NAME : Mehul Uttam
PRN : 1032222936
TY CSE AIDS - A (A1 Batch)

Assignment 3

Topic: - Implementation of Tree based Classifiers (Decision tree).

Algorithms used: Decision tree

Theory:-

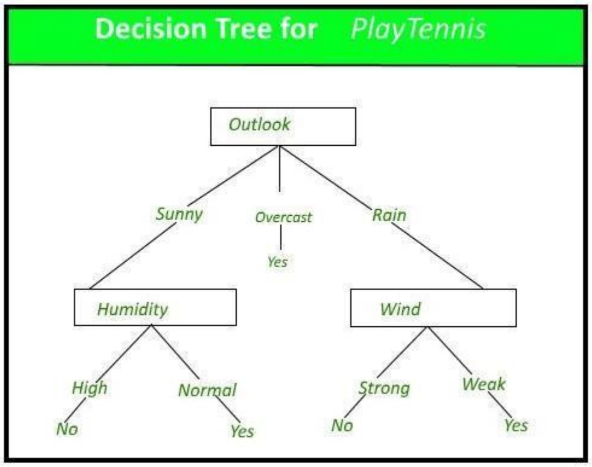
1) Tree based classifiers:-

There are two types of Tree based classifiers.

a) Decision tree:-

A decision tree is supervised machine learning algorithm that can be used for both classification and regression. A decision tree is simply a series of sequential decisions made to reach a specific result.

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Advantages:

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
5. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

Disadvantage:

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
3. Decision tree often involves higher time to train the model.
4. Decision tree training is relatively expensive as the complexity and time has taken are more.
5. The Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

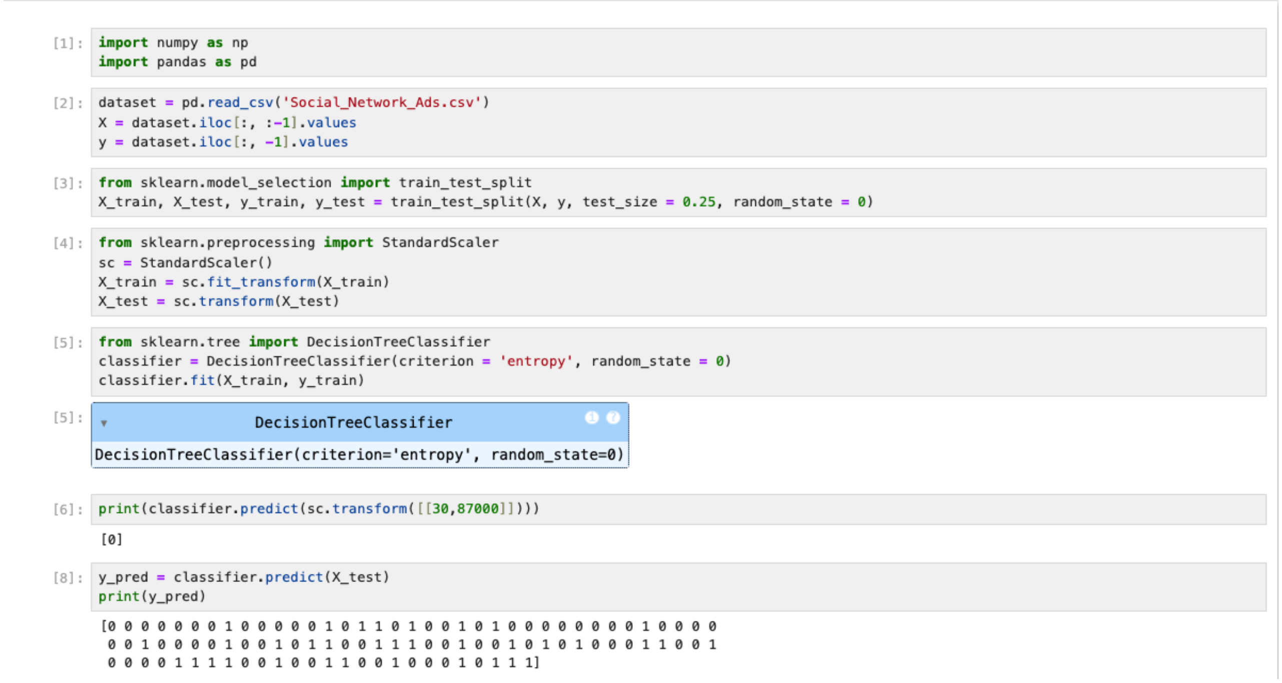
Methodology:

We will start by loading the dataset and splitting it into training and testing sets using a 70/30 split. We will then fit a decision tree classifier to the training data using the default hyperparameters in scikit-learn. We will evaluate the performance of our model on the testing data using the accuracy score and the confusion matrix.

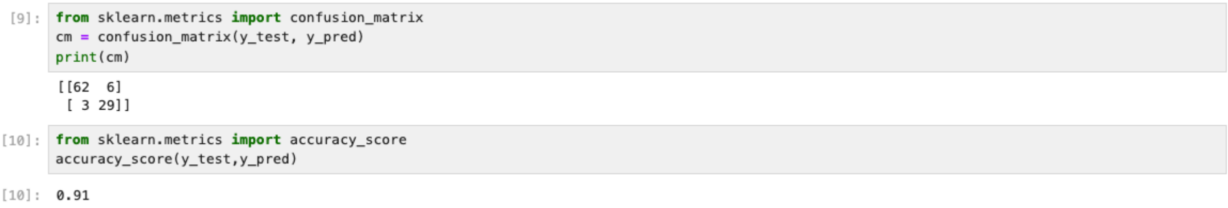
Implementation:-

1. Read the .csv file of dataset
2. Display few observations
3. Perform data preprocessing(handling missing data, etc)
4. Create the independent and dependent variables
5. Standardization of data
6. Plot few graphs to understand/explore the data.
7. Perform feature importance and find the most significant features.
8. Split the data into training and test sets.
9. Create the objects of classifiers.
10. Fit the data in model to train it.
11. Analyze the performance of the classifiers.

Results:



```
[1]: import numpy as np
     import pandas as pd

[2]: dataset = pd.read_csv('Social_Network_Ads.csv')
     X = dataset.iloc[:, :-1].values
     y = dataset.iloc[:, -1].values

[3]: 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)

[4]: from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)

[5]: from sklearn.tree import DecisionTreeClassifier
     classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
     classifier.fit(X_train, y_train)

[5]: ▾          DecisionTreeClassifier
     DecisionTreeClassifier(criterion='entropy', random_state=0)

[6]: print(classifier.predict(sc.transform([[30,87000]])))

     [0]

[8]: y_pred = classifier.predict(X_test)
     print(y_pred)

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



```
[9]:  from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test, y_pred)
      print(cm)

      [[62  6]
       [ 3 29]]

[10]: from sklearn.metrics import accuracy_score
      accuracy_score(y_test,y_pred)

[10]: 0.91
```

Conclusion:

In this lab, we implemented a decision tree classifier to predict whether a person is likely to purchase a product or not based on their age, income. performance analysis of decision Tree classifier was done.

FAQs:

1) What is the Decision Tree classifier?

Decision Tree classifier is a supervised learning algorithm used for classification tasks. It splits the dataset into subsets based on the most significant features, using a tree-like structure of decisions and their possible consequences. Each internal node represents a decision based on a feature, each branch represents the outcome of the decision, and each leaf node represents a class label. The goal is to predict the target variable by following the decision rules from the root to a leaf.

2) What are some advantages of decision trees?

- **Simple and Easy to Interpret**: Decision trees are easy to visualize and understand, making them suitable for interpreting model decisions.

- **Handles Both Numerical and Categorical Data**: Unlike many other algorithms, decision trees can work well with both types of data.

- **Requires Little Data Preparation**: Decision trees don’t require feature scaling or normalization.

- **Handles Nonlinear Relationships**: Decision trees can model complex, nonlinear relationships between features and the target variable.

- **Robust to Outliers**: They are less sensitive to outliers as the splitting decision is based on majority rules.

3) How does a decision tree work?

A decision tree works by recursively splitting the dataset into smaller subsets based on feature values. At each node, the algorithm selects the feature that results in the most significant reduction in **impurity** (using measures like **Gini Index**, **Entropy**, or **Variance** for regression). The process continues until a stopping criterion is met (e.g., maximum tree depth or minimum number of samples in a node). The final prediction is made by following the decision path from the root node to a leaf node, where the class label or predicted value is assigned.

4) How do you prevent overfitting in a decision tree?

- **Pruning**: Reduce the size of the tree by removing branches that provide little power in predicting the target variable (see more on pruning below).
- **Set Maximum Depth**: Limit the depth of the tree so that it doesn’t grow too complex.
- **Minimum Samples per Leaf**: Restrict the minimum number of samples required to form a leaf node.
- **Minimum Samples per Split**: Require a minimum number of samples before allowing a node to split.
- **Cross-Validation**: Use cross-validation techniques to check the generalization of the model and avoid overfitting.

5) What is pruning in decision trees?

- **Pruning** is a technique used to reduce the size of a decision tree by removing nodes that provide little power in improving prediction accuracy. It helps prevent overfitting by simplifying the tree structure. There are two types of pruning:
    - **Pre-pruning (Early Stopping)**: Stop growing the tree once a specific condition is met (e.g., maximum depth, minimum number of samples).
    - **Post-pruning**: First grow the tree fully and then remove branches that have little impact on model accuracy, based on validation set performance.

Pruning ensures the model generalizes better to unseen data by reducing complexity while maintaining good performance.