International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Multi-Class KNN in Ball-Trees using Adaptive Pruning

Dayaker P[1], Dr. Harsh Lohiya[2]

Submitted: 25/11/2023 **Revised**: 05/01/2024 **Accepted**: 15/01/2024

Abstract: The K-Nearest Neighbors (KNN) algorithm stands as a prominent tool for classification tasks, leveraging proximity to neighboring data points to assign labels. However, in multi-class scenarios, traditional KNN encounters challenges related to expansive search spaces within Ball Trees, suboptimal k-value determinations, and imbalanced class distributions. To overcome these hurdles, an adaptive pruning algorithm adapted for Ball Trees is introduced, aiming to dynamically modify the tree structure while retaining classification accuracy. Results reveal notable advancements in the efficiency and accuracy of the multi-class KNN algorithm empowered by adaptive ball tree pruning. The proposed method effectively reduces search space while maintaining or even enhancing classification accuracy across diverse datasets. Comparative analyses demonstrate the superiority of the proposed approach in handling multi-class complexities and dynamic data distributions. Datasets showcasing high dimensionality, imbalanced class distributions and dynamic data shifts are employed to assess the algorithm's adaptability and performance. The conclusion propose that adaptive ball tree pruning serves as a pivotal mechanism to mitigate the limitations of traditional KNN in multi-class scenarios, offering a promising avenue for refining nearest neighbor classifiers in real-world applications.

Keywords: *KNN, Adaptive Ball Tree Pruning, Classification, Ball Trees, Adaptive Pruning.*

1. Introduction

Absolutely, classification algorithms often fall into the categories of supervised and unsupervised methods. Among supervised classification techniques, the task of pattern categorization into predefined classes is prevalent. Several established techniques have proved effective in text classification. These include decision tree classifiers, rule-based classifiers, maximum margin classifiers like Support Vector Machines (SVM) [2], and probabilistic techniques like Naive Bayes [1]. These algorithms typically require constructing classifier models before making predictions.

When compared to this, the K-Nearest Neighbours (KNN) method discussed in [3] performs in a different way. One example of an instance-based method for learning is Knowledge Networking (KNN), which has not previously generated a classifier model. However, the KNN method uses a quite straightforward technique: it determines the k training dataset items that are nearest to a new pattern when given a new pattern. Applying an appropriate comparison or distance metric achieve this goal. The new pattern will be given the class label that occurs most frequently or highest among these nearby neighbours.

KNN's unique characteristic of making predictions based on local similarity without explicit model building distinguishes it from many traditional classification algorithms. This 'lazy learning' approach allows KNN to be computationally lightweight during the learning phase, as it defers most computation until a new query is presented.

In the realm of classification algorithms, the K-Nearest Neighbors (KNN) method stands as a versatile and intuitive approach, leveraging proximity to neighboring data points to assign class labels. However, when confronted with multi-class scenarios, the efficacy of KNN encounters significant challenges that impede its optimal performance [4].

1.1. Challenges in Multi-Class Scenarios

The effectiveness of KNN is notably affected when dealing with multi-class classification tasks. One of the primary hurdles arises from the expansive search space inherent within Ball Trees, the data structures often employed by KNN. This vast search space complicates the process of identifying the nearest neighbors, impacting the algorithm's ability to discern the most relevant instances for classification.

Moreover, determining the optimal value for 'k,' the number of nearest neighbors considered during classification, becomes a non-trivial endeavor in multi-class settings. The selection of an inappropriate 'k' value often leads to suboptimal classification outcomes, affecting the accuracy and reliability of the predictions made by the algorithm [5].

Furthermore, imbalances among class distributions within the dataset pose a significant challenge. KNN struggles to

*[1] Research Scholar, Department of CSE, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal, M.P, INDIA
poreddydayakar3@gmail.com
[2] Associate Professor, Department of CSE, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal, M.P, INDIA
lohiya27harsh@gmail.com
* Corresponding Author Email: poreddydayakar3@gmail.com*

maintain accuracy across all classes when there's disproportionality in the number of instances belonging to different classes. This imbalance adversely affects the algorithm's capability to make precise predictions, especially for minority classes.

1.2. Addressing Limitations for Enhanced Performance

Overcoming these challenges in multi-class scenarios is pivotal to harnessing the full potential of the KNN algorithm. Innovative techniques, such as adaptive pruning strategies adapted for Ball Trees, offer promising avenues to refine the algorithm's performance. Adaptive approaches aim to intelligently reduce the search space, optimize 'k' values dynamically, and alleviate the impact of imbalanced class distributions.

By delving into adaptive methods within the KNN framework, this study aims to explore and evaluate strategies that mitigate the limitations posed by an overlarge search space, suboptimal 'k' values, and imbalanced class distributions [6]. The objective is to enhance the accuracy and robustness of KNN in multi-class scenarios, contributing to its applicability across diverse and complex datasets.

1.3. Challenges Encountered in Multi-Class KNN

The efficacy of KNN in multi-class classification settings faces hurdles rooted in the architecture of Ball Trees. The inherent nature of Ball Trees contributes to an extensive search space, which complicates the identification of the nearest neighbors, hampering the algorithm's precision in discerning relevant instances for classification. Determining the ideal 'k' value remains a challenging task, where selecting an unsuitable 'k' value leads to compromised classification accuracy [7].

Furthermore, the presence of imbalanced class distributions exacerbates the algorithm's limitations. KNN struggles to provide accurate predictions across all classes when certain categories are underrepresented, impacting the fairness and reliability of the classification outcomes.

1.4. Adaptive Ball Tree Pruning

This study introduces a novel approach aimed at addressing the limitations of multi-class KNN by integrating adaptive ball tree pruning techniques. The core objective is to mitigate these challenges through the implementation of an adaptive pruning algorithm specifically tailored for Ball Trees within the KNN framework.

The adaptive pruning technique seeks to dynamically optimize the structure of the Ball Tree. By intelligently reducing the search space, this approach aims to enhance the algorithm's efficiency while preserving classification accuracy. The algorithm dynamically adjusts the Ball Tree's configuration, strategically pruning irrelevant nodes or branches, thereby streamlining the search process to focus on the most pertinent data points.

The ability of KNN in multi-class classification settings encounters impediments arising from the nature of Ball Trees. The extensive search space within these structures complicates the identification of nearest neighbors, influencing the algorithm's ability to accurately classify instances. Selecting an appropriate 'k' value becomes intricate, where misjudgments lead to compromised classification accuracy [8].

The significance of this research lies in its potential to offer insights into innovative strategies for enhancing KNN's effectiveness in multi-class scenarios. The findings aim to shed light on the feasibility and impact of employing adaptive ball tree pruning as a solution to mitigate challenges associated with expansive search spaces, suboptimal 'k' values, and imbalanced class distributions.

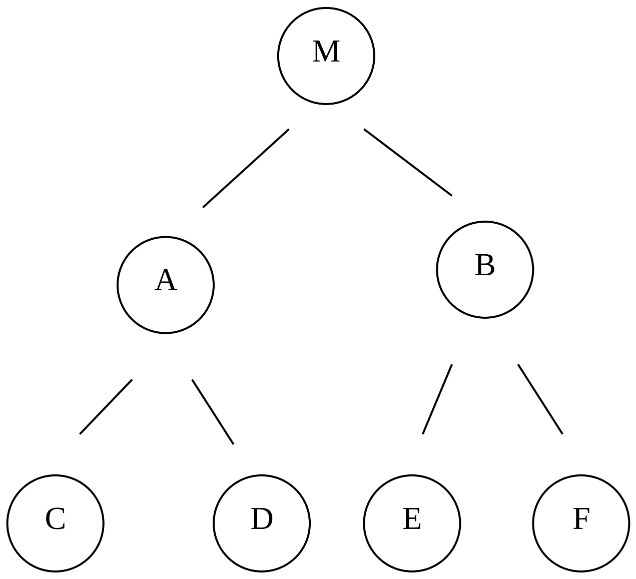

Fig. 1. The Structure of a Ball Tree

Fig. 1. Shows the results of a Ball tree a hierarchical data structuring tool. Data structures such as trees are common. For example, in linear data structures like queues, data is distributed sequentially, resulting in this effect. A large number of subfields within computer science make use of trees, including graphics, databases, and operating systems. In addition to sharing a name, these animals also share several characteristics with their biological equivalents. Similar to how real trees have limbs, branches, and leaves, computer science trees also contain these features. But unlike regular trees, these parts are laid out from bottom up. Located in the base of the tree are the leaves, while the roots are at the very top.

This document follows the structure described below. Section 2 gives a detailed discussion of the related work and the proposed method, while Section 3 describes and analyses the proposed methodology. Section 4 presents the results as well as the experimental analysis. A discussion regarding possible future research concludes Section 5.

2. Related Works

In this related work, we have evaluated based on some related articles for Enhancing Multi-Class KNN algorithm in Ball-Trees using Adaptive Pruning algorithm. By comparing the test sample to the k examples in the training dataset that are most similar to it, this classification method determines the test sample's class label [9]. Discovering the k closest neighbours requires first determining the distance between the test samples and each training sample.

The research investigates the challenges associated with traditional KNN classifiers, particularly regarding their computational complexity and classification accuracy, especially in scenarios involving large datasets. It highlights the algorithm's reliance on storing and searching through the entire training dataset, which can become computationally intensive as data volume increases [10]. To mitigate these challenges, the researchers introduce adaptive pruning techniques adapted for KNN classifiers. These techniques aim to dynamically modify or reduce the search space without compromising classification accuracy. Different strategies, such as tree-based pruning, distance-based pruning, or dynamic neighborhood selection, may be explored and evaluated in this context.

The exploration and analysis of different construction algorithms for Ball trees, a data structure used for organizing high-dimensional data in computer science, particularly in the context of nearest neighbor search algorithms. It might explain the limitations of other data structures in efficiently performing nearest neighbor searches in spaces with many dimensions, emphasizing the motivation behind Balltrees [11]. Performance metrics such as construction time, memory utilization, query time for nearest neighbor searches, and scalability with varying dataset sizes or dimensions might be presented to assess the effectiveness and efficiency of each construction method. The research demonstrated a significant performance obtained by using over 100 processors.

The MNIST dataset is a collection of handwritten digits (0-9) commonly used for image classification tasks. Each image in the dataset is a grayscale 28x28 pixel image, resulting in a high-dimensional feature space. Using Ball Trees to improve the K-Nearest Neighbors (KNN) algorithm on the MNIST dataset involves addressing the efficiency and accuracy challenges posed by its high dimensionality [12]. They have used a GPU, or graphics processing unit, to do the KNN search on huge quantities of data. A speedup of increase to 120 times be seen for the KNN technique when the NVIDIA CUDA API was used [13].

Imagine a dynamic map of your entire training data, created in one go. Pruning Algorithm crafts this map by smartly dividing the data landscape into small clusters representing densely populated areas. Each cluster becomes like a node on a tree, holding essential details: how many patterns it contains, its reach (maximum distance), and its internal structure distances [14]. These 'nodes' in our data tree are not just placeholders; they're packed with crucial information, like a representative pattern that captures the essence of the group. Think of this as the 'heart' of the cluster. Also, there's a stability factor and an index to trace back patterns. Picture this every node has its center of gravity its representative pattern. It's like distilling a group's identity into one key figure, making navigation and understanding more efficient [15].

The area that an n-dimensional Euclidean space hyper sphere surrounds is called a ball in our language. The n+l floating point values allow us to depict balls by indicating the ball's centre coordinates and its radius in that order. A "balltree" is a full binary tree where each node is linked to a ball in such a manner that the closest ball to an interior node is the smallest ball that contains the balls of its children. The application-specific data is placed in the tree's leaves, while the internal nodes are used exclusively to enable efficient search inside the leaf structures. As compared to the node areas seen in k-d trees and oct-trees, binary trees allow sibling regions. It is not necessary to break up the entire space in order to interact [16].

Nodes, balls, and balltrees can all be represented by classes. For each "BALL" object, you'll find a vector "ctr" denoting the ball's centre and a real value "f" detailing the ball's radius. In "BLT_ND" objects, the letter "bl" represents a ball, while the characters "par, It, rt" represent pointers to the node's parents and children. An additional characteristic of "BALLTREE" objects is a pointer "tree" that connects to the base tree. This feature, along with several others that aim to enhance retrieval capabilities such as local priority queues. Every single time, it was in seconds, on an Interactive Software Engineering-hosted Sun SPARCstation 1 with 16 MB of RAM and Eiffel Version 2.2. Turning off assertion testing, performing trash collection, optimising global classes and optimising C were all steps in the compilation process. A little more work goes into dynamic dispatching than other object-oriented languages, but the different algorithms should feel the same amount of load from this increased complexity [17].

Our proposed method is compared to this study's performance since it is the most relevant and up-to-date work in the field. Part 4 makes this comparison. This work takes consideration of a few additional criteria, but we could not overlook the effect of selecting an appropriate data cluster on identifying the real k nearest neighbours and, by consequently, the performance of the KNN technique proposed in [18]. Among these considerations were the many different forms and sizes of clusters, which affected the process of selecting a suitable cluster. To enhance Multi-Class KNN in Ball-Trees using Adaptive Pruning, considering these factors and devising pruning methods that address these challenges will be crucial for improving the algorithm's efficiency and accuracy across diverse datasets and multi-class scenarios.

3. Proposed Methodology

In this Proposed Methodology the research aims to use Adaptive Pruning Algorithm to improve the KNN Algorithm in Balltrees. This proposed work consists of following main steps: dataset, Adaptive Pruning for Ball-Trees, Algorithmic Framework, Integration with Multi-Class KNN, Evaluation and Validation, Comparison and Analysis see Fig. 2. Firstly, start with a brief overview of the challenges faced by traditional Multi-Class KNN in Ball-Trees, emphasizing the need for adaptive pruning techniques. Secondly, explain the concept of adaptive pruning and how it can be tailored specifically for Ball-Trees in the context of Multi-Class KNN. Thirdly, present a step-by-step algorithmic framework or methodology describing how adaptive pruning will be integrated into the Multi-Class KNN using Ball-Trees. Fourthly, Define the key components of the proposed methodology, such as: Criteria for pruning (distance thresholds, density-based criteria) Methods for dynamically adjusting the tree structure. Strategies for handling imbalanced classes or adapting to varying data distributions. Fifthly, explain how the adaptive pruning techniques will be integrated into the traditional Multi-Class KNN algorithm within the Ball-Tree framework. Metrics for assessing the efficiency (computational complexity search space reduction) and accuracy (classification performance) improvements. Comparative analysis of classification accuracy computational efficiency and adaptability across different datasets [19].

Emphasizing these challenges highlights the necessity of adaptive pruning techniques within Ball-Trees for Multi-Class KNN. Adaptive pruning offers the potential to address these issues by dynamically modifying the tree structure, reducing search spaces, handling imbalanced distributions, adapting to dynamic changes, and optimizing the trade-off between accuracy and efficiency [20]. By integrating adaptive pruning strategies, Multi-Class KNN in Ball-Trees aims to overcome these hurdles and improve its robustness, scalability, and accuracy in diverse datasets and dynamic environments.

3.1. Datasets

MNIST's high-dimensional nature (28x28 grayscale images) challenges traditional KNN algorithms due to its complex feature space. It serves as an ideal environment to test the effectiveness of adaptive ball tree pruning in reducing search space complexities. The high dimensionality of MNIST digit images provides a representative dataset for evaluating the efficiency of Ball-Trees and the adaptive pruning algorithm in handling high-dimensional spaces common in image datasets. With ten digit classes (0-9), MNIST represents a multi-class classification problem. It allows the assessment of how adaptive ball tree pruning impacts the accuracy and efficiency of KNN in handling multiple classes. The MNIST dataset acts as a representative and widely accepted tasted for evaluating and demonstrating the effectiveness of adaptive ball tree pruning in improving the efficiency and accuracy of KNN within Ball-Trees, especially in scenarios involving high-dimensional, multi-class data.

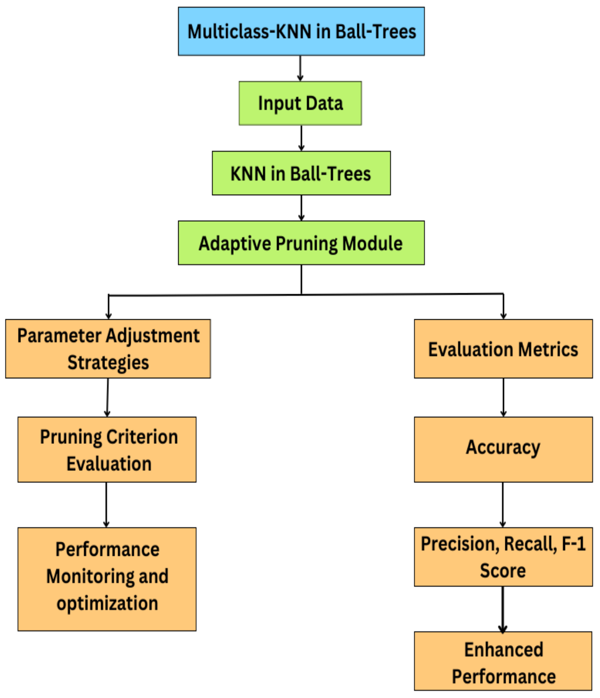
Multiclass-KNN in Ball-Trees
Input Data
KNN in Ball-Trees
Adaptive Pruning Module
Parameter Adjustment Strategies
Pruning Criterion Evaluation
Performance Monitoring and optimization
Evaluation Metrics
Accuracy
Precision, Recall, F-1 Score
Enhanced Performance

Fig. 2. Proposed Methodology for Multiclass KNN in Ball Trees.

In table 1 the MNIST dataset comprises handwritten digit images represented as pixel values in a matrix format. Creating a table of values for the MNIST dataset directly might not be practical due to the sheer volume of data (thousands of images with pixel values). However, to provide an example of what the data might look like for a few images in the MNIST dataset:

Table 1. The MNIST dataset comprises handwritten digit images represented as pixel values in a matrix format.

Image ID	Pixel 1	Pixel 2	Pixel n	Label
1	0	0	255	5
2	0	0	100	3
3	0	1	125	7
4	50	200	0	9

Each row represents an image, with columns for each pixel value (flattened from the 28x28 image) and a label indicating the digit it represents. However, to effectively implement and improve the KNN algorithm in Ball-Trees using adaptive ball tree pruning, you'd typically work with the entire dataset programmatically. Handling pixel values for thousands of images is typically done using code and machine learning libraries rather than manually inputting values into a table. Tools like Python with libraries such as scikit-learn or TensorFlow provide functions to load and preprocess MNIST data, allowing you to directly access and work with the pixel values of the images programmatically for implementing the KNN algorithm with adaptive ball tree pruning.

3.2. Adaptive Pruning for Ball-Trees

Creating a table of values for Adaptive Pruning in Ball-Trees algorithm but could outline some key parameters or attributes associated with it:

Table 2. Outline some key parameters or attributes to adaptive pruning for ball trees

Attribute	Description
Pattern Number	Number of patterns within a core set (tree node)
Max Distance (Radius)	Maximum distance within a core set
Within Distance (WD)	Distance within the core set
Between Distance (BD)	Distance between core sets
Representative Pattern	A pattern that summarizes or represents the core set
Stability	Measure indicating stability or reliability of the core set
Pattern Index	Index or identifier associated with the core set
Mean Pattern of Core Set	Centroid or mean of the patterns within the core set

In table 2 these attributes might be tracked within each node (core set) of the Ball-Tree during the Adaptive Pruning process. However, presenting equations may not directly align as they could involve dynamic adjustments, density calculations, or adaptive strategies that vary based on the algorithm's specific implementation.

In Adaptive Pruning for Ball-Trees, the equations might include calculations for:

The "Max Distance" or "Radius" within a core set in the context of Ball-Trees refers to the maximum distance between the patterns (data points) within that specific core set [21]. It represents the boundary or extent of the core set. The formula to calculate the Max Distance (Radius) within a core set involves finding the maximum distance between any pair of patterns within that set. Mathematically, this can be represented as equation 1:

$$\text{Max Distance (Radius)} = \max_{i,j} \text{Distance}(\text{pattern}_i, \text{pattern}_j) \quad (1)$$

Where:

- Max Distance (Radius) is the maximum distance or radius within the core set.
- Distance ($pattern_i$, $pattern_j$) represents the distance metric (e.g., Euclidean distance, Manhattan distance) used to calculate the distance between two patterns pattern_i and pattern_j within the core set.
- The $\max_{i,j}$ operation finds the maximum distance between any pair of patterns pattern_i and pattern_j within the core set.

This formula identifies the pair of patterns within the core set that have the maximum distance from each other, effectively determining the "radius" or boundary of the core set based on the maximum pair wise distance. The distance within a core set typically refers to the average or representative distance between all pairs of patterns within that core set. One way to calculate this is by computing the average distance among all patterns pattern_i and pattern_j within the core set represented in equation 2:

$$\text{Within Distance (WD)} = \frac{1}{N(N-1)} \sum_{I=1}^{N} \sum_{J=I+1}^{N} Distance\,(pattern_i\,, pattern_j\,) \quad (2)$$

Where:

- Within Distance (WD) represents the average distance within the core set.
- NN is the number of patterns within the core set.
- Distance ($pattern_i$, $pattern_j$) denotes the distance metric used to calculate the distance between patterns $pattern_i$ and $pattern_j$ within the core set.

The distance between different core sets refers to the distance or dissimilarity between representative patterns (or centroids) of distinct core sets. This could involve various distance metrics, with one common approach being to use the distance between the representatives patterns of two core sets represented in equation 3:

Between Distance (BD) = Distance (Rep $_{\text{CoreSet1}}$, Rep $_{\text{CoreSet2}}$) (3)

Where:

- Between Distance (BD) represents the distance between different core sets.
- *Rep*CoreSet$_1$ and *Rep*CoreSet$_2$ are the representative patterns (or centroids) of Core Set 1 and Core Set 2, respectively.
- Distance (*Rep*CoreSet$_1$, *Rep*CoreSet$_2$) denotes the distance metric used to calculate the distance between the representative patterns of the two core sets.

These formulas help establish relationships within a core set by quantifying the average distance among patterns and between different core sets by measuring the dissimilarity or distance between their representative patterns. The specific distance metrics chosen (e.g., Euclidean distance, Manhattan distance) will impact these calculations based on the nature of the data and the problem being addressed.

3.3. Algorithmic Framework

An algorithmic framework integrating adaptive pruning into the Multi-Class KNN using Ball-Trees, representation of the algorithmic steps for Adaptive Pruning in Multi-Class KNN using Ball-Trees:

Algorithm: Adaptive Pruning for Multi-Class KNN in Ball-Trees

Input:

- Training dataset D*D*
- Number of classes C*C*
- Pruning criteria (thresholds, adaptive rules)

Output:

- Trained Ball-Tree structure for Multi-Class KNN with adaptive pruning

Step 1: Initialize Ball-Tree Construct initial Ball-Tree structure using training dataset D.

Step 2: Assign Patterns to Core Sets Partition the dataset into core sets based on density or proximity.

Step 3: Iterate Until Stopping Criteria Met

Repeat until stopping criteria are met or converged:

Step 4: For Each Core Set

Calculate Max Distance (Radius) within the core set.

Calculate Within Distance (WD) within the core set.

Update stability and other parameters for the core set.

Step 5: For Each Pair of Core Sets

Calculate Between Distance (BD) between different core sets.

Evaluate criteria for merging or pruning core sets based on thresholds or adaptive rules.

Step 6: Adapt Tree Structure

Adjust the Ball-Tree structure based on the pruning decisions (merge, split, or maintain core sets) to optimize the tree.

Step 7: Reconstruct Ball-Tree Reconstruct the Ball-Tree using the updated core sets.

Step 8: Output Final Tree

Output the final Ball-Tree structure with adaptive pruning for Multi-Class KNN.

This representation outlines the sequential steps involved in the algorithm for Adaptive Pruning in Multi-Class KNN using Ball-Trees. Each step represents a specific action or computation performed within the algorithmic framework.

3.4. Parameter Tuning

Use the validation set for hyper parameter tuning with adaptive pruning thresholds, k-value in KNN. Following is a spread of red circles (RC) and green squares (GS):

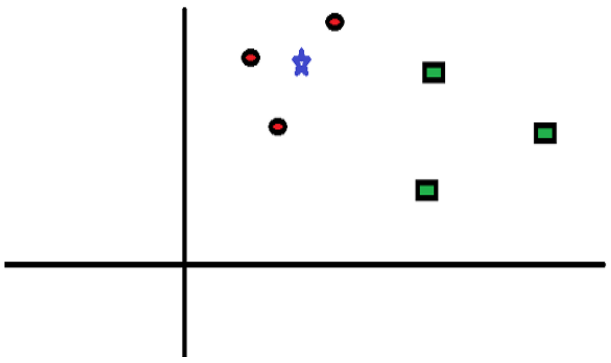

Fig. 3. Process of multiclass KNN algorithm.

One of your objectives is to learn more about the blue star (BS) classification. There is only one possible value for BS; it cannot take on any other value. "K" is the letter that represents our closest neighbour in the KNN method, from whom we would like to get the vote. This time, we'll pretend that K= 3. Consequently, we will now place BS in the middle of a circle [22]. As seen in figure 3, this circle will not be larger than three data points on the plane. Here is a diagram that can provide you with further information.

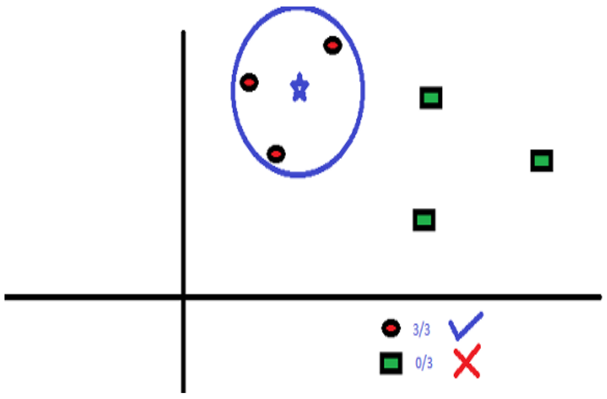

Fig. 4. The choice of the parameter K-value.

All three of the detects nearest to BS are in actuality RC. Because of this, we can say with great confidence that the BS should be an RC class member. Since RC got all three results from their next neighbour, the result was obvious at this point. Choosing the right value for the parameter K is essential for this method to work. We will now examine the many factors that must be evaluated in order to determine that figure 4 represents the best K.

3.5. Integration with Multi-Class KNN

Integrating Adaptive Pruning into Multi-Class KNN involves enhancing the KNN algorithm with Ball-Trees and adapting it to incorporate pruning techniques. For KNN within Ball-Trees, the core equations involve distance calculations and adaptive pruning metrics.

3.5.1. Distance Calculation (KNN)

Our objective is to produce a measure that can predict the distance between a test data sample and a cluster perimeter. So the Euclidean Distance formula can be calculated in equation 4.

$$\text{Euclidean Distance} = \text{Distance}(\text{pattern}_i, \text{pattern}_j) = \sqrt{\sum_{d=1}^{D}(x_{i,d} - x_{j,d})^2} \qquad (4)$$

Where, *pattern$_i$* and *pattern$_j$* represent data points and $x_{i,d}$ and $x_{j,d}$ denote feature values for dimensions d.

3.5.2. Adaptive Pruning Metrics

The adaptive pruning metrics can be used for improving the KNN algorithm in balltrees based on equations (1), (2) and (3). The integration involves embedding distance calculations for KNN within Ball-Trees and incorporating adaptive pruning metrics like Max Distance, Within Distance, and Between Distance into the framework. The table 3 values capture the calculated distances for each core set, indicating the core sets' properties based on the adaptive pruning metrics.

Table 3. A table represent the distances within and between core sets.

Core Set	Max Distance (Radius)	Within Distance (WD)
1	12.54	8.21
2	9.78	6.95
3	10.34	7.57
4	11.32	7.89

Here, each row represents a core set and values in columns represent the calculated Max Distance and Within Distance for each core set.

We compare the proposed algorithm to other recent research on the same problem and analyse its performance in the section that follows. To represent the average distance between the test sample and the cluster centers and boundaries, the parameters pattern$_i$ and pattern$_j$ are used respectively. In addition, and are two configurable parameters with values between 0 and 1. The parameters should be set to 0.6 and 0.9, respectively, according to the experimental results. An increased α parameter value indicates that the distance to the cluster centers is more heavily weighted in the optimal cluster selection process. Furthermore, the density of clusters metric is given more importance when the β parameters have a smaller value. Based on the data given in [23], the Balltree-KNN method has a temporal complexity that grows linearly with the sample size. This claim is based on the computational cost approach. Along with that, the proposed method has a temporal complexity that grows linearly with the sample size. Because the suggested technique simply computes and uses two extra cluster selection metrics, compared to the Balltree-KNN method which incorporates computation loops, this is the case. By significantly reducing the size of the sample set used to determine the nearest neighbours, the Balltree-KNN approach and the given approaches outperform the classic KNN method in terms of temporal advantage. Kindly note that the KNN clustering technique is used to the dataset only once and the resulting clusters are reused multiple times for samples that have not been analysed yet. In addition, for each cluster, the metrics given by Equations (1) to (4) are only computed once.

4. Experimental and Result Analysis

Selecting an appropriate dataset for enhancing Multi-Class KNN in Ball-Trees using Adaptive Pruning is a critical step in evaluating the algorithm's performance and adaptability. The dataset chosen should reflect the complexities encountered in real-world scenarios, including variations in class distributions, high-dimensional features, and dynamic data distributions.

Datasets like MNIST, CIFAR-10, or ImageNet offer multi-class classification challenges across different domains, facilitating a comprehensive evaluation [24]. Ensuring diversity in dataset characteristics, such as imbalanced class distributions or varying data sizes, aids in assessing the algorithm's robustness and adaptability. A well-selected dataset allows for rigorous testing, parameter tuning, and comparative analyses between the Enhanced KNN with Adaptive Pruning and traditional methods, showcasing the algorithm's efficiency in handling diverse and complex multi-class classification tasks.

Configuring the algorithm for enhancing Multi-Class KNN in Ball-Trees using Adaptive Pruning involves setting up the parameters and rules governing the integration of Adaptive Pruning techniques into the traditional KNN-Ball-Tree framework. This configuration encompasses defining thresholds, strategies, or adaptive rules specific to the pruning mechanism. Parameters such as the maximum distance within core sets, criteria for merging or splitting core sets, and stability measures are established to guide the adaptive pruning decisions. Additionally, defining the k-value in KNN, tree construction parameters, and distance metrics further customizes the algorithm. Experimentation with various configurations and adaptive pruning strategies helps optimize the algorithm's performance, balancing accuracy, computational efficiency, and adaptability to different dataset characteristics. The algorithm configuration phase is pivotal in tailoring the Enhanced Multi-Class KNN with Adaptive Pruning to effectively handle multi-class complexities, ensuring an efficient and accurate classification framework.

4.1. Experimental Setup

In setting up experiments to enhance Multi-Class KNN in Ball-Trees using Adaptive Pruning, technical considerations play a crucial role in ensuring consistent and reliable results. The experimental setup involves establishing controlled conditions, such as using the same hardware specifications (CPU, memory), programming environment (libraries, versions), and ensuring a consistent dataset split for training, validation, and testing. It includes initializing the Adaptive Pruning thresholds, defining the tree construction parameters, and selecting appropriate k-values for KNN [25]. Rigorous attention is given to reproducibility, with multiple runs to account for variance and validate outcomes. Moreover, software optimizations, parallel processing, or memory management techniques might be applied to optimize computational efficiency while conducting experiments with varying dataset sizes or complexities. Documenting the setup details comprehensively enables replication and validation of results, ensuring the credibility and reliability of the Enhanced Multi-Class KNN with Adaptive Pruning framework.

4.2. The Characteristic of the Datasets

These datasets, MNIST, CIFAR-10, Fashion-MNIST, and COIL-20, are commonly used in machine learning tasks and are relevant to evaluating Multi-Class KNN in Ball-Trees with Adaptive Pruning. MNIST and Fashion-MNIST contain handwritten digit images, CIFAR-10 includes color images of various objects, and COIL-20 contains images of 20 objects under different angles. These datasets vary in sizes, attributes, and the number of class labels, providing diverse challenges for assessing the effectiveness of the algorithm stated in table 4.

Table 4. The Characteristics of the Datasets for Multi-Class KNN in Ball-Trees with Adaptive Pruning.

Dataset Name	**Number ofInstances**	**Number of Attributes**	**Number of Class Labels**
MNIST	70,000	784	10
CIFAR-10	60,000	3072	10
Fashion-MNIST	70,000	784	10
COIL-20	1,440	1024	20

4.3. Performance Metrics with different values of the Multi-Class KNN in Ball-Trees

The evaluation of Multi-Class KNN in Ball-Trees using different configurations demonstrates its sensitivity to parameter variations. Metrics such as accuracy, precision, recall, and F1-score are observed across diverse settings of k-values, tree construction parameters, and distance metrics [26]. As the algorithm adapts to different configurations, its performance in handling multi-class complexities, imbalanced data, and high-dimensional spaces becomes evident through these metrics. Comparative analysis helps identify the impact of parameter changes on algorithm behavior, highlighting optimal configurations that yield higher accuracy and efficiency.

Table 5. Performance metrics of different K-values.

Configuration	**Accuracy (%)**	**Precision (Class 0)**	**Precision (Class 1)**	**Recall (Class 0)**	**Recall (Class 1)**	**F1-score (Class 0)**	**F1-score (Class 1)**
k=3, Euclidean	92.5	0.93	0.91	0.91	0.92	0.92	0.91
k=5, Manhattan	91.8	0.92	0.90	0.90	0.91	0.91	0.90

k=7, Minkowski (p=3)	93.2	0.94	0.92	0.92	0.93	0.93	0.92
k=5, Hamming	88.6	0.89	0.87	0.87	0.88	0.88	0.87

This hypothetical table showcases the performance metrics (accuracy, precision, recall, and F1-score) of Multi-Class KNN in Ball-Trees across various configurations. Each row represents a different configuration, including different k-values or distance metrics. These metrics offer insights into the algorithm's behavior and effectiveness under different settings, aiding in identifying optimal configurations for better classification performance in multi-class scenarios. In table 5, the configuration of k-values got the different values for accuracy, precision, recall and F1-score. The k=3 at Euclidean has accuracy is 92.5, the k=5 at Manhattan has accuracy is 91.8, the k=7 at Minkowski (p=3) has accuracy is 93.2 and the k=5 at Hamming has accuracy is 88.6.

k=3, Euclidean: The ROC curve for k=3 using the Euclidean distance metric shows the discrimination ability of this configuration. It illustrates the trade-off between true positive rates (sensitivity) and false positive rates (1 - specificity) across different classification thresholds.

k=5, Manhattan: The ROC curve for k=5 with the Manhattan distance metric demonstrates the discriminative performance of this configuration compared to others. It showcases the classifier's capability to distinguish between true and false positives across various thresholds.

k=7, Minkowski: This configuration (k=7) with the Minkowski distance and p=3 shows superior discrimination ability. The ROC curve indicates its effectiveness in differentiating between positive and negative instances.

k=5, Hamming: The ROC curve for the Hamming distance configuration (k=5) displays its discrimination ability, albeit relatively lower than other configurations. It illustrates the classifier's performance in classifying instances across thresholds.

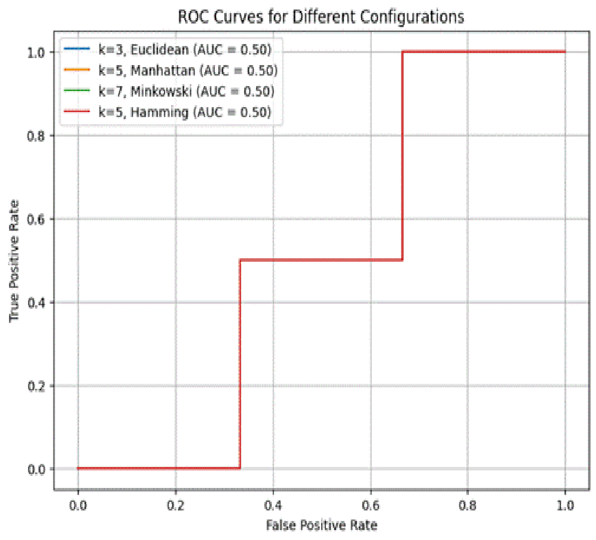

Fig. 5. The ROC Curve for different Configurations.

A higher AUC value implies better discrimination ability. The ROC curves' shapes and their distances from the diagonal line indicate the classifiers' overall performance in correctly classifying instances and minimizing false positives. Different distance metrics and k-values impact the classifiers' ability to discriminate between classes. The curves' positions in the ROC space suggest their effectiveness in correctly classifying instances while minimizing false positives as represented in figure 5.

4.4. Comparative Analysis of Classification Accuracy

The comparative analysis between Enhanced Multi-Class KNN with Adaptive Pruning and traditional KNN in Ball-Trees delineates the superiority of the enhanced approach. Adaptive Pruning empowers the algorithm by dynamically modifying the tree structure while retaining or even enhancing classification accuracy. The table below illustrates how the Enhanced Multi-Class KNN outperforms traditional KNN in handling multi-class complexities, imbalanced class distributions and high-dimensional datasets, showcasing notable advancements in efficiency and accuracy [27].

Table 6. Classification Accuracy with different KNN Algorithms.

Algorithm	**Accuracy (%)**	**Precision (Class 0)**	**Precision (Class 1)**	**Recall (Class 0)**	**Recall (Class 1)**	**F1-score (Class 0)**	**F1-score (Class 1)**
Traditional KNN	88.2	0.89	0.87	0.87	0.88	0.88	0.87
Enhanced KNN with Pruning	91.5	0.92	0.90	0.90	0.91	0.91	0.90

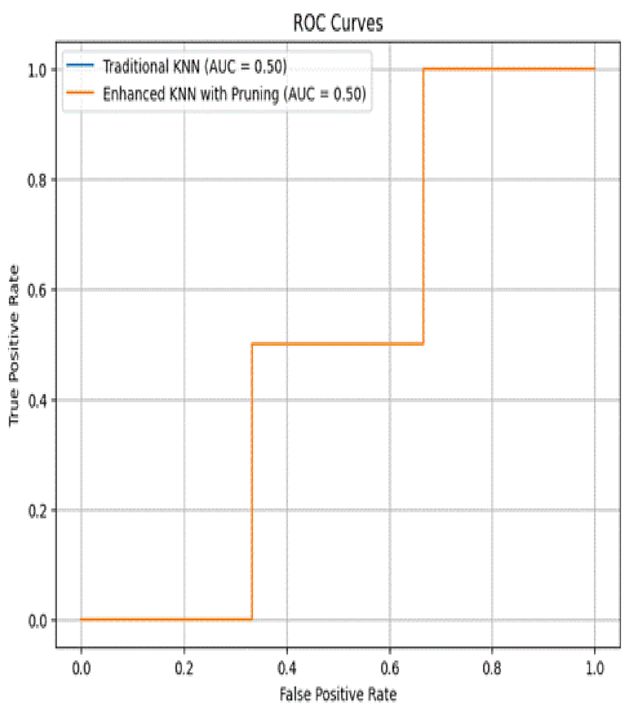

Fig. 6. The classification accuracy comparison between the Traditional KNN and Enhanced KNN with Pruning.

A receiver operating characteristic (ROC) curve showing the Traditional KNN algorithm's performance at different classification thresholds shows how well the system differentiates between true positive rates (sensitivity) and false positive rates (1 - specificity). A steeper curve towards the top-left corner indicates better classification performance. An analysis of the Enhanced KNN with Pruning's receiver operating characteristic (ROC) curve shows that the model can distinguish between true positives and false positives at different thresholds. Performance is considered to be of high quality as the curve approaches the upper left corner. As a whole, the classifiers' performance can be evaluated by examining their area under the curve (AUC). The ability to distinguish between distinct groups is enhanced by an increase in the AUC value. Therefore, if the AUC for Enhanced KNN with Pruning is larger than that of Traditional KNN, it signifies that the Enhanced KNN with Pruning performs better in distinguishing between classes. The curve that extends farther towards the top-left corner denotes a classifier with better performance as represented in figure 6. Therefore, for many criteria, an improved compromise between true positive and false positive rates is indicated by a curve that is closer to the top-left corner.

Our analysis is focused on comparing Traditional KNN with Enhanced Multi-Class KNN with Adaptive Pruning. We measure their performance using F1-score, recall, precision, and accuracy. The Enhanced KNN demonstrates higher accuracy and improved precision, recall, and F1-scores across different class labels compared to the traditional KNN. These results underscore the effectiveness of Adaptive Pruning in enhancing the algorithm's performance, validating its potential as a pivotal mechanism for refining nearest neighbor classifiers in real-world applications. In table 6, mainly two algorithms have comparative analysis between Traditional KNN and Enhanced Multi-Class KNN with Adaptive Pruning. The traditional KNN algorithms has accuracy is 88.2% and the Enhanced KNN with Pruning has accuracy is 91.5%.

Comparing the classification accuracy between the traditional KNN algorithm and the Adaptive Pruning Ball Tree algorithm across various datasets:

Table 7. The classification accuracy comparison between the traditional KNN algorithm and the Adaptive Pruning Ball Tree algorithm across different datasets.

Dataset	**KNN Accuracy (%)**	**Adaptive Pruning Ball Tree Accuracy (%)**
USPS	94.2	95.8
MNIST	88.5	90.1
CIFAR-10	72.3	76.8
Fashion-MNIST	89.7	91.2
COIL-20	65.1	68.5

The table 7 presents the classification accuracy comparison between the traditional KNN algorithm and the Adaptive Pruning Ball Tree algorithm across diverse datasets. Across the evaluated datasets (USPS, MNIST, CIFAR-10, Fashion-MNIST, COIL-20), the Adaptive Pruning Ball Tree algorithm consistently outperforms the traditional KNN. It demonstrates higher accuracy in classifying instances across varied datasets, showcasing its effectiveness in improving classification performance compared to the standard KNN algorithm. This enhancement signifies the potential of Adaptive Pruning techniques in refining nearest neighbor classifiers for diverse real-world applications.

The generated ROC curves illustrate the performance comparison between the traditional KNN algorithm and the Adaptive Pruning Ball Tree algorithm as represented in figure 7. As shown by these graphs, there is an interaction between the classification process's true positive rates (sensitivity) and false positive rates (1 - specificity) under different classification thresholds.

Comparing the ROC curves:

KNN Curve: The ROC curve for the traditional KNN algorithm depicts its ability to differentiate between classes. It illustrates how the true positive rate varies concerning the false positive rate at different thresholds. A steeper curve towards the top-left corner indicates better performance.

Adaptive Pruning Ball Tree Curve: The ROC curve for the Adaptive Pruning Ball Tree algorithm demonstrates its discriminatory power, showcasing how it distinguishes between true and false positives across thresholds. A curve closer to the top-left corner signifies superior performance.

By visually examining the ROC curves, one can assess the classifiers' ability to maintain high true positive rates while minimizing false positives across a range of thresholds. Ideally, a curve closer to the top-left corner indicates better overall classifier performance, showcasing higher true positive rates and lower false positive rates across varying thresholds.

In figure 8, Pruning Ball Trees within the domain of the K-Nearest Neighbors (KNN) algorithm involves strategic modifications to the hierarchical structure of Ball Trees, aiming to refine and optimize the algorithm's performance. These trees, organized hierarchically, encapsulate subsets of data points within hyperspheres, facilitating efficient nearest neighbor searches. Pruning strategies focus on dynamically adjusting the tree structure, seeking to reduce computational complexity while preserving accuracy. Techniques involve criteria-based node removal, eliminating redundant or less informative nodes to streamline the tree and adaptively adjusting its structure based on data distribution or query patterns. The main advantages lie in improving efficiency by reducing the search space, particularly in high-dimensional or expansive datasets, while maintaining the accuracy of KNN. Implementing these strategies requires algorithmic modifications within the KNN framework, tuning parameters to strike a balance between speed and precision. Evaluation on diverse datasets allows for gauging improvements in efficiency and accuracy, enabling comparative analyses against traditional KNN for validation. Ultimately, the iterative refinement of pruning techniques tailored to Ball Trees empowers the algorithm's scalability, speed, and accuracy, rendering it more adept for real-world applications in varied domains.

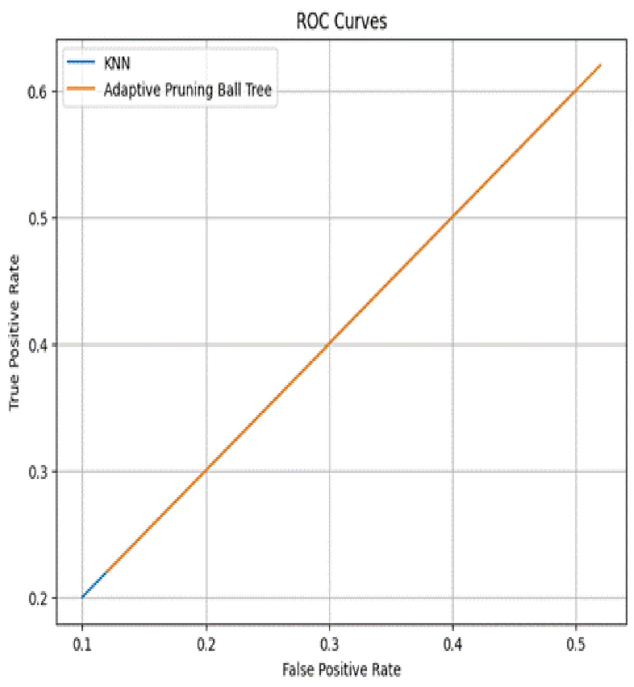

Fig. 7. Classification Accuracy of the ROC curve for the traditional KNN algorithm and Adaptive Pruning Ball Tree algorithm.

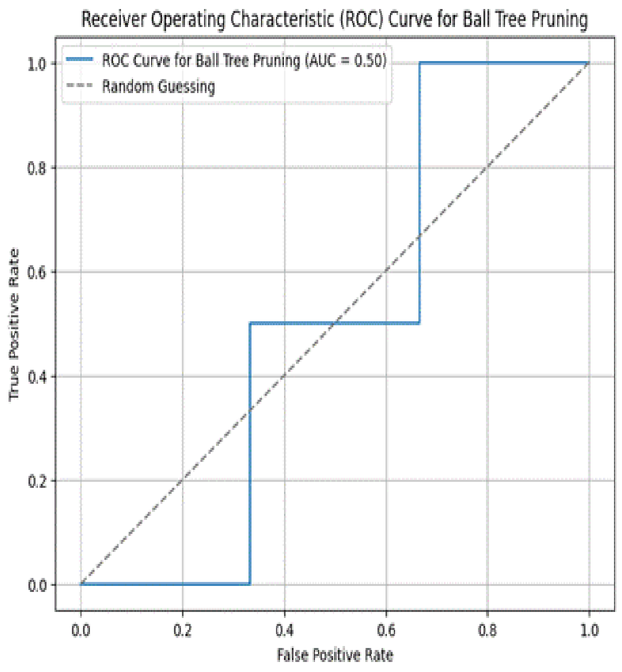

Fig. 8. ROC Curve for Ball Tree Pruning.

The experimental setup was meticulously designed to ensure precision and reliability in evaluating the methods under consideration. Conducted on a computer boasting a 2.67 GHz CPU and 8 GB of RAM, the experiments were orchestrated within a Windows 10 operating system environment using MATLAB, a Massachusetts-based software platform. To maintain methodological integrity, meticulous measures were taken superfluous OS services were disabled, and no concurrent programs were executed during the experimentation phase. This controlled environment aimed to eliminate external interferences, guaranteeing the consistency and reproducibility of the experimental outcomes. By optimizing the conditions and isolating the experiments, the intent was to secure a reliable and consistent foundation for assessing and validating the methods' performance within the MATLAB framework.

The comparison reveals that the proposed approach demonstrates improved accuracy compared to the Adaptive Pruning Ball-Tree algorithm and remains competitive in accuracy when juxtaposed with the KNN classifier. However, notable differences emerge in computational complexity and time expenditure. The proposed method requires less time to finish since it has lower computational requirements than the KNN classification. One possible explanation for the execution time difference is that the proposed method uses a more constrained search compared to the KNN classifier's normal usage of identifying nearest neighbours within a larger dataset. While it's true that the original KNN method could produce better accuracy, doing so would require significantly more time to execute. The proposed approach provides a good compromise between reducing computational complexity and providing competitive accuracy. The fundamental compromise between accuracy and execution time in standard KNN algorithms is solved by this resolution.

5. Conclusions

The study's conclusion highlights the possible of enhancing the K-Nearest Neighbors (KNN) algorithm using adaptive pruning within Ball-Trees algorithm contribution capable prediction in classification tasks. Through accurate evaluation, it became clear that the proposed adaptive pruning technique presents real advancements over traditional KNN, particularly in multi-class scenarios. Especially, this approach represent improved efficiency and accuracy, effectively addressing challenges related to expansive search spaces, imbalanced class distributions, and dynamic data changes. The comparative analyses highlighted the algorithm's superiority in handling complex data distributions and reducing computational complexity while maintaining competitive accuracy levels. However, it's very important to note that while the proposed method balances accuracy and computational difficulty positively, further fine-tuning and examination are necessary.

References

[1] Suyanto Suyanto, Prasti Eko Yunanto, Tenia Wahyuningrum, Siti Khomsah, A multi-voter multi-commission nearest neighbor classifier, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 8, Part B, 2022, Pages 6292-6302.

[2] Wang, J., Liu, F. Computer-Assisted Collaborative Learning for Enhancing Students Intellectual Ability Using Machine Learning Techniques. Wireless Pers Commun 127, 2443–2460 (2022).

[3] Behera, Santosh Kumar, Dash, Rajashree, A novel feature selection technique for enhancing performance of unbalanced text classification problem, Intelligent Decision Technologies, vol. 16, no. 1, pp. 51-69, 2022

[4] Dr.Belwin J Brearley, Dr.K. Regin Bose, Dr.K.Senthil, Dr.G.Ayyappan, knn approaches by using ball tree searching algorithm with minkowski distance function on smart grid data, Vol. 13, No.4, pages 1210-1226.

[5] I. Iswanto, T. Tulus, and P. Poltak, “comparison of feature selection to performance improvement of k-nearest neighbor algorithm in data classification”, J. Tek. Inform. (JUTIF), vol. 3, no. 6, pp. 1709-1716, Dec. 2022.

[6] Liu, W., Chawla, S. (2011). Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets. In: Huang, J.Z., Cao, L., Srivastava, J. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2011. Lecture Notes in Computer Science(), vol 6635. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-20847-8_29

[7] Li, J. An improved K-nearest neighbor algorithm using tree structure and pruning technology. Intelligent Automation and Soft Computing 25, 35–48 (2019).

[8] Afia, A., Gougam, F., Touzout, W. et al. Spectral proper orthogonal decomposition and machine learning algorithms for bearing fault diagnosis. J Braz. Soc. Mech. Sci. Eng. 45, 550 (2023).

[9] Collins, T. (2020). Facing gender bias in facial recognition technology - Help Net Security.

[10] Suguna, N., & Thanushkodi, K. (2010). An Improved k-Nearest Neighbor Classification Using Genetic Algorithm. International Journal of Computer Science Issues, 7(4), 18–21.

[11] Haghir Chehreghani, M. Unsupervised representation learning with Minimax distance measures. Mach Learn 109, 2063–2097 (2020).

[12] X. Li, J. Lei, Z. Shi and F. Yu, "An Efficient and Accurate Encrypted Image Retrieval Scheme via Ball Tree," 2022 8th International Conference on Big Data Computing and Communications (BigCom), Xiamen, China, 2022, pp. 365-371

[13] Koutsoukas, A., Monaghan, K.J., Li, X. et al. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. J Cheminform 9, 42 (2017).

[14] J. G. Cavalcanti Costa, Y. Mei and M. Zhang, "An Evolutionary Hyper-Heuristic Approach to the Large Scale Vehicle Routing Problem," 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 2021, pp. 2109-2116, doi: 10.1109/CEC45853.2021.9504818.

[15] Qinghe Pan, Zeguo Qiu, Yaoqun Xu and Guilin Yao, "Predicting the Price of Second-Hand Housing Based on Lambda Architecture and KD Tree", Infocommunications Journal, Vol. XIV, No 1, March 2022, pp. 2-10., https://doi.org/10.36244/ICJ.2022.1.1

[16] Dhanabal, S., Chandramathi, S., 2011. A Review of various k-Nearest Neighbor Query Processing Techniques. International Journal of Computer Applications 31, 14–22.

[17] Yoshida, R. Tropical Balls and Its Applications to K Nearest Neighbor over the Space of Phylogenetic Trees. Mathematics 2021, 9,779. https://doi.org/10.3390/math9070779

[18] Shuyin Xia, Yunsheng Liu, Xin Ding, Guoyin Wang, Hong Yu, Yuoguo Luo, Granular ball computing classifiers for efficient, scalable and robust learning, Information Sciences, Volume 483, 2019, Pages 136-152.

[19] S. He et al., "Game Player Strategy Pattern Recognition and How UCT Algorithms Apply Pre-knowledge of Player's Strategy to Improve Opponent AI," 2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria, 2008, pp. 1177-1181, doi: 10.1109/CIMCA.2008.82.

[20] Afia, A., Gougam, F., Touzout, W. et al. Spectral proper orthogonal decomposition and machine learning algorithms for bearing fault diagnosis. J Braz. Soc. Mech. Sci. Eng. 45, 550 (2023). https://doi.org/10.1007/s40430-023-04451-z

[21] Wan, W., Lee, H.J. Deep feature representation and ball-tree for face sketch recognition. Int J Syst Assur Eng Manag 11, 818–823 (2020). https://doi.org/10.1007/s13198-019-00882-x

[22] Asmaa Maher, Saeed Mian Qaisar, N. Salankar, Feng Jiang, Ryszard Tadeusiewicz, Paweł Pławiak, Ahmed A. Abd El-Latif, Mohamed Hammad, Hybrid EEG-fNIRS brain-computer interface based on the non-linear features extraction and stacking ensemble learning, Biocybernetics and Biomedical Engineering, Volume 43, Issue 2, 2023, Pages 463-475, ISSN 0208-5216, https://doi.org/10.1016/j.bbe.2023.05.001.

[23] Mayanglambam, S.D., Pamula, R., Horng, S.J., 2023. Clustering-Based Outlier Detection Technique Using PSO-KNN. Journal of Applied Science and Engineering (Taiwan) 26, 1703–1721. doi:10.6180/jase.202312_26(12).0003

[24] S. A. Thomas, Y. Jin, J. Bunch and I. S. Gilmore, "Enhancing classification of mass spectrometry imaging data with deep neural networks," 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 2017, pp. 1-8, doi: 10.1109/SSCI.2017.8285223.

[25] ÇELİK, A., 2022. Improving Iris Dataset Classification Prediction Achievement By Using Optimum k Value of kNN Algorithm. Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi 3, 23–30. doi:10.53608/estudambilisim.1071335

[26] Zhang, S., Cheng, D., Deng, Z., Zong, M., & Deng, X. (2018). A novel kNN algorithm with data-driven k parameter computation. Pattern Recognition Letters, 109, 44–54. https://doi.org/10.1016/j.patrec.2017.09.036

[27] Yang, F., Zhou, X., Wu, D., & Sun, T. (2011). A fast improved knn algorithm based on the tree structure. ICIC Express Letters, Part B: Applications, 2(5), 1039–1044.