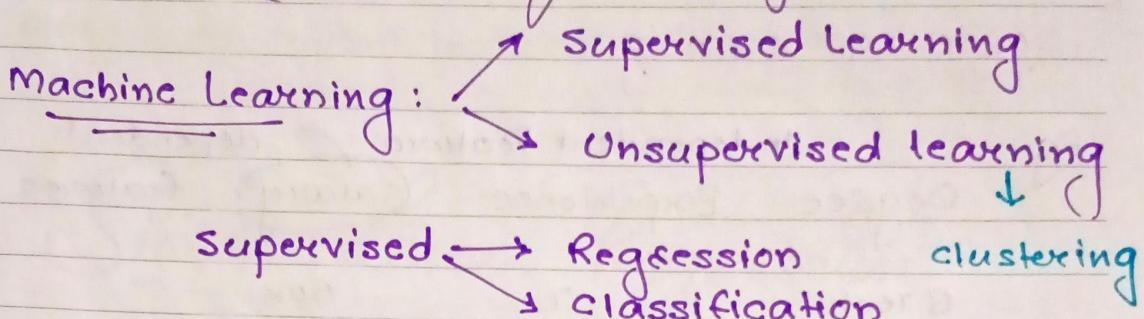


Machine Learning.



Supervised (output is known)

Regression →

1. Linear Regression
2. Polynomial Regression
3. SVR
4. Decision Tree
5. Random Forest
6. Xgboost , 7. KNN

Classification: →

1. Logistic Regression
2. SVM
3. Decision Tree
4. Random Forest
5. Naive Bayes
6. KNN

Unsupervised learning: (output is unknown)

clustering →

DBScan, kmeans,
Hierarchical,
silhouette scoring

Supervised Learning

Eg. Independent feature → dependent features

Degree	Experience	(Salary)
--------	------------	----------

B.Tech	7	50k
PhD	2	70k
-	-	-
!		

O/p = continuous → Regression problem

Eg.

Independent Feat.	dependent	
No. of play hrs	No. of study hrs	Pass/Fail feat.

O/p = categorical = classification problem

- Flight Price Prediction → Regression problem
- Algerian Fire Forest → classification problem
- Predict Air Quality Index → Regression Problem
- Rain Prediction → classification Problem
- Buying day of a person → classification problem

Note: Independent features is basically input features.

Simple Linear Regression.

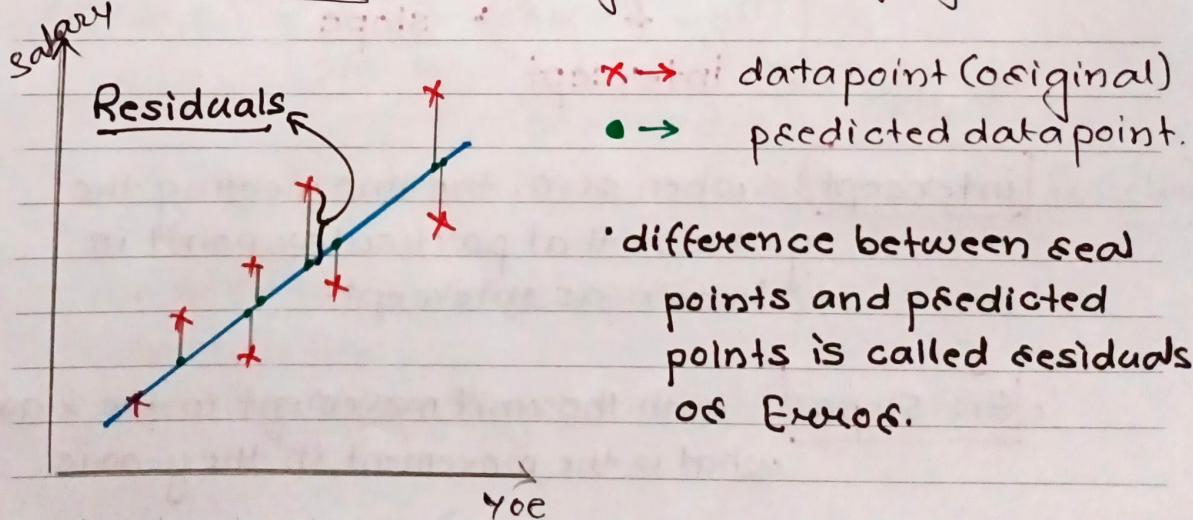
(one independent feature and one dependent feature)

Eg. aim: to create a model, which takes input as height and predict weight.

dataset: height, weight.

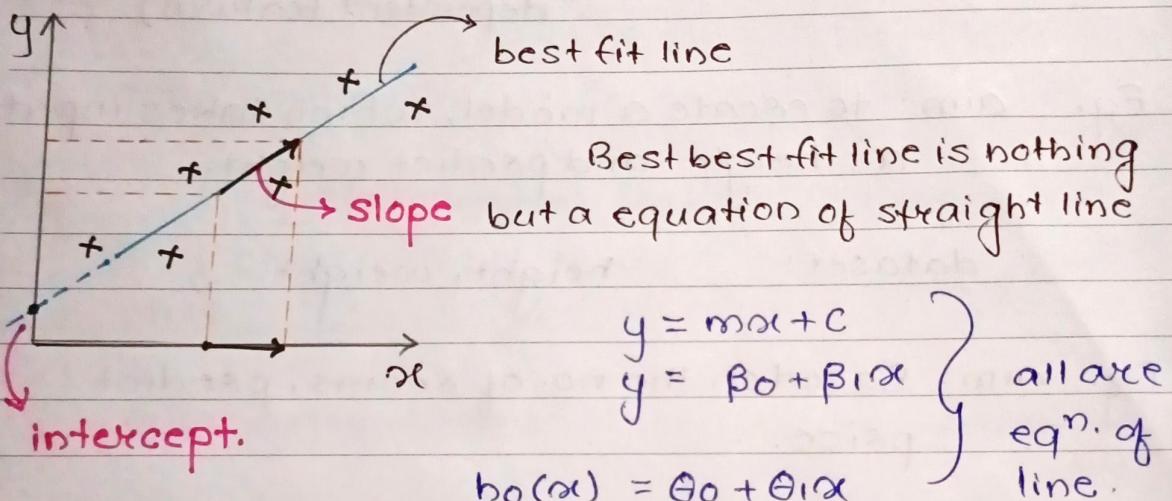
Eg. aim: Based on the no. of rooms, predict price.

Eg. model: year of experience and salary
predict: salary band on 1st year.



- Based on the training dataset, it finds the best fit line in such a way that the sum of difference between real points and predicted should be minimum.

First, we need to understand why are creating a straight line?



$$h_0(x) = \theta_0 + \theta_1 x$$

↓ ↘ slope
intercept

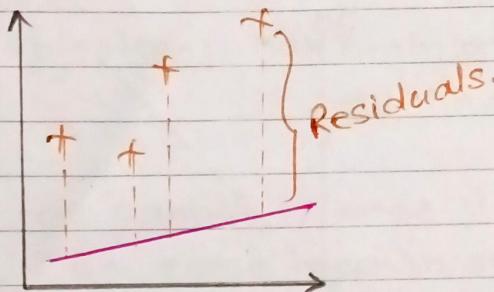
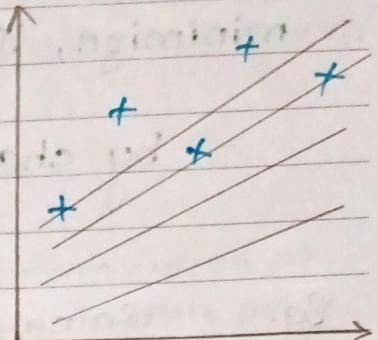
intercept: when $x=0$, the line meeting the ~~longest~~ y-axis, that particular point is known as intercept.

θ_1 : Slope: with the unit movement in the x-axis, what is the movement in the y-axis.

- By changing θ_0 and θ_1 , best fit line will be change.

θ_0, θ_1 : changing the values, will change the line.

after some iterations, we will get a best fit line which is known as training of the model.



we need to minimize this error using equation called as cost function.

Cost Function:

for easy calculations (derivatives)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

predicted

actual.

mean square error. (one of the cost function)

- we need to minimize cost function to get the best fit line.
- dividing by m to get the average (mean).

final aim:

$$\text{minimize, } J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x)^{(i)} - y^{(i)})^2$$

• by changing the values of θ_0 and θ_1 .

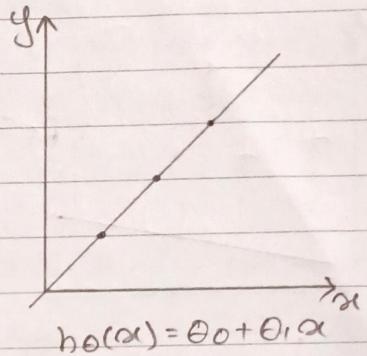
Let us consider

Eg. Training dataset.

x	y
1	1
2	2
3	3

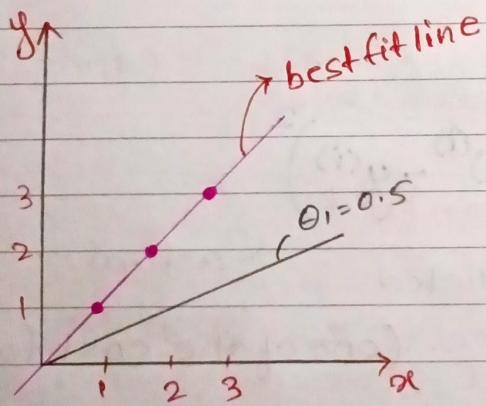
$$\theta_0 = 0$$

$$h_\theta(x) = \theta_1 x$$



Now, let's assume $\theta_1 = 1$

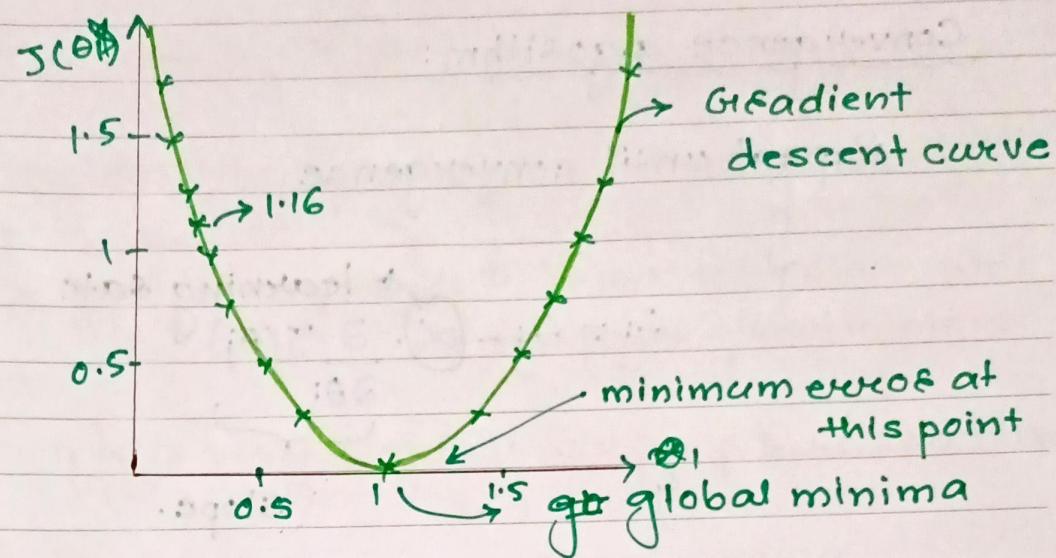
$$\therefore h_\theta(x) = x.$$



$$\begin{aligned} J(\theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x)^{(i)} - y^{(i)})^2 \\ &= \frac{1}{3} [0 + 0 + 0] \\ &= 0. \end{aligned}$$

when $\theta_1 = 0.5$

$$\begin{aligned} J(\theta_1) &= \frac{1}{3} [(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \\ &= \underline{1.16}. \end{aligned}$$



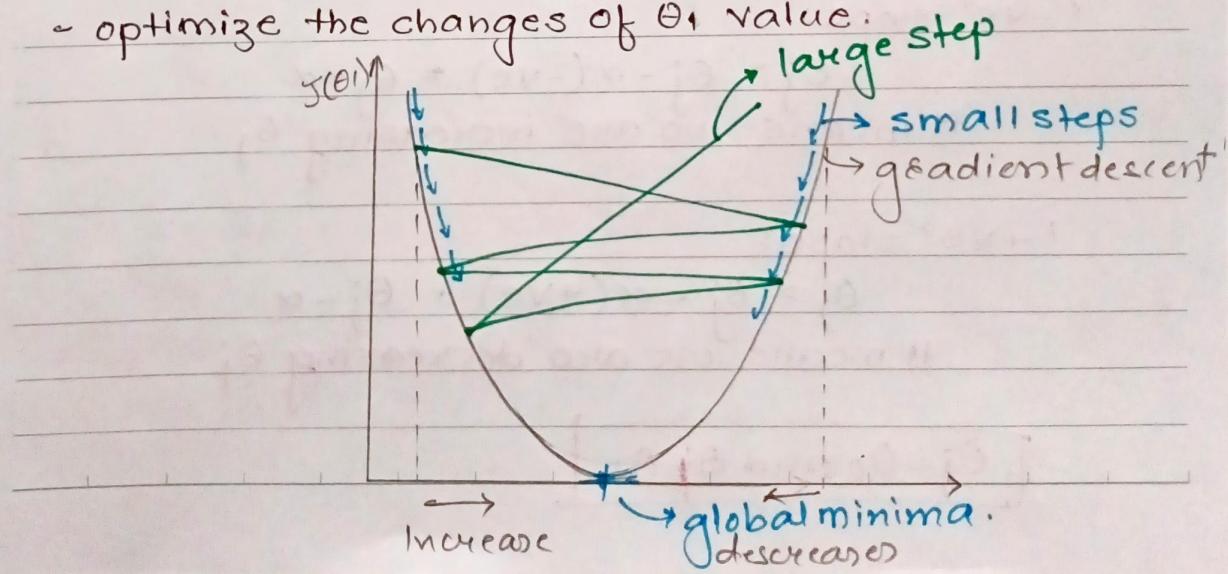
Objective: our main aim is to come near global minima.

we cannot change θ value manually, there should be some mechanism to change θ value to get the global minima.

To overcome this, we use Convergence algorithm.

Convergence algorithm:

- optimize the changes of θ_1 value.



Convergence algorithm:

Repeat until convergence,

{

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta_j)}{\partial \theta_j}$$

learning rate
slope.

}

+ve or -ve slope:

left side of the line facing downwards
→ -ve' slope

right side of the line facing upwards
→ +ve' slope

-ve' slope:

$$\theta_j = \theta_j - \alpha(-\text{ve}) = \theta_j + \alpha$$

It means we are increasing θ_j .

+ve' slope:

$$\theta_j = \theta_j - \alpha(+\text{ve}) = \theta_j - \alpha$$

It means we are decreasing θ_j .

$\theta_j = \theta_0 \text{ and } \theta_j \neq \theta_1$

α , Learning Rate

It decides the speed of the convergence.

If α is very small: It will take more time to reach global minima.

If α is very large: It will jump here and there, and won't reach to global minima.

α , should be around 0.001 for smaller steps.

convergence algorithm:

$$\text{cost function } J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

↓
predicted ↓
actual θ₁ truth point

mean square error.

m = no. of datapoint.

difference between cost function and loss function.

cost function: we find error for all the points and take average of it.

loss function: we find error for observed points and if we find error for all the points then it is loss function.

$$\begin{aligned} \text{loss function} &:= \left(\underbrace{h_{\theta}(x^{(i)})}_{\text{predicted value}} - \underbrace{(y^{(i)})}_{\text{actual value}} \right)^2 \\ &= \left(\hat{y}_j - y_j \right)^2 \end{aligned}$$

* for every single point we need to find loss function.

To achieve global minima,

derivative w.r.t θ_0 , $j=0$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(\alpha)^{(i)} - y^{(i)})^2 \right\}$$

$$= \frac{1}{2m} h_{\theta}(\alpha) = \theta_0 + \theta_1 \alpha$$

$$= \frac{\partial}{\partial \theta_0} \left\{ \frac{1}{2m} \sum_{i=1}^m [(\theta_0 + \theta_1 \alpha)^i - y^i]^2 \right\}$$

$$\frac{\partial}{\partial \alpha} [(1+\alpha) - y]^2 = \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i) \times 1$$

$$= \underline{\underline{2[(1+\alpha) - y]}} \times 1 = \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i)$$

derivative w.r.t. θ_1 , $j=1$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_1} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(\alpha)^{(i)} - y^{(i)})^2 \right\}$$

$$= \frac{\partial}{\partial \theta_1} \left\{ \frac{1}{2m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i)^2 \right\}$$

$$= \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i) \alpha.$$

Repeat until convergence

{

$$\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^m (h_\theta(x)^i - y_i)^2$$

$$\theta_1 := \theta_1 - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (h_\theta(x)^i - y_i) x^{(i)}$$

}

* learning rate: speed of convergence

Types of cost function:

- ① MSE: mean square error
- ② MAE: mean absolute error
- ③ RMSE: root mean square error.

Mean Square Error

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

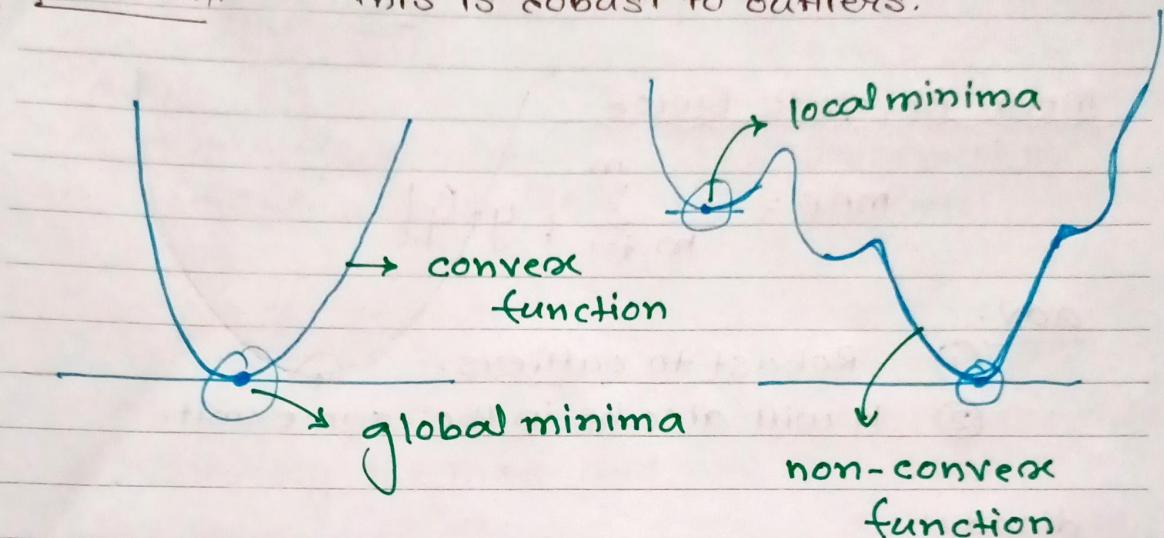
actual.
 $\hat{y} = \theta_0 + \theta_1 x$
predicted.

Adv

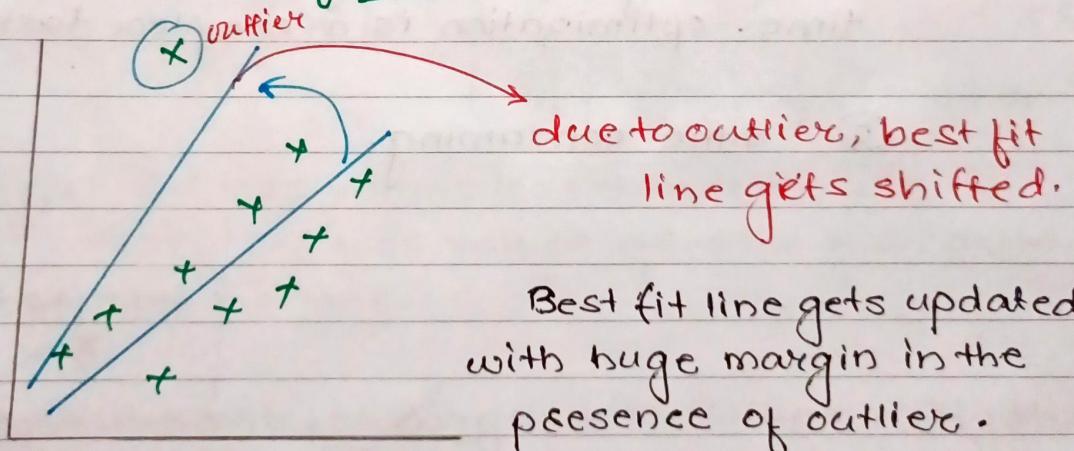
- This equation is differentiable.
- It also has one global minima.

disadv

- This is robust to outliers.



∴ main adv. of MSE that it has convex function.



Eg. (salary) \rightarrow dependent feature

Experience \rightarrow independent feature

$$(y - \hat{y})^2 \text{ (lakhs)}^2$$

\hookrightarrow unit changing \rightarrow time complexities increased.

we don't do scaling for dependent feature.

$(y - \hat{y})^2 \rightarrow$ squared \rightarrow error \rightarrow penalized
 \downarrow
Increased.

Mean Absolute Error

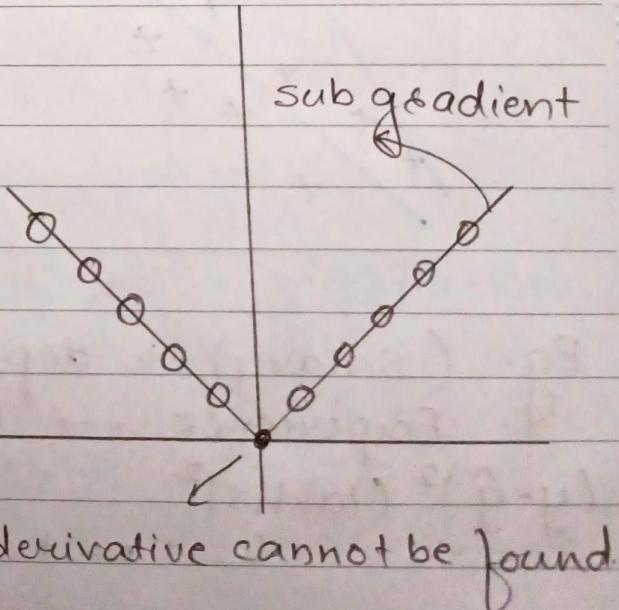
$$MAE = \frac{1}{m} \sum_{i=1}^m |y - \hat{y}|$$

adv:

- ① Robust to outliers.
- ② It will also be in the same unit.

disadv:

- ① convergence usually takes more time. optimization is a complex task.
- ② time consuming.



Root mean Square Error (RMSE)

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

adv:

- It is differentiable.
- unit remain same.

disadv:

- Not robust to outliers.

Huber Loss Function

The Huber loss offers the best of both worlds by balancing the MSE and MAE together.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2} (y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2} \delta^2 & \text{otherwise.} \end{cases}$$

It says: for loss values less than delta, use the MSE, for loss values greater than delta, use the MAE.

using the MAE for larger loss values mitigates the weight that we put on outliers so that we still get a well-sounded model.

At the same time, we use the MSE for the smaller loss values to maintain a quadratic function near the centre.

This has the effect of magnifying the loss values as long as they are greater than 1.

Once the loss for those data points dips below 1, the quadratic function down-weights them to focus the training on the higher-error data points.

Note: use the Huber loss any time you feel that you need a balance between giving outliers some weight, but not too much.

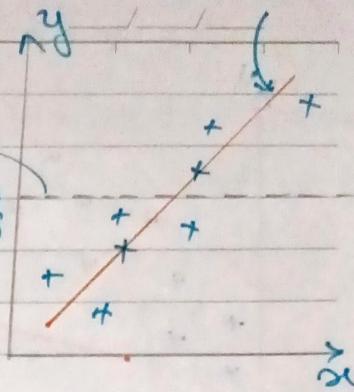
How can we check if a model is good or not?
→ using performance metrics.

Performance metrics

- ① R-squared
- ② Adjusted-R squared

R-squared: measures the performance of the model.

Best fit line



$$R\text{-squared} = 1 - \frac{SS_{\text{Res}}}{SS_{\text{Total}}}$$

SS_{Res} = Sum of square Residuals

SS_{Total} = sum of square average

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

low value high value

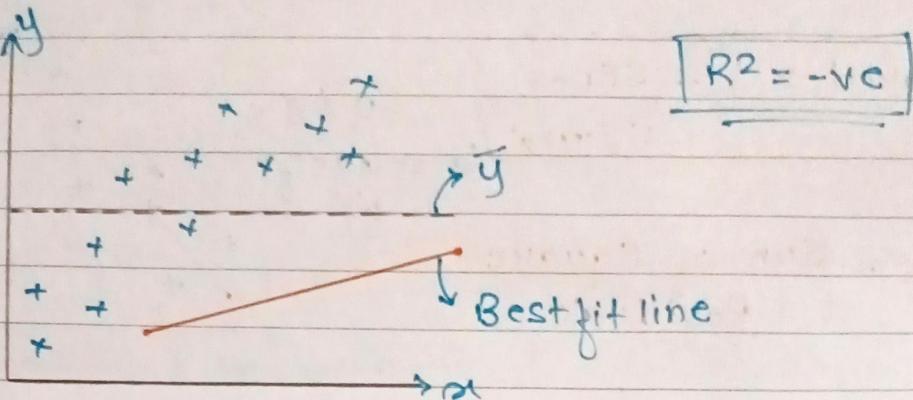
\bar{y} = average of y.

$$= 1 - \left\{ \frac{\text{small value}}{\text{Bigger value}} \right\} \rightarrow \text{small value}$$

R-squared ≤ 1

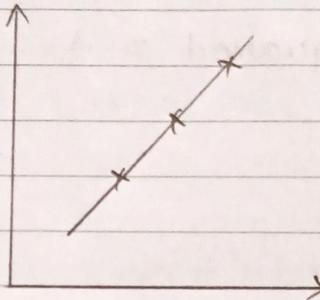
If R-squared = 0.85 = 85% accurate
= 0.75 = 75% accurate

* If R-squared is 've' then the model is not working is good.



Here $(y_i - \hat{y}) > (y_i - \bar{y}) \rightarrow R^2$ will be negative.

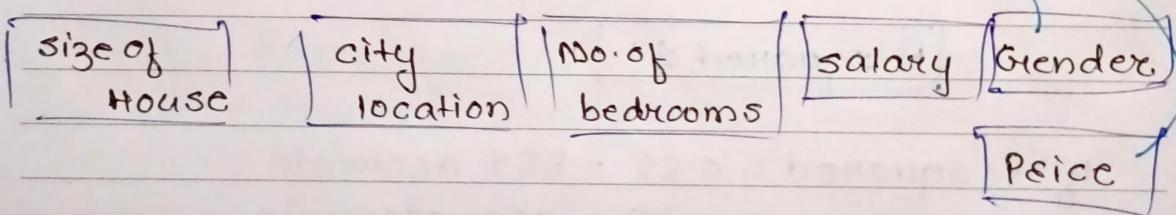
$$R^2 = 1 \rightarrow$$



Adjusted R-squared

Eg.

No direct correlation



Before, when only size of house was present as an independent feature to get the price and has some R-squared value. But with addition of city location R² got increase. after that with addition of no.of bedroom, gender the value got increase, but there is ~~not~~ no direct correlation between gender and price, the value shouldn't increased.

R^2	features added	adj. R^2	Independent feature
65%	size of house	63%	P=1
75%	location	73%	P=2
88%	no. of bedrooms	86%	P=3
90%	Gender	85%	P=4

↓ no direct correlation w/ price
 * very slightly increase, but this shouldn't happen.
 To solve this problem we use Adjusted R^2 .

$$\text{Adjusted } R^2 = \frac{1 - \frac{(1-R^2)(N-1)}{N-p-1}}{1}$$

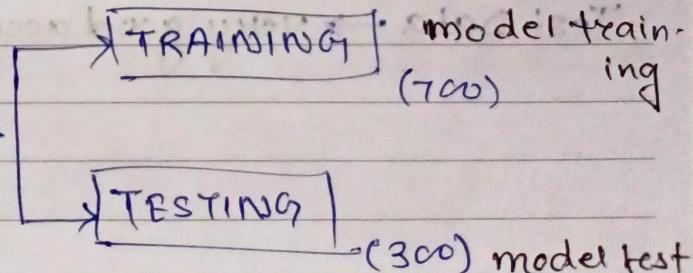
$$N = \text{No. of data points} \quad p = \text{No. of independent features}$$

* Adjusted R^2 is the best metrics to evaluate the model.

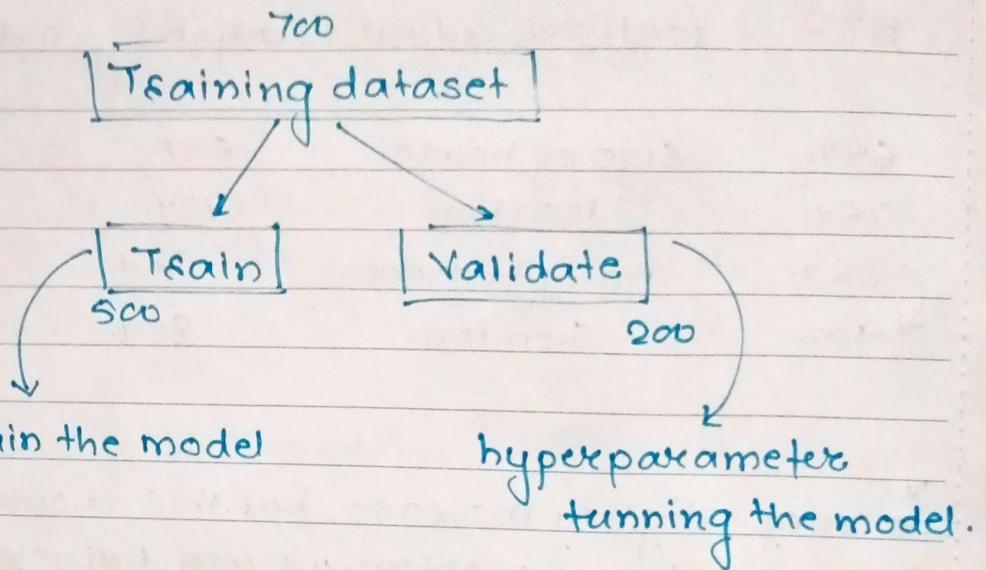
Overfitting and Underfitting

dataset

1000 datapoints



overfitting and underfitting (Bias and Variance)



MODEL

Generalize

Model

Ttrain Data \rightarrow very good accuracy (90%)

Test Data \rightarrow very good accuracy (85%)

our aim is to get good test and train accuracy.

Ttrain Data \rightarrow very good accuracy 90% [low bias]

Test Data \rightarrow very good accuracy 85% [low variance]

Overfitting
TRAIN → very good accuracy (90%) [low Bias]

TEST → Bad accuracy (50%) [High Variance]

Underfitting

TRAIN → model accuracy is low. [High bias]

TEST → model accuracy is low/high. [low or high variance]

- * we can solve this issue by performing hyperparameter tuning or by increasing no. of dataset.