Data Science & Machine Learning

~Soumyansh

Simple Linear Regression →

Supervised Machine Learning →

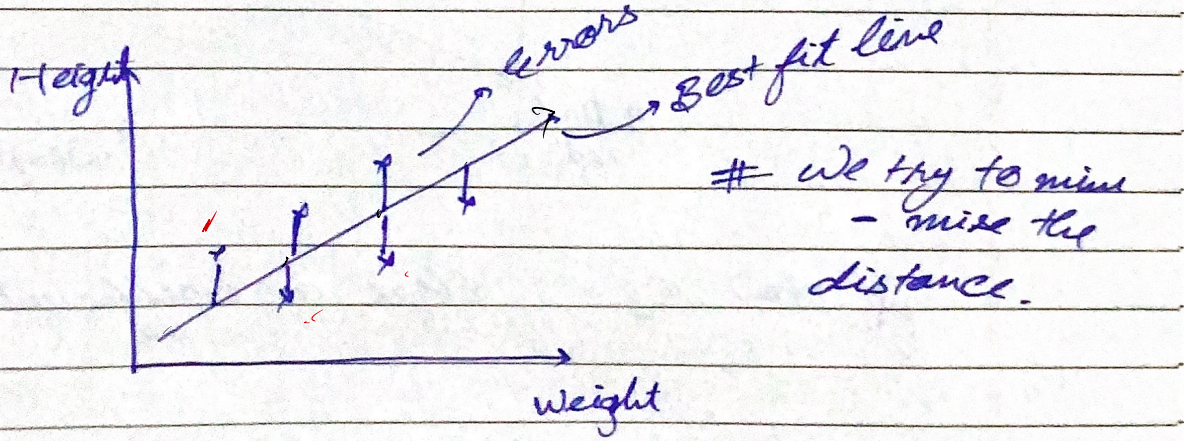
Dataset →

I/P feature	Weight	Height (Dependent feature)
	74 kg	170 cm ✓
	80 kg	180 cm ✓
	75 kg	175.5 ✓
	-	-

Past Data

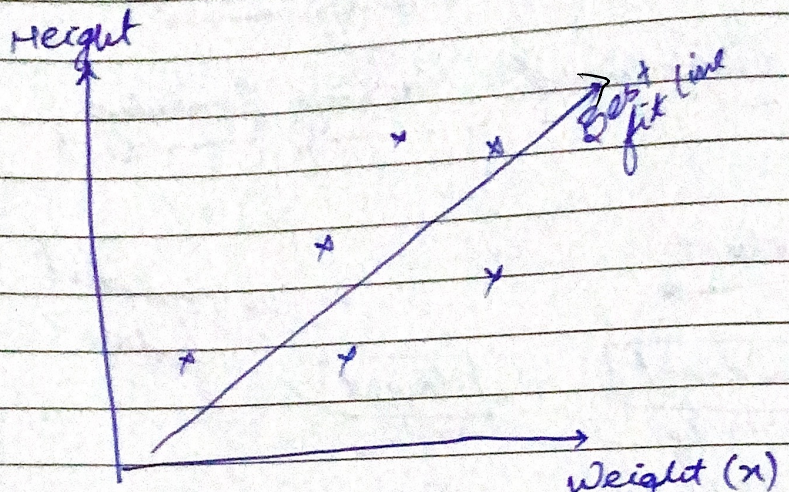
New weight → [Model] → Height

(Training)



We try to minimise the distance.

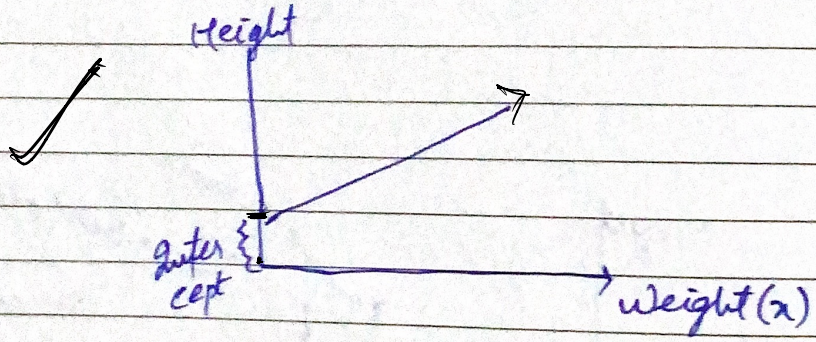
$y = mx + c$



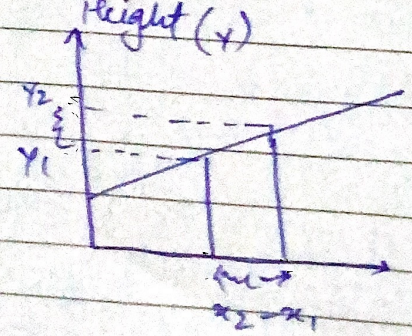
$h_\theta(x) = \theta_0 + \theta_1 x$

Now θ_0 = Intercept

If $x = 0$ then $h_\theta(x) = \theta_0$



Now θ_1 = slope or coefficient



$y = mx + c$

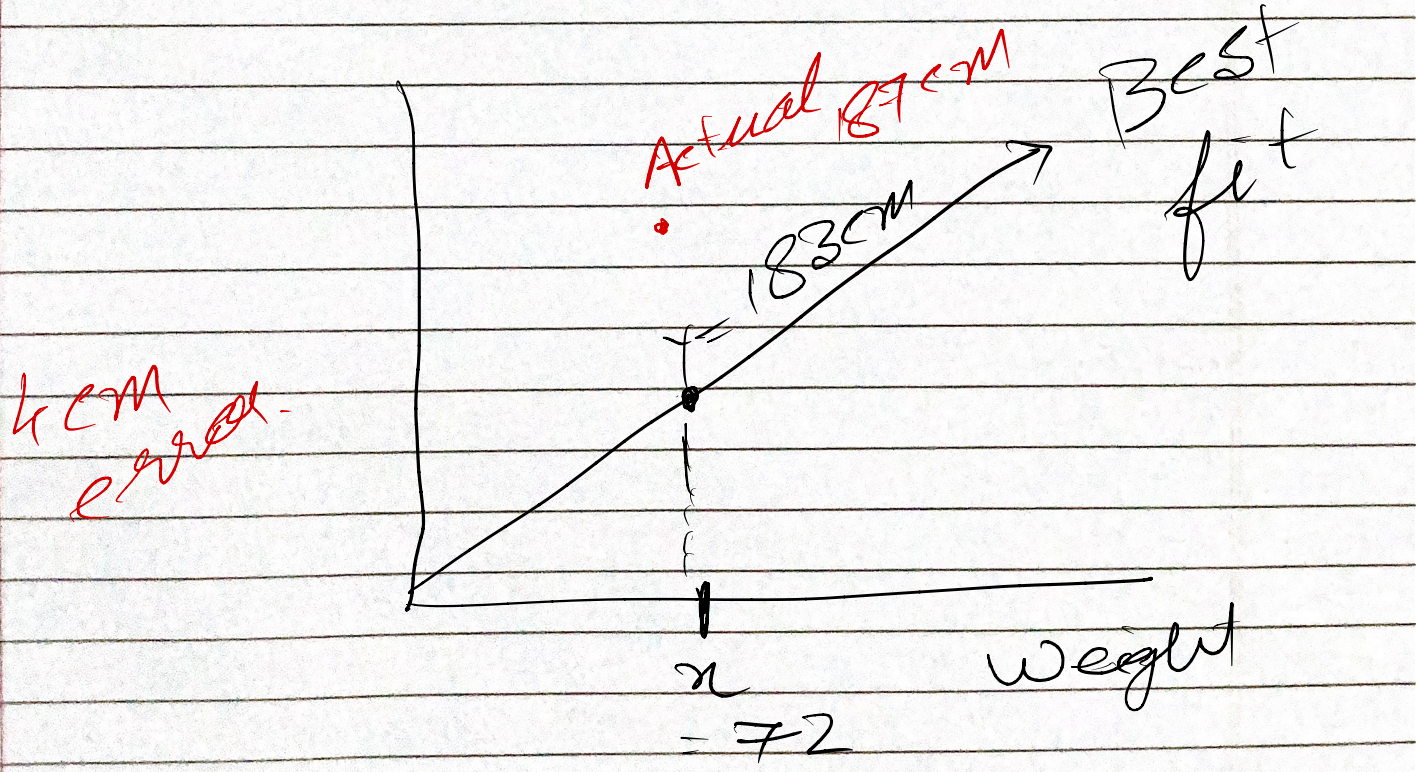
$\frac{y_2 - y_1}{x_2 - x_1}$

Error = $y - \hat{y}$ [where $\hat{y}$ = predicted value of y by model]

Now our aim is to minimise the sum of all errors

$$\min \sum \{y - \hat{y}\}$$

$\hat{y}$ = predicted



Machine Learning → Notes Part 2

Cost function

Gradient Descent → An algorithm to minimize a function by optimizing its parameters

Let's say there is a Science Test Max marks = 50

Friend asks you to guess his marks?

Guess 1 → 45 ?
friend → Too far
Guess 2 → 40 ?
friend → Still far
Guess 3 → 37
friend → Very close

$Y = mx + c$

Imp → In GD we start with random guess & then slowly move to the right answer.
How fast/slow will converge to the answer is determined by Learning rate

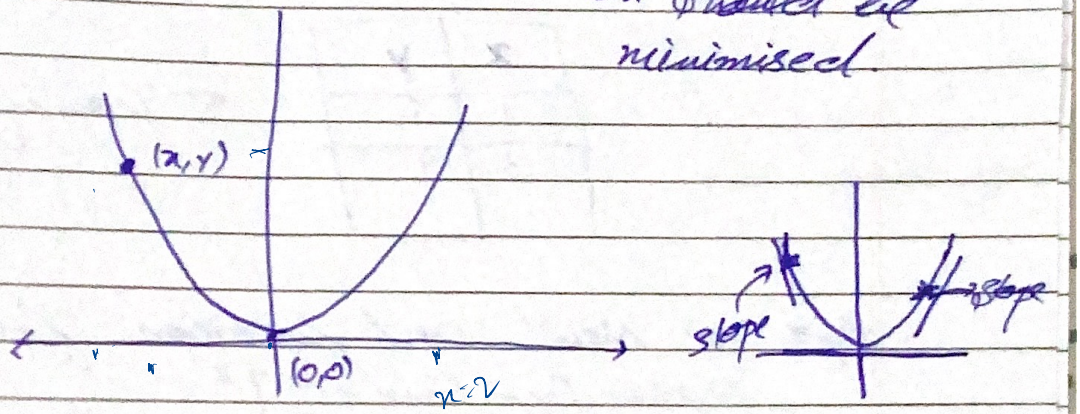
New value = Old value − Stepsize

where Stepsize = Learning rate × Slope

Ex → A simple square function

$f(x) = x^2$

Now as per GD this function should be minimised.



Now Step 1) Random guess (−2, 3) (x & y)

Step 2 → # Here $x = -2$ then $f(x) = x^2$ &

Slope of function = $\frac{d}{dx} f(x) = 2x = 2(-2)$

$= -4$ ✓

Step 3 → If $x = -1$ then $2x$ i.e slope $= -2$ ✓

Hence we are getting close to zero i.e moving in correct direction

Step 3 → If $x = +2$ then $2x$ i.e slope $= 4$ which is beyond zero so we're in wrong direction now

Step 4 → Now $\boxed{\text{new value} = \text{old value} - \text{slope} \times LR}$

& if a function has multiple parameters unlike $f(x) = x^2$, i.e function having more than one parameter

Ex Cost func^n for regression =

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

Step5 → Now there are two params m & c

Step6 → Again random guess let's say
$c=0$, $m=1$

x	y
1	2
3	4

← Training Data

Step7 → Now cost function for linear regression
$= [y-(mx+c)]^2$

plugging the values for training data (1,2) & (3,4)

$J(m,c) = [2-(c+m\times 1)]^2 + [4-(c+3m)]^2$

partial derivative $\frac{dJ}{dc} = -2[2-(c+m)] - 2[4-(c+3m)]$

for $c=0$ & $m=1$

differentiation wrt c ↓ $= -2[2-1] - 2[4-3]$

$= -4$

Step8 Now because i differentiated wrt c
So my
new c = old c − LR × (−4) ← slope
$= 0 - (0.001) \times (-4)$
$= 0.004$

Step9 Similar to above now we'll differentiate wrt m & new value of m will be found

Step 10 → Now new values will be taken for consideration & entire process will get repeated

Step 11 → When algo will stop?
Ans → When there won't be much improvement on cost for values of c & m

Imp → Choosing learning Rate: Try to be moderate

Machine Learning Notes → 3 →

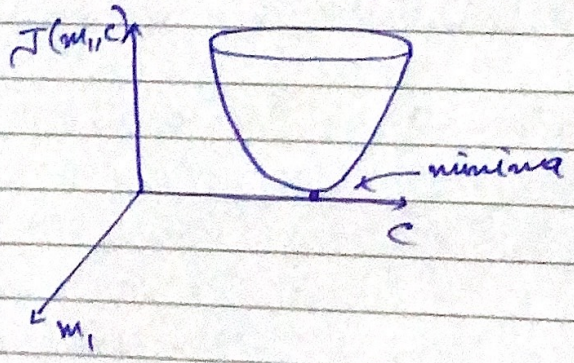
Multiple Linear Regression →

House pricing Dataset →

No. of Rooms	Size of house	Location	Price
1	1800	X	100000 $
2	1900	Y	200600 $
3	2000	Z	1000000 $

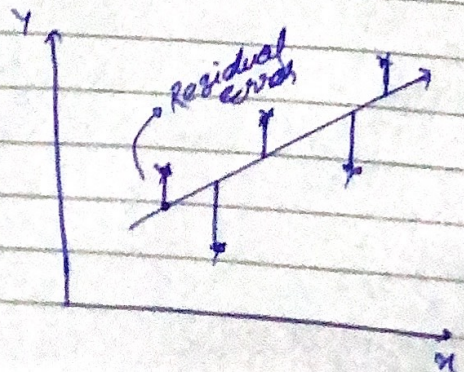
$$y = m_1x_1 + m_2x_2 + m_3x_3 \cdots + m_nx_n + c$$

Example curve → Multiple I/p features →



Performance Metrics used in Linear Regression →

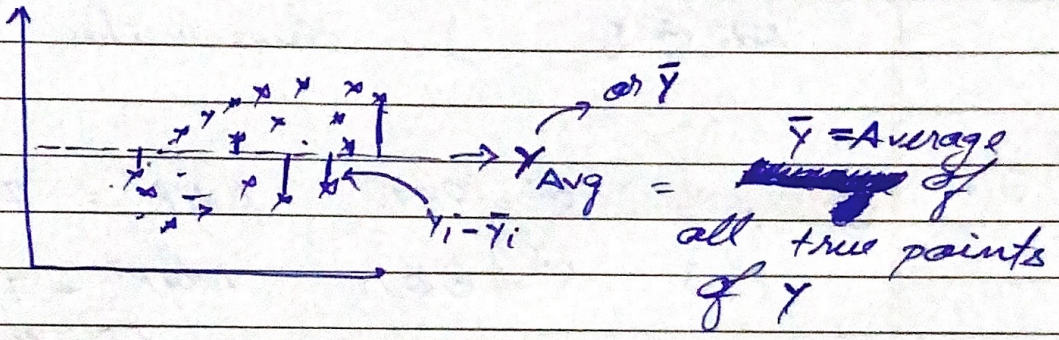
1) $R\text{squared} = 1 - \frac{SS_{Residual}}{SS_{Total}}$



y true value, $\hat{y}$ predicted

SS Residual (Sum of Square Residual) $= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

SS Total (Sum of square Total) $= \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$



$\bar{y}$ = Average of all true points of y

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

$1 - \frac{\text{Small Number}}{\text{Big Number}} \Rightarrow 1 - \text{Small Number} \approx 1$

0.70 → 70% Accuracy

0.85 → 85% " & so on

2) Adjusted R Squared → Extra feature (Owner POB)

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

N = no. of data points

P = no. of independent features

Dataset

No. of Rooms	Size	Location	Owner DOB	Price
—	—	—	—	—

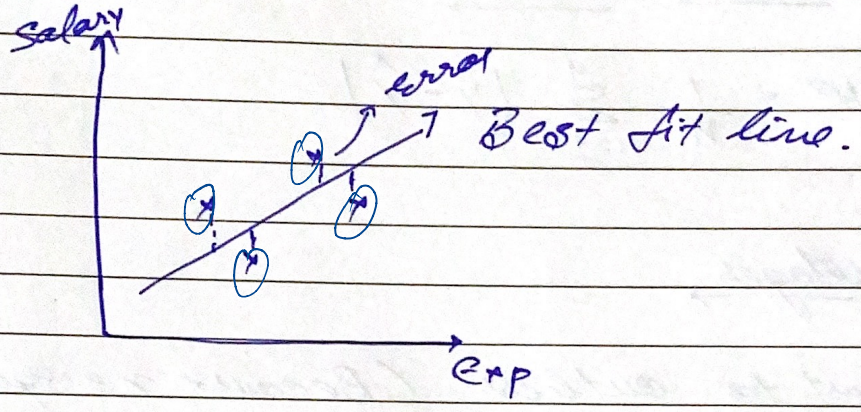
$R^2 \uparrow$
$Adj\ R^2 \downarrow$
} via penalty through denominator.

Ex $p=2$ $R^2 = 80\%$ $Adj\ R^2 = 76\%$.

$p=3$ $R^2 = 85\%$ $Adj\ R^2 = 79\%$.

(Penalty of non correlated feature)

Machine Learning Notes → 4



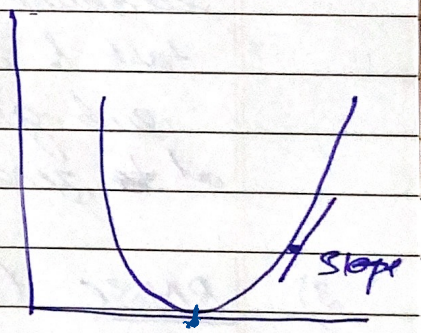
⟹ When we want to focus on Error metrics.

1) Mean Squared Error → **MSE**

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

Advantages →

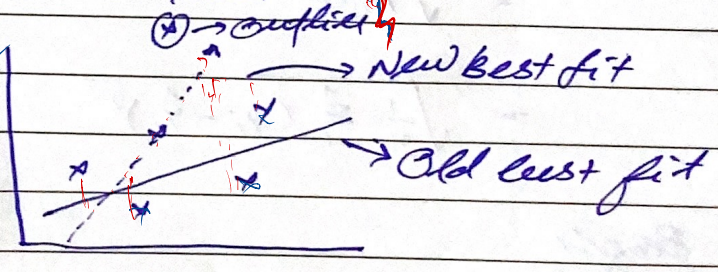
1) Differentiable at all points
2) Converges faster.



Disadvantages →

1) Not robust to outliers

∴ if this MSE will increase.



2) Output is not in the same unit.
Ex let's say salary error = 5000
then MSE will be $(5000)^2$ squared

2) Mean Absolute Error (MAE) **MAE**

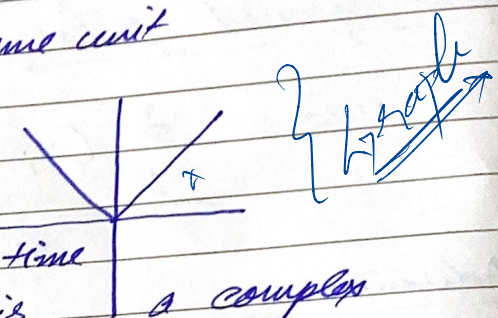
$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y - \hat{y}|$$

Advantages →

① Robust to outliers (Because no squaring) like MSE

② O/p will be in the same unit

Disadvantage →



1) Convergence takes more time because optimisation is a complex task & differentiables are happening at sub gradients & no differentiation at ~~to~~ zero

3) RMSE (Root mean Squared Error) **RMSE**

$$= \sqrt{MSE}$$

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

Benefit →

1) Same unit because of sqrt
2) Differential

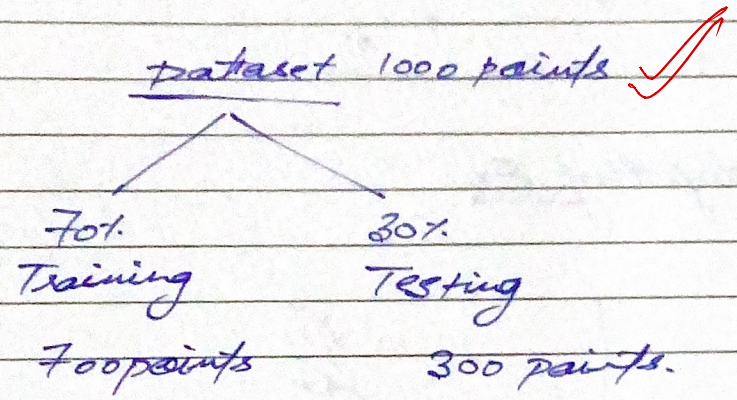
Disadvantage

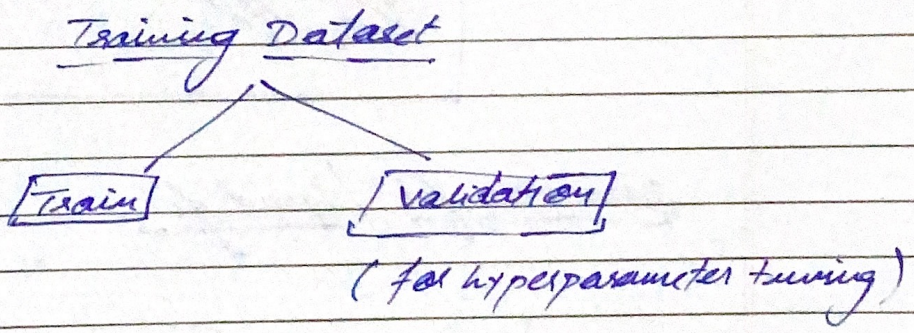
1) Not robust to ~~outliers~~ outliers

Machine Learning Notes - 5

Overfitting & Underfitting

1) Training dataset
2) Testing dataset
3) Validation dataset





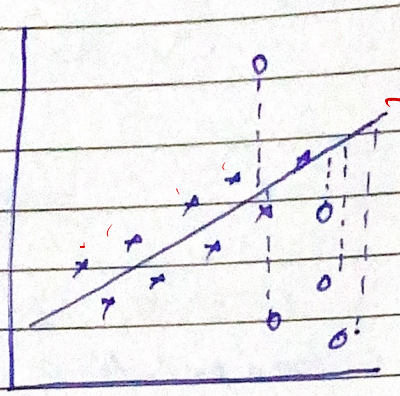
Overfitting

Very good accuracy with training data ie 90%.
Bad accuracy with test data, Ex → 50%.

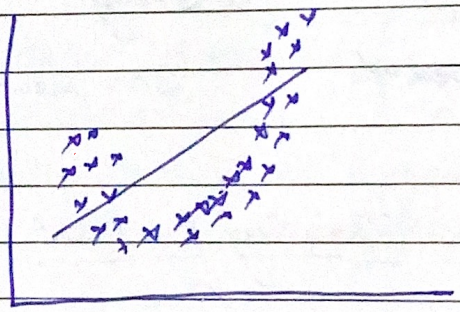
Underfitting

Training score is low (50-)
Testing score is low as well (50-)

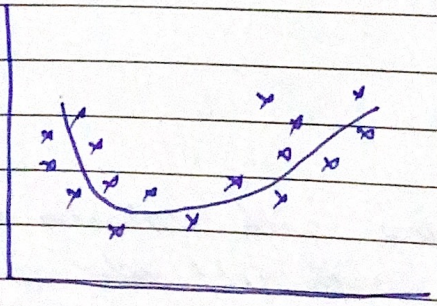
Overfitting Ex



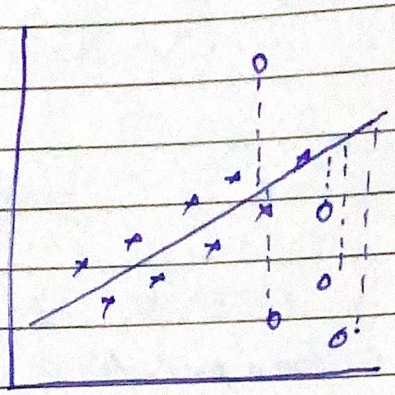
Underfitting Ex →



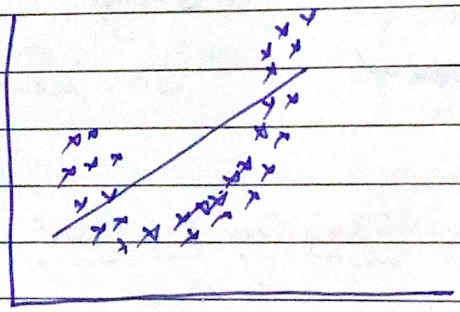
Balanced fit / Good fit →



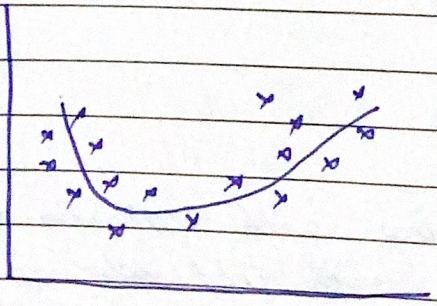
Overfitting Ex



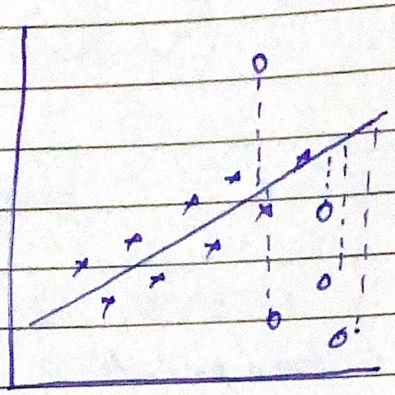
Underfitting Ex →



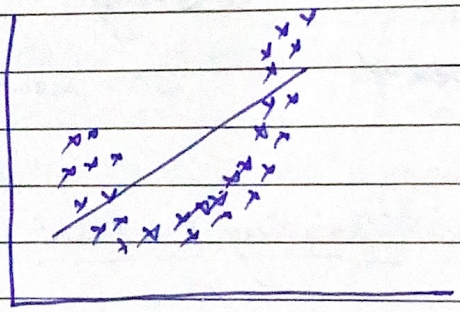
Balanced fit / Good fit →



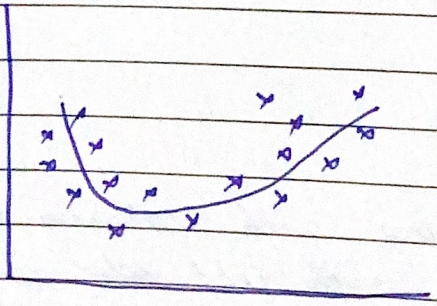
Overfitting Ex



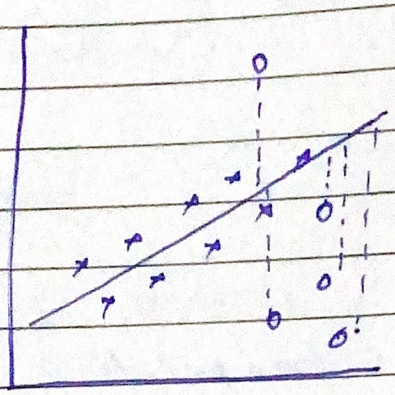
Underfitting Ex →



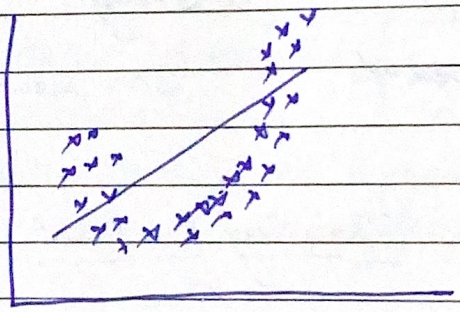
Balanced fit / Good fit →



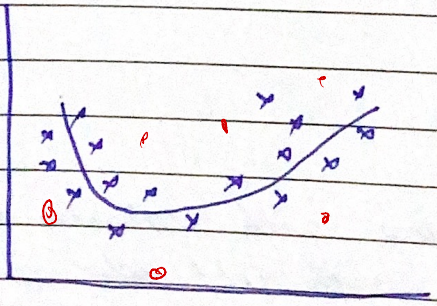
Overfitting Ex



Underfitting Ex →



Balanced fit / Good fit →



Machine Learning Note-6

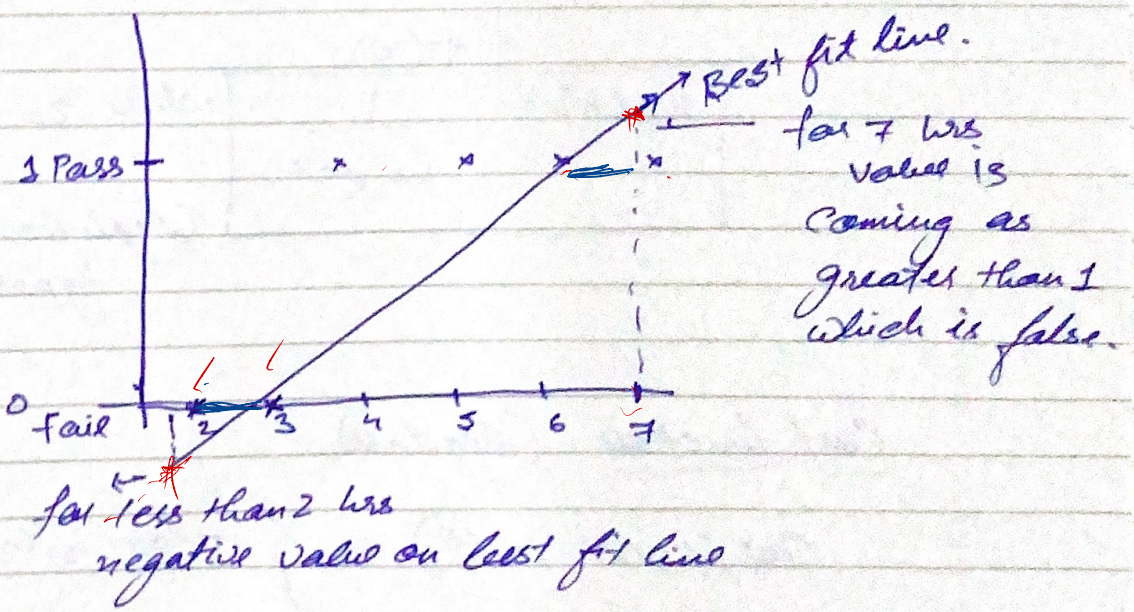
Logistic Regression (Binary classification)

Dataset → O/P (Binary Categories)

(Indep. endent Variable) Study hours. →	O/P (Dependent variable)
2	fail
3	fail
4	Pass
5	Pass
6	Pass
7	Pass

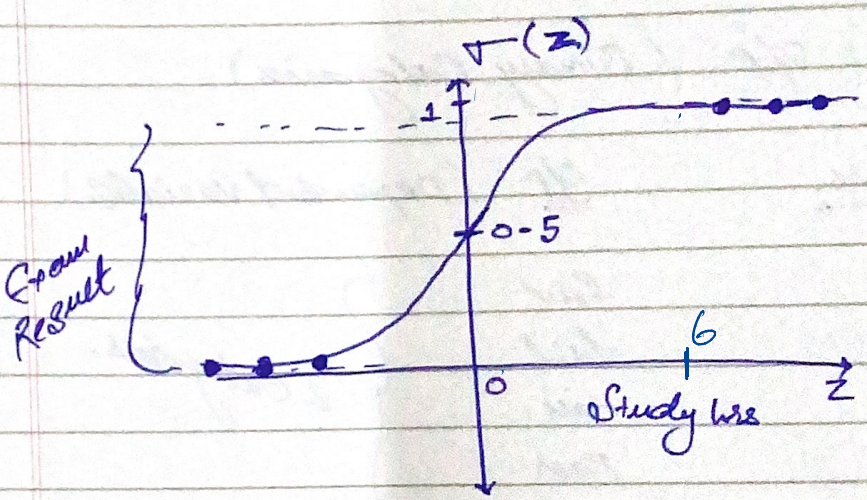
} 2 categories.

Why classification problem can't be solved using Simple Linear Regression? →



Logistic Regression → (Email / Spam / Non Spam)

Sigmoid function → $\frac{1}{1+e^{-z}}$ (Always b/w 0 & 1)



ie if $z \geq 0$ then $\sigma(z) \geq 0.5$

Logistic Regression = Sigmoid function on Simple Linear regression

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1)$$

$$= \sigma(z)$$

$$h_\theta(x) = \frac{1}{1+e^{-z}} \quad \text{where } z = \theta_0 + \theta_1 x_1$$

(Logistic Regression hypothesis)

Cost function (Log loss)

$$\text{Cost}(h_\theta(x)^i, y^{(i)}) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

Ex $\theta_0 = 1.5$, $\theta_1 = 0.6$ & $x = 5$ hrs

$$h_\theta(x) = \frac{1}{1 + e^{-z}}$$

$$= \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1)}}$$

$$= \frac{1}{1 + e^{-(1.5 + 3)}} = \frac{1}{1 + e^{-4.5}} = .818$$

ie near to 1 Hence pass

Imp → O/p is always categorical in Logistic Regression