HW 1 Car Classification Using Decision Tree

1. Objective:

Build a Decision Tree classifier using the Car dataset and the sklearn library to classify cars into two categories: "good" or "bad." Evaluate the model's performance and experiment with hyperparameter tuning to improve accuracy.

Project Goals:

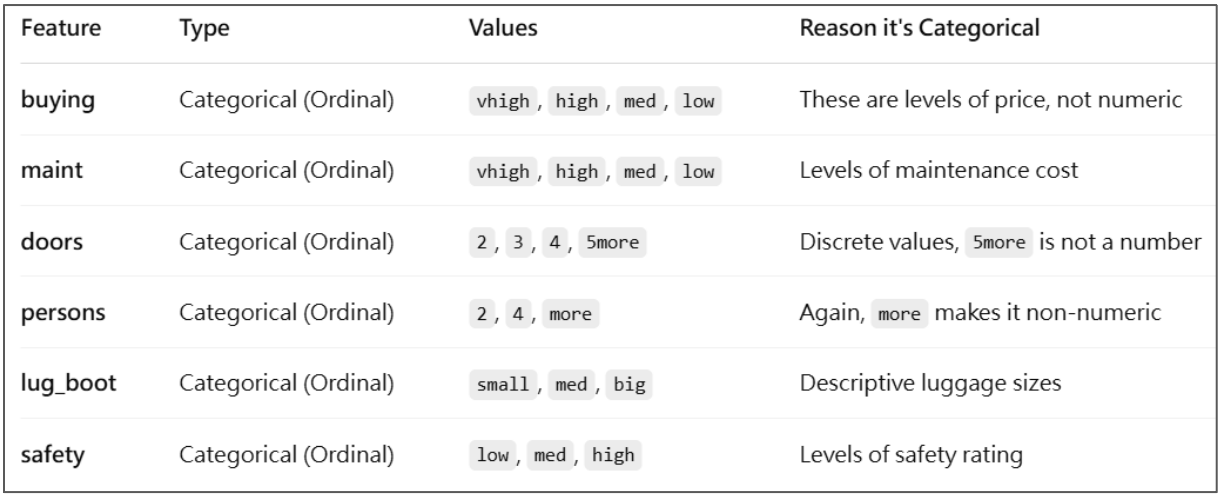
- Gain experience in preprocessing data, including handling categorical, nominal, and continuous variables.
- Learn how to preprocess datasets by cleaning, normalizing, and discretizing data attributes, and reduce dimensionality when necessary.
- Gain experience in constructing decision trees and understanding their internal workings.
- Develop skills in evaluating models built with data mining techniques.
- Familiarize with Python and key machine learning libraries, such as Scikit-learn, Numpy, and Matplotlib.

2. Dataset

You will use the Car Evaluation dataset available in the UCI Machine Learning Repository: Car Evaluation Dataset. (http://archive.ics.uci.edu/ml/datasets/Car+Evaluation)

Data Set Characteristics:	Multivariate	**Number of Instances:**	1728	**Area:**	N/A
Attribute Characteristics:	Categorical	**Number of Attributes:**	6	**Date Donated**	1997-06-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	570379

Attributes of the Car Dataset:



Feature	Type	Values	Reason it's Categorical
buying	Categorical (Ordinal)	`vhigh`, `high`, `med`, `low`	These are levels of price, not numeric
maint	Categorical (Ordinal)	`vhigh`, `high`, `med`, `low`	Levels of maintenance cost
doors	Categorical (Ordinal)	`2`, `3`, `4`, `5more`	Discrete values, `5more` is not a number
persons	Categorical (Ordinal)	`2`, `4`, `more`	Again, `more` makes it non-numeric
lug_boot	Categorical (Ordinal)	`small`, `med`, `big`	Descriptive luggage sizes
safety	Categorical (Ordinal)	`low`, `med`, `high`	Levels of safety rating

3. Steps:

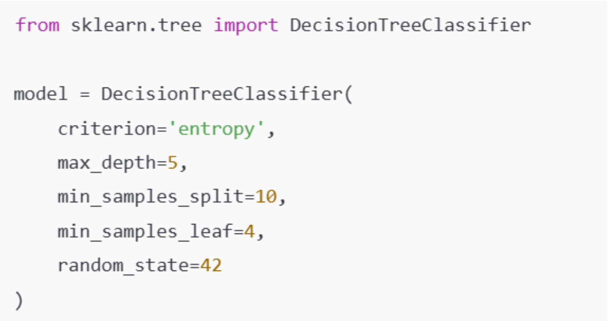
The main steps of build a car classifier using sklearn DecisionTreeClassifier includes:

Step 1: Load and Preprocess the Data

1. **Load the dataset:**
   - Import the dataset and read it into a pandas DataFrame.
   - Explore the dataset and check for missing values.
2. **Data Cleaning and Transformation:**
   - Convert categorical variables (like "buying", "maint", etc.) into numerical format using techniques such as LabelEncoder or OneHotEncoder.
   - Map the class column to binary values: "good" -> 1, "bad" -> 0.
3. **Split the data:**
   - Divide the dataset into features (X) and target (y).
   - Split the data into training and testing sets using train_test_split().

Step 2: Build a Decision Tree Classifier

1. **Initialize the classifier:**
   - Create a DecisionTreeClassifier using the sklearn.tree.DecisionTreeClassifier class.
   - Set the random seed for reproducibility.
   - Configure the hyperparameters of a DecisionTreeClassifier. For example:



```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=5,
    min_samples_split=10,
    min_samples_leaf=4,
    random_state=42
)
```

2. **Train the model:**
    - Fit the model to the training data.
3. **Make predictions:**
    - Use the trained model to make predictions on the test set.

Step 3: Evaluate the Model

1. **Evaluate performance:**
    - Calculate and display the accuracy of the model using accuracy_score.
    - Print a confusion matrix using confusion_matrix and classification_report.
2. **Visualize the decision tree:**
    - Plot the decision tree using plot_tree or export_graphviz.

Step 4: Hyperparameter Tuning

1. **Tune hyperparameters:**
    - Experiment with different values for max_depth, min_samples_split, min_samples_leaf, and criterion.
    - Use GridSearchCV or RandomizedSearchCV for systematic hyperparameter optimization.
2. **Evaluate the tuned model:**
    - After tuning, retrain the model and evaluate its performance again.

Step 5: Report Results

1. **Summarize findings:**
    - Write a brief summary of the model's performance, including any improvements achieved through hyperparameter tuning.
    - Discuss any observations regarding the importance of various features.

4. Submission:

Please zip and submit the following to iLearning:

- A Python script or Jupyter notebook containing the code.
- A summary report of the results in PDF format, including the confusion matrix, accuracy, precision, recall, F1 and any hyperparameter tuning outcomes.
    - Show your algorithm configuration such as the splitting criterion, max_depth, min_samples_split, min_samples_leaf, min_impurity_decrease, and so on

Bonus:

- **Feature importance analysis:**
    - After training the Decision Tree, analyze which features are most important for making the classification decision using feature_importances.