Machine learning

(main)

- Supervised ML
  - Regression
    - linear
    - Ridge & lasso
    - polynomial Reg.
    - Multi linear
    - Non-linear
    - OLS
    - Time series forecasting.
  - classification
    - logistic
    - KNN
    - DT
    - Naive Byes
    - RF
    - XGB
    - GBM
    - SVM
    - ANN
    - catboost, adaboost
- Unsupervised ML
  - clustering
    - kmeans
    - DBScan
    - hierarchical
  - apriori
  - Recommdati
    - Content
    - Memory
      - User based
      - item based
  - Dimentionality Reduction
    - PCA
    - tSNE
    - ICA
    - SVD
    - Auto Encoding.

Supervised ML →

We will start with Regression → linear

linear Regression

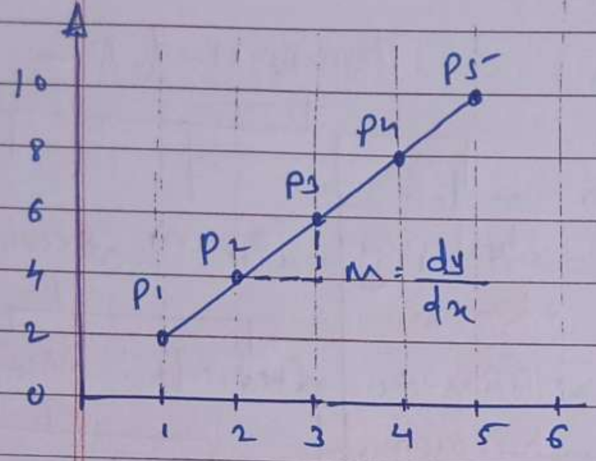
- Simple linear Regression (SLR)
- Multiple linear Regression (MLR)
- polynomial Regression

- Simple linear Regression (SLR) :- when for Continuous output we have only one feature then it is called SLR

for Ex.

CGPA	package
6.0	2 LPA
7.2	3 LPA
8.9	6 LPA.

Basics of linier Regression.



Equation of straight line

$$y = mx + c$$

find value of m and c

$P_2\ (2, 4) \longrightarrow x_1, y_1$

$P_3\ (3, 6) \longrightarrow x_2, y_2$

$$m = \text{slope} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-4}{3-2}$$

$$m. = 4/1 = 4 = 2$$

slope means with every change in value of x how much differs in y

Intercept = c

$P_4\ (x_3\ y_3) \longrightarrow (4, 8)$

$$y = mx + c$$

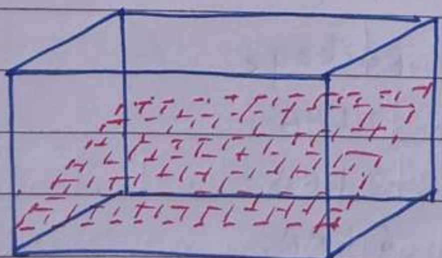
$$8 = 2(4) + c$$

$$8 - 8 = c = 0$$

Inference : The above line eq^n is function that relate x and y.
for given value of x we can find corresponding value of y.

what if we have two or more than two indepedent variable ?

→ then it is called multiple linear regression



(3D)

simple linear regression : $y = mx + c$

multiple linear regression

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n + b.$$

Advantages: 1) simple to implement

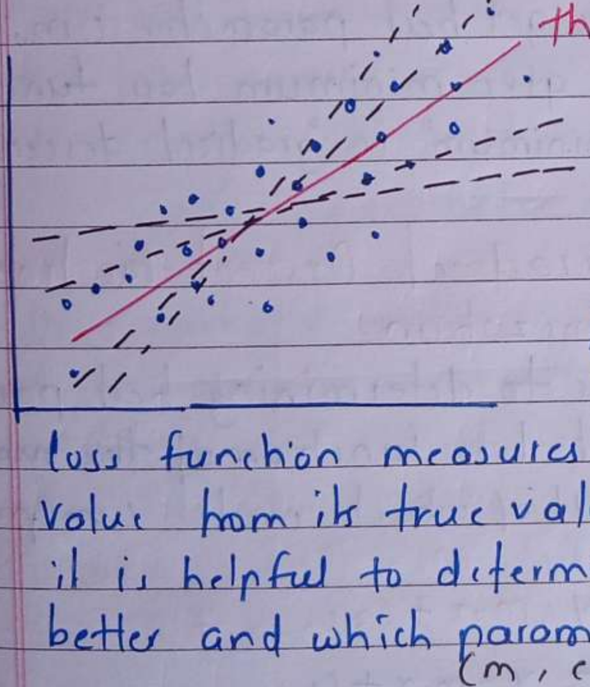
2). perform well on the data with linear relationship.

Disadvantages: 1) Not suitable for data having non-linear relationship.

2) underfitting issue

3) sensitive to outlier

Mathematical Understanding :-



so here purpose is to find best fitted line (model) for linear regression.

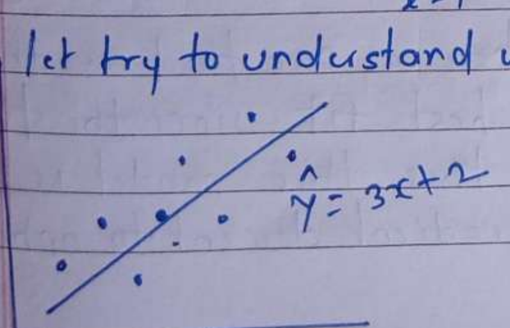
{ this where loss function comes in picture }

- loss function measures how far an estimated value from its true value.
- it is helpful to determine which model performs better and which parameters are better.

(m, c)

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

let try to understand with the ex. $m = 3$, $c = 2$



x	y	$\hat{y}$
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20.

Now find its loss function.

$$loss = \frac{[(10-8)^2+(14-11)^2+(18-14)^2+(22-17)^2+(26-20)^2]}{5}$$

$$= \frac{4+9+16+25+36}{5} = \frac{90}{5} = 18$$

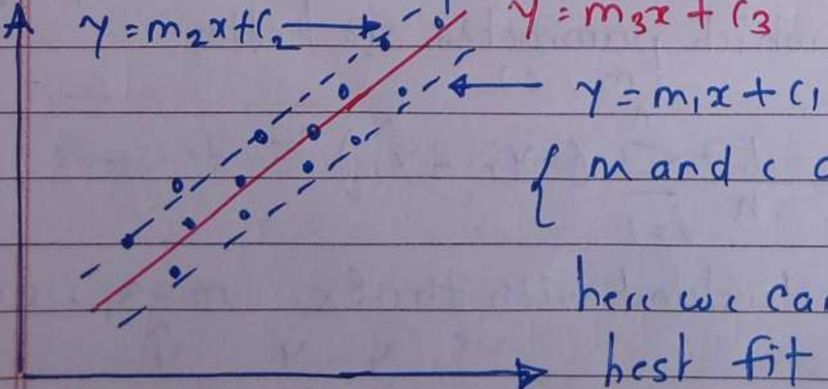
{ reason to take squar:. so that positive and negative value may not cancel each other }

low loss value → High Accuracy
high loss value → low Accuracy

We can improve the model by some optimization technique called as "gradient descent" where repeate the process iteratively untill we get best parameter (m, c) for which model will give minimum loss function. Called as 'global minimum" in "gradient descent"

How we can use gradient Descent for linear regression for optimization.

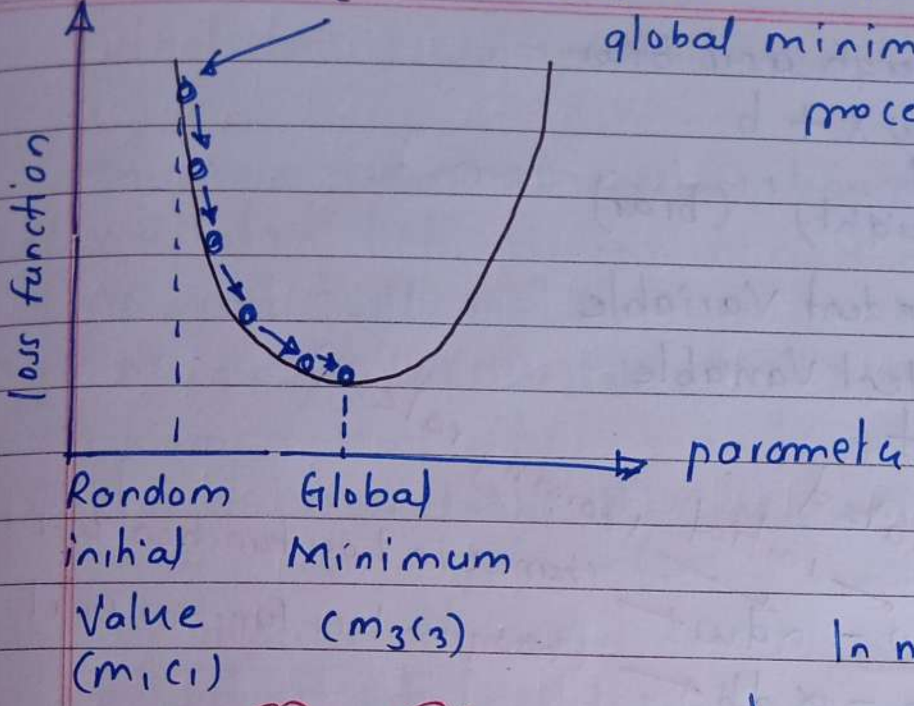
→ optimization refers to determining best parameter for model such that loss function of the model decreases as result of which model can predict more accurately.



{ m and c are the parameters }

here we can find model 3 is best fit since the loss function is least and thus this model is optimum this where we used gradient descent to optimize model.

{this will step down untill reach down to global minimum by iterative process}



In machine learning

updated → $m = m_1 - LD_m$ | $w = w - L\,dw$

$c = c_1 - LD_c$ — initial | $b = b - L\,db$

m - slope

c - Intercept

w = weight

b = bias.

L - learning rate : it is magnitude of change that you want in parameter during iteration.

Dm : partial derevative of loss function w.r.t m

Dc : partial derevative of loss function w.r.t c.

$$D_m = \frac{\partial(\text{loss function})}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)$$

$$\boxed{= \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \hat{y}_i).}$$

Ily $$D_c = \frac{\partial(\text{loss function})}{\cancel{\partial m}\ \partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)$$

$$\boxed{= \frac{-2}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)}$$

Interms weigh and bias

$$y = \underset{\substack{\uparrow\\(\text{weight})}}{w}x + \underset{(\text{bias})}{b}$$

x - independent Variable

y - dependent Variable

w - weight

b - bias

$$\overset{\text{updated}}{w} = \overset{\text{initial}}{w} - \overset{\text{learning rate}(\alpha)}{\alpha}\,dw \leftarrow \text{change in Loss function w.r.t } w$$

$$b = b - \alpha\, db \leftarrow \text{change in loss function w.r.t. } b.$$

learning rate:- it is tuning porometer in an optimiza on algorithm that determine step size at each iteratio while moving toword minimum of loss function.

$$dw = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \hat{y}_i)$$

$$db = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i).$$

for Multiple linear Regression: for one target we have two or more than two independent variables.

prediction Equation for MLR

$$\hat{y} = a_1x_1 + a_2x_2 + \cdots a_nx_n + b.$$

and,

Regression Equation

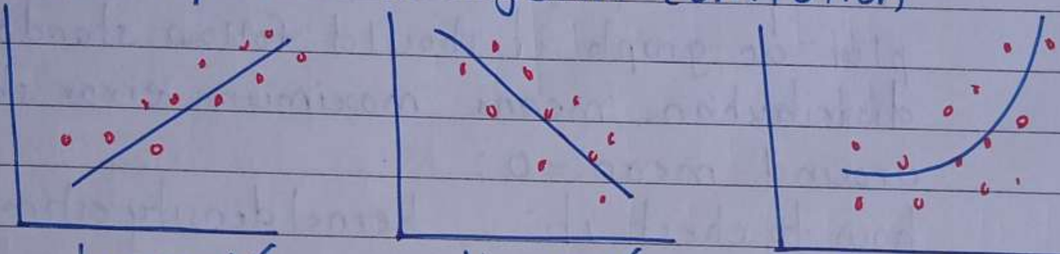
$$\underset{\substack{\uparrow\\\text{Actual}\\\text{Value}}}{y} = \underbrace{a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n + b}_{\text{predicted value}} + \underset{\substack{\uparrow\\\text{Error}}}{\varepsilon}$$

What are assumptions of linear Regression?

there are five main assumption in linear regression.

1. linear Relation between input and output.
2. No multicolinearity.
3. Normality of Residual.
4. Homoscedasticity.
5. No autocorrelation of error

1. **Assumption 1** :- there should be linear relation between individual feature and target (output) it could be positive or negative correlation



linear ✓ linear ✓ Non-linear (X)

applicable to multiple linear problem as well. Not ok.
how to check this : scatter plot : (feature 1 Vs target) (feature 2 Vs target).

2. **Assumption 2** :- Multicollinearity :- it mean there all the feature should be independent or should not have any correlation among themselves.

why, what the problem?

In multi linear Regression model for 3D we draw a hyperplane.

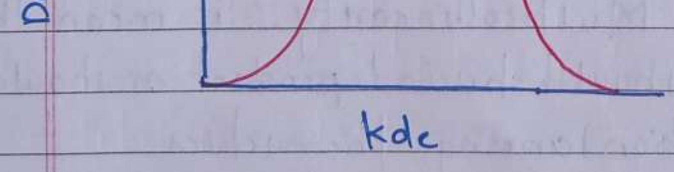
$$y = a_1x_1 + a_2x_2 + a_3x_3 + b.$$

where a_1 represent what will be the change in y with respect to x_1 assuming x_2 and x_3 constant. but if it violate this assumption then model will not perform good. (Ex of two physics scientist) how to check multicollinearity :-

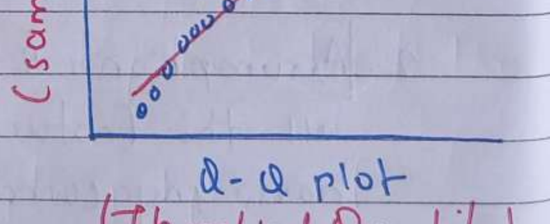
1) VIF (Variance Inflation factor) :-
if it is around 1 then feature don't have the issue and if it is 5 or more than that then that perticular feature have multicollinearity issue and need to remove it

2). another method is to find out correlation between all the features (Heatmap)

3. **Assumption 3.** - Normality of Residual.
it says that when error (actual - predicted) plot or graph it should follow standard normal distribution. means maximum error should around mean = 0.
how to check it : kernel density estimation (kde)
or Q-Q plot.



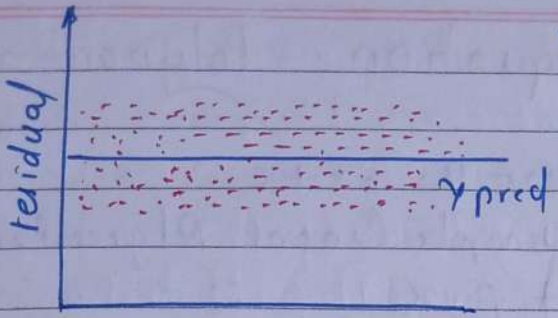
kde



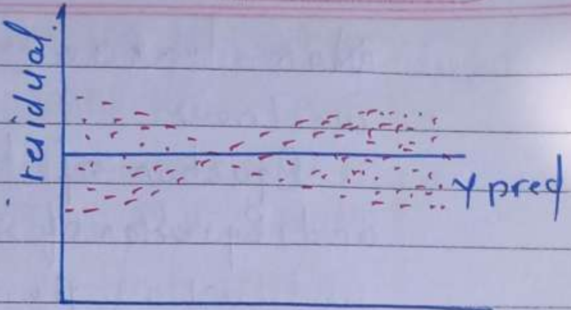
Q-Q plot
(Therotical Quantile)

4. **Assumption 4** : Homoscedasticity.
$\underbrace{\text{Homo}}_{\text{same}}\underbrace{\text{scedasticity}}_{\text{scattered / spred}}$

: spread of residual should be equally or uniforlly scattered. if it is not it called as heteroscedastisity which is not desirable.



(desirable)

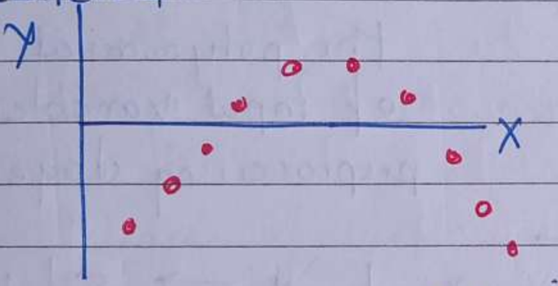


(Not desirable)

how to check:- scatter plot (ypred, residual).

Assumption 5: No autocorrelation of Error.

if we plot the error there should not be any specific pattern instead it should randomly scattered.



positive Autocorrelation
(Not desirable)

Negative Autocorrelation
(desirable).

how to check: pll. plot (residual)

Summary chart

	Assumption	Severity	prediction	Inference
1)	linear Relation	High	✓	✓
2)	Multicollinearity.	Medium	x	✓
3)	Normality.	low	x	✓
4)	Homoscedasticity	High	✓	✓
5)	Autocorrelation of Error.	–	–	–

Non-linear Equation - Polynomial Regression

we know

Equation of line $y = mx + c$

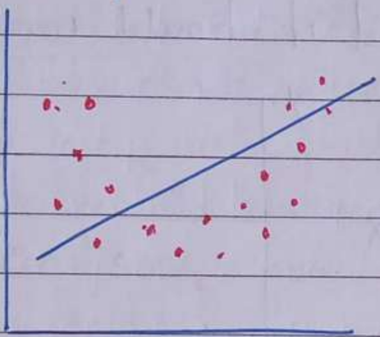
and Equation of simple Linear Regression

$$y = \beta_0 + \beta_1 x$$

and for multiple linear Equation.

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

this is applicable only when data is linear but what if data is not linear ?



In such scenario we extract the polynomial features of input variable in preprocessing stage

let say for ex $x \mid y \rightarrow 5 \mid 10$

- for x we want to make polynomial of degree 2 then we will convert $x \rightarrow x^0, x^1, x^2, y$
  so it will be 1, 5, 25
- this way we create a new data for training the extra polynomial feature try to extract their non-linear relationship.
- its formula become for simple polynomial regression.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

For degree 3

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$

★ Now how we will know the perfect value for the degree _ since this is hyperparameter.

if we keep it low then may be it cause underfitting means it may be not able to learn the all attributes. and if we select very high then there is chance of overfitting or overlearned that's why our job is to find out optimum value

- In case if we have two features x_1, x_2, Y then for degree 2 our simple polynomial Equation would become

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^4$$

Interview Que: why polynomial Eqn essentially called as Linear Regression

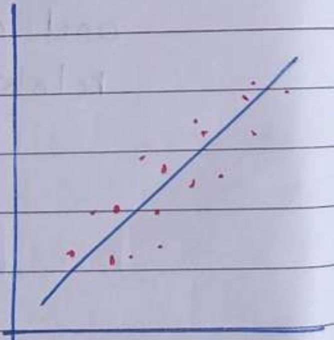
Ans: when we talk about linear Regression we talk about relation between y and coefficients of features and degree of coefficient is still one and thus relation between y and coefficient is still linear

Ordinary least square :- (OLS Algorithm)

- it is method for estimating the parameters of linear regression model.
- it aim to find the values of the linear regression models parameter (i.e coefficient) that minimize the sum of squared residual.
- the residuals are differences between observed values of dependent variable and predicted values of dependent variable given w.r.t independent variable
- OLS Algoritham assumes that the errors are normally distributed with zero mean and constant variance and that there is no multicollinearity (high correlation) among the independent variables.
- other method like generalized least square or weighted least square, should be used in case where these assumption are not meet.

let understand with problem

x	1	2	3	4	5	6	7
y	1.5	3.8	6.7	9.0	11.2	13.6	16.



we will calculate the equation for the best fit line where all the point will be as close as possible by least square method.

	x	y	xy	x^2
	1	1.5	1.5	1
	2	3.8	7.6	4
	3	6.7	20.1	9
	4	9.0	36	16
	5	11.2	56	25
	6	13.6	81.6	36
	7	16	112	49.
Σ	28	61.8	314.8	140.

$\Sigma x = 28$ $\Sigma y = 61.8$

$\Sigma xy = 314.8$ $\Sigma x^2 = 140$

$n = 7$

(number of data points)

$$\boxed{y = mx + b}$$

$$M = \frac{n\sum xy - \sum x \sum y.}{n\sum x^2 - (\sum x)^2} : \frac{7(314.8)-(28)(61.8)}{7(140)-(28)^2}$$

$m = \frac{473.2}{196}$ = ~~2t~~ 2.4142857.

$$b = \frac{\sum y - m\sum x}{n} = \frac{61.8 - 2.4142857(28)}{7}$$

$b = -0.828571.$

to get linca. equation weshowd plug value. in

$y = mx + b.$

~~y = 2.41x *~~ $y = \hat{2}.41x - 0.83.$

let test it. for 2	$\hat{y}$	Yact
$y = 2.41(2) - 0.83 =$	3.99	3.8
$y = 2.41(5) - 0.83 =$	11.22	11.2
$y = 2.41(7) - 0.83 =$	16.09	16.

Syntax: statsmodel.api.OLS (y, x).

y: depedent Variable   x - indepedent Variable.

-
  ```
  Import statsmodel-api as sm
  import pandas as pd.
  ```

-
  ```
  #reading data from csv
  df = pd.read-csv('train.csv')
  ```

-
  ```
  # defining the variables
  x = df['x'].tolist()
  y = df['Y'].tolist()
  ```

```
# adding thi constont term x
  x = sm. add_constant (x).

# performming regression ond fitting model.
result = sm. OLS (y, x) . fit ()

# print summory tabl,
print ( resull. summary () ).
```

What is Ridge Regression. (L2 Regularization).

Ridge regression is model tuning method. that is used to analyse any data that suffers from multicollinearity This method performs L2 regularization when issue of multicollinearity occurs, least square are unbiased. and variance are large. this result in predicted values being for away from actual values.

Regularization:- it is technique used to calibrate Machine learning model to minimize adjusted loss function and avoid overfitting and underfitting.

there are three types of regularization techniques:

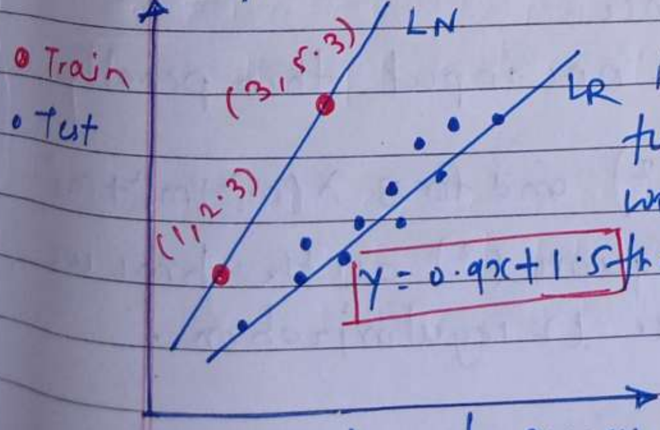
1) Ridge Regression (L2 regularization)
2) Lasso Regression (L1 regularization).
3) Elastic Net (combo of Ridge and Lasso)

- Ridge Regression:-

overfitting:- means your train acc is too high but test accuracy is very low.

$y = mx + b$

m or slope which is coefficient of x is defining change in y, so to reduce overfitting means to reduce slope.



if we train the model only on two datapoint then best fit line would be the line which pass through both point

$y = 1.5x + 0.8$

Now we have to convey our model to choose LR and Not LN

$$L = \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{} + \underbrace{\lambda(m^2)}_{} \quad ①$$

before we were only reducing this term which is nothing but Error / Residual

Now we add penalty which will add help for the first term

λ = hyperparameter

m = slope.

Now we will calculate loss for both line for $\lambda = 1$ with eqn ①

Since line is passing through both point perfectly that's why 1st term will be zero

$$L = 0 + 1(1.5)^2$$

$$\boxed{= 2.25}$$

$$(2.3 - 0.9 - 1.5)^2 + (5.3 - 2.7 - 1.5)^2 + (0.9)^2$$

$$= (0.1)^2 + (1.1)^2 + (0.9)^2$$

$$= \boxed{2.03}$$

here we getting significant reduction in loss for the new line

As our model can see this change it will select 2nd model although it will give bad accuracy on training since Variance is significantly reduced although bias increased.

why we called L2 since

- if we have more than one input then penalty would be

$$\lambda(m_1^2 + m_2^2) \text{ and for 3 } \lambda(m_1^2 + m_2^2 + m_3^2)$$

Since we are doing square $(^2)$ all the time we called it L2 Norm or L2 regularization.

Lasso Regression (L1 Regularization)

- this is also help to reduce overfitting.
- In Ridge regression we have seen:

$$L = \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_1)^2}_{\text{MSE}} + \underbrace{\lambda ||w||^2}_{\text{penalty term}}$$

$$\underbrace{(w_1^2 + w_2^2 + \cdots w_n^2)}_{\text{coefficient in MLR}}$$

Lasso is just another variation of Ridge

$$L = \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_1)^2}_{} + \lambda ||w||$$

OR

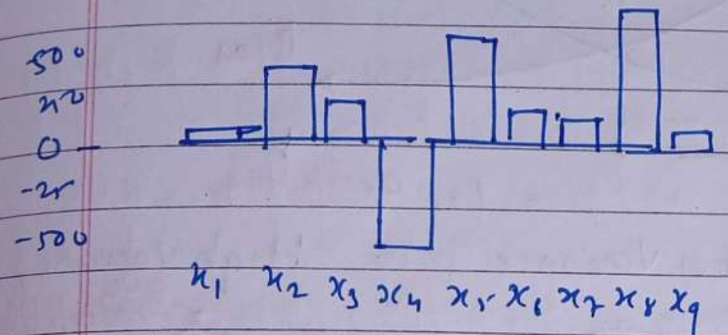
$$= \text{MSE} + \lambda[|w_1| + |w_2| + |w_3| + \cdots + |w_n|]$$

- In ridge regression for any value of λ there were always some value for coefficient of input feature but in lasso if you continuously increase value of λ at certain point you will zero value for some of coefficient which are not important so here we unknowing doing feature selection and it is advantage of Lasso.
- so when you are working on high dimensional data and some feature are not imp we should prefer Lasso over Ridge.

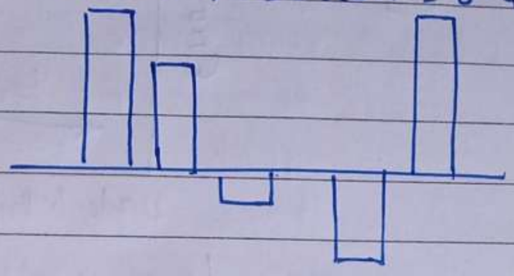
there are some keypoints need to discuss about Lasso.

1) How coefficient are affected. ?

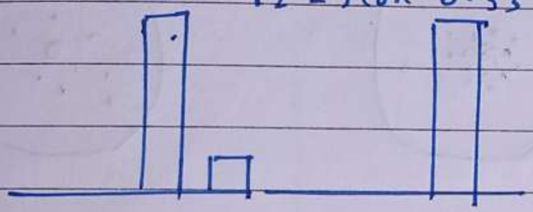
$\lambda = 0$, r_2_score = 0.44



$\lambda = 0.1$, r_2_score = 0.43



$\lambda = 1$ r_2 - score 0.33



$\lambda = 10$ r_2-score - 0.01

{ All coefficient will be zero }

2. Higher coefficient are affected more

- generally as you increased λ coefficient value will be decrease gradually toward zero
- usually higher coefficients are affected first/rapidly penalize
- for it we should be cautious for selecting optimum value to do proper feature selection

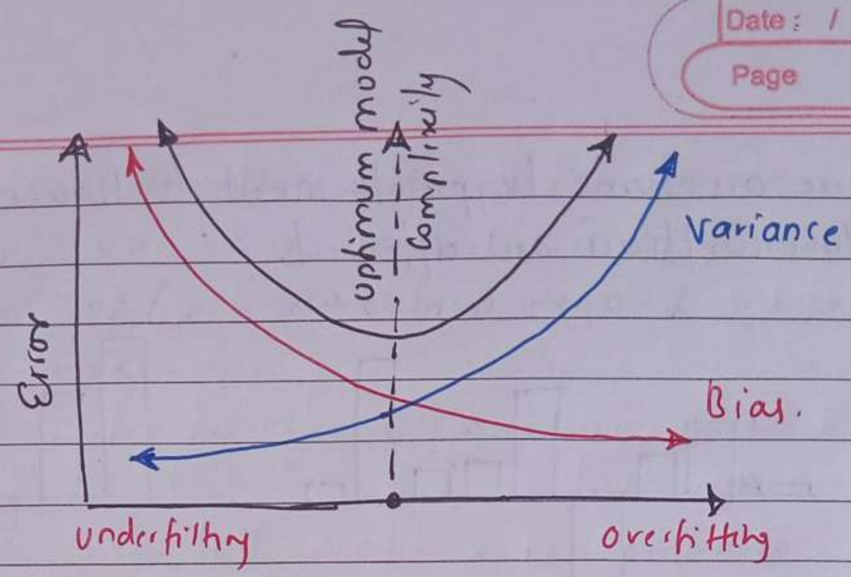
3. Impact on Bias and Variance

- As we know if λ increases, overfitting decreases (↓) which lead to increase in bias (↑)

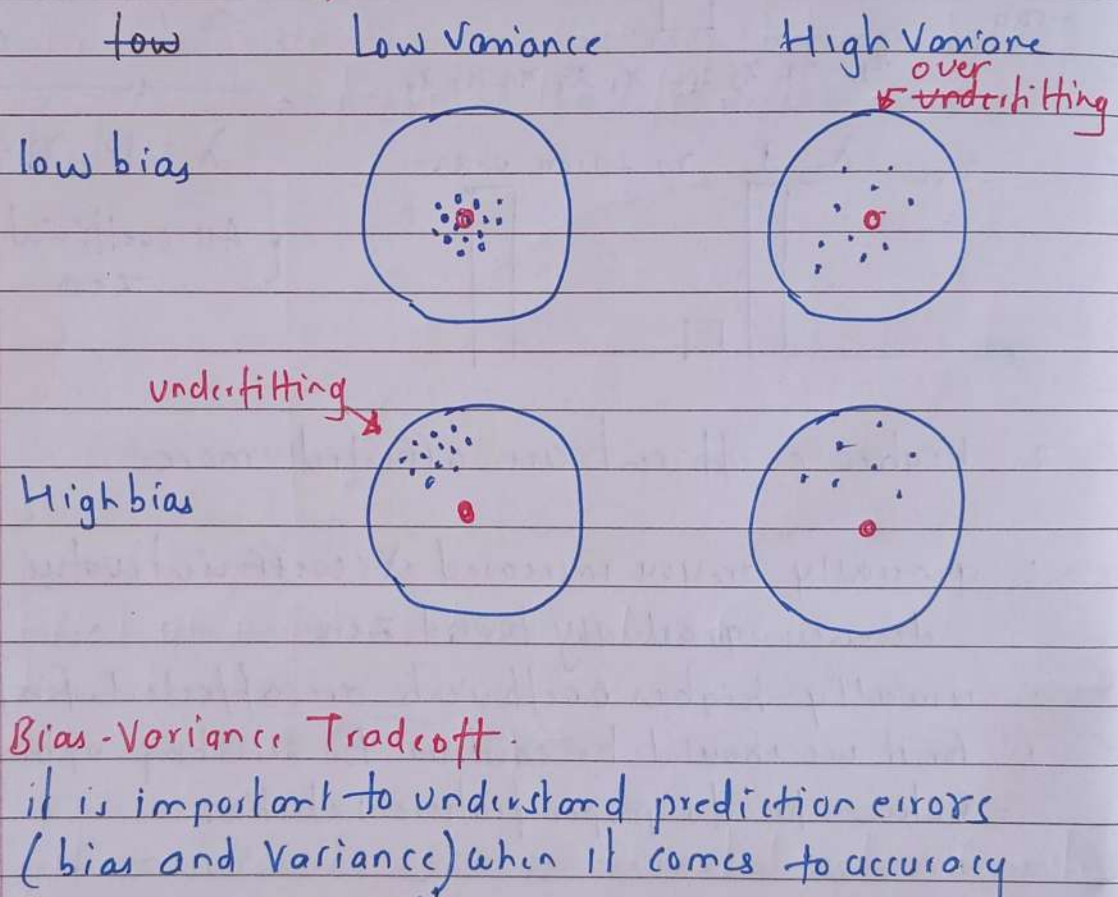
High Bias → Underfitting

High Variance → overfitting

fig a.



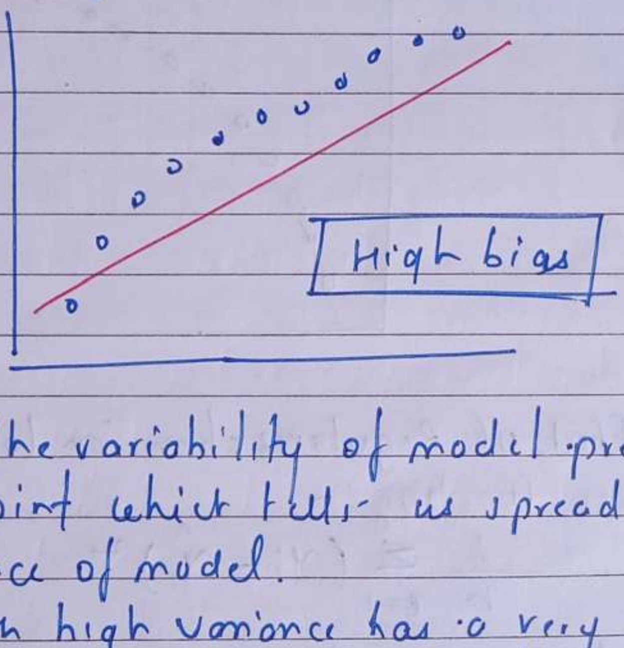
fig(B



Bias-Variance Tradeoff.

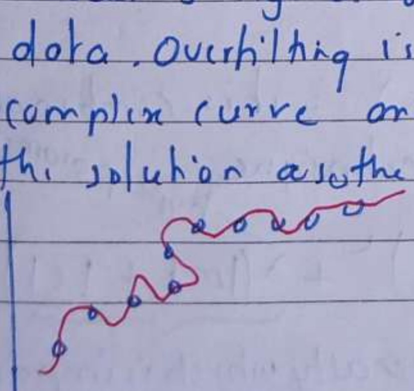
- it is important to understand prediction errors (bias and Variance) when it comes to accuracy in any ML Algorithm
- Thre is tradeoff between model's ability to minimize bias and variance which is referred to as best solution for selecting a value of regularization constant
- proper understanding of these error would help to avoid the overfitting and underfitting of a dataset while training algorithm.

- **Bias :-** Bias is known as difference between prediction values by ML model and correct values. Being high in biasing will give large error in training as well as testing. & that's why it is always recommended that algorithm should always be low biased to avoid underfitting.
- by high bias data is predicted is in straight line format thus not fitting accurately in the data in the dataset such fitting called as underfitting.
- This happen the hypothesis is too simple or linear in nature



Variance . The variability of model prediction for given data point which tells us spread of our data is called variance of model.

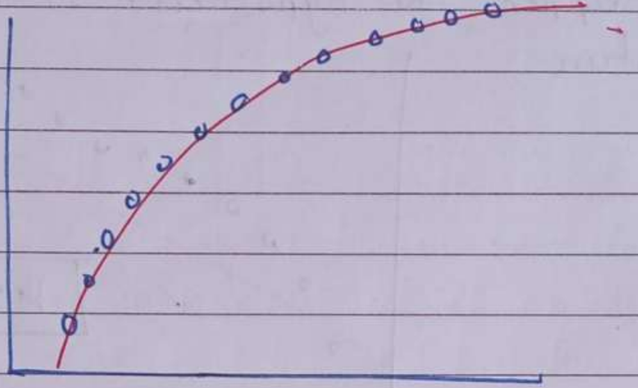
- The model with high variance has a very complex fit to training data and thus not able to fit on the test data or data hasn't seen as result such model works very well on training data but has high error rates on test data
- when model is high on variance it is said to as overfitting of data. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high



high variance data look like follows

- Bias-Variance tradeoff:- if the algorithm is too simple (hypothesis with linear eq^n) then it may be on high bias and low variance and thus it is error prone.

- if error fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias, in the latter condition the new entries will not perform well. there is something between both of these condition known as tradeoffs or bias-variance tradeoff



4) Effect of Regularization on loss function

loss function:-

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

- it measures how far an estimated value from its true value
- if we are training on different models LR, DT, RF to know which model performs better and which parameters are better loss function is useful.

Elastic Net:- (L1+L2) : it is combination of both regularization technique.

$$L_1\,Reg = \frac{\sum(y_i-\hat{y}_i)^2}{n} + \underbrace{\overset{\text{hyperparameter}}{\lambda}(|m|+|c|)}_{\text{penalty which is imposed on paramet}}$$

$$L_2 \text{ reg} = \frac{\Sigma (y_i - \hat{y}_i)^2}{n} \underbrace{+ \lambda (m^2 + c^2)}_{\text{penalty}}$$

Elastic $(L_1 + L_2)$:-

$$\frac{\Sigma (y_i - \hat{y}_i)^2}{n} + \lambda [(|m| + |c|) + (m^2 + c^2)]$$

adding combination percentage of Lasso & Ridge

$$\frac{\Sigma (y_i - \hat{y}_i)^2}{n} + \lambda [\underbrace{c (|m| + |c|)}_{\text{lasso}} + \underbrace{(1-c)(m^2 + c^2)}_{\text{Ridge}}]$$

c is between 0 to 1

$c = 1$ Lasso

$c = 0$ RIDGE

$c = 0.05$ = 50% Ridge & 50% Lasso.

Summary :- Ridge is majority is going to focus on regularization but Lasso is going to focus on feature selection as well

- if my job is only feature selection i will go for Lasso and if i want regularization then i will go for Ridge. and if i want both the i would go for Elastic Net.