INDIAN INSTITUTE OF TECHNOLOGY MADRAS ZANZIBAR

School of Science and Engineering

Z5007: Programming and Data Structures
M.Tech Data Science & Artificial Intelligence

PROJECT PROGRESS REPORT (MILESTONE 2)

Decision Tree Classifier Implemented from Scratch Using Custom Data Structures

Submitted by:

Student: Akula Balakrishna (ZDA25M012)
Email: zda25m012@iitmz.ac.in

Student: Gaurav jha (ZDA25M005)
Email: zda25m005@iitmz.ac.in

Instructor: Innocent Nyalala

Submission Date: December 23, 2025

Contents

1 Progress Summary

This project implements a **Decision Tree Classifier from scratch** using fundamental data structures in Python, without relying on external machine learning libraries such as `scikit-learn` for training.

The objective is to predict whether an individual earns more than $50,000 per year using the **Adult Income Dataset**. The dataset contains both numerical and categorical attributes, making it suitable for evaluating split criteria and tree construction logic.

As of Milestone 2, all core components of the decision tree have been implemented, including custom data structures, impurity computation, and recursive tree construction with pre-pruning. The project is approximately **65–70% complete**.

Component Status

Component	Status
TreeNode Data Structure	Complete
Custom Hash Table	Complete
Custom Queue (BFS)	Complete
Dataset Loading & Cleaning	Complete
Gini Impurity Computation	Complete
Best Split Selection	Complete
Recursive Tree Construction	Complete
Prediction (Tree Traversal)	Complete
Evaluation & Benchmarking	Partial

2 System Architecture

The system follows a modular pipeline:

1. Dataset loading and preprocessing
2. Impurity-based split evaluation
3. Recursive tree construction with pre-pruning
4. Prediction via tree traversal
5. Evaluation and diagnostics

3 Technical Details

3.1 Custom Data Structures

3.1.1 TreeNode

Each node stores:

- Feature index used for splitting

- Threshold or categorical split set
- Left and right child pointers
- Class frequency counts
- Node depth and sample count

A node is a leaf if no further splitting improves impurity or stopping conditions are met.

3.1.2 Custom Hash Table

A custom hash table is used to store class-frequency counts during impurity calculation. All operations run in expected amortized $O(1)$ time.

3.1.3 Queue

A custom queue supports breadth-first traversal and will be used for post-pruning in later stages.

3.2 Decision Tree Split Criterion

The primary split criterion used is **Gini Impurity**. For a node containing K classes with class probabilities $p_1, p_2, \ldots, p_K$, Gini impurity is:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{K} p_i^2$$

For a binary classification problem ($K = 2$):

$$\text{Gini}(D) = 1 - (p_1^2 + p_2^2)$$

3.3 Gini Impurity Reduction

For a candidate split that divides dataset D into left and right subsets D_L and D_R, the weighted impurity after the split is:

$$\text{Gini}_{\text{split}} = \frac{|D_L|}{|D|}\text{Gini}(D_L) + \frac{|D_R|}{|D|}\text{Gini}(D_R)$$

The impurity reduction (also called Gini gain) is:

$$\Delta\text{Gini} = \text{Gini}(D) - \text{Gini}_{\text{split}}$$

The split with the **maximum impurity reduction** is selected.

3.4 Handling Feature Types

Numerical Features: Binary splits of the form:

$$x_j \leq t \quad \text{vs.} \quad x_j > t$$

where t is a candidate threshold.

Categorical Features: Binary splits of the form:

$$x_j \in S \quad \text{vs.} \quad x_j \notin S$$

where S is a subset of categorical values.

3.5 Pre-pruning Conditions

Tree growth is stopped when any of the following conditions is met:

- Maximum depth is reached
- Number of samples at a node is below a threshold
- Impurity reduction ΔGini is below a minimum value

3.6 Prediction Rule

For a test sample x, prediction is performed by traversing the tree from the root to a leaf. At a leaf node, the predicted class $\hat{y}$ is:

$$\hat{y} = \arg\max_c \ \text{count}(c)$$

where $\text{count}(c)$ is the frequency of class c at that leaf.

3.7 Time Complexity Analysis

Let:

- N = number of samples
- M = number of features

Tree Construction:

$$O(N \cdot M \cdot \log N)$$

Prediction (per sample):

$$O(\text{tree depth})$$

Hash table operations run in expected $O(1)$ time.

4 Challenges and Solutions

High Computational Cost: Mitigated by limiting candidate thresholds and using efficient frequency counting.

Overfitting: Controlled using pre-pruning constraints.

Categorical Explosion: Handled using binary splits and grouping of rare categories.

5 Current Results

The decision tree trains successfully on the Adult Income dataset and produces meaningful predictions. Detailed benchmarking against `scikit-learn` will be included in the final milestone.

6 Remaining Work

- Reduced-error post-pruning
- Benchmarking against scikit-learn
- Additional evaluation metrics
- Final report and presentation