Decision Tree Classifier Tutorial

This tutorial walks through using the Decision Tree Automation package with examples and explanations of the underlying algorithm.

Table of Contents

Introduction to Decision Trees

Decision trees are supervised learning algorithms used for both classification and regression tasks...

Key Advantages of Decision Trees

- Interpretability: Easy to understand and visualize
- No preprocessing required: Can handle both numerical and categorical data without normalization
- Handles non-linear relationships: Can model complex patterns
- Feature importance: Naturally provides insight into which features are most important

How Decision Trees Work

- Start with all data at the root node
- Find the best feature and threshold to split the data
- Create child nodes based on the split
- Recursively repeat until stopping criteria are met
- Predict the majority class in each leaf node

Quick Start Guide

Installation

```
pip install numpy pandas matplotlib seaborn scikit-learn
```

Basic Usage from Command Line:

```
python decision_tree_automation.py --filepath your_dataset.csv --target target_column_name
```

Basic Usage from Python Code:

```
from decision_tree_automation import run_decision_tree_analysis

tree, accuracy = run_decision_tree_analysis(
    filepath='your_dataset.csv',
    target_column='target_column_name'
)
```

Walkthrough with Example Dataset

Step 1: Prepare your dataset

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
iris_df.to_csv('iris_dataset.csv', index=False)
```

Step 2: Run the analysis

```
from decision_tree_automation import run_decision_tree_analysis

tree, accuracy = run_decision_tree_analysis(
    filepath='iris_dataset.csv',
    target_column='target',
    max_depth=3,
    criterion='gini'
)
```

Step 3: Review the results The program will print accuracy, classification report, and generate visualizations.

Understanding the Output

- Decision Tree Visualization: Shows decision and leaf nodes with split criteria.
- Feature Importance: Highlights most influential features.
- Confusion Matrix: Shows performance metrics such as TP, FP, TN, FN.

Customizing Your Decision Tree

Key Parameters to Adjust:

Maximum Depth (max_depth):

```
run_decision_tree_analysis(filepath='data.csv', max_depth=4)
```

Impurity Criterion (criterion):

```
run_decision_tree_analysis(filepath='data.csv', criterion='entropy')
```

Test Size (test_size):

```
run_decision_tree_analysis(filepath='data.csv', test_size=0.25)
```

Advanced Usage

Direct Access to the Classifier:

```
from decision_tree_automation import DecisionTreeClassifier, load_dataset
from sklearn.model_selection import train_test_split

X, y, feature_names = load_dataset('your_dataset.csv', 'target_column_name')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

tree = DecisionTreeClassifier(max_depth=5, min_samples_split=5, criterion='entropy')
tree.fit(X_train, y_train, feature_names)
predictions = tree.predict(X_test)
tree_structure = tree.tree_structure
```

Custom Visualizations:

```
nodes = tree.tree_structure

for node in nodes:
    if 'value' in node and node['value'] is not None:
        print(f"Leaf node at depth {node['depth']} with path {node['path']}")
        print(f"  Predicts class: {node['value']}")
```

Algorithm Deep Dive

Split Selection, Gini Impurity, Entropy, Information Gain

```
Gini(S) = 1 - Σ(p_j²)
Entropy(S) = -Σ(p_j * log2(p_j))
Gain(S, A) = Impurity(S) - Σ((|S_v|/|S|) * Impurity(S_v))
```

Recursive Partitioning and Prediction Process explained.