ARTIFICIAL NEURAL NETWORKS

FINAL PROJECT REPORT

About Dataset:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. All patients here are females at least 21 years old of Pima Indian heritage.

Predict whether the patients in the dataset have diabetes or not?

The dataset contains total of 768 records and each record contains 9 features and all the columns are listed below:

Features:

Pregnancies - Number of times pregnant.

Glucose - Plasma glucose concentration 2 hours in an oral glucose tolerance test.

Blood Pressure - Diastolic blood pressure (mm Hg).

Skin Thickness - Triceps skin fold thickness (mm).

Insulin - 2-Hour serum insulin (mu U/ml).

BMI - Body mass index (weight in kg/ (height in m) ^2).

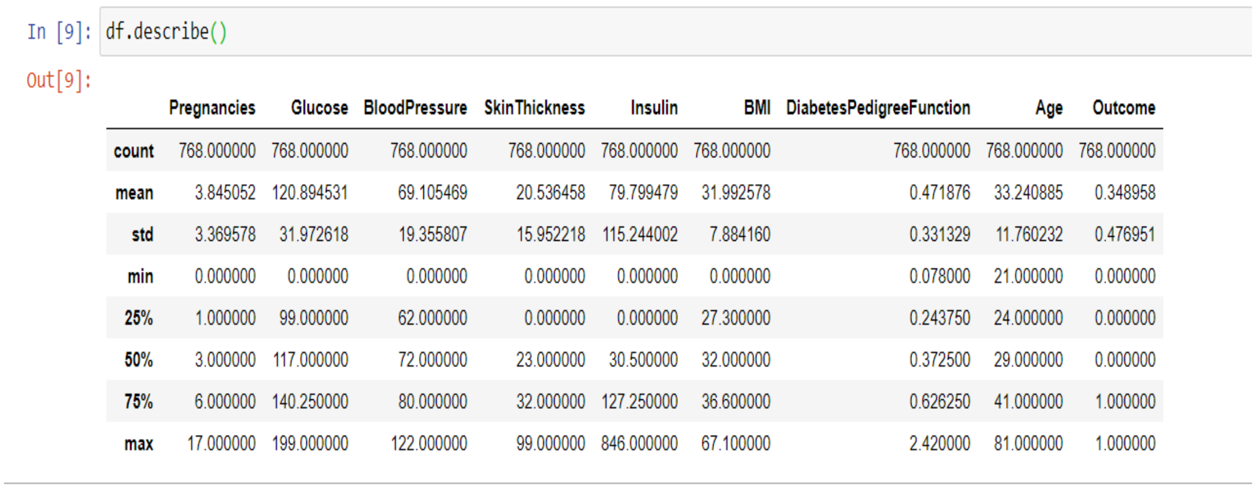
Diabetes Pedigree Function - Diabetes pedigree function.

Age - Age (years).

Outcome - Class variable (0 or 1) 268 of 768 are 1, the others are 0.

Describing Dataset:

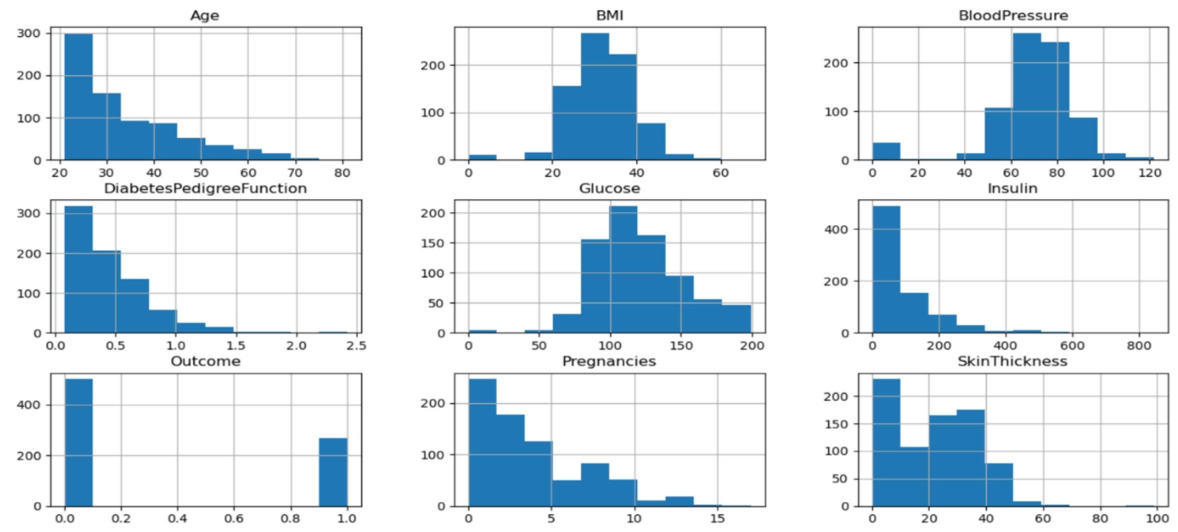
Before going any further, we should first check how the data has been distributed and get a little more information on the data. This can be done using describe function in pandas. From the image below we can say that the average age of the patient is 33 years old and minimum age is 21 years and Blood Pressure is also an interesting feature where average blood pressure stood at 69.10 and where 75 percent of the people are in ideal blood pressure range of 60mmHg - 80 mmHg. This gave more insights on the data.



```
In [9]: df.describe()

Out[9]:
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin         BMI  DiabetesPedigreeFunction         Age     Outcome
count   768.000000  768.000000     768.000000     768.000000  768.000000  768.000000                768.000000  768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479   31.992578                  0.471876   33.240885    0.348958
std       3.369578   31.972618      19.355807      15.952218  115.244002    7.884160                  0.331329   11.760232    0.476951
min       0.000000    0.000000       0.000000       0.000000    0.000000    0.000000                  0.078000   21.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000   27.300000                  0.243750   24.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000   32.000000                  0.372500   29.000000    0.000000
75%       6.000000  140.250000      80.000000      32.000000  127.250000   36.600000                  0.626250   41.000000    1.000000
max      17.000000  199.000000     122.000000      99.000000  846.000000   67.100000                  2.420000   81.000000    1.000000
```

Also check the data has been distributed across each feature (here we left out the outcome feature)

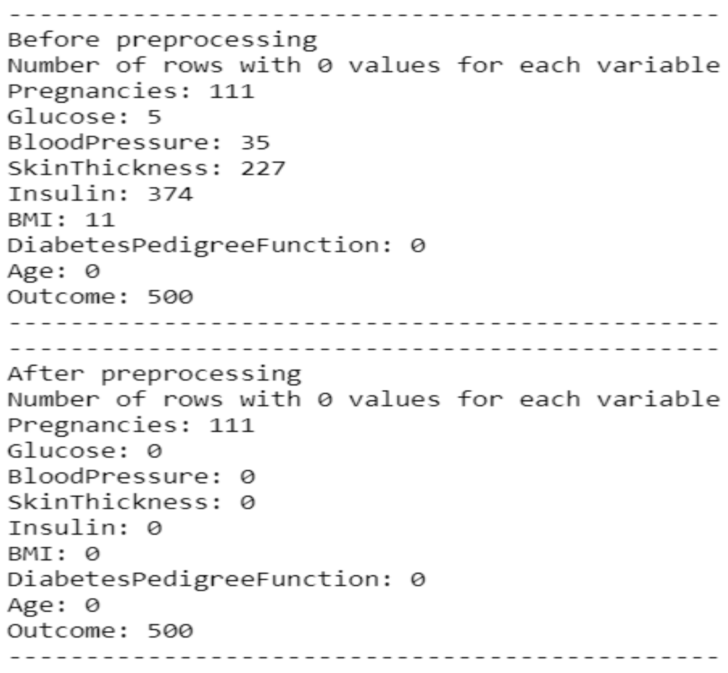


Data Preprocessing:

Why is it necessary to preprocess the data?

Because there is always chance of data containing null values or any other missing values or you can remove unwanted features and data that cannot be used, and it will lead bad results. So, it is always good practice to preprocess and clean the data before feeding it to the Machine Learning Algorithm.

For this dataset I have preprocessed the data and I have replaced the values that 0 with average values of that feature.



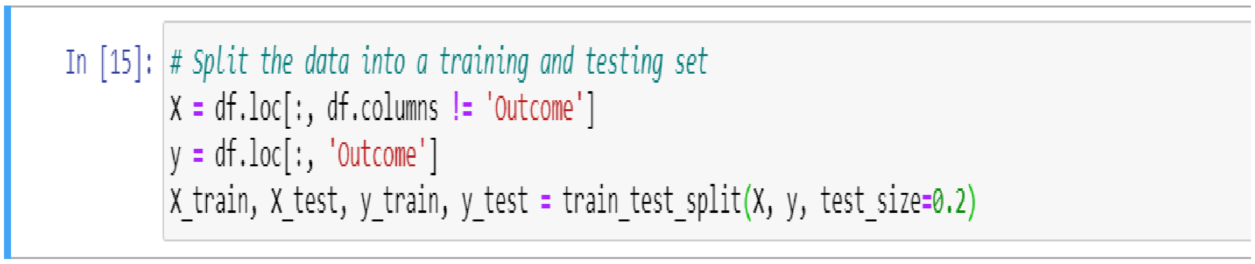
```
----------------------------------------------
Before preprocessing
Number of rows with 0 values for each variable
Pregnancies: 111
Glucose: 5
BloodPressure: 35
SkinThickness: 227
Insulin: 374
BMI: 11
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 500
----------------------------------------------
----------------------------------------------
After preprocessing
Number of rows with 0 values for each variable
Pregnancies: 111
Glucose: 0
BloodPressure: 0
SkinThickness: 0
Insulin: 0
BMI: 0
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 500
----------------------------------------------
```

Splitting the Data:

Why should we split the data?

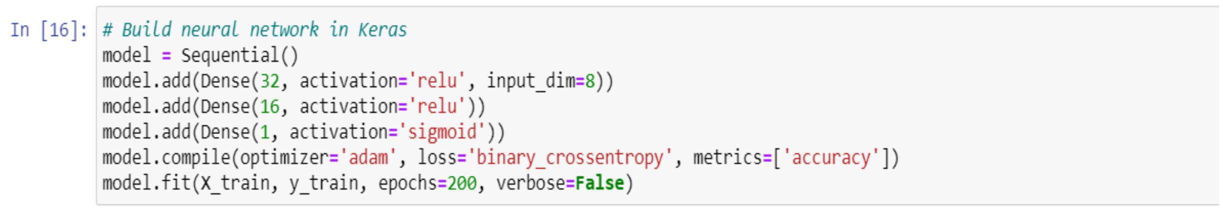
If we don't split the data the problem is that we are training the model on the dataset and again we are testing the model with same dataset so in this case we will get good results but if we test the same model on the new data it may not perform best as it has performed on the current dataset. So, it is always good practice to split the data into training and test set.

Here for this dataset I am splitting it into training and test as show in the image below:



```
In [15]: # Split the data into a training and testing set
         X = df.loc[:, df.columns != 'Outcome']
         y = df.loc[:, 'Outcome']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

After splitting the data, I am feeding the data to a neural network as defined below:



```
In [16]: # Build neural network in Keras
         model = Sequential()
         model.add(Dense(32, activation='relu', input_dim=8))
         model.add(Dense(16, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))
         model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
         model.fit(X_train, y_train, epochs=200, verbose=False)
```

As you can see from the code I have used a simple neural network to train the model it has 2 hidden layers one with 32 nodes and activation function is relu and other hidden layer with 16 nodes activation function as relu and the final node is the output layer where I am using sigmoid to get the binary output. I am also compiling the model using adam as my optimizer with the default learning rate 0.001.

How to Check if our model has performed better?

Because it is a classification problem, we can use accuracy and f1 score to determine how well our model has done.

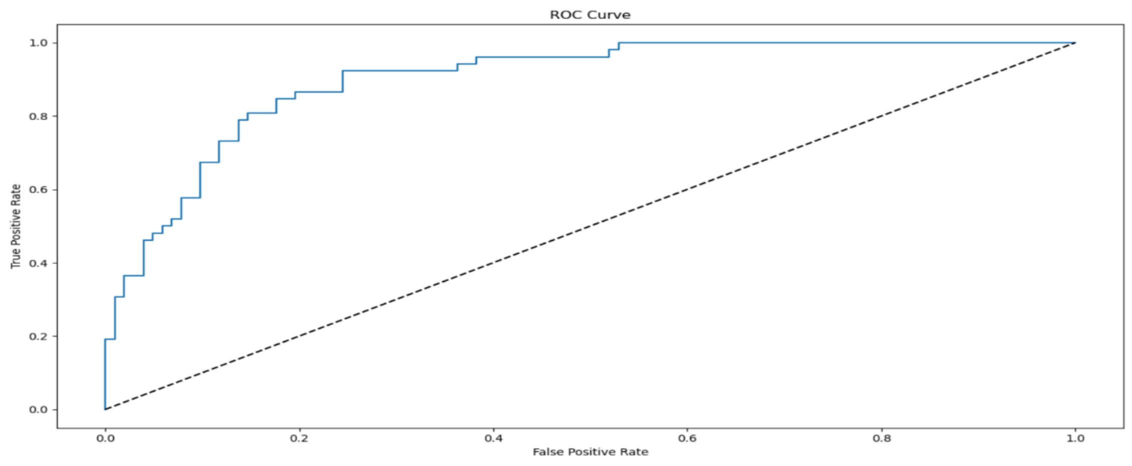
Accuracy Results:

Training Accuracy: 89.90%
Testing Accuracy: 81.17%

F1Score:

```
0.6947368421052632
```

ROC Curve:



Conclusion:

From the results of accuracy and f1score I can say that my model has performed better in giving results 70 percent of the times but we can improve the model performance and results if we train the by feeding it more data.