Name: Saloni Vishwakarma

Roll No: C1-13

Import Libraries

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
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 seaborn as sns
```

Import Dataset

```
pd.set_option('display.max_rows', None)
d1=pd.read_csv('Diabetes.csv')
print(d1)
d1.info()
```

```
     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
5              5      116             74              0        0  25.6
6              3       78             50             32       88  31.0
7             10      115              0              0        0  35.3
8              2      197             70             45      543  30.5
9              8      125             96              0        0   0.0
10             4      110             92              0        0  37.6
11            10      168             74              0        0  38.0
12            10      139             80              0        0  27.1
13             1      189             60             23      846  30.1
14             5      166             72             19      175  25.8
15             7      100              0              0        0  30.0
16             0      118             84             47      230  45.8
17             7      107             74              0        0  29.6
18             1      103             30             38       83  43.3
19             1      115             70             30       96  34.6
```

```
20             3      126             88             41      235  39.3
21             8       99             84              0        0  35.4
22             7      196             90              0        0  39.8
23             9      119             80             35        0  29.0
24            11      143             94             33      146  36.6
25            10      125             70             26      115  31.1
26             7      147             76              0        0  39.4
27             1       97             66             15      140  23.2
28            13      145             82             19      110  22.2
29             5      117             92              0        0  34.1
30             5      109             75             26        0  36.0
31             3      158             76             36      245  31.6
32             3       88             58             11       54  24.8
33             6       92             92              0        0  19.9
34            10      122             78             31        0  27.6
35             4      103             60             33      192  24.0
36            11      138             76              0        0  33.2
37             9      102             76             37        0  32.9
38             2       90             68             42        0  38.2
39             4      111             72             47      207  37.1
40             3      180             64             25       70  34.0
41             7      133             84              0        0  40.2
42             7      106             92             18        0  22.7
43             9      171            110             24      240  45.4
44             7      159             64              0        0  27.4
45             0      180             66             39        0  42.0
46             1      146             56              0        0  29.7
47             2       71             70             27        0  28.0
48             7      103             66             32        0  39.1
49             7      105              0              0        0   0.0
50             1      103             80             11       82  19.4
51             1      101             50             15       36  24.2
52             5       88             66             21       23  24.4
53             8      176             90             34      300  33.7
54             7      150             66             42      342  34.7
55             1       73             50             10        0  23.0
56             7      187             68             39      304  37.7
```

Explore Data

```
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
 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
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
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
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
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

Creating Classification Model

```
DC=DecisionTreeClassifier()
```

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

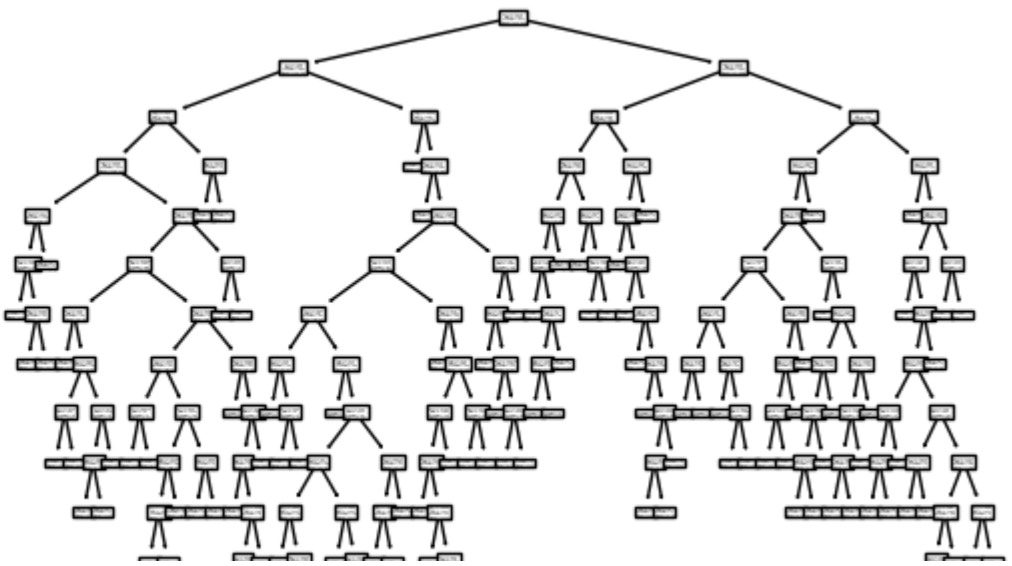
```
print(y_pred)
```

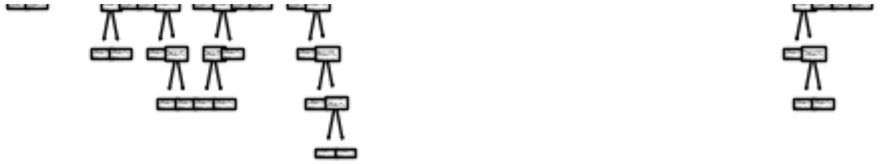
```
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0
 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0
 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0
 0 0 0 0 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

```
tree.plot_tree(DC)
plt.savefig('o1.pdf')
```





Model Evaluation Metrics

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

```
     accuracy 0.7012987012987013
```

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

```
print(cf)
```

```
     [[77 22]
      [24 31]]
```

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

```
                   precision    recall  f1-score   support

                0       0.76      0.78      0.77        99
                1       0.58      0.56      0.57        55

         accuracy                           0.70       154
        macro avg       0.67      0.67      0.67       154
     weighted avg       0.70      0.70      0.70       154
```

Accuracy = (TP+TN) / (TP+FP+TN+FN)

Precision tells us how many of the correctly predicted cases turned out to be positive.

Precision= TP / (TP+FP)

Recall tells us how many of the actual positive cases we were able to predict correctly with our model. Recall = TP / (TP+FN)

F1 Score = 2 / ((1/Recall) + (1/Precision))

Support is the number of actual occurrences of the class in the specified dataset.

KFold Cross Validation

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

```
DC=DecisionTreeClassifier()
```

```
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(DC, x, y, cv=kf)
```

```
print(cv_results)
```

```
[0.65584416 0.61038961 0.74025974 0.75163399 0.73202614]
```

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

```
0.6980307274424922
```