**U19EC046 | ML | LAB 5**

**AIM**

Write a program to implement the Decision Tree classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets. Also generate the random forest classifier.

**THEORY:**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.​

The decisions or the test are performed on the basis of features of the given dataset.​ It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:​

* Information Gain​
* Gini Index​

**Information Gain**​:

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.​
* It calculates how much information a feature provides us about a class.​
* According to the value of information gain, we split the node and build the decision tree.​
* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

**Gini Index:**

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.​
* An attribute with the low Gini index should be preferred as compared to the high Gini index.​
* It only creates binary splits, and the CART (Classification and Regression Tree) algorithm uses the Gini index to create binary splits.

**ALGORITHM**

1. Import necessary libraries
2. Read the dataset using pandas
3. Split the dataset into features and results
4. Split the features and results into training and testing dataset using scikit learn

test\_train\_split

1. Apply pre-processing if needed, use scikit-learn preprocessing class
2. Create an instance of Decision Tree classifier using gini or entropy
3. Train the model using fit method
4. Predict the result using predict method on model
5. Find the accuracy and confusion matrix using scikit-learn matrices

**CODE**

1. Decision Tree Classification

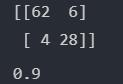
|  |
| --- |
| **import numpy as np**  **import matplotlib.pyplot as plt**  **import pandas as pd**  **dataset = pd.read\_csv('Social\_Network\_Ads.csv')**  **X = dataset.iloc[:, :-1].values**  **y = dataset.iloc[:, -1].values**  **from sklearn.model\_selection import train\_test\_split**  **X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)**  **from sklearn.preprocessing import StandardScaler**  **sc = StandardScaler()**  **X\_train = sc.fit\_transform(X\_train)**  **X\_test = sc.transform(X\_test)**  **from sklearn.tree import DecisionTreeClassifier**  **classifier = DecisionTreeClassifier(criterion = 'gini', random\_state = 0)    *# or use criterion = 'gini'***  **classifier.fit(X\_train, y\_train)**  **y\_pred = classifier.predict(X\_test)**  **y\_pred = classifier.predict(X\_test)**  **print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))**  **from sklearn.metrics import confusion\_matrix, accuracy\_score**  **cm = confusion\_matrix(y\_test, y\_pred)**  **print(cm)**  **accuracy\_score(y\_test, y\_pred)** |

1. Random forest Classifier

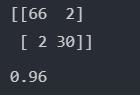
|  |
| --- |
| **import numpy as np**  **import matplotlib.pyplot as plt**  **import pandas as pd**  **dataset = pd.read\_csv('Social\_Network\_Ads.csv')**  **X = dataset.iloc[:, :-1].values**  **y = dataset.iloc[:, -1].values**  **from sklearn.model\_selection import train\_test\_split**  **X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)**  **from sklearn.preprocessing import StandardScaler**  **sc = StandardScaler()**  **X\_train = sc.fit\_transform(X\_train)**  **X\_test = sc.transform(X\_test)**  **from sklearn.ensemble import RandomForestClassifier**  **classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 2837)**  **classifier.fit(X\_train, y\_train)**  **y\_pred = classifier.predict(X\_test)**  **print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))**  **from sklearn.metrics import confusion\_matrix, accuracy\_score**  **cm = confusion\_matrix(y\_test, y\_pred)**  **print(cm)**  **accuracy\_score(y\_test, y\_pred)** |

**OUTPUT**

1. Decision tree



1. Random Forest Classifier



**CONCLUSION**

In this practical we have studied and implements decision tree classifier using scikit-learn library. We also learn about different attribute selection measures and usually gini index is used as it considers both positive and negative sample.