Experiment-5: Decision Trees from Scratch

October 8, 2025

Objective

In this assignment, you will implement a **Decision Tree Classifier from scratch using numpy.** and apply it to the **Adult Income Dataset**. The task is to predict whether a person earns more than $50K per year. You will build the tree, evaluate it, and perform both pre-pruning and post-pruning.

Dataset

We will use the **Adult Income Dataset** from UCI.

- Dataset link: Adult Dataset
- Task: Binary classification ($\leq 50K$ vs. $> 50K$).
- Features: Mix of categorical (e.g., workclass, education, occupation) and numeric (e.g., age, hours-per-week).
- Target: Income.

Loading Instructions

Use the following code to fetch the dataset:

```
pip install ucimlrepo
```

```
from ucimlrepo import fetch_ucirepo
```

```
adult = fetch_ucirepo(id=2)
X = adult.data.features   # features (pandas DataFrame)
y = adult.data.targets    # target (pandas DataFrame)
```

—

Instructions

Follow these steps carefully. Do not skip any part.

1. Data Preparation

- Handle missing values (drop or impute).
- Encode categorical variables into numeric values (e.g., label encoding).
- Split the dataset as:
  - 80% training
  - 20% validation
  - 20% test

  Use the validation set to tune depth and pruning.

2. Build a Decision Tree From Scratch

- Implement a tree recursively.
- At each split:
  1. Compute both **Gini Impurity** and **Entropy**.
  2. For each feature and split, calculate the weighted impurity of child nodes.
  3. Choose the split with the highest information gain (lowest impurity).
- Continue splitting until:

  - All samples in a node have the same label, OR
  - Maximum depth is reached, OR
  - No further improvement in impurity.

- Implement a function to predict for new samples.

3. Pre-Pruning (Restricting Tree Growth)

While building the tree:

- Limit maximum depth (**try depths = 2, 4, 6, and unlimited**).
- Require at least a minimum number of samples (e.g., 5) to split.
- Require a minimum impurity decrease (optional).

4. Post-Pruning (Reduced Error Pruning)

1. First grow a full tree.
2. Then, for each internal node:
   - Replace it with a leaf (majority class).
   - Check validation accuracy.
3. If accuracy does not decrease, keep the pruning.
4. Repeat until no further improvement.

5. Evaluation

- Train using the training set.
- Tune depth and pruning using validation set.
- Report final results on test set.
- Metrics to report:
  - Accuracy
  - Precision, Recall, F1-score

  - Confusion Matrix

- Compare your implementation with `sklearn.tree.DecisionTreeClassifier`.

6. Experiments to Perform

- Compare **Gini vs. Entropy**.
- Compare different depths (2, 4, 6, unlimited).
- Show effect of pruning (pre-pruned vs. post-pruned vs. full tree).
- Identify the most important features (which features are used at the top of the tree).

—