Name: Saloni Vishwakarma

Roll No: C1-13

Import Libraries

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Import Dataset

```
d1=pd.read_csv('Diabetes.csv')
print(d1)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
```

```
766                    0.349   47        1
767                    0.315   23        0

[768 rows x 9 columns]
```

```
d1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
d1.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	I
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

```
d2=d1.loc[(d1['Glucose']!=0) & (d1['BloodPressure']!=0) &
          (d1['SkinThickness']!=0) & (d1['Insulin']!=0) & (d1['BMI']!=0)]
```

```
d2.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	I
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	

count	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
mean	3.301020	122.627551	70.663265	29.145408	156.056122	33.086224
std	3.211424	30.860781	12.496092	10.516424	118.841690	7.027659
min	0.000000	56.000000	24.000000	7.000000	14.000000	18.200000
25%	1.000000	99.000000	62.000000	21.000000	76.750000	28.400000
50%	2.000000	119.000000	70.000000	29.000000	125.500000	33.200000
75%	5.000000	143.000000	78.000000	37.000000	190.000000	37.100000
max	17.000000	198.000000	110.000000	63.000000	846.000000	67.100000

```
d1['Glucose'].replace(0,d2['Glucose'].mean(),inplace=True)
d1['BloodPressure'].replace(0,d2['BloodPressure'].mean(),inplace=True)
d1['SkinThickness'].replace(0,d2['SkinThickness'].mean(),inplace=True)
d1['Insulin'].replace(0,d2['Insulin'].mean(),inplace=True)
d1['BMI'].replace(0,d2['BMI'].mean(),inplace=True)
```

```
d1.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	I
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.692888	72.325800	29.151052	155.795560	32.466469	
std	3.369578	30.436043	12.101807	8.790943	85.021487	6.875558	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	
50%	3.000000	117.000000	72.000000	29.145408	156.056122	32.400000	
75%	6.000000	140.250000	80.000000	32.000000	156.056122	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

Splitting Dataset

```
x=d1.iloc[:,0:8]
y=d1.iloc[:,8]
print(x.shape)
print(y.shape)
```

```
(768, 8)
(768,)
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(614, 8)
(614,)
(154, 8)
(154,)
```

Creating Classification Model

```
rfclassifier=RandomForestClassifier()

rfclassifier=rfclassifier.fit(x_train,y_train)
y_pred=rfclassifier.predict(x_test)
```

```
print("Actual Values:")
print(y_test)
print("Predicted values:")
print(y_pred)
```

```
Actual Values:
285    0
101    0
581    0
352    0
726    0
      ..
563    0
318    0
154    1
684    0
643    0
Name: Outcome, Length: 154, dtype: int64
Predicted values:
[0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0
 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 0 0
 1 1 1 0 0 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0
 0 0 0 1 1 0]
```

```
d2=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
d2.head()
```

	Actual	Predicted
285	0	0
101	0	0
581	0	0
352	0	0
726	0	0

Model Evaluation

```
print("accuracy",metrics.accuracy_score(y_test,y_pred))
```

```
accuracy 0.7922077922077922
```

```
cf=confusion_matrix(y_test,y_pred)
```

```
print(cf)
```

```
[[88 11]
 [21 34]]
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        99
           1       0.76      0.62      0.68        55

    accuracy                           0.79       154
   macro avg       0.78      0.75      0.76       154
weighted avg       0.79      0.79      0.79       154
```

KFold Cross Validation

```
x=d1.iloc[:,0:8]
y=d1.iloc[:,8]
```

```
rfclassifier=RandomForestClassifier()
```

```
import sklearn.metrics as metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn import model_selection
kf = KFold(n_splits=5)
cv_results = model_selection.cross_val_score(rfclassifier, x, y, cv=kf)
```

```
print(cv_results)
```

```
[0.72077922 0.73376623 0.77922078 0.83660131 0.73202614]
```

```
print(cv_results.mean())
```

```
0.7604787369493253
```