)}{d\, m_{old}} \\ c_{new} = c_{old} - \eta \dfrac{d(CF)}{d\, c_{old}} \end{cases}$$

η → learning rate (eta)

or

Repeat until convergence (simultaneously)

$$\theta_j := \theta_j - \eta \frac{d J(\theta_j)}{d \theta_j}$$

$J = \theta_1$

$\theta_1 = \theta_1 - \eta \dfrac{d J_{(\theta_1)}}{d \theta_1}$

$\theta_0 = \theta_0 - \eta \dfrac{d J_{(\theta_0)}}{d J_0}$

m = slope = $\dfrac{y_2 - y_1}{x_2 - x_1}$ (change in y / change in x)

↓ for linear line

Here $\dfrac{dy}{dx} \approx \dfrac{\partial y}{\partial x}$ (partial derivative) is slope for curve

$\dfrac{dy}{dx} = \dfrac{\Delta y}{\Delta x}$

$$\frac{dx^n}{dx} = nx^{(n-1)}$$

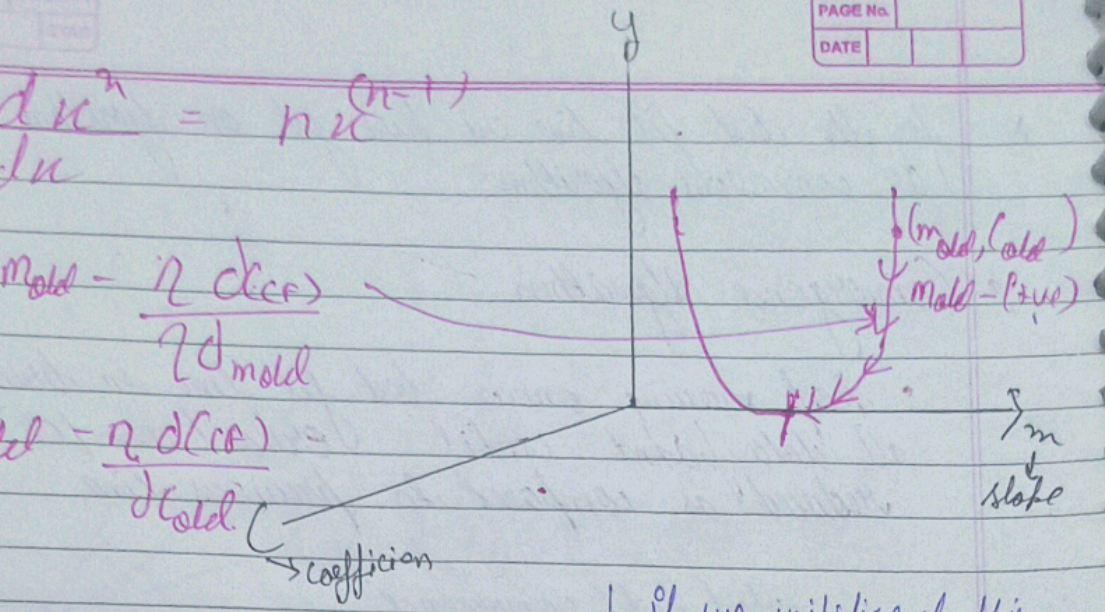
$$m_{new} = m_{old} - \frac{\eta \, d(cf)}{d m_{old}}$$

$$c_{new} = c_{old} - \frac{\eta \, d(cf)}{d c_{old}}$$

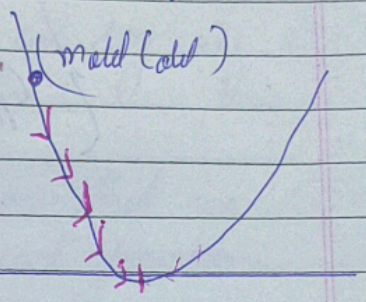
This Conversion Algo also called as gradient diescent

Lets say if it is $\Rightarrow m_{(old)} - (-value)$

$\Rightarrow m_{old} + value$



If we initilize at this point?



Note → About Slope

Slope (-ve)

Slope (+ve)

when slope is -ve

-ve m, c

θ_0, θ_1

is increasing

Slop is +ve

when (m, c)

is getting reduced

θ_0, θ_1

$m_{new} << m_{old}$

$$m_{new} = m_{old} - \eta \frac{\partial(CF)}{\partial m_{old}}$$

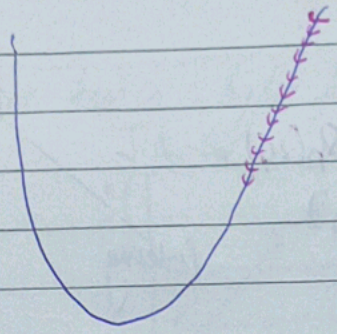
$$C_{new} = C_{old} - \eta \frac{\partial(CF)}{\partial C_{old}}$$

η is learning rate

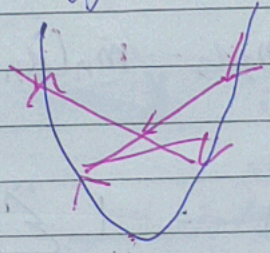
↓

decides convergence speed

if η is very small → takes very very high time to convert



if η is bigger (very, very) → overshoot the minima & keep escalating.



→ That's why η should be ~~minimum~~ (0 to 1) not to large not too

Repeat until convergence {

$$\theta_j := \theta_j - \eta \frac{\partial J(\theta_j)}{\partial \theta_j}$$ deij

→ also called gradient descent.

}

Multiple Linear Regression

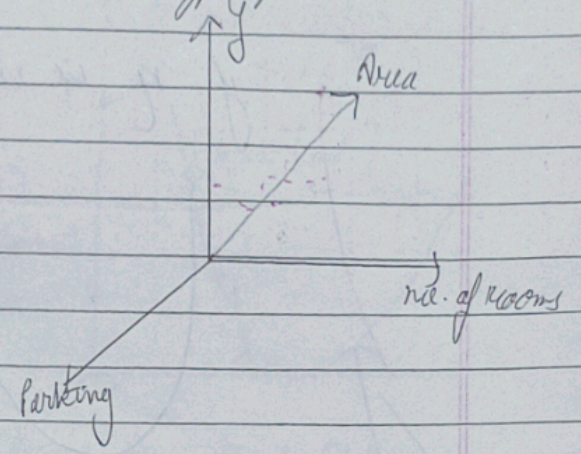
Only on feature can not be enough to predict target variable.

eg→ There are multiple independent variables that contribute in predicting target variable and this is called as multiple linear regression

graph of MLR looks like a Hyper plane

Data set

No of Rooms	Area	Parking	Price



$H_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \cdots \theta_n (x_n)$

$(\theta_0, \theta_1, \theta_2 \cdots \theta_n)$

optimal coefficient.

$y_{pred} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \cdots \beta_n (x_n)$

$l = c + m_1 x_1 + m_2 x_2 + m_3 x_3 \cdots m_n (x_n)$

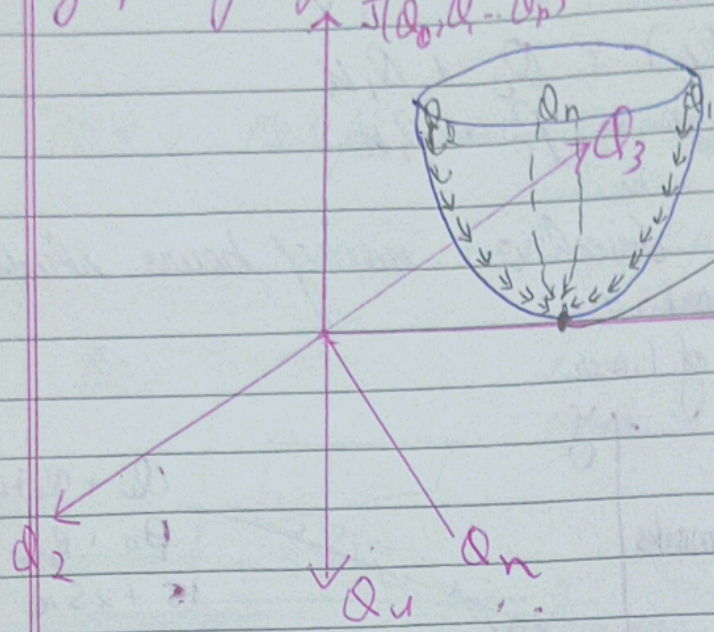
Cost fun of MLR

$$J(\theta_0, \theta_1, \theta_2 \cdots \theta_n) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2$$

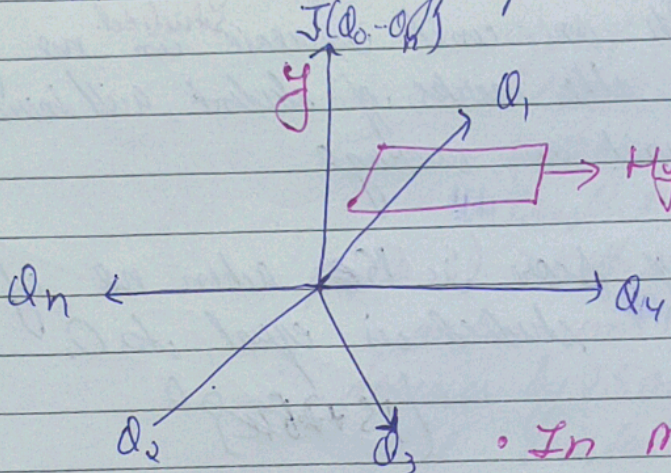
$$(\theta_0 + \theta_1 x_1 + \theta_2 (x_2) \cdots \theta_n x_n)$$

minimize the eq using gradient decent to get optimal coeff $(\theta_0, \theta_1, \theta_2 \cdots \theta_n)$

graph of gradient decent of MLR



in MLR we have a hyper plane (multidimensional plane)



so what abt the intercept (c)?

- In MLR the intercept is point where hyperplane intersect the vertical axis (y)

target variable

Interpretation of slope coefficient

SLR → $H_\theta(x) = \theta_0 + \theta_1 x$

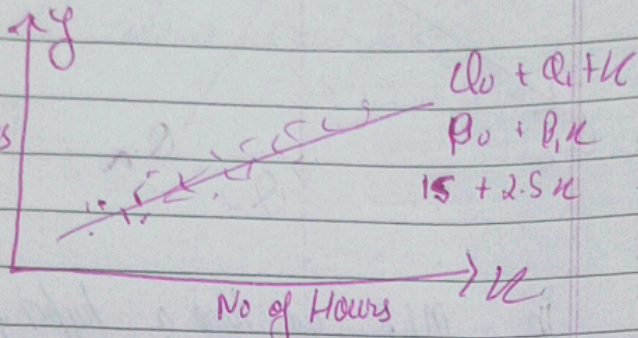
$y_{pred} = \beta_0 + \beta_1 x$

Marks of student based on no. of hours studied.

y = marks

x = no of hours

$H_\theta(x) = \beta_0 + \beta(x)$ with 1 hours studied



Interpretation → with one unit increase in no of hours studied, the marks of student will increase by 2.5 unit on average

β_1

β_0 → Average score is 15 when no of hours studied is equal to 0.

$(15 + 2.5x)^0$

→ But this interpretation is not always true

for eg → House price → $20 + 1.6x$
based on area

↓

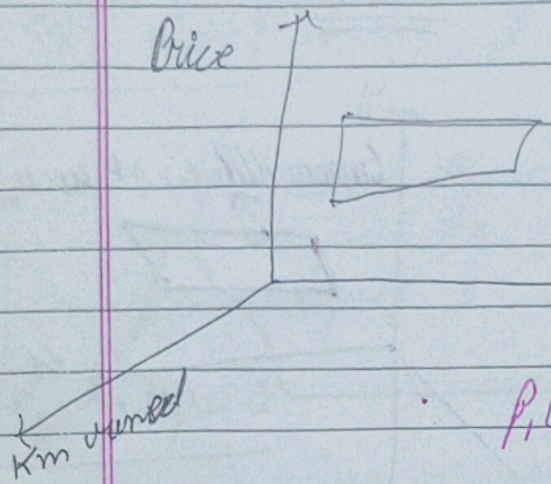
Hi There if β_0 = 20 lac on average when area is 0,

but is area is 0 How house price is 20 lac ?

The correct interpretation → There are multiple interpretation like this is on average price etc.

MLR → Multiple linear regression

$$y_0(u) = \beta_0 + \beta_1 u_1 + \beta_2 u_2 \cdots \beta_n u_n$$



$y_0(u)$ = 3 − 0.5 * Age of car + 2.5 * Km runned

$\beta_1(0.5)$ → with 1 unit increase in age of car, selling price decrease by (0.5) on an average keeping β_2 (Km runs) constant

$\beta_2(2.5)$ → with a 1 unit increase in ~~at~~ km runs (β_2) selling price decreases by 2.5 on an average keeping age constant

β_0 → on average selling price is 3 cr keeping all factors constant.

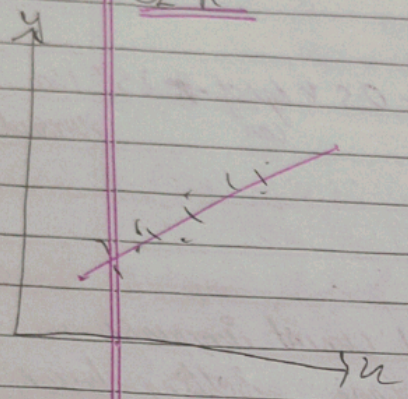
Polynomial ~~Linear~~ Regression →

Till now

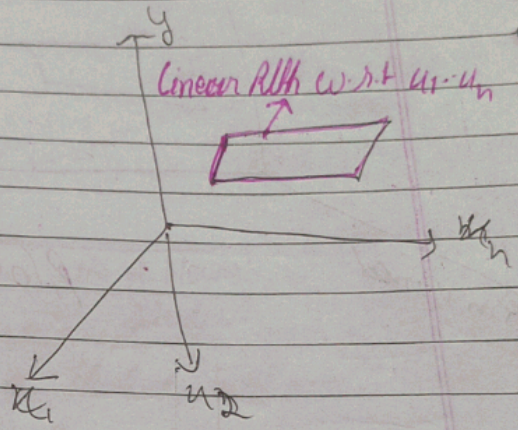
SLR = $h_\theta(x) = \beta_0 + \beta_1 x$

MLR = $h_\theta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \cdots \beta_n (x_n)$

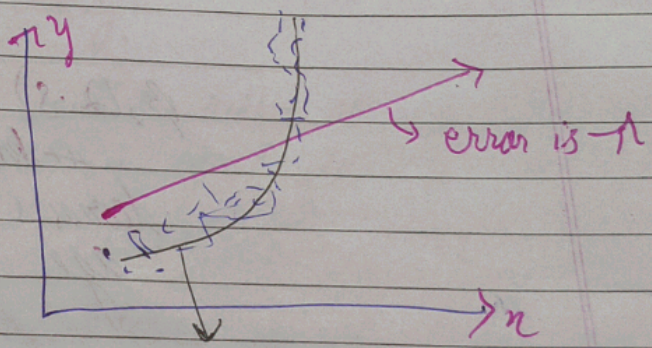
SLR



MLR



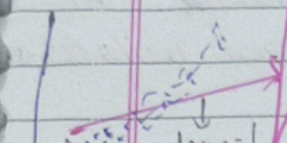
if data is like



~~try~~ to make non linear best fit line

Polynomial Regression

1 → Simple Polynomial Regression → (1 Dep. Variable
1 Indep. Variable)



polynomial degree 0 : $y_0(u) = a_0 = a_0 \times 1 = a_0 u_1^0$

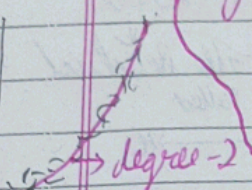
polynomial degree 1 : $y(u) = a_0 u^0 + a_1 u_1^{(1)}$

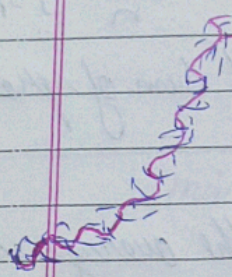
$= a_0 + a_1 u_1$

↳ SLR

polynomial degree 2 : $y_0(u) = a_0 u^0 + a_1 u^1 + a_2 u^2$

degree 3 : $y_0(u) = a_0 u^0 + a_1 u^1 + a_2 u^2 + a_3 u^3$





Note → as degree ↑ you might get overfitting model as it tries to connect all points

2 → Multiple polynomial Regression (n → indep variable

u_1	u_2	u_3	y

polynomial degree 2 → $y_i(u) = a_0 + a_1 u_1 + a_2 u_2 + a_3 u_3 + a_4 u_1^2 + a_5 u_2^2 + a_6 u_3^2 + a_7 u_1 u_2 + a_8 u_2 u_3 + a_9 u_3 u_1$

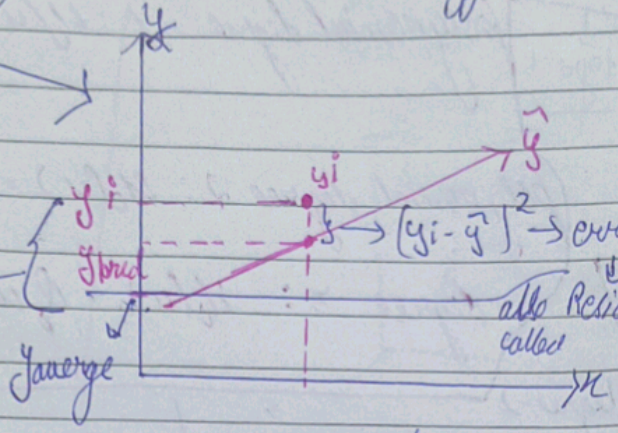
- Original I.V & power 2 ($u_1^2 + u_2^2 + u_3^2$)
  + Cross product for all ($a_1 u_2$, $u_2 u_3$, $u_3 u_1$)

Till now we have learnt ~~it~~ about SLR, MLR, PR but how we interpret our model performs good?

for that now we have evaluation matrix → to know performance we have different methods :-

1 → R^2

2 → Adjusted R^2



* error / RSS / $\frac{S.S.E}{n}$ → MSE

$\sum_{i=1}^{n}$ error

Residual Sum of squares

- why $(\bar{y})$ $y_{average}$?

$\bar{y}$ → if you don't build any model, the average value is the baseline solution.

eg → whats the rent of room in your neighbourhood.

- SSR (Sum of Square due to regression) ⇒ explained variation in y by best fit line

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

- S.S.E / R.S.S → unexplained variation

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

NOTE :- R^2 closer to 1 → model is close to perfect

R^2 closer to 0 → model is not good.

- TSS (Total Sum of Squared error) → Total variation in y / diff in y w.r.t to baseline ($\bar{y}$)

TSS = Explained Variation + Unexplained error
SSR RSS / S.S.E

~~RSS~~ $R^2 = 1 - \frac{RSS}{T.S.S}$ or ~~RSS~~ $\frac{SSR}{TSS}$

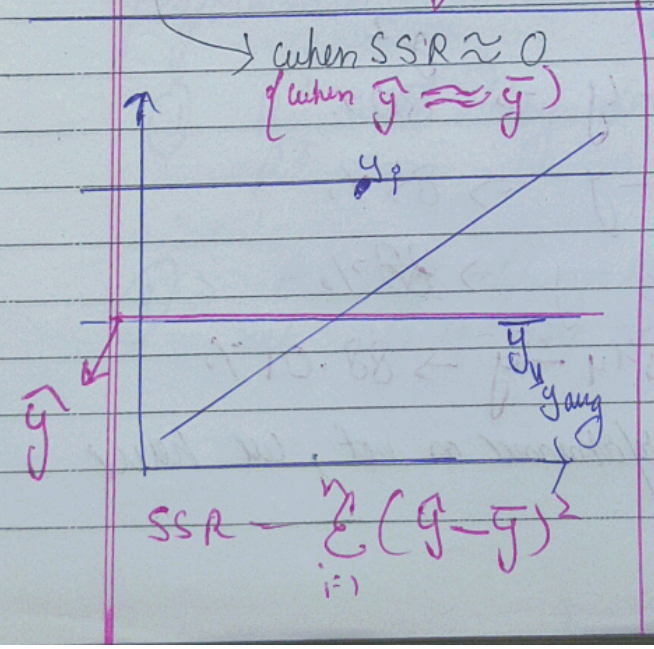
R square → coefficient of determination → out of total variation, R^2 is the variation which is explained.

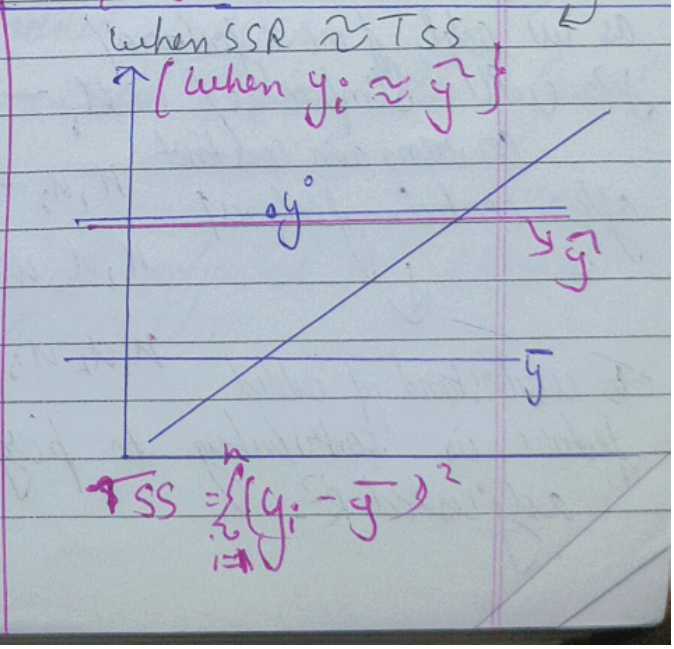
or → Total percentage of variation explained by model

$$R^2 = \frac{SSR}{TSS} \times 100$$

min value of $R^2 = 0$ when $\left(\frac{SSR}{TSS} \text{ & } SSR \approx 0\right)$

max value of $R^2 = 1$ when $\left(\frac{SSR}{TSS} \text{ & } SSR \approx TSS\right)$





Can R^2 can be negative

↓

yes → when → if best fit line is very far from $\bar{y}$ (baseline model)

↓

which is not possible

↓

So R^2 can be negative but it is not possible due to nature of algorithm

2 → Adjusted R^2

$\rightarrow$ % explained variance in y due to x

x_1 area of house	x_2 No of room	x_3 city distance from house	y price of house

as we add more feature r^2 will improve/ or remain constant after certain features

r^2

$x_1 - y \Rightarrow 80\%$

$x_1 x_2 - y \Rightarrow 85\%$

$x_1 x_2 x_3 - y \Rightarrow 88\%$

$x_1 x_2 x_3 x_4 - y \Rightarrow 88.01\%$

↓

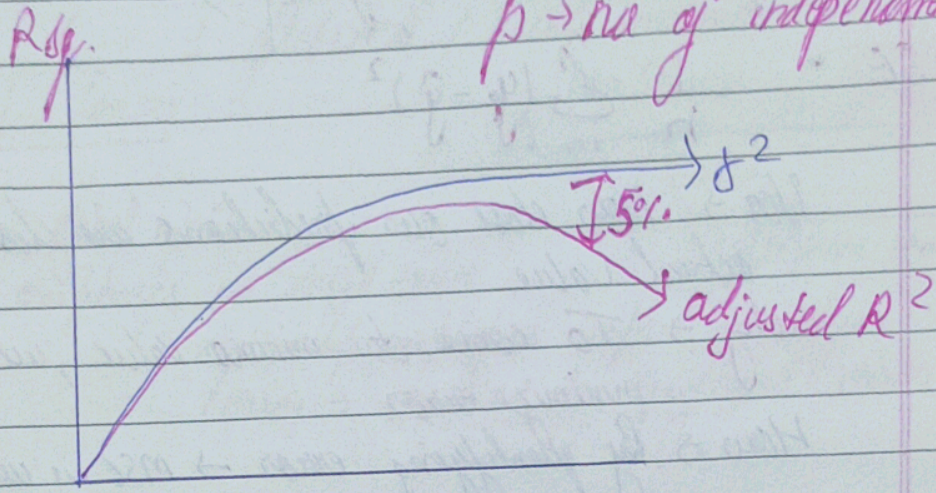
To understand if added feature is contributing to performance or not, we have adjusted R^2.

To calculate {adjusted R^2} we have a formula

$$1 - \frac{(1 - R\text{square})(N-1)}{(N-P-1)}$$

$n \rightarrow$ no of data points

$p \rightarrow$ no of independent variables



→ at some point R^2 becomes constant but when the R^2 → gets constant → adjusted R^2 start decreasing.

if difference b/w (R^2 & adjusted R^2) that means it is not contributing much to performance. ↳ that added feature

① R^2 will always be ~~lesser~~ greater than adjusted R^2.

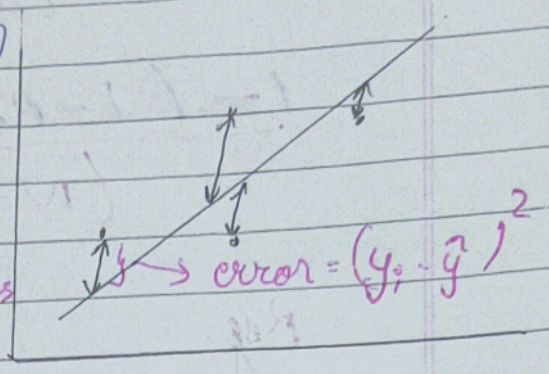
② → ~~R^2~~ Diff. b/w R^2 & adjusted R^2 should not be more than 5%.

- MSE (Mean Squared Error)

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

Mean squared error = $\frac{\text{Sum of error}}{n}$

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



Idea → How close our predictions are close to actual value.

Why → To come to minima value, we need to minimize error

How → By quantifying error → MSE is used here

- ~~error~~ Lower MSE, Higher performance

MSE measure average of squared error b/w predicted and actual values.

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

Why Squaring difference?

→ it ensure the value do not cancel out each other. $((-3)^2 + (+3)^2 = 9)$

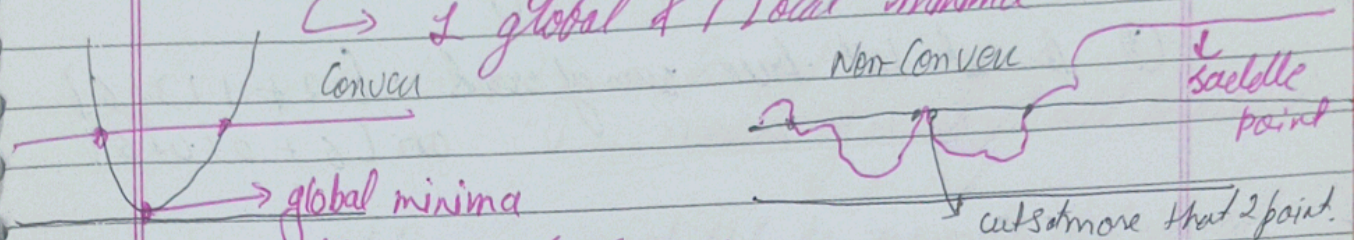
$(-3 + 3 = 0)$

↓

Not error — x ?

Advantages of MSE

- Error don't cancel out each other
- MSE is used as CF
- MSE is differentiable $\theta_j - \eta \frac{\partial CF}{\partial \theta_j}$
- used as gradient descent as CF
- it is convex function
  - ↳ 1 global & 1 local minima



- emphasis on large errors
  - ↳ sq → amplifies impact of larger error → sensitive to significant error

Disadvantage

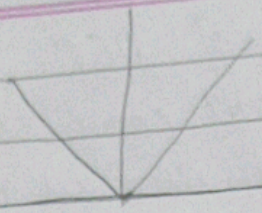
- Not robust to outlier $(y_{act} - y_{pred})^2$ ↑↑
- Not in same unit.
  - ↳ unit's squared.

(2) Mean Absolute error → measure absolute difference b/w actual & predicted value.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}|$$

Advantage →

- does not squares the unit
- less sensitive to outlier
- more interpretable → easy cal



• Disadvantage

① Convergence usually takes more time, optimizing is complex

② Time Consuming, Model training

③ Do not tell true sum of errors [2+1+3+6] or [6+0+0+6]

④ RMSE Root mean sq. error

$\sqrt{MSE}$

Advantage:-

- units not changes
- differentiable
- less sensitive to outlier
- magnifies larger error

Disadvantage

- interpretation → complex

Assumptions of linear Regression

1 -> linearity -> linear relation b/w (x & y)

2 -> independence -> Rows (observation) should be independent.

3 -> Homoscedasticity -> Constant variance
- variance of error should be constant

4 -> Normality -> error should be normally distributed.

5 -> ~~Re~~ features should not related or should have least relation -> multicollinearity.

M.L Pipeline

1. Data ingestion / Reading dataset
2. EDA
3. Data preparation
4. Divide data into dependent & independent (y & x)
5. Train-test split
6. Scale the data (optional)
7. Model Train
8. Model evaluation

Pickling of model

pickle → allows you to store python object on a disk

Python pickle module is used for searializing and de-searializing a python object structure.

Searializing ⇒ it is a process of converting data structure or object into the format that can easily stored, transmitted or persisted. During searlization, objects state converted into a stream of bytes or other that can be stored on the disk, sent over a network or store in memory

de-searializing → it is a process of reconstructing a searlized object back to the python object or what ever its primary form was.

The searlized object/ data is read and converted back into a data structure or object that matches original

Multi-Collinearity

in stat → correlation is How two values are correlated, like when x → Yes, y increases / decrease by some -% chance this is the interpretation by this [correlation]

$x \rightarrow y$ by $\approx 94\%$

if we have multiple features $x_1, x_2, x_3, x_4 \cdots x_n$

→ Here one feature can correlated to several features

Corr b/w two (Indep. Vs) at a time

$Corr(x_1, x_2) = 95$

$Corr(x_2, x_5) = 80$

• Two points

• if x has high correlation with y → the model will be good → good thing

1 → • what will happen if feature among themselves has good correlation?

2 → • To measure relationship b/w two variable/features we have correlation. but how to measure correlation among multiple features.

1) Correlation among features itself →

if $corr(x_1, x_2) \approx 95\%$.

↳ by this in terms of variation we can say $x_1 \approx x_2$

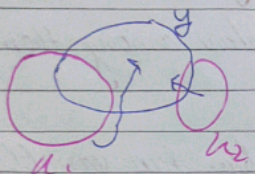
if $x_1 = x_2$ } → The interpretation of MLR becomes tough (which feature is contributing to y will be tough to decide as x_1 & x_2 are same)

(2) Measure of correlation among multiple features

• for that we have ~~multiple~~ multicolinearity.

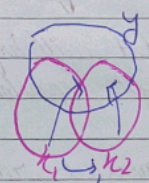
many together linear relation

multicolinearity → When a feature exhibits a linear relationship with more than two variables.



both x_1, x_2 are corr. with y

so no multicolinearity



↳ multicolinearity.

Why multicolinearity ?

Why it is a concern

→ affects interpolation

→ increase computation

→ it increases overfit

x_2 → defining same

overfitting increases

Solution →

• To measure mult

inflation factor

with high

• RFE (Recursive

VIF → is a measure

VIF

I

Why multicolinearity ?

Why it is a concern?

→ affects interpolation $x_1 = x_2$

$\beta_1 x_1 + \beta_2 x_2 \rightarrow y$

x_1 or x_2?

→ increase computation if $x_1 = x_2$ then why to compute/use both (x_1, x_2) as our feature

→ it increases overfitting if alone x_1 can explain y then why ~~but~~ when we use x_2 → defining same then it memorise and chances of overfitting increases

Solution →

- To measure multicolinearity we have VIF (Variation inflation pattern) and drop feature one by one with high VIF.
- RFE (Recursive Feature Elimination)

x_1 - x (correlation/heatmap)

VIF → is a measure of amount of multicollinearity in regression.

$x_1 (x_2 x_3 x_4 x_5)$

$$VIF_i = \frac{1}{1 - R_i^2}$$ { R^2 → % Variation in y explained by y }

dependent → $x_1 \rightarrow y$

Independent → $x_2 x_3 x_4 x_5 \rightarrow x$

x_1 x_2 x_3

y x

R^2 = % of variation explained on y by x.

Case I ⇒ $x_1 \approx (x_2, x_3)$ → VIF_{x_1} ⇒ $\frac{x_1}{y}$ & $\frac{x_2 x_3}{x}$

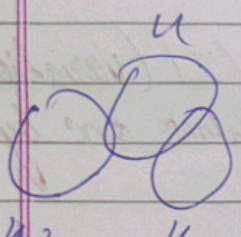
$$VIF_{x_1} = \frac{1}{1 - R_{x_1}^2}$$

Case II ⇒ $x_2 \approx (x_1, x_3)$ = $VIF_{x_2} = \frac{1}{1 - R_{x_2}^2}$

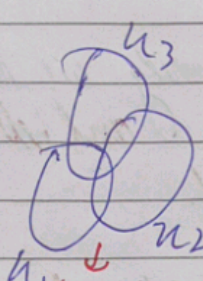
Case III ⇒ $x_3 \approx (x_2, x_1)$ = $VIF_{x_3} = \frac{1}{1 - R_{x_3}^2}$

We don't need to train this on model by ourself

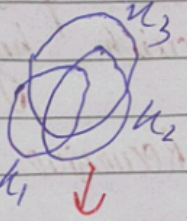
Their will be a Variation Inflation Factors there we just pass all features it will give us all possible cases



No Multicolinearity



Low Multical.



High Multicolinearity

Note when VIF > 10, → High Multicolinearity

↓

Higher VIF → Drop the feature one by one

if VIF > 10 → High → $VIF = \frac{1}{1 - R_i^2} \Rightarrow 10 = \frac{1}{1 - R_i^2}$

$10 - 10R_i^2 = 1 \Rightarrow R_i^2 = \frac{10 - 1}{10}$

$$R_i^2 = \frac{10-1}{10} = \frac{9}{10} = 0.90$$

Kinds of multicolinearity

There are two kinds of multicolinearity.

1→ Data-Based Colinearity → present in data itself due to its nature

- eg → Latitude, Longitude

2→ Structured based multicolinearity → caused due to new feature from existing feature

eg → Time | Distance | Speed = $\frac{D}{T}$

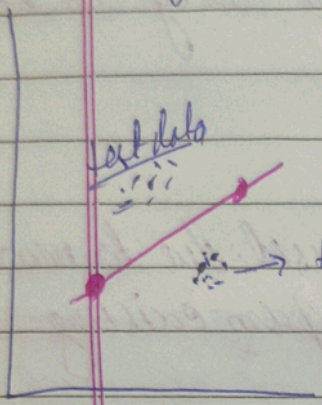
* We talked that using VIF drop feature one by one what if there are 1000 feature

→ There comes RFE → Recursive Feature Elimination
→ it will make model with all 1000 feature
→ Start dropping the feature which has VIF > 10
→ one by one
→ Does it Recursively until desired no. of features is achived.

Regularization

↓ To add something to reduce ~~anything~~ overfitting

To regularise / To penalise



Data

train (Acc ↑) (low bias) (low train error)

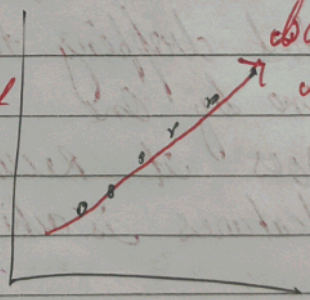
test (Acc ↓) (High var) (High test error)

overfit

Since best fit line passes through both data point, so the model is overfitted model

→ overfitted model means model is performing well on train data but not on test data

Analogy behind this



→ lets say this is our best fit line for our ~~no~~ train ~~to~~ data

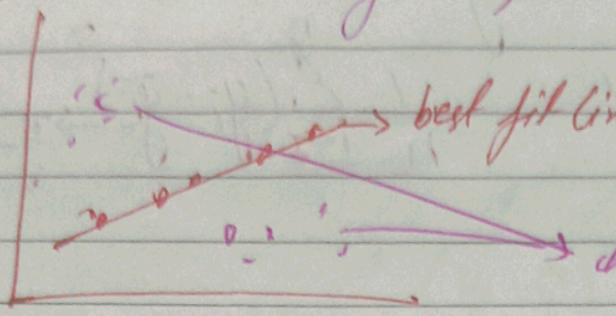
→ we got best fit line using gradient descent

→ Here our test accuracy is 100%.
↳ as our data points lies on the best fit line

- This is case of overfitting.

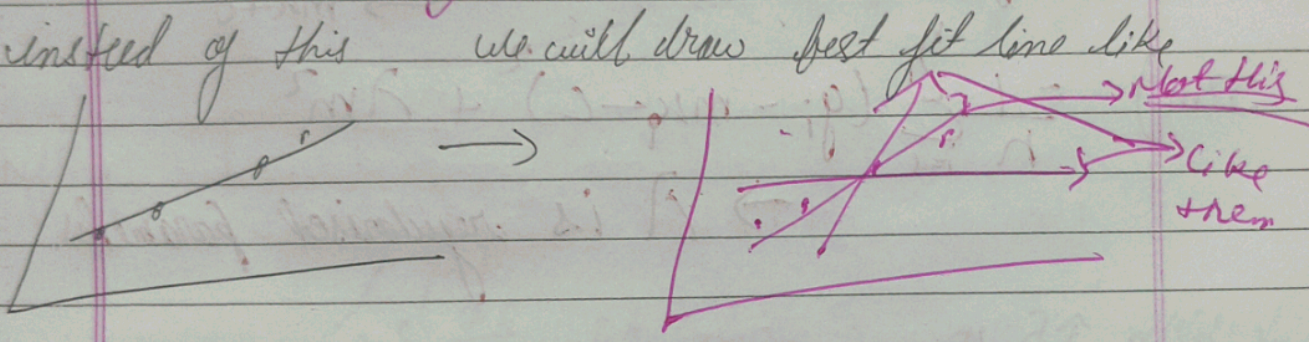
, because in train data its good, but in test data.

data points were not in best fit line.



Sol^n • So in order to make sure our model performs well on ~~test data~~ data (unseen) as well, we will introduce some error/penalize while training model itself.

instead of this we will draw best fit line like



→ While training we introduced some small error in order to increase the test accuracy. Because of this concept of regularization came into picture.

Regularization → To add something to reduce overfitting, that something is error.

Standard cost function

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \quad \bar{y} \to y_{pred}$$

$y_{pred} \to mx + c$

$\theta_1 x + \theta_0$

How to introduce error in this?

Regularised cost function / Regularised linear model

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_{act} - y_{pred})^2 + \lambda m^2$$

$y_{pred} \to mx + c$

$$J(m) = \frac{1}{n}\sum_{i=1}^{n}(y_i - mx_i - c)^2 + \lambda m^2$$

$\to \lambda$ is regularised parameter.

if error = 0 as accuracy = 100% for ~~of~~ overfitted model

$$\sum_{n}(y_i - y_{pre})^2 = 0$$

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pre})^2 + \lambda\theta^2$$

$(y_i - y_{pre})^2 \to 0$

$$J(\theta) \Rightarrow 0 + \lambda\theta^2$$

This error is introduced here

Different types of regularization

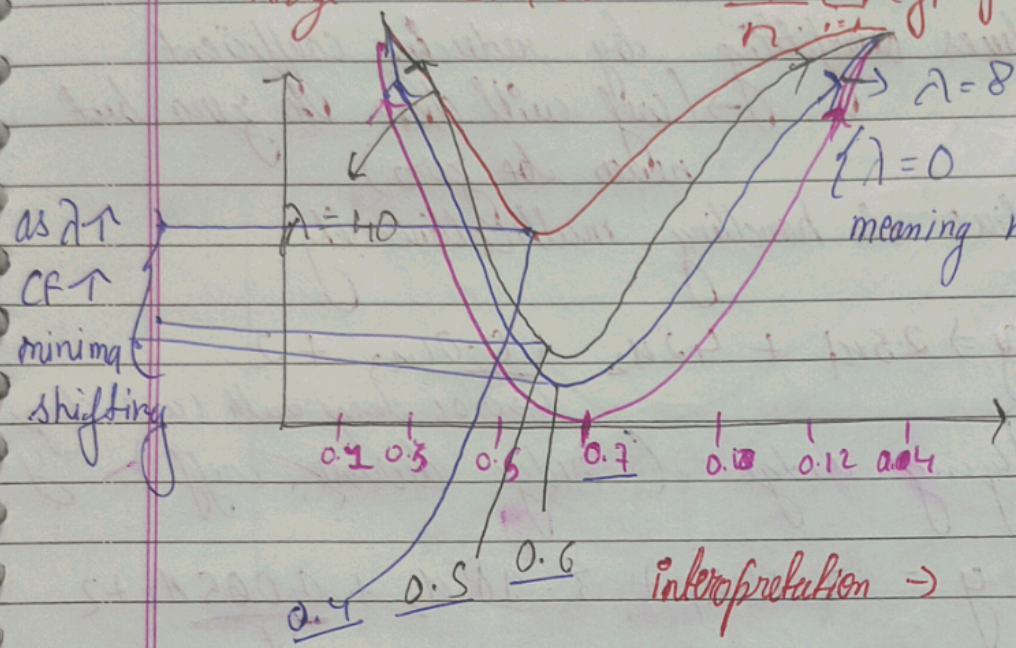
1 → Ridge Regression (also called as L2 Penalty, L2 Norm or Tikhonov regularization)

In SLR, $\frac{J(\theta)}{CF} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$ → pow-2 that's why L2 regularization

Ridge = $CF(J\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 + \lambda\theta^2$

↳ slope

↳ hyperparameter



interpretation → as λ↑, m/slope ↓

Note → Slope (m) decreases but never become zero in Ridge

SLR → $H_\theta(x) = \theta_0 + \theta_1 x$

$CF = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 + \lambda\sum_{j=1}^{p}\theta_j^2$

$(y_i - \hat{y})^2 \hookrightarrow H_\theta(x)$

n ↳ no. of data points p → no of predictor variables

Note → The intercept θ_0 is not included in find term, Regularization applies only on coeffi $\theta_1, \theta_2, \theta_3 \ldots \theta_p$

In MLR

$$H_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$CF = \frac{1}{n}\sum_{i=1}^{n}(y_i - H_\theta(x_i))^2 + \lambda(\theta_1^2 + \theta_2^2 + \theta_3^2)$$

Ridge advantages →

→ Reduces overfitting by reducing coefficient
↳ (coef will close to zero but never be zero)

→ efficient in handling multicollinearity

eg) $y \Rightarrow 2.3x_1 + 4.2x_2 + 0.01x_3 + 2$
↳ 0.01 change with 1 unit change in y

after applying ridge (ridge decrease coeff)

$$y = 2.1x_1 + 3.9x_2 + 0.005x_3 + 2$$

↓
effect reduced

Disadvantage

→ doesnot make less imp feature coefficient 0 as we need ~~not~~ no effect of some irrelevent feature on prediction

* Lasso Regression (L1 Regularization, L1 Norm) or feature selection

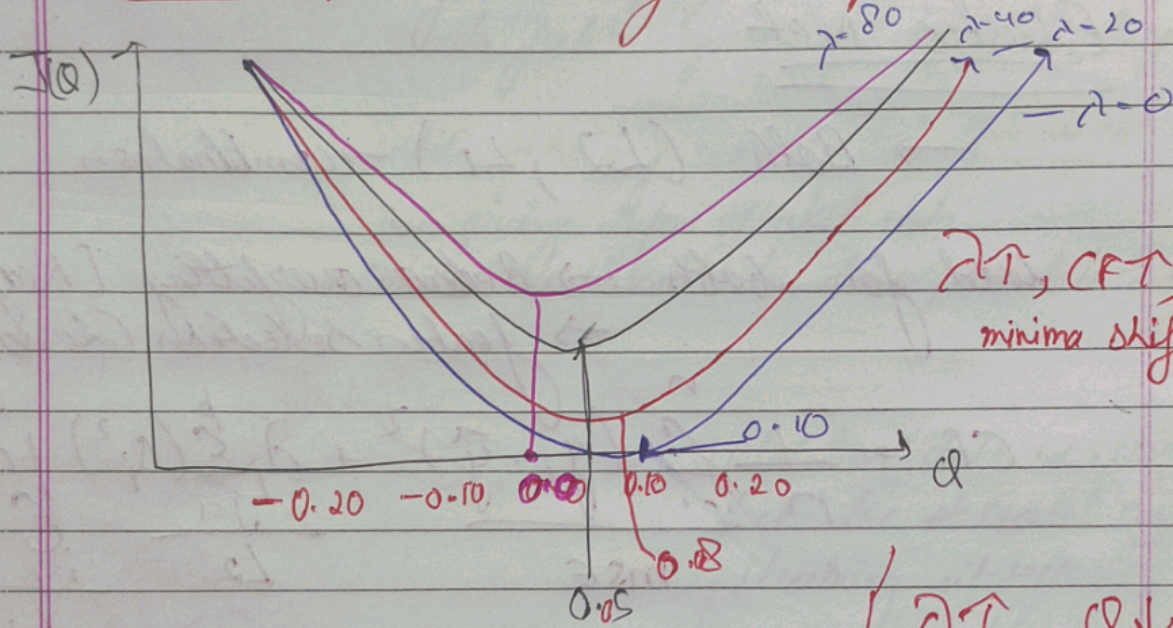
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \lambda \sum_{j=1}^{p} |\theta_j|^1$$

(Here power is 1 that's why it is L1 → Regularization)

$$\Downarrow$$

$$\text{error} + \lambda(|\theta_1| + |\theta_2| \ldots |\theta_p|)$$

→ similar to ridge (intercept is excluded)



λ↑, CF↑, minima shift

(λ↑, θ↓)

Here θ coefficient decrease with increase in λ, and it sometimes reduced to 0 as well

↳ irrelevant features of self becomes 0 ~~in~~, that means no impact of feature in prediction

Advantage of Lasso

→ in order to remove irrelevent features it is very useful (Sparsity)

Lasso → push coefficient towards 0/exact 0

→ multicolinearity & overfitting reduced.

3 Elastic Net

→ Both (L2, L1) → combination

used for both → Reduce overfitting (Ridge)
→ feature selection (Lasso)

$$CF = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 + \lambda_1 \sum_{j=1}^{b}(\theta_j^2) + \lambda_2 \sum_{i=1}^{b}|\theta_j|$$

MSE | L2 | L1

λ_1, λ_2 → Hyper parameter
range → $[0, \infty]$

Why we using all these things.

Occam's Razor Principle → Simpler models are best model as compare to complex

↓

Regularization does it

Cross Validation, its types and hyperparameter tuning

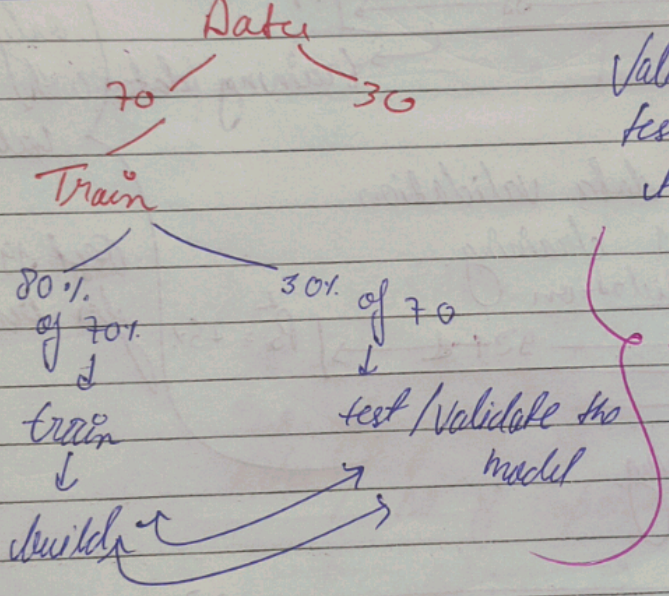
Data → 70 → Train Data
Data → 30 → Test Data → Representative of unseen data

Scenario → You trained a model, got 80% accuracy on train data but 40% on test data. What will you do?

→ is you will go again to training process and do feature engineering and repeat all things?

while retraining we ← No this is not allowed
are giving info of test data
which should not be done, test data → unseen data.

⇒ Do this



Validation data is used to test the ~~of~~ model while training itself

why?
→ you don't want data to leak
→ when we don't fix random state, each time we get diff R^2. To be confident we take average of all runs

Cross Validation → experimenting with different arrangement of same data to build different model of same algorithm

train data	Exp 1	train	Validate	Accuracy 1
	Exp 2	train	"	" 2
	Exp 3	train	"	" 3
	Exp 4	"	"	" 4

Mean of (1, 2, 3, 4)

Types of Cross Validation

(1) Leave One Out Cross Validation (LOOCV)

in train → 100 obs

Exp 1 → [1st obs | 99] → $R_1^2 = 80\%$

1st obs → validation

99 → training data

Here only one data will be for validation

1st data validation
rest 99 training.

Exp 2 → [1 | 2nd data | 98+1] → $R_2^2 = 75\%$

1 → validation

98+1 → training

rest 99 will for training

Exp 100 → [← test 99 → | 100] → $R_{100}^2 = 77\%$

$$\text{avg } R^2 \Rightarrow \frac{1}{n} \sum_{i=1}^{n} (R_i^2)$$

Disadvantage of LOOCV

→ Since only 1 df is for test, train size become very huge for large data, and time increased accordingly.

→ overfitting increases as → training size ≈ total size

(2) Leave p out cross validation { p > 1 }

Exp-1 [5 | ← 95 →]

Exp-2 [— | 5 | 90+5]

Exp $\binom{n}{p}$ [→ | 5]

(3) K-Fold Cross Validation → very famous in industry

k = 5
↓ no. of group you want to divide the data

no of experiment = $\frac{100}{20}$ = 5

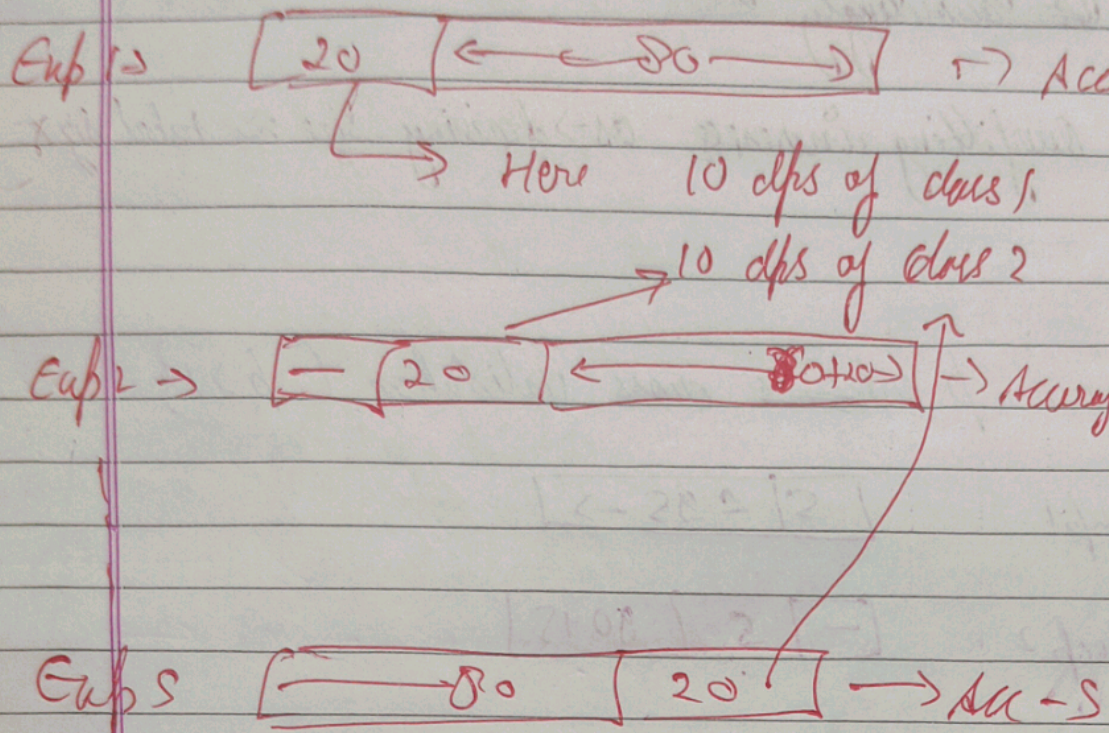
Exp-1 [20 | ← 80 →] → Rsq 1

Exp-2 [← 0 → | 20 | ← 60+20 →] → Rsq 2

Exp-5 [← 80 → | 20] → Rsq 3

4-> Stratified K-fold cross validation.

k = 5 , n = 100 class 0 - 60
class 1 - 40



In each validation set, there will be equal representation of each of the class.

* Hyperparameter Tuning

Hyperparameter → external config. of model that are not learnd from data but are set prior to training process.

$$\text{Ridge} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 + \frac{\lambda}{2m}m^2$$

$y = mx + c$ learned from data.

λ hyperparamete ↓ external.

model performance is heavily influenced by (λ) ↳ Hyperparameter

$\lambda = 0, 1, 2, \ldots 100$ ↓

How to know best value ~~for~~ for λ from all possibilities.

Hyperparameter with cross validation
↳ finding best hyperparameter while ~~training~~ training model.

1-> Grid Search (CV) -> cross validation.
2 -> Random Search CV

1. Grid Search

$\lambda = 1, 2, 5, 10, 15$

$p = p_1, p_2, p_3$

All possible combi. of λ, p

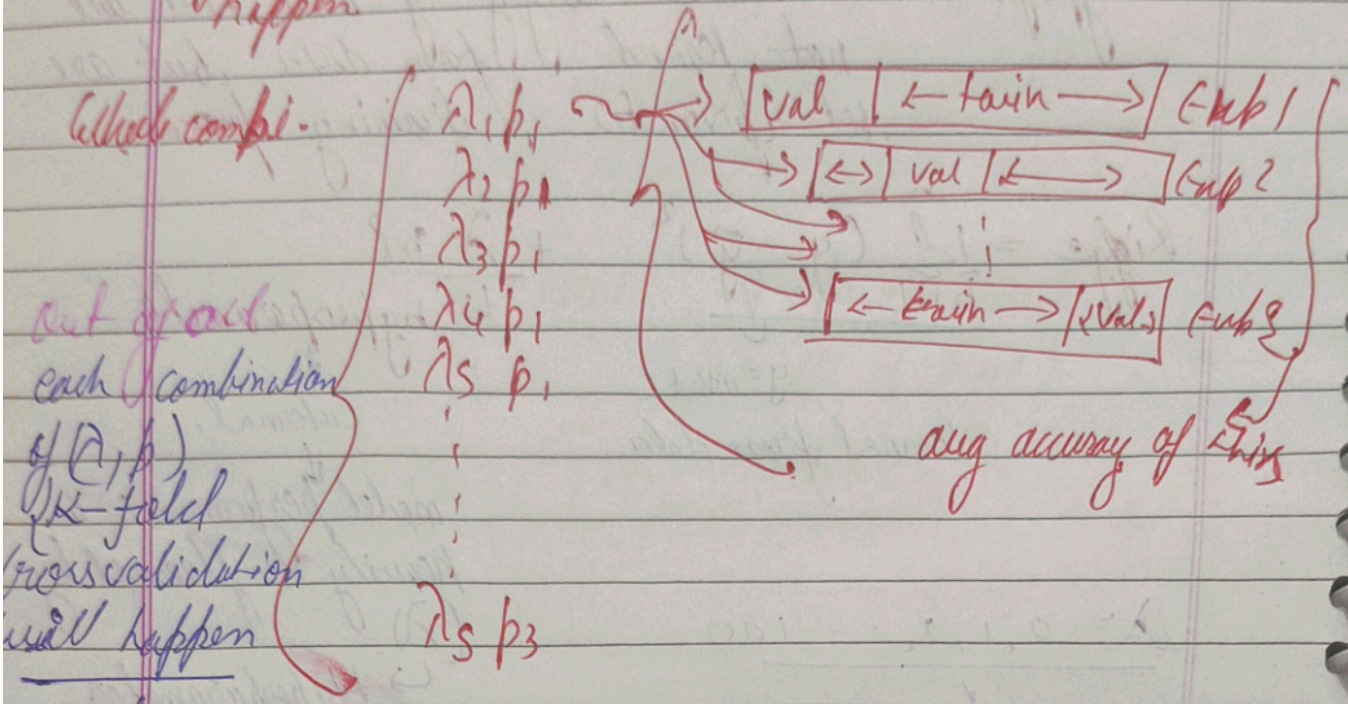
	λ_1	λ_2	λ_3	λ_4	λ_5
p_1	$1p_1$	$2p_1$	$5p_1$	$10p_1$	$15p_1$
p_2				$10p_2$	
p_3		$2p_3$			$15p_3$

Best model → combination of (p, λ) which gives max accuracy.

NOTE → if Data is small instead of train-test use k-fold cross validation

Hyperparameter tuning with cross-validation

for each (λ, p) internally cross validation will happen



↳ best accuracy will use for hyperparameter

Total no. of model. → p × λ × 5

Time Complexity is very high.

2 → Randomized Search CV.

↳ we will not see all combi.

instead we have n_iter. = 5

↓

select random n combi of (λ, p)