Linear Regression Principle

16 August 2023 15:50

Linear Regression is a fundamental supervised machine learning algorithm employed for making predictions on continuous numerical outcomes. This technique involves establishing a linear connection between various data points within a dataset, allowing it to infer and quantify the relationships between input features and the target prediction. By identifying the best-fitting straight line through the data, Linear Regression enables accurate extrapolation and interpolation of values, making it a powerful tool for forecasting and understanding the interplay between variables in real-world scenarios.

Let's learn the working of linear regression using example -

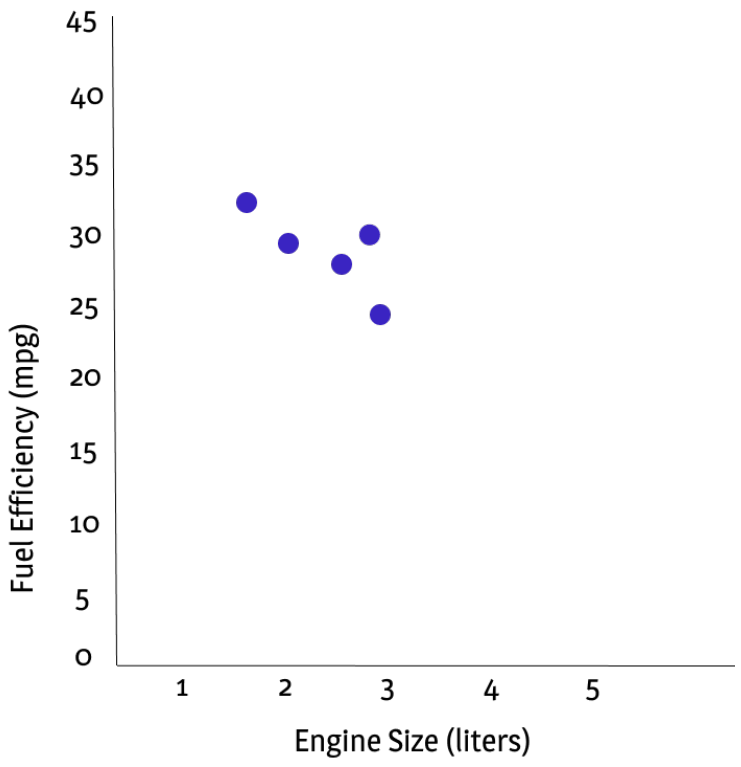
Cars Fuel Efficiency 🚕 🚕

Engine Size (liters)	**Fuel Efficiency (mpg)**
2.0	30
2.5	28
1.8	32
3.0	25
2.2	29

The data, about which we are going to discuss is about car fuel efficiency. There are two columns , **Engine Size (liters)** and **Fuel Efficiency (mpg).** Here, linear regression tries to find a linear relation between two attributes such that it can accurately predicts the next **Fuel Efficiency (mpg)** value.

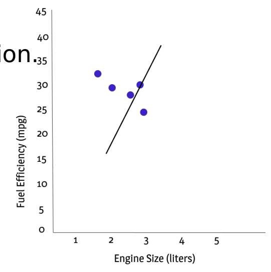
Engine Size (liters) is an independent feature and **Fuel Efficiency (mpg)** is dependent feature or target feature as we are predicting it.

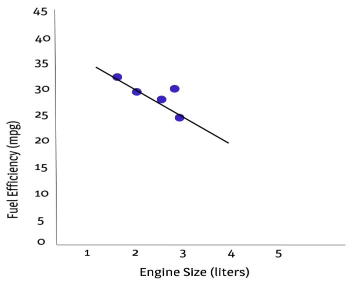
First, plot this data on the graph like below :

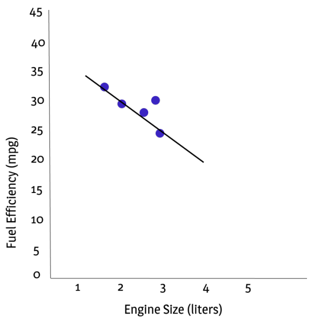


After plotting, linear regression tries to fit the best line on the datapoints such that almost each datapoint will intersect with the line. For example we can fit many lines on the datapoints like below :

Below, we can there is an inverse proportion relation.







So there are many possible lines that we can fit on the datapoints, **Question** arises that how we choose the best fit line which can predict the future accurately most of the times.

Before that, let's see some mathematical concepts -

Equation of Line :-

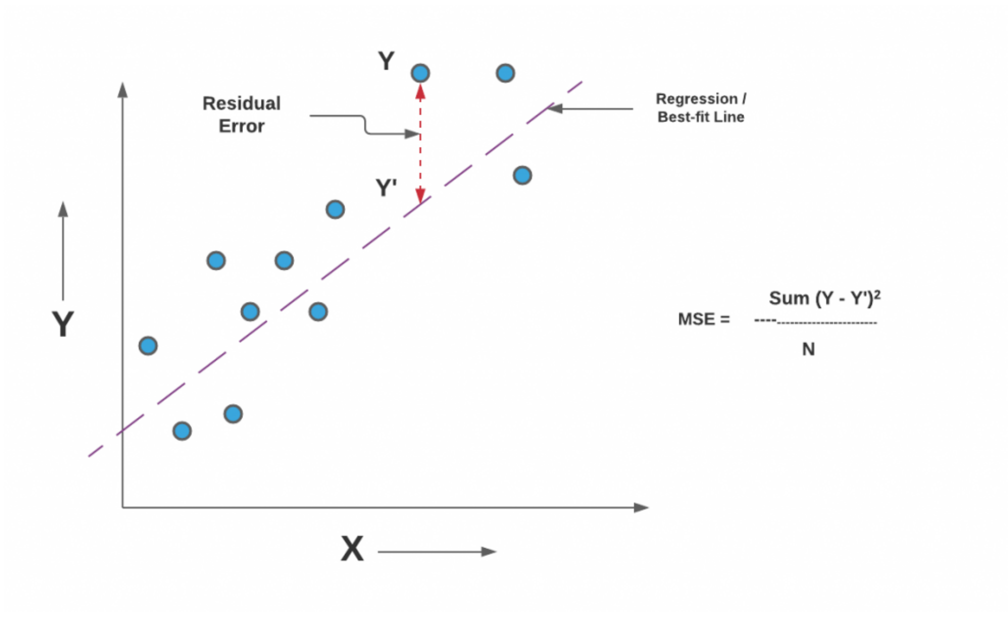
y = mx + c , here

- Y = **output** value which model is going to predict
- m = **slope or inclination** of the line that model is fitting corresponding to independent axes (In this case **x-axis**)
- c = **intercept** is the distance on axis from origin where line is intersecting

Now, we'll use this equation more and more in the concept of linear regression. Let's answer the question above :-
To make the best fit line, we'll iterate through different lines by changing **slope** and **intercept** of line and measures the **error** of that line using **mean squared error function** and The line which gives the minimal error becomes our best fit line.

Some other **questions** arises now :

Error function is a mathematical method through we'll calculate the difference between actual output value and predicted output value. So to minimize it means actual output value is very near to equal to predicted output value. Let'l look at the below diagram



Here, blue datapoints are the actual output which we are calling it **Y** and predicted output value lies on the best fit line which we are calling it **Y'** and the difference between actual and predicted is called **residual error** . Now calculating the difference is not that much simple like calculating signed difference because it results in negative value as well.

Actual output is obtained through our dataset and predicted output is calculated using our new line equation which is **y = mx+ c**.

There are many error calculating functions like **mean squared error**, **ordinary least squared error**, r**oot mean squared error** etc. Let's look at **mean squared error function** to calculate the residual error then proceed to minimizing it.

Formula of Mean Squared Error is :-

MSE = (1/n) * Σ(y' - y)^2, here

We are calculating the difference of predicted and actual output value of each datapoints and squaring it (to remove negative values) then taking average of these differences.

Ok , so now we have a clarity first we'll start by creating random lines and calculates **MSE** for each lines and select the line which is giving us minimum error. Cool 👍 👍

But are we going to build lines randomly ??, but upto how many times, right ??

Trying out different lines means selecting different values of **m (slope)** and **c (intercept)** everytime right ??

Why not we'll try to select some optimal values of these everytime using some method so that we'll reach to our minimal error function quickly and got our optimized parameters I.e. **m** and **c** .

Here, comes **Gradient Descent Method**

Gradient Descent is an optimization technique to minimize the loss function iteratively. It involves updating the model parameters in the opposite direction of the gradient of the loss function to gradually converge to the optimal values.

- Initialize **m** and **b** with random values (For e.g. **m** = **b** = 0 {initial case}).
- Calculate the gradients of the loss with respect to **m** and **b**.
- Update **m** and **b** using the gradients and a learning rate (step size).

Repeat the above steps until the loss converges to a minimum.

Here, according to the second point, after initializing m and c parameters, we'll calculate partial derivate of error function with respect to m and c also like below :-

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

Derivative with respect to m

$$D_c = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)$$

Derivative with respect to c

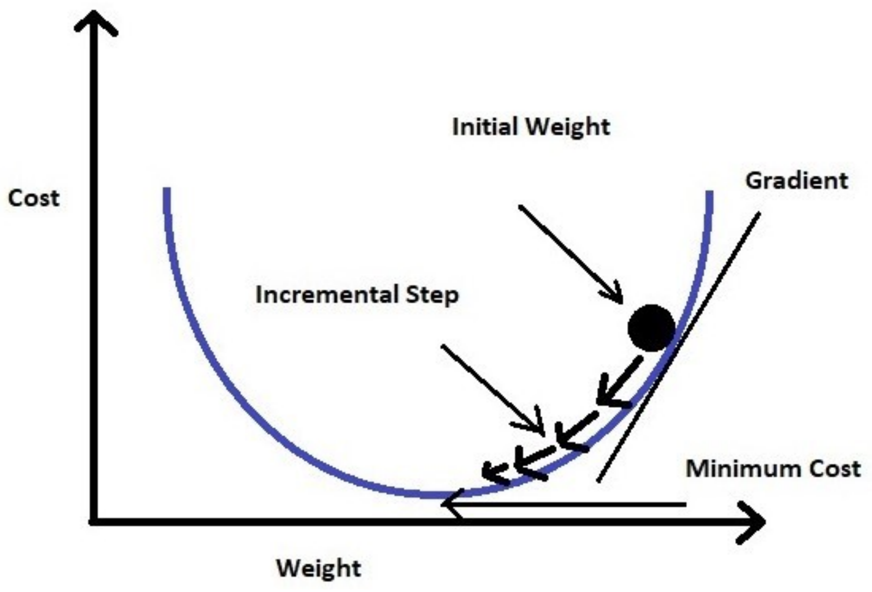
Now, after calculating these values, update **m** and **c** again using below formula :-

$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

Update slope and intercept

Do these two steps until we get the value of cost / error function near to zero. If we plot the value of cost function against weight (m) / bias (intercept), it looks like a bowl shaped curve.



Gradient Descent