**LINEAR REGRESSION**

Definition:- In the most simple words, **Linear Regression** is the supervised Machine Learning model in which the **model finds the best fit linear line between the independent and dependent variable**

(or)

It finds the linear relationship between the dependent and independent variable

Advantages of linear regression:-

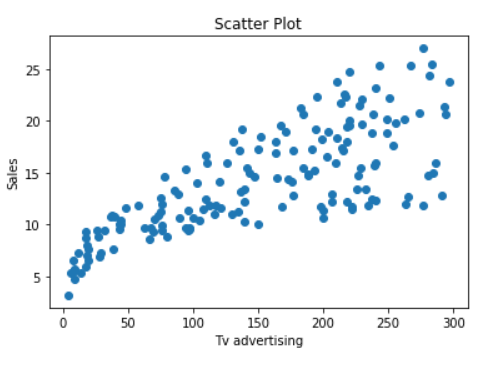
* Linear Regression is simple to implement and easier to interpret the output coefficients.
* If we know the relationship dependent and independent variable if the relationship is linear then this algorithm is best to use because of its less complexity when compared to other algorithm
* It  is the extrapolation beyond a specific data set

Disadvantages of linear regression:-

* Linear regression is quite sensitive to outliers
* Data must be independent
* It only looks at mean of dependent variable

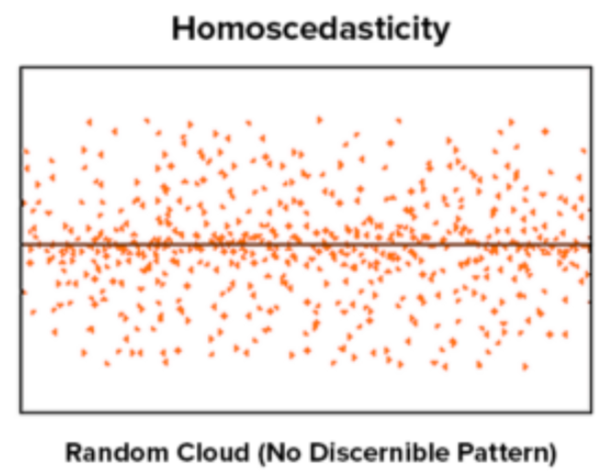
Assumptions of linear regression:-

1. **Linear relationship:-**  It captures the linear relationship between the feature and target. This can be validate by plotting a scatter plot between feature and the target

****

The above scatter plot of the feature TV vs Sales tells us that as the money invested on Tv advertisement increases the sales also increases linearly

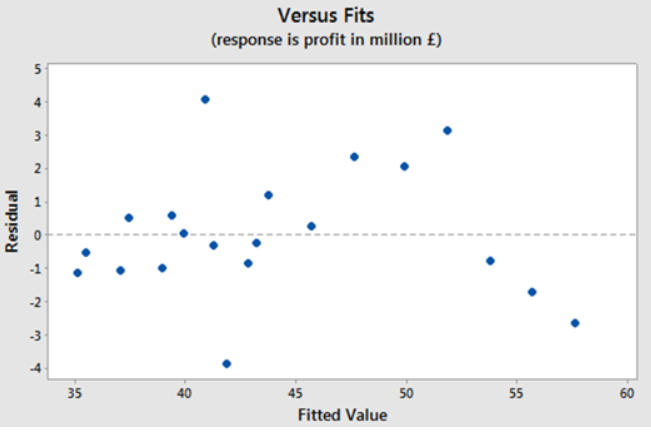
1. **Homoscedasticity:-**  It describes a situation in which the noise or random disturbance in the relationship between the feature and target is the same across the all values of independent variables



From the above graph we can observe that there is no specific pattern that is no constant among the

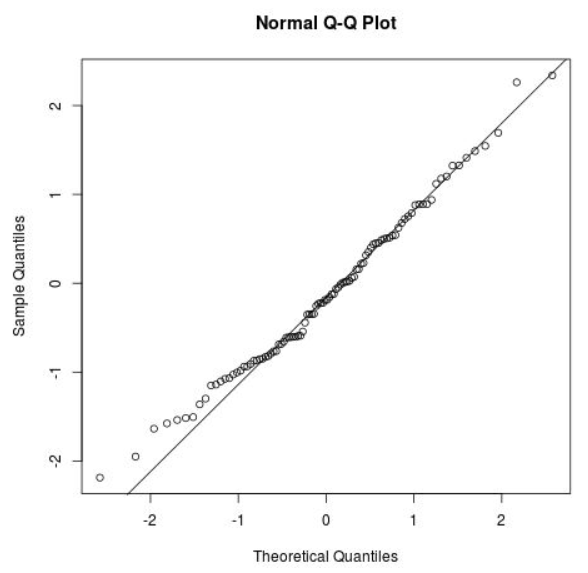
residuals

1. **Independence:-**  Here the residuals are independent, this is mostly relevant when we are working with the time series data. We don’t want any pattern among the consecutive residuals.



From the above we cannot have any positive auto correlation because we cannot see any cyclic pattern. Also it does not show any negative auto correlation.

1. **Normality:-** In this the residuals are normally distributed . We can check is assumption visually by using Q-Q plot means quantile-quantile plot it is a type of plot where we determine whether the residual model follows a normal distribution or not.



From the above Q-Q plot we can say that the residuals roughly follows normal distribution

**Types of linear regression:-** There are two types of linear regressions they are 1) Simple linear regression

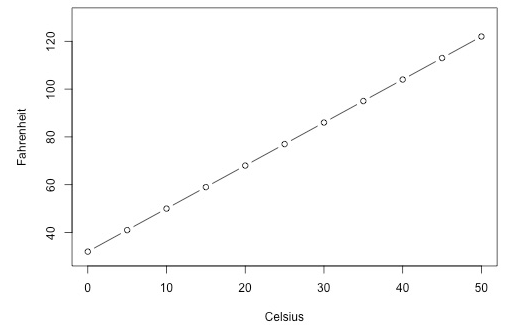
2) Multiple linear regression

**Simple linear regression:-** With simple linear regression when we have a single input, we can use statistics to estimate

the coefficient . This requires that you calculate statistical properties from the data such as means, standard deviations,

correlations and covariance. All of the data must be available to traverse and calculate statistics.

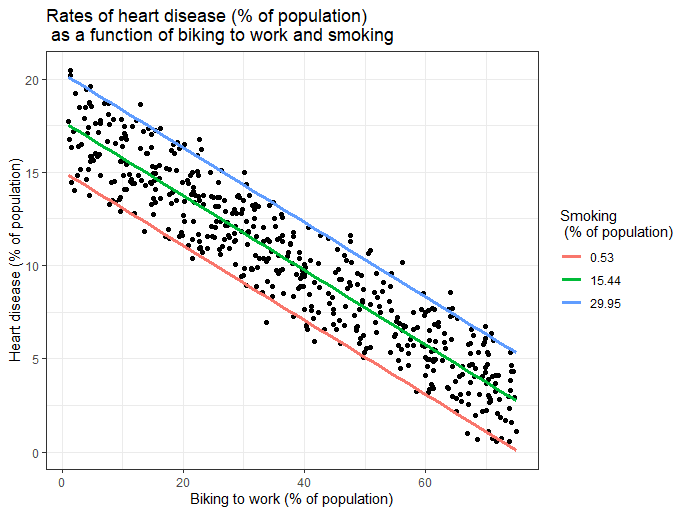
The equation of simple linear regression is y=mx+c here ‘x’ is independent variable and ‘y’ is dependent variable, ‘m’ is slope and ‘c’ is intercept



The above graph is example of simple linear regression because the both dependent and independent variables are linearly increasing

Multiple linear regression:- **Multiple linear regression** is used to estimate the relationship between **two or more independent variables**and**one dependent variable**.

Formula for multiple linear regression is **y=mx1+mx2+……+mxn+c** if we observe means y is one dependent variable and x is independent variable which has many.



Above graph is example for the multiple linear regression it explains about the rates of heart disease as a function of biking to work and smoking

**Metrics Evaluation and use of it :-** In simple words we can say metrics is used to evaluate the performance of the model, there are totally 6 metrics in linear regression they are **1) R Squared**

**2) Adjusted R Square**

**3) Mean Square Error (MSE)**

**4) Mean Absolute Error (MAE)**

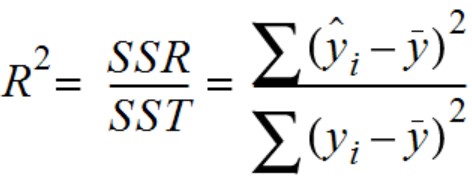
**5) Root Mean Squared Error (RMSE)**

**6) Root Mean Squared Log Error (RMSLE)**

**Why we use Metrics evaluation :-** Machine learning model cannot have 100 percent efficiency otherwise the model is known as a biased model. which further includes the concept of overfitting and underfitting.

So to build and deploy a generalized model we require to Evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result.

**R Squared:-** R Square is a metric that tells the performance of your model, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. basically R squared calculates how must regression line is better than a mean line. We can say R Square is also known as goodness of fit.



The above is the formula for R Square. Here **ŷ** predicted value and **ȳ** means mean of y.

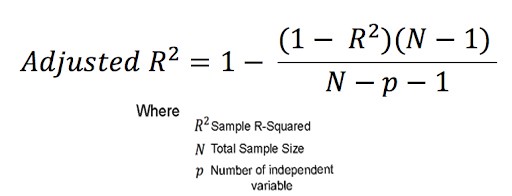
from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test,y\_pred)

print(r2)

it is code to evaluate R Squared metrics

**Adjusted R Squared:-** The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases. But the problem is when we add an irrelevant feature in the dataset then at that time R2 sometimes starts increasing which is incorrect. Hence, To control this situation Adjusted R Squared came into existence



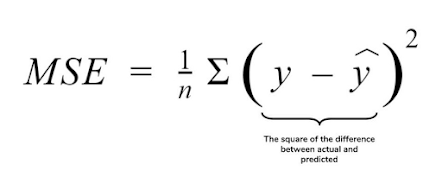
The above is the formula for Adjusted R Squared.

adj\_r2\_score = 1 - ((1-r2)\*(n-1)/(n-p-1))

print(adj\_r2\_score)

the above is code to evaluate the Adjusted R Squared

**Mean squared Error:-** MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value. It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.



The above is the formula of Mean Squared Error

from sklearn.metrics import mean\_squared\_error

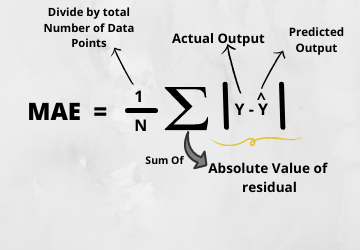
print("MSE",mean\_squared\_error(y\_test,y\_pred))

The above is the code to evaluate the Mean Squared Error

**Mean Absolute Error:-** MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

To better understand, let’s take an example you have input data and output data and use Linear Regression, which draws a best-fit line. Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.

so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss



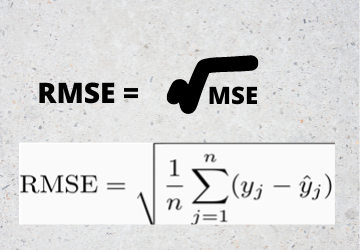
The above is the formula for mean absolute error

from sklearn.metrics import mean\_absolute\_error

print("MAE",mean\_absolute\_error(y\_test,y\_pred))

The above is the code to evaluate the Mean Absolute Error

**Root Mean Squared Error:-** It is nothing but a square root of mean squared error

 This is the formula for Root Mean Squared Error.

print("RMSE",np.sqrt(mean\_squared\_error(y\_test,y\_pred)))

It is the code evaluate the RMSE

**Root Mean Squared Log Error:-** Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.

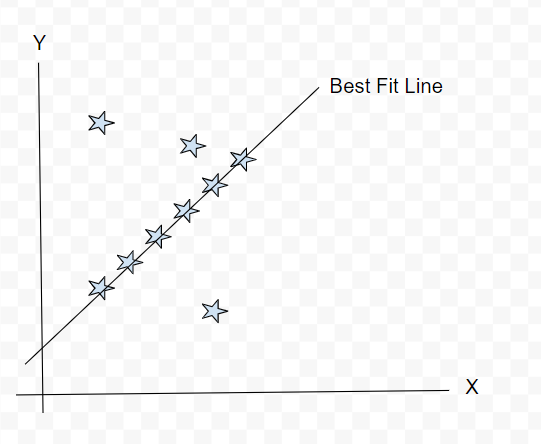
To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE

print("RMSE",np.log(np.sqrt(mean\_squared\_error(y\_test,y\_pred))))

The above is the code to evaluate the RMSLE

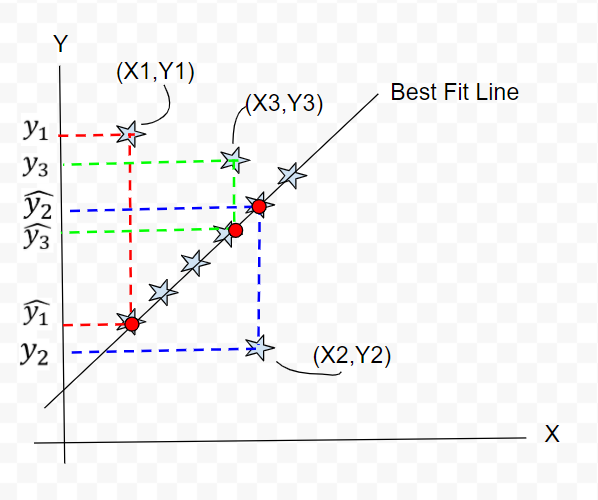
**Mathematical intuition in Linear regression:-** The following are the steps to find the mathematical intuition in linear regression

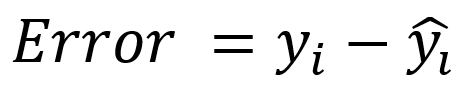
**Step 1:-**  **Plot of the Independent & Dependent Variables. Draw the best fit line (Approx.).**



**Step 2:-** **Calculate the individual errors.** Error is defined based on the actual & the predicted value.

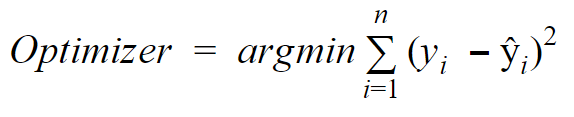
In our Sample Example, we have 3 points which are not in line with our best fit line. So we need to calculate the error function





**Step 3:-** **Calculating the minimum sum of squares of errors or Ordinary Least-squares.**

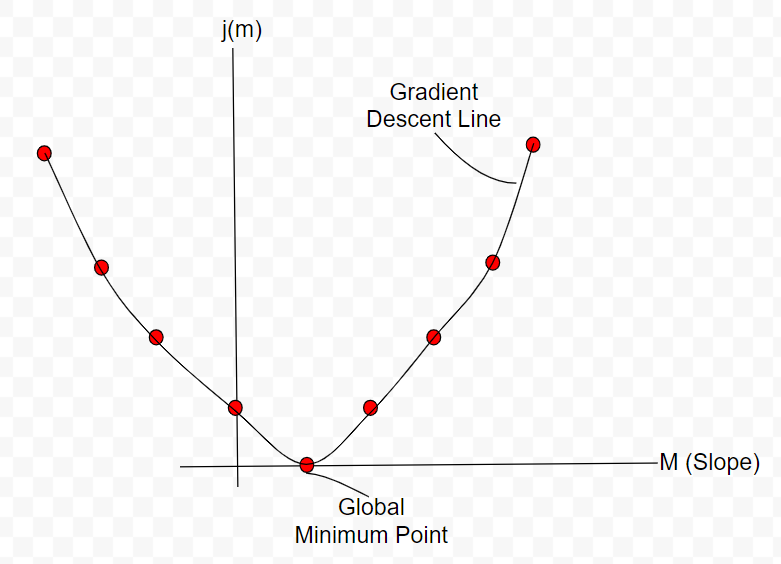
As we have calculated the individual Error for all the points, we are now going to sum & consider the min value to evaluate the best fit line by using the below formula.

 Once we have our Best Fit Line on the Linear Model, we are having the freedom to predict the dependent values based on the given independent variables

**What’s the basis of selecting the Best Fit Line?**

The cost function formula will provide the errors individually based on the predicted & actual variables. But we can’t directly select the best fit line based on some visual exploration.

In the above formula, we will have an independence of exploring the slope (m) into different values. Based on the varying m value we will arrive at the Gradient Decent point for finding the global minimum point, below fig for quick reference.



In the prediction of a target variable, we will be knowing the correspondence X value (input function) based on this by using the global minimum point we are going to plot the Best Fit Line.

**Gradient Descent:-** Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost)

It is used to minimise the mean squared error by calculating the gradient of the cost function.

**Stochastic Gradient Descent-**

Gradient descent can be slow to run on very large datasets.

Because one iteration of the gradient descent algorithm requires a prediction for each instance in the training dataset, it can take a long time when you have many millions of instances.

In situations when you have large amounts of data, you can use a variation of gradient descent called stochastic gradient descent.

In this variation, the gradient descent procedure described above is run but the update to the coefficients is performed for each training instance, rather than at the end of the batch of instances.

**Lasso regression:-** **Lasso**is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros. During training, the objective function become

**Ridge regression:-** Ridge takes a step further and penalizes the model for the sum of squared value of the weights. Thus, the weights not only tend to have smaller absolute values, but also really tend to penalize the extremes of the weights, resulting in a group of weights that are more evenly distributed.