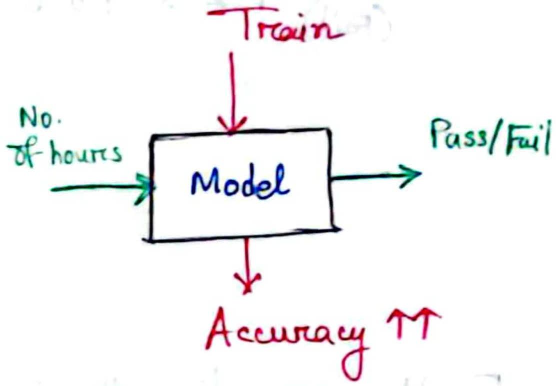
Logistic Regression

It aims to solve classification problems.

Datasets

Independent Feature → No. of play hours; Pass/Fail → Dependent Feature

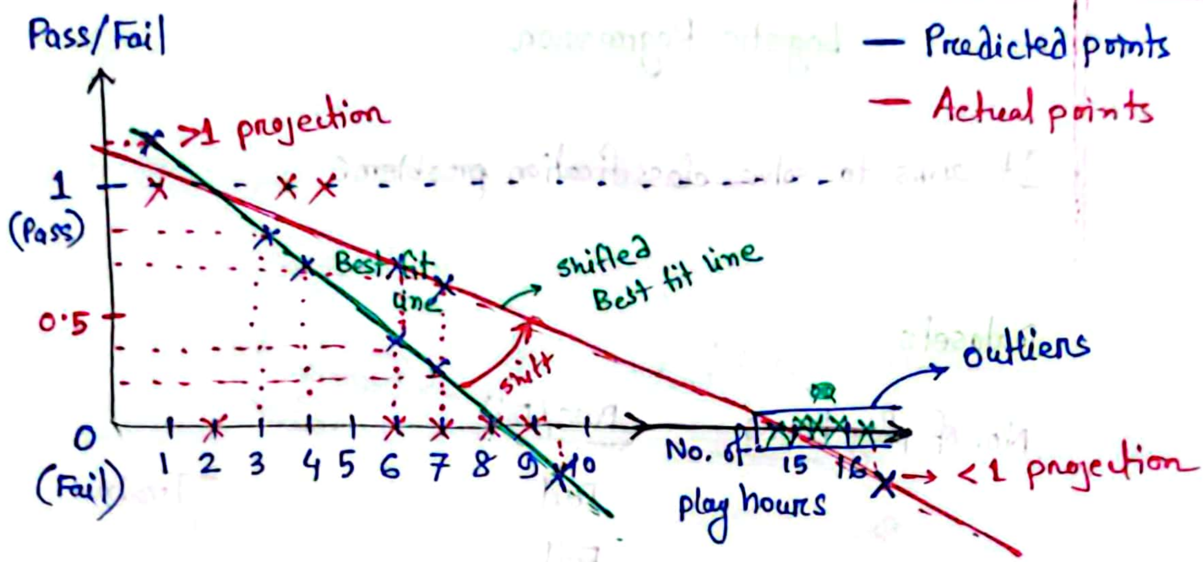
No. of play hours	Pass/Fail
9	Fail
8	Fail
7	Fail
6	Fail
5	Pass
4	Pass
1	Pass
2	Fail



Logistic Regression solves:
- → Binary Classification ***
- → Multiclass classification

Question: Can we solve classification problem using Regression?

Let's plot the graph of linear regression of the above dataset and check what happens.



We took a threshold 0.5
if predicted point >0.5 = Pass
Predicted point < 0.5 = Fail

For this assumption, the best fit line working perfectly.

But if the dataset contains some more values (16, 17, 20) then the Best fit line will shift according to that. So for that shift of line, we could see that, previously for values like (6,7) our prediction was coming correct (<0.5) But now they are getting (>0.5) which means predictions getting wrong now.

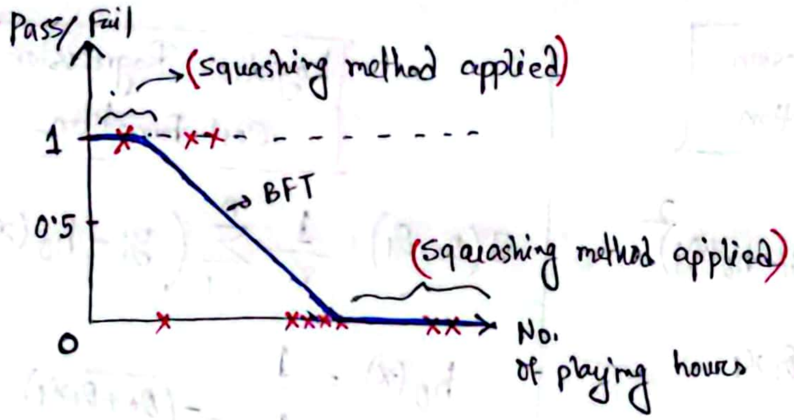
So, key problem here is

→ Best fit line changes because of outliers and predictions get wrong.

→ Also for some values, projected values are getting above 1 and some below 0 which should not be the case.

What logistic regression model will do is, it will not ~~to~~ let the best fit line to shift when the outliers are present also it will use a concept called "squashing" to make the best fit line between 0 to 1 range. Logistic regression "solves" both the problems.

How Logistic regression solves the problem:



Our best fit line equation before was ⇒ $\boxed{h_\theta(x) = \theta_0 + \theta_1 x}$

⇓

To apply squashing, we use "sigmoid activation function" which takes the line equation as input and provide a output whose range is "[0,1]"

$\boxed{\text{Sigmoid Activ. Function}}$

⇓

$\boxed{\text{Output } [0,1]}$

So, "Sigmoid Function" is responsible for ~~washing~~ squashing

Notation of sigmoid Function:

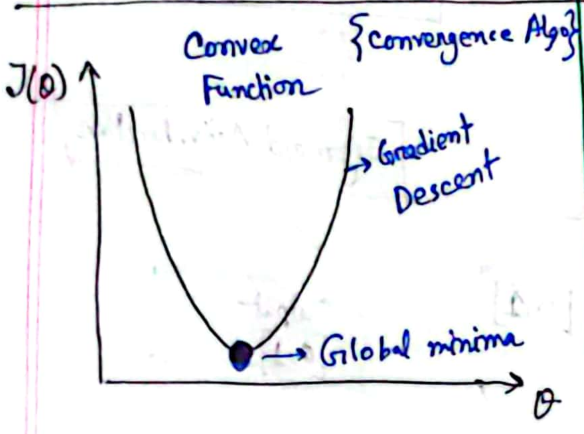
$$\sigma = \frac{1}{1+e^{-z}} \quad \Big| \quad z = h_\theta(x) = \theta_0 + \theta_1 x$$

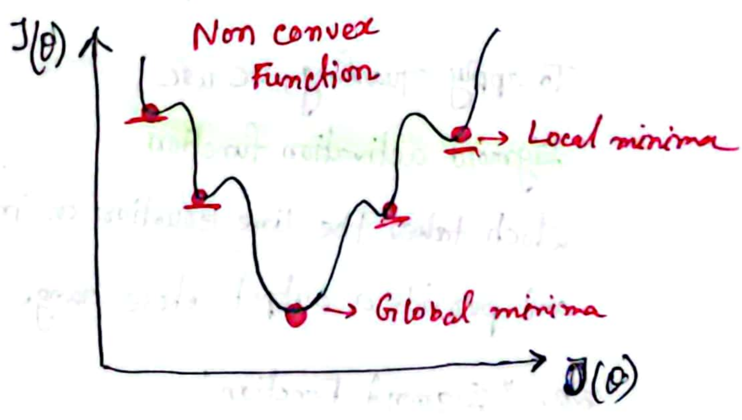
So, for logistic Regression, $\boxed{h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1)}$

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

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

Linere Regression Cost Function	Logistic Regression Cost Function
$J(\theta_0,\theta_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x)_i\right)^2$	$J(\theta_0,\theta_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x)_i\right)^2$
$h_\theta(x) = \theta_0 + \theta_1 x_1$	$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$





For logistic regression we can't use this cost function as we can see the function plot has a lot of local minima. So, it is almost always possible for θ to stuck in one of the local minima and not getting into the global minima. That's why for Logistic Regression we use "Log loss cost function" which will provide only one global minima

Log Loss Cost Function:

$$J(\theta_0,\theta_1) = \begin{cases} -\log(h_\theta(x)) & \text{when } y=1 \\ -\log(1-h_\theta(x)) & \text{when } y=0 \end{cases}$$

Combining those we can write:

This is also a convex function

$$J(\theta_0,\theta_1) = \boxed{-y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))}$$

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

if, $y=1$, then, $J(\theta_0,\theta_1) = -\log(h_\theta(x))$

if, $y=0$, then $J(\theta_0,\theta_1) = -\log(1-h_\theta(x))$

Final aim is to "minimizing the cost function"

→ by changing (θ_0,θ_1)

→ By using convergence Algorithm

CamScanner

(Ridge)

Logistic Regression Cost Function with L2 Regularization:

→ This method is used to "Reduce overfitting"

Formula →

$$J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \underbrace{\lambda \sum_{i=1}^{n} \left(\underset{\theta_1}{\text{slope}}\right)^2}_{L_2 \text{ Regularization}}$$

Logistic Regression Cost Function with L1 Regularization:

Formula →

$$J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \underbrace{\lambda \sum_{i=1}^{n} \left|\underset{\theta_1}{\text{slope}}\right|}_{L_1 \text{ Regularization}}$$

Logistic Regression Cost Function with Elastic net: (L1 + L2) Regularization

Formula: →

$$J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \underbrace{\lambda \sum_{i=1}^{n} \left(\underset{\theta}{\text{slope}}\right)^2 + \lambda \sum_{i=1}^{n} \left|\underset{\theta}{\text{slope}}\right|}_{L_1 + L_2 \text{ Regularization}}$$

In scikit learn, we can find a parameter 'c' which is →

$$\boxed{c = \frac{1}{\lambda}}$$ inverse Relationship