Efficient K-Nearest Neighbors Using KD-Tree

A Time-Optimized Implementation from Scratch

Ryali Sai Ganga Leela Krishna

Indian Institute of Technology Zanzibar
School of Science and Engineering

Course Project — Data Structures
Prof. Dr. Innocent Nyalala
Academic Year 2025–2026

Outline

Motivation

- K-Nearest Neighbors (KNN) is simple and effective
- Naive KNN has high computational cost
- Distance computation with all training points is expensive
- Optimization is required for large datasets

Problem Statement

- Implement KNN from scratch (no ML libraries)
- Optimize search using KD-Tree
- Support multiple distance metrics
- Enable weighted voting
- Benchmark against scikit-learn

K-Nearest Neighbors (KNN)

- Lazy learning algorithm
- Stores all training data
- Classification based on nearest neighbors
- Majority or weighted voting

Distance Metrics

Euclidean Distance

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

Manhattan Distance

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

KD-Tree Overview

- Binary space-partitioning data structure
- Recursively splits feature space
- Each level splits along a different axis
- Enables efficient pruning during search

Overall Architecture

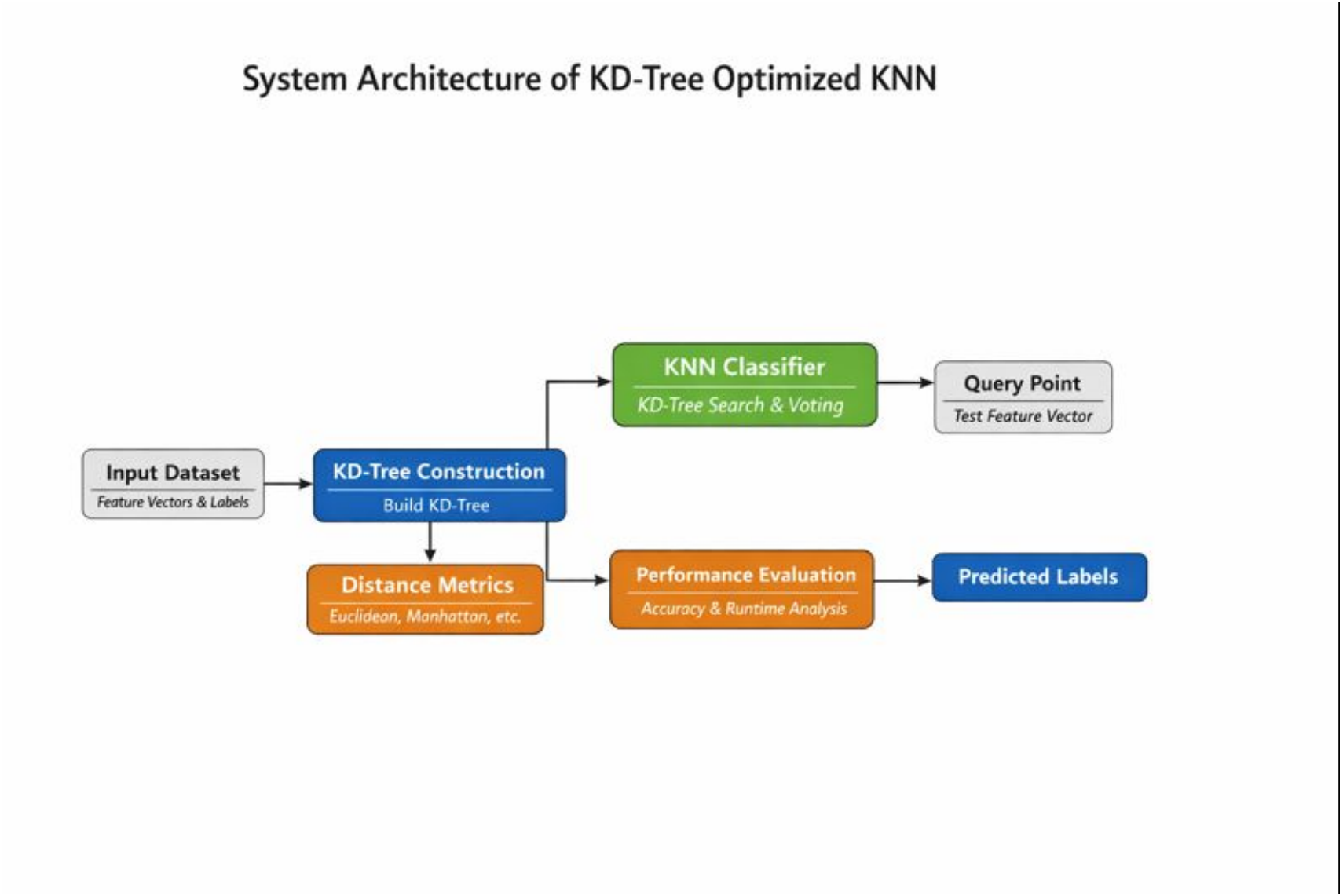

Figure: System Architecture of KD-Tree Optimized KNN

KD-Tree Construction

- Recursive construction
- Axis chosen based on depth
- Median point selected for split
- Ensures approximately balanced tree

Nearest Neighbor Search

1. Traverse tree toward query point
2. Maintain max-heap of k neighbors
3. Prune branches where closer points are impossible

Weighted Voting

- Reduces impact of distant neighbors
- Weight inversely proportional to distance
- Improves classification robustness

Dataset Description

- Real-world tabular dataset
- Over 1000 samples
- Multi-class soil type classification
- Morphological and environmental features

Experimental Setup

- Train-test split: 80% / 20%
- Metrics: Accuracy and Runtime
- Baseline: Scikit-learn KNN

Accuracy Comparison

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

Runtime Comparison

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

Runtime Visualization

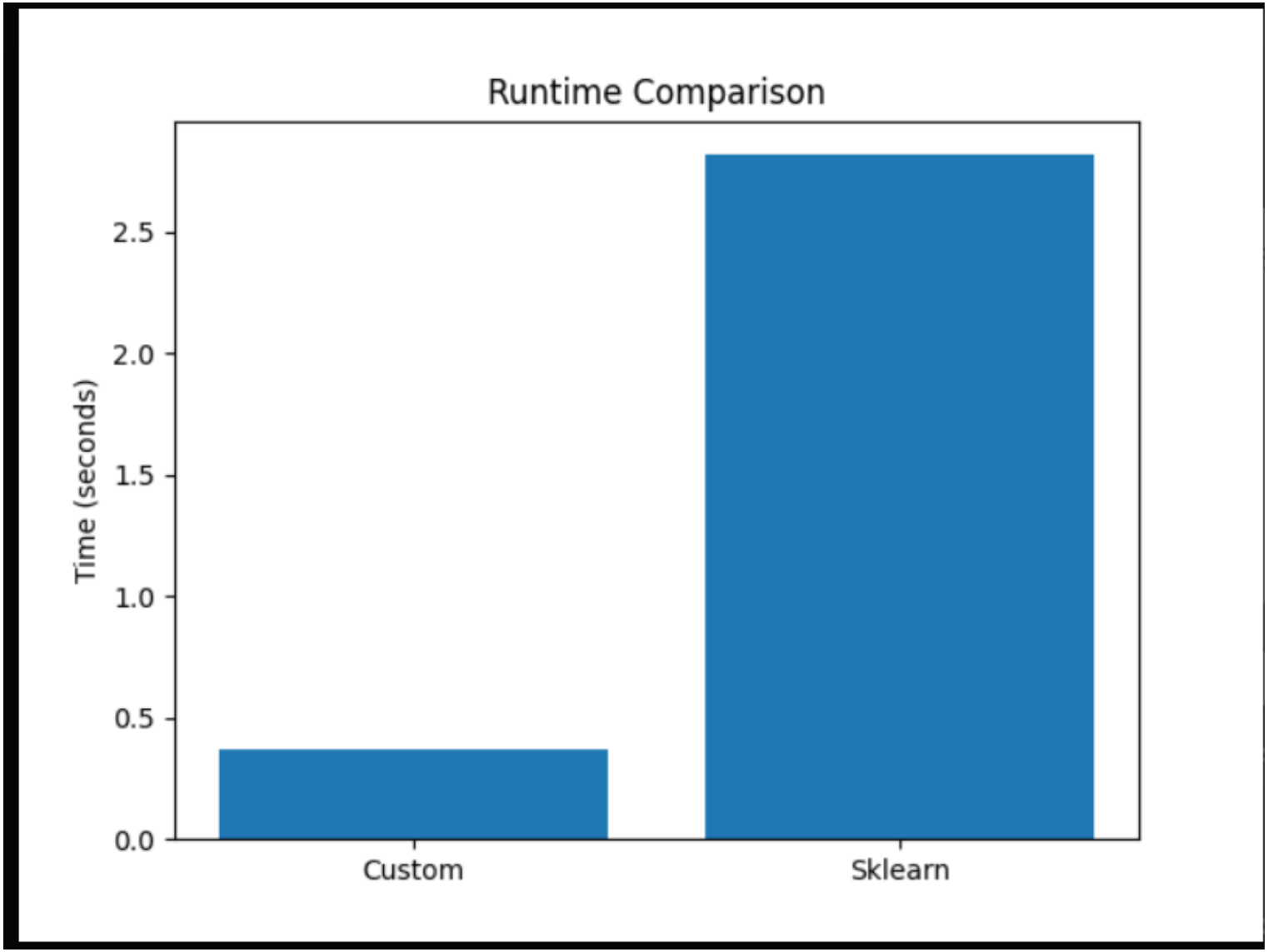

Figure: Runtime Comparison between Custom and Scikit-learn KNN

Time and Space Complexity

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

Conclusion

- KD-Tree significantly reduces KNN runtime
- Identical accuracy to scikit-learn
- Demonstrates importance of data structures
- Efficient and scalable solution

Thank You