AML Lab Assignment-2:

Name: Gurvinder Kaur Matharu
PRN: 1032230432
Roll No.: PA14

AIM:

Implement Tree based Classifiers

THEORY:

Tree-based classifiers are machine learning algorithms that use decision trees as a predictive model. They include algorithms like Random Forest, Gradient Boosting, and AdaBoost, which construct a hierarchy of decision rules or trees to make predictions based on input features.

Decision trees are hierarchical structures that make sequential decisions by splitting data into smaller subsets based on the most significant features. They represent a flowchart-like structure where nodes represent features, branches depict decisions, and leaves signify the outcomes or predictions. By recursively partitioning the data, decision trees form a set of rules that facilitate classification or regression tasks in machine learning.

Random Forest is an ensemble learning method that constructs multiple decision trees during training. It combines predictions from various trees to improve accuracy and reduce overfitting by aggregating their outputs through a voting or averaging mechanism. By randomly selecting subsets of features and data samples for each tree, it promotes diversity among individual trees, enhancing the overall predictive power of the model.

Gradient boosting is an ensemble learning technique that builds a predictive model by sequentially combining weak learners (usually decision trees) to minimize errors by focusing on the mistakes of prior models, resulting in a strong, accurate predictor.

CODE EXECUTION & OUTPUT:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn import tree
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('heart.csv')
```

```
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
```

```
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
data.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	rest
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000

```
data.shape
```

```
(1025, 14)
```

Considering features that have a high correlation with the target variable

```
X = data[['age', 'sex', 'cp', 'thalach','exang', 'oldpeak', 'slope', 'ca', 'thal']]
y = data['target']
```

Splitting the dataset into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model = DecisionTreeClassifier(max_depth=6)
```

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6)
```

```
y_pred = model.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
0.8926829268292683
```

```
report = classification_report(y_test, y_pred, target_names=['Class 0', 'Class 1'])
```

```
print(report)
```

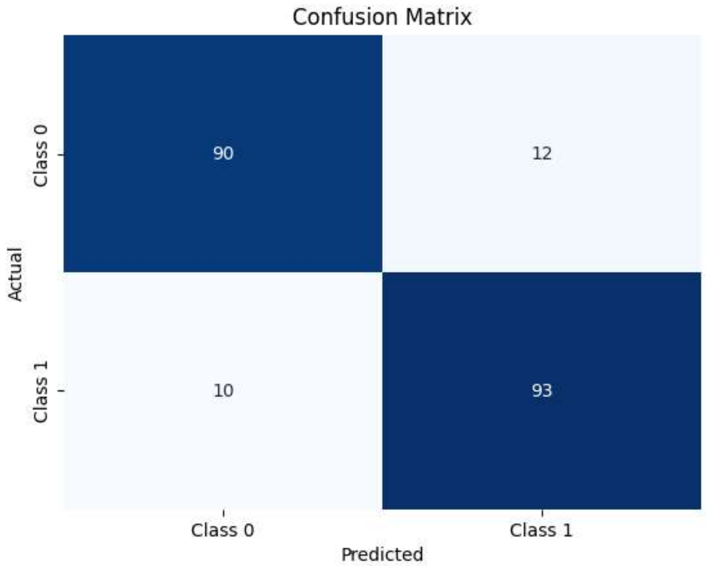
```
              precision    recall  f1-score   support

     Class 0       0.90      0.88      0.89       102
     Class 1       0.89      0.90      0.89       103

    accuracy                           0.89       205
   macro avg       0.89      0.89      0.89       205
weighted avg       0.89      0.89      0.89       205
```

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

```
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.title(f'Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
fig = plt.figure(figsize=(15,8))
tree.plot_tree(model, feature_names=['age', 'sex', 'cp', 'thalach','exang', 'oldpeak', 'slope', 'ca', 'thal'], class_names=['Class 0', 'Class
```