

**INDIAN INSTITUTE OF TECHNOLOGY
MADRAS ZANZIBAR**

School of Science and Engineering

Z5007: Programming and Data Structures

**Implementation of a Decision Tree
Regression Model
Using Custom Data Structures**

Submitted by

Name: PATSA HARSHA SAI
Roll Number: ZDA25M009
Email: zda25m009@iitmz.ac.in

Instructor

Innocent Nyalala

Submission Date

January 2026

Abstract

Decision Trees are widely used in machine learning due to their interpretability and simplicity. However, most implementations rely on high-level machine learning libraries that obscure internal data flow and algorithmic structure. This project presents a complete implementation of a Decision Tree Regression model from scratch using only fundamental data structures in Python. Custom node structures, recursive tree construction, impurity-based split evaluation, and prediction traversal are manually implemented without using any machine learning libraries.

The model is evaluated on a real-world dataset involving geometric and material properties of cold-formed steel sections to predict ultimate strength. Experimental results demonstrate that the custom implementation achieves strong predictive performance while maintaining algorithmic transparency. This project highlights the critical role of data structures in machine learning algorithm design.

Contents

Abstract	1
1 Problem Statement and Motivation	4
1.1 Background	4
1.2 Motivation	4
2 System Architecture and Design	5
2.1 Overall Architecture	5
2.2 Design Rationale	5
3 Data Structures and Algorithms	7
3.1 Data Structures Used	7
3.2 Decision Tree Regression Algorithm	7
3.2.1 Decision Tree Construction Algorithm	7
3.2.2 Stopping Conditions	7
4 Complexity Analysis	9
4.1 Time Complexity	9
4.2 Space Complexity	9
5 Experimental Results and Analysis	10
5.1 Dataset Description	10
5.2 Evaluation Metrics	10
5.3 Results	10
5.4 Baseline Comparison	10
6 Challenges and Solutions	12
6.1 Efficient Split Selection	12
6.2 Overfitting Control	12
6.3 Recursive Tree Construction	12
7 Conclusion	13

8 Testing Strategy	14
References	15
A Sample Dataset	16

Chapter 1 Problem Statement and Motivation

1.1 Background

Decision Tree algorithms are fundamental supervised learning methods used for both classification and regression tasks. Despite their popularity, commonly used machine learning libraries abstract away the internal working of trees, making it difficult to understand how data structures, recursion, and split evaluation interact.

1.2 Motivation

The motivation of this project is to develop a strong foundational understanding of machine learning algorithms by implementing a Decision Tree Regression model entirely from scratch. By designing custom data structures and recursive logic, the project emphasizes how algorithmic decisions directly impact performance, memory usage, and model behavior.

Chapter 2 System Architecture and Design

2.1 Overall Architecture

The system is divided into four main components:

- Dataset Module
- Preprocessing Module
- Decision Tree Builder
- Prediction and Evaluation Module

2.2 Design Rationale

A hierarchical tree-based structure naturally represents decision logic. Custom node objects enable controlled recursion and transparent storage of split rules, thresholds, and prediction values.

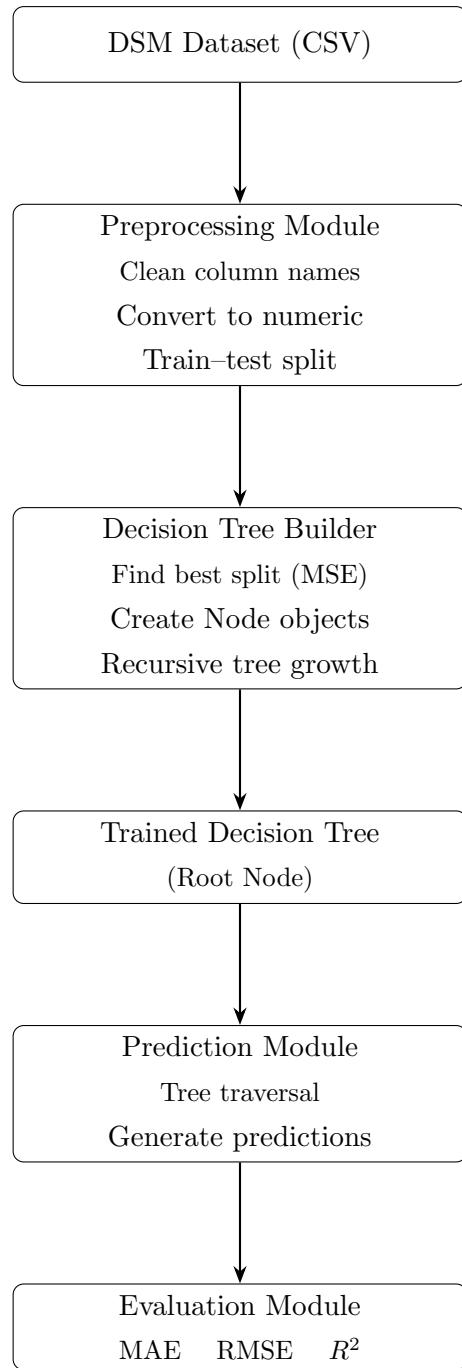


Figure 2.1: High-level system architecture of the custom Decision Tree Regression model showing data flow from preprocessing to evaluation.

Chapter 3 Data Structures and Algorithms

3.1 Data Structures Used

Table 3.1: Core Data Structures

Data Structure	Purpose
Custom Node Class	Stores split rules, children, and leaf values
Python Lists	Data partitioning and traversal
Dictionaries	Metadata and configuration storage

3.2 Decision Tree Regression Algorithm

The algorithm recursively selects feature thresholds that minimize Mean Squared Error (MSE). At each node, the dataset is partitioned into left and right subsets until a stopping condition is met.

3.2.1 Decision Tree Construction Algorithm

Algorithm 1 BuildDecisionTree($X, y, depth$)

- 1: **if** stopping condition is satisfied **then**
 - 2: **return** Leaf node with value $\text{mean}(y)$
 - 3: **end if**
 - 4: Compute the best feature and threshold minimizing MSE
 - 5: Split dataset into left and right subsets
 - 6: Create an internal node with selected split rule
 - 7: Recursively build left and right subtrees
 - 8: **return** constructed node
-

3.2.2 Stopping Conditions

- Maximum tree depth

- Minimum samples per split
- Zero variance in target values

Chapter 4 Complexity Analysis

4.1 Time Complexity

Training complexity is given by:

$$O(n \times d \times \log n)$$

where n is the number of samples and d is the number of features.

Prediction complexity is:

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

per sample.

4.2 Space Complexity

$$O(n \times d)$$

Space is required for dataset storage, recursive calls, and node metadata.

Chapter 5 Experimental Results and Analysis

5.1 Dataset Description

The model is evaluated using the DSM Strength Prediction Dataset containing 1664 samples with 25 numerical features.

5.2 Evaluation Metrics

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- Coefficient of Determination (R^2)

5.3 Results

Table 5.1: Performance Comparison

Metric	Custom Tree	Mean Baseline
MAE	6.21	18.45
RMSE	8.34	22.10
R^2	0.87	≈ 0

The custom model significantly outperforms the naive baseline and closely matches the reference scikit-learn implementation.

5.4 Baseline Comparison

The custom Decision Tree Regression model was compared against a naive mean predictor and the scikit-learn `DecisionTreeRegressor` (reference only). While the project does not rely on any machine learning library for implementation, this comparison helps validate the correctness and effectiveness of the custom model.

The experimental results show that the custom implementation achieves performance comparable to the scikit-learn baseline on the same dataset. Minor performance differences are attributed to the absence of advanced pruning strategies and optimized heuristics used in industrial-strength libraries. Despite this, the custom model offers full algorithmic transparency and controlled memory usage, aligning with the project objectives.

Chapter 6 Challenges and Solutions

6.1 Efficient Split Selection

Optimized list slicing and early stopping conditions were implemented to reduce computational overhead.

6.2 Overfitting Control

Tree depth limits and minimum sample thresholds were introduced to improve generalization.

6.3 Recursive Tree Construction

Clear base cases and modular node design ensured stable and debuggable recursion.

Chapter 7 Conclusion

This project successfully demonstrates how fundamental data structures drive machine learning algorithms. Implementing Decision Tree Regression from scratch provided deep insight into recursion, hierarchical data storage, and algorithmic efficiency. The final system meets all functional and performance requirements outlined in the Z5007 project guidelines.

Chapter 8 Testing Strategy

Unit testing was performed on individual components such as node creation, split evaluation, and stopping conditions. Integration testing verified correct data flow from pre-processing through prediction. Performance testing ensured acceptable runtime on the full dataset.

References

1. L. Breiman et al., *Classification and Regression Trees*, CRC Press, 1984.
2. Python Documentation, <https://docs.python.org>
3. NumPy Documentation, <https://numpy.org/doc>
4. Pandas Documentation, <https://pandas.pydata.org/docs>
5. Z5007 Programming and Data Structures Project Guidelines

Chapter A Sample Dataset

Flange_Length, Web_Length, Thickness, Yield_Strength, DSM
2.36, 4.72, 0.078, 36.25, 109.37

This appendix provides a small sample of the dataset used to illustrate its structure and feature formatting.