⇒ Logistic Regression

- Used for Binary classification

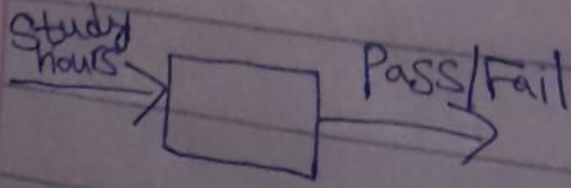
⇒ Classification means output will be categorical variables

⇒ Binary means there will be 2 categories

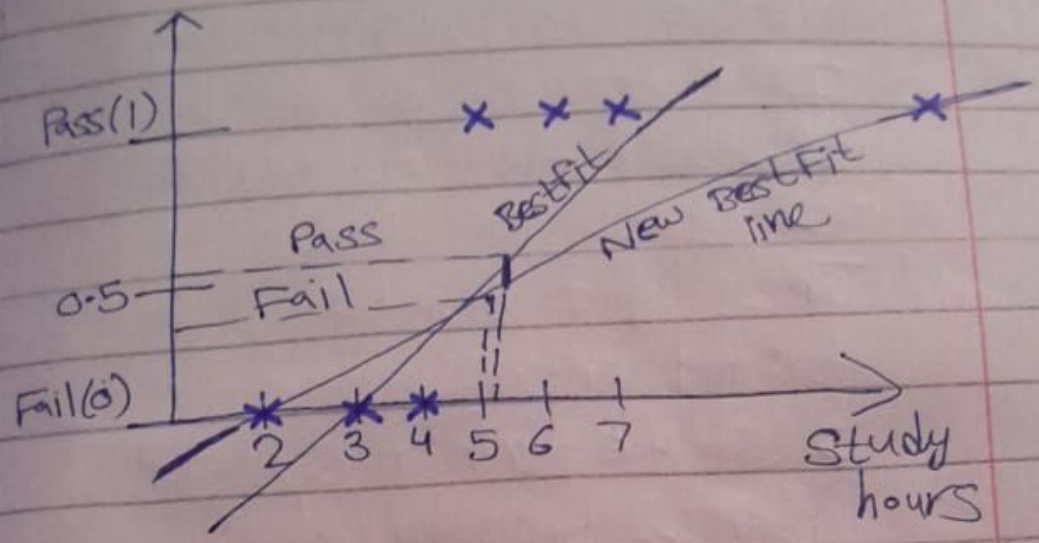
e.g

Dataset — Independent Feature ↑ / Dependent Feature ↑

Study hours	Pass/Fail
2	Fail
3	Fail
4	Fail
5	Pass
6	Pass
7	Pass



- Can we solve this problem using Linear Regression Let's plot it first



Now in Linear Regression we draw a best fit line. We can define a condition that for a new point if prediction is less than 0.5 considered 0 greater than 0.5 considered one

- But problem is that if we add any outlier the whole line will change and even the pass points start failing

⇒ And we can also get value greater than 1 or less than 0

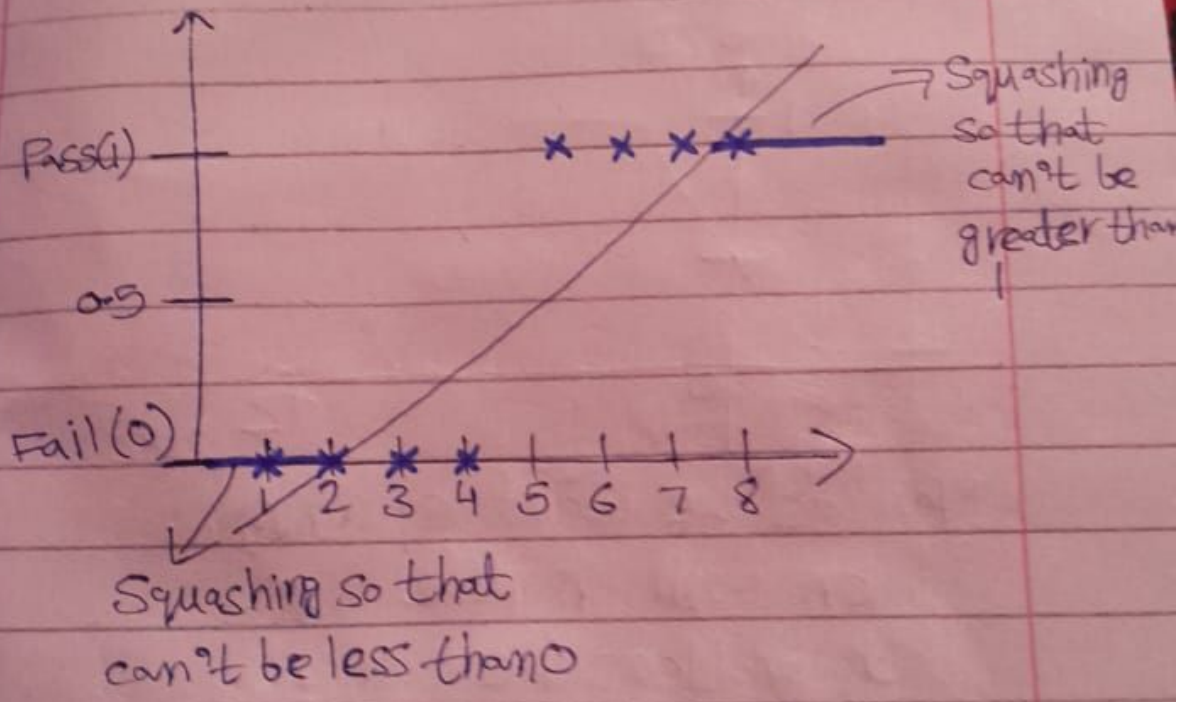
- Why we cannot use linear regression for classification

① If there is any outlier the best fit line already change

② The output can be greater than 1 or less than 0 (so we perform squashing which is possible using Logistic Regression)

⇒ Logistic Regression:

- How logistic regression solves classification problem:
  For not getting output greater than 1 or less than 0 we have to squash our best fit line



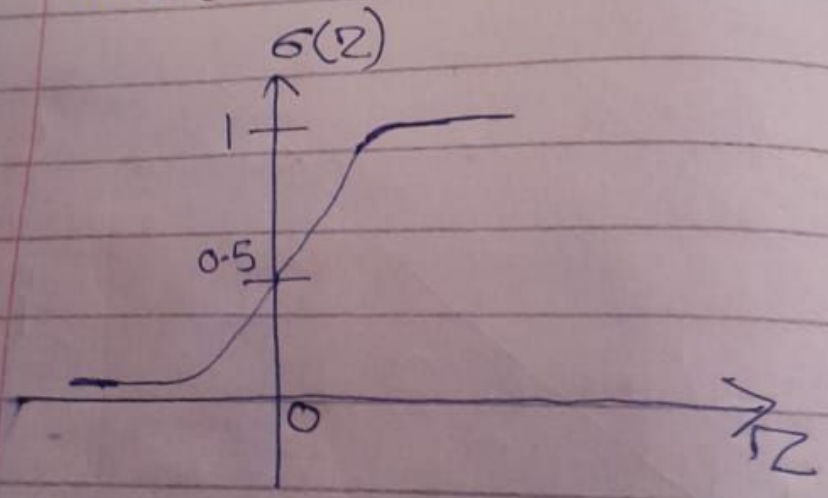
We know equation for best-fit line is:

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

Sigmoid Activation:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

- $0 \leq \sigma(z) \leq 1$
  (Mean never less than 0 or greater than 1)
- $\sigma(z) \geq 0.5$ if $z \geq 0$



- Now as we want **squashing** which will handle our both problems i.e outliers and output less than 0 or greater than 1
  So we will apply **Sigmoid activation** to our best fit line

Now our equation of best fit line after sigmoid activation will become

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

$$\therefore \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\boxed{h_\theta(x) = \frac{1}{1+e^{-z}}} \Rightarrow \text{Logistic Regression hypothesis}$$

where $\boxed{z = \theta_0 + \theta_1 x}$

Linear Regression cost function:

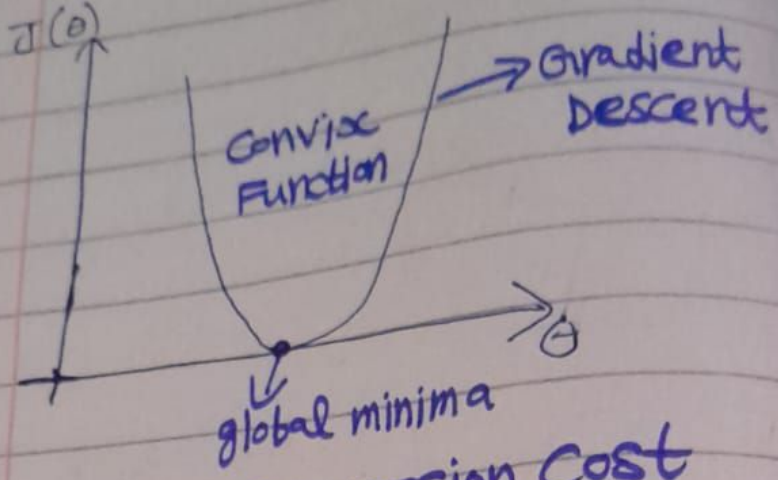
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x)^{(i)} - y^{(i)})^2$$

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

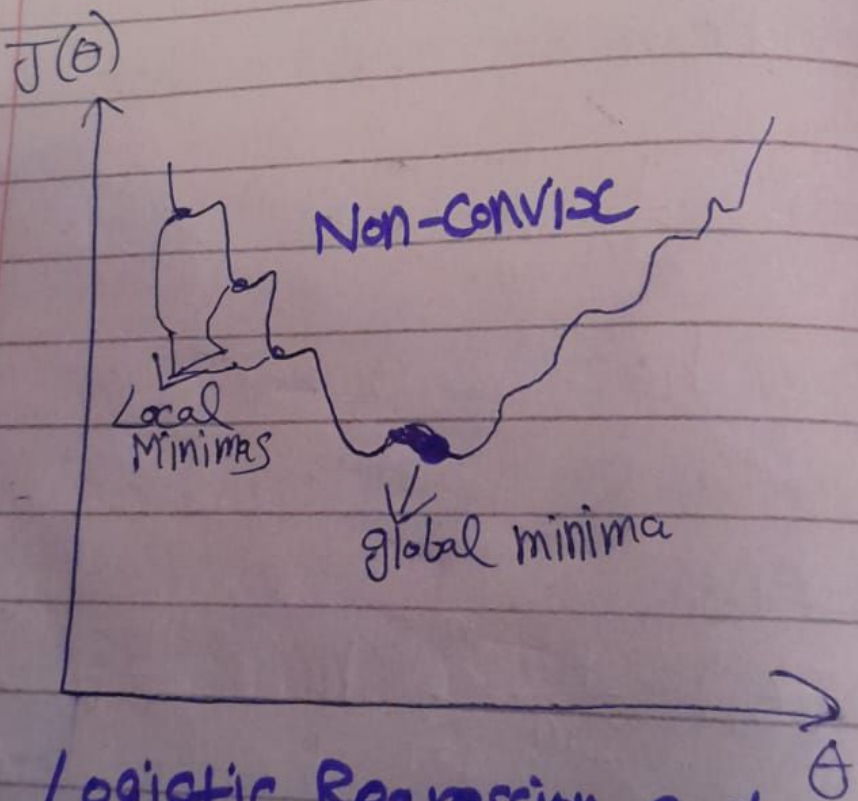
Logistic Regression Cost Function:

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

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



(Linear Regression Cost Function)



Logistic Regression Cost Function

- These local minimas cause problem because at them θ becomes 0

We know Logistic Regression Cost Function

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

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

- Now this is giving us Non-Convex in which due to Local Minimas we are facing problems
- To convert it to Convex we have to make some changes Let that part in our cost function $\text{cost}(h_\theta(x)^{(i)}, y^{(i)})$ we change it:

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

↓

LogLoss (which give us convex function)

$$\text{Cost}(h_\theta(x)^{(i)}, y^{(i)}) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

So Now we will be able to get Convex function (Gradient Descent)

Now let's replace it in our main cost function:

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

where

$$h_\theta(x) = \frac{1}{1+e^{-z}} \; ; \; z = \theta_0 + \theta_1 x$$

- Our aim is to minimize cost function by changing θ_0 and θ_1
- **Convergence Algorithm:**

Repeat till convergence

{

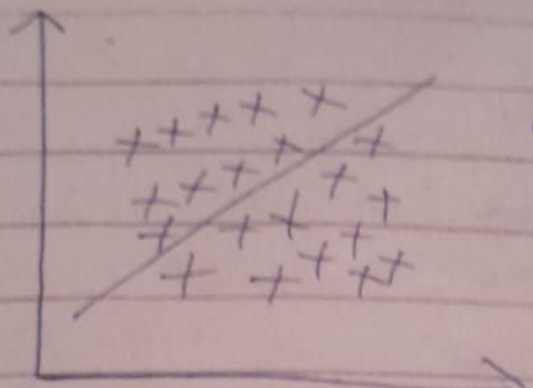
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j}(J(\theta_0,\theta_1))$$

}

$\alpha \Rightarrow$ Learning rate

⇒ Performance Metrics:
Topics to be covered

① Confusion Matrix
② Accuracy
③ Precision
④ Recall
⑤ F-Beta Score



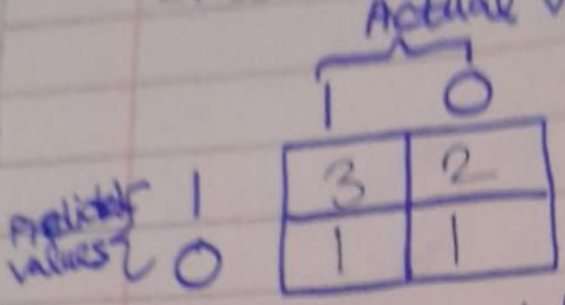
Logistic Regression

Let Dataset

Independent Features		True output	predicted output by Logistic Regression Model
x_1	x_2	y	$\hat{y}$
—	—	0	1
—	—	1	1
—	—	0	0
—	—	1	1
—	—	1	1
—	—	0	1
—	—	1	0

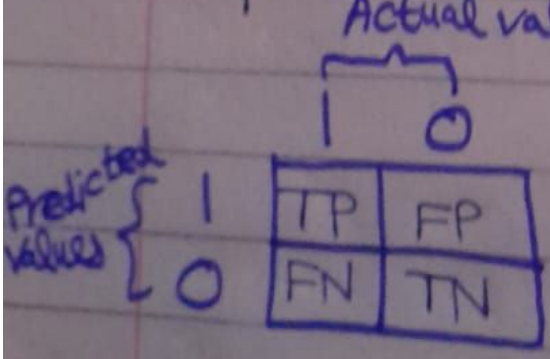
① Confusion Matrix

Confusion Matrix in Binary classification is a 2×2 matrix.



∴ using the dataset let's fill it by adding count

- Now we know that diagonal elements are correct predictions as actual and predicted values are same and Non-Diagonals are wrong predictions.



TP ⇒ True Positive FP ⇒ False Positive
FN ⇒ False Negative TN ⇒ True Negative

- Accuracy:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{Accuracy} = \frac{3+1}{3+2+1+1} = \frac{4}{7}$$

Accuracy = 0.57

Let I have a dataset

Dataset → 1000 datapoints → 900(1)
→ 100(1)

So ratio of output is 9:1 (Imbalanced dataset)

⇓

So we don't use accuracy in such case we use precision and recall

③ Precision:

$$\text{Precision} = \frac{TP}{TP+FP}$$

- It tells out of all actual values how many are correctly predicted

$$\text{Precision} = \frac{3}{3+2} = \frac{3}{5} = \boxed{0.6}$$

④ Recall

$$Recall = \frac{TP}{TP+FN}$$

- Out of all predicted values, how many are correctly predicted

$$Recall = \frac{3}{3+1} = \frac{3}{4} = \boxed{0.75}$$

When should we use precision and when should we use recall

- Usecases 01:

Spam classification

Mail → Spam
Model → Spam
} Good

Mail → Not Spam
Model → Spam
} Blunder

	Spam (1)	Not spam (0)
Spam (1)	TP	FP
Not spam (0)	FN	TN

⇒ Actual

↓ Predicted

- So we identified major blunder which is FP in our case so we have to reduce FP so we use Precision = $\frac{TP}{TP+FP}$

Usecase 02:

To predict whether the patient has diabetes or not

Truth → diabetes
Model → Not diabetes
} Blunder

Truth → diabetes
Model → diabetes
} good

Truth → Not diabetes
Model → Diabetes
} Also error (But not that much big blunder)

	Diabetes (1)	Not Diabetes (0)
Diabetes (1)	TP	FP
Not Diabetes (0)	FN	TN

⇒ Actual

↓ Predicted

- So we identified major blunders is FN which we have to reduce so we will use

$$\text{recall} = \frac{TP}{TP+FN}$$

⑤ F-Beta Score:

$$\text{F-Beta Score} = (1+\beta^2)\frac{\text{Precision} * \text{recall}}{\text{Precision}+\text{recall}}$$

1. If FP and FN are both important then $\beta = 1$

$$\text{F1 score} = 2 \times \frac{\text{Precision} * \text{recall}}{\text{Precision}+\text{recall}}$$

⇓ Harmonic Mean

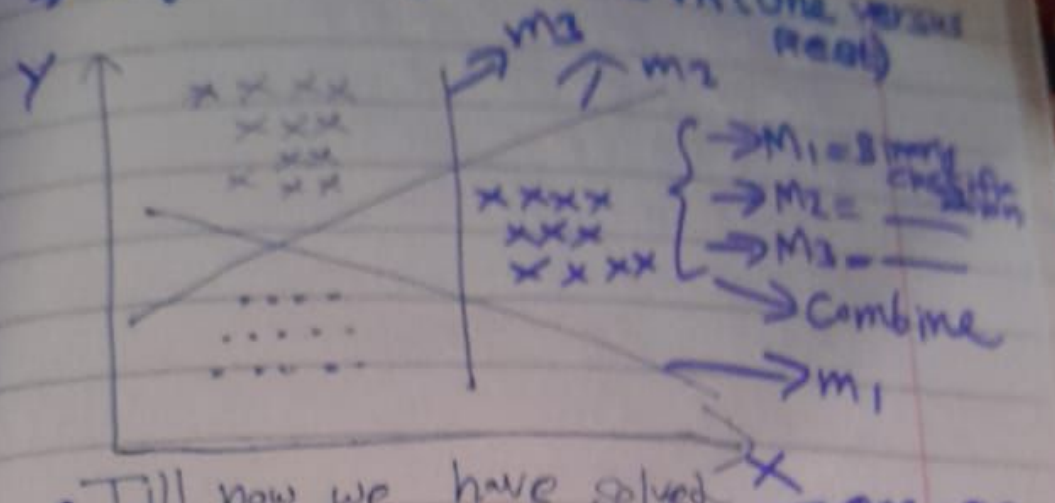
2. If FP is more important than FN then $\beta = 0.5$

$$\text{F0.5 score} = (1+(0.5)^2)\frac{\text{Precision} * \text{recall}}{\text{Precision}+\text{recall}}$$

3. If FN is more important than FP then $\beta = 2$

$$\text{F2 score} = (1+(2)^2)\frac{\text{Precision} * \text{recall}}{\text{Precision}+\text{recall}}$$

⇒ Logistic Regression OVR (one versus rest)



- Till now we have solved
  - Binary classification let one versus one
  - we have more than 2 i.e 3 categories in output for this we will use Logistic Regression (OVR)

- One Versus Rest (OVR)

f1	f2	f3	output(O1, O2, O3)
—	—	—	O1
—	—	—	O2
—	—	—	O3
—	—	—	O3
—	—	—	O1
—	—	—	O2

- In it internally we create models that perform binary classification
- As we have 3 output categories we will make 3 binary classification models and then we combine the output of these models

Let's take example

f_1	f_2	f_3	O/P	O_1	O_2	O_3
—	—	—	O_1	1	0	0
—	—	—	O_2	0	1	0
—	—	—	O_3	0	0	1
—	—	—	O_1	1	0	0
—	—	—	O_3	0	0	1
—	—	—	O_2	0	1	0

Model

$M_1 \leftarrow I/P\{f_1, f_2, f_3\}\ O/P\{O_1\}$
$M_2 \leftarrow I/P\{f_1, f_2, f_3\}\ O/P\{O_2\}$
$M_3 \leftarrow I/P\{f_1, f_2, f_3\}\ O/P\{O_3\}$

New test data
- $M_1 \rightarrow 0.25$ (O_1)
- $M_2 \rightarrow 0.20$ (O_2)
- $M_3 \rightarrow 0.55$ (O_3)

As $M_3 > M_2$ and $M_3 > M_1$

So our output will be O_3