BAKU HIGHER OIL SCHOOL

INFORMATION TECHNOLOGY DEPARTMENT
INFORMATION SECURITY
DIVISION

Machine Learning

Home Assignment 5

Assignment name: Decision Tree Algorithm

Student name: Huseyn Abdullayev

Group number: CS 25

Instructor: Leyla Muradkhanli

Dataset Description and Preprocessing Steps

The dataset used for this assignment is the PlayTennis Dataset. It contains 14 samples of weather conditions, with information on whether to play tennis or not.

The dataset includes the following key columns:

• Outlook: Weather outlook (Sunny, Overcast, Rain).

• Temperature: Temperature level (Hot, Mild, Cool).

• Humidity: Humidity level (High, Normal).

• Wind: Wind strength (Weak, Strong).

• Play Tennis: Target variable (Yes, No).

The target variable is "Play Tennis," which is categorical (binary).

Data Preprocessing Steps:

Loading Data:

The dataset was imported using pandas.read_csv('PlayTennis.csv').

task.py > ...

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Load the dataset
data = pd.read_csv('PlayTennis.csv')

# Display the first few rows to verify loading (optional)
print(data.head())
```

Data Cleaning:

No missing values were present in the dataset. All features are categorical, so encoding was required.

Encoding Categorical Labels:

All features and the target "Play Tennis" are categorical. LabelEncoder from scikit-learn was used to convert them to numeric values.

```
13  # Prepare features (X) and target (y)
14  X = data.drop('Play Tennis', axis=1)
15  y = data['Play Tennis']
16
17  # Encode categorical features and target
18  label_encoders = {}
19  for col in X.columns:
20      le = LabelEncoder()
21      X[col] = le.fit_transform(X[col])
22      label_encoders[col] = le
23
24  le_y = LabelEncoder()
25  y = le_y.fit_transform(y)
26  label_encoders['Play Tennis'] = le_y
27
```

Feature Selection and Splitting:

Selected features: Outlook, Temperature, Humidity, Wind (as X).

Target: Play Tennis (as y).

The dataset was then split into 70% training data (9 samples) and 30% testing data (5 samples) using train_test_split() with random_state=42 for reproducibility.

```
28  # Split the data into 70% training and 30% testing
29  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
30
```

Summary of Model Implementation

Since Decision Trees can handle categorical data after encoding, no scaling was necessary.

A Decision Tree Classifier model from sklearn.tree was chosen for this binary classification task. Models were trained using the training data (9 samples) with different criteria: gini and entropy.

Decision Trees classify by building a tree structure where each node represents a feature split that best separates classes based on the chosen criterion.

```
31  # Define criteria to use
32  criteria = ['gini', 'entropy']
33
34  # Dictionary to store models and accuracies
35  models = {}
36  accuracies = {}
37
38  # Train Decision Tree models using different criteria
39  for criterion in criteria:
40      # Create Decision Tree model with the specified criterion
41      model = DecisionTreeClassifier(criterion=criterion)
42
43      # Fit the model on training data
44      model.fit(X_train, y_train)
45
46      # Store the model
47      models[criterion] = model
48
49      # Predict on test data
50      y_pred = model.predict(X_test)
51
52      # Calculate accuracy
53      acc = accuracy_score(y_test, y_pred)
54
55      # Store accuracy
56      accuracies[criterion] = acc
```

Predictions were made on the test data, and performance metrics were calculated.

```python
# Report and compare accuracies
print("Model Accuracies:")
for criterion, acc in accuracies.items():
    print(f"{criterion.capitalize()} criterion: {acc}")

# Print evaluation details for each criterion
for criterion in criteria:
    model = models[criterion]
    y_pred = model.predict(X_test)
    print(f"\nClassification Report (for {criterion.capitalize()} criterion):")
    print(classification_report(y_test, y_pred))
    print(f"Confusion Matrix (for {criterion.capitalize()} criterion):")
    print(confusion_matrix(y_test, y_pred))
```

Model Accuracies:

• Gini criterion: 0.6

• Entropy criterion: 0.6

Classification Report (for Gini criterion):

precision recall f1-score support

0 0.50 0.50 0.50 2

1 0.67 0.67 0.67 3

accuracy 0.60 5

macro avg 0.58 0.58 0.58 5

weighted avg 0.60 0.60 0.60 5

Confusion Matrix (for Gini criterion):

[[1 1]

[1 2]]

Classification Report (for Entropy criterion):

precision recall f1-score support

0 0.50 0.50 0.50 2

1 0.67 0.67 0.67 3

accuracy 0.60 5

macro avg 0.58 0.58 0.58 5

weighted avg 0.60 0.60 0.60 5

Confusion Matrix (for Entropy criterion):

[[1 1]

[1 2]]

Interpretation and Conclusion

The models perform moderately, with both gini and entropy criteria achieving 60% accuracy, indicating the small PlayTennis dataset has some separability but limited samples lead to lower performance. There are misclassifications in both classes. Weather features like Outlook likely influence the decision strongly. The Decision Tree classifier achieved moderate accuracy on the PlayTennis dataset. This demonstrates that Decision Trees are suitable for classification tasks with interpretable rules. Compared to SVM, Decision Trees are easier to visualize and understand but can overfit on small data. In real-world applications, Decision Trees' interpretability is an advantage, but they can be prone to variance. Further improvements could include using ensemble methods like Random Forest or pruning the tree. This experiment shows Decision Trees' effectiveness with structured, small datasets.