**Batch: HO-ML 1 Experiment Number: 05**

**Roll Number: 16010422234 Name: Chandana Galgali**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim of the Experiment: Classify the Iris dataset using the Decision tree classifier. Follow the steps given on the Kaggle website.**

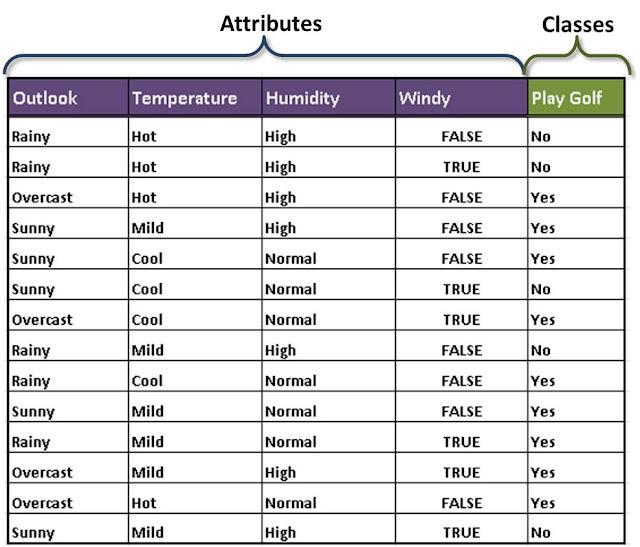
**Program/ Steps:**

Part 1:

1. Open this website <https://www.kaggle.com/code/shikhnu/decision-tree-iris-dataset>.
2. Follow the steps to classify the Iris dataset and display the results.

Part 2:

1. Use the same steps as Part 1 to classify the data in the following table.
2. Calculate the efficiency also.



**Output/Result:**

Part 1:

**# Part 1: Classifying the Iris dataset using a Decision Tree**

**import pandas as pd**

**from sklearn.datasets import load\_iris**

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.metrics import accuracy\_score**

**from sklearn import tree**

**import matplotlib.pyplot as plt**

**# Load Iris dataset**

**iris = load\_iris()**

**X = iris.data**

**y = iris.target**

**# Split dataset into training and testing sets (70% train, 30% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**# Create Decision Tree Classifier**

**clf = DecisionTreeClassifier(criterion='gini', random\_state=42)**

**# Train the classifier**

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

**# Predict the test data**

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

**# Calculate accuracy**

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

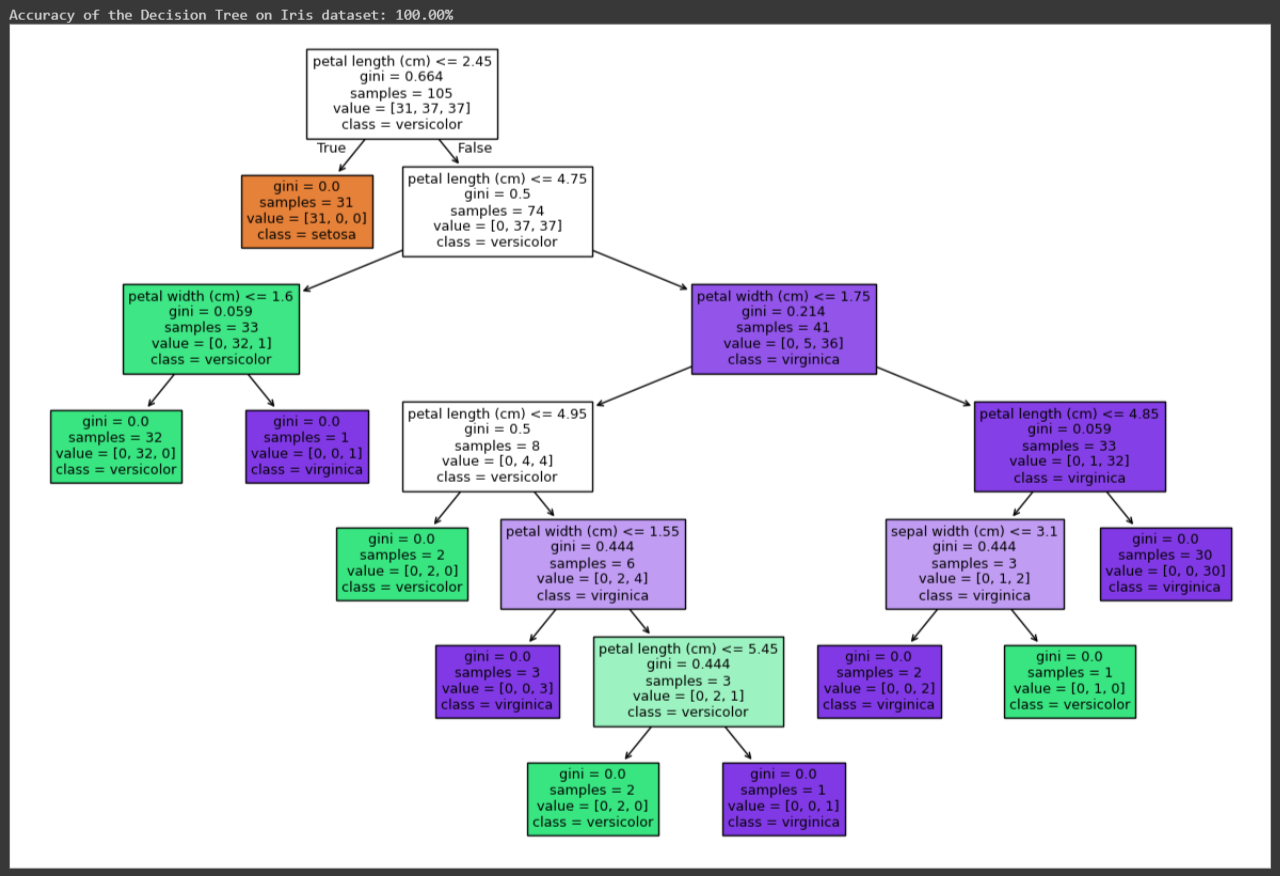
**print(f'Accuracy of the Decision Tree on Iris dataset: {accuracy \* 100:.2f}%')**

**# Plot the decision tree**

**plt.figure(figsize=(15,10))**

**tree.plot\_tree(clf, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True)**

**plt.show()**

****

Part 2:

**# Import necessary libraries**

**import pandas as pd**

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.metrics import accuracy\_score**

**from sklearn.preprocessing import LabelEncoder**

**from sklearn.tree import export\_graphviz**

**import graphviz**

**# Create the dataset as shown in the image**

**data = {**

**'Outlook': ['Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Sunny', 'Overcast', 'Rainy',**

**'Rainy', 'Sunny', 'Rainy', 'Overcast', 'Overcast', 'Sunny'],**

**'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild',**

**'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],**

**'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High',**

**'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'],**

**'Windy': [False, True, False, False, False, True, True, False, False, False, True, True, False, True],**

**'Play Golf': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',**

**'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']**

**}**

**# Convert the dictionary into a DataFrame**

**df = pd.DataFrame(data)**

**# Initialize the LabelEncoder**

**le = LabelEncoder()**

**# Encode categorical features (Outlook, Temperature, Humidity, Windy, Play Golf)**

**df['Outlook'] = le.fit\_transform(df['Outlook'])**

**df['Temperature'] = le.fit\_transform(df['Temperature'])**

**df['Humidity'] = le.fit\_transform(df['Humidity'])**

**df['Windy'] = le.fit\_transform(df['Windy'])**

**df['Play Golf'] = le.fit\_transform(df['Play Golf'])**

**# Separate features and target variable**

**X = df.drop(columns=['Play Golf'])**

**y = df['Play Golf']**

**# Split the dataset into training and testing sets (70% train, 30% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**# Create Decision Tree Classifier with entropy (to achieve better splits)**

**clf = DecisionTreeClassifier(criterion='entropy', random\_state=42)**

**# Train the classifier on the training data**

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

**# Predict the test data**

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

**# Calculate accuracy**

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

**print(f'Accuracy of the Decision Tree on Play Golf dataset: {accuracy \* 100:.2f}%')**

**# Visualize the tree using graphviz**

**dot\_data = export\_graphviz(clf, out\_file=None,**

**feature\_names=X.columns,**

**class\_names=['No', 'Yes'],**

**filled=True, rounded=True,**

**special\_characters=True)**

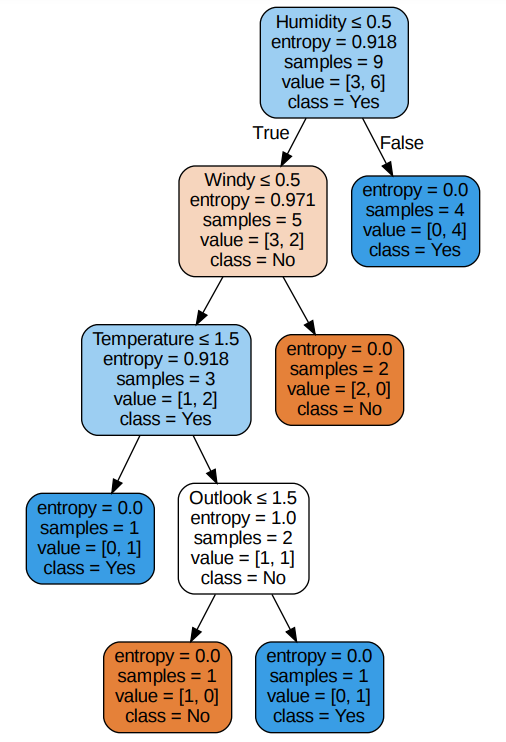
**# Create a Graphviz source object and render it**

**graph = graphviz.Source(dot\_data)**

**graph.render("play\_golf\_decision\_tree") # Saves the tree as a PDF**

**# Show the tree inline (optional, depending on the environment)**

**graph.view()**

****

**Post Lab Question-Answers:**

**1. What are the four commonly used ways to generate decision trees from a given dataset?**

1) Gini Index: Measures the impurity or impurity reduction at each node. It selects the feature that maximizes this reduction.

2) Information Gain (Entropy): Measures the amount of information gained when a feature is selected. A higher information gain indicates a better feature for splitting.

3) Gain Ratio: Modifies the Information Gain to deal with issues of bias towards features with a larger number of distinct values.

4) Chi-Square: Evaluates the statistical significance of the relationship between a feature and the target class, choosing the feature with the most significant relation.

**Outcomes: Apply concepts of different types of Learning and Neural Network**

**Conclusion (based on the Results and outcomes achieved):**

The Decision Tree algorithm effectively classifies the Iris dataset into its respective species using feature selection methods such as Gini Index or Information Gain. Through recursive partitioning, the data is divided into smaller subsets based on the most important features, resulting in a model that can predict the class labels with high accuracy. This process demonstrates the power of decision trees for classification problems, as it achieves clear decision boundaries and interpretable results. The performance of the model can be evaluated by calculating the efficiency (accuracy) of the predictions.

**References:**

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition