**Assignment 5**

**Data Analytics II**

1. Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset

Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset

# What is logistic Regression

* Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
* Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
* Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

1. **Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset**

Our dataset contains some information about all of our users in the social network, including their User ID, Gender, Age, and Estimated Salary. The last column of the dataset is a vector of booleans describing whether or not each individual ended up clicking on the advertisement (0 = False, 1 = True). Let's import the relevant libraries, the dataset, and establish which variables are either dependent or independent. We'll continually print out any changes that we've made to the data at the bottom of our code cells.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('../input/Social\_Network\_Ads.csv')

dataset.head()

If we wanted to determine the effect of more independent variables on the outcome (such as Gender), we would have to implement a Dimensionality Reduction aspect to the model because we can only describe so many dimensions visually. However, right now we are only worried about how the users' Age and Estimated Salary affect their decision to click or not click on the advertisement. To do this, we will extract the relevant vectors from our dataset: the independent variables (X) and the dependent variable (y). The following code segment describes the selection of the entire third and fourth columns for X, as well as the entire fifth column for y. Again, we'll print out our data in order to help visualize the model.

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

print(X[:3, :])

print('-'\*15)

print(y[:3])

We now need to split our data into two sets: a training set for the machine to learn from, as well as a test set for the machine to execute on. This process is referred to as Cross Validation and we will be implementing SciKit Learn's appropriately named 'train\_test\_split' class to make it happen. Industry standard usually calls for a training set size of 70-80% so we'll split the two.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

print(X\_train[:3])

print('-'\*15)

print(y\_train[:3])

print('-'\*15)

print(X\_test[:3])

print('-'\*15)

print(y\_test[:3])

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.transform(X\_test)

print(X\_train[:3])

print('-'\*15)

print(X\_test[:3])

Now we are ready to build our Logistic Regression Model. We create an object of the LogisticRegression() class and refer to it as our 'classifier' for obvious reasons. The random state variable simply allows us to all get the same outcome but can be changed to alter the results slightly. We then fit the classifier to the training set with the aptly named .fit() method so that it can understand the correlations between X and y. Lastly, we will test the classifier's predictive power on the test set. The Logistic Regression's .predict() method will give us a vector of predictions for our dataset, X\_test.

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0, solver='lbfgs' )

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

print(X\_test[:10])

print('-'\*15)

print(y\_pred[:10])

print(y\_pred[:20])

print(y\_test[:20])

Now that we've preprocessed the data, fit our classifier to the training set, and predicted the dependent values for our test set, we can use a Confusion Matrix to evaluate exactly how accurate our Logistic Regression model is. This function will compare the calculated results in our y\_pred vector to the actual observed results in y\_test to determine how similar they are. The more values that match, the higher the accuracy of the classifier.

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

This Confusion Matrix tells us that there were 89 correct predictions and 11 incorrect ones, meaning the model overall accomplished an 89% accuracy rating. This is very good and there are many ways to improve the model by parameter tuning and sample size increasing, but those topics are outside the scope of this project. Our next step is to create visualizations to compare the training set and the test set.

*# Visualizing the Training set results*

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.6, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j **in** enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

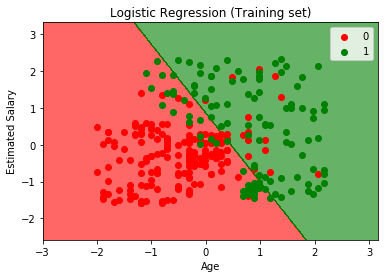
plt.title('Logistic Regression (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

****

*# Visualizing the Test set results*

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.6, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j **in** enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

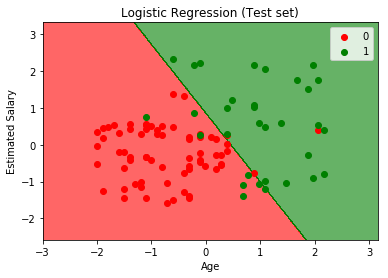
plt.title('Logistic Regression (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

****

1. **Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.**

## ****Confusion Matrix****

A confusion matrix is used to judge the performance of a classifier on the test dataset for which we already know the actual values. Confusion matrix is also termed as Error matrix. It consists of a count of correct and incorrect values broken down by each class. It not only tells us the error made by classifier but also tells us what type of error the classifier made. So, we can say that a confusion matrix is a performance measurement technique of a classifier model where output can be two classes or more. It is a table with four different groups of true and predicted values.

## ****Terminologies in Confusion Matrix****

The confusion matrix shows us how our classifier gets confused while predicting. In a confusion matrix we have four important terms which are:

1. **True Positive (TP)**
2. **True Negative (TN)**
3. **False Positive (FP)**
4. **False Negative (FN)**

We will explain these terms with the help of visualisation of the confusion matrix:

This is what a confusion matrix looks like. This is a case of a 2-class confusion matrix. On one side of the table, there are predicted values and on one side there are the actual values.

Let’s discuss the above terms in detail:

### True Positive (TP)

Both actual and predicted values are Positive.

### True Negative (TN)

Both actual and predicted values are Negative.

### False Positive (FP)

The actual value is negative but we predicted it as positive.

### False Negative (FN)

The actual value is positive but we predicted it as negative.

## ****Performance Metrics****

Confusion matrix not only used for finding the errors in prediction but is also useful to find some important performance metrics like Accuracy, Recall, Precision, F-measure. We will discuss these terms one by one.

### Accuracy

As the name suggests, the value of this metric suggests the accuracy of our classifier in predicting results.

It is defined as:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

A 99% accuracy can be good, average, poor or dreadful depending upon the problem.

### Precision

Precision is the measure of all actual positives out of all predicted positive values.

It is defined as:

Precision = TP / (TP + FP)

### Recall

Recall is the measure of positive values that are predicted correctly out of all actual positive values.

It is defined as:

Recall = TP / (TP + FN)

High Value of Recall specifies that the class is correctly known (because of a small number of False Negative).

### F-measure

It is hard to compare classification models which have low precision and high recall or vice versa. So, for comparing the two classifier models we use F-measure. F-score helps to find the metrics of Recall and Precision in the same interval. Harmonic Mean is used instead of Arithmetic Mean.

F-measure is defined as:

F-measure = 2 \* Recall \* Precision / (Recall + Precision)

The F-Measure is always closer to the Precision or Recall, whichever has a smaller value.

**Calculation of 2-class confusion matrix**

Let us derive a confusion matrix and interpret the result using simple mathematics.

Let us consider the actual and predicted values of y as given below:

|  |  |  |
| --- | --- | --- |
| **Actual y** | **Y predicted** | **Predicted y with threshold 0.5** |
| 1 | 0.7 | 1 |
| 0 | 0.1 | 0 |
| 0 | 0.6 | 1 |
| 1 | 0.4 | 0 |
| 0 | 0.2 | 0 |

Now, if we make a confusion matrix from this, it would look like:

|  |  |  |
| --- | --- | --- |
| N=5 | **Predicted 1** | **Predicted 0** |
| **Actual: 1** | 1 (TP) | 1 (FN) |
| **Actual: 0** | 1 (FP) | 2 (TN) |

This is our derived confusion matrix. Now we can also see all the four terms used in the above confusion matrix. Now we will find all the above-defined performance metrics from this confusion matrix.

**Accuracy**

Accuracy **= (**TP + TN) / (TP + TN + FP + FN)

So, Accuracy = (1+2) / (1+2+1+1)

= 3/5 which is 60%.

So, the accuracy from the above confusion matrix is 60%.

**Precision**

Precision = TP / (TP + FP)

= 1 / (1+1)

=1 / 2 which is 50%.

So, the precision is 50%.

**Recall**

Recall = TP / (TP + FN)

= 1 / (1+1)

= ½ which is 50%

So, the Recall is 50%.

**F-measure**

F-measure = 2 \* Recall \* Precision / (Recall + Precision)

= 2\*0.5\*0.5 / (0.5+0.5)

= 0.5

So, the F-measure is 50%.

**Confusion Matrix in Python**

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt # Importing the required libraries

import seaborn as sns

%matplotlib inline

os.chdir("C:\\Users\\ABC\\Desktop\\bank")

df=pd.read\_csv("bank.csv", delimiter=";",header='infer')

df.head()

df.columns # Columns in the dataset

df.shape # There are 4521 rows and 17 columns in data

df.info () # Checking info of data

df.dtypes # Checking the data types of variables in data

df.describe() # Summary statistics of numerical columns in data

df.isnull().sum() # Checking the missing value in data. We can see that there is no missing value in data.

df.corr() # Correlation matrix

sns.heatmap(df.corr()) # Visualization of Correlation matrix Using heatmap