

Name: Girish Nandanwar	Course: 22CT744 – Lab. Machine Learning
Roll No: A-56	Department: Computer Technology

Practical No.7

Aim: To implement Decision Trees classifier

Theory:

1. Basic Concept

- The tree consists of:
 - Root Node: Represents the entire dataset and the first feature chosen for splitting.
 - Decision Nodes: Intermediate nodes where data is split based on certain feature thresholds.
 - Leaf Nodes: Represent the final class labels or output values.

The goal is to create branches that lead to the most homogeneous subsets possible.

2. How It Works

1. The algorithm evaluates all available features and selects the one that best separates the data.
 2. It uses measures like Information Gain (Entropy-based) or Gini Index to decide the best split.
 3. This process continues recursively until:
 - The data is perfectly classified, or
 - A stopping condition is reached (e.g., max depth or min samples per leaf).
-

3. Splitting Criteria

- Entropy: Measures impurity. Lower entropy = higher purity.

$$\text{Entropy} = -\sum p_i \log_2(p_i) \quad \text{Entropy} = -\sum p_i \log_2(p_i)$$

- Information Gain: Reduction in entropy after a dataset is split on an attribute.

$$\text{IG} = \text{Entropy}_{\text{parent}} - \sum \frac{n_i}{n} \text{Entropy}_{\text{child}_i} \quad \text{IG} = \text{Entropy}_{\text{parent}} - \sum \frac{n_i}{n} \text{Entropy}_{\text{child}_i}$$

- Gini Index: Probability of misclassifying a sample.

$$\text{Gini} = 1 - \sum (p_i)^2 \quad \text{Gini} = 1 - \sum (p_i)^2$$

4. Advantages

- Easy to understand and interpret.
 - Requires little data preprocessing (no need for feature scaling).
 - Can handle both categorical and numerical data.
-

5. Disadvantages

- Prone to overfitting, especially with small datasets.
 - Small variations in data can result in a different tree structure.
 - Biased toward features with more levels.
-

6. Evaluation Metrics

To assess performance:

- Accuracy: Ratio of correctly predicted observations to total observations.

- Confusion Matrix: Summarizes true positives, false positives, etc.
- Precision, Recall, F1-Score: For class-wise performance analysis.

Code:

```
# Decision Tree Classification
```

```
# Importing the libraries
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
# Importing the dataset
```

```
dataset = pd.read_csv('Social_Network_Ads.csv')
```

```
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, -1].values
```

```
# Splitting the dataset into the Training set and Test set
```

```
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)
```

```
print(X_train)
```

```
print(y_train)
```

```
print(X_test)
```

```
print(y_test)
```

```
# Feature Scaling
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
print(X_test)
```

```
# Training the Decision Tree Classification model on the Training set
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
```

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

```
# Predicting a new result
```

```

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

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)

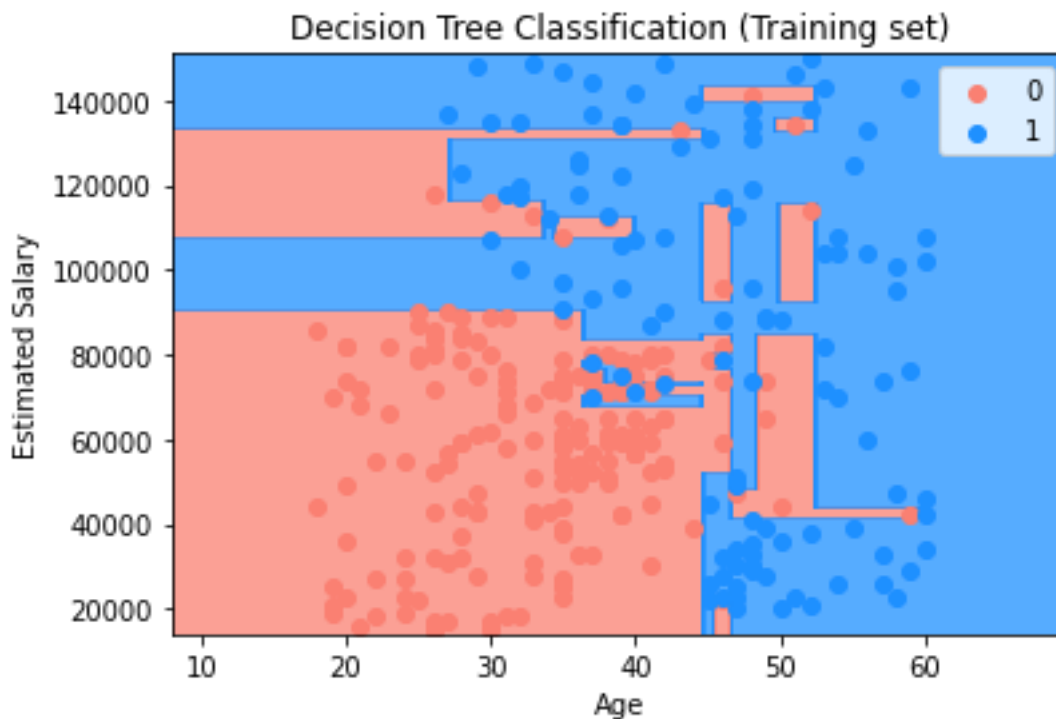
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train, y_train)
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),
X2.ravel()])).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test, y_test)
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),
X2.ravel()])).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)

```

```
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Result:



Conclusion:

Decision Trees are a foundational model for classification tasks and serve as the basis for more advanced ensemble methods like Random Forests and Gradient Boosted Trees. They are intuitive, interpretable, and effective for datasets with a mix of categorical and numerical features.