# Logistic Regression

Classification Steps:

1. Create the dataframe properly-->pd.read\_csv(),pd.read\_excel()
2. Preprocessing the data:
   1. Feature Selection-->domain knowledge-->drop()
   2. Handling the missing values-->isnull().sum(),fillna(),dropna()
3. Converting the categorical data into numerical-->map(),get\_dummies(),LabelEncoder()
4. Create X and Y-->X=ind var, Y=dep var-->X=df.values[:,:-1], Y=df.values[:,-1]
5. Scale the data[Optional]-->MinMaxScaler(),StandardScaler()
6. Splitting the data into Train and Test(validation)-->train\_test\_split()
7. Build the model:
   1. Create the model-->obj=AlgoName()
   2. Train the model-->obj.fit(X\_train, Y\_train)
   3. Predict using the model-->Y\_pred=obj.predict(X\_test)
8. Evaluate the model:
   1. confusion\_matrix(Y\_test,Y\_pred)
   2. accuracy\_score(Y\_test,Y\_pred)
   3. classification\_report(Y\_test,Y\_pred)
9. Tuning the model:
   1. Feature Selection
   2. Dedicated approach-->Adjusting the threshold
   3. Stochastic Gradient Descent

Model is ready--classifier-->adult\_data.csv

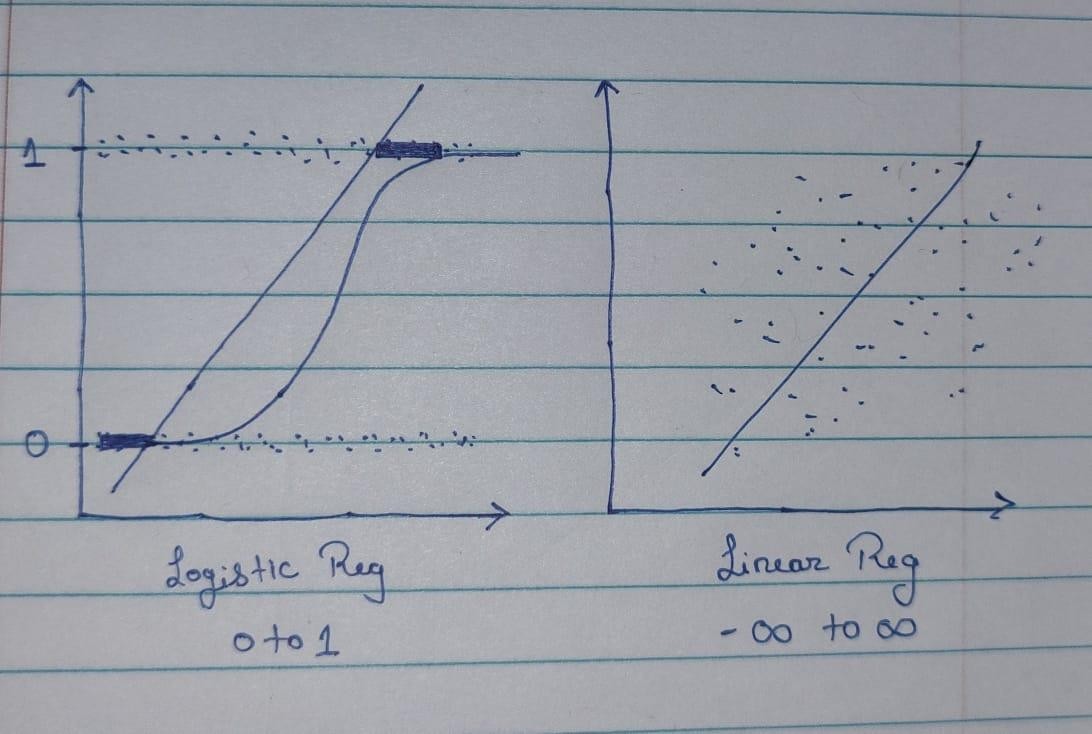
Follow the steps on the adult\_test.csv:

1. Create a dataframe properly-->adult\_test
2. Preprocessing the data:
   1. Feature selection-->eliminate fnlwgt,education
   2. Handling the missing values
3. Converting categorical values to numerical
4. Create X\_test\_new and Y\_test\_new
5. Scaling the data-->X\_test\_new-->X\_test\_new=scaler.transform(X\_test\_new),fit() not to be implemented
6. Y\_pred\_prob=classifier.predict\_proba(X\_test\_new)
7. Use the if-else code with threshold=0.46(optimum) and generate Y\_pred\_new
8. Evaluating the model:
   1. confusion\_matrix(Y\_test\_new,Y\_pred\_new)
   2. accuracy\_score(Y\_test\_new,Y\_pred\_new)
   3. classification\_report(Y\_test\_new,Y\_pred\_new)

* Logistic Regression Used for Classification
* It is Supervised Machine Learning algorithm
* Only Suitable for binary Classification
* It is not Suitable for Multiple Classes

In linear regression we have range from -∞ to +∞, but in logistic Regression we have range from 0 to 1 as we are working on binary classification of data, so the data points would be either at 0 or 1 which makes the best fit line from linear regression not suitable for such type of data, so the solution to this would be any value going above one would be considered as 1 and any value going below zero would be considered as 0 , which again gives rise to a problem of having three straight lines, so this was also solved by making it as a S- shaped curve giving rise to the sigmoid function.

**Sigmoid Function –** It allows to convert any real value data to the range of 0 to 1

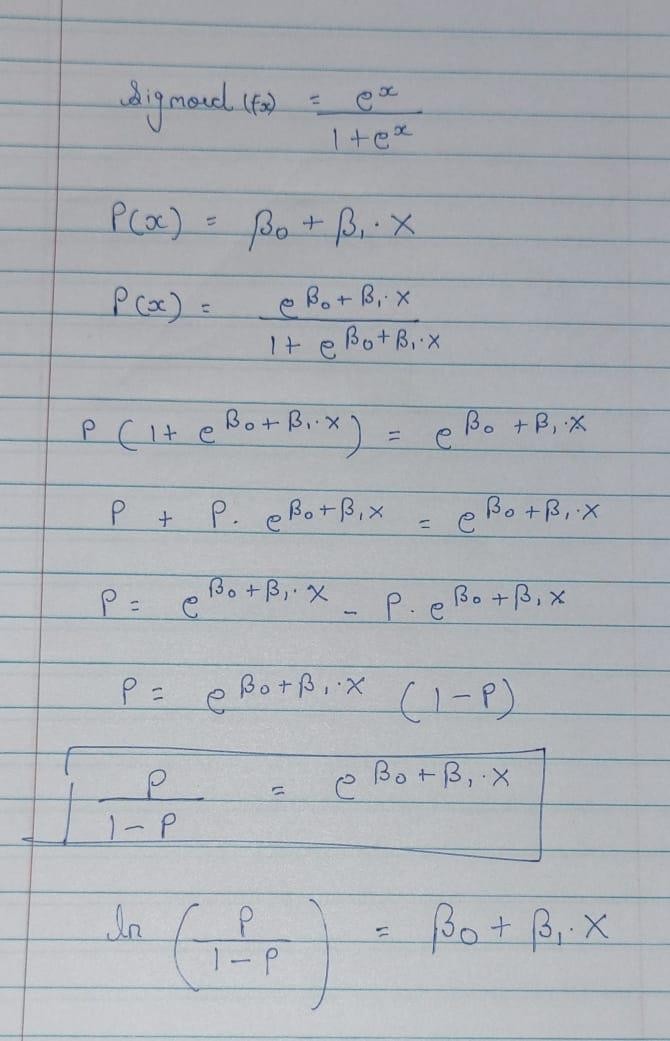


𝑒𝑥

Sigmoid f(x) =

1+𝑒𝑥

# Logistic regression equation is derived from the sigmoid function



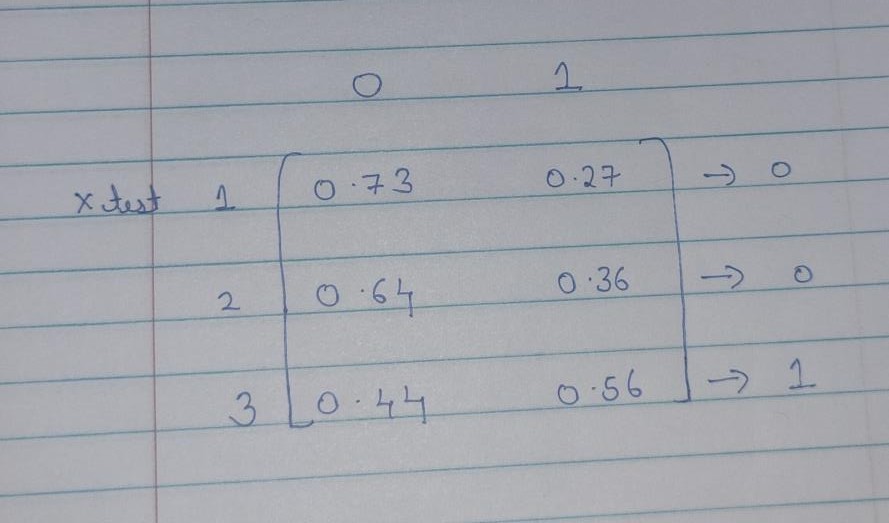
Were,

* + P/1 - P = Odd’s Ratio  probability of success divided by the probability of failure
  + We take log to eliminate ‘e’ to get the linear regression equation hence the name **Logistic regression**
  + when we take log the LHS is called as **logit function** So while performing logistic regression in the background we have Linear Regression running, it is because of the logit function the values are in the range of 0-1

# Probability Matrix

Some data points are at 0 and some at 1, while doing prediction assumption is midpoint 0.5 is default threshold anything less than 0.5 is belong to class zero and more than 0.5 belong to class 1, however the values being continuous it reaches to 0 or 1 internally with the help of **probability matrix**

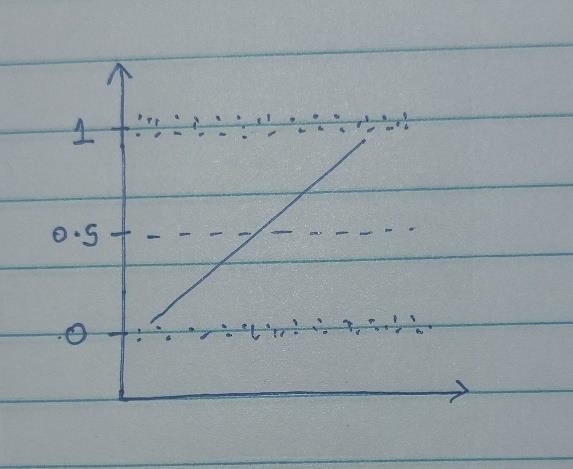
Probability matrix will be generated in the background based on y values Probability matrix is finally responsible for giving the final y value



# Threshold

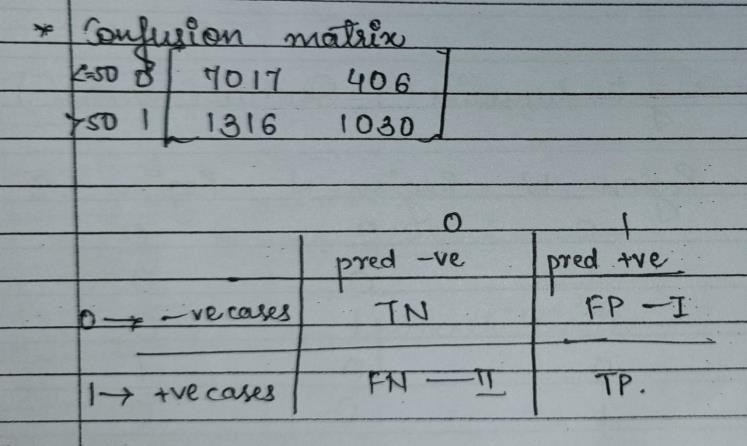
Anything < 0.5 == 0

Anything > 0.5 ==1



# Evaluation

* Confusion Matrix



* + Accuracy Score  percentage of correct predictions, higher the accuracy better the model
    - TN+TP/TP+FP+FN+TP
  + Recall Value  accuracy score of individual classes
    1. Class 0 – TN/TN + FP
    2. Class 1 – TP/TP + FN
  + Precision  It tells you about how relevant are the predictions

1. Class 0 : How many negative predictions are correct?

= TN/TN + FN

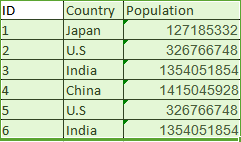
1. Class 1 - How many positive predictions are correct?

= TP/TP + FP

* + F1 Score Value  harmonic mean of precision and recall 2\*(precision\*recall) / precision + recall

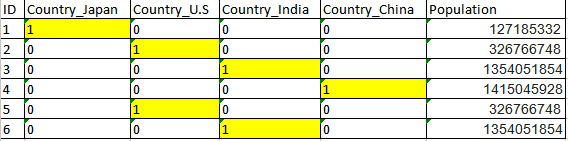
# Encoding (converting categorical data to numeric):

1. Manual Encoding  map()

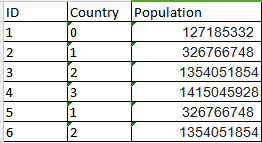


1. Dummy Variable  get\_dummies(), OneHotEncoder()

* one hot encoding takes a column which has categorical data, which has been label encoded and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.
* It becomes more confusing to understand
* No of steps goes on increasing so it is not so preferred function.



1. Creating levels labelEncoder ()
   * Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n\_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.
   * It assigns values to the variable in ascending alphabetical order



# Overfitting and Underfitting:

* Overfitting: Good performance on the training data, poor performance to unseen data.
* Underfitting: Poor performance on the training data poor performance to unseen data.