Decision Tree Classification Model

In [50]:

```
# Step-1: Importing the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [28]:

```
# Step-2: Load Data set
dataset= pd.read_csv("E:\\mylab\\dataset\\processed.cleveland.data.csv", names=['age','sex'
dataset_mean= dataset
```

In [40]:

```
dataset.tail()
```

Out[40]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
298	45	1	1	110	264	0	0	132	0	1.2	2	0.000000	7.(
299	68	1	4	144	193	1	0	141	0	3.4	2	2.000000	7.(
300	57	1	4	130	131	0	0	115	1	1.2	2	1.000000	7.(
301	57	0	2	130	236	0	2	174	0	0.0	2	1.000000	3.(
302	38	1	3	138	175	0	0	173	0	0.0	1	0.672241	3.(

In [41]:

```
#Step-3: Data Preprocessing

# Filling missing values Statistics measures
print("*****Before Fill Missing values Row 166,192,287,302********")
print(dataset_mean.loc[287])
dataset1=dataset_mean
df1=pd.DataFrame(dataset1)
#print(df1)

print("-------- Mean of Column 11 'ca' --------")
print(df1['ca'].mean())
df1.fillna(df1.mean(), inplace=True)
print("*****After Fill Missing values Row 166,192,287,302********")
print(df1.loc[[166,192,287,302]])

print("-------- Mean of Column 12 'thal' --------")
print(df1['thal'].mean())
df1.fillna(df1.mean(), inplace=True)
print("*****After Fill Missing values Row 87,266********")
print(df1.loc[[87,266]])


#dataset=df1
#print(dataset)
```

```
*****Before Fill Missing values Row 166,192,287,302********
age          58.000000
sex           1.000000
cp            2.000000
trestbps    125.000000
chol        220.000000
fbs           0.000000
restecg       0.000000
thalach     144.000000
exang         0.000000
oldpeak       0.400000
slope         2.000000
ca            0.672241
thal          7.000000
output        0.000000
Name: 287, dtype: float64
-------- Mean of Column 11 'ca' --------
0.6722408026755853
*****After Fill Missing values Row 166,192,287,302********
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
166   52    1   3       138   223    0        0      169      0      0.0
192   43    1   4       132   247    1        2      143      1      0.1
287   58    1   2       125   220    0        0      144      0      0.4
302   38    1   3       138   175    0        0      173      0      0.0

     slope        ca  thal  output
166      1  0.672241   3.0       0
192      2  0.672241   7.0       1
287      2  0.672241   7.0       0
302      1  0.672241   3.0       0
-------- Mean of Column 12 'thal' --------
4.73421926910299
```

```
*****After Fill Missing values Row 87,266********
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
87    53    0   3       128   216    0        2      115      0      0.0
266   52    1   4       128   204    1        0      156      1      1.0

     slope   ca      thal  output
87       1  0.0  4.734219       0
266      2  0.0  4.734219       2
```

In [8]:

```
# Data Set Description
#Number of patients
n_patients = dataset.shape[0]

#Number of features
n_features = dataset.shape[1]-1

dataset["output"].replace(to_replace=[1,2,3,4],value=1,inplace=True)

#With Heart disease
heart_disease = dataset[dataset['output'] == 1].shape[0]

#Without Heart Disease (Healthy individuals)
no_heart_disease = dataset[dataset['output'] == 0].shape[0]

#Result Output
print("Total number of patients: {} ".format(n_patients))
print("Number of features: {}".format(n_features))
print("Number of patients with heart disease: {}".format(heart_disease))
print("Number of patients without heart disease: {}".format(no_heart_disease))
```

```
Total number of patients: 303
Number of features: 13
Number of patients with heart disease: 139
Number of patients without heart disease: 164
```

In [13]:

```
# Extract feature columns
feature_cols = list(dataset.columns[0:13])


# Show the list of columns
print("Feature columns:\n{}".format(feature_cols))
```

```
Feature columns:
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exan
g', 'oldpeak', 'slope', 'ca', 'thal']
```

In [14]:

```
# Separate the data into feature data and target data (X_all and y_all, respectively)
X= dataset[feature_cols]
y= dataset['output'].values

# Show the feature information by printing the first five rows
print("\nFeature values:")
X.head()
```

Feature values:

Out[14]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0

In [16]:

```
# Step-3: Split the Dataset into Traning and Testing data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=5)
print(X_train)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
3     37    1   3       130   250    0        0      187      0      3.5
55    54    1   4       124   266    0        2      109      1      2.2
225   34    0   2       118   210    0        0      192      0      0.7
224   63    0   4       108   269    0        0      169      1      1.8
75    65    0   3       160   360    0        2      151      0      0.8
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
8     63    1   4       130   254    0        2      147      0      1.4
73    65    1   4       110   248    0        2      158      0      0.6
118   63    1   4       130   330    1        2      132      1      1.8
189   69    1   3       140   254    0        2      146      0      2.0
206   58    1   4       128   259    0        2      130      1      3.0

     slope   ca  thal
3        3  0.0   3.0
55       2  1.0   7.0
225      1  0.0   3.0
224      2  2.0   3.0
75       1  0.0   3.0
..     ...  ...   ...
8        2  1.0   7.0
73       1  2.0   6.0
118      1  3.0   7.0
189      2  3.0   7.0
206      2  2.0   7.0

[212 rows x 13 columns]
```

In [17]:

```
#The dimension of Training and Testing Data
print(X_train.shape)
print (X_test.shape)
```

```
(212, 13)
(91, 13)
```

In [39]:

```
# Normalization Step
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(X_train)
X_train = scaler.transform(X_train)
print("----After Z-score Normalization on X_train-------")
print(X_train)

scaler.fit(X_test)
X_test = scaler.transform(X_test)
print("----After Z-score Normalization on X_test-------")
print(X_test)
```

```
----After Z-score Normalization on X_train-------
[[-1.91736161  0.67975655 -0.16656264 ...  2.36151212 -0.68283167
  -0.93461042]
 [-0.06178394  0.67975655  0.8720044  ...  0.68151021  0.3635441
   1.13614677]
 [-2.24481649 -1.47111492 -1.20512967 ... -0.9984917  -0.68283167
  -0.93461042]
 ...
 [ 0.92058071  0.67975655  0.8720044  ... -0.9984917   2.45629564
   1.13614677]
 [ 1.57549048  0.67975655 -0.16656264 ...  0.68151021  2.45629564
   1.13614677]
 [ 0.37482257  0.67975655  0.8720044  ...  0.68151021  1.40991987
   1.13614677]]
----After Z-score Normalization on X_test-------
[[-1.85828815  0.70128687 -0.16222142 ... -0.9335927  -0.05302469
  -0.81856114]
 [ 0.78936134  0.70128687 -2.2710999  ...  0.58349544  1.48316063
  -0.81856114]
 [-1.62805776  0.70128687  0.89221782 ... -0.9335927  -0.83079106
   1.26892886]
 ...
 [ 1.48005251  0.70128687  0.89221782 ...  0.58349544  1.48316063
   1.26892886]
 [ 0.78936134  0.70128687  0.89221782 ...  0.58349544  0.32618478
  -0.81856114]
 [ 0.44401575  0.70128687 -0.16222142 ... -0.9335927   1.48316063
   1.26892886]]
```

In [53]:

```
# Step-4: Fit the Model using ID3 Classifier
# training and prediction through a Descision Tree Classifier
for i in range(1,6):
    dtree_model = DecisionTreeClassifier(criterion='entropy', max_depth =i)
    dtree_model.fit(X_train, y_train)   # Training
    dtree_predictions = dtree_model.predict(X_test) # Testing
    # accuracy on X_test
    accuracy = accuracy_score(dtree_predictions, y_test)
    print("Accuracy for depth="+str(i)+":",accuracy)
    # creating a confusion matrix
    cm = confusion_matrix(y_test, dtree_predictions)
    print(cm)
```

```
Accuracy for depth=1: 0.7802197802197802
[[44  9]
 [11 27]]
Accuracy for depth=2: 0.7252747252747253
[[31 22]
 [ 3 35]]
Accuracy for depth=3: 0.8461538461538461
[[46  7]
 [ 7 31]]
Accuracy for depth=4: 0.8571428571428571
[[48  5]
 [ 8 30]]
Accuracy for depth=5: 0.7582417582417582
[[38 15]
 [ 7 31]]
```

In [52]:

```
# Step-5: Test the Model using CART Classifier
# training and prediction through a Descision Tree Classifier
for i in range(1,6):
    dtree_model = DecisionTreeClassifier(criterion='gini',max_depth =i)
    dtree_model.fit(X_train, y_train)
    dtree_predictions = dtree_model.predict(X_test)
    #print(dtree_predictions)
    # accuracy on X_test
    accuracy = accuracy_score(y_test,dtree_predictions)
    print("Accuracy for depth="+str(i)+":",accuracy)
    # creating a confusion matrix
    cm = confusion_matrix(y_test, dtree_predictions)
    print(cm)
```

```
Accuracy for depth=1: 0.7802197802197802
[[44  9]
 [11 27]]
Accuracy for depth=2: 0.6923076923076923
[[35 18]
 [10 28]]
Accuracy for depth=3: 0.8571428571428571
[[47  6]
 [ 7 31]]
Accuracy for depth=4: 0.7802197802197802
[[41 12]
 [ 8 30]]
Accuracy for depth=5: 0.7472527472527473
[[37 16]
 [ 7 31]]
```

In [ ]: