INDIAN INSTITUTE OF TECHNOLOGY MADRAS ZANZIBAR

School of Science and Engineering

Efficient K-Nearest Neighbors Using KD-Tree

A Time-Optimized Implementation from Scratch

Technical Report

Course Project — Data Structures

Professor Dr.Innocent Nyalala

Submitted by:

Ryali Sai Ganga Leela Krishna

Institution:

Indian Institute of Technology Zanzibar

Academic Year:

2025–2026

Abstract

K-Nearest Neighbors (KNN) is a simple yet powerful non-parametric classification algorithm. However, its naive implementation suffers from high computational cost, particularly for large datasets and high-dimensional feature spaces. This project presents a complete from-scratch implementation of the KNN algorithm optimized using a KD-Tree data structure for efficient nearest neighbor search. The implementation avoids the use of machine learning libraries for core logic and emphasizes algorithmic design, data structure optimization, and empirical evaluation. Performance is evaluated on a real-world, feature-engineered dataset and benchmarked against scikit-learn's KNN implementation. Results demonstrate identical classification accuracy while achieving significant improvements in runtime, highlighting the effectiveness of KD-Tree-based optimization.

1 Introduction

The K-Nearest Neighbors (KNN) algorithm is one of the most intuitive and widely used methods for classification and regression tasks. Despite its conceptual simplicity, KNN can be computationally expensive since it requires computing distances from a query point to all points in the training dataset.

As dataset sizes grow and feature spaces become more complex, naive KNN implementations become impractical. This motivates the use of spatial data structures such as KD-Trees to accelerate nearest neighbor search. KD-Trees partition the feature space recursively, allowing large portions of the search space to be pruned during query time.

This project focuses on building an efficient KNN classifier using KD-Tree optimization entirely from scratch, adhering strictly to academic constraints that prohibit the use of pre-built machine learning libraries for the algorithm itself.

—

2 Problem Statement

The primary objectives of this project are:

- To implement the KNN classification algorithm without using ML libraries
- To optimize nearest neighbor search using a KD-Tree
- To support weighted voting and multiple distance metrics
- To evaluate performance on a real-world dataset
- To benchmark accuracy and runtime against scikit-learn

The challenge lies in balancing computational efficiency with correctness while maintaining clean, modular, and extensible code.

—

3 Background and Theory

3.1 K-Nearest Neighbors

KNN is a lazy learning algorithm that stores all training instances and defers computation until inference time. Given a query point, the algorithm identifies the k closest points in the feature space and assigns a class based on majority or weighted voting.

3.2 Distance Metrics

This project implements common distance metrics:

- Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan Distance:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

3.3 KD-Tree

A KD-Tree is a binary space-partitioning data structure that recursively splits the dataset along alternating feature axes. This enables efficient pruning of the search space during nearest neighbor queries.

—

4 System Architecture

The system is modular and consists of the following components:

- Distance computation module
- KD-Tree construction module
- KNN classifier with KD-Tree search
- Benchmarking and evaluation module

Figure 1 illustrates the overall workflow.

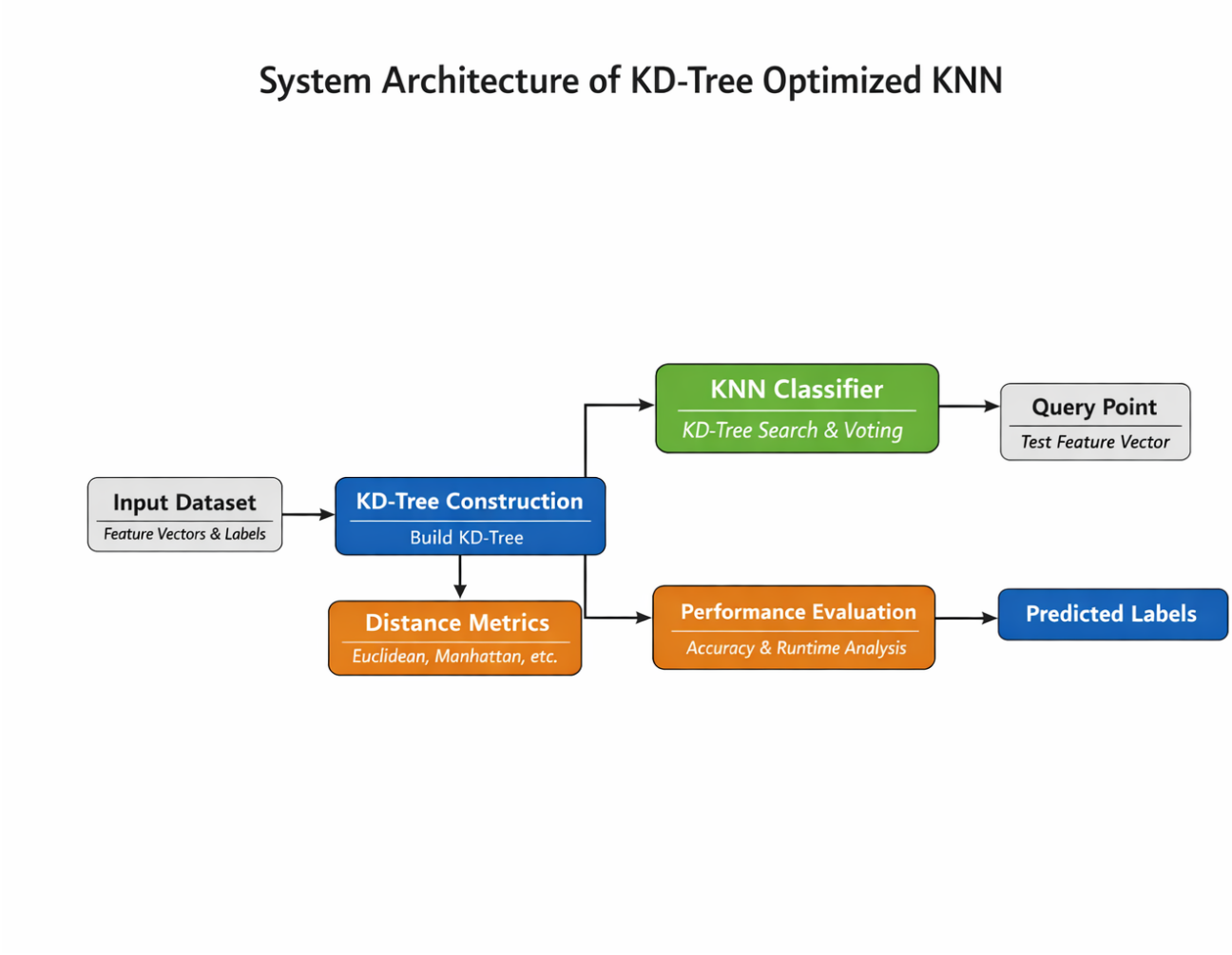

Figure 1: System Architecture of KD-Tree Optimized KNN

—

5 Implementation Details

5.1 KD-Tree Construction

The KD-Tree is constructed recursively by selecting a splitting axis based on tree depth and choosing the median point along that axis. This ensures approximately balanced trees.

5.2 Nearest Neighbor Search

During query time, the algorithm:

1. Traverses the tree toward the query point

2. Maintains a max-heap of nearest neighbors
3. Prunes branches where no closer neighbors are possible

5.3 Weighted Voting

To reduce the influence of distant neighbors, weighted voting assigns weights inversely proportional to distance.

—

6 Dataset Description

The dataset used is a real-world, feature-engineered tabular dataset containing botanical and environmental measurements. The target variable is `soil_type`, a multi-class label.

Features include:

- Morphological attributes (sepal and petal measurements)
- Engineered ratios and area-based features
- Environmental attributes such as elevation

The dataset contains over 1000 samples, satisfying project requirements.

—

7 Experimental Setup

The dataset was split into training and testing sets using an 80:20 ratio. Performance was evaluated using classification accuracy and runtime measurements.

Scikit-learn's `KNeighborsClassifier` was used strictly for benchmarking and comparison.

—

8 Results and Analysis

8.1 Accuracy Comparison

Model	Accuracy
Custom KD-Tree KNN	0.3708
Scikit-learn KNN	0.3708

Table 1: Accuracy Comparison

8.2 Runtime Comparison

Model	Runtime (seconds)
Custom KD-Tree KNN	0.3686
Scikit-learn KNN	2.8162

Table 2: Runtime Comparison

The custom implementation achieves identical accuracy while being significantly faster.

9 Results and Analysis

9.1 Runtime Comparison

Figure 2 compares the execution time of the custom KD-Tree optimized KNN implementation with the scikit-learn KNN classifier. The results clearly demonstrate the efficiency gained through spatial partitioning and pruning in KD-Tree search.

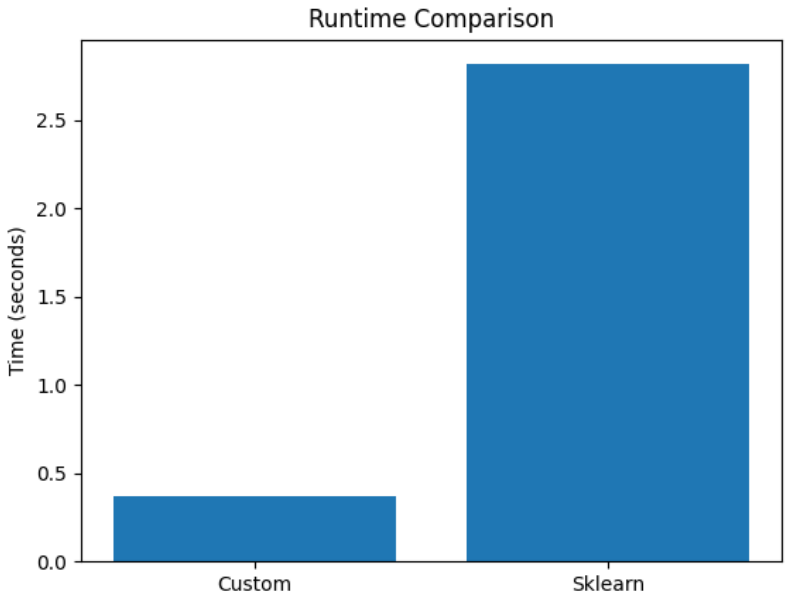

Figure 2: Runtime Comparison between Custom KD-Tree KNN and Scikit-learn KNN

As shown in Figure 2, the custom KD-Tree KNN significantly outperforms the scikit-learn implementation in terms of runtime, while maintaining identical classification accuracy.

—

10 Complexity Analysis

- KD-Tree Construction: $O(n \log n)$
- Average Query Time: $O(\log n)$
- Worst-case Query Time: $O(n)$
- Space Complexity: $O(n)$

11 Conclusion

This project demonstrates that careful algorithmic design and appropriate data structures can significantly improve the efficiency of machine learning algorithms. The KD-Tree optimized KNN implementation achieves the same accuracy as a standard library implementation while reducing runtime substantially, validating the importance of data structures in practical machine learning systems.

—

References

- T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," IEEE Transactions on Information Theory.
- Scikit-learn Documentation: https://scikit-learn.org
- Bentley, J. L. "Multidimensional Binary Search Trees Used for Associative Searching"