```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving diabetes.csv to diabetes (1).csv

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(color_codes = True )
#ignore warning messages
import warnings
warnings.filterwarnings('ignore')
```

```
pima = pd.read_csv('diabetes.csv')
pima
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

```
pima.describe().T
```

	count	mean	std	min	25%	50%	
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	14
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	8
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	3

```
pima.isnull().values.any()
```

```
False
```

```
pima['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

There are no NULL values, but 0 values do exist, following steps have been taken to replace them:

```
pima[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = pima[['Glucose
print(pima.isnull().sum())
```

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
pima['Glucose'].fillna(pima['Glucose'].mean(), inplace = True )
pima['BloodPressure'].fillna(pima['BloodPressure'].mean(), inplace = True )
pima['SkinThickness'].fillna(pima['SkinThickness'].mean(), inplace = True )
pima['Insulin'].fillna(pima['Insulin'].mean(), inplace = True )
pima['BMI'].fillna(pima['BMI'].mean(), inplace = True )
```

```
print(pima.isnull().sum())
```

```
Pregnancies               0
Glucose                   0
BloodPressure             0
SkinThickness             0
Insulin                   0
BMI                       0
DiabetesPedigreeFunction  0
Age                       0
```

```
Outcome                     0
dtype: int64
```

Results of Data Cleaning:

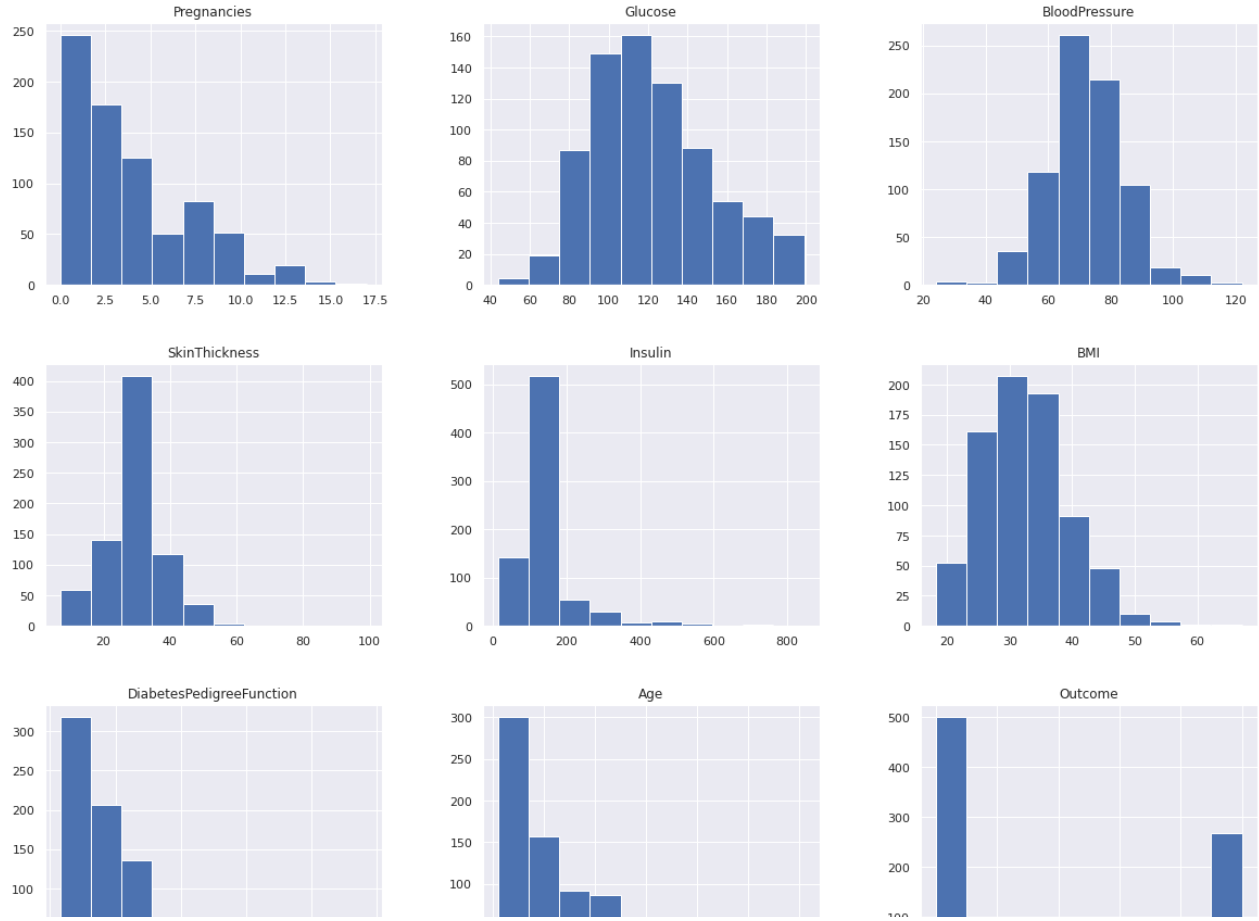
```
pima.describe().T
```

	count	mean	std	min	25%	50%
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.000000
Glucose	768.0	121.686763	30.435949	44.000	99.75000	117.000000
BloodPressure	768.0	72.405184	12.096346	24.000	64.00000	72.202592
SkinThickness	768.0	29.153420	8.790942	7.000	25.00000	29.153420
Insulin	768.0	155.548223	85.021108	14.000	121.50000	155.548223
BMI	768.0	32.457464	6.875151	18.200	27.50000	32.400000
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.372500
Age	768.0	33.240885	11.760232	21.000	24.00000	29.000000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.000000

Distribution of Data

```
pima.hist(figsize=(20,16), grid=True )
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6718d32510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718c96290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718c4d890>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6718bfee90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718bbe4d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718bf3ad0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6718bb6190>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718b6b6d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6718b6b710>]],
      dtype=object)
```




```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
```

Splitting Data into Training and Testing sets

```
X = pima.drop('Outcome', axis = 1)
y = pima['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_
= 17)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.size)
print(y_test.size)
```

```
(537, 8)
(231, 8)
537
231
```

Building a model using Naive Bayes Classifier

```
nbModel = GaussianNB()
```

```
nbModel.fit(X_train, y_train)
```

```
GaussianNB()
```

```
nb_y_pred = nbModel.predict(X_test)
```

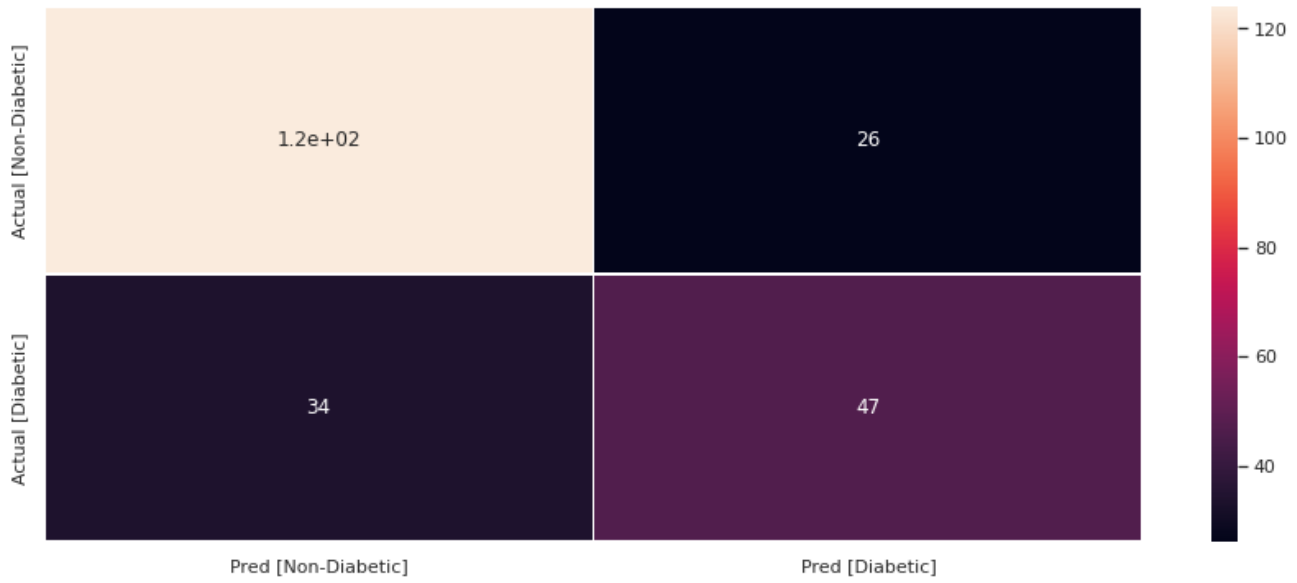
Evaluating the model using Confusion Matrix

```
nbConfusion = metrics.confusion_matrix(y_test, nb_y_pred)
nbConfusion
```

```
array([[124,  26],
       [ 34,  47]])
```

```
ylabel = ["Actual [Non-Diabetic]","Actual [Diabetic]"]
xlabel = ["Pred [Non-Diabetic]","Pred [Diabetic]"]
plt.figure(figsize=(15,6))
sns.heatmap(nbConfusion, annot=True , xticklabels = xlabel, yticklabels = ylabel,
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f671839e490>
```




```
print('Accuracy of Naive Bayes Classifier is: ', nbModel.score(X_test,y_test) * 10
```

```
Accuracy of Naive Bayes Classifier is:  74.02597402597402 %
```

```
print(classification_report(y_test, nb_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81       150
           1       0.64      0.58      0.61        81

    accuracy                           0.74       231
   macro avg       0.71      0.70      0.71       231
weighted avg       0.74      0.74      0.74       231
```

```
TP = nbConfusion[1,1]
TN = nbConfusion[0,0]
FP = nbConfusion[0,1]
FN = nbConfusion[1,0]
```

```
Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
Specificity = TN / (TN + FP)
print("Precision: ", Precision)
print("Recall (Sensitivity): ", Recall)
print("Specificity: ", Specificity)
```

```
Precision:  0.6438356164383562
Recall (Sensitivity):  0.5802469135802469
Specificity:  0.8266666666666667
```

The above values indicate the following: Precision tells us, when the model predicts yes, how often it is correct Recall tells us, when the actual value is positive, how often it is correct Specificity , or true negative rate, tells us the proportion of actual negatives that are correctly identified

ROC AUC Score

```
ROC_AUC = metrics.roc_auc_score(y_test, nb_y_pred)
print("ROC AUC Score: ", ROC_AUC)
```

```
ROC AUC Score:  0.7034567901234567
```

This tells us how much the model is capable of distinguishing between classes; so the higher the score, the better the model is. The score is 70.3% , which means the model is performing well.

F-measure Shows the balance between Precision and Recall

```
f1 = (2*Precision*Recall)/(Precision+Recall)
print("F1 Score: ", f1)
```

```
F1 Score:  0.6103896103896104
```