Machine Learning.

Machine Learning: → Supervised Learning
→ Unsupervised learning

Supervised → Regression
→ classification

clustering

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.

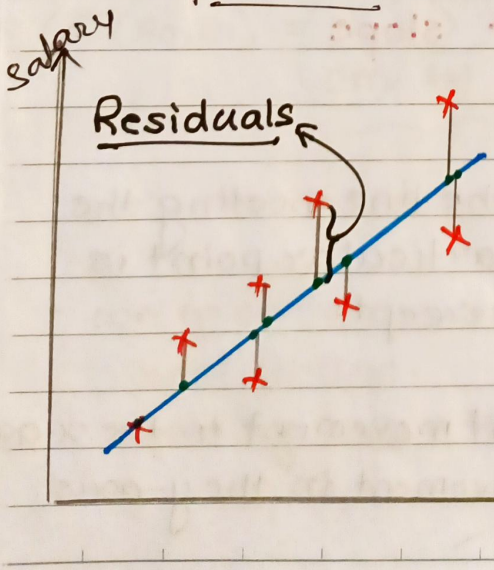
(one independent feature and one dependent feature)

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 up year.



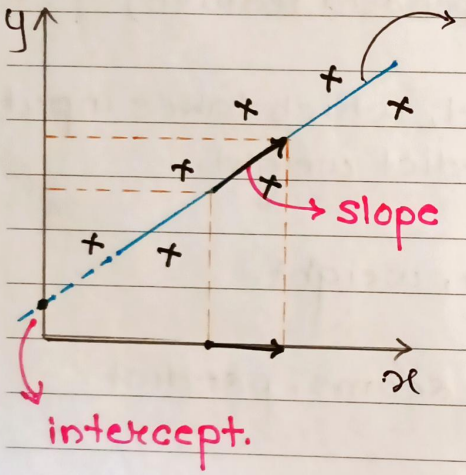
× → datapoint (original)

● → predicted datapoint.

• difference between real points and predicted points is called residuals or Errors.

- 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?



Best best fit line is nothing but a equation of straight line

$y = mx + c$
$y = \beta_0 + \beta_1 x$
$h_\theta(x) = \theta_0 + \theta_1 x$

} all are eqn. of line.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

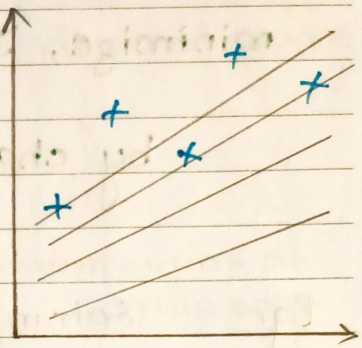
θ_0 → intercept, θ_1 → slope

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

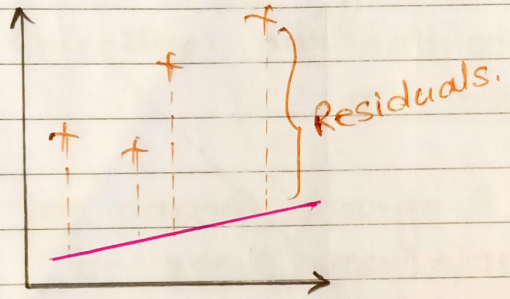
θ_1: slope: with the unit movement in the x-axis, what is the movement in the y-axis.

- 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 errors using equation called as cost function.

Cost Function:

for easy calculations (derivatives)

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

$h_\theta(x)^{(i)}$ → predicted

$y^{(i)}$ → actual.

mean square errors. (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:

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.

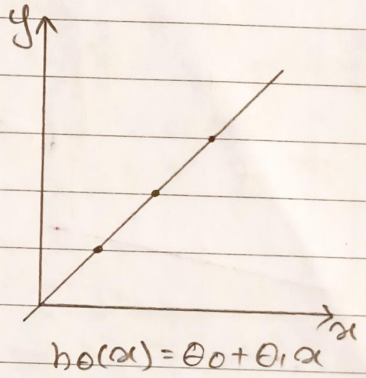
Eg. Training dataset.

x	y
1	1
2	2
3	3

Let us consider

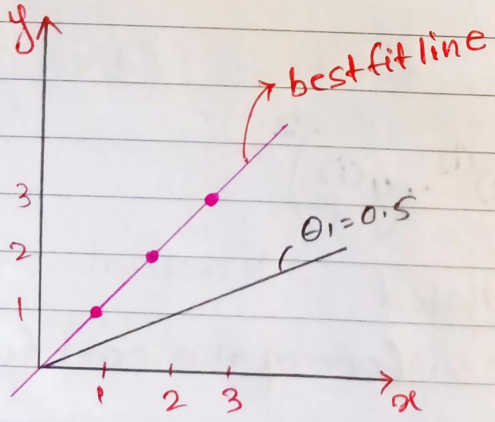
$\theta_0 = 0$

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



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

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



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

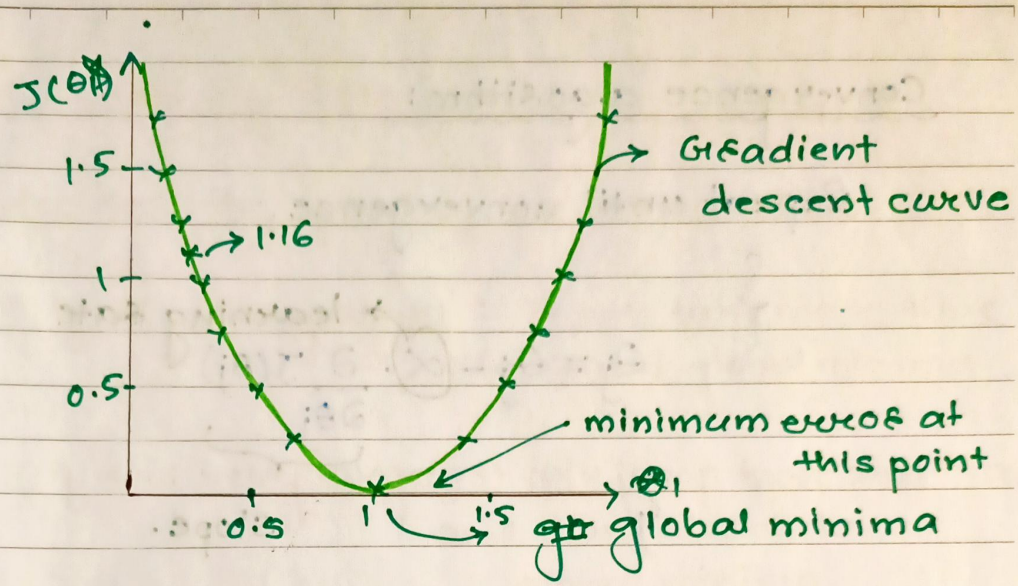
$$= \frac{1}{3} [0 + 0 + 0]$$

$$= 0.$$

when $\theta_1 = 0.5$

$$J(\theta_1) = \frac{1}{3} \left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2 \right]$$

$$= 1.16.$$



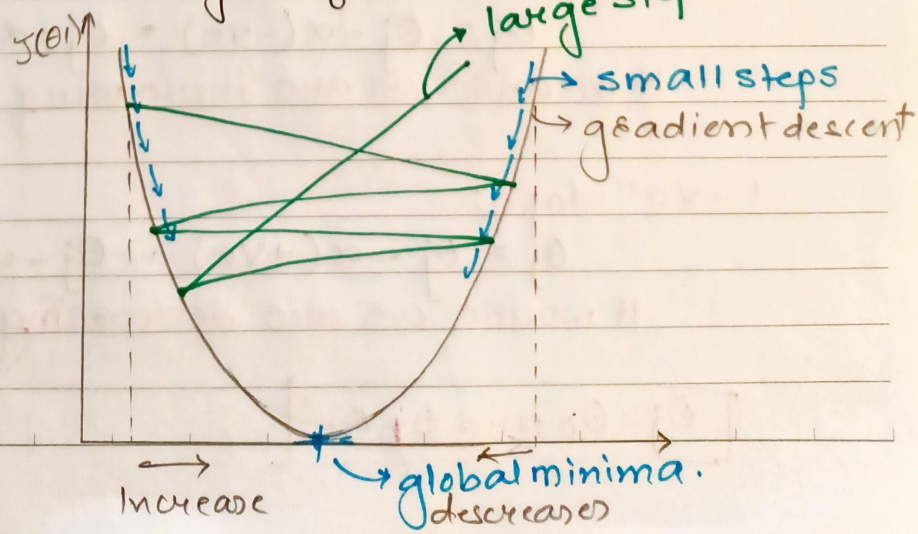
Objective: our main aim is to come near global minima.

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,

{

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

(α: learning rate; $\frac{\partial}{\partial \theta_j} J(\theta_j)$: slope.)

}

+ve or -ve slope:

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

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

'-ve' slope:

$$\theta_j = \theta_j - \alpha(-ve) = \theta_j + \alpha$$

It means we are increasing θ_j.

'+ve' slope:

$$\theta_j = \theta_j - \alpha(+ve) = \theta_j - \alpha$$

It means we are decreasing θ_j

$\theta_j = \theta_0$ and ~~θ_j~~ θ_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:

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

$J(\theta_0, \theta_1)$: cost function

mean square Error.

$h_\theta(x)^{(i)}$: predicted

$y^{(i)}$: actual or truth point

m = no. of datapoint.

difference between cost function and loss function.

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

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

$$\text{loss function} := \left(h_\theta(x)^{(i)} - (y)^i \right)^2$$

$h_\theta(x)^{(i)}$: predicted value

$(y)^i$: actual value

$$= \left(\hat{y}_j - y_j \right)^2$$

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

To achieve global minima,

derivative w.r.t θ_0, $j=0$

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

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

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

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

$\frac{\partial}{\partial x} \{(1+x) - y\}^2$

$= 2[(1+x) - y] \times 1$

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

derivative w.r.t. θ_1, $j=1$

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

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

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

Repeat untill convergence

{

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

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

}

* 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

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

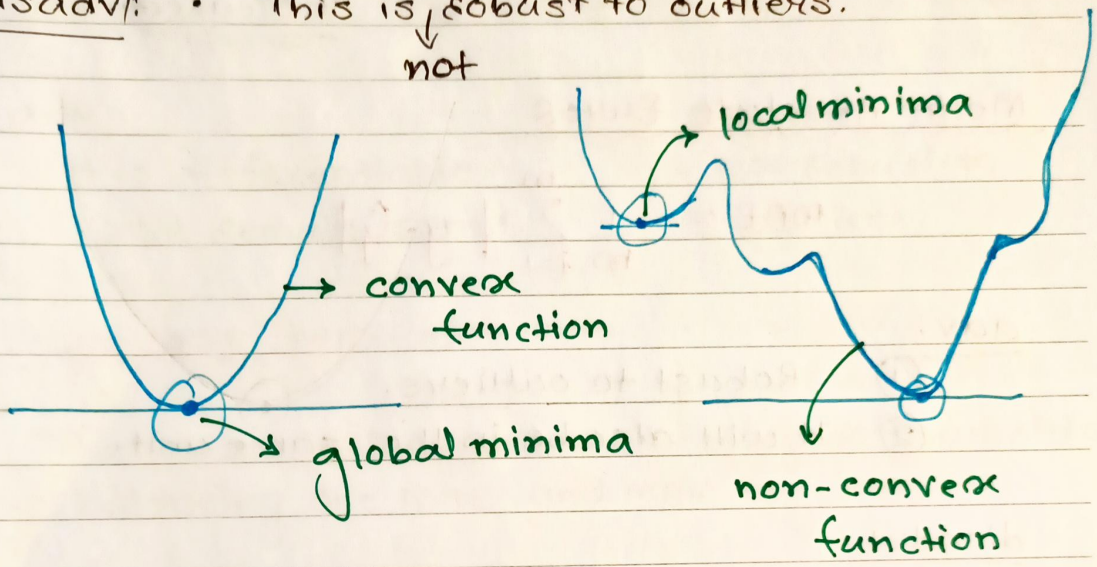
y → actual.

$\hat{y}$ → predicted.

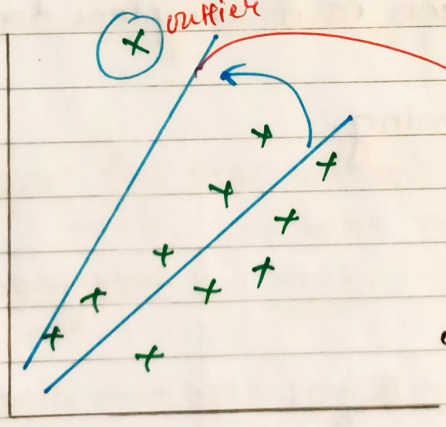
$\therefore \hat{y} = \theta_0 + \theta_1 x$

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

disadv:
- This is not robust to outliers.



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



due to outlier, best fit line gets shifted.

Best fit line gets updated with huge margin in the presence of outlier.

Eg. (salary) → dependent feature
Experience → independent feature

$(y-\hat{y})^2$ (lakhs)^2
↳ unit changing → time complexities increases.

we don't do scaling for dependent feature.

$(y-\hat{y})^2$ → squared → errors → penalized

↓

Increased.

Mean Absolute Errors

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

adv:

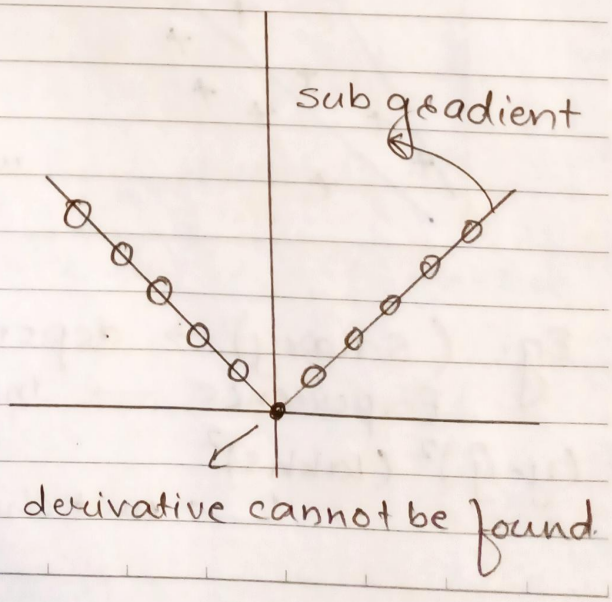
① Robust to outliers.

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)

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

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.

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

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-rounded 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.

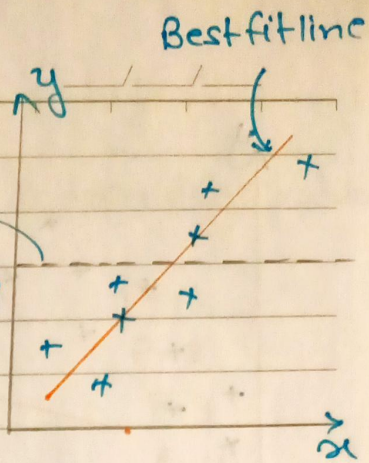
How can we check if a model is good or not?

→ using performance metrics.

Performance metrics

1. R-squared
2. Adjusted-R squared

R-squared: measures the performance of the model.



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

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

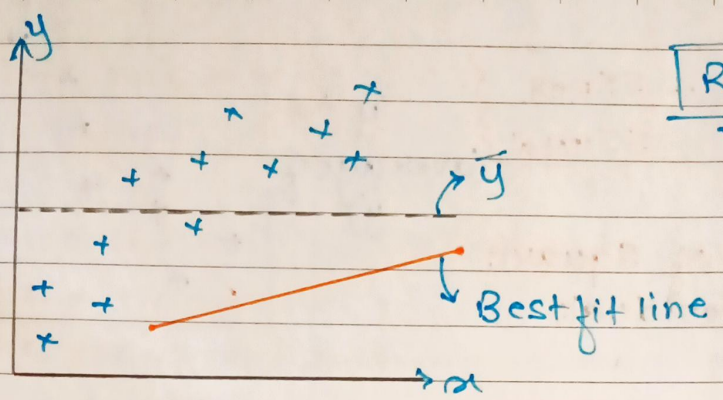
$\bar{y}$ = average of y.

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

$R\text{-squared} \leq 1$

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

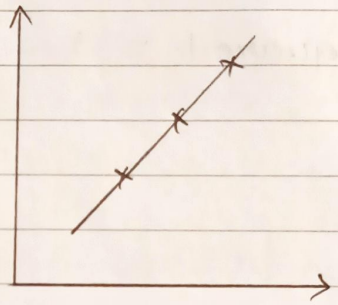
* If R-squared is '-ve' then the model is not working is good.



R^2 = -ve

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

$R^2 = 1$ →



Adjusted R-squared

Eg.

No direct correlation

size of House	city location	No. of bedrooms	salary	Gender	Price

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 R^2 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.r.t price

* very slightly increase, but this shouldn't happen.

To solve this problem we use Adjusted R^2.

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

N = No. of datapoints

P = No. of independent features

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

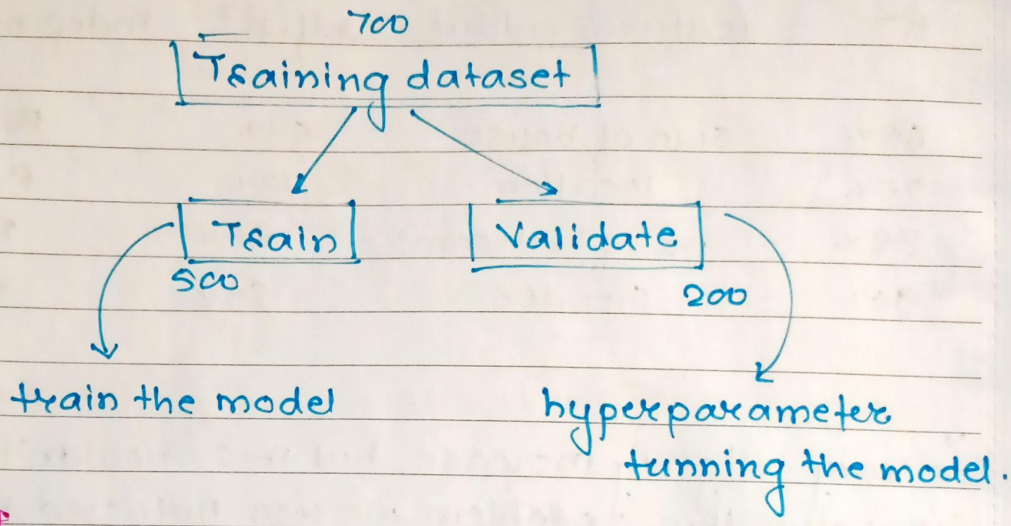
Overfitting and Underfitting

dataset

1000 datapoints → TRAINING (700) model training

→ TESTING (300) model test

overfitting and underfitting (Bias and Variance)



MODEL

Generalize Model

Train Data ⟶ very good accuracy (90%.)

Test Data ⟶ very good accuracy (85%.)

our aim is to get good test and train accuracy.

Train Data ⟶ very good accuracy 90%. [low bias]

Test Data ⟶ 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 tunning or by increasing no. of dataset.