**Lab Manual: Decision Tree Classifiers**

**Objective:** To understand the working of Decision Tree Classifiers and their application in making decisions based on the restaurant dataset. The goal is to explore the features that contribute to the decision-making process and visualize the importance of each feature.

**Tools Used:**

* Python
* Libraries: pandas, sklearn, matplotlib

**Task 1: Understanding the Dataset**

**Step 1**: Load the dataset

The dataset contains various attributes related to dining out decisions, such as whether it's raining, the type of food, the price range, and whether a reservation is required. These attributes are used to predict whether a group will wait for a table at a restaurant.

# Import necessary libraries

import pandas as pd

# Define the column names manually

column\_names = ['Alternate', 'Bar', 'Fri/Sat', 'Hungry', 'Patrons', 'Price','Raining', 'Reservation', 'Type', 'WaitEstimate', 'Wait']

# Load the dataset without a header

data = pd.read\_csv('restaurant.csv', header=None, names=column\_names)

# Print the first few rows of the dataset to ensure it's loaded correctly

print("First few rows of the dataset:")

print(data.head())

**Step 2**: Analyze the structure of the dataset

Understand the types of variables and the target class Wait, which indicates if the group waits for a table.

**Task 2: Data Preprocessing**

**Step 1**: Prepare the features (X) and target (y) for model training

# Split the data into features (X) and target (y)

X = data.drop('Wait', axis=1)

y = data['Wait']

# Convert categorical variables to numerical using get\_dummies

X = pd.get\_dummies(X)

**Step 2**: Split the data into training and testing sets

from sklearn.model\_selection import train\_test\_split

# Split the data into training and testing sets (80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Task 3: Model Training**

**Step 1**: Train a Decision Tree Classifier on the training set

from sklearn.tree import DecisionTreeClassifier

# Train the Decision Tree Classifier

clf = DecisionTreeClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

**Step 2**: Evaluate the model's performance

from sklearn.metrics import accuracy\_score, classification\_report

# Predict the test set results

y\_pred = clf.predict(X\_test)

# Calculate the accuracy of the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Generate a classification report

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

**Task 4: Visualization**

**Step 1**: Visualize the decision tree

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

# Plot the Decision Tree

plt.figure(figsize=(20,10))

plot\_tree(clf, feature\_names=X.columns, class\_names=True, filled=True)

plt.show()

**Step 2**: Visualize the feature importance

# Get the feature importances

feature\_importances = clf.feature\_importances\_

# Create a DataFrame for better visualization of feature importances

features\_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature\_importances})

# Sort the DataFrame by importance

features\_df = features\_df.sort\_values(by='Importance', ascending=False)

# Plot the feature importances

plt.figure(figsize=(10, 6))

plt.barh(features\_df['Feature'], features\_df['Importance'], color='skyblue')

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.title('Feature Importance in Decision Tree')

plt.gca().invert\_yaxis() # To display the most important feature at the top

plt.show()

**Task 5: Analysis and Interpretation**

* **Decision Tree Plot**: Examine the plotted decision tree to understand how the model splits the data at each node based on different features.
* **Feature Importance**: Identify which features are most influential in making decisions.

**Conclusion:** Through this lab, you have learned how to implement a Decision Tree Classifier using the restaurant dataset. You also explored how to preprocess the data, train a model, evaluate its performance, and interpret the results through visualization techniques.