

# Capstone Project for Health Care

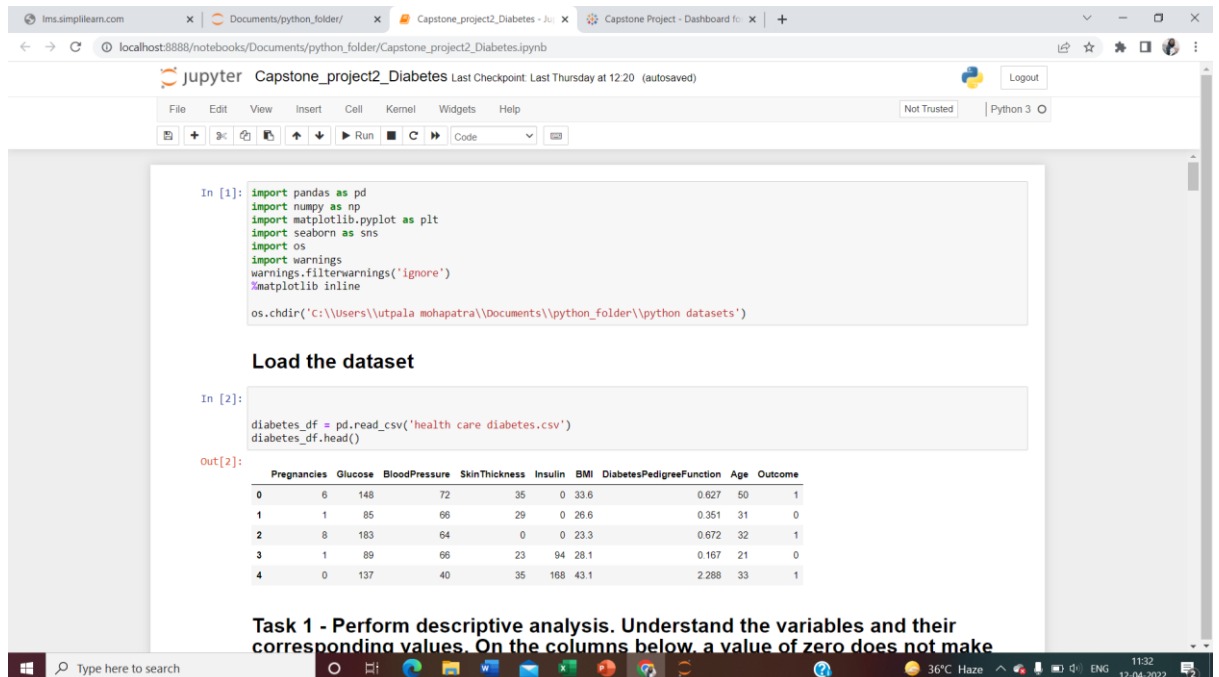
**Description-** NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases) research creates knowledge about and treatments for the most chronic, costly, and consequential diseases.

- The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Build a model to accurately predict whether the patients in the dataset have diabetes or not.

## Data Exploration-

**Task 1 -** Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

### Load Dataset



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

os.chdir('C:\\Users\\utpala mohapatra\\Documents\\python_folder\\python_datasets\\')
```

**Load the dataset**

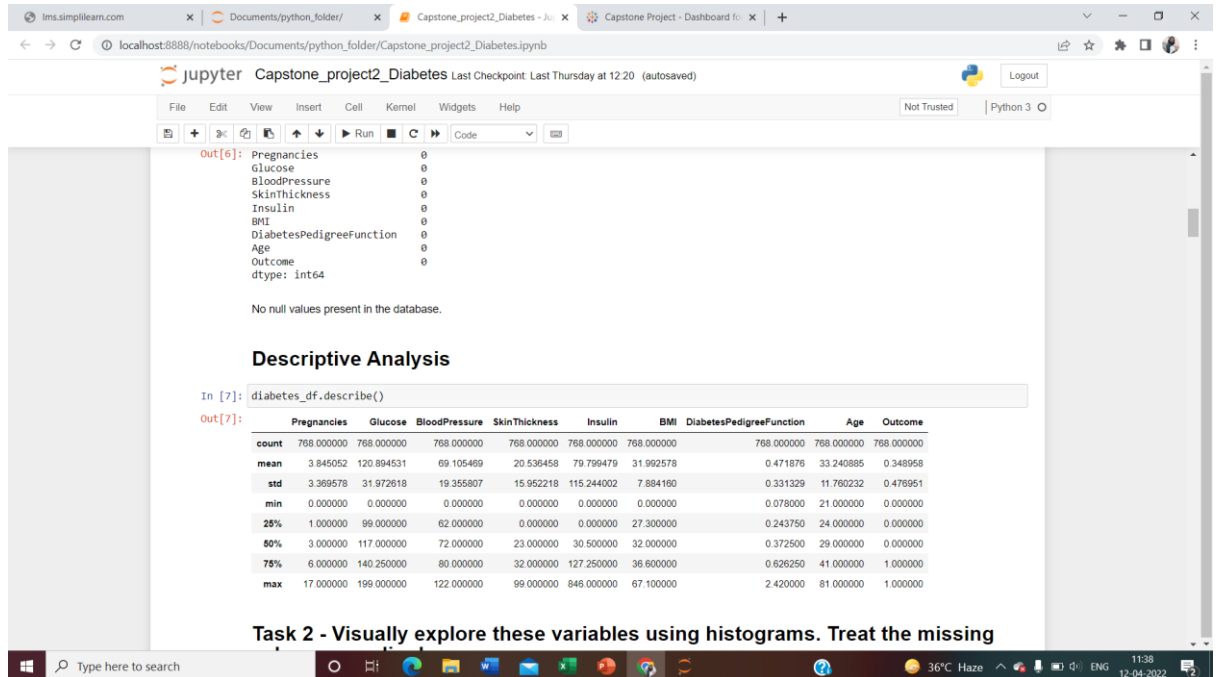
```
In [2]: diabetes_df = pd.read_csv('health care diabetes.csv')
diabetes_df.head()
```

**Out[2]:**

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

**Task 1 - Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make**

## Descriptive Analysis of the data.



The screenshot shows a Jupyter Notebook interface with the following content:

```
Out[6]: Pregnancies      0
         Glucose          0
         BloodPressure    0
         SkinThickness    0
         Insulin          0
         BMI              0
         DiabetesPedigreeFunction 0
         Age              0
         Outcome          0
         dtype: int64

No null values present in the database.
```

### Descriptive Analysis

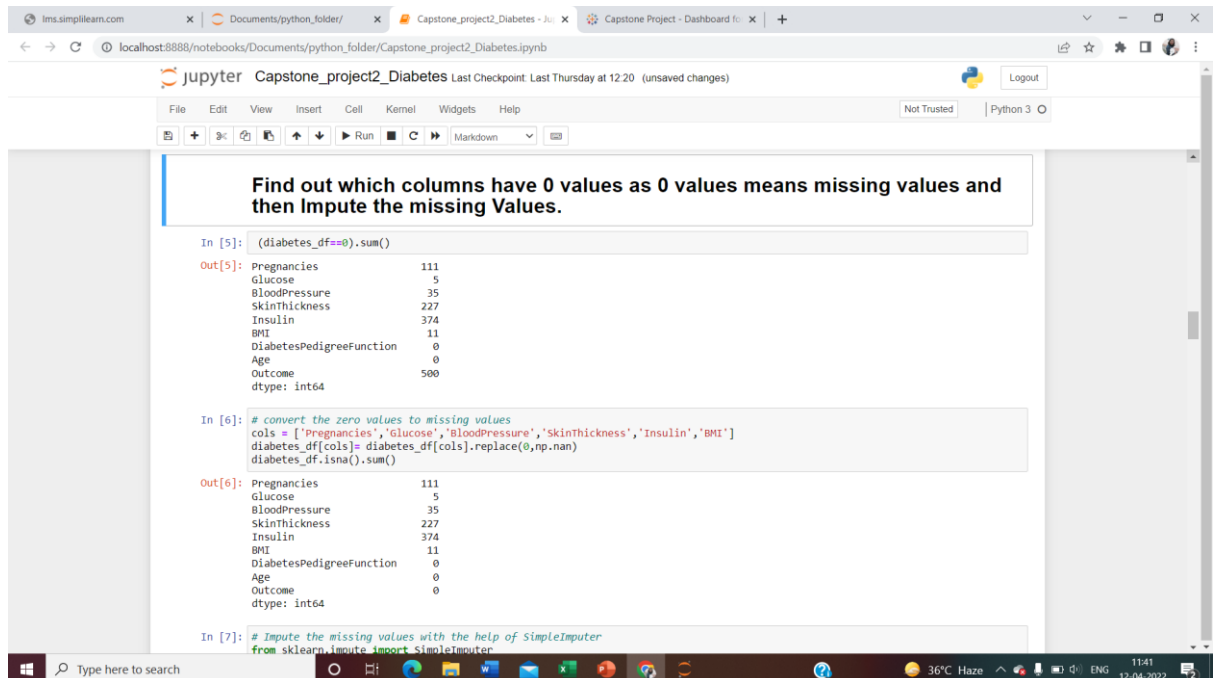
```
In [7]: diabetes_df.describe()
```

```
Out[7]:
```

	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	0.471878	33.240895	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

**Task 2 - Visually explore these variables using histograms. Treat the missing**

## Check for missing values.



The screenshot shows a Jupyter Notebook interface with the following content:

**Find out which columns have 0 values as 0 values means missing values and then Impute the missing Values.**

```
In [5]: (diabetes_df==0).sum()
```

```
Out[5]: Pregnancies      111
         Glucose         5
         BloodPressure    35
         SkinThickness    227
         Insulin          374
         BMI              11
         DiabetesPedigreeFunction 0
         Age              0
         Outcome          500
         dtype: int64
```

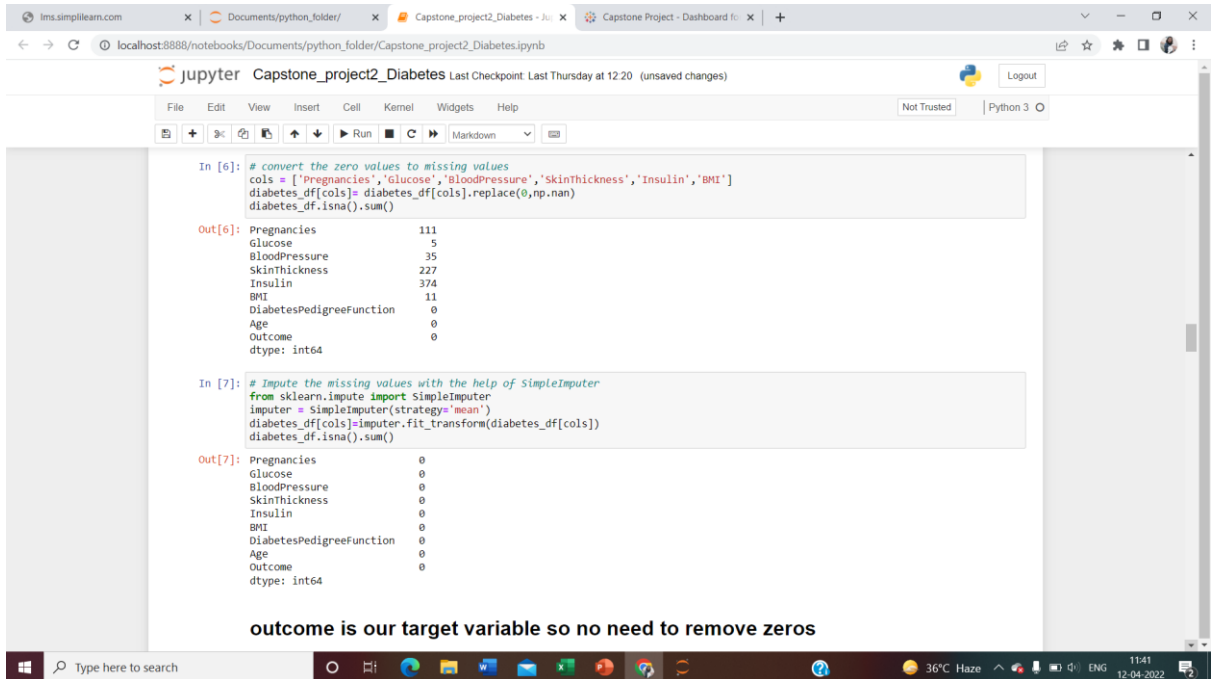
```
In [6]: # convert the zero values to missing values
        cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
        diabetes_df[cols] = diabetes_df[cols].replace(0, np.nan)
        diabetes_df.isna().sum()
```

```
Out[6]: Pregnancies      111
         Glucose         5
         BloodPressure    35
         SkinThickness    227
         Insulin          374
         BMI              11
         DiabetesPedigreeFunction 0
         Age              0
         Outcome          0
         dtype: int64
```

```
In [7]: # Impute the missing values with the help of SimpleImputer
        from sklearn.impute import SimpleImputer
```

**Task 2 -** Visually explore these variables using histograms. Treat the missing values accordingly.

## Impute missing values



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [6]: # convert the zero values to missing values
cols = ['pregnancies', 'glucose', 'bloodPressure', 'skinThickness', 'Insulin', 'BMI']
diabetes_df[cols] = diabetes_df[cols].replace(0, np.nan)
diabetes_df.isna().sum()

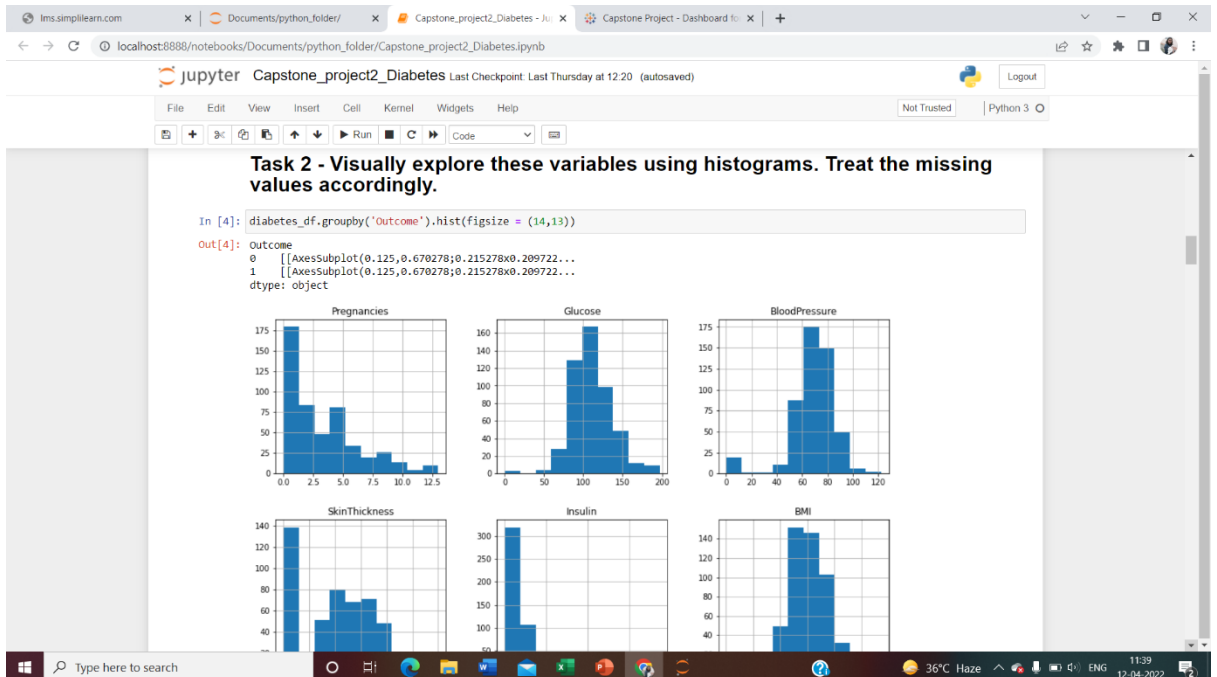
Out[6]: pregnancies      111
        glucose          5
        bloodPressure    35
        skinThickness    227
        Insulin          374
        BMI              11
        DiabetesPedigreeFunction  0
        Age              0
        Outcome          0
        dtype: int64

In [7]: # Impute the missing values with the help of SimpleImputer
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
diabetes_df[cols] = imputer.fit_transform(diabetes_df[cols])
diabetes_df.isna().sum()

Out[7]: pregnancies      0
        glucose          0
        bloodPressure    0
        skinThickness    0
        Insulin          0
        BMI              0
        DiabetesPedigreeFunction  0
        Age              0
        Outcome          0
        dtype: int64
```

outcome is our target variable so no need to remove zeros

## Visual analysis by Histograms



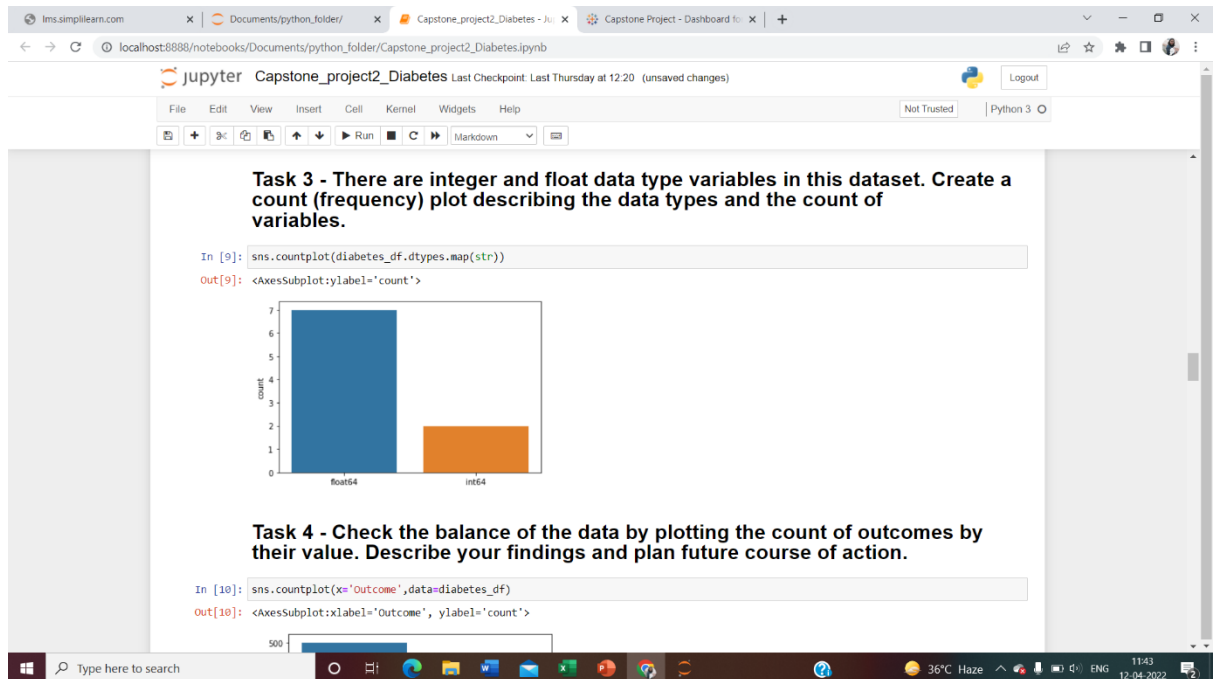
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [4]: diabetes_df.groupby('Outcome').hist(figsize = (14,13))

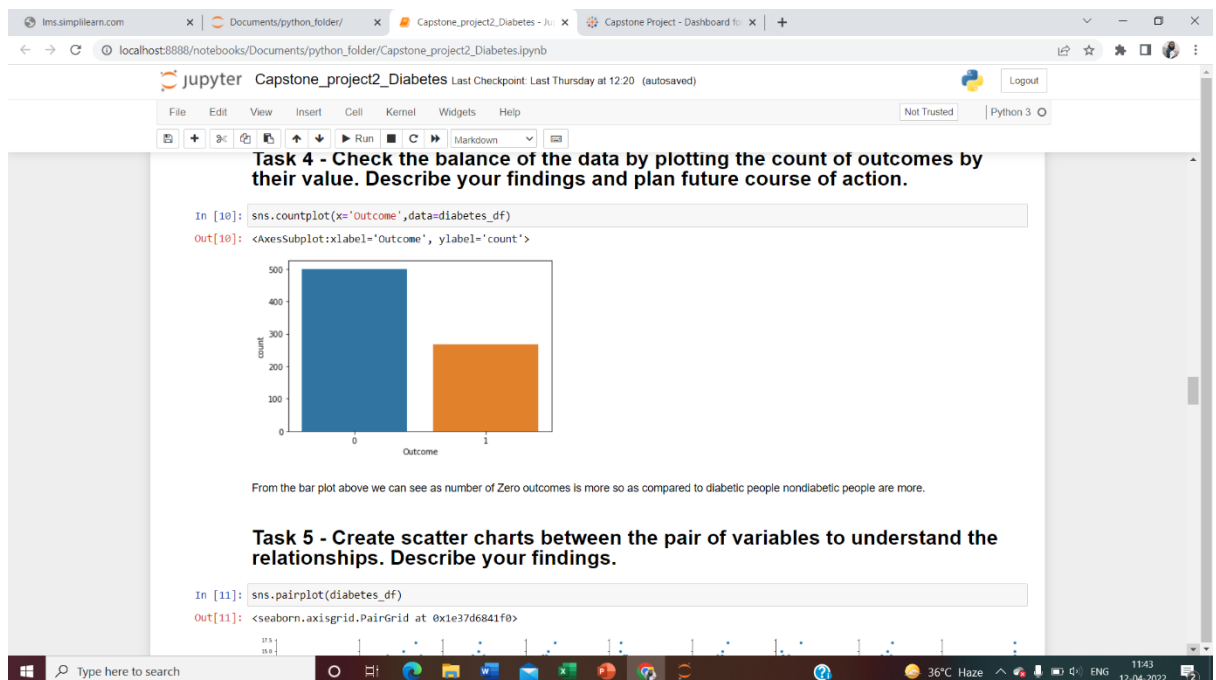
Out[4]: Outcome
0      [AxesSubplot(0.125,0.670278;0.215278x0.209722...
1      [AxesSubplot(0.125,0.670278;0.215278x0.209722...
        dtype: object
```

The output displays six histograms arranged in a 2x3 grid, showing the distribution of variables for two outcomes (0 and 1). The variables are: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, and BMI. The histograms show the frequency of values for each variable, with Outcome 0 generally having higher frequencies than Outcome 1.

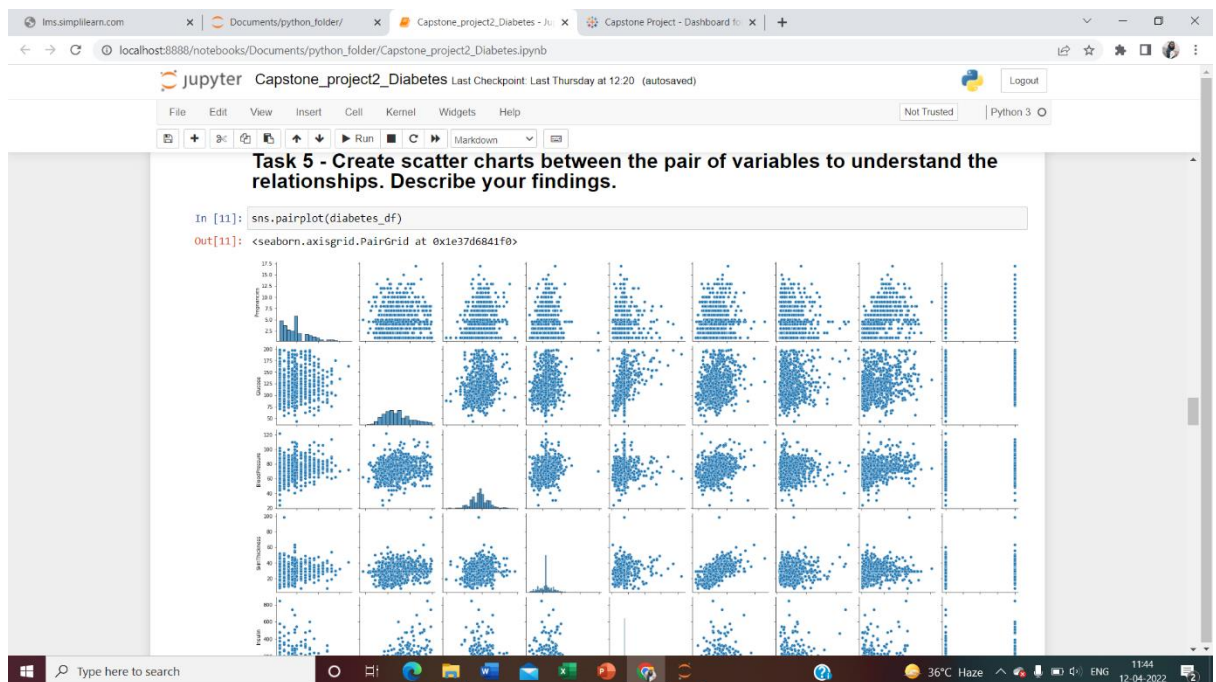
**Task 3** – There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.



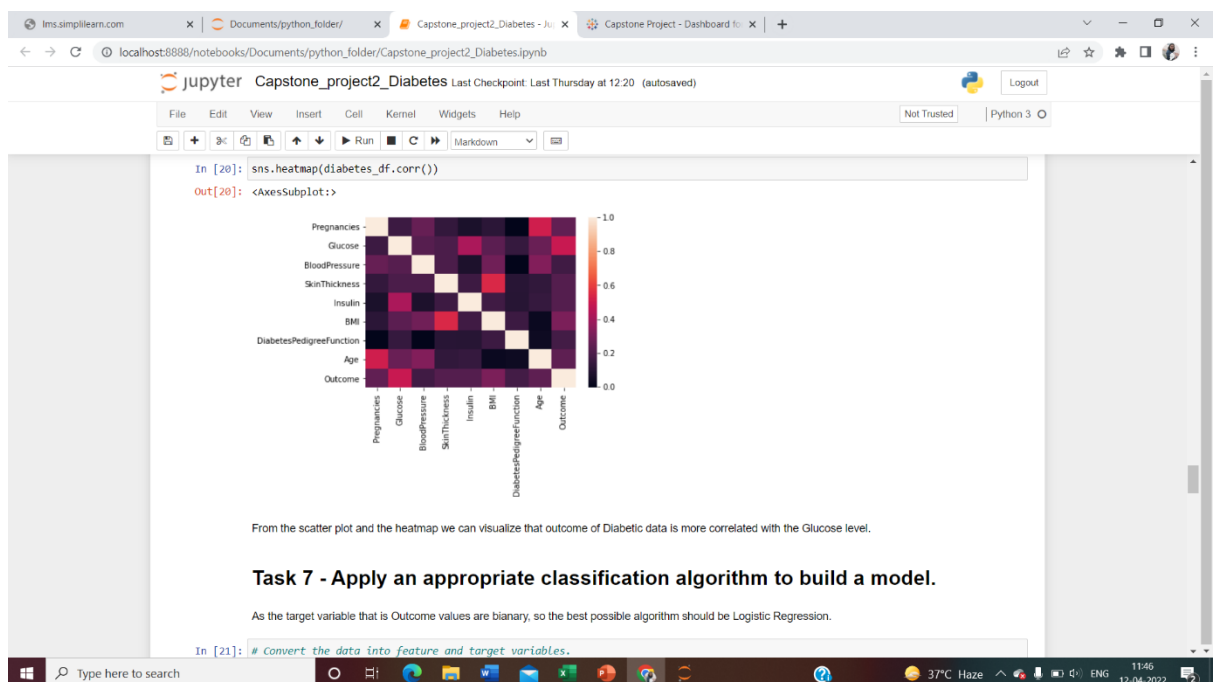
**Task 4** – Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.



## Task 5 – Create scatter charts between the pair of variables to understand the relationships. Describe your findings.



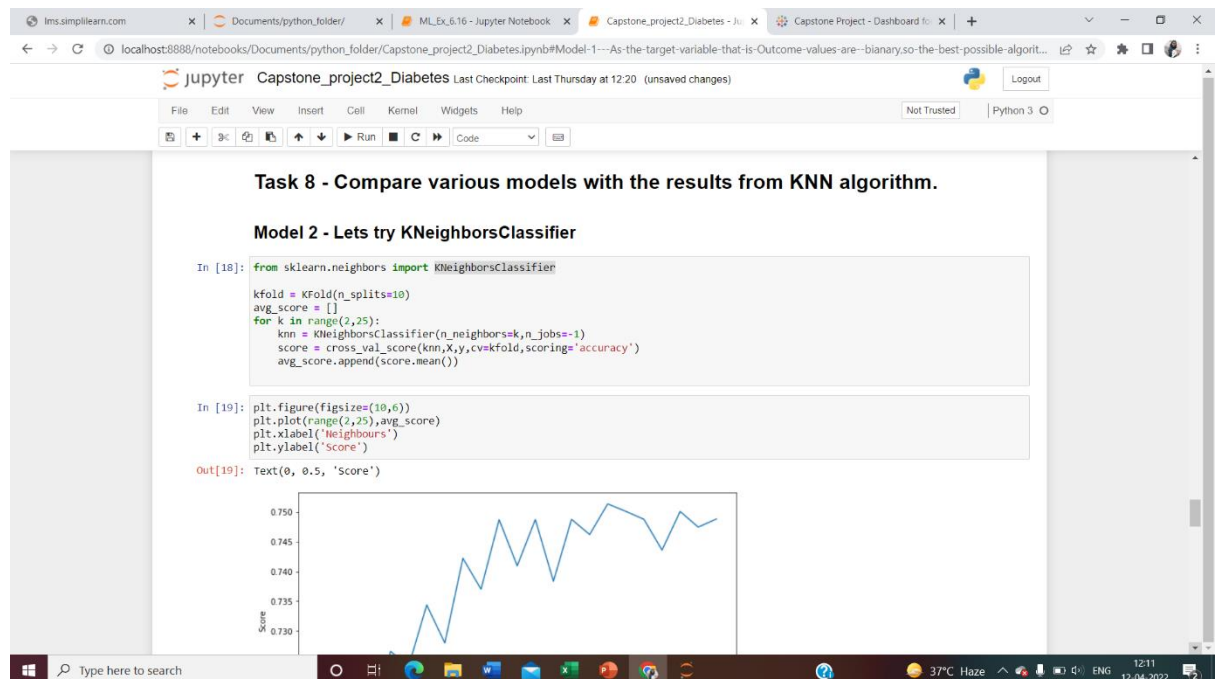
## Task 6 – Perform correlation analysis. Visually explore it using a heatmap.

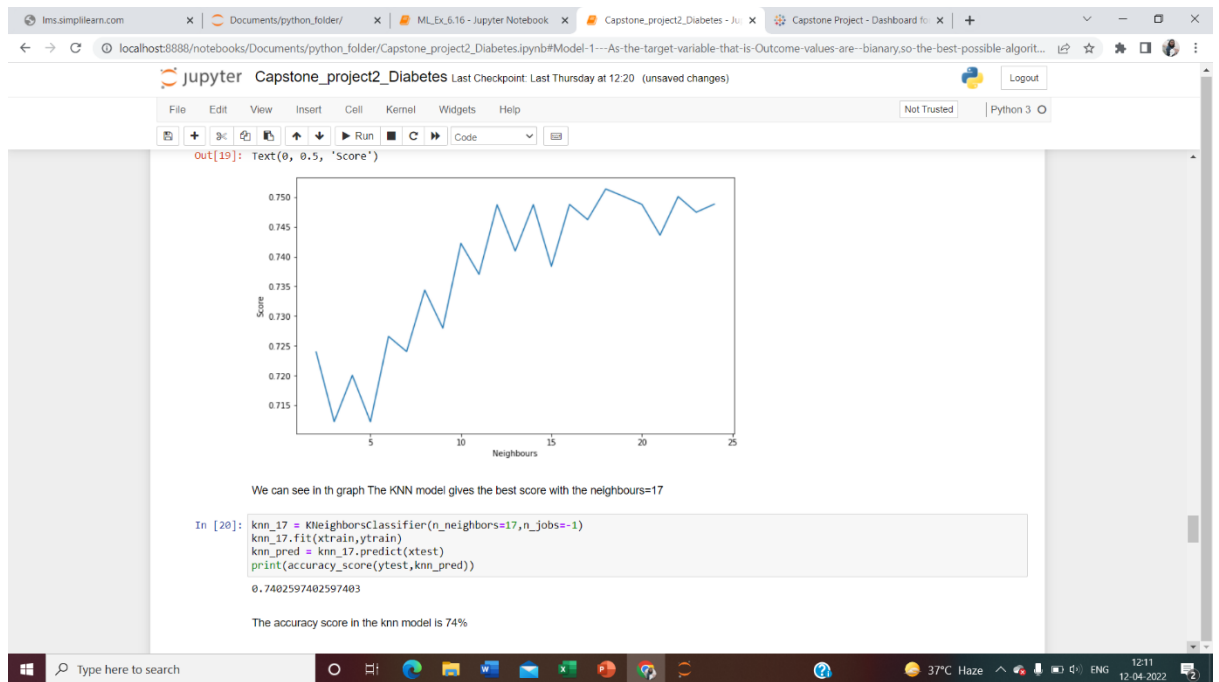


ANS - As the target variable is binary so Logistic regression should be the suitable model for this data to predict best results.

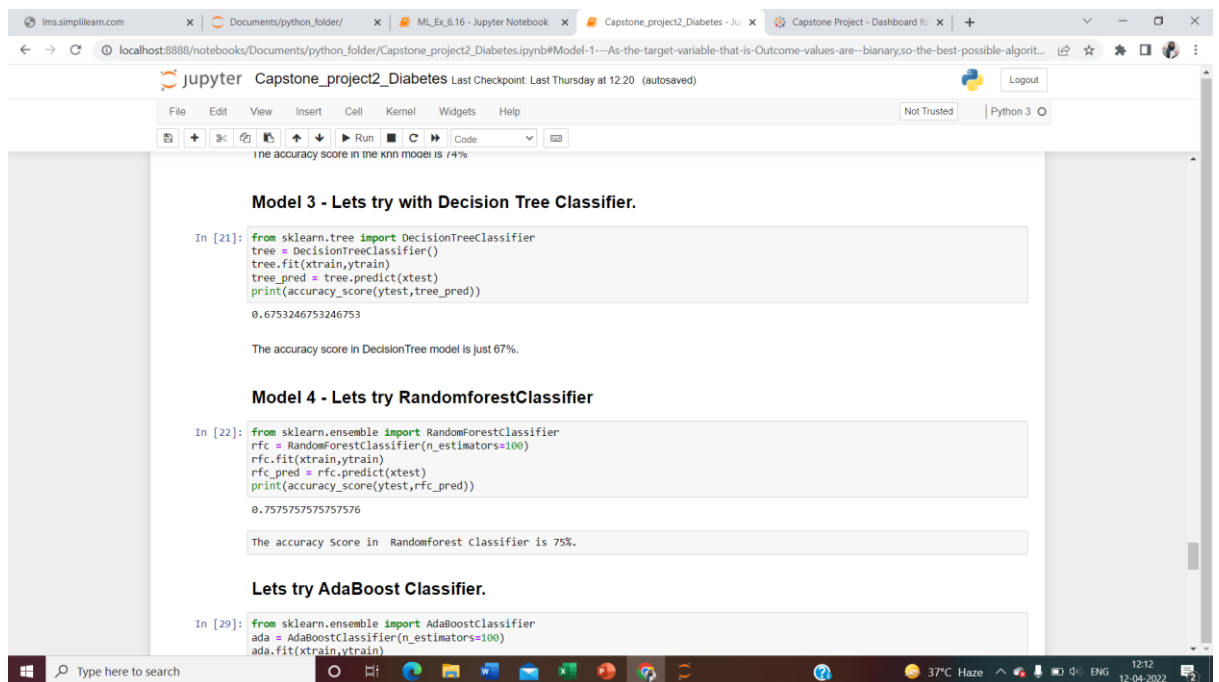
**Task 7** – Compare various models with the results from KNN algorithm.

## KNN model



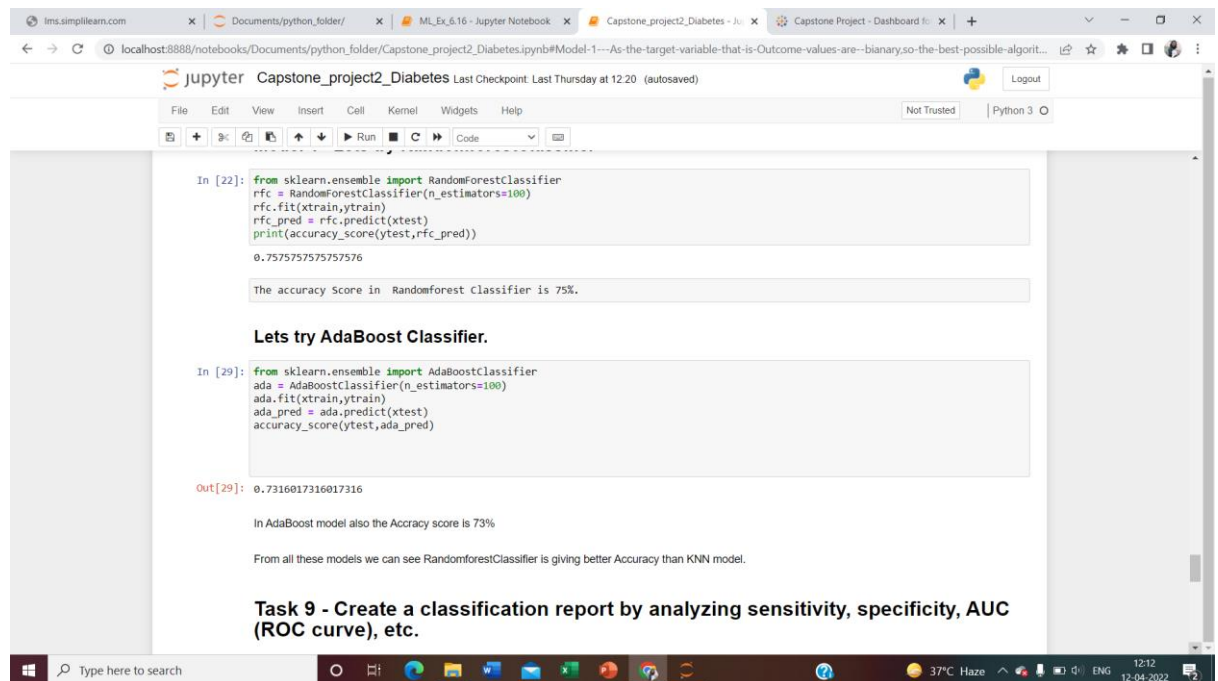


## DecisionTree and RandomForest Classifier





# Adaboost Classifier



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell imports RandomForestClassifier, fits it to training data, and prints the accuracy score on test data, which is 0.7575757575757576. The second cell imports AdaBoostClassifier, fits it to training data, and prints the accuracy score on test data, which is 0.7316017316017316. Below the code, there is a text box stating 'The accuracy Score in RandomForest Classifier is 75%.' and another text box stating 'In AdaBoost model also the Accuracy score is 73%'. A final text box concludes that 'RandomforestClassifier is giving better Accuracy than KNN model.'

```
In [22]: from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(xtrain,ytrain)
rfc_pred = rfc.predict(xtest)
print(accuracy_score(ytest,rfc_pred))
0.7575757575757576

The accuracy Score in RandomForest Classifier is 75%.

Lets try AdaBoost Classifier.

In [29]: from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(n_estimators=100)
ada.fit(xtrain,ytrain)
ada_pred = ada.predict(xtest)
accuracy_score(ytest,ada_pred)

Out[29]: 0.7316017316017316

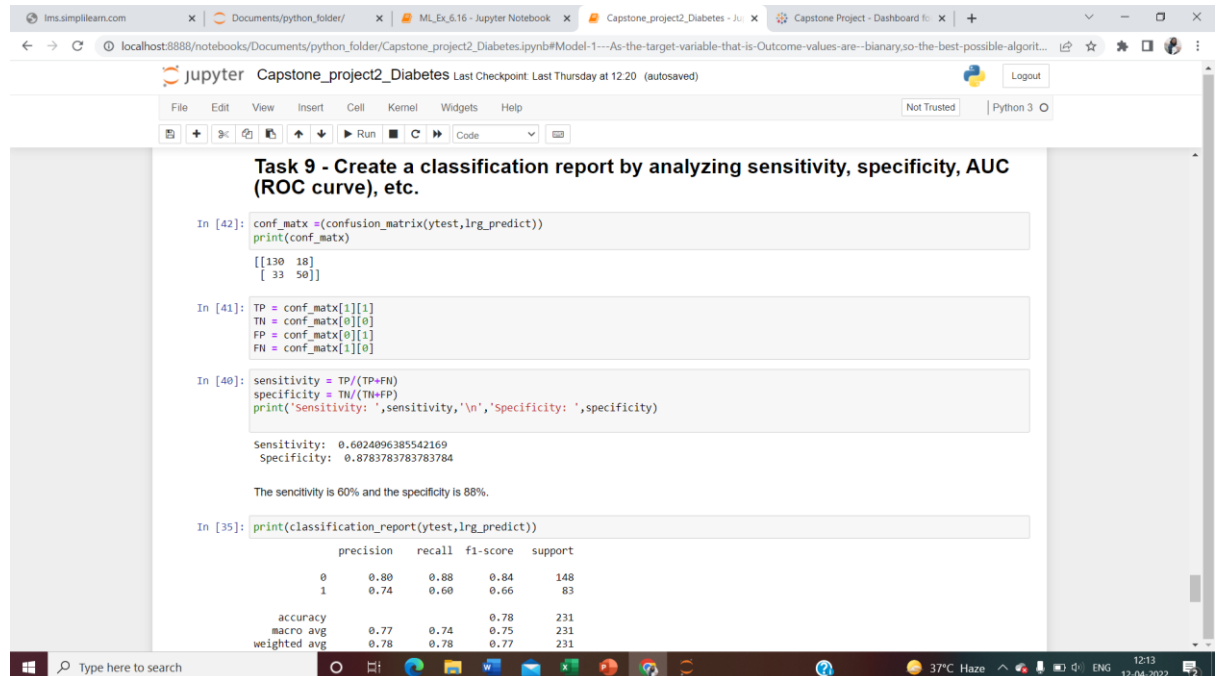
In AdaBoost model also the Accuracy score is 73%

From all these models we can see RandomforestClassifier is giving better Accuracy than KNN model.
```

**Task 9 - Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc.**

**Task 8** – Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc.

## Sensitivity, Specificity and Classification Report



The screenshot shows a Jupyter Notebook interface with three code cells. The first cell prints the confusion matrix for the test data. The second cell calculates and prints the sensitivity and specificity. The third cell prints the classification report for the test data.

```
In [42]: conf_matx = confusion_matrix(ytest,lrg_predict)
print(conf_matx)
[[130  18]
 [ 33  50]]

In [41]: TP = conf_matx[1][1]
TN = conf_matx[0][0]
FP = conf_matx[0][1]
FN = conf_matx[1][0]

In [40]: sensitivity = TP/(TP+FN)
specificity = TN/(TN+FP)
print('Sensitivity: ',sensitivity,'\n','Specificity: ',specificity)

Sensitivity: 0.6024096385542169
Specificity: 0.8783783783783784

The sencitivity is 60% and the specificity is 88%.

In [35]: print(classification_report(ytest,lrg_predict))

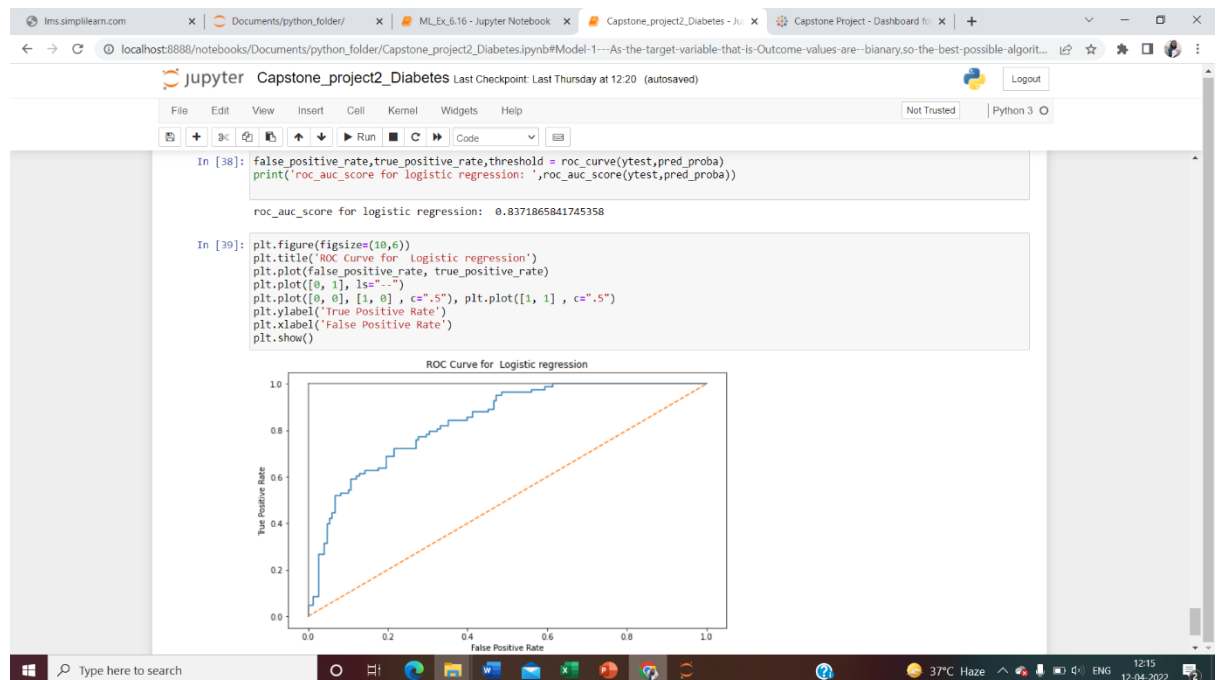
              precision    recall  f1-score   support

     0       0.80       0.88       0.84        148
     1       0.74       0.60       0.66         83

 accuracy          0.77       0.74       0.75        231
 macro avg         0.78       0.78       0.77        231
 weighted avg         0.77       0.78       0.77        231
```



# ROC curve



**Task 5** - Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard should entail the following:

- Pie chart to describe the diabetic or non-diabetic population
- Scatter charts between relevant variables to analyze the relationships
- Histogram or frequency charts to analyze the distribution of the data
- Heatmap of correlation analysis among the relevant variables

