In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Importing Dataset

In [2]:
```
df = pd.read_csv('diabetes.csv')
```

In [3]:
```
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Checking Null Value

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [5]:
```
x = df.drop(['Outcome'],axis = 1)

y = df['Outcome']
```

In [6]:
```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size = 0.20)
```

Decision Tree

In [7]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(xtrain,ytrain)

dtc.score(xtest,ytest)
```

Out[7]:
```
0.7207792207792207
```

In [8]:
```
pred_dtc = dtc.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_dtc)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
```

```
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_dtc))

from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_dtc))

from sklearn.metrics import classification_report

print(classification_report(ytest, pred_dtc))
```

```
Confusion Matrix :
 [[87 30]
 [13 24]]
Accuracy :  0.7207792207792207
specificity1 :  0.7435897435897436
Sensitivity :  0.6486486486486487
MCC :  0.351193939778555
AUC :  0.6961191961191961
              precision    recall  f1-score   support

           0       0.87      0.74      0.80       117
           1       0.44      0.65      0.53        37

    accuracy                           0.72       154
   macro avg       0.66      0.70      0.66       154
weighted avg       0.77      0.72      0.74       154
```

K-Nearest Neighbors

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 9)

knn.fit(xtrain,ytrain)
```

```
knn.score(xtest,ytest)
```

Out[9]: 0.7792207792207793

In [10]:

```
pred_knn = knn.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_knn)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_knn))

from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_knn))

from sklearn.metrics import classification_report

print(classification_report(ytest, pred_knn))
```

```
Confusion Matrix :
 [[94 23]
 [11 26]]
Accuracy :  0.7792207792207793
specificity1 :  0.8034188034188035
Sensitivity :  0.7027027027027027
MCC :  0.46425425741937526
AUC :  0.753060753060753
              precision    recall  f1-score   support
```

```
              0       0.90      0.80      0.85       117
              1       0.53      0.70      0.60        37

       accuracy                           0.78       154
      macro avg       0.71      0.75      0.73       154
   weighted avg       0.81      0.78      0.79       154
```

Random Forest

In [11]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(xtrain,ytrain)

rfc.score(xtest,ytest)
```

Out[11]:
```
0.7857142857142857
```

In [12]:
```
pred_rfc = rfc.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_rfc)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_rfc))
```

```
from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_rfc))

from sklearn.metrics import classification_report

print(classification_report(ytest, pred_rfc))
```

```
Confusion Matrix :
 [[95 22]
 [11 26]]
Accuracy :  0.7857142857142857
specificity1 :  0.811965811965812
Sensitivity :  0.7027027027027027
MCC :  0.47473127343114396
AUC :  0.7573342573342574
              precision    recall  f1-score   support

           0       0.90      0.81      0.85       117
           1       0.54      0.70      0.61        37

    accuracy                           0.79       154
   macro avg       0.72      0.76      0.73       154
weighted avg       0.81      0.79      0.79       154
```

Logistic Regression

In [13]:

```
from sklearn.linear_model import LogisticRegression

logReg = LogisticRegression()

logReg.fit(xtrain,ytrain)

logReg.score(xtest,ytest)
```

```
C:\Users\janoj\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[13]: 0.7792207792207793

In [14]:
```
pred_logReg = logReg.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_logReg)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_logReg))

from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_logReg))

from sklearn.metrics import classification_report

print(classification_report(ytest, pred_logReg))
```

```
Confusion Matrix :
 [[96 21]
 [13 24]]
Accuracy :  0.7792207792207793
specificity1 :  0.8205128205128205
Sensitivity :  0.6486486486486487
MCC :  0.4407543676618945
AUC :  0.7345807345807346
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       117
           1       0.53      0.65      0.59        37
```

```
              precision    recall  f1-score   support

    accuracy                         0.78       154
   macro avg       0.71      0.73      0.72       154
weighted avg       0.80      0.78      0.79       154
```

Support Vector Machine (Linear)

In [15]:
```
from sklearn.svm import SVC

svml = SVC(kernel = 'linear')

svml.fit(xtrain,ytrain)

svml.score(xtest,ytest)
```

Out[15]: 0.7987012987012987

In [16]:
```
pred_svml = svml.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_svml)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_svml))

from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_svml))
```

```
from sklearn.metrics import classification_report

print(classification_report(ytest, pred_svml))
```

```
Confusion Matrix :
 [[97 20]
 [11 26]]
Accuracy :  0.7987012987012987
specificity1 :  0.8290598290598291
Sensitivity :  0.7027027027027027
MCC :  0.4963873813774586
AUC :  0.7658812658812658
              precision    recall  f1-score   support

           0       0.90      0.83      0.86       117
           1       0.57      0.70      0.63        37

    accuracy                           0.80       154
   macro avg       0.73      0.77      0.74       154
weighted avg       0.82      0.80      0.81       154
```

Support Vector Machine (RBF)

In [17]:
```
from sklearn.svm import SVC

svmr = SVC(kernel = 'rbf')

svmr.fit(xtrain,ytrain)

svmr.score(xtest,ytest)
```

Out[17]:
```
0.8311688311688312
```

In [18]:
```
pred_svmr = svmr.predict(xtest)

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(ytest,pred_svmr)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))

accuracy1=(cm1[0,0]+cm1[1,1])/total1
```

```
print ('Accuracy : ', accuracy1)

specificity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('specificity1 : ', specificity1)

sensitivity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity1)

from sklearn.metrics import matthews_corrcoef

print("MCC : ",matthews_corrcoef(ytest,pred_svmr))

from sklearn.metrics import roc_auc_score

print("AUC : ",roc_auc_score(ytest,pred_svmr))

from sklearn.metrics import classification_report

print(classification_report(ytest, pred_svmr))
```

```
Confusion Matrix :
 [[106  11]
 [ 15  22]]
Accuracy :  0.8311688311688312
specificity1 :  0.905982905982906
Sensitivity :  0.5945945945945946
MCC :  0.5212132688864239
AUC :  0.7502887502887504
              precision    recall  f1-score   support

           0       0.88      0.91      0.89       117
           1       0.67      0.59      0.63        37

    accuracy                           0.83       154
   macro avg       0.77      0.75      0.76       154
weighted avg       0.83      0.83      0.83       154
```