Name : Suvidhi V. Pareek

Class / Div : TE / 03

Roll No : 09

| Experiment No. 5 |
| --- |
| Decision Tree Induction |
| Date of Performance: 05-08-2024 |
| Date of Submission: 12-08-2024 |

**Aim:** To implement Decision tree Induction for data classification.

**Objective:** Develop a program to implement a decision tree algorithm for classifying data based on attributes.

**Theory**:-

Decision Tree Induction is a popular machine learning technique used for classification and regression tasks. A decision tree is a flowchart-like structure in which each internal node represents a test on an attribute, each branch corresponds to the outcome of the test, and each leaf node represents a class label (or target value).

The process begins with the entire dataset and splits it into subsets based on an attribute that results in the most significant information gain or the least impurity. Common algorithms for building decision trees include ID3, C4.5, and CART. Decision trees are favored for their interpretability and simplicity, as they can be visualized easily, making it clear how decisions are derived from the input data.

**Key Concepts:**

* **Entropy**: Measures the uncertainty in a dataset. Lower entropy indicates a more homogenous group.
* **Information Gain**: The reduction in entropy after a dataset is split on an attribute. The attribute with the highest information gain is chosen for splitting.
* **Gini Index**: Another metric for measuring impurity in a dataset, often used in CART algorithms.

**Implementation Steps:**

1. **Data Preparation**: Collect and preprocess the dataset (e.g., handling missing values, encoding categorical variables).
2. **Tree Construction**: Use an algorithm to recursively split the dataset based on the chosen attributes until stopping criteria are met (e.g., all instances in a node belong to the same class or a maximum tree depth is reached).
3. **Pruning**: Simplify the tree by removing branches that have little importance, which helps improve generalization to unseen data.
4. **Evaluation**: Assess the performance of the decision tree using metrics such as accuracy, precision, recall, and F1-score.

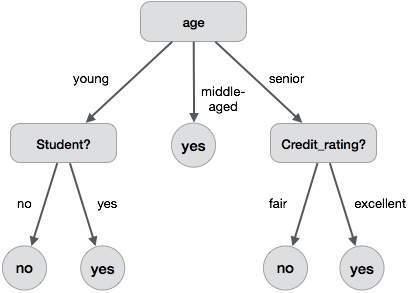


Figure 1: Decision Tree Example

* **Advantages**:
  + **Interpretability**: Decision trees provide clear and interpretable models that can be visualized.
  + **No Assumptions**: They do not assume any specific distribution of the data.
  + **Versatility**: They can handle both numerical and categorical data.
* **Disadvantages**:
  + **Overfitting**: They can create overly complex trees that do not generalize well.
  + **Instability**: Small changes in the data can lead to different splits, affecting the model's consistency.
* **Applications**:
  + Decision trees are widely used in various fields, including finance (credit scoring), healthcare (diagnosis), marketing (customer segmentation), and more due to their simplicity and effectiveness.

**Code and output:**

# Import necessary libraries

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

import matplotlib.pyplot as plt

from sklearn import tree

# Upload the dataset

from google.colab import files

uploaded = files.upload()

# Check if 'dataset.csv' is uploaded

import os

print("Uploaded files:", os.listdir())

# Load the dataset

df = pd.read\_csv('play\_tennis.csv')

# Display the first few rows of the dataframe

df.head()

# Encode categorical features

label\_encoders = {}

for column in df.columns:

if df[column].dtype == 'object':

le = LabelEncoder()

df[column] = le.fit\_transform(df[column])

label\_encoders[column] = le

# Features and target variable

X = df[['outlook', 'temp', 'humidity', 'wind']]

y = df['play']

# Split the data into training and test sets

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

# Create and train the decision tree model

clf = DecisionTreeClassifier()

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

# Make predictions and evaluate the model

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

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

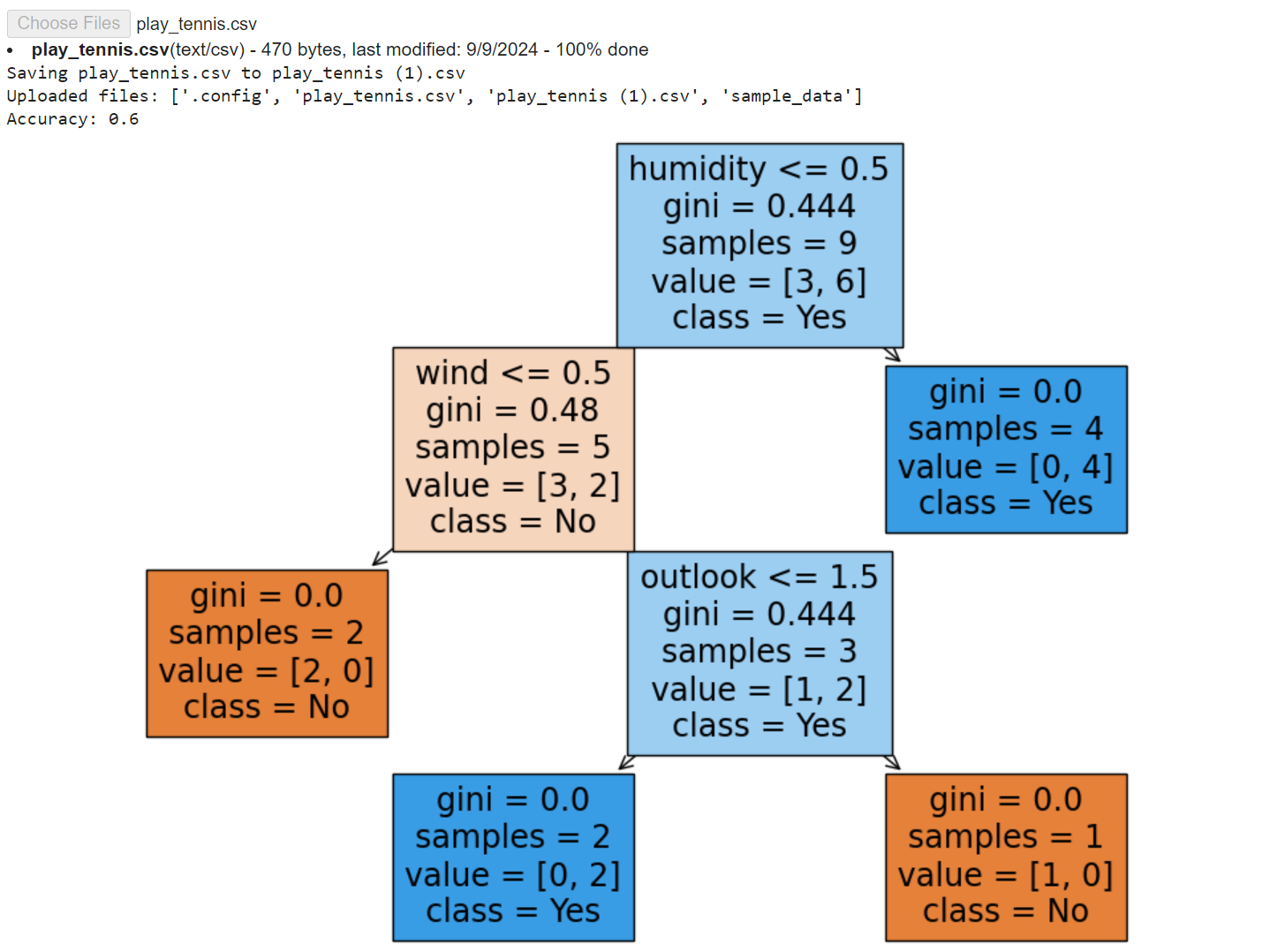
print(f"Accuracy: {accuracy}")

# Plot the decision tree

plt.figure(figsize=(12,8))

tree.plot\_tree(clf,feature\_names=X.columns,class\_names=['No','Yes'],filled=True)

plt.show()



**Conclusion**: Comment on how decision tree helps in classifying data based on attributes. A decision tree classifies data by recursively splitting it based on attribute values, creating a clear path from root to leaf nodes for accurate decision-making.