# **Group A : Data Science**

# **Assignment No : 5**

**Title of the Assignment: Data Analytics II**

1. Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset

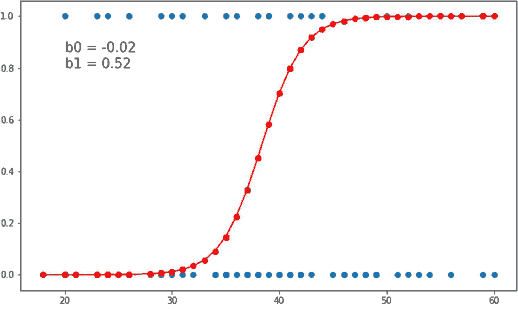
**Objective of the Assignment:**

Students should be able to perform the data Analytics operation using Python on any open source dataset

**Prerequisite:**

1. Basic of Python Programming
2. Concept of Data Analytics

**Contents for Theory:**

1. Logistic Regression
2. Logistic Model
3. Loss Function
4. The Gradient Descent Algorithm
5. Implemention
6. Confusion Matrix
7. **Logistic Regression**

In statistics logistic regression is used to model the probability of a certain class or event.

Logistic regression is similar to linear regression because both of these involve estimating the values of parameters used in the prediction equation based on the given training data. Linear regression predicts the value of some continuous, dependent variable. Whereas logistic regression predicts the probability of an event or class that is dependent on other factors. Thus the output of logistic regression always lies between 0 and 1. Because of this property it is commonly used for classification purpose.

# **Logistic Model**

Consider a model with features *x1, x2, x3 … xn*. Let the binary output be denoted by *Y*, that can take the values 0 or 1.

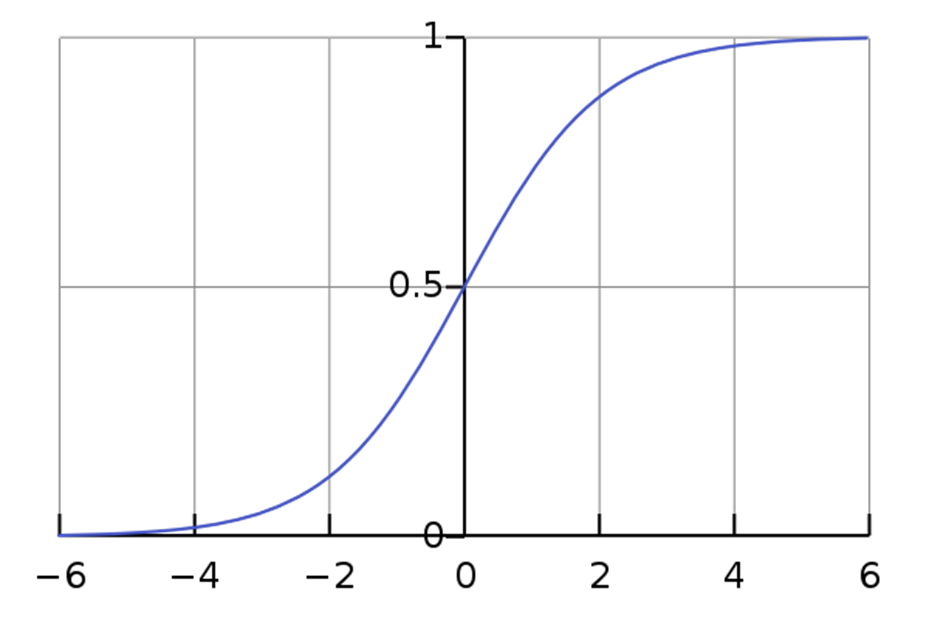
Let *p* be the probability of *Y = 1*, we can denote it as *p = P(Y=1)*.

The mathematical relationship between these variables can be denoted as:

p

ln( 1 − p ) = b0 + b1x1 + b2x2 + b3x3. . . bnxn

S(x) = 1

 1 + e−x

Now we will be using the above derived equation to make our predictions. Before that we will train our model to obtain the values of our parameters b0, b1, b2. . . that result in least error. This is where the error or loss function comes in.

1. **Loss Function**

The loss is basically the error in our predicted value. In other words it is a difference between our predicted value and the actual value. We will be using the L2 Loss Function to calculate the error. Theoretically you can use any function to calculate the error. This function can be broken down as:

1. Let the actual value be yi. Let the value predicted using our model be denoted as Find the difference between the actual and predicted value.

2. Square this difference.

3. Find the sum across all the values in training data.



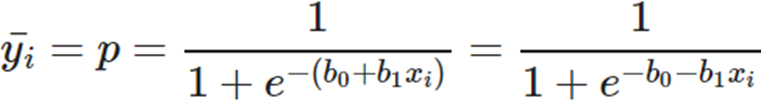
Now that we have the error, we need to update the values of our parameters to minimize this error. This is where the "learning" actually happens, since our model is updating itself based on it's previous output to obtain a more accurate output in the next step. We will be using the Gradient Descent Algorithm to estimate our parameters. Another commonly used algorithm is the Maximum Likelihood Estimation.

1. **The Gradient Descent Algorithm**

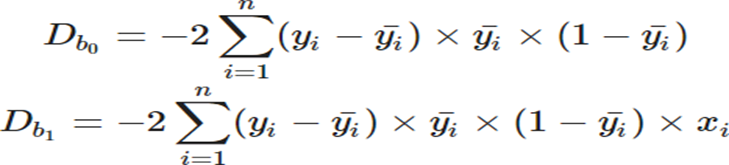
You might know that the partial derivative of a function at it’s minimum value is equal to 0. So gradient descent basically uses this concept to estimate the parameters or weights of our model

by minimizing the loss function.

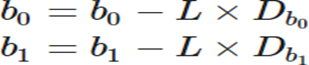
For simplicity, for the rest of this part let us assume that our output depends only on a single feature x. So we can rewrite our equation as:



Thus we need to estimate the values of weights b0 and b1 using our given training data. Initially let b0=0 and b1=0. Let L be the learning rate. The learning rate controls by how much the values of b0 and b1 are updated at each step in the learning process. Here let L=0.001. Calculate the partial derivative with respect to b0 and b1. The value of the partial derivative will tell us how far the loss function is from it’s minimum value. It is a measure of how much our weights need to be updated to attain minimum or ideally 0 error. In case you have more than one feature, you need to calculate the partial derivative for each weight b0, b1 … bn where n is the number of features.



Next we update the values of b0 and b1:



We repeat this process until our loss function is a very small value or ideally reaches 0 (meaning no errors and 100% accuracy). The number of times we repeat this learning process is known as iterations or epochs.

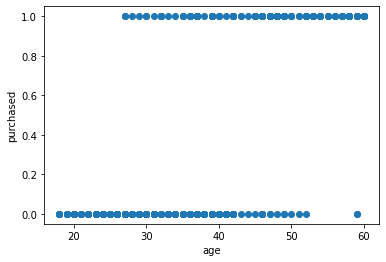
1. **Implemention**
2. **Implementing the Model**

The data describes information about a product being purchased through an advertisement on

social media. We will be predicting the value of Purchased and consider a single feature, Age to

predict the values of Purchased. You can have multiple features as well.





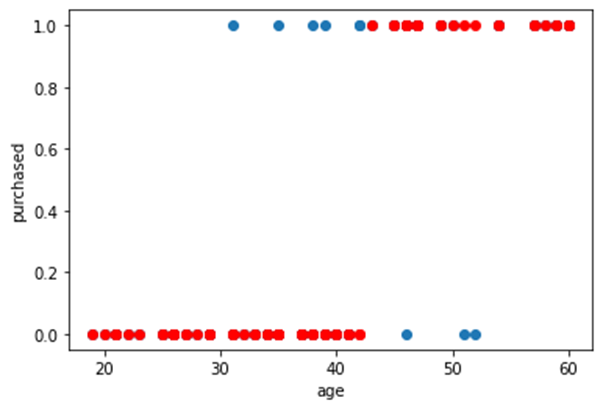
We need to normalize our training data, and shift the mean to the origin. This is important to get accurate results because of the nature of the logistic equation. This is done by the normalize method. The predict method simply plugs in the value of the weights into the logistic model equation and returns the result. This returned value is the required probability.

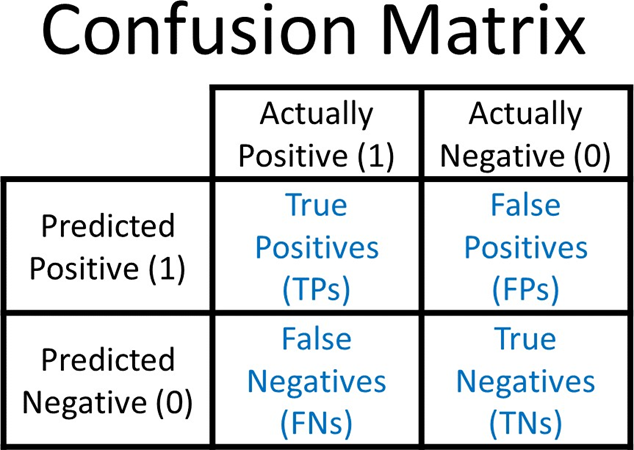
The model is trained for 300 epochs or iterations. The partial derivatives are calculated at each iterations and the weights are updated. You can even calculate the loss at each step and see how it approaches zero with each step.

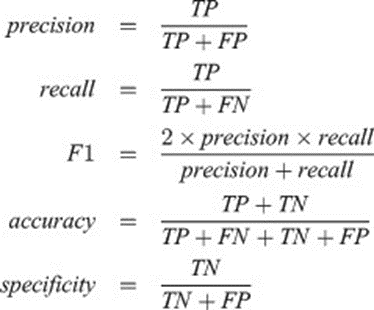
1. **Implementing using Sklearn**

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

The accuracy using this is 86.25%, which is very close to the accuracy of our model that we implemented from scratch !



1. **C****onfusion Matrix**

****

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known

**Some Basic Terms**

**True Positive -**

Label which was predicted Positive and is actually Positive.

**True Negative -**

Label which was predicted Negative and is actually Negative.

**False Positive -**

Label which was predicted as Positive, but is actually Negative. In Hypothesis Testing it is also known as Type 1 error or the incorrect rejection of Null Hypothesis, refer this to read more about Hypothesis testing.

**False Negatives -**

Labels which was predicted as Negative, but is actually Positive. It is also known as Type 2 error, which leads to the failure in rejection of Null Hypothesis.

Accuracy 87.50%:

**Precision**

It is the ‘Exactness’, ability of the model to return only relevant instances. If your use case/problem statement involves minimizing the False Positives, i.e. in current scenario if you don’t want the Forged Notes to be labelled as Authentic by the Model then Precision is something you need.

Precision = tp/(tp+fp)

**Recall**

It is the ‘Completeness’, ability of the model to identify all relevant instances, True Positive Rate, aka Sensitivity. In the current scenario if your focus is to have the least False Negatives i.e. you don’t Authentic Notes to be wrongly classified as Forged then Recall can come to your rescue.

Recall = tp/(tp+fn)

Recall 0.76

**Error rate**

Error rate (ERR) is calculated as the number of all incorrect predictions (FN + FP) divided by the total number of the dataset (P + N). The best error rate is 0.0, whereas the worst is 1.0.



In [11] :

Error rate 0.12

*#Error rate*

err **=** (fp **+** fn)**/**(tp **+** tn **+** fn **+** fp) print("Error rate {:0.2f}"**.**format(err))

**Questions**

Q.1 What is confusion matrix?

Q.2 What is difference between linear and logistic regression?

Q.3 Obtain sigmoid function.

Q.4 Explain gradient descent function.