**Logistic Regression**

It is a type of supervised learning. We will be using Logistic Regression to perform classification of a data set.

We will focus on binary classification, so that we have two outputs, a positive and negative, (1 or 0). For example, we could classify email as either spam or not spam, or tumors as either malignant or benign. There are only two possible outcomes in both these cases.To perform binary classification, we will be using the logistic function to perform logistic regression.

**IMPORTS**

The standard libraries, such as numpy, pandas, Series, Dataframe are imported. Also the math module and the data visualization modules like matplotlib.pyplot, seaborn etc are also imported. A new module to be imported is called ‘Statsmodels’ from which we will be using the dataset. To evaluate our ML results, we also import metrics, LogisticRegression, train\_test\_split from sklearn.

**Basic Mathematical Overview:**

The logistic function can take an input from negative to positive infinity and it has always has an output between 0 and 1. The logistic function is defined as:  
σ(t)=1/(1+e^−t)

We then, plot the logistic function to visualize it.

Now, for logistic function, if we view t as a linear function, as seen in the linear regression, with a variable x we could express t as:  
t=β0+β1x

Therefore, the logistic function can be re-written as:

F(x)=1/(1+e^−(β0+β1x))

This is the value of the logistic function of the linear regression expression. Here, we have a linear regression expression value that can vary from positive to negative infinity, but after the transformation due to the logistic expression we will have an output of F(x) that ranges from 0 to 1.

Now, we can assume that F(x) as the probability that the dependent variable is a success case. It can also be thought of as the Binomial Distribution.

**Dataset Analysis:**

The dataset that we are going to work with is a data set from a 1974 survey of women by Redbook magazine. In this, many married women were asked if they have had extramarital affairs. It is packed within the statsmodels module.This dataset has been viewed skeptically. But we will ignore all the issues with the dataset and focus on the logistic regression.

Now, we will approach this dataset as a classification problem. We use a statsmodel dataset so that we can have the option of working through additional example datasets included in SciKit Learn and their own tutorials. From the statsmodel website, we know that we have the following descriptions about the data:

How each woman rate their marriage, age, years married, children, religious, education, occupation, husband’s occupation, measure of time of affair.

**Data Visualization:**

Now as we are given details about women, we can determine whether they have participated in an affair, or not. The best way to get a clear idea is by data visualization.

* We visualize the data by putting it in a dataframe and then displaying a few rows using .head() or .tail().
* After this, we start our classfication by creating a new column called 'Had\_Affair'.
* We set this column equal to 0 if the affairs column is 0 , otherwise to 1.
* We do this by creating a function that sets the value and apply the function to the dataframe using the .apply() method.
* Then, we group the dataframe based on the Had\_Affair column, using .groupby() method and calling the mean aggregate function.
* Thus, for each Had\_Affair value, we take the ean of all the column values.
* We notice that the women who had affairs were slightly older,married longer, and slightly less religious and less educated. However, the mean values of both classes are very close for all variables, so we cannot be certain. So, we use plots.
* So, we create the factorplot with ‘age’ as X axis with hue as ‘Had\_Affair’. From this we notice a higher probability of an affair as age increases.
* We now create a factor plot with ‘number of years’ as the X axis to notice the trend. We notice that the probability of having an affair increases with the number of years married.
* We now create a factor plot with ‘number of children’ as the X axis to notice the trend. We notice that less children results in a lower probability of an affair.
* Finally, we create a factor plot with ‘education’ as the X axis to notice the trend. This results in an ambiguous result. So, we cannot claim anything from these observations.

**Data Preparation:**

If we look at the data, we'll notice that two columns, Occupation and Husband's Occupation, are unlike the others.. These columns are in a format known as Categorical Variables. Basically they are in set quantity/category, so that 1.0 and 2.0 are separate variables, not values along a spectrum that goes from 1-2 (e.g. There is no 1.5 for the occupation column).

But, fortunately Pandas has a built-in method of getting dummy variables and creating new columns from them.

* For this, we create new dataframes for the categorical variables. These dummies have six columns instead of just one column. This is because, there are six possible categories for occupation and husband’s occupation. So, for each value, pandas sets 0 if it’s not in that category, and 1 if it is.
* The reason we create these dummy variables, is that if we leava them as one column only, our regression will get confused. And it is going to assume that the variable is along a spectrum like 1.2 or 2.4 instead of discreet 1,2,3,4,5,6 like categorical variables.
* Then, we rename the columns to make them more readable.
* Now, we create the X and Y datasets for our logistic regression. we create a new DataFrame X without the occupation columns or the Had\_Affair coulmn. We add axis = 1 to signify that we are dropping columns not rows.
* Then, we concatenate the dummy variables. Then we concatenate this with the X dataframe.
* Finally, we set the Y dataset as the Had\_Affair column.

**Multicollinearity Consideration:**

Multicollinearity occurs due to the dummy variables.We will be dropping the occ1 and hocc1 columns to avoid multicollinearity.Our model begins to get distorted because one of the dummy variables can be linearly predicted from the others, as the dummy variables are highly correlated.

We solve this problem by dropping one of the dummy variables from each set, we do this at the cost of sacrificing a data set point. We also drop the affairs column. This is because it is a repetition of our Y target, instead of 0 and 1 it just has 0 or a number, so we'll need to drop it for our target to make sense.

To use Y with SciKit Learn, we need to set it as a 1-D array. This is called flattening the array. There is a built in method .ravel() in numpy to do this.

**Logistic Regression with SciKit Learn**

Creating a logical regression is very similar to creating a linear regression.We'll first  create the model, the fit the data into the model, and check our accuracy score. Then we'll split the data into testing and training sets.

* We can check our accuracy score by using the .score() method.
* Then, we compare this to the original Y data, using the null error rate. We can do this by simply taking the mean of the Y data, since it is in the format 1 or 0, we can use the mean to calulate the percentage of women who reported having affairs.
* We can see that  our model just simply guessed "no affair" we would have 1-mean ratio of accuracy. So, we are doing better than the null error rate, but not by so much.
* Now, we check the coefficients of our model to check what seemed to be the stronger predictors. We thus notice that a positive coefficient corresponds to increasing the likelihood of having an affair while a negative coefficient means it corresponds to a decreased likelihood of having an affair as the actual data value point increases.
* And, as we have seen above, an increased marriage rating corresponded to a decrease in the likelihood of having an affair. Increased religiousness also seems to correspond to a decrease in the likelihood of having an affair.

**Testing and Training Data Sets**

* To split the data, we use train\_test\_split() method. Then we create a new log\_model and then fit the model using .fit() method.
* Now, we predict the classes of the testing data set, by using the .predict() method.
* Finally, we compare the predicted classes to the actual test classes, to get the accuracy score.