
**Faculty of Computer Science and Information Technology**

# An Improve k-NN Classifier using Similarity Distance Plot-Data Reduction and Dask for Big Datasets

**Abdul Muqtasid bin Rushdi**

**Master of Science**
**2025**

# An Improve k-NN Classifier using Similarity Distance Plot-Data Reduction and Dask for Big Datasets

Abdul Muqtasid bin Rushdi

A thesis submitted

In fulfillment of the requirements for the degree of Master of Science

(Knowledge Technology)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2025

## DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

……

Signature

Name: Abdul Muqtasid bin Rushdi

Matric No.: 22020181

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 9th July 2025

## ACKNOWLEDGEMENT

I would like to thank to Almighty Allah S.W.T for giving my strength and patience to complete my study. Then, I would like to express my gratitude to my supervisor Dr Mohammad bin Hossin who always provided guidance, encouragement and support in my journey to complete this research. Without their persistent and commitment, I will not be able to finish this research. Not to forget, I would like to thank all my friends in Postgraduate Laboratory for the constructive discussion and moral supports. I also want to thank G13 staffs who always helping me with tools and technical support to carry out my experiments.

I would like to thank my parent for their patience, financial and moral supports. Finally, I would like to acknowledge the support from Ministry of Higher Education Malaysia and Universiti Malaysia Sarawak (UNIMAS) through the Fundamental Research Grant Scheme: FRGS/1/2021/ICT02/UNIMAS/02/6.

## ABSTRACT

The k-Nearest Neighbour (k-NN) algorithm is one of the most widely used Instance-Based Learning methods due to its simplicity and ease of implementation. However, k-NN faces two major challenges, particularly when applied to large datasets. First, it requires substantial memory to store all training instances and compute distances during classification. Second, this results in slower classification speeds, as the algorithm must search through a large number of instances to classify each test sample. To address these challenges, this study proposes a data reduction technique called Similarity Distance Plot–Data Reduction (SDP-DR), which aims to reduce the volume of data stored, thereby improving classification speed. To accommodate big-scale data, the method integrates a parallel computing framework, Dask, during both the reduction and classification processes. The classification of test data follows the conventional k-NN procedure. The performance of SDP-DR is systematically evaluated using benchmark datasets of varying sizes. Comparisons are made against the original k-NN and several advanced data reduction and classification methods, including RIS, DROP3, ATISA1, LHS-FKNN, CQ-EKNN, and GEK-NN. The evaluation criteria include classification accuracy, classification time, data reduction rate, and reduction time. Experimental results demonstrate that the proposed method significantly reduces data storage requirements, improves classification speed, and maintains or even enhances accuracy compared to both the original k-NN and other state-of-the-art methods. Overall, SDP-DR demonstrates strong potential as a scalable and accurate approach for enhancing instance-based classifiers in big-scale data environments.



**Keywords:** k-NN, data reduction, similarity distance, big data, Dask

***Pengklasifikasi k-NN yang Dipertingkat melalui Plot Jarak Keserupaan–Pengurangan Data dan Dask bagi Set Data Skala Besar***

## *ABSTRAK*

*Algoritma k-Nearest Neighbour (NN) merupakan salah satu kaedah Instance-Based Learning yang paling banyak digunakan kerana kesederhanaannya dan kemudahan pelaksanaannya. Namun begitu, k-NN menghadapi dua cabaran utama, terutamanya apabila digunakan pada set data berskala besar. Pertama, ia memerlukan memori yang besar untuk menyimpan semua data latihan dan mengira jarak semasa proses pengelasan. Kedua, kelajuan pengelasan menjadi perlahan kerana algoritma perlu mencari dalam sejumlah besar data untuk mengelaskan setiap sampel ujian. Bagi menangani cabaran ini, kajian ini mencadangkan satu teknik pengurangan data yang dikenali sebagai Plot Jarak Kesamaan-Pengurangan Data (PJK-PD), yang bertujuan untuk mengurangkan jumlah data yang disimpan, sekaligus meningkatkan kelajuan pengelasan. Untuk menampung data berskala besar, kaedah ini mengintegrasikan rangka kerja pengkomputeran selari, Dask, dalam kedua-dua proses pengurangan dan pengelasan. Proses pengelasan data ujian mengikuti prosedur asal k-NN. Prestasi PJK-PD dinilai secara sistematik menggunakan pelbagai set data penanda aras, dan dibandingkan dengan k-NN asal serta kaedah pengurangan data dan pengelasan lain seperti RIS, DROP3, ATISA1, LHS-FKNN, CQ-EKNN, dan GEK-NN. Penilaian berdasarkan ketepatan, masa pengelasan, kadar pengurangan, dan masa pengurangan. Hasil eksperimen menunjukkan bahawa PJK-PD berjaya mengurangkan keperluan storan, mempercepatkan pengelasan, dan mengekalkan atau meningkatkan ketepatan berbanding kaedah sedia ada.*

***Kata kunci:*** *k-NN, Pengurangan Data, Jarak Kesamaan, Data Raya, Dask*

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| ATISA1 | Adaptive Threshold-Based Instance Selection Algorithm |
| B | Balance |
| BPLSH | Border Point Extraction Based on Local-Sensitive Hashing |
| CD | Class Distribution |
| CNCA | Condensed Neighbourhood Components Analysis |
| CNN | Condensed Nearest Neighbour |
| CQ-EKNN | Composite Quantization Ek-NN |
| CQ-KNN | Composite Quantization of k-NN |
| DAG | Directed Acyclic Graph |
| DCM | Data Closest to Mean |
| DEKNN | Distributed Ek-NN Classification |
| DR | Data Reduction |
| DROP3 | Decremental Reduction Optimization Procedure 3 |
| DSOS | Dynamic Self-Organising Swarm |
| EGDIS | Enhanced Global Density-Based Instance Selection |
| EIB | Extreme Imbalanced |
| EIS | Evidential Instance Selection |
| EIS-AS | Evidential Instance Selection with Apache Spark |
| ENN | Edited Nearest Neighbour |
| FD | First Data |
| FKNN | Fuzzy k-Nearest Neighbour |
| GDIS | Global Density-Based Instance Selection |
| GEK-NN | Global Exact Ek-NN |

| | |
|---|---|
| GHAHS-FKNN | Global Approximate Hybrid Spill Tree |
| HDFS | Hadoop Distributed File System |
| HPC | High-Performance Computing |
| I | Integer |
| IB | Imbalanced |
| IBL | Instance Based Learning |
| IG | Information Gain |
| k-NN | k-Nearest Neighbour |
| KNN-IS | k-Nearest Neighbour Instance Selection |
| L | Large |
| LDA | Linear Discriminant Analysis |
| LEK-NN | Local Approximate Ek-NN |
| LHS-FKNN | Local Hybrid Spill Tree FKNN |
| LSH | Local-Sensitive Hashing |
| M | Medium |
| Mchen | Multilabel Chen algorithm |
| MEC | Mean Of Each Column |
| MjC | Majority Class |
| MnC | Minority Class |
| MRHC | Multilabel Reduction Through Homogeneous Clustering |
| MRSP | Multilabel Reduction Through Space Partitioning |
| ms | Milliseconds |
| MV | Missing Value |
| N | Nominal |
| NN | Nearest Neighbour |

| | |
|---|---|
| NoA | No. Of Attribute |
| NoI | No. Of Instances |
| PC | Personal Computer |
| PCA | Principal Component Analysis |
| PG | Prototype Generation |
| PS | Prototype Selection |
| *ps* | Sign Test |
| PSNB | Prototype Selection Neighbourhood Boundary |
| *pw* | Wilcoxon Test |
| QDPSKNN | Quad-Division Prototype Selection Based k-NN |
| R | Real Number |
| RAM | Random Access Memory |
| RDD | Resilient Distributed Datasets |
| RFE | Recursive Feature Elimination |
| RIS | Ranking-Based Instance Selection |
| S | Small |
| SDP-DR | Similarity Distance Plot – Data Reduction |
| *ss* | Win/Draw/Lose Record |
| TG | Reduce Set |
| TR | Training Set |
| TS | Testing Set |

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

The k-nearest neighbour (k-NN) algorithm is one of the most prominent methods within Instance-Based Learning (IBL) due to its simplicity, ease of implementation, and ease of debugging (Ali et al., 2020; Cunningham & Delany, 2021). The k-NN algorithm classifies data points by identifying the *k* closest neighbours in a feature space and assigning the label that appears most frequently among them. This process requires an accurately annotated dataset, where each instance is classified by distinct attributes and corresponding labels or values. To achieve this, the k-NN algorithm determines the proximity of each data to a testing instance using a chosen similarity distance measure. The *k* nearest data points are selected as neighbours, and the target point's label is assigned based on the most frequently observed label among its neighbouring points. While k-NN works efficiently for small and medium-sized datasets, its performance deteriorates with large datasets. The primary challenges associated with k-NN for big data classification are the substantial memory required to store all prototypes and the slower classification process due to exhaustive searching among all prototypes (Garcia et al., 2012; Gong et al., 2021; Kasemtaweechok & Suwannik, 2016).

Recent studies have highlighted the application of the k-NN algorithm for big data classifications (Ali et al., 2020; Su et al., 2021; Xing & Bei, 2020). The term 'big data' emerged in the mid-1990s and became popularised in 2006 following the development of the second generation of the World Wide Web. Big data refers to large, complex datasets that are difficult to store, manage, and process using traditional data processing tools (Liu, 2015). Classifying big data using k-NN presents two key challenges:

i. The need for significant memory space to store all instances as k-NN prototypes for the classification process (Garcia et al., 2012; Gong et al., 2021; Kasemtaweechok & Suwannik, 2016).

ii. Reduced classification speed due to the exhaustive search through all prototypes to classify the testing set (TS) (Gong et al., 2021).

To address these challenges, researchers have proposed several solutions. One approach involves leveraging distributed and parallel computing frameworks such as Apache Spark (Gong et al., 2023a; Gong et al., 2021), Apache Hadoop (Mostafaeipour et al., 2021), and Dask (Rezanejad & Danesh, 2024). These frameworks distribute the k-NN classification process across multiple computing nodes, enabling efficient handling of large datasets. While Apache Spark and Apache Hadoop are designed for large-scale distributed computing, Dask facilitates parallel computing on a single personal computer (PC) and supports Python libraries such as scikit-learn (Pedregosa et al., 2011), NumPy (Harris et al., 2020), and pandas (McKinney, 2010), thereby reducing the learning curve for developers.

Another approach to addressing the challenges of k-NN in the context of big data is to improve the algorithm itself through dimensionality reduction (Jaruenpunyasak & Duangsoithong, 2021; Nobre & Neves, 2019; Solorio-Fernández et al., 2019) and data reduction techniques (Alexandropoulos et al., 2019; Aslani & Seipel, 2021; Sisodia & Sisodia, 2022). Dimensionality reduction involves decreasing the number of features in the dataset while preserving essential information. This can be achieved using methods like feature selection or feature extraction. Feature selection filters out irrelevant or redundant attributes, while feature extraction generates new features using techniques like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Autoencoders. By

reducing the number of features, dimensionality reduction accelerates machine learning algorithms, including k-NN.

Data reduction techniques (DRTs) on the other hand aim to reduce the size of the training dataset while maintaining classification accuracy. These techniques fall into two main categories: Prototype Selection (PS) (Garcia et al., 2012) and Prototype Generation (PG) (Triguero et al., 2012). Prototype Selection identifies and retains a subset of training instances that are crucial for classification, whereas Prototype Generation creates new prototypes by aggregating similar training instances of the same class. Both approaches reduce storage requirements and computational overhead (Ougiaroglou et al., 2023).

Prototype Selection can be categorised into condensation, edition, and hybrid techniques (Garcia et al., 2012). Condensation techniques create a consistent training set by removing unnecessary instances, while edition techniques focus on eliminating noisy instances to improve classifier accuracy. Hybrid techniques integrate both approaches to remove noisy and superfluous instances simultaneously.

Prototype Generation is further classified into four main approaches (Triguero et al., 2012):

i. Class re-labelling: Adjusts the class labels of training samples to correct mislabelling errors.
ii. Centroid-based generation: Creates artificial prototypes by averaging the attribute values of similar examples.
iii. Space splitting: Partitions the feature space and replaces partitioned areas with representative prototypes.

iv. Positioning adjustment: Uses optimisation techniques to adjust prototype positions in the feature space.

Based on the challenges and existing solutions, this study proposes a new approach for k-NN classification that leverages data reduction to address memory and speed limitations when handling big data. The central idea is to reduce the number of prototypes while maintaining classification performance. To achieve this, a novel data reduction method is developed and further enhanced through a parallel computing framework. This approach aims to provide an efficient and accessible solution for big data classification.

The next section discusses the problem statement, followed by the research questions, research aim, and objectives, research scope, and expected outcomes. The chapter concludes with an outline of the thesis organization.

## 1.2 Problem Statement

One of the primary challenges associated with using the k-nearest neighbours (k-NN) algorithm for large datasets is the substantial storage required to retain all instances as prototypes for classification purposes. This storage requirement adversely affects classification speed since the process must search through all prototypes to classify a test instance. This operation becomes increasingly slow as the number of prototypes grows (Gong et al., 2021; Papanikolaou et al., 2021; Syriopoulos et al., 2025). To overcome these issues, researchers have explored dimensionality reduction and data reduction techniques to improve the efficiency of the k-NN algorithm.

Dimensionality reduction aims to reduce the number of features in a dataset while preserving critical information, although this process can lead to some loss of information. According to Li et al. (2020), this loss of information does not significantly affect

classification performance in some cases. Nevertheless, dimensionality reduction is not suitable for all datasets, particularly healthcare and biological datasets, as the process may obscure the original meaning of features that have clear physical interpretations (Jia et al., 2022).

On the other hand, data reduction focuses on reducing the number of prototypes, thereby lowering computational complexity. While this approach shortens the processing time, excessive reduction can negatively impact classification accuracy and result in misclassification (Li & Dai, 2022). To address this, some researchers have incorporated fuzzy logic into the k-NN algorithm (Mailagaha Kumbure & Luukka, 2024). This approach has been shown to improve classification accuracy. However, it also increases computational overhead, requiring more processing power and longer classification times.

Recently, researchers have turned to distributed and parallel computing frameworks to enhance the overall performance of the k-NN algorithm for big data. Frameworks like Apache Spark and Hadoop enable the partitioning and parallelisation of the classification process. While these frameworks provide substantial scalability, they also impose significant hardware requirements. In particular, Apache Spark processes data in random access memory (RAM), whereas Hadoop processes data on hard disks. Consequently, Apache Spark is preferred due to its faster processing times, although it incurs higher costs as a result of increased memory consumption. These limitations highlight the need for more lightweight and accessible solutions that can support scalable big data processing without incurring excessive computational costs.

Although numerous data reduction methods have been proposed, many rely on divide-and-conquer and merging strategies or on distributed paradigms such as MapReduce.

While these approaches improve scalability, they often introduce additional computational overhead and complexity. Furthermore, existing methods that integrate machine learning techniques for prototype selection tend to require extensive resources and may not generalize well across different datasets. Consequently, there remains a need for a data reduction approach that is both computationally efficient and capable of maintaining high classification accuracy for big data applications.

Although alternative algorithms such as Random Forests, Support Vector Machines, and Neural Networks are widely applied in big data classification, k-NN remains a valuable benchmark due to its simplicity, interpretability, and strong performance in diverse domains when paired with effective data reduction techniques. Unlike model-based approaches, k-NN is non-parametric and instance-based, making it highly adaptable to dynamic datasets without retraining. Improving the efficiency of k-NN through lightweight data reduction and parallel processing therefore not only enhances its scalability for big data applications but also provides methodological insights that can benefit other instance-based learning algorithms.

Another unresolved challenge is the application of similarity distance in data plotting and reduction. There are not specific research that address the use of similarity distance for both plotting and data reduction. Typically, similarity distance is used to compute the degree of similarity between two data points but not as a data reduction method.

## 1.3 Research Questions

Based on the aforementioned problem, several research questions are identified:

i. How can a Similarity Distance Plot–Data Reduction algorithm be designed using data reduction approaches?
ii. How can the utilisation of the Similarity Distance Plot–Data Reduction algorithm and Dask, particularly for big data, contribute to substantial improvements in k-NN for big data classification?
iii. How does the proposed method compare to the original k-NN in terms of classification accuracy, prototype storage, and duration required to complete the classification process when handling datasets of varying sizes?

## 1.4 Aim and Objectives of the Study

This study aims to propose an enhancement approach for the k-Nearest Neighbours (k-NN) algorithm by utilising a Similarity Distance Plot–Data Reduction technique, leveraging the capabilities of Dask to address classification challenges in big data scenarios.

The specific objectives required to accomplish the aim of this study include:

i. To investigate existing prototype selection techniques, parallel and distributed computing approaches, and similarity distance measures used to enhance k-NN classification methods particularly in the context of big dataset.
ii. To develop a Similarity Distance Plot–Data Reduction technique integrated with Dask, incorporating the nearest neighbour rule.
iii. To evaluate the effectiveness and efficiency of the proposed method in comparison with the original kNN-IS algorithm and several state-of-the-art data reduction

methods in terms of classification performance, prototype storage, classification time, and reduction time.

## 1.5 Scope of the Study

This study focuses on designing, developing, and evaluating the Similarity Distance Plot-Data Reduction algorithm to improve k-NN classification in terms of efficiency and effectiveness. Prototype selection will be used as a data reduction technique. The performance of the proposed algorithms will be evaluated using a variety of benchmark datasets from the UCI Machine Learning Repository (Asuncion, 2007) and the KEEL dataset repository (Alcalá-Fdez et al., 2011), with results compared against the original k-NN algorithm and other data reduction techniques. The evaluation will be conducted on datasets of varying sizes, including small, medium, and big data. For big data, the focus is on the volume of structured data. Additionally, this study does not address challenges associated with imbalanced datasets or the curse of dimensionality. This study also does not aim to address issues related to noisy data or missing values. To further enhance the efficiency and effectiveness of the proposed algorithm, Dask was employed to ensure scalability across datasets of varying sizes.

## 1.6 Significance of the Study

The following sections outline the key contributions of this research across knowledge, economic, and community aspects:

Knowledge Contribution

The significance of this study lies in the new method produced by implementing data reduction techniques with similarity distances, offering performance comparable to or

potentially better than the traditional k-NN algorithm. This advance understanding and development in the field of data classification techniques.

### Economic Contribution

The second key contribution of the study is economic in nature, as it reduces the time complexity of the classification process. This benefits end users by significantly shortening the time required to obtain results when compared to the original k-NN, leading to cost savings and increased efficiency.

### Community Contribution

Finally, this study is significant for researchers and data scientists. Data scientists and researchers can quickly classify big data using the proposed solution without requiring high-performance computing resources. This democratizes access to big data analysis and supports wider community engagement in data science activities.

## 1.7 Study Timeline

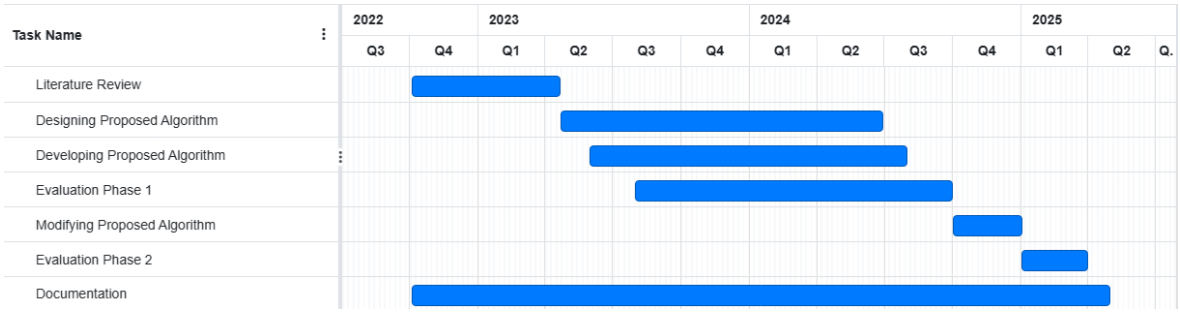


**Figure 1.1:** Study timeline gantt chart

## 1.8 Organisation of the Thesis

This thesis is structured into five main chapters. A brief explanation of each chapter is provided below:

Chapter 1 introduces the research problems encountered in this field, along with the research objectives, scope, and significance of the study.

Chapter 2 presents a literature review on the background of k-NN and the challenges it faces. This chapter also reviews improvements made by researchers in the past five years, followed by an examination of big data classification and enhancements to the k-NN algorithm for this context.

Chapter 3 presents the research methodology employed in this study, outlining the research flow, datasets used, data preparation techniques, a comprehensive explanation of the framework supporting the proposed algorithm, the experimental setup, system requirements, and the evaluation metrics used to assess the performance of the proposed algorithms.

Chapter 4 presents the experimental results and statistical analyses used to evaluate the proposed algorithms. This chapter also includes a comparison against the original k-NN and other related methods. Key findings are discussed to highlight essential insights.

Finally, Chapter 5 concludes the study, summarising the key findings and highlighting the limitations of the study. Suggestions for future research directions are also provided.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Overview

This chapter begins with an introduction to the k-nearest neighbour algorithm. Then, it continues to discuss problems faced by the k-NN algorithm, such as the curse of dimensionality and computational complexity, along with corresponding solutions, including dimensionality reduction and data reduction techniques. The section on solutions also includes other improvements to the k-NN algorithm that affect the general classification performance, such as the impact of the similarity distance metric and the value of *k* in k-NN. This chapter then introduces big data, and the problems faced when classifying big data datasets. This is followed by solution proposed by previous researcher to solve the problem, which includes using a distributed framework such as Apache Spark, Apache Hadoop, and Dask. Finally, this chapter concludes with a summary.

## 2.2 k-Nearest Neighbour

Cunningham and Delany (2021) state that the intuition behind Nearest Neighbour (NN) classification is straightforward: examples are classified based on the class of their nearest neighbours. Since it is often useful to consider more than one neighbour, the technique is more widely known as k-Nearest Neighbour (k-NN) classification, where *k* nearest neighbours is used to determine the class. NN is also referred to by various names, such as Memory-based classification, because the training data must be held in memory during runtime. NN, also known as a lazy learning technique, as induction is deferred until runtime. Finally, NN is referred to as example-based or case-based classification because it is based on the training examples only.

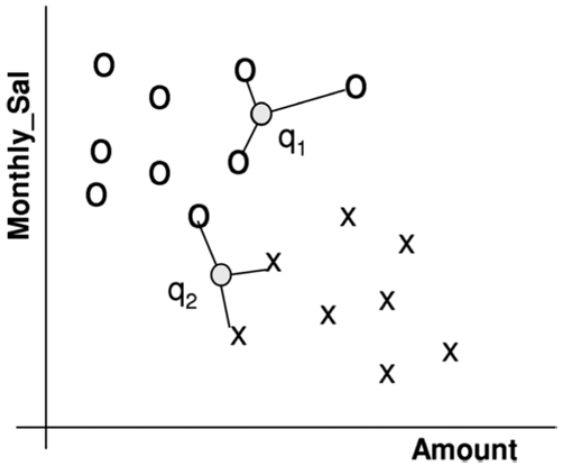


**Figure 2.1:** 3-Nearest Neighbour Classification in a 2D Feature Space (Monthly_Sal and Amount) adopted from Cunningham and Delany (2021)

The basic concept of k-NN is illustrated in Figure 2.1, which shows a 3-Nearest Neighbour classifier on a two-class problem in a two-dimensional feature space. According to Cunningham and Delany (2021), the decision for $q_1$ in this example is straightforward: all three of its nearest neighbours belong to class O; hence, it is classified as O. Since $q_2$ has two neighbours of class X and one of class O, the situation is more complicated than $q_1$. This situation can be resolved using simple majority or distance-weighted voting (Equation 2.3). K-NN classification comprises two stages: the first is determining the nearest neighbours, and the second is determines the class label based on those neighbours. Assume we have a training dataset D made up of $D = \{x_i | i = 1, \dots, n\}$, $where\ n = |D|$. The examples are characterised by a collection of features F, and any numeric features have been normalised to the range [0,1]. Each training example is assigned a class label $y_j \in Y$. The k-NN goal is to classify an unknown example q. For each $x_i \in D$, we can calculate the distance between q and $x_i$ as follows:

$$d(q, x_i) = \sum_{f \in F} w_f \delta(q_f, x_{if})$$ Equation 2.1

This expression represents a summation over all features in $F$, with $w_f$ representing the weight for each feature. Here, $q_f$ denotes the value of feature $f$ or the query instance $q$, and $x_{if}$ denotes the value of the feature $f$ for the *i-th* training instance $x_i$. There is a wide range of possible formulations for this distance metric; a basic version for continuous and discrete attributes is given below:

$$\delta(q_f, x_{if}) = \begin{cases} 0 & f \text{ discrete and } q_f = x_{if} \\ 1 & f \text{ discrete and } q_f ! = x_{if} \\ |q_f - x_{if}| & f \text{ continuous} \end{cases}$$ Equation 2.2

Based on this distance metric, the k-nearest neighbours are selected. These neighbours can then be used to establish the class of $q$ in a variety of ways. The simplest way is to assign the class of the nearest neighbours to the query instance. When determining the class of the query, it often makes sense to give the nearest neighbours more weight. One common technique is weighted voting, where each neighbour casts a vote on the class of the query case with votes weighted by the inverse of their distance to the query. This is a general and widely used method for improving classification accuracy.

$$Vote(y_j) = \sum_{i=1}^{k} \frac{1}{d(q, x_i)^p} 1(y_j, y_i)$$ Equation 2.3

Where $d(q,x_i)$ is the distance between the query instance $q$ and its *i-th* nearest neighbour $x_i$, as defined in Equation 2.1. The function $1(y_j, y_i)$ is an indicator function that returns 1 if the class label of neighbour $x_i$ matches $y_j$, and 0 otherwise. Thus, the vote assigned to class $y_j$ is the sum of the inverse distance contributions from all neighbours of that class.

The parameter $p$ is typically set to 1, but larger value can be used to further reduce the influnce of distance neighbours.

## 2.3 Challenges and Solutions for k-NN

The k-NN classification algorithm is performs well for basic classification tasks, but classifying datasets that are more complex and contain an enormous amount of data presents an entirely different set of challenges. As mentioned in Chapter 1, the challenges encountered during classification using k-NN are as follows: a) the requirement for a large memory space to store all instances and their distances to neighbouring instances for classification. b) Slower classification speed due to the search through all samples for classifying the test instance (Garcia et al., 2012; Kasemtaweechok & Suwannik, 2016). Both issues are caused by the large number of instances, which contributes to computational complexity, and the high number of dimensions, which leads to the curse of dimensionality. Another problem that needs to be addressed is the selection of the similarity distance measure and the choice of the $k$ parameter that plays an essential role in affecting the generalisation process of the k-NN algorithm. In the next section, we will go into the curse of dimensionality, computational complexity, selection of similarity distance, and the selection of the $k$ parameter in greater depth, along with potential solutions for each challenge.

### 2.3.1 Curse of Dimensionality

According to Bellman (1962), the curse of dimensionality indicates that the number of samples required to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the function's input variables (i.e., dimensionality). This concept helps in understanding the challenges associated with high-dimensional data (Beyer et al., 1999). The number of features or covariates in high-dimensional data can even exceed

the number of independent samples (Narisetty, 2020). Researchers often use dimensionality reduction techniques to deal with the curse of dimensionality. Dimensionality reduction refers to reducing the number of features in a resource-intensive computation without sacrificing important information. Reducing the number of features simplifies the problem space and improves computational efficiency. Dimensionality reduction has also been applied in image classification, where the goal is to extract essential features from images by mapping high-dimensional data to low-dimensional space and using the data in low-dimensional space to represent high-dimensional features (Zu et al., 2019). According to Jaruenpunyasak and Duangsoithong (2021), there are two main strategies for dimensionality reduction: feature selection and feature extraction.

Feature selection filters out irrelevant or redundant features from the original feature and retains a meaningful subset of the data (Jaruenpunyasak & Duangsoithong, 2021; Nobre & Neves, 2019; Solorio-Fernández et al., 2019). Examples of feature selection techniques include Information Gain (IG), Relief, Fisher Score, and LASSO (Nobre & Neves, 2019; Solorio-Fernández et al., 2019). In contrast, feature extraction generates a new feature dataset. Feature extraction uses techniques such as Principle Components Analysis (PCA) (Rani et al., 2019), linear discriminant analysis (LDA), and autoencoders to discover the dataset's prominent features or attributes. By reducing the number of features that characterise the dataset, feature extraction improves the speed of the classification process (Nobre & Neves, 2019; Solorio-Fernández et al., 2019).

### 2.3.2 Computational Complexity

Computational complexity is another issue that needs to be highlighted as it affects the overall classification performance when using the k-NN algorithm. Two types of

computational complexity exist: time complexity and space complexity. Time complexity is a metric for quantifying the computational time required to solve a problem. It is represented mathematically as a function of the input size. The standard method for describing the growth rate of an algorithm's time complexity with respect to input size is to use Big O notation. According to Cormen et al. (2022), this notation provides an upper bound on the growth rate of the algorithm's running time as the input size rises. This concept is illustrated by an algorithm with a time complexity of O($n$), indicates that the execution time increases linearly with the size of the input ($n$).

On the other hand, space complexity refers to the amount of memory an algorithm requires relative to the input size. Like time complexity, space complexity is often expressed using the Big O notation, which represents an upper bound on the growth of the algorithm's memory usage with respect to input size (Bharill et al., 2020). As a result, it is critical to design algorithms with low computational complexity to efficiently classify large volumes of data.

Implementing data reduction techniques can help address computational complexity by reducing the volume of data involved in the classification process. Data reduction techniques are classified into prototype selection (PS) and prototype generation (PG). Both methods seek to produce a representative dataset with a smaller training set size than the original and a similar or even greater classification accuracy for new incoming data (Garcia et al., 2012; Triguero et al., 2012). Prototype Selection algorithms choose prototypes (or instances) from the original training set, whereas Prototype Generation algorithms create new prototypes by aggregating similar training instances of the same class.

#### 2.3.2.1 Prototype Selection

An ideal prototype selection method preserves the original classification accuracy while effectively eliminating superfluous instances within the shortest possible execution time. However, designing an ideal prototype selection method is complicated, and approaches that achieve high reduction rates and classification accuracy tend to suffer from long execution times and are making them unsuitable to large datasets (Aslani & Seipel, 2021). Garcia et al. (2012) provide a comprehensive prototype selection (PS) overview. According to Garcia et al. (2012), the prototype selection process can be further divided into subtypes: direction of search, type of selection, and evaluation of the search process.

##### i. Direction of Search

Several directions can be taken when searching for a subset *S* of prototypes to keep from the training set TR. The type of search for PS includes incremental, decremental, batch, mixed, and fixed search methods.

**Incremental Search:** An incremental search starts with an empty subset *S* and adds each instance in TR to *S* if it meets specified criteria (Alexandropoulos et al., 2019; Cunha et al., 2023; Garcia et al., 2012).

**Decremental Search:** A decremental search starts with *S* equal to TR and removes instances from *S* that do not meet the selection criteria (Alexandropoulos et al., 2019; Cunha et al., 2023; Garcia et al., 2012).

**Batch Search:** In batch search, the decision to remove instances is made collectively, based on predefined criteria. All instances that meet the removal criteria are eliminated simultaneously from *S* (Cunha et al., 2023; Garcia et al., 2012).

**Mixed Search:** A mixed search begins with a preselected subset *S*, which may be chosen randomly or through an incremental or decremental process. Instances can then be iteratively added to or removed from *S* if they meet specific criteria (Alexandropoulos et al., 2019; Garcia et al., 2012).

**Fixed Search:** A fixed search is a subtype of the mixed search method. In this approach, the number of instance additions and removals is fixed. The final number of prototypes is determined at the start of the learning phase and remains unchanged throughout the process (Garcia et al., 2012).

ii. Type of Selection

The type of selection is primarily determined by the type of search conducted by PS algorithms. It refers to whether the algorithms attempt to preserve border points, central points, or another specific subset of instances.

**Condensation:** Condensation approaches emphasise preserving points near the decision boundaries, often referred to as border points. These methods maintain accuracy on the training set, but they may negatively affect generalisation accuracy on the test set. However, condensation approaches are often effective in reducing dataset size, as most data contains fewer border points than internal points (Alexandropoulos et al., 2019; Bi et al., 2021; Cunha et al., 2023; Garcia et al., 2012; Gong et al., 2021; Jiménez et al., 2022).

**Edition:** Edition approaches aim to eliminate border points. They focus on removing noise or points that are inconsistent with their neighbours. This process smooths decision boundaries by removing nearby border points. However, these algorithms typically retain internal points, even if those points do not significantly affect the decision boundaries. The

result is often improved generalisation accuracy for test data, though the reduction rate is generally lower (Alexandropoulos et al., 2019; Bi et al., 2021; Cunha et al., 2023; Garcia et al., 2012; Gong et al., 2021; Jiménez et al., 2022).

**Hybrid:** Hybrid approaches aim to find the smallest subset S that maintains or improves generalisation accuracy on test data. By combining criteria from the previous two strategies, these methods allow for the removal of both internal and border points. The k-NN classifier is particularly well-suited to these methods and often achieves significant improvements even when using a reduced subset of instances (Alexandropoulos et al., 2019; Bi et al., 2021; Cunha et al., 2023; Garcia et al., 2012; Gong et al., 2021; Jiménez et al., 2022).

iii. Evaluation of Search

k-NN is a straightforward technique for guiding the search process in a prototype selection (PS) algorithm. The primary goal is to generate predictions for non-definitive selections and facilitate comparisons between different selection outcomes. This capability is crucial for determining the quality criteria used in the evaluation process, which are further categorised as follows:

**Filter:** The filter approach applies the k-NN rule to partial subsets of the data to establish criteria for adding or removing instances. This method does not use leave-one-out validation. Although using subsets of the training data improves the computational efficiency of these algorithms, it may not necessarily enhance classification accuracy (Alexandropoulos et al., 2019; Garcia et al., 2012; Jiménez et al., 2022).

**Wrapper:** The wrapper approach applies the k-NN rule to the entire training set using the leave-one-out validation scheme. This combination provides a robust estimate of generalisation accuracy, contributing to improved performance on test data. However, it is important to note that each decision requires a complete execution of the k-NN rule across the training set, which can lead to high computational costs during the learning phase (Alexandropoulos et al., 2019; Garcia et al., 2012; Jiménez et al., 2022).

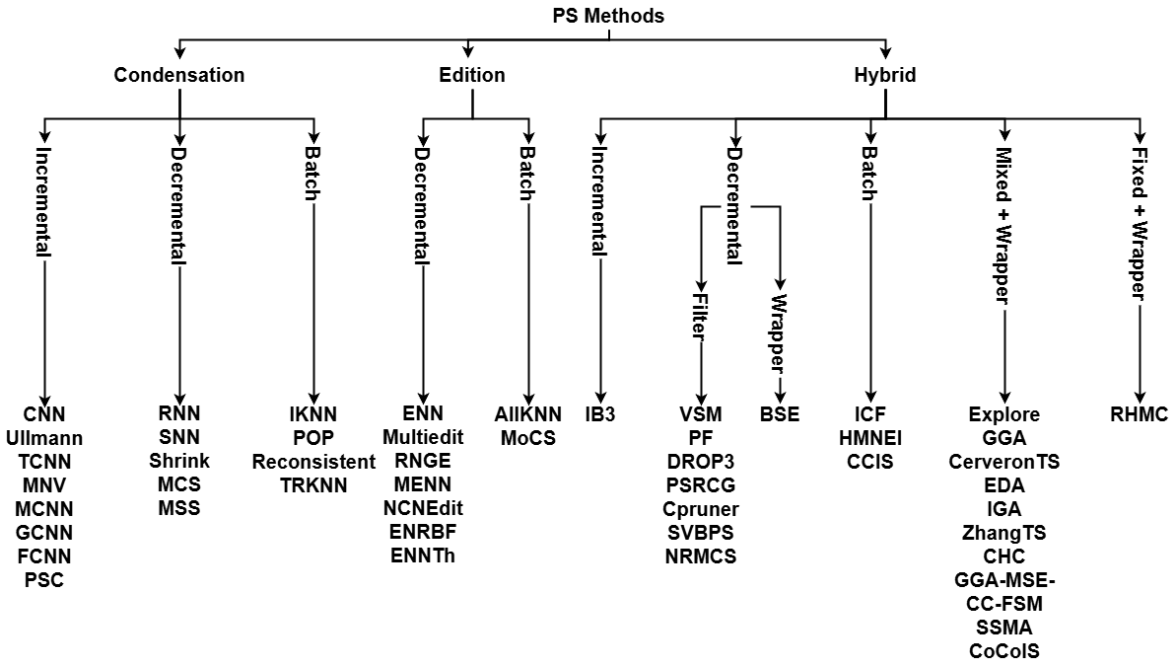


**Figure 2.2:** Prototype Selection Taxanomy adopted from Garcia et al. (2012)

Figure 2.2 illustrates the prototype selection taxonomy proposed by Garcia et al. (2012). Jankowski and Orliński (2019) proposed a method that uses trust-border instances to retain only trustworthy instances located at class boundaries. The algorithm also employs a tree structure to improve efficiency.

Sisodia and Sisodia (2022) introduced a quad-division prototype selection method based on k-NN (QDPSKNN), which addresses imbalanced class distributions by dividing

the majority class training data into four equal parts and applying k-NN to each group to select promising prototypes in numbers equivalent to the minority class instances.

Rico-Juan et al. (2019) proposed an extension to rank-based prototype selection by introducing a new family of hybrid PS algorithms known as the ranked method. Rank-based methods (Hernandez-Leal et al., 2013; Rico-Juan & Iñesta, 2012) are designed to sort the prototypes in the training set according to their expected relevance to the 1-NN classifier. Prototypes are then selected based on this relevance order, with the final training set size controlled by a user-defined parameter.

The authors proposed two extensions to the ranked method. The first extension performs a class-wise selection of prototypes rather than selecting instances globally. The second extension introduces a tuning parameter to set the per-class probability density to be maintained.

Wilson and Martinez (2000) proposed Decremental Reduction Optimisation Procedure 3 (DROP3), an advanced prototype selection algorithm designed to enhance accuracy and reduce dataset size in instance-based learning classifiers, particularly k-Nearest Neighbour (k-NN). The DROP3 algorithm operates through three main steps. First, it applies a noise-filtering process using the Edited Nearest Neighbour (ENN) rule, removing any instance misclassified by its k-Nearest Neighbours. Next, it sorts the remaining instances based on their distance to the nearest enemy, which refers to the closest instance belonging to a different class, prioritising those farthest from decision boundaries for potential removal. Finally, DROP3 incrementally removes instances, beginning with those farthest from their nearest enemy, provided their removal does not negatively affect the classification accuracy

of neighbouring instances. This approach effectively retains essential boundary instances while discarding redundant interior points, resulting in a compact yet accurate training set.

ATISA1, introduced by Cavalcanti et al. (2013), is an incremental prototype selection method that employs an adaptive threshold to determine instance relevance. It first filters noisy instances using the Edited Nearest Neighbour (ENN) method. Each remaining instance is then assigned an individual threshold, calculated as the distance to its nearest enemy, which is the closest instance belonging to a different class. This threshold defines a hyperspherical coverage region around each instance. During the selection phase, ATISA1 evaluates instances by adding them to the reduced dataset if they lie outside the coverage area defined by previously selected instances, or if their removal would result in misclassification. This approach ensures the preservation of essential instances, whether at the decision boundaries or within the interior of the feature space, resulting in efficient reduction and stable or improved classification performance compared to existing techniques.

Li and Dai (2022) proposed a fast prototype selection algorithm based on classification neighbourhood boundary approximation (PSNB). This method comprises two fundamental processes: selection and update. The selection process is the core prototype learning phase, responsible for determining which patterns are retained as prototypes. The second process, which is similar to the Edited Nearest Neighbour (ENN) method, continuously maintains the quality of the prototype set by eliminating prototypes that are unsuitable for optimising the classification boundary or that contribute minimal interference.

Suárez et al. (2021) proposed the Condensed Neighbourhood Components Analysis (CNCA) method. This approach selects samples iteratively during the optimisation process,

allowing the linear transformation and the set of prototypes to be learned simultaneously. In cases where certain classes are highly underrepresented, the algorithm allows all samples from those classes to be forcibly included in the condensed sample set and excluded from removal. This mechanism is a key feature of the proposed method, as it ensures that under sampling only affects the majority classes. Additionally, when classes are well separated, the method is capable of selecting a single high-quality prototype per class. Finally, the method supports an arbitrary number of classes, making it applicable to multi-class problems and not limited to binary class imbalance scenarios.

Cavalcanti and Soares (2020) introduce the Ranking-based Instance Selection (RIS) algorithm for instance-based classification tasks. RIS operates through two primary phases: ranking and selection. In the ranking phase, RIS calculates a score for each instance based on its relationship with all other instances in the training set, reflecting its importance or "difficulty" in classification. Instances surrounded by others from the same class receive higher scores, indicating safer and more homogeneous areas, whereas instances close to class boundaries, which are potentially noisy or ambiguous, receive lower scores. In the selection phase, RIS retains only relevant instances, meaning those that are neither redundant nor noisy, by prioritising higher-scoring instances and removing those that fall within hyperspheres centred around higher-scored instances of the same class. RIS is available in three variants (RIS1, RIS2, RIS3), which differ in how scores are scaled and how instances are selected, balancing accuracy and data reduction.

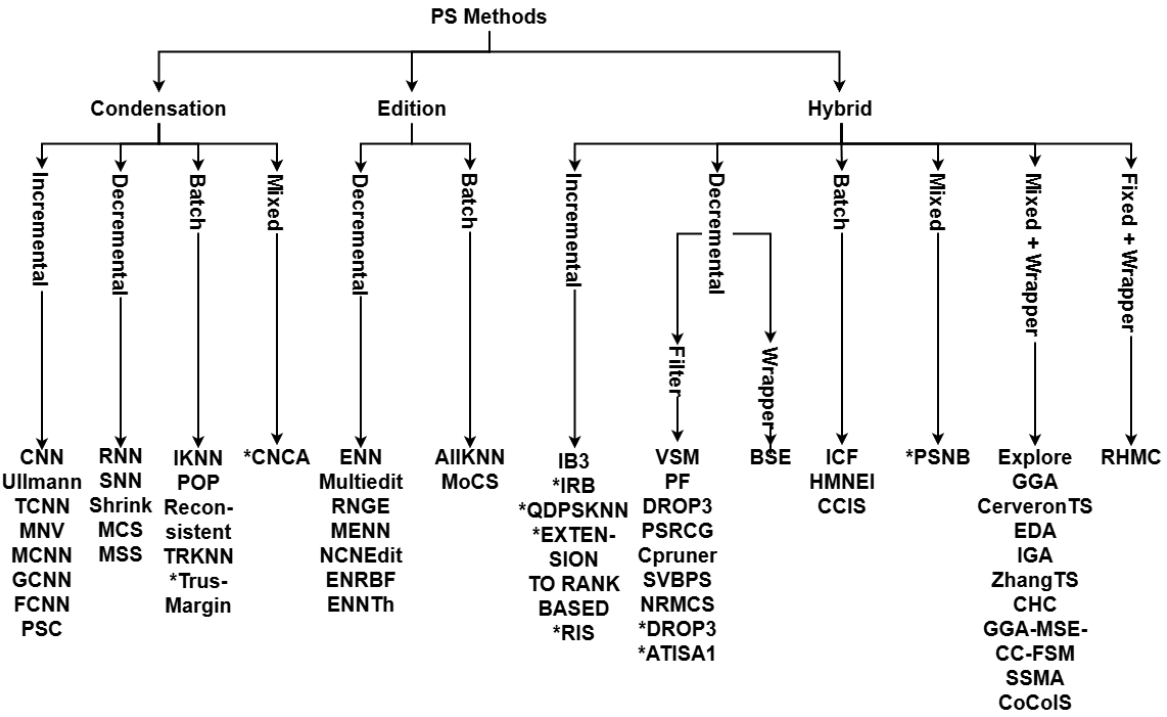


**Figure 2.3:** Updated Prototype Selection Taxonomy adapted from Garcia et al. (2012)

Figure 2.3 shows the updated prototype selection taxonomy based on previous research by Garcia et al. (2012) and refined through our literature review, incorporating previously proposed methods such as PSNB, QDPSKNN, RIS and a lot more. The highlighted method is denoted with the '*' symbol."Prototype Generation

The prototype generation (PG) family represents one of the most competitive alternatives due to its notable reduction capabilities compared to other data reduction (DR) strategies (Escalante et al., 2015; Valero-Mas et al., 2023). In a broad sense, PG derives an alternative reference set for the classifier by performing various selection and merging operations on elements of the initial corpus, guided by specific heuristics (Triguero et al., 2012). Generally, PG achieves higher performance rates than prototype selection (PS), but its application is often limited by the type of data representation used (Castellanos et al., 2021).

A study by Triguero et al. (2012) presents a detailed prototype generation (PG) technique. The authors also highlight several essential criteria for PG. The first is the type of reduction, which parallels that of prototype selection (PS) and includes incremental, decremental, fixed, and mixed approaches. The second is the resulting generation, which corresponds to the selection types in PS: condensation, edition, and hybrid. The most critical criterion is the generation mechanism, followed by the evaluation of the search process.

iv. Type of Reduction

The prototype generation (PG) approach aims to identify a reduced set of prototypes, denoted as the reduced set (TG), that effectively represents the training set (TR). The types of reduction in PG include incremental, decremental, fixed, and mixed reduction strategies.

**Incremental Reduction:** In an incremental reduction approach, the process begins with an empty reduced set (TG) or by selecting representative prototypes from each class. Subsequently, a sequence of prototype additions or modifications is performed. Compared to non-incremental algorithms, a key advantage of this approach is its potential for high speed and lower storage requirements during the learning phase. Moreover, this method allows the algorithm to determine the appropriate number of prototypes for each dataset. However, when a large number of prototypes is required to adapt to TR, there is a risk of overfitting (Triguero et al., 2012).

**Decremental Reduction:** In decremental reduction, the process starts with TG equal to TR, after which the algorithm reduces or modifies the set. This can be achieved through various techniques, such as merging, relocating, or removing prototypes, as well as reassigning class labels. A key advantage of decremental schemes is that they have access

to all training instances during decision-making. However, a significant drawback is their typically high computational cost (Triguero et al., 2012).

**Fixed Reduction:** Fixed reduction is a common approach in PG. Based on a user-defined parameter specifying the percentage of TR to retain, these algorithms calculate the final number of prototypes in TG. The main limitation of this method is its dependence on user-specified parameters and the specific characteristics of the dataset. Nonetheless, the primary objective of these strategies is to enhance classification accuracy (Triguero et al., 2012).

**Mixed Reduction:** In a mixed reduction approach, the process begins by generating a preselected subset TG, either through random selection of a fixed set or by applying a prototype selection (PS) method. Prototype additions, modifications, and removals are then performed within TG. This strategy combines the benefits of the previously mentioned methods, allowing multiple adjustments to address the limitations of fixed reduction. However, these methods are often prone to overfitting and can incur high computational costs (Triguero et al., 2012).

v. Resulting Generation Set

This component refers to the output set generated by the technique, indicating whether the final set preserves border points, central points, or a combination of both.

**Condensation:** Condensation methods include techniques that produce a reduced set of prototypes located near decision boundaries, often referred to as border points. The rationale for retaining border points is that internal points have minimal influence on decision boundaries, and their removal typically has a limited effect on classification

accuracy. These methods maintain high accuracy on the training set but may negatively affect generalisation accuracy on the test set. Nonetheless, because most datasets contain fewer border points than internal points, condensation methods are often effective for dataset size reduction (Triguero et al., 2012).

**Edition:** Edition methods, in contrast, aim to eliminate or adjust border points. They target noisy instances or those that do not align with their nearest neighbours (NNs), resulting in smoother decision boundaries. However, these algorithms generally retain internal points, even if such instances contribute little to decision boundary formation. The primary benefit of edition methods lies in improved generalisation accuracy on test data, although they tend to achieve lower reduction rates (Triguero et al., 2012).

**Hybrid:** Hybrid methods aim to identify the smallest subset TG that preserves or enhances generalisation accuracy on test data. To achieve this, they allow modifications to both internal and border points based on algorithm-specific criteria. The NN classifier is particularly well-suited to these methods, often yielding significant improvements even with a moderately reduced prototype set (Triguero et al., 2012).

vi. Generation Mechanisms

This component describes the various mechanisms used in the literature to construct the final TG set.

**Class Re-labelling:** This generation mechanism involves modifying the class labels of TR samples suspected of being mislabelled or belonging to incorrect classes. Its aim is to address imperfections in the training set, such as noisy, mislabelled, or atypical instances.

The primary benefit is improved generalisation accuracy on test data, while the reduction rate typically remains unchanged (Castellanos et al., 2021; Triguero et al., 2012).

**Centroid-Based:** Centroid-based techniques involve generating artificial prototypes by merging similar instances. This is typically done by calculating the average attribute values of grouped samples, known as centroids. The primary goal of these methods is to identify and select a representative set of instances. While they can achieve significant reduction rates, they may also lead to reduced classification accuracy (Castellanos et al., 2021; Rezaei & Nezamabadi-pour, 2022; Triguero et al., 2012).

**Space Splitting:** Space-splitting techniques apply heuristics to partition the feature space and generate new prototypes. The idea is to divide TR into distinct regions, each of which is then replaced with a representative instance to define decision boundaries similar to those in the original set. These methods operate at the space level by distinguishing regions through partitioning, whereas centroid-based methods function at the data level by selecting representative instances. The reduction potential of space-splitting techniques often depends on the number of regions required to adequately describe the data (Castellanos et al., 2021; Rezaei & Nezamabadi-pour, 2022; Triguero et al., 2012).

**Positioning Adjustment:** This approach focuses on adjusting the positions of selected prototypes from the initial set, usually through an optimisation process. The new positions are determined in an m-dimensional space by adding or subtracting specific quantities from the prototypes' attribute values. This mechanism is often combined with either fixed or mixed reduction strategies (Castellanos et al., 2021; Rezaei & Nezamabadi-pour, 2022; Triguero et al., 2012).

vii. Evaluation of Search

The nearest neighbour (NN) algorithm serves as an effective heuristic to guide the search process in the prototype generation (PG) method. Selections made using this heuristic must be accompanied by an evaluation measure that enables comparison among multiple alternatives. The evaluation criterion depends on whether or not NN is applied during the evaluation process.

**Filter:** Filter methods are characterised by the absence of the NN rule during the evaluation phase. The reduced set is selected using alternative heuristics. While these methods are generally faster than NN-based approaches, they may result in lower accuracy and reduced overall performance (Triguero et al., 2012).

**Semi-wrapper:** In semi-wrapper methods, NN is applied to partial data to establish decision criteria. Consequently, NN performance is evaluated over localised subsets of the data, primarily including prototypes that are most influential in the decision-making process. This approach offers a compromise between efficiency and accuracy (Triguero et al., 2012).

**Wrapper:** In wrapper methods, the NN rule fully guides the search process, using the entire training set along with the leave-one-out validation scheme. This combination yields a robust estimate of generalisation accuracy, often resulting in improved test performance. However, each decision requires a full computation of the NN rule across the entire training set, making the evaluation phase computationally expensive (Triguero et al., 2012).

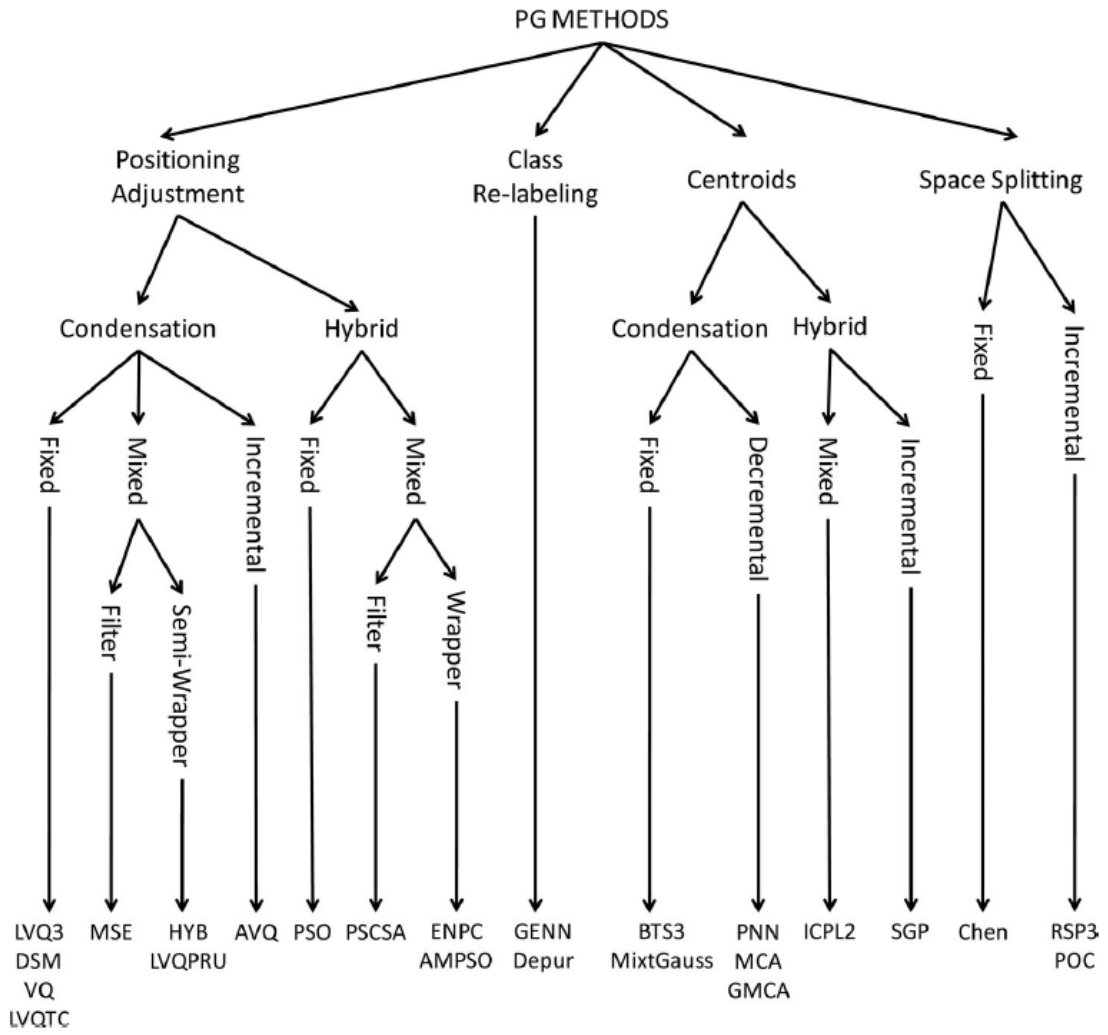


**Figure 2.4:** Prototype Generation Hierarchy adopted from Triguero et al. (2012)

Figure 2.4 illustrate prototype generation hierarchy. Due to the importance of PG for efficient k-NN-based classification, a considerable amount of research has been devoted to proposing novel strategies and improving the existing ones (Nanni & Lumini, 2011; Valero-Mas et al., 2023).

Ougiaroglou et al. (2021) Ougiaroglou et al. (2021) proposed Multilabel Reduction through Homogeneous Clustering (MRHC), a multi-label extension of the RHC method for training data reduction using prototype generation. As illustrated in Figure 2.5, MRHC uses label representatives as initial centroids for k-means clustering. Homogeneous clusters are

retained as prototypes, while non-homogeneous clusters are recursively subdivided. This process reduces the dataset to a smaller set of representative prototypes; for example, 16 original instances are reduced to 5 prototypes. MRHC highlights the potential of clustering-based approaches for efficient reduction in multi-label classification problems.

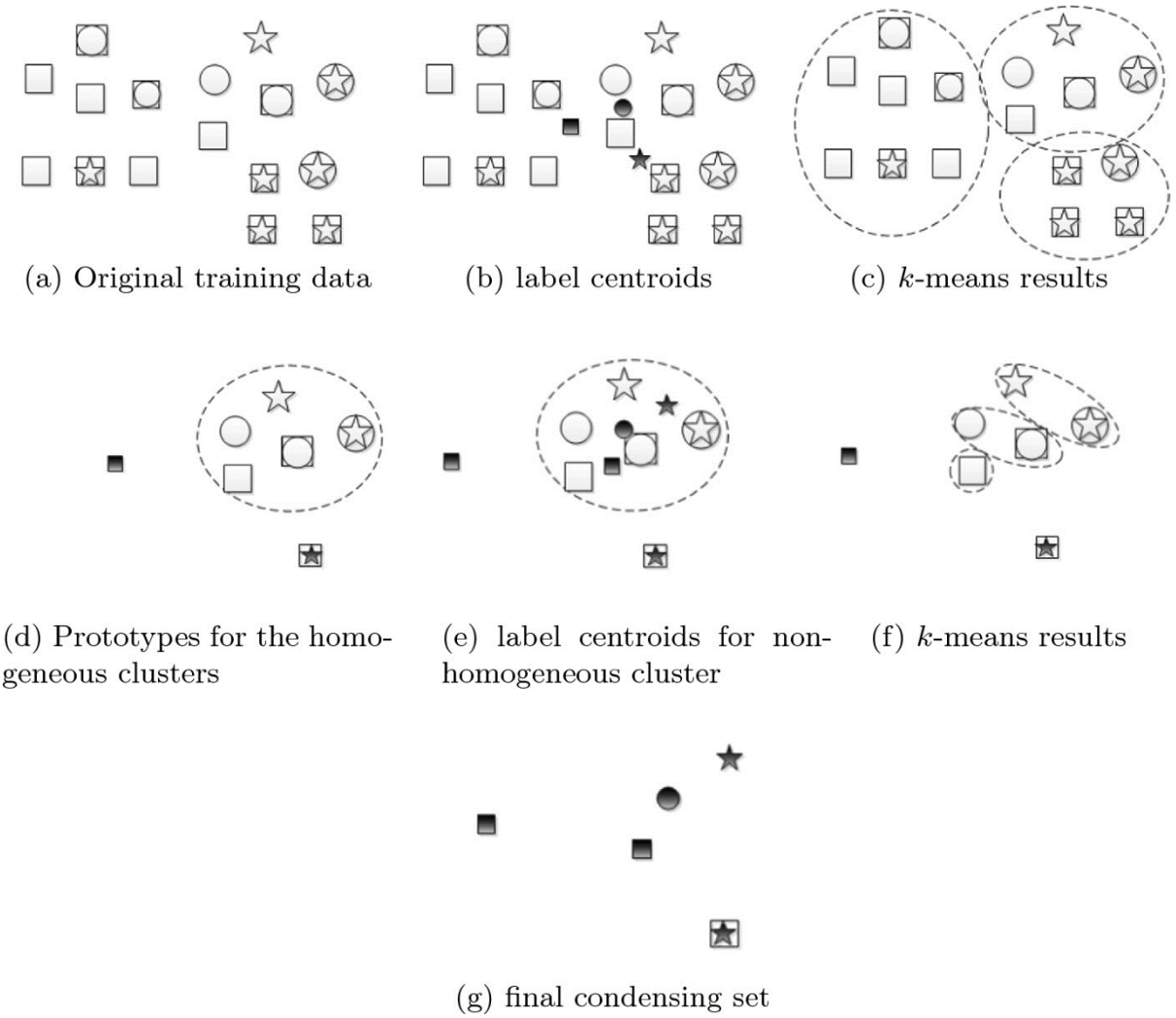


**Figure 2.5:** MRHC Execution Example adopted from (Ougiaroglou et al., 2021)

Valero-Mas et al. (2023) proposed four adaptations of the earlier multi-label method. The first adaptation is the Multilabel Chen algorithm (MChen). Instead of merging elements from different classes, the authors implemented a policy from (Ougiaroglou et al., 2021) for

multilabel reduction through homogeneous clustering (MRHC), where the resulting prototype retains the labels present in at least half of the cluster instances. The second adaptation is Multilabel Reduction through Space Partitioning 1 (MRSP1), based on the original RSP1 method. In this adaptation, prototype merging is applied such that for each cluster, one prototype is generated for each class present within it. The authors modify this by applying a label-set-based approach, where each unique label set is treated as a separate class. Instances with the same label set are merged and assigned to that set. The third adaptation, Multilabel Reduction through Space Partitioning 2 (MRSP2), is derived from the RSP2 method and extends the concept of class overlap to the multilabel space. In this adaptation, each label set is again treated as a distinct class. The degree of overlap is calculated as the ratio between the average distance of instances belonging to different label sets and the average distance of instances within the same label set. The final adaptation is Multilabel Reduction through Space Partitioning 3 (MRSP3), an extension of the RSP3 method. To estimate the number of clusters automatically in MRSP3, the authors generalise RSP3's cluster homogeneity criterion. They apply the condition from (Ougiaroglou et al., 2021), which considers a multilabel cluster homogeneous if there is at least one common label among all prototypes in the cluster.

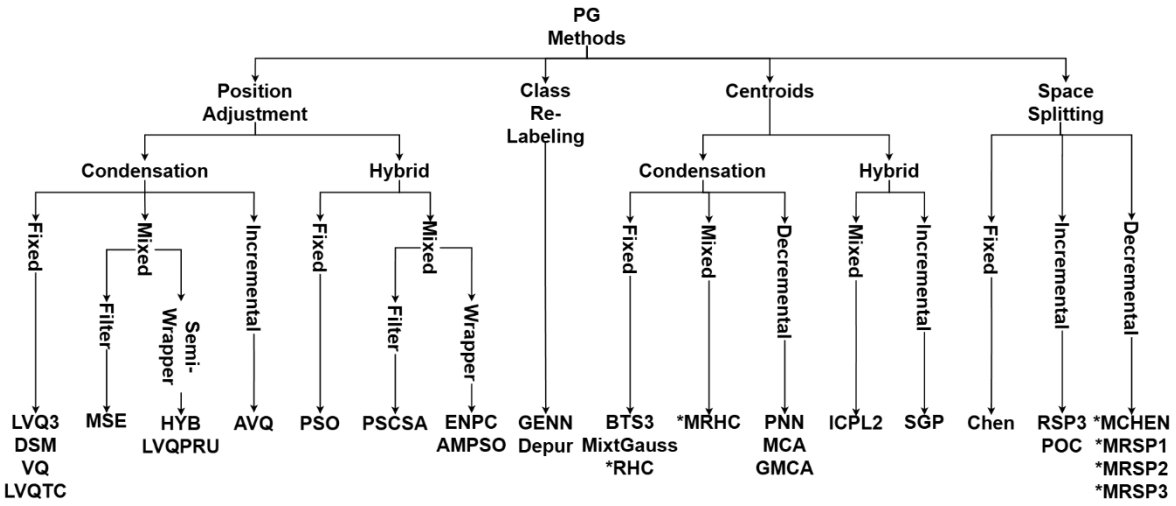


**Figure 2.6:** Prototype Generation Taxanomy Update adapted from Triguero et al. (2012)

Figure 2.6 shows the updated prototype generation taxonomy based on previous research by Triguero et al. (2012) , including the current PG method. The current method is marked with the "*" symbol.

### 2.3.3 Selection of Parameter *k*

The selection of the parameter *k* significantly influences the overall performance of the k-NN algorithm (Rodriguez-Martinez & Torres-Sospedra, 2021). The value of *k* determines how many nearest neighbours the algorithm considers when classifying a test instance. Zhang (2021) outlines several methods for selecting an appropriate *k* for k-NN. The first approach assumes that users will manually specify a value of *k* when applying k-NN classification. However, this assumption places a considerable burden on users, as determining an effective *k* is often non-trivial.

The second approach involves utilising all samples in the training dataset to assist with *k* selection. This method addresses the challenge by relying on the complete set of training instances. There are three main strategies for computing *k* under this approach:

i. **Global *k* assignment:** A single *k* value is assigned to the entire sample space. Liu et al. (2010) demonstrated that this approach may be ineffective, as many test samples are unsuitable for a fixed *k*. Using a single *k* often leads to poor prediction performance (Zhang, Cheng, et al., 2018).

ii. **Dynamic *k* assignment:** To overcome the limitations of a fixed *k*, Zhang, Cheng, et al. (2018) proposed the Sparse k-NN (S-kNN) approach. Instead of assigning the same *k* value to all test samples, this method dynamically assigns different *k* values based on the data distribution. It employs example-driven *k* parameter computation using a sparse coefficient matrix to reconstruct correlations between training and test samples. The key advantage lies in leveraging this sparse correlation to determine an optimal *k* for each test sample.

iii. **Approximate *k* estimation:** The third strategy approximates the optimal *k* for a test instance by assigning it the optimal *k* value of its nearest neighbour. Zhang, Li, et al. (2018) addressed the high computational cost of assigning different *k* values by introducing the *K* tree, a structure designed to efficiently locate the nearest neighbour and retrieve its corresponding *k* value.

In summary, while selecting an appropriate *k* is essential for accurate classification, dynamic and approximate methods generally offer more adaptable and efficient solutions than a fixed global *k*. However, although dynamic approaches is the best in addressing the limitations of a single *k*, their computational complexity remains high, and no single selection method has been universally recognized as the best.

### 2.3.4 Similarity Distance

As previously discussed, the distance between two points is fundamental in determining the class of a test instance. Similarity distance plays a crucial role in k-NN classification, as it defines the relationship between the test data and the training instances. From a scientific and mathematical perspective, distance is defined as the quantitative measure of how far apart two items are. The terms dissimilarity and distance are often used interchangeably. Distance measures that satisfy the properties of a metric are referred to as metric distances, while those that do not are sometimes called divergences. The term similarity is sometimes used synonymously with proximity, and similarity measures are frequently referred to as similarity coefficients (Cha, 2007). Cha (2007) lists the most commonly used similarity and distance functions in k-NN classification. However, many performance enhancements to the standard k-NN algorithm rely on selecting an appropriate distance metric (Cunningham & Delany, 2021). Similarity distance is typically used to compute the similarity or dissimilarity between two data points. In this study, we extend its application to prototype reduction, where similarity distance is used not only for classification but also as part of the reduction mechanism in the proposed method.

Similarity distances such as Euclidean (Equation 2.4) and Manhattan (Equation 2.5) are among the most commonly used distance measures in k-NN classification (Cunningham & Delany, 2021). Kumar and Singla (2021) found that using the Euclidean distance metric within the k-NN algorithm yields superior results compared to the Manhattan distance. Meanwhile, Arora et al. (2022) discovered that using the Canberra distance (Equation 2.6) produces more consistent results across most datasets when applied to k-NN classification. However, no study in the past five years has specifically investigated the use of similarity distance for reducing the number of prototypes.

$$d_{Euc} = \sqrt{\sum_{i=1}^{d} |P_i - Q_i|^2} \quad \text{Equation 2.4}$$

$$d_{Man} = \sum_{i=1}^{d} |P_i - Q_i| \quad \text{Equation 2.5}$$

$$d_{Can} \sum_{i=1}^{d} \frac{|P_i - Q_i|}{P_i + Q_i} \quad \text{Equation 2.6}$$

where:

- $P = (P_1, P_2, \ldots, P_3)$ represents the feature vector of the first instance,
- $Q = (Q_1, Q_2, \ldots, Q_3)$ represents the feature vector of the second instance,
- $d$ is the distance,
- $P_i$ and $Q_i$ are the values of the *i-th* ranges from 1 to n of $P$ and $Q$, respectively

Alfeilat et al. (2019) classified similarity and dissimilarity distance measures into eight major families, comprising 54 distinct metrics. Building on this, Muniswamaiah et al. (2023) expanded the taxonomy to include 74 different distance measures. These metrics vary significantly in their mathematical formulations, ranging from basic count-based differences to more complex correlation-based computations. The choice of distance measures can significantly affect the performance of distance-based instance selection algorithms, particularly when it aligns with the characteristics of the dataset. The k-NN classifier, in particular, is highly sensitive to the selected distance metric, with substantial differences in classification accuracy observed across various measures. Some distance functions demonstrate greater robustness to noise than others. For example, the Hassanat distance consistently yielded superior results across multiple datasets (Alfeilat et al., 2019). Additionally, Alfeilat et al. (2019) noted that distance functions from the same family, with similar mathematical structures, often produce comparable classification outcomes.

Empirical findings by Fuangkhon (2021) further underscore the importance of selecting distance functions based on data types. Specifically, Chebyshev, Cosine, and Minkowski distances were found to be effective for integer-type data in terms of data reduction performance. For categorical data, the Minkowski distance was recommended, while the Jaccard distance was most effective for real-valued data. These insights reinforce the importance of tailoring the distance metric to the underlying data type in order to optimise classification outcomes. However, these recommendations are not universally applicable, as prior studies have consistently demonstrated that no single distance measure is optimal across all datasets (Alfeilat et al., 2019; Fuangkhon, 2021; Hu et al., 2016; Kalra et al., 2022; Muniswamaiah et al., 2023).

## 2.4 Big Data

In general, big data refers to extremely large and complex datasets that are difficult to store, manage, and process using traditional data processing tools (Liu, 2015). Notably, big data is commonly characterised by three dimensions, known as the 3Vs (Laney, 2001). The first is volume, which refers to the vast amount of data generated from various sources, presenting significant challenges for storage and analysis (Chi et al., 2016; Li et al., 2015). The second is variety, as big data is typically produced in multiple types and formats. It may arrive already partially integrated or require additional combination and transformation for a specific use. As a result, considerable effort is often needed to handle diverse data types and complex structures. The third dimension is velocity, which denotes the high speed at which data is generated and transmitted from numerous sources (Li et al., 2015). In recent years, additional dimensions have been proposed. According to Gautam and Chatterjee (2020), big data is now frequently described using seven key dimensions, commonly referred to as the 7Vs, as illustrated in Figure 2.7.

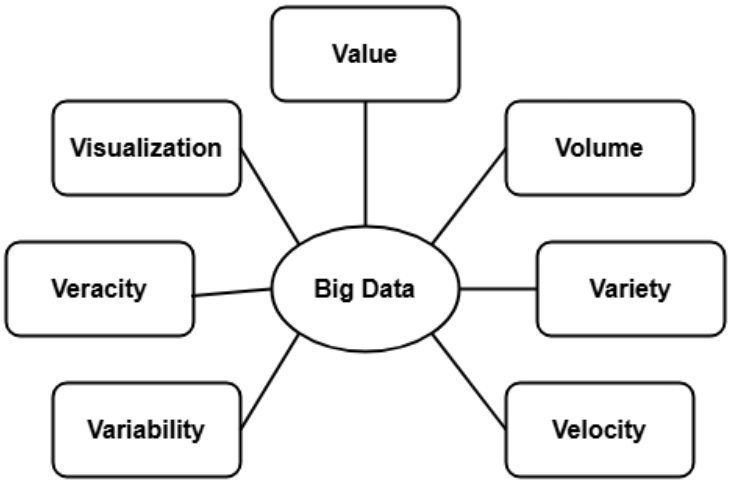


**Figure 2.7:** Seven V's of Big Data adopted from Gautam and Chatterjee (2020)

**Volume:** This dimension refers to the massive amount of data generated and stored, which has evolved from gigabytes (GB) to zettabytes (ZB) and even yottabytes (YB).

**Variety:** This refers to the diverse forms of data, including structured, unstructured, semi-structured, and mixed data formats.

**Velocity:** This describes the speed at which data is generated, transmitted, and processed.

**Variability:** In contrast to variety, variability focuses on understanding and interpreting the meaning of raw data within different contexts. Data can exist in multiple states, such as clear, consistent, connected, or ambiguous.

**Veracity:** This dimension refers to the reliability, accuracy, and trustworthiness of data.

**Visualisation:** Visualisation is crucial for effectively presenting large and complex datasets. It involves using visual tools such as charts and graphs to convey information more clearly than traditional text-heavy formats like spreadsheets or documents.

**Value:** After addressing the preceding dimensions, the ultimate goal is to ensure that data, when properly processed and analysed, delivers meaningful and actionable value.

As previously discussed, the key challenges in classifying big data using the k-NN algorithm are: (a) the need for large memory space to store all instances and their distances to neighbouring instances for classification, and (b) slower classification speed due to the time required to search all samples when classifying a test instance (Garcia et al., 2012; Kasemtaweechok & Suwannik, 2016). Both problems are closely associated with the curse of dimensionality and computational complexity, which were addressed in the previous subsection. In the context of big data, the methods for addressing these issues remain the same: dimensionality reduction to mitigate the curse of dimensionality, and data reduction to reduce computational complexity. However, when dealing with big data, additional tools are required to efficiently process large-scale datasets. Distributed computing frameworks and parallelisation tools such as Apache Spark, Apache Hadoop, and Dask are commonly used by researchers to manage the demands of big data processing. The following subsection discusses solutions proposed in previous studies for addressing the curse of dimensionality, computational complexity, and the integration of distributed and parallel computing frameworks to facilitate big data classification.

### 2.4.1 Curse of Dimensionality

In big data, the curse of dimensionality is a common challenge due to the high number of dimensions or features typically present in such datasets. To address this issue,

researchers often apply dimensionality reduction techniques, which include feature selection and feature extraction.

#### 2.4.1.1 Feature Selection

Research by Rong et al. (2019) highlights several limitations of traditional feature selection methods when applied to big data. The first issue is that these methods typically require extensive processing time, making it difficult for the learning process to keep pace with the rapid changes in characteristic of big data environments. Secondly, big data often contains a large number of irrelevant or redundant features, as well as varying degrees and types of noise. This significantly increases the complexity of the feature selection task. Finally, the presence of unreliable or falsified data, which may arise from diverse acquisition methods or data loss, further complicates effective feature selection.

In recent years, Su et al. (2021) introduced the Distributed Rough Evidential k-NN (rough EK-NN) algorithm for large sample sizes and/or high-dimensional data, which performs feature selection and classification simultaneously. The rough EK-NN algorithm reduces redundant input features and, consequently, the classification complexity of the decision boundary. The study also applied the Apache Spark framework to handle big data. Spark parallelises the required computations using a distributed data structure called Resilient Distributed Datasets (RDDs), which allow data stored in main memory to persist and be reused.

#### 2.4.1.2 Feature Extraction

Feature extraction is less commonly used in big data applications compared to feature selection. Nevertheless, some studies have employed feature extraction techniques in big data contexts. For example, Ma et al. (2020) applied Recursive Feature Elimination (RFE)

to perform feature extraction. RFE is a greedy algorithm based on feature ranking. It begins with the complete set of features and iteratively removes the least relevant ones according to a specified feature ranking criterion, continuing this process until the desired number of features is reached. Feature relevance is determined by the coefficients or feature importance scores provided by the base model.

### 2.4.2 Computation Complexity

Nevertheless, dimensionality reduction offers limited benefits when the dataset is very large, as time complexity depends more on the size of the training data than on the number of features (Wang et al., 2019).

One major limitation in classifying big data is that not all users have access to sufficient computing resources to analyse the dataset comprehensively. Therefore, data reduction becomes essential for extracting meaningful information from the data. In cases where the number of instances exceeds the number of dimensions, the required computational time is $O(np^2)$. This level of time complexity is too high for big data applications and may even exceed the capacity of available computing infrastructure. To overcome this limitation, data reduction techniques must be employed (Wang et al., 2018). As mentioned earlier, data reduction is typically achieved through two approaches: prototype selection and prototype generation.

#### 2.4.2.1 Prototype Selection

Prototype selection has demonstrated strong potential in alleviating the computational burden associated with the training phase when working with large datasets (Nalepa & Kawulok, 2019). The concept of prototype selection is based on the idea that

training data are not used to describe classes explicitly, but rather to support accurate class separation (Aslani & Seipel, 2021).

Wang et al. (2019) implemented a data reduction method that selects representative instances from each data cluster. The selection approach is based on the initial centre selection technique from the k-means++ algorithm, proposed by (Arthur & Vassilvitskii, 2007). Prototype selection algorithms, a subset of data reduction methods, require optimisation under Apache Spark to improve their suitability for practical applications (Gong et al., 2021). Gong et al. (2021) introduced two such methods, Evidential Instance Selection (EIS) and Evidential Instance Selection with Apache Spark (EIS-AS), which implement prototype selection using Spark and are specifically designed for handling big data. The primary distinction between EIS and EIS-AS lies in their use of Apache Spark and the procedure for identifying the k-Nearest Neighbours for each instance. Malhat et al. (2020) proposed two additional prototype selection algorithms: Global Density-based Instance Selection (GDIS) and Enhanced Global Density-based Instance Selection (EGDIS). Both methods utilise density measures to identify which prototypes should be retained in the reduced set. The main advantage of the GDIS algorithm is its ability to preserve instances that represent the boundaries between different classes in the training set, which positively influences classification accuracy. However, maintaining these boundary instances results in a lower reduction rate. To address this limitation, the authors developed the EGDIS algorithm to improve the reduction rate achieved by GDIS.

Aslani and Seipel (2020) proposed DR.LSH, which is based on the premise that certain redundant samples do not significantly contribute to the representation of each class. Its primary goal is to identify and remove such samples from large datasets efficiently.

DR.LSH accelerates the process of detecting redundant samples by leveraging the concept of local sensitivity. It identifies redundancy by measuring similarity, based on partitioning the data space into multiple buckets across several layers. Redundant samples, relative to a given sample (A), are those whose similarity to sample A exceeds a pre-defined threshold. DR.LSH comprises two main steps: hashing data points into buckets, and Measuring similarity and removing highly similar (i.e., redundant) samples. Later, Aslani and Seipel (2021) introduced Border Point extraction based on Locality-Sensitive Hashing (BPLSH) as an enhancement to LSH. BPLSH preserves border patterns as support vector candidates and retains a small number of interior instances to represent class extent. It searches for border instances in heterogeneous areas, which are regions containing instances from different classes. The method is based on the idea that a border instance has heterogeneous neighbours and is also the nearest neighbour of a sample from another class. Generally, BPLSH involves two major phases: Identifying the buckets for each sample and Detecting border samples and eliminating dispensable instances. Figure 2.8 presents the updated prototype selection taxonomy, reflecting methods applied in big data contexts. Methods marked with "**" indicate those specifically designed for big data.

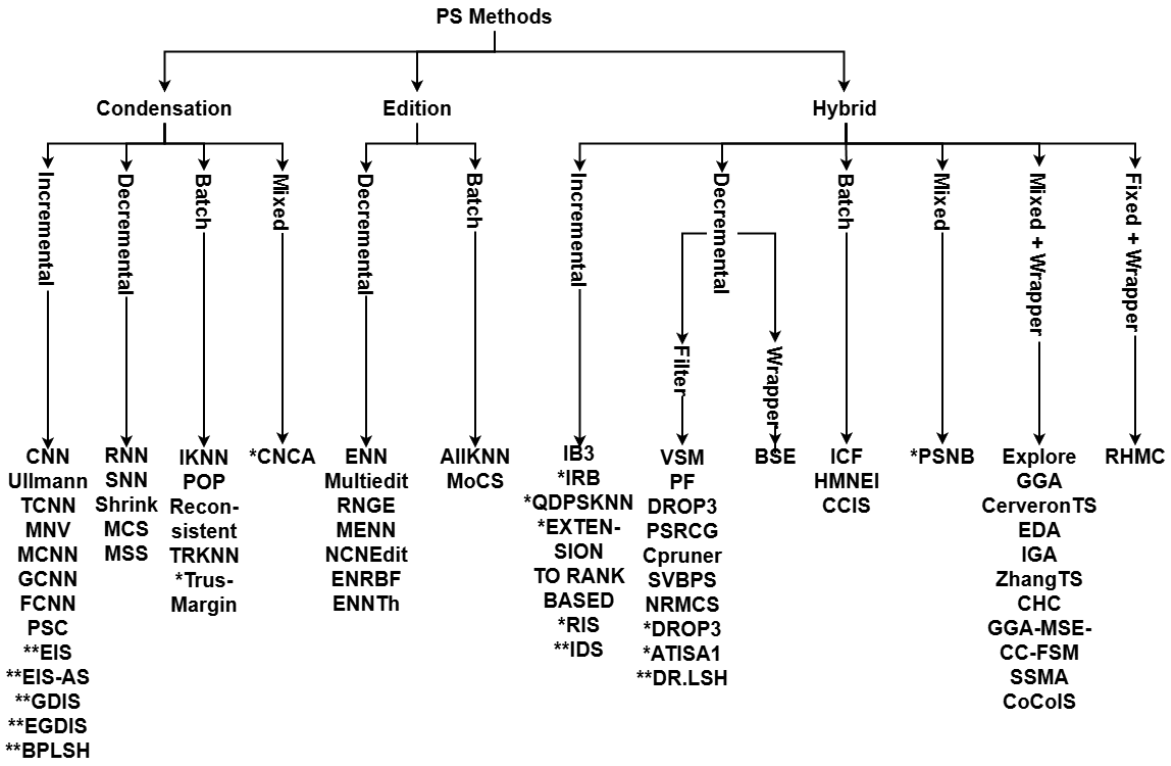


**Figure 2.8:** Updated Taxonomy of Prototype Selection Techniques for Big Data, Adapted from Garcia et al. (2012)

2.4.2.2 Prototype Generation

Many existing prototype generation (PG) approaches exhibit at least $O(n^2)$ time complexity (where $n$ is the number of instances), making them impractical for addressing big data challenges due to their computational cost. To overcome these limitations, Triguero et al. (2015) proposed a distributed solution named MRPR, which enables the application of existing PG techniques to large datasets. However, a significant drawback remains: the quadratic time complexity inherent in most PG methods still applies, particularly when the growth in data size substantially exceeds the increase in the number of computing units. In such cases, the solution becomes unfeasible. Additionally, the method's performance may degrade as the degree of parallelism increases, further exacerbating this limitation (Elkano et al., 2018).

Elkano et al. (2018) present CHI-PG, a distributed MapReduce-based prototype generation method that operates in linear time $O(n)$ and maintains constant classification accuracy regardless of the degree of parallelism used. This approach generates prototype types using a simple strategy derived from the rule generation constructs introduced by Chi et al. (1996). Nguyen et al. (2020) proposed a Dynamic Self-Organising Swarm (DSOS). DSOS is a particle swarm algorithm designed to emulate the foraging behaviours of animals. In DSOS, particles navigate through and search for food sources in a high-dimensional space. Here, each data input serves as a food signal, attracting particles to move toward its proximity. As a particle gets closer to the food source, it gains more energy, although its energy gradually depletes over time. Notably, a food source has limited availability, and only particles close to it can consume it to replenish energy. Particles that share a food source become connected (i.e., neighbours) and guide one another toward incoming food signals. If a food signal is too far from all existing particles, a new particle is generated at that location. Connected particles without a shared food source experience a weakening of their connections due to increasing repulsive forces, ultimately leading to their disconnection. Particles with zero energy or no remaining connections are eventually removed from the swarm. This swarm behaviour allows DSOS to dynamically and efficiently capture the underlying data distribution. Figure 2.9 presents the updated prototype generation taxonomy, which reflects the methods applied to big data. Methods marked with "**" indicate those specifically designed for big data applications.

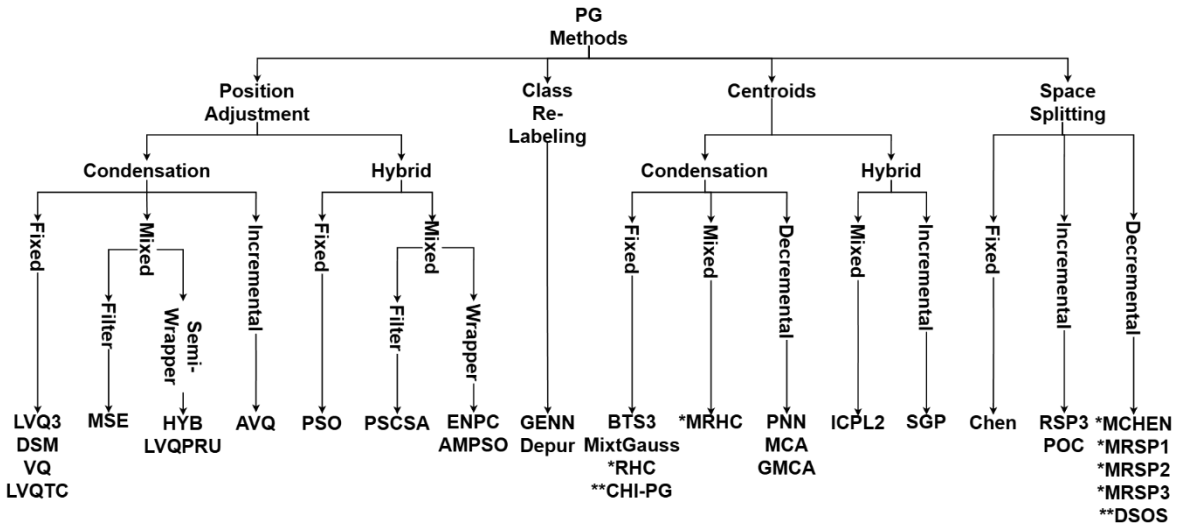


**Figure 2.9:** Updated Taxonomy of Prototype Generation Techniques for Big Data, Adapted from Triguero et al. (2012)

### 2.4.3 Distributed and Parallelization Frameworks

Distributed frameworks aim to efficiently manage and process large datasets across clusters of computers. Prominent examples include Apache Hadoop and Apache Spark. According to Kumar and Mohbey (2019), Hadoop employs the MapReduce programming model, which is designed to distribute applications across multiple computing nodes. The two primary components of Hadoop are the Hadoop Distributed File System (HDFS) and the MapReduce framework. MapReduce is characterised by several distinct features:

i. Data Partitioning and Distribution: It partitions data into equal-sized blocks, creates replicas for redundancy, and distributes them evenly across the file system, relieving users from manual data location and distribution management.

ii. Fault Tolerance: MapReduce automatically retries and re-executes failed tasks independently, ensuring robust reliability without affecting other ongoing processes.

iii. Dynamic Load Balancing: It enhances overall performance by reassigning incomplete tasks from overloaded or slow nodes to available idle nodes, thereby optimising resource utilisation.

The MapReduce model operates in two key stages: map and reduce. In the map stage, data inputs are processed using a map function, producing numerous key-value pairs. In the subsequent reduce stage, these key-value pairs are aggregated to generate the final outputs. The two stages are coordinated through key-value operations that ensure proper grouping and processing.

In Hadoop, data files are evenly segmented and stored within the Hadoop Distributed File System (HDFS). Map functions are typically assigned to workers that possess the corresponding data segments, thereby minimising data transfer overhead. MapReduce leverages data locality by placing computation close to the data storage, enabling faster access. Its architecture also includes automatic fault detection and recovery mechanisms, reducing the user's concern about node failures.

In contrast, Apache Spark offers significant performance improvements through its in-memory processing capabilities. Kumar and Mohbey (2019) emphasise Spark's ability to retain intermediate data in memory, enabling faster iterative processing. Spark introduces Resilient Distributed Datasets (RDDs), which are immutable, distributed datasets that support various built-in transformations. These RDDs are cached across worker nodes, significantly accelerating data reuse. Spark achieves fault tolerance through lineage information, allowing it to reconstruct RDDs without explicit replication in the event of node failure. Furthermore, Spark employs a Directed Acyclic Graph (DAG) to manage data flows efficiently, replacing Hadoop’s traditional two-stage MapReduce model.

The Directed Acyclic Graph (DAG) in Spark is a fundamental component of its execution model. It is referred to as "directed" because operations are executed sequentially in a predefined order, and "acyclic" because it contains no loops or cycles. This structure ensures that each stage depends on the successful completion of the preceding one, while individual tasks within a stage can run independently. Figure 2.10 illustrates the Directed Acyclic Graph.

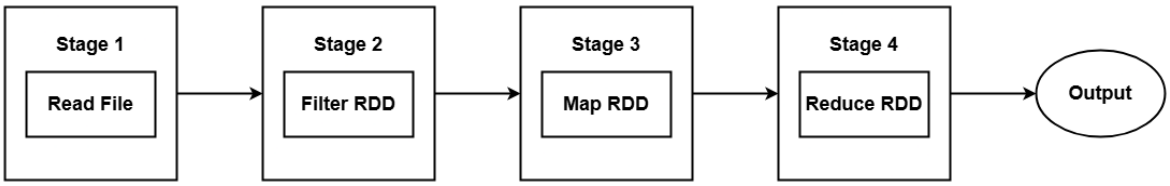


**Figure 2.10:** Directed Acyclic Graph (DAG)

At a conceptual level, a Directed Acyclic Graph (DAG) represents the logical execution plan of a Spark job. When a Spark application is submitted, Spark translates high-level operations (i.e. transformations and actions defined in user code) into a DAG of executable stages and tasks.

Maillo et al. (2017) developed an iterative, Spark-based k-NN method (kNN-IS) to enhance the performance of the k-NN algorithm on big data. Their approach retains the fundamental structure of k-NN while enabling scalability through Apache Spark.

Similarly, Juez-Gil et al. (2021) proposed Approx-SMOTE, a parallel implementation of the SMOTE algorithm specifically tailored for Apache Spark. Unlike traditional SMOTE or its big data variant (SMOTE-BD), which rely on exact k-NN computations, Approx-SMOTE adopts an approximate k-NN approach, significantly

improving scalability. This method achieves execution speeds up to 30 times faster, without compromising classification accuracy.

Srinivas and Kancharla (2019) utilised Hadoop's MapReduce framework for efficient big data processing, specifically aiming to improve the computational complexity of the k-NN algorithm. Gong et al. (2021) also employed Apache Spark to manage large-scale datasets, extending their Evidential Instance Selection (EIS) approach into a distributed, parallelised version (EIS-AS). Their methodology significantly reduced computational bottlenecks, demonstrating effective data simplification and improved scalability for handling extensive datasets.

Maillo et al. (2019) introduced two approximate fuzzy k-nearest neighbour classifiers explicitly designed for big data processing: Local Hybrid Spill Tree FkNN (LHS-FkNN) and Global Approximate Hybrid Spill Tree FkNN (GAHS-FkNN). Both methods leverage Spark's distributed and parallel computing capabilities, particularly during the MapReduce stages, to efficiently partition and process data. Specifically, Spark is employed in both the class membership degree calculation and classification stages, substantially improving runtime and scalability. LHS-FkNN processes smaller data subsets independently, while GAHS-FkNN uses a globally informed approximate tree structure to manage large-scale datasets.

Gong et al. (2022) proposed the Global Exact Evidential k-Nearest Neighbour (GE2K-NN) classifier, leveraging Apache Spark’s distributed computing capabilities to address the computational challenges inherent in the traditional Evidential k-NN (EK-NN). Their approach, later renamed DEKNN due to naming conflicts, partitions the training dataset, conducts parallel local candidate neighbour searches, and merges the results to

accurately determine global neighbours. GE2K-NN employs adaptive computations based on Dempster–Shafer theory, demonstrating superior classification accuracy and scalability. However, its performance is heavily dependent on available computational resources, and the addition of nodes beyond a certain threshold yield diminishing returns due to increased communication overhead.

Building on previous work, Gong et al. (2023b) developed two scalable Evidential k-NN classifiers: Global Exact EK-NN (GEK-NN) and Local Approximate EK-NN (LEK-NN). Both are implemented on Apache Spark to support distributed computing. GEK-NN operates without an explicit training phase, partitioning datasets for local nearest neighbour searches and merging the results to achieve global accuracy. In contrast, LEK-NN includes a distinct training stage, where classifier parameters are learned using distributed gradient descent across multiple partitions, and the results are aggregated to form a comprehensive global model.

Rezanejad and Danesh (2024) examined Dask as a distributed framework to enhance k-NN performance. Introduced by Rocklin (2015), Dask is a Python-based tool that enables seamless integration with popular libraries such as NumPy (Harris et al., 2020), Pandas (McKinney, 2010), and Scikit-learn (Pedregosa et al., 2011). Dask provides parallel and distributed computing through an intuitive, high-level interface. Its actor-model-based architecture and just-in-time scheduling approach efficiently manage task execution, reduce overhead, and ensure fault tolerance by automatically reassigning tasks in the event of node failures. Additionally, Dask supports distributed execution in high-performance computing (HPC) environments, effectively handling large-scale data processing tasks (Kumar, 2023).

Table 2.1 summarises the comparison between Apache Spark, Hadoop, and Dask based on their architecture, data storage, supported programming languages, processing speed, ease of use, and infrastructure costs.

**Table 2.1:** Comparison Table for Apache Spark, Hadoop and Dask

| Feature | Apace Spark | Hadoop (MapReduce) | Dask |
|---|---|---|---|
| Architecture | Distributed, In-memory processing | Distributed, Disk-based | Parallel, In-memory (local and distributed) |
| Data Storage | In-memory (RAM) | Disk | In-memory (RAM) |
| Programming Language | Scala, Java, Python, R | Java (mainly), Python, C++ | Python (primary) |
| Processing Speed | Fast (RAM-based, 100x faster than Hadoop) | Slow (disk I/O bottleneck) | Faster than Hadoop, and 50% faster (Rocklin, 2015) than Spark on standard benchmarks |
| Ease of Use | Moderate (requires knowledge of RDDs) | Hard (complex MapReduce coding) | Easy (familiar API like NumPy/Pandas) |
| Cost (Infrastructure) | High (RAM-intensive) | High (Disk I/O heavy) | Low (can run locally) |

## 2.5 Summary

This chapter has discussed key areas relevant to the current study. It began with a recap of how the k-nearest neighbours (k-NN) algorithm works. It then examined the main challenges associated with k-NN, particularly the curse of dimensionality and computational complexity, followed by a discussion of the corresponding solutions, including dimensionality reduction and data reduction techniques. The solution subsection also covered other enhancement methods that influence overall classification performance, such

as the selection of the $k$ parameter and the choice of similarity distance. In addition, this chapter explored the defining characteristics of big data and the challenges it poses, specifically the curse of dimensionality and computational complexity, and reviewed how previous studies have addressed these issues. Finally, it reviewed distributed computing frameworks that assist in managing large-scale datasets.

# CHAPTER 3

# METHODOLOGY

## 3.1 Overview

The objective of this chapter is to present the methodology employed in conducting the study. It outlines the research steps undertaken and provides a detailed description of the benchmark datasets used in the experimental evaluation. Furthermore, the chapter explains the pre-processing steps applied to these datasets to prepare them for experimentation. The experimental setup, system requirements, and the evaluation and analysis methods used in the study are also discussed in detail. The chapter concludes with a summary that encapsulates the key points covered.

## 3.2 Research Flow

This section elaborates on the processes and procedures followed in this study. The research adopts a combination of build and empirical methodologies. A summary of the research procedures is presented in Figure 3.1.

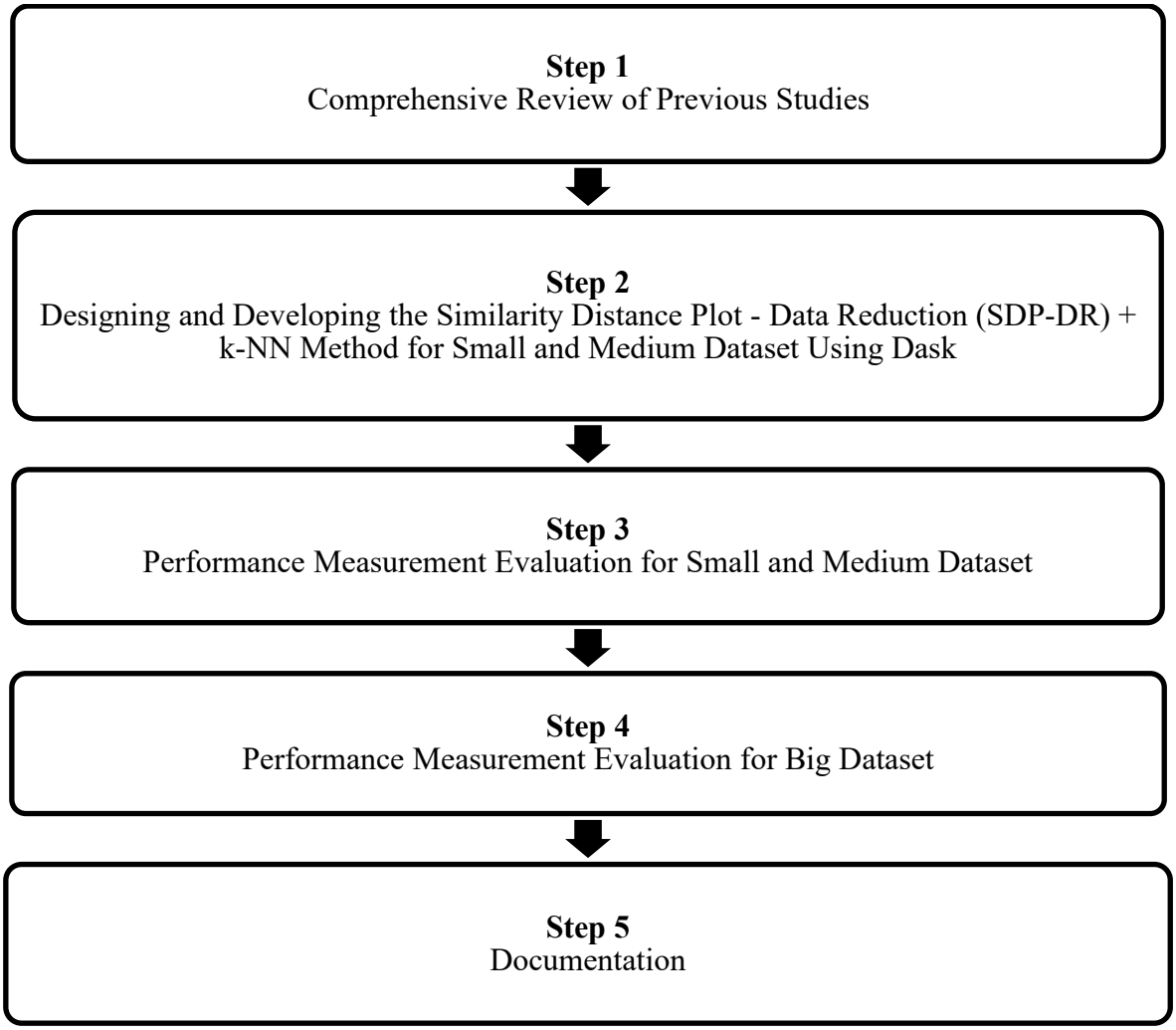


**Figure 3.1:** The Seven Main Research Steps

### 3.2.1 Step 1 - Comprehensive Review of Previous Studies

This step establishes the methodological foundation by linking the limitations of the original k-NN algorithm (as reviewed in Chapter 2) with the enhancements required for this study. Specifically, it justifies the selection of Prototype Selection (PS) and Prototype Generation (PG) approaches as the basis for the proposed SDP-DR method, and highlights the relevance of distributed frameworks (Apache Spark, Hadoop, Dask) in addressing big data classification challenges..

### 3.2.2 Step 2 – Designing and Developing the Similarity Distance Plot - Data Reduction (SDP-DR) + k-NN Method for Small and Medium Dataset using Dask

This step involves constructing the Similarity Distance Plot–Data Reduction (SDP-DR) method, which utilises similarity distance to automatically select prototypes as part of the Prototype Selection (PS) process. The first phase of the proposed method involves loading the benchmark dataset. Before splitting the dataset into training and testing sets, it must undergo data preparation and pre-processing. During this stage, complete datasets are obtained from either the KEEL repository (Alcalá-Fdez et al., 2011) or the UCI Machine Learning Repository (Asuncion, 2007). Initially, the dataset, which is downloaded in .txt format, is converted to .csv format to facilitate easier processing. The conversion, along with the pre-processing steps, is carried out in Jupyter Notebook using Python. After format conversion, the dataset undergoes the following pre-processing steps:

i. **Nominal-to-Integer Transformation:** Nominal values are converted into integers to ensure compatibility with subsequent operations.
ii. **Normalisation:** The data is normalised using the Min–Max normalisation technique to scale features within a specified range. The Min–Max normalisation formula is as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \qquad \text{Equation 3.1}$$

After the normalisation process is completed, the dataset is ready to be divided into training and testing sets. Following this, stratified 10-fold cross-validation is employed to split the dataset into training data (TR) and testing data (TS). The TR is then processed using the SDP-DR method to create a model used for classifying the TS. The SDP-DR process begins by identifying an anchor within each training fold. This means that each fold has its

own anchor and corresponding reduced set. The reduced sets from all folds are then combined to form a unified reduced model. Subsequently, the similarity distance between the selected anchor and the training data is computed using the Euclidean distance. The formula for calculating Euclidean distance is presented in Equation 2.4.

As a member of the Minkowski family of distance metrics, Euclidean distance is widely recognised and frequently utilised across a range of applications. Its primary advantage lies in its computational efficiency, with a time complexity of $O(n)$ making it well-suited for datasets with varying dimensions (Shirkhorshidi et al., 2015). This efficiency is particularly critical in contexts with limited computational resources or where rapid processing is required. Furthermore, empirical research has consistently shown that Euclidean distance is effective in handling datasets with both dense and sparse class structures (Shirkhorshidi et al., 2015). This effectiveness stems from its sensitivity to the magnitude of differences between data points, allowing it to detect subtle variations in the data. As a result, it is particularly suitable for the proposed method, which aims to uncover patterns and relationships within complex datasets.

In the proposed method, three types of anchors are introduced:

i. **First Data (FD) anchor:** uses the first data point in the dataset as the anchor.
ii. **Mean of Each Column (MEC) anchor:** calculates the mean of each column and uses it as the anchor.
iii. **Data Closest to Mean (DCM) anchor:** calculates the mean of each column and selects the data point closest to this mean as the anchor.

The method applies anchor-based prototype selection to efficiently reduce the size of the training dataset. Algorithm 1 presents the pseudocode for anchor selection. Once

similarity distances are calculated, the values are sorted from the closest to the farthest relative to the anchor. The head and tail of each subclass are chosen as subclass representatives, while the data in the centre are discarded. This strategy, referred to as "remove internal", is a type of condensed data reduction approach. After representative selection, the chosen data points serve as prototypes for the classification process.

Figure 3.2 illustrates the prototype selection process using the Remove Border strategy. The dataset consists of instances from three classes: C1x (green), C2x (blue), and C3x (red), where x represents the number of instances in each class. In the original dataset (top row), each class contains multiple instances. The middle row shows the removal of border instances (e.g., C12, C13, and C33), which lie near the decision boundaries between classes and may cause ambiguity during classification. The bottom row presents the reduced dataset after removing these border instances, leaving only the most representative prototypes for each class. This reduction not only decreases the number of distance calculations required during k-NN classification, improving computational efficiency, but also reduces potential noise at class boundaries, thereby enhancing classification accuracy.

Figure 3.3 presents the overall process flow of the proposed Similarity Distance Plot–Data Reduction (SDP-DR) method. The process begins with the training data, followed by the selection of an anchor point from the dataset. Next, the similarity distance between the anchor and each training instance is calculated. These distances are then sorted, and the data is plotted in a one-dimensional graph. Based on the plotted distribution, a prototype selection technique is applied to identify the most representative instances. The reduced dataset is then used to build a new classification model. Algorithm 2 present prototype selection by removing internal instances and retaining border or representative instances that are more

likely to influence decision boundaries. The data is assumed to be sorted based on similarity distance (e.g., from the SDP-DR process). A linked list is used to traverse the data in order. The algorithm compares each instance to its neighbour: if two consecutive instances belong to different classes, and their similarity distances differ, the current instance is considered informative and kept. This process continues for the entire dataset. The final output is a reduced dataset that preserves only the most meaningful prototypes, potentially improving classification accuracy and efficiency.

**ALGORITHM 1: FIND ANCHOR**

*Input: Dataset*
*Anchor Type: FD, DCM, MEC*

*Output:*
*- Selected anchor* $\boldsymbol{a}$

*Begin:*
*If anchor type is First Data (FD):*
    *Select first data entry from the dataset as* $\boldsymbol{a}$
*end*

*If anchor type is Data Closed to Mean (DCM):*
    *Calculate the mean for each column = mean(dataset)*
    ***For** each data entry **do**:*
        *Calculate the absolute difference between the data entry and the column means*
        *Sum all the differences*
        *Append the data*
    ***end***
    *Sort the list by summed differences*
    *Extract the data entry with the smallest difference to be the anchor as* $\boldsymbol{a}$
***end***

*If anchor type is Mean of Each Column (MEC):*
    *Calculate the mean for each column = mean(dataset)*
    *Store the mean as the anchor point as* $\boldsymbol{a}$
***end***
***return** anchor* $\boldsymbol{a}$
***end***

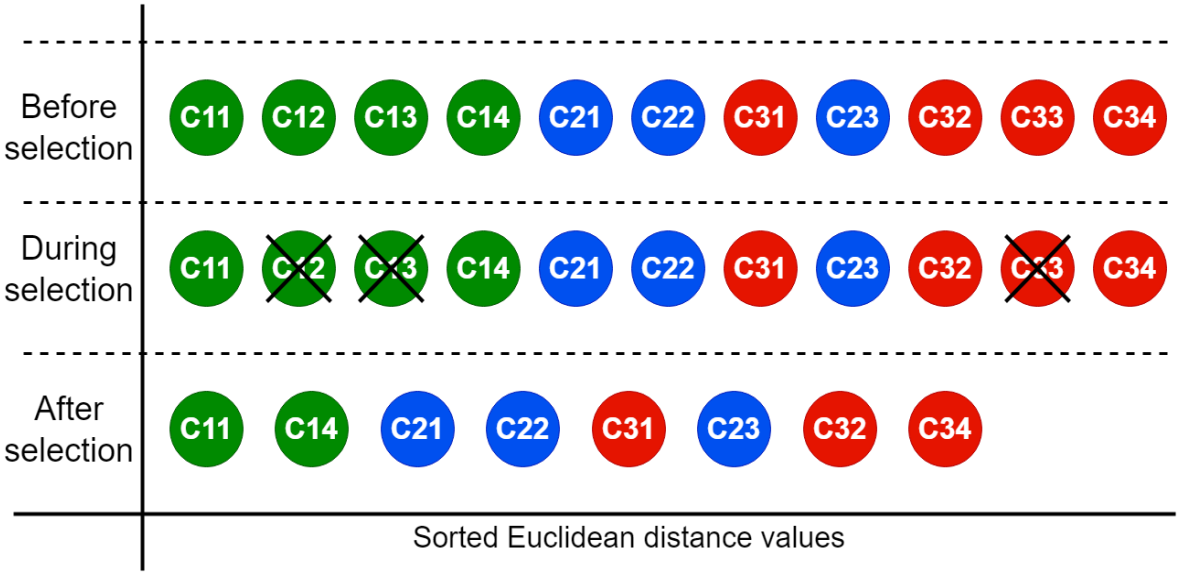


**Figure 3.2:** SDP-DR Prototype Selection Process

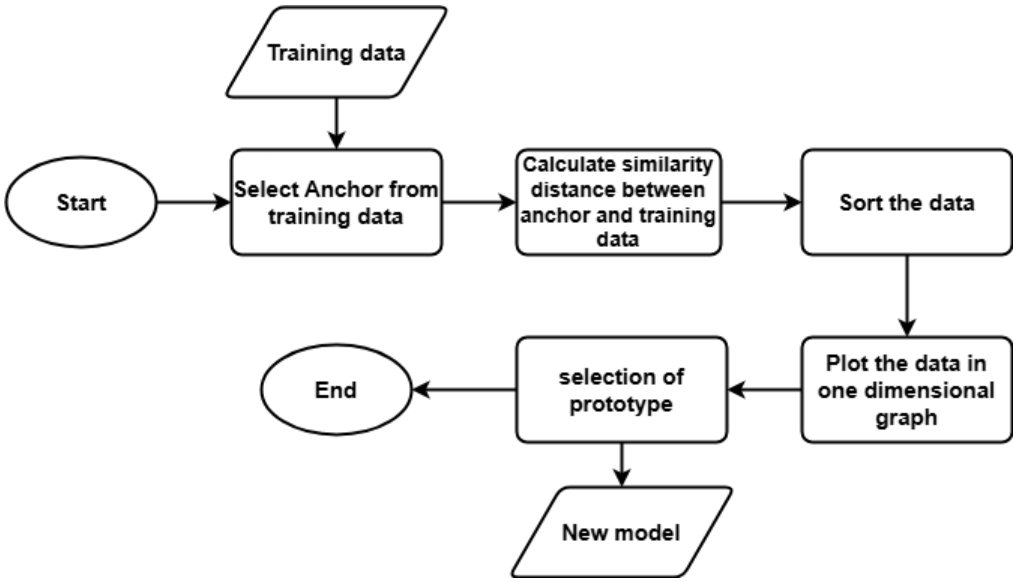


**Figure 3.3:** SDP-DR Flowchart

**ALGORITHM 2: PROTOTYPE SELECTION USING REMOVE INTERNAL**

```
    Input: Dataset
    Output: New reduce model
1   Begin:
2   temp = empty list
3   repNode = head.node
4   currentNode = head.node
5   For each currentNode in the linked list do:
6       If the next node exists do:
7           If the class of the next node is differs from the class of currentNode do:
8               If the similarity distance value of repNode differ from currentNode do:
9                   Add the currentNode data to temp
10              end
11          end
12          Add the next node data to temp
13          Update repNode = next node
14      end
15  end

16  While currentNode is not None do:
17      If next node is None do:
18          Add the currentNode data to temp
19      end
20      return new reduce model = temp
21  End
```

For the testing phase, the *k* value for the k-NN algorithm is defined; in this study, *k* =1, 3, 5, 7 is used. The Euclidean distance formula, as shown in Table 3.1, is then applied to calculate the similarity distance between all prototypes and the TS records. These values are sorted in ascending order, from the closest to the farthest from each TS record. Based on this sorted list, the class of each TS record is determined by selecting the nearest prototypes. This process is repeated until all records in the TS set are classified. All processes are executed within the Dask framework. Figure 3.4 illustrates the overall process of the proposed k-NN classification system using the Similarity Distance Plot–Data Reduction (SDP-DR) method. The process starts by loading the dataset, which is then split using k-fold cross-validation to separate training and testing data. The SDP-DR method is applied to the training data to generate a new reduced prototype set for each fold. A suitable *k* value is defined for the k-NN classifier, which is then used to classify the testing data. Figure 3.5 shows the overall

architecture of the proposed system, implemented using Dask for parallel processing. The dataset is divided into ten folds, where nine folds are used as the training set and one fold as the testing set. Each of the nine training folds undergoes a data reduction process, generating a reduced set of instances. These reduced sets are then combined and used for classification using the k-NN algorithm. This process was repeated ten times, with each fold serving once as the testing set. The reported results represent the average across all folds. The classification performance is evaluated, and metrics such as Mean Reduction Time, Mean Reduction Rate, Mean Accuracy, and Mean Classification Time are recorded. Algorithm 3 describes the complete process of combining the Similarity Distance Plot–Data Reduction (SDP-DR) method with the k-NN classifier. The process begins by applying stratified k-fold cross-validation to split the dataset while maintaining class distribution. For each fold, the SDP-DR technique is used to reduce the training data. The reduction process involves selecting an anchor, calculating similarity distances, sorting the data, and applying a prototype selection method to remove internal instances. After reduction, the k-NN classifier is applied using the reduced data. The algorithm records several performance metrics across all folds, including mean accuracy, mean classification time, mean reduction rate, and mean reduction time, to evaluate the effectiveness and efficiency of the approach.

Based on the proposed design, selecting an appropriate anchor and similarity distance within the SDP-DR framework is crucial to achieving optimal results in terms of classification accuracy, reduced prototype storage, and faster classification time. The proposed method is implemented in Visual Studio Code, using Python along with the scikit-learn library (Pedregosa et al., 2011), Dask, Numpy (Harris et al., 2020), and Pandas (McKinney, 2010) to evaluate classification accuracy, classification time, reduction percentage, and prototype reduction for each dataset.

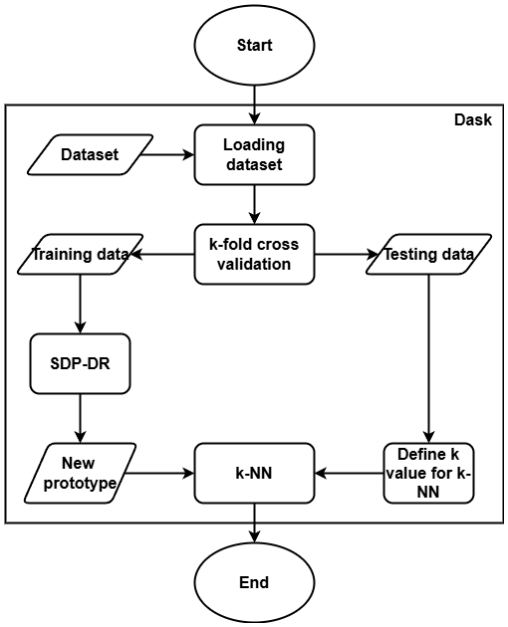


**Figure 3.4:** SDP-DR + k-NN (Dask) Flowchart

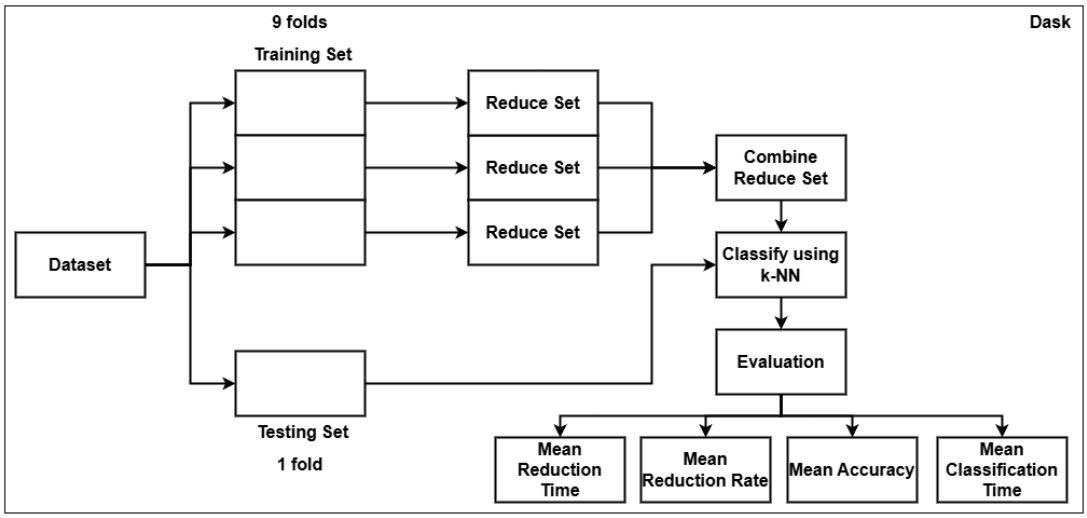


**Figure 3.5:** SDP-DR + k-NN (Dask) Framework

**ALGORITHM 3: SDP-DR + K-NN**

***Input:***
*- Dataset: The input data to be processed*
*- Anchor Type: Type of anchor point (FD, DCM, MEC)*
***Output:***
*- Mean Accuracy: Average accuracy across all folds*
*- Mean Classification Time: Average time for classification*
*- Mean Reduction Rate: Average percentage of data reduction*
*- Mean Reduction Time: Average time spent on data reduction*
***Begin:***

*Function Stratified k-fold cross-validation:*
    *Read the dataset and convert class labels to numerical values.*
    *Initialize stratified k-fold cross-validation*
    ***For** each train, test split in stratified k-fold **do**:*
        *Extract and store train and test data for each fold*
    ***End***

*Start reduction time* $\boldsymbol{Start\_Time_{red}}$
*Function SDP-DR:*
    *Split training data into 9 parts*
    *For each data split:*
        *Find anchor* $\boldsymbol{a}$ *= find anchor (anchor type)*
        *Calculate distances = calculate distances (*$\boldsymbol{a}$*,* $\boldsymbol{Data_{tra}}$*, Euclidean distance)*
        *Sort the calculated* $\boldsymbol{Data_{tra}}$
        *Prototype selection* $\boldsymbol{pr}$ *= prototype selection (*$\boldsymbol{Data_{tra}}$*, remove internal)*
        *Remove duplicate = remove duplicate(**pr**)*
        *Combine* $\boldsymbol{pr = new\ pr}$
    *End*
    *Stop reduction time* $\boldsymbol{Stop\_Time_{red}}$

    *Start Classification time* $\boldsymbol{Start\_Time_{class}}$
    *Classification using k-NN = k-NN (**new pr,** *$\boldsymbol{Data_{test}}$*, Euclidean distance)*
    *Accuracy* $\boldsymbol{acc}$ *= ration of correct prediction of the k-NN*
    *Stop Classification time* $\boldsymbol{Stop\_Time_{class}}$
    *Reduction time* $\boldsymbol{t_{red} = Stop\_Time_{red} - Start\_Time_{red}}$
    *Reduction Rate* $\boldsymbol{Red_{rate}} = \frac{\boldsymbol{Data_{tra} - pr}}{\boldsymbol{Data_{test}}}$
    *Classification time* $\boldsymbol{t_{class} = Stop\_Time_{class} - Start\_Time_{class}}$

*mean accuracy = Equation 3.1*
*mean Reduction Time = Equation 3.2*
*mean Classification time = Equation 3.3*
*mean Reduction Rate = Equation 3.4*
***end***

### 3.2.3 Step 3 - Performance Measurement Evaluation for Small and Medium Dataset

Classification performance is evaluated using mean accuracy, calculated as shown in Equation 3.2. Reduction time is assessed by computing the mean reduction time, as defined in Equation 3.3. Similarly, classification time is measured using the mean classification time, derived from Equation 3.4. The reduction rate is determined using Equation 3.5.

$$Mean\ accuracy\ (\%) = \frac{total\ accuracy}{10}\ x\ 100\% \quad \text{Equation 3.2}$$

$$Reduction\ time\ (ms) = \frac{end\ reduction\ time - start\ reduction\ time}{10} \quad \text{Equation 3.3}$$

$$Classification\ time\ (ms) = \frac{end\ testing\ time - start\ testing\ time}{10} \quad \text{Equation 3.4}$$

$$Reduction\ rate\ (\%) = \frac{Original\ data - Reduce\ data}{Original\ data}\ x\ 100\% \quad \text{Equation 3.5}$$

### 3.2.4 Step 4 - Performance Measurement Evaluation for Big Dataset

The proposed method is evaluated in terms of classification performance, time efficiency, and prototype storage capacity. This evaluation is carried out using large benchmark datasets obtained from the UCI Machine Learning Repository (Asuncion, 2007) and Keel Repository (Alcalá-Fdez et al., 2011).

Classification performance is measured using mean accuracy, calculated using Equation 3.2. Reduction time and classification time are assessed using their respective mean values, derived from Equation 3.3 and Equation 3.4. The reduction rate is determined using Equation 3.5. All evaluation formulas are summarised in Table 3.2.

### 3.2.5 Step 5 – Documentation

All design components, experimental results, and analyses are documented in the form of a thesis and academic articles.

## 3.3 Datasets Used

For the evaluation of the proposed algorithms, a variety of benchmark datasets were selected for comparative experiments. These datasets were obtained from the KEEL Repository (Alcalá-Fdez et al., 2011) and the UCI Machine Learning Repository (Asuncion,

2007). The selection includes 17 binary-class and 20 multi-class datasets from diverse application domains. Furthermore, the datasets exhibit variation in the relative proportions between classes. Table 3.1 summarises the selected datasets, categorised by class proportion as balanced (B), imbalanced (IB), or extremely imbalanced (EIB). A dataset is considered extremely imbalanced if the minority class comprises 5% or less of the total instances (He & Garcia, 2009). However, as the primary objective of this study is to evaluate algorithm performance across a wide range of datasets, class proportion and attribute count are not the main focus. Dataset size (i.e. number of instances) is categorised as small (S), medium (M), or big dataset (BD) according to the criteria established by (Garcia et al., 2012; Gautam & Chatterjee, 2020; Triguero et al., 2012). Small datasets contain 2,000 or fewer instances; medium datasets contain 2,001 to 20,000 instances; and large datasets have more than 20,000 instances. Attribute sizes are categorised using the specifications from (Chuang et al., 2011). Datasets with 19 or fewer attributes are considered small, those with fewer than 50 attributes are medium, and datasets with 50 or more attributes are classified as large. In total, 22 small, 10 medium, and 5 large datasets (by volume) were selected. Based on attribute size, 26 datasets are small, 7 are medium, and 4 are large.

**Table 3.1:** Brief Description on Benchmark Datasets

| Dataset | NoI | Size | Class | NoA | MV | CD | MnC(%) | MjC(%) |
|---|---|---|---|---|---|---|---|---|
| Zoo (Alcalá-Fdez et al., 2011) | 101 | S | 7 | 16 | No | IB | 0.04 | 0.41 |
| Appendicitis (Alcalá-Fdez et al., 2011) | 106 | S | 2 | 7 | No | IB | 0.2 | 0.8 |
| Hayes-roth (Alcalá-Fdez et al., 2011) | 160 | S | 3 | 4 | No | IB | 0.19 | 0.41 |
| Wine (Alcalá-Fdez et al., 2011) | 178 | S | 3 | 13 | No | IB | 0.27 | 0.4 |
| House vote (Alcalá-Fdez et al., 2011) | 232 | S | 2 | 16 | No | IB | 0.47 | 0.53 |
| Heart (Alcalá-Fdez et al., 2011) | 270 | S | 2 | 13 | No | IB | 0.44 | 0.56 |
| Haberman (Alcalá-Fdez et al., 2011) | 306 | S | 2 | 3 | No | IB | 0.26 | 0.74 |
| Bupa (Alcalá-Fdez et al., 2011) | 345 | S | 2 | 6 | No | IB | 0.42 | 0.58 |
| Ionosphere (Alcalá-Fdez et al., 2011) | 351 | S | 2 | 33 | No | IB | 0.36 | 0.64 |
| Movement-Libras (Alcalá-Fdez et al., 2011) | 360 | S | 15 | 90 | No | B | 0.07 | 0.07 |
| Monk-2 (Alcalá-Fdez et al., 2011) | 432 | S | 2 | 6 | No | IB | 0.47 | 0.53 |
| Led7digit (Alcalá-Fdez et al., 2011) | 500 | S | 10 | 7 | No | IB | 0.07 | 0.11 |
| Wdbc (Alcalá-Fdez et al., 2011) | 569 | S | 2 | 30 | No | IB | 0.37 | 0.63 |
| Balance (Alcalá-Fdez et al., 2011) | 625 | S | 3 | 4 | No | IB | 0.07 | 0.46 |
| Australian (Alcalá-Fdez et al., 2011) | 690 | S | 2 | 14 | No | IB | 0.44 | 0.56 |
| Pima (Alcalá-Fdez et al., 2011) | 768 | S | 2 | 8 | No | IB | 0.35 | 0.65 |
| Mammographic (Alcalá-Fdez et al., 2011) | 830 | S | 2 | 5 | No | IB | 0.49 | 0.51 |
| Vehicle (Alcalá-Fdez et al., 2011) | 846 | S | 4 | 18 | No | B | 0.24 | 0.26 |
| Vowel (Alcalá-Fdez et al., 2011) | 990 | S | 11 | 13 | No | B | 0.5 | 0.5 |
| Contraceptive (Alcalá-Fdez et al., 2011) | 1473 | S | 3 | 9 | No | IB | 0.23 | 0.43 |
| Yeast (Alcalá-Fdez et al., 2011) | 1484 | S | 10 | 8 | No | IB | 0.003 | 0.31 |

**Table 3.1** continued

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Winequality-Red (Alcalá-Fdez et al., 2011) | 1599 | S | 11 | 11 | No | IB | 0.006 | 0.43 |
| Titanic (Alcalá-Fdez et al., 2011) | 2201 | M | 2 | 3 | No | IB | 0.32 | 0.68 |
| Segment (Alcalá-Fdez et al., 2011) | 2310 | M | 7 | 19 | No | B | 0.14 | 0.14 |
| Abalone(Alcalá-Fdez et al., 2011) | 4174 | M | 28 | 8 | No | EIB | 0.00024 | 0.17 |
| Winequality-White (Alcalá-Fdez et al., 2011) | 4898 | M | 11 | 11 | No | IB | 0.001 | 0.45 |
| Texture (Alcalá-Fdez et al., 2011) | 5500 | M | 11 | 40 | No | B | 0.09 | 0.09 |
| Optdigits (Alcalá-Fdez et al., 2011) | 5620 | M | 10 | 64 | No | B | 0.1 | 0.1 |
| Satimage (Alcalá-Fdez et al., 2011) | 6435 | M | 7 | 36 | No | IB | 0.1 | 0.24 |
| Marketing (Alcalá-Fdez et al., 2011) | 6876 | M | 9 | 13 | No | IB | 0.07 | 0.18 |
| Twonorm (Alcalá-Fdez et al., 2011) | 7400 | M | 2 | 20 | No | B | 0.5 | 0.5 |
| Coil2000 (Alcalá-Fdez et al., 2011) | 9822 | M | 2 | 85 | No | IB | 0.06 | 0.94 |
| Kddcup (Asuncion, 2007) | 494020 | BD | 23 | 41 | No | EIB | 0.00000404 | 0.57 |
| Epsilon (Asuncion, 2007) | 500000 | BD | 2 | 2000 | No | B | 0.50 | 0.50 |
| Poker (Asuncion, 2007) | 1025009 | BD | 10 | 10 | No | EIB | 0.0000078 | 0.50 |
| Susy (Asuncion, 2007) | 5000000 | BD | 2 | 18 | No | IB | 0.45 | 0.54 |
| Higgs (Asuncion, 2007) | 11000000 | BD | 2 | 28 | No | IB | 0.47 | 0.53 |

**Note:** NoI - no. of Instances, NoA - no. of Attribute, MV- missing value, CD – class distribution, MnC – minority class, MjC – majority class, S – small, M – medium, BD – big dataset, B – balance, IB – imbalanced, EIB – extremely imbalanced.

## 3.4 System Requirements

Jupyter Notebook was used during the data pre-processing stage. All dataset files were provided as flat files with the .csv extension. Furthermore, all results and analyses were presented using Microsoft Excel and Tableau 2019.4. The proposed technique was

developed using Python and the Dask framework within Visual Studio Code. All experiments involving small and medium-sized datasets were conducted on a personal computer (PC) equipped with an Intel® Core™ i5-7200U CPU running at 2.5 GHz (up to 3.1 GHz with Turbo Boost), 12 GB of RAM, and a 1 TB NVMe SSD, operating on Windows 10 Professional. For experiments involving big data, a PC provided by Universiti Malaysia Sarawak was used. This machine featured an Intel® Xeon® E5-1620 CPU at 3.60 GHz, 64 GB of RAM, and a 500 GB HDD for storage.

## 3.5 Summary

This chapter has outlined the procedures, implementation steps, and processes involved in conducting the study. It has also described the preparation of benchmark datasets to ensure accurate results and analysis. Furthermore, the implementation of the proposed method was explained in detail, along with the hardware and software requirements for conducting the experiments. The results have been analysed and validated based on the methodology presented in this chapter. The findings and their interpretation are discussed in the subsequent chapter.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Overview

This chapter presents the experimental evaluation of the proposed methodology, focusing on its effectiveness and efficiency in data classification across various scenarios. It revisits the methodologies applied in the study and outlines the evaluation techniques used to assess model performance. Benchmark algorithms, including k-Nearest Neighbour (k-NN), Condensed Nearest Neighbour (CNN), and Edited Nearest Neighbour (ENN), are employed to provide a comparative analysis. The evaluation centres on four key metrics: mean accuracy, classification time, reduction rate, and reduction time. These metrics collectively offer a comprehensive understanding of the model's classification capabilities and the efficiency of the data reduction process. Experiments are conducted on datasets of varying sizes, including small, medium, and large datasets, with large datasets emphasizing structured data volume. The primary goal is to evaluate overall accuracy and reduction rate, rather than focusing on dataset-specific issues such as class imbalance, noise sensitivity, high dimensionality, or missing values. To support scalability and improve computational efficiency, the Dask framework is utilized for handling datasets across all scales. The chapter concludes with a discussion of the results, highlighting performance improvements and trade-offs, and validating the proposed method's potential to enhance the general performance of k-NN while minimizing computational resource usage compared to established benchmark algorithms.

## 4.2 Performance Measurement Evaluation

This subsection describes the evaluation method used to assess the performance and efficiency of the proposed classification model in comparison with benchmark algorithms, namely k-Nearest Neighbour (k-NN), Condensed Nearest Neighbour (CNN), and Edited Nearest Neighbour (ENN). The evaluation framework employs stratified 10-fold cross-validation, which ensures that the class distribution remains consistent across each fold, thereby providing an unbiased and reliable assessment of model accuracy and efficiency. The evaluation process focuses on four key metrics: mean accuracy, classification time, reduction rate, and reduction time. Mean accuracy reflects the overall correctness of the classifications, while classification time measures the model's computational efficiency. Reduction rate and reduction time are critical for evaluating the effectiveness and efficiency of the data reduction process, demonstrating the proposed method's ability to reduce data volume without compromising classification performance. The results obtained from this evaluation provide insight into the strengths and limitations of the model when applied to various datasets, enabling a comprehensive analysis of its effectiveness relative to standard benchmark methods.

### 4.2.1 Performance Measurement Evaluation for Small and Medium Datasets

The first evaluation analysis presents the results for small and medium datasets, focusing on four key metrics: mean accuracy, mean classification time, mean reduction rate, and mean reduction time. This is followed by a comparison between the proposed method and other data reduction techniques in terms of mean accuracy. The algorithms evaluated in this analysis are:

i. k-NN: k- Nearest Neighbour

ii. ENN: Edited Nearest Neighbour

iii. CNN: Condensed Nearest Neighbour

iv. RIS: Ranking-based Instance Selection

v. ATISA1: Adaptive Threshold-based Instance Selection Algorithm

vi. DROP3: Decremental Reduction Optimization Procedure 3

vii. SDP-DR FD: Similarity Distance Plot-Data Reduction (First Data as Anchor)

viii. SDP-DR DCM: Similarity Distance Plot-Data Reduction (Data Closed to Mean as Anchor)

ix. SDP-DR MEC: Similarity Distance Plot-Data Reduction (Mean of Each Column as Anchor)

### 4.2.2 Datasets

The dataset collection, sourced from the KEEL dataset repository, encompasses a diverse range of datasets used in the experiments, each characterised by varying attributes, instance counts, and class distributions. These datasets span multiple domains, including healthcare (e.g., Mammographic, Pima, Heart), finance (e.g., Australian, Marketing), and image processing (e.g., Optdigits, Texture). The attribute types vary across datasets and include real (R), integer (I), and nominal (N) values. For instance, the Coil2000 dataset consists exclusively of integer attributes, while datasets such as Wine and Segment contain only real-valued attributes. Others, such as Australian and Zoo, include a mix of real, integer, and nominal features, reflecting the diversity in data representation.

The dataset sizes range from small datasets such as Zoo (101 instances) and Wine (178 instances) to medium-scale datasets like Marketing (6,876 instances) and Coil2000 (9,822 instances). Similarly, class distributions vary from binary classification tasks (e.g.,

Heart, Haberman, and Titanic) to multi-class problems, such as Vowel with 11 classes, Winequality-White with 11 classes, and Movement-Libras with 15 classes.

This dataset diversity ensures a comprehensive evaluation of the proposed method, enabling a robust assessment of its performance across datasets with varying sizes, attribute types, and class distributions. By testing on such a broad spectrum of datasets, demonstrating the effectiveness of the approach across a range of real-world classification scenarios.

**Table 4.1:** Small and Medium Datasets

| Dataset | Attribute (R/I/N) | Instances | Classes |
|---|---|---|---|
| Abalone | 8 (7/0/1) | 4174 | 28 |
| Appendicitis | 7 (7/0/0) | 106 | 2 |
| Australian | 14 (3/5/6) | 690 | 2 |
| Balance | 4 (4/0/0) | 625 | 3 |
| Bupa | 6 (1/5/0) | 345 | 2 |
| Coil2000 | 85 (0/85/0) | 9822 | 2 |
| Contraceptive | 9 (0/9/0) | 1473 | 3 |
| Haberman | 3 (0/3/0) | 306 | 2 |
| Hayes-roth | 4 (0/4/0) | 160 | 3 |
| Heart | 13 (1/12) | 270 | 2 |
| House votes | 16 (0/0/16) | 232 | 2 |
| Ionosphere | 33 (32/1/0) | 351 | 2 |
| Led7digit | 7 (7/0/0) | 500 | 10 |
| Mammographic | 5 (0/5/0) | 830 | 2 |
| Marketing | 13 (0/13/0) | 6876 | 9 |
| Monk-2 | 6 (0/6/0) | 432 | 2 |
| Movement-Libras | 90 (90/0/0) | 360 | 15 |
| Optdigits | 64 (0/64/0) | 5620 | 10 |
| Pima | 8 (8/0/0) | 768 | 2 |
| Satimage | 36 (0/36/0) | 6435 | 7 |
| Segment | 19 (19/0/0) | 2310 | 7 |
| Texture | 40 (40/0/0) | 5500 | 11 |
| Titanic | 3 (3/0/0) | 2201 | 2 |
| Twonorm | 20 (20/0/0) | 7400 | 2 |
| Vehicle | 18 (0/18/0) | 846 | 4 |
| Vowel | 13 (10/3/0) | 990 | 11 |
| Wdbc | 30 (30/0/0) | 569 | 2 |
| Wine | 13 (13/0/0) | 178 | 3 |
| Winequality-Red | 11 (11/0/0) | 1599 | 11 |
| Winequality-White | 11 (11/0/0) | 4898 | 11 |
| Yeast | 8 (8/0/0) | 1484 | 10 |
| Zoo | 16 (0/0/16) | 101 | 7 |

### 4.2.3 Results of Mean Reduction Time

The graphical representation of reduction time across different algorithms provides initial insight into computational efficiency. Figure 4.1 to Figure 4.5 illustrate the reduction time (RT) of ENN, CNN, and the proposed SDP-DR variants (FD, DCM, MEC) across 32 datasets. Figure 4.2 uses a slightly different reduction time scale (in milliseconds) due to CNN exhibiting significantly longer processing times compared to other methods. Therefore, the maximum reduction time scale was adjusted to 120,000 milliseconds to accommodate this variation. For clarity, the datasets are sorted in ascending order based on the number of instances, as shown in Table 4.1. A clear trend emerges in which CNN consistently exhibits the highest reduction time across all datasets, making it the most computationally expensive. In contrast, the proposed SDP-DR methods (FD, DCM, MEC) show significantly lower reduction times, demonstrating their ability to reduce computational complexity. Among the SDP-DR variants, SDP-DR (FD) consistently records the lowest reduction time across multiple datasets, indicating its potential efficiency for large-scale classification problems.

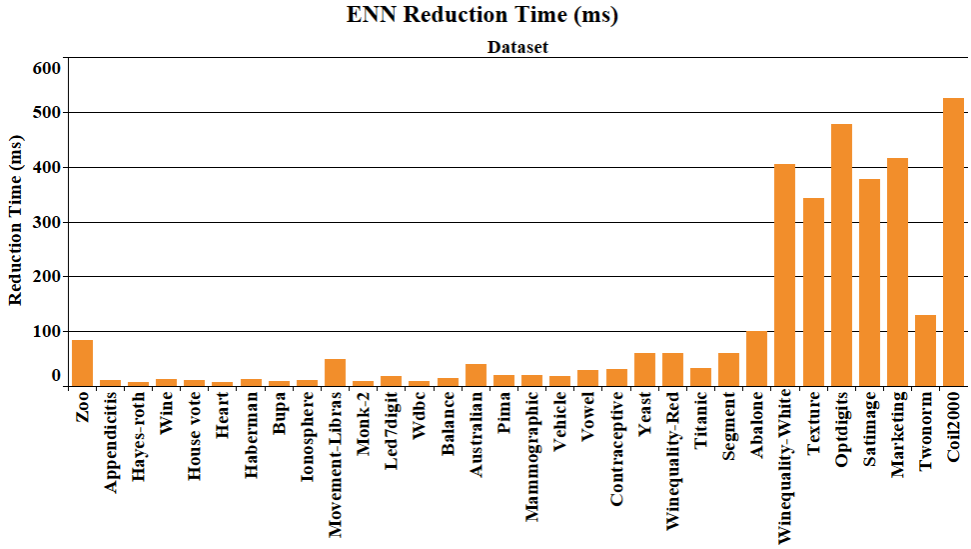


**Figure 4.1:** ENN Reduction Time (ms)

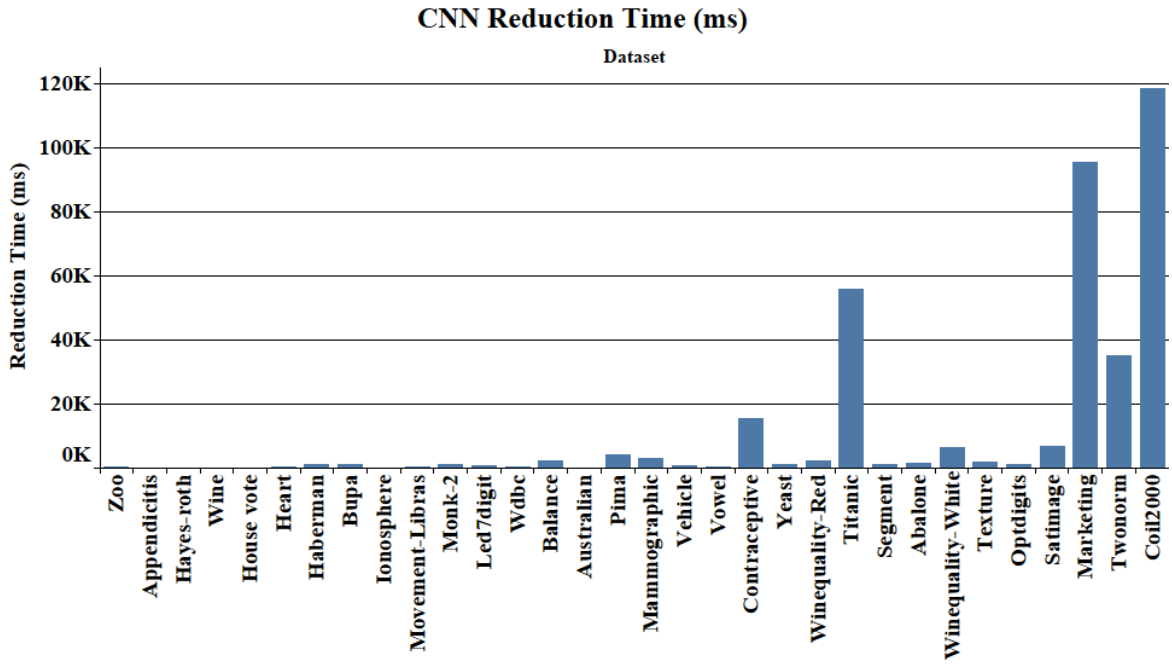


**Figure 4.2:** CNN Reduction Time (ms)

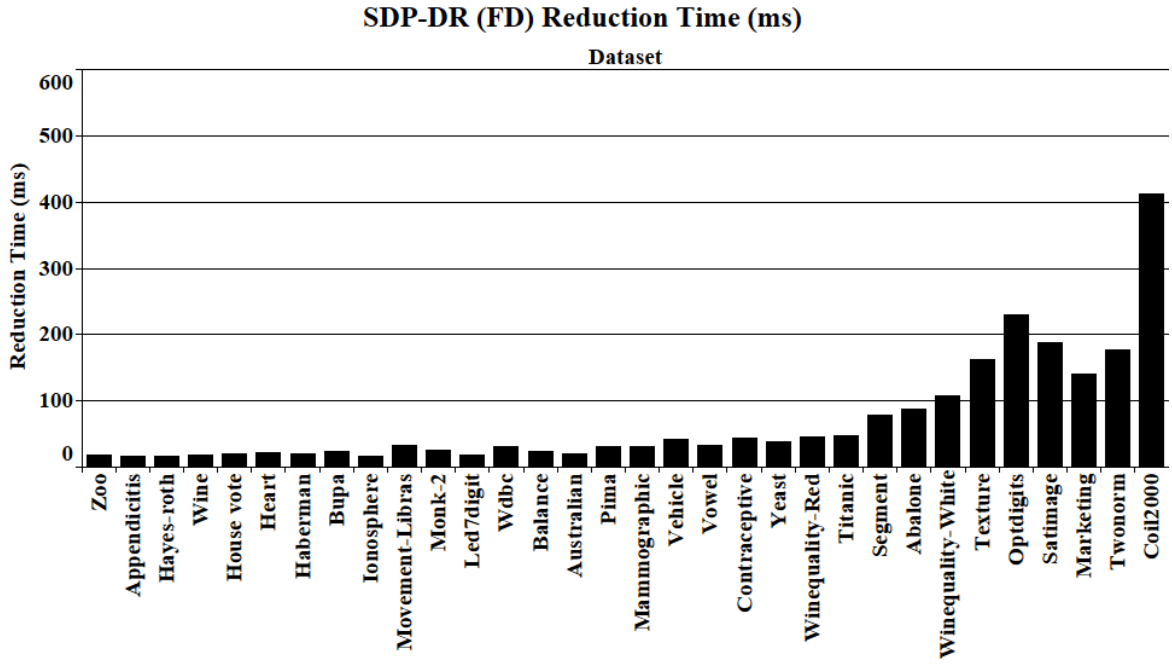


**Figure 4.3:** SDP-DR (FD) Reduction Time (ms)

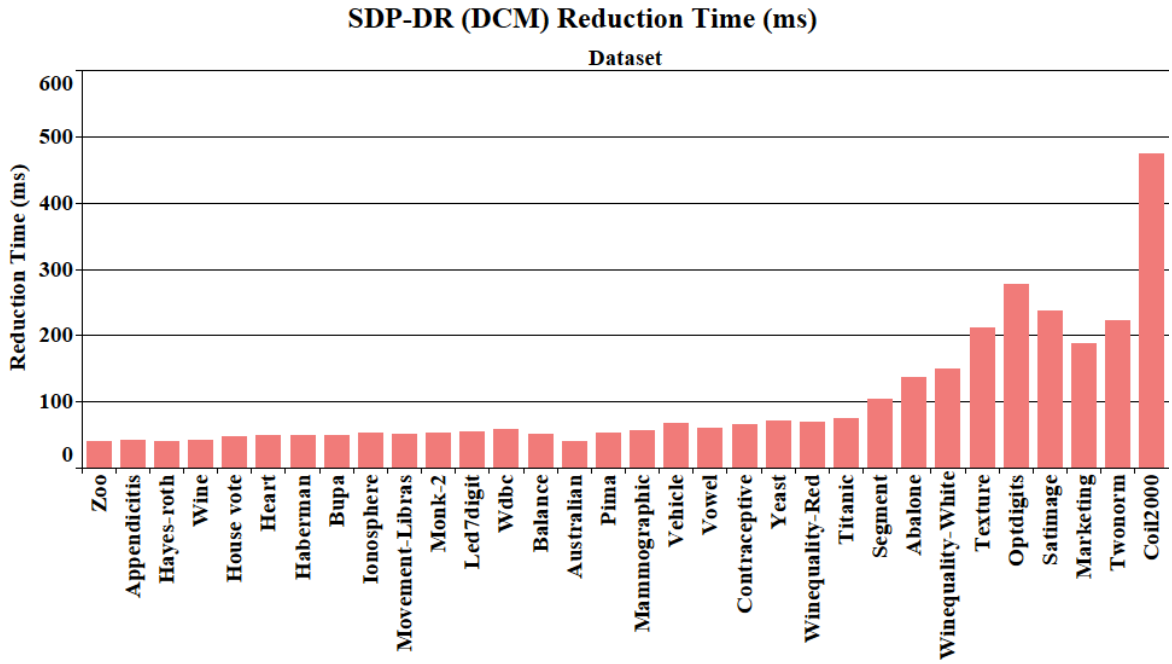


**Figure 4.4:** SDP-DR (DCM) Reduction Time (ms)

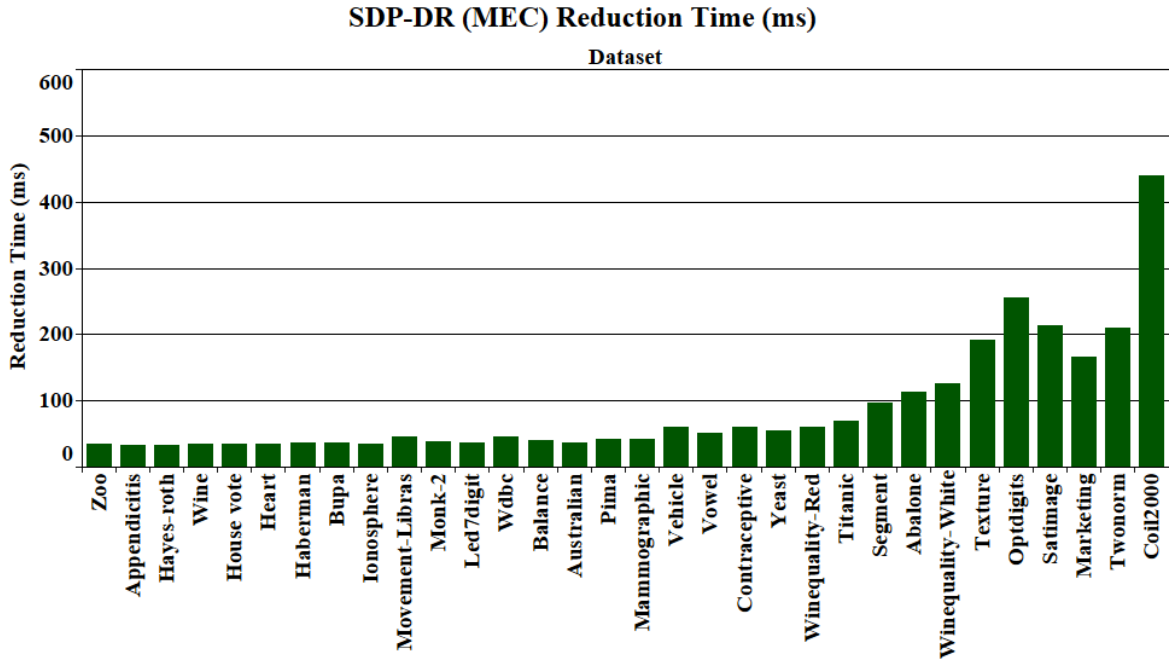


**Figure 4.5:** SDP-DR (MEC) Reduction Time (ms)

Referring to Table 4.2, which presents the mean reduction time across all datasets, it is observed that CNN has the highest average reduction time (11,230.63 ms), confirming its

inefficiency. ENN achieves an average reduction time of 106.76 ms, making it computationally faster than CNN but still not the most efficient. Among the SDP-DR methods, SDP-DR (FD) records the lowest mean reduction time at 69.41 ms, followed by SDP-DR (MEC) at 87.59 ms, and SDP-DR (DCM) at 101.20 ms. While these results suggest that the proposed SDP-DR methods are computationally efficient, the mean reduction time alone does not establish statistical significance. Therefore, further statistical analysis is required to determine whether the observed differences in reduction time are significant. Table 4.2 summarizes the full results presented in Appendix B, Table B.1.

**Table 4.2:** Average of Mean Reduction Time

| Dataset | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|
| **AVERAGE** | 106.76 | 11230.63 | **69.41** | 101.20 | 87.59 |

To statistically validate the observed reduction times, Table 4.3 presents the results of the Sign Test (*ps*) and Wilcoxon Test (*pw*), where a p-value below 0.05 in both tests indicates a statistically significant difference. The analysis confirms that CNN differs significantly from all other methods ($p < 0.00001$ for both tests), reaffirming that it is the slowest method and consistently outperformed in terms of reduction time. When comparing ENN with the SDP-DR variants, the results show that SDP-DR (FD) and ENN are not significantly different ($ps = 0.28884$, $pw = 0.58920$), despite SDP-DR (FD) achieving a lower mean reduction time. This suggests that, although SDP-DR (FD) appears more efficient, its reduction time is not statistically superior to that of ENN. Similarly, SDP-DR (DCM) ($ps = 0.00146$, $pw = 0.22246$) and SDP-DR (MEC) ($ps = 0.07710$, $pw = 0.33204$) do not exhibit statistically significant differences when compared with ENN, indicating that their computational efficiency is comparable.

Among the SDP-DR variants, the statistical tests confirm that SDP-DR (FD) is significantly faster than both SDP-DR (DCM) ($ps < 0.00001$, $pw < 0.00001$) and SDP-DR (MEC) ($ps < 0.00001$, $pw < 0.00001$). These findings establish SDP-DR (FD) as the most computationally efficient variant within the SDP-DR framework. While SDP-DR (FD) achieves the lowest average reduction time (69.41 ms), its difference from ENN is not statistically significant, indicating that SDP-DR(FD) efficiency cannot be definitively considered superior to ENN. However, its statistically significant advantage over SDP-DR (DCM) and SDP-DR (MEC) reinforces its position as the most efficient method within the SDP-DR approach.

**Table 4.3:** Sign and Wilcoxon Tests for Mean Reduction Time

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | |
|---|---|---|---|---|---|---|
| | | | CNN | FD | DCM | MEC |
| Mean Accuracy | ENN | *ss* | ***00/00/32** | 13/00/19 | 07/00/25 | 11/00/21 |
| | | *ps* | < .00001 | 0.28884 | 0.00146 | 0.07710 |
| | | *pw* | < .00001 | 0.58920 | 0.22246 | 0.33204 |
| | CNN | *ss* | | ****32/00/00** | ****32/00/00** | ****32/00/00** |
| | | *ps* | | < .00001 | < .00001 | < .00001 |
| | | *pw* | | < .00001 | < .00001 | 0.00026 |
| | FD | *ss* | | | ***00/00/32** | ***00/00/32** |
| | | *ps* | | | < .00001 | < .00001 |
| | | *pw* | | | < .00001 | < .00001 |
| | DCM | *ss* | | | | ****32/00/00** |
| | | *ps* | | | | < .00001 |
| | | *pw* | | | | < .00001 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

### 4.2.4 Evaluation of Mean Reduction Rate

The graphical representation of the relationship between reduction time and storage for different algorithms is shown in Figure 4.6 to Figure 4.10, providing insights into how reduction time correlates with the remaining data storage after applying prototype selection techniques. Ideally, a point closer to the origin indicates a more efficient method, where the

data reduction process is completed rapidly and the remaining storage requirement is minimal. The storage percentage (%) is calculated using the following formula:

$$Storage\ (\%) = \ 100 - reduction\ rate \qquad \text{Equation 4.1}$$

Figure 4.6 to Figure 4.10 present the relationship between reduction time and storage for ENN, CNN, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC). The graphs reveal that CNN achieves a lower percentage of storage, indicating that it removes a substantial portion of the data. However, this comes at the cost of significantly longer reduction time compared to the other methods, making it computationally inefficient. In contrast, the SDP-DR variants offer a better trade-off, achieving competitive storage percentages while requiring significantly less computational time. Among them, SDP-DR (FD) consistently demonstrates the lowest reduction time while maintaining effective storage reduction, making it a promising alternative for balancing storage efficiency and computational cost.

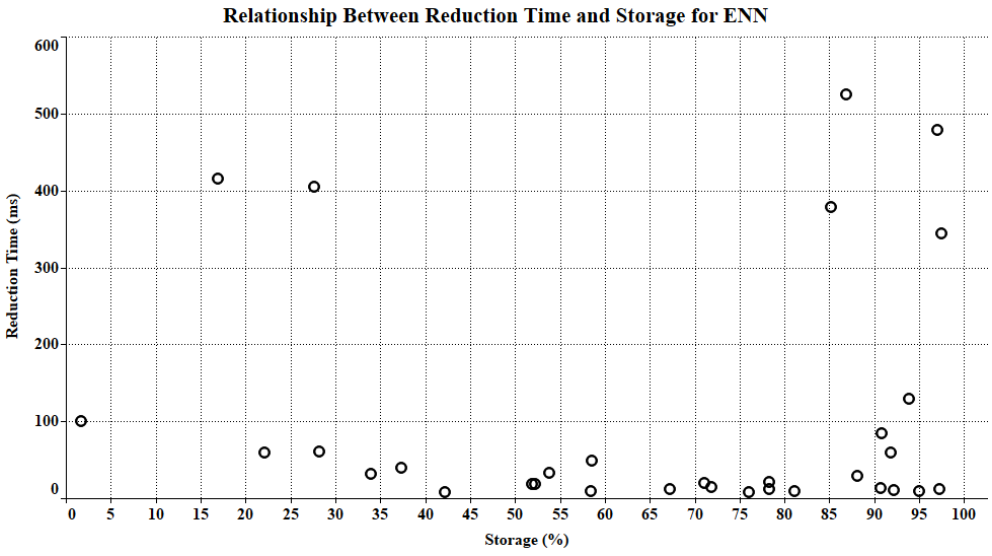


**Figure 4.6:** Relationship between Reduction Time and Storage for ENN

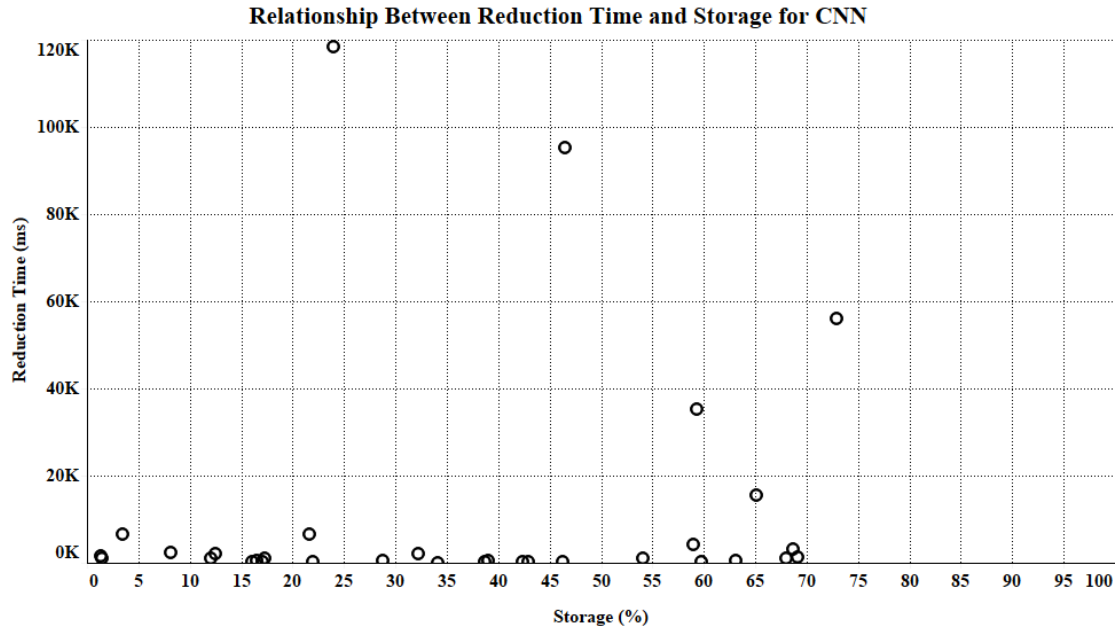


**Figure 4.7:** Relationship between Reduction Time and Storage for CNN

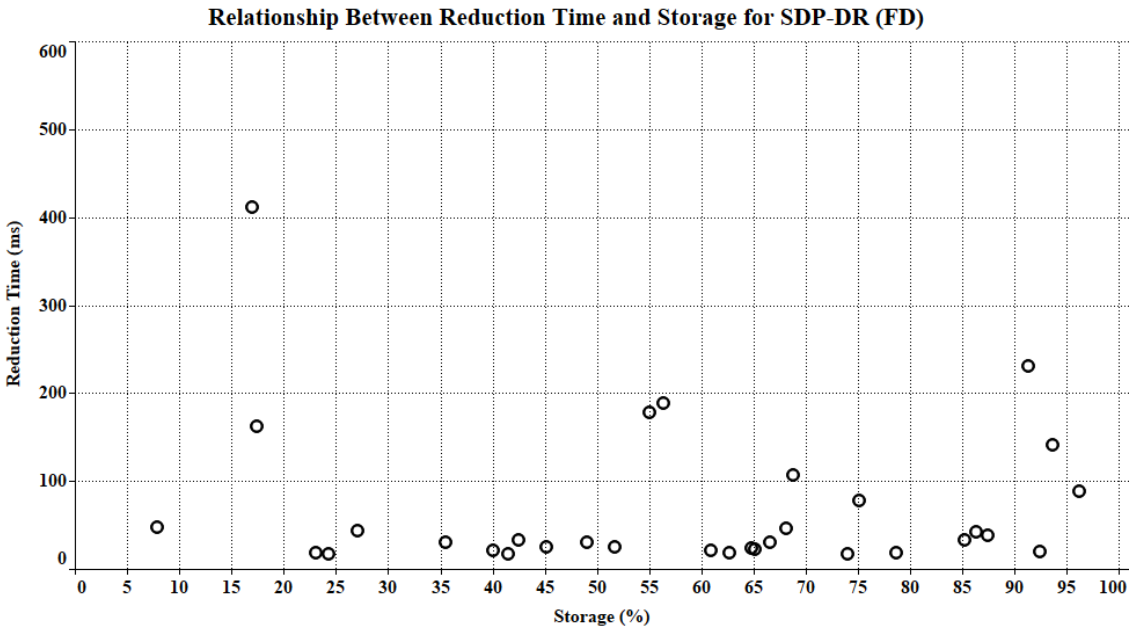


**Figure 4.8:** Relationship between Reduction Time and Storage for SDP-DR (FD)

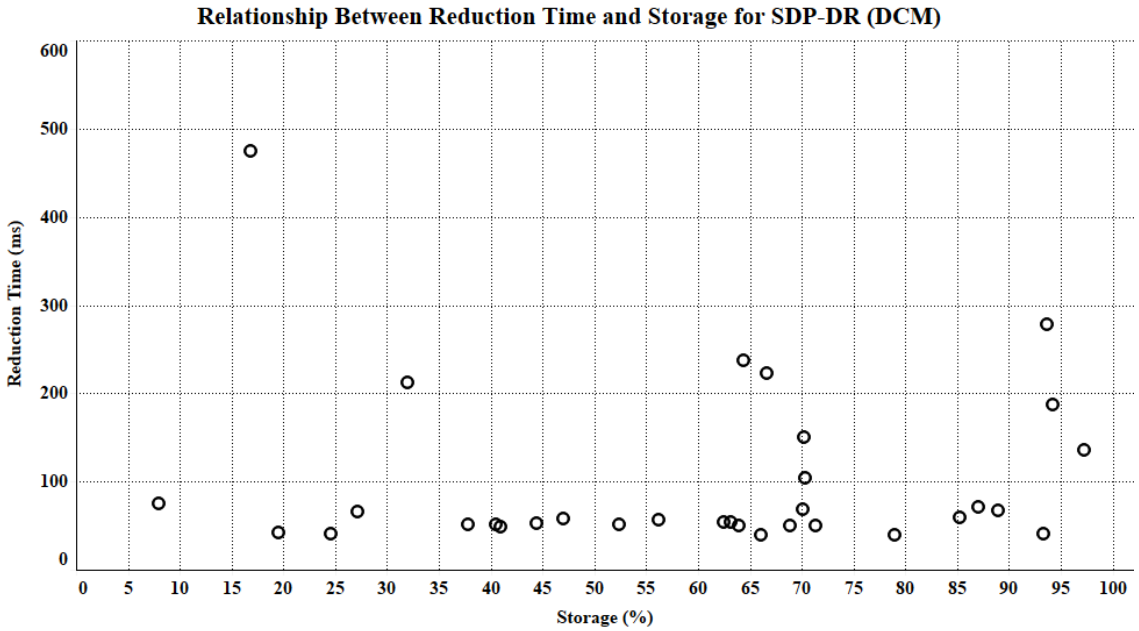


**Figure 4.9:** Relationship between Reduction Time and Storage for SDP-DR (DCM)

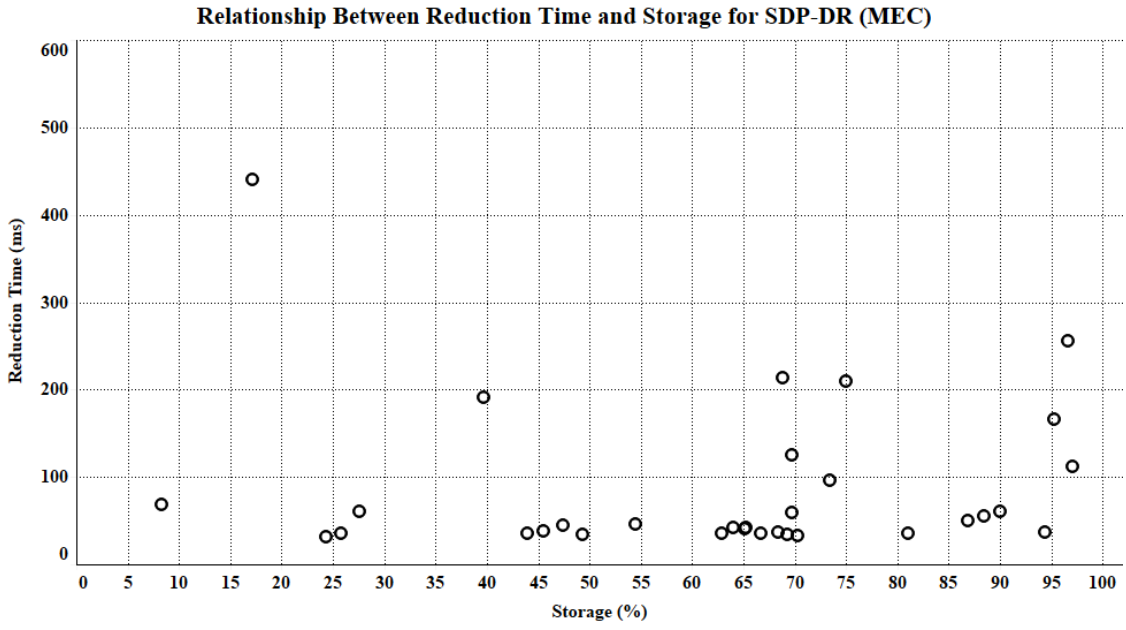


**Figure 4.10:** Relationship between Reduction Time and Storage for SDP-DR (MEC)

Examining Table 4.4, which presents the mean reduction rate across 32 datasets, notable differences in reduction efficiency can be observed. CNN achieves the highest average reduction rate (63.66%), indicating that it removes a substantial amount of data.

However, this high reduction rate comes at the cost of excessive computational time, making CNN impractical for large-scale applications. In contrast, ENN has the lowest average reduction rate (33.90%), meaning it retains more data than CNN and the SDP-DR variants. While ENN is computationally faster than CNN, its low reduction rate suggests that it is less effective at reducing storage requirements. Among the SDP-DR variants, SDP-DR (FD) achieves the highest reduction rate (42.17%), followed by SDP-DR (DCM) (40.53%) and SDP-DR (MEC) (37.47%). These results suggest that the proposed SDP-DR methods effectively reduce data while maintaining lower reduction times compared to CNN. Table 4.4 summarizes the full results presented in Appendix B, Table B.2. However, since the reduction rate alone does not establish statistical superiority, further analysis through significance testing is necessary.

**Table 4.4:** Average of Mean Reduction Rate Across 32 Datasets

| Dataset | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|
| **AVERAGE** | 33.90 | **63.66** | 42.17 | 40.53 | 37.47 |

To statistically validate these findings, Table 4.5 presents the results of the Sign Test (*ps*) and Wilcoxon Test (*pw*), where a p-value below 0.05 in both tests indicates a statistically significant difference. The results confirm that CNN is significantly different from SDP-DR (DCM) ($ps = 0.00468$, $pw = 0.00398$) and SDP-DR (MEC) ($ps = 0.01333$, $pw = 0.00214$), indicating that CNN achieves a higher reduction rate than these two methods with strong statistical significance. However, the difference between CNN and SDP-DR (FD) is not statistically significant ($ps = 0.15730$, $pw = 0.01510$), suggesting that SDP-DR (FD) achieves a reduction rate comparable to CNN, but with significantly lower reduction time.

When comparing ENN to the SDP-DR variants, the results indicate statistical differences between ENN and SDP-DR (FD) ($ps$ = 0.03389, $pw$ = 0.19706), ENN and SDP-DR (DCM) ($ps$ = 0.03389, $pw$ = 0.23014), and ENN and SDP-DR (MEC) ($ps$ = 0.03389, $pw$ = 0.33204). Although the Sign Test ($ps$) suggests significance, the Wilcoxon Test ($pw$) values are greater than 0.05, indicating that these differences do not meet the standard criteria for statistical significance, which require both $ps$ and $pw$ to be below 0.05.

Among the SDP-DR variants, SDP-DR (FD) is statistically superior to both SDP-DR (DCM) ($ps$ = 0.00706, $pw$ = 0.03486) and SDP-DR (MEC) ($ps$ = 0.00002, $pw < 0.00001$), confirming that it achieves significantly better data reduction. Additionally, SDP-DR (DCM) is statistically superior to SDP-DR (MEC) ($ps$ = 0.00468, $pw$ = 0.00038), reinforcing its advantage in reduction rate. These findings confirm that the proposed SDP-DR methods, particularly SDP-DR (FD), achieve significantly better data reduction than ENN while maintaining a computational advantage over CNN. Although CNN exhibits a high reduction rate, its excessive reduction time makes it less practical for large-scale applications. Among the SDP-DR variants, SDP-DR (FD) consistently outperforms both SDP-DR (DCM) and SDP-DR (MEC) in terms of reduction rate, with statistically significant differences.

**Table 4.5:** Sign and Wilcoxon Tests for Mean Reduction Rate

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | |
|---|---|---|---|---|---|---|
| | | | CNN | FD | DCM | MEC |
| Mean Accuracy | ENN | *ss* | ****04/00/28** | 10/00/22 | 10/00/22 | 10/00/22 |
| | | *ps* | < .00001 | 0.03389 | 0.03389 | 0.03389 |
| | | *pw* | 0.00002 | 0.19706 | 0.23014 | 0.33204 |
| | CNN | *ss* | | 20/00/12 | ***24/00/08** | ***23/00/09** |
| | | *ps* | | 0.15730 | 0.00468 | 0.01333 |
| | | *pw* | | 0.01510 | 0.00398 | 0.00214 |
| | FD | *ss* | | | ****23/01/08** | ****28/00/04** |
| | | *ps* | | | 0.00706 | 0.00002 |
| | | *pw* | | | 0.03486 | < .00001 |
| | DCM | *ss* | | | | ****24/00/08** |
| | | *ps* | | | | 0.00468 |
| | | *pw* | | | | 0.00038 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

### 4.2.5 Evaluation of Mean Accuracy

The classification accuracy of the SDP-DR variants (FD, DCM, MEC) was evaluated against k-NN, ENN, and CNN across different values of $k$ ($k$ = 1, 3, 5, 7) to analyse how prototype selection techniques affect model performance as $k$ varies. The results, presented in Table 4.6 to Table 4.13, provide insights into how accuracy changes with different dataset reduction approaches and highlight the effectiveness of the proposed SDP-DR framework in preserving classification performance while reducing data.

Classification accuracy represents the proportion of correct group assignments when comparing various techniques on a dataset that is naturally divided into subsets. This is measured by evaluating how accurately new observations are classified using a rule developed from a labelled training dataset (Pizer et al., 2020). In our experiment, we use stratified 10-fold cross-validation; therefore, the mean accuracy is used to indicate the overall accuracy of the algorithm.

For $k = 1$ (Table 4.6), SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) achieve higher accuracy (69.84%, 70.28%, and 70.26%, respectively) than k-NN (68.02%) and significantly outperform CNN (64.46%) and ENN (63.80%). These results indicate that SDP-DR effectively reduces data while preserving accuracy, unlike CNN, which removes too many instances and thus degrades performance. Table 4.6 summarizes the full results presented in Appendix B, Table B.3. The statistical analysis (Table 4.7) shows that k-NN significantly outperforms both ENN ($ps = 0.01333$, $pw = 0.01108$) and CNN ($ps = 0.03389$, $pw = 0.00758$), confirming CNN's poor classification performance due to excessive data reduction. When comparing k-NN with the SDP-DR variants, no statistically significant differences are observed, as all p-values exceed 0.05 ($ps > 0.05$, $pw > 0.05$). This suggests that SDP-DR successfully reduces data while maintaining classification accuracy comparable to k-NN. However, when comparing k-NN with SDP-DR (DCM), the win/draw/loss record is 14/02/16, indicating that k-NN wins 14 times, draws twice, and loses 16 times against SDP-DR (DCM). This suggests that SDP-DR (DCM) provides a slight advantage over k-NN in classification accuracy, further reinforcing its effectiveness in maintaining performance while reducing dataset size. Additionally, ENN performs significantly worse than SDP-DR (DCM) ($ps = 0.01955$, $pw = 0.02088$), SDP-DR (MEC) ($ps = 0.01333$, $pw = 0.00350$), and SDP-DR (FD) ($ps = 0.01333$, $pw = 0.00350$), confirming the superiority of SDP-DR over ENN. CNN also performs significantly worse than all SDP-DR methods ($ps < 0.05$, $pw < 0.05$), further reinforcing its limitations in classification accuracy. These results suggest that SDP-DR methods effectively preserve classification accuracy while reducing dataset size, making them superior to ENN and CNN.

**Table 4.6:** Average of Mean Classification Accuracy for k-NN ($k = 1$)

| | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 68.02 | 63.80 | 64.46 | 69.84 | **70.28** | 70.26 |

**Table 4.7:** Sign and Wilcoxon Tests for Mean Accuracy of k-NN ($k = 1$)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | ****23/0/09** | ****22/00/10** | 17/01/014 | 14/2/016 | 17/01/014 |
| | | *ps* | 0.01333 | 0.03389 | 0.59001 | 0.71500 | 0.59001 |
| | | *pw* | 0.01108 | 0.00758 | 0 .87288 | 0.36282 | 0.51570 |
| | ENN | *ss* | | 18/00/14 | ***09/01/022** | ***09/00/23** | ***09/00/23** |
| | | *ps* | | 0.47950 | 0.01955 | 0.01333 | 0.01333 |
| | | *pw* | | 0.77948 | 0.02088 | 0.00350 | 0.00350 |
| | CNN | *ss* | | | ***08/00/024** | ***08/00/024** | ***07/00/025** |
| | | *ps* | | | 0.00468 | 0.00468 | 0.00146 |
| | | *pw* | | | 0.00452 | 0.00168 | 0.00128 |
| | FD | *ss* | | | | 11/02/019 | 11/01/020 |
| | | *ps* | | | | 0.14413 | 0.10600 |
| | | *pw* | | | | 0.17068 | 0.23014 |
| | DCM | *ss* | | | | | 14/02/016 |
| | | *ps* | | | | | 0.71500 |
| | | *pw* | | | | | 0.60306 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

For $k = 3$ (Table 4.8), SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) continue to demonstrate strong performance (69.97%, 70.62%, and 71.17%, respectively), exceeding the accuracy of k-NN (69.54%), ENN (64.86%), and CNN (61.43%). The consistency in accuracy across different SDP-DR variants suggests that the proposed method is effective in data reduction without compromising classification accuracy. Table 4.8 summarizes the full results presented in Appendix B, Table B.4. In the statistical analysis (Table 4.9), a similar pattern is observed. k-NN continues to significantly outperform CNN ($ps = 0.00010$, $pw = 0.00034$), reaffirming CNN's inefficiency in classification accuracy. Additionally, k-NN is significantly better than ENN in both the Sign Test and Wilcoxon Test ($ps = 0.01955$, $pw = 0.03940$), confirming that k-NN provides superior classification accuracy compared to ENN.

When comparing k-NN with the SDP-DR variants (FD, DCM, MEC), no statistically significant differences are observed ($ps > 0.05$, $pw > 0.05$), demonstrating that SDP-DR methods effectively retain crucial classification information while reducing dataset size. Furthermore, ENN is significantly worse than SDP-DR (MEC) ($ps = 0.03389$, $pw = 0.00480$), suggesting that SDP-DR (MEC) provides the strongest improvement over ENN. CNN continues to perform significantly worse than all SDP-DR variants ($ps < 0.05$, $pw < 0.05$), confirming its unstable and unreliable performance. These results highlight that SDP-DR methods maintain strong classification performance while achieving meaningful data reduction, making them an optimal choice for efficient prototype selection.

**Table 4.8:** Average of Mean Classification Accuracy for k-NN ($k = 3$)

| | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 69.54 | 64.86 | 61.43 | 69.97 | 70.62 | **71.17** |

**Table 4.9:** Sign and Wilcoxon Tests for Mean Accuracy of k-NN ($k = 3$)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | ****22/00/10** | ****27/00/05** | 20/01/011 | 18/01/013 | 16/01/015 |
| | | *ps* | 0.03389 | 0.00010 | 0.10600 | 0.36917 | 0.85746 |
| | | *pw* | 0.02260 | 0.00034 | 0.16452 | 0.71138 | 0.88866 |
| | ENN | *ss* | | ****22/01/009** | 11/00/021 | 11/00/021 | ***10/00/022** |
| | | *ps* | | 0.01955 | 0.07710 | 0.07710 | 0.03389 |
| | | *pw* | | 0.03940 | 0.05614 | 0.01596 | 0.00480 |
| | CNN | *ss* | | | ***06/01/25** | ***05/00/27** | ***06/00/26** |
| | | *ps* | | | 0.00064 | 0.00010 | 0.00041 |
| | | *pw* | | | 0.00080 | 0.00038 | 0.00034 |
| | FD | *ss* | | | | ***09/02/21** | ***6/01/25** |
| | | *ps* | | | | 0.02846 | 0.00064 |
| | | *pw* | | | | 0.06288 | 0.00208 |
| | DCM | *ss* | | | | | 9/03/20 |
| | | *ps* | | | | | 0.04109 |
| | | *pw* | | | | | 0.05876 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

For $k$ = 5 (Table 4.10), the SDP-DR variants maintain stable classification performance, with MEC achieving the highest accuracy (70.74%), followed by DCM (69.72%) and FD (69.55%), while k-NN records an accuracy of 68.40%. ENN (64.95%) remains below k-NN, and CNN (58.82%) continues to exhibit the lowest accuracy. These results indicate that SDP-DR methods remain effective in maintaining accuracy while reducing dataset size. Table 4.10 summarizes the full results presented in Appendix B, Table B.5. The statistical analysis (Table 4.11) confirms that k-NN significantly outperforms CNN ($ps$ = 0.00041, $pw$ = 0.00068), reinforcing CNN's poor performance. Additionally, k-NN is significantly better than ENN ($ps$ = 0.01955, $pw$ = 0.02926), confirming that ENN is weaker in classification accuracy. Comparisons between k-NN and SDP-DR variants (FD, DCM, MEC) show no statistically significant differences ($ps > 0.05$, $pw > 0.05$), demonstrating that SDP-DR methods retain essential classification information while reducing data. However, when comparing k-NN with SDP-DR (MEC), the win/loss record is 16/00/16, meaning that k-NN wins 16 times, loses 16 times, and has no draws against SDP-DR (MEC). This suggests that both methods perform similarly across datasets, further supporting the conclusion that SDP-DR (MEC) effectively preserves classification accuracy despite significant data reduction. Additionally, ENN is significantly worse than SDP-DR (MEC) ($ps$ = 0.02846, $pw$ = 0.00694), confirming that MEC provides a more effective balance between accuracy and reduction. CNN continues to perform significantly worse than all SDP-DR methods ($ps < 0.05$, $pw < 0.05$), demonstrating its instability in maintaining classification accuracy.

**Table 4.10:** Average of Mean Classification Accuracy for k-NN ($k$ = 5)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 68.40 | 64.95 | 58.82 | 69.55 | 69.72 | **70.74** |

**Table 4.11:** Sign and Wilcoxon Tests for Mean Accuracy of k-NN ($k$ = 5)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | ****22/01/009** | ****26/00/06** | 19/01/012 | 20/00/012 | 16/00/16 |
| | | *ps* | 0.01955 | 0.00041 | 0.20867 | 0.15730 | 1.00000 |
| | | *pw* | 0.02926 | 0.00068 | 0.68180 | 0.58920 | 0.71884 |
| | ENN | *ss* | | ****22/00/010** | ***11/00/021** | 12/00/020 | ***09/01/021** |
| | | *ps* | | 0.03389 | 0.07710 | 0.15730 | 0.02846 |
| | | *pw* | | 0.02574 | 0.02574 | 0.02852 | 0.00694 |
| | CNN | *ss* | | | ***07/00/25** | ***06/00/26** | ***05/00/27** |
| | | *ps* | | | 0.00146 | 0.00041 | 0.00010 |
| | | *pw* | | | 0.00086 | 0.00072 | 0.00034 |
| | FD | *ss* | | | | 14/02/16 | ***08/01/23** |
| | | *ps* | | | | 0.71500 | 0.00706 |
| | | *pw* | | | | 0.79486 | 0.00168 |
| | DCM | *ss* | | | | | ***07/01/24** |
| | | *ps* | | | | | 0.00226 |
| | | *pw* | | | | | 0.00438 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

For k = 7 (Table 4.12), the SDP-DR variants continue to perform well, with MEC achieving the highest accuracy (70.79%), followed by DCM (69.72%) and FD (69.38%). These values remain higher than k-NN (67.92%) and significantly outperform CNN (57.47%) and ENN (64.53%). These results indicate that SDP-DR ensures robust classification performance across different $k$ values, reinforcing its practicality for real-world applications. Table 4.12 summarizes the full results presented in Appendix B, Table B.6. The statistical analysis (Table 4.13) confirms that SDP-DR methods maintain high accuracy across varying $k$ values. k-NN significantly outperforms CNN ($ps$ = 0.01333, $pw$ = 0.00466) but does not differ significantly from ENN ($ps$ = 0.20867, $pw$ = 0.03400), suggesting that k-NN and ENN perform similarly at $k$ = 7. When comparing k-NN and the SDP-DR variants (FD, DCM, MEC), no statistically significant differences are observed ($ps$ > 0.05, $pw$ > 0.05), confirming that SDP-DR effectively reduces dataset size while maintaining classification accuracy. However, when comparing k-NN with SDP-DR (MEC), the win/draw/loss record is 14/01/17, meaning that k-NN wins 14 times, draws once, and loses

17 times against SDP-DR (MEC). This suggests that SDP-DR (MEC) provides a slight overall advantage over k-NN in terms of accuracy, reinforcing its effectiveness as a reliable prototype selection method. Additionally, ENN performs significantly worse than SDP-DR (MEC) ($ps$ = 0.01333, $pw$ = 0.00244), reinforcing the advantage of MEC over ENN. CNN again performs significantly worse than all SDP-DR variants ($ps$ < 0.05, $pw$ < 0.05), confirming its poor classification reliability. These findings indicate that SDP-DR methods maintain strong classification accuracy across different $k$ values, making them ideal for real-world applications that require prototype selection.

**Table 4.12:** Average of Mean Classification Accuracy for k-NN ($k$ = 7)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 67.92 | 64.53 | 57.47 | 69.38 | 69.72 | **70.79** |

**Table 4.13:** Sign and Wilcoxon Tests for Mean Accuracy of k-NN ($k$ = 7)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | 19/01/12 | ****23/00/09** | 15/03/14 | 15/01/16 | 14/01/17 |
| | | *ps* | 0.20867 | 0.01333 | 0.85268 | 0.85746 | 0.59001 |
| | | *pw* | 0.03400 | 0.00466 | 0.92828 | 0.62414 | 0.27134 |
| | ENN | *ss* | | ****22/01/09** | ***09/00/23** | ***10/01/21** | ***09/00/23** |
| | | *ps* | | 0.01955 | 0.01333 | 0.04819 | 0.01333 |
| | | *pw* | | 0.03400 | 0.00932 | 0.00804 | 0.00244 |
| | CNN | *ss* | | | ***06/00/26** | ***05/00/27** | ***05/00/27** |
| | | *ps* | | | 0.00041 | 0.00010 | 0.00010 |
| | | *pw* | | | 0.00124 | 0.00050 | 0.00042 |
| | FD | *ss* | | | | 14/02/16 | 11/03/19 |
| | | *ps* | | | | 1.00000 | 0.14413 |
| | | *pw* | | | | 0.91240 | 0.01828 |
| | DCM | *ss* | | | | | 10/02/20 |
| | | *ps* | | | | | 0.06789 |
| | | *pw* | | | | | 0.02320 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

The statistical results across all $k$ values confirm that the SDP-DR methods (FD, DCM, MEC) consistently outperform CNN ($ps$ < 0.05, $pw$ < 0.05) and maintain

classification accuracy comparable to k-NN ($ps > 0.05$, $pw > 0.05$while reducing dataset size. When comparing k-NN with SDP-DR (MEC), the win/draw/loss records are 16/00/16 for $k = 5$ and 14/01/17 for $k = 7$, showing a slight advantage for SDP-DR (MEC). Similarly, SDP-DR (DCM) achieves a 14/02/16 record against k-NN for $k = 1$, reinforcing its effectiveness in preserving classification performance. ENN shows inconsistent trends, often falling behind the SDP-DR methods, particularly MEC, which significantly outperforms ENN. (e.g., $ps = 0.02846$, $pw = 0.00694$ for $k = 5$, $ps = 0.01333$, $pw = 0.00244$ for $k = 7$). Meanwhile, CNN remains the least effective method, consistently performing worse than all SDP-DR variants ($ps < 0.05$, $pw < 0.05$). These findings establish SDP-DR as a reliable and computationally efficient alternative to traditional prototype selection techniques, offering both high classification accuracy and efficient data management.

### 4.2.6 Evaluation of Mean Classification Time

Figure 4.11 to Figure 4.16 illustrate the classification time (ms) for $k = 1$ across k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM), providing a comparison of computational efficiency among prototype selection methods. Figure 4.13 shows that CNN achieves the lowest classification time due to its aggressive data reduction strategy, but this comes at the cost of accuracy loss. SDP-DR (FD) and SDP-DR (DCM) maintain classification times that are lower than k-NN while retaining useful data for classification. In contrast, ENN demonstrates inconsistent performance, occasionally reducing classification time but not as efficiently as CNN or the SDP-DR variants. Figure 4.11 indicates that k-NN exhibits the highest classification time among all methods, confirming that direct classification without prototype selection leads to increased computational costs.

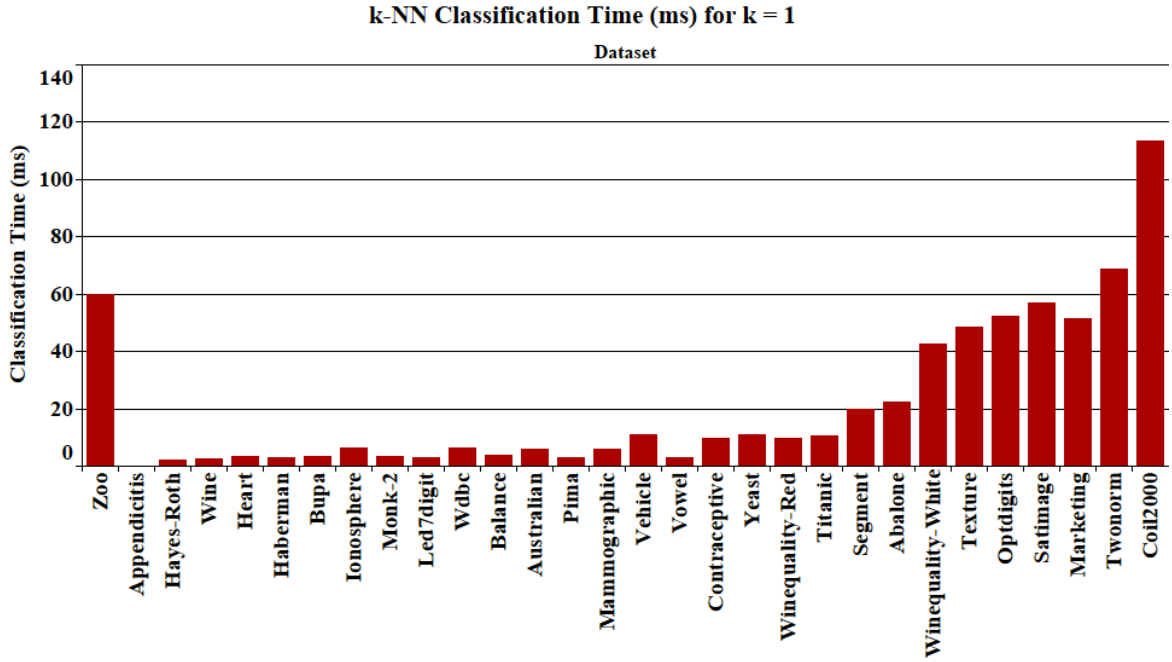


**Figure 4.11:** k-NN Classification Time (ms) for $k = 1$

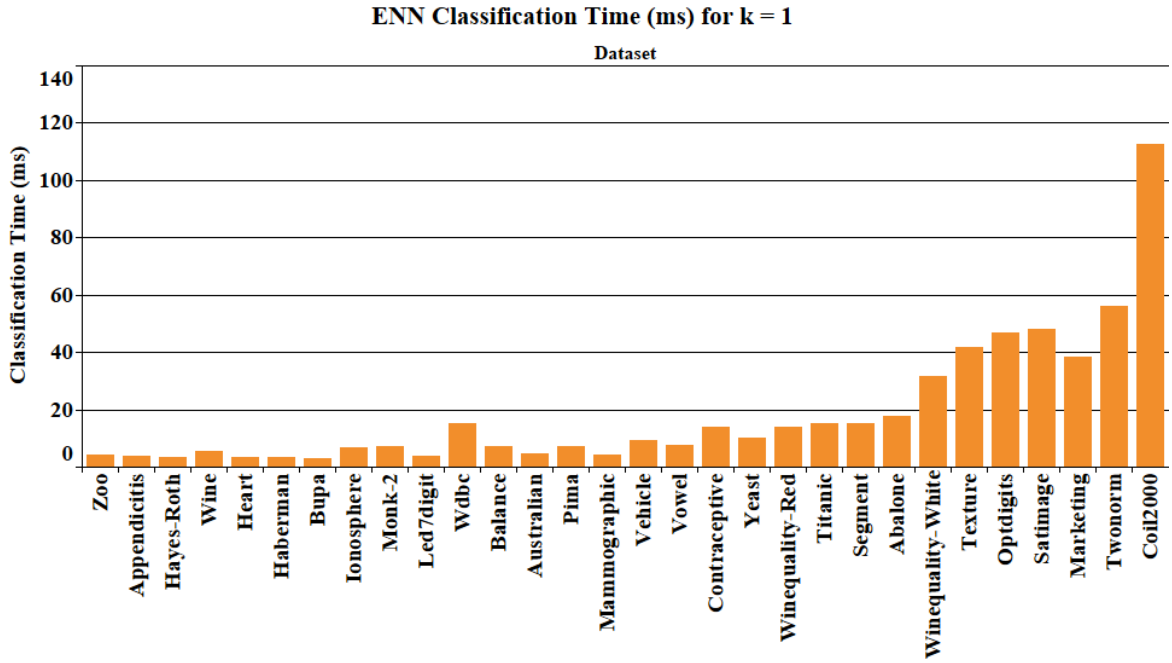


**Figure 4.12:** ENN Classification Time (ms) for $k = 1$

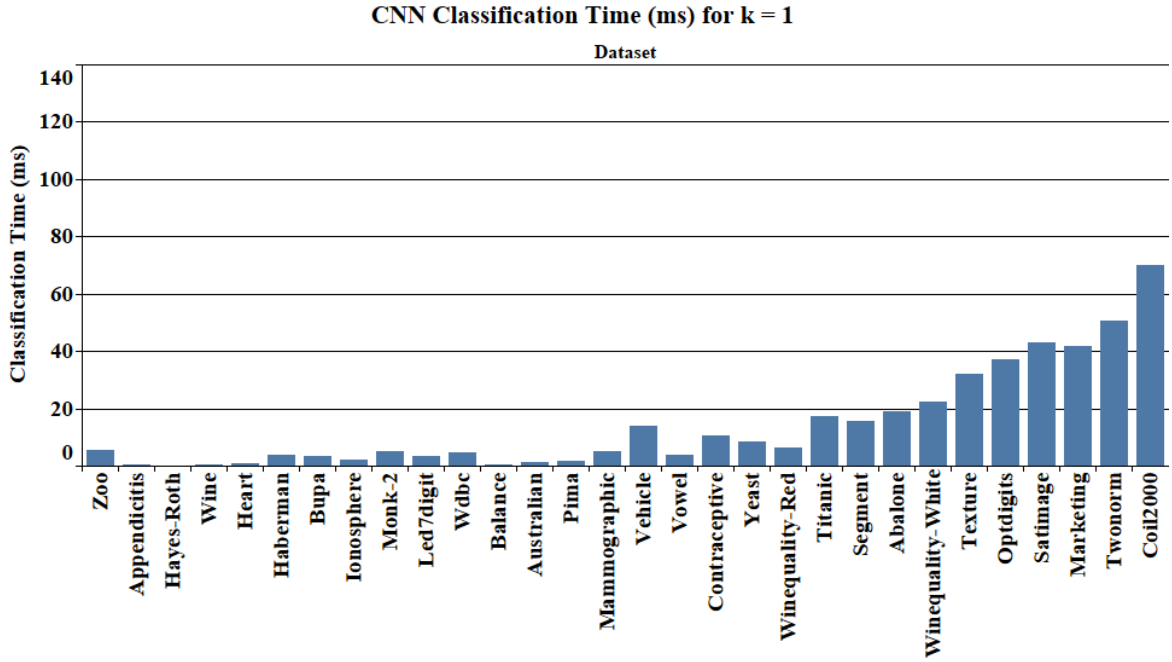


**Figure 4.13:** CNN Classification Time (ms) for $k = 1$

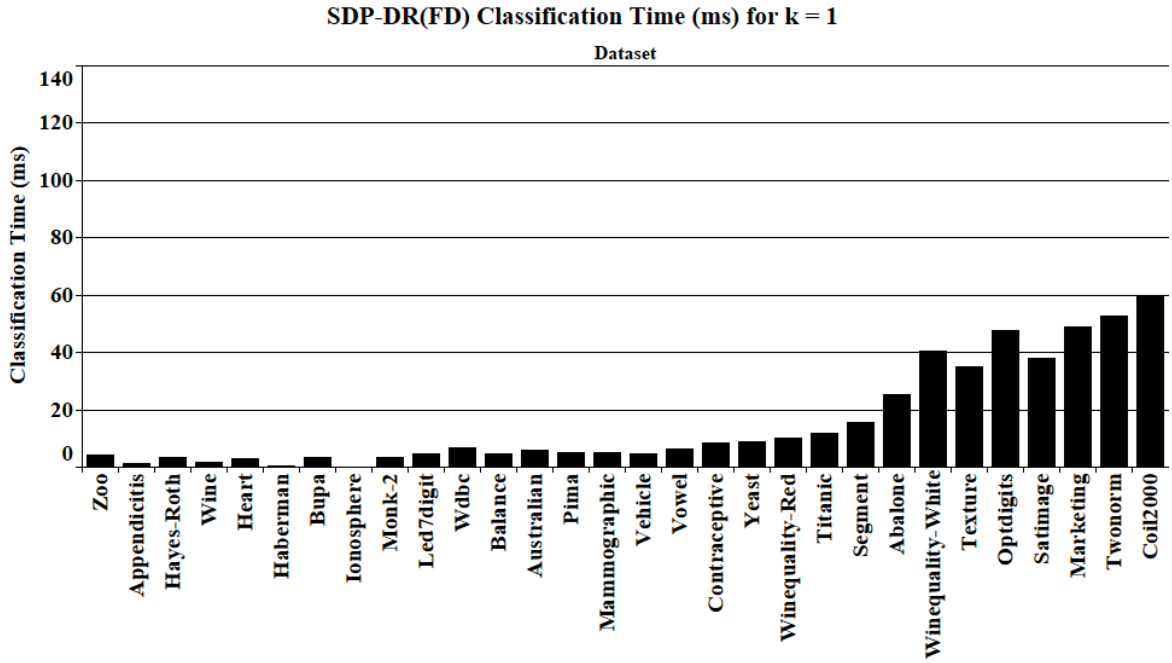


**Figure 4.14:** SDP-DR (FD) Classification Time (ms) for $k = 1$

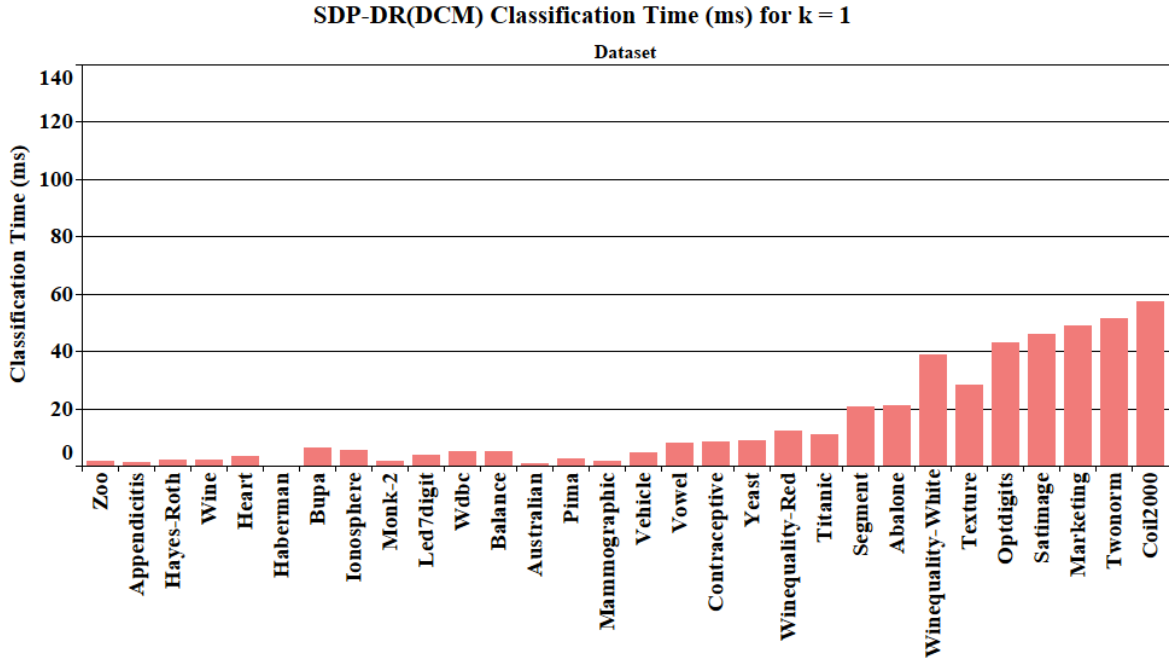


**Figure 4.15:** SDP-DR (DCM) Classification Time (ms) for $k = 1$

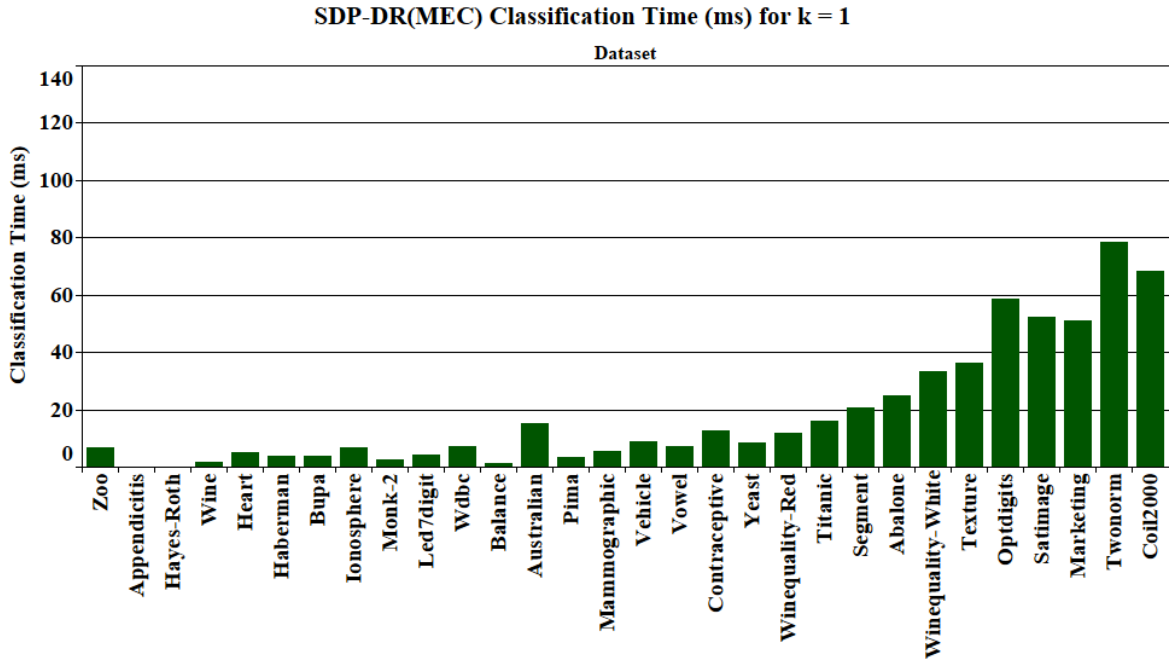


**Figure 4.16:** SDP-DR (MEC) Classification Time (ms) for $k = 1$

Table 4.14 presents the mean classification time across 32 datasets for k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC). The results indicate that k-NN has the

highest average classification time (20.46 ms), confirming its computational burden when processing the full dataset. CNN achieves the lowest classification time (13.79 ms), reinforcing its efficiency in reducing computational cost but at the expense of data loss. Among the SDP-DR methods, SDP-DR (DCM) records the lowest classification time (14.50 ms), making it the most computationally efficient among the SDP-DR variants. Table 4.14 summarizes the full results presented in Appendix B, Table B.7. Table 4.15 provides the statistical significance analysis for classification time using the Sign Test (*ps*) and Wilcoxon Test (*pw*), with significance confirmed if both values are below 0.05. The results indicate that k-NN is significantly slower than CNN (*ps* = 0.01333, *pw* = 0.00108), confirming CNN's computational advantage. SDP-DR (DCM) significantly reduces classification time compared to k-NN (*ps* = 0.03389, *pw* = 0.00758), reinforcing its effectiveness in optimising computational efficiency. Meanwhile, SDP-DR (MEC) and SDP-DR (FD) do not show significant differences from k-NN (*ps* > 0.05, *pw* > 0.05), indicating that they maintain classification times similar to k-NN while still benefiting from reduced dataset sizes. These findings confirm that SDP-DR (DCM) offers the best balance between computational efficiency and classification accuracy for $k = 1$, making it the optimal choice for reducing classification time in k-NN while preserving meaningful data.

**Table 4.14:** Average of Mean Classification Time for k-NN ($k = 1$)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 20.46 | 17.84 | **13.79** | 15.03 | 14.50 | 17.64 |

**Table 4.15:** Sign and Wilcoxon Tests for Mean Classification Time of k-NN ($k$ = 1)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | 17/00/15 | ****23/00/09** | 18/00/14 | ****22/00/10** | 15/00/17 |
| | | *ps* | 0.72367 | 0.01333 | 0.47950 | 0.03389 | 1.00000 |
| | | *pw* | 0.37886 | 0.00108 | 0.06148 | 0.00758 | 0.95216 |
| | ENN | *ss* | | ****22/00/10** | 21/00/11 | ****25/00/007** | 15/00/17 |
| | | *ps* | | 0.03389 | 0.07710 | 0.00146 | 0.72367 |
| | | *pw* | | 0.00116 | 0.03940 | 0.01352 | 0.98404 |
| | CNN | *ss* | | | 12/00/20 | 11/00/21 | ***07/01/24** |
| | | *ps* | | | 0.15730 | 0.07710 | 0.00226 |
| | | *pw* | | | 0.12114 | 0.36812 | 0.00124 |
| | FD | *ss* | | | | 19/00/13 | 11/00/21 |
| | | *ps* | | | | 0.28884 | 0.07710 |
| | | *pw* | | | | 0.14986 | 0.02574 |
| | DCM | *ss* | | | | | 12/00/20 |
| | | *ps* | | | | | 0.15730 |
| | | *pw* | | | | | 0.01352 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

Figure 4.17 to Figure 4.22 illustrate the classification time (ms) for $k$ = 3 across k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC), providing insights into computational efficiency among these prototype selection methods. The results show that CNN maintains the lowest classification time due to its aggressive instance reduction, but as previously noted, this comes at the cost of reduced classification accuracy. Meanwhile, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) effectively reduce classification time compared to k-NN while preserving critical data, confirming their balance between efficiency and accuracy. ENN, however, demonstrates variability in performance, reducing classification time in some cases but not as effectively as CNN or the SDP-DR variants. Additionally, k-NN exhibits the highest classification time, reaffirming that direct classification without prototype selection results in increased computational cost.

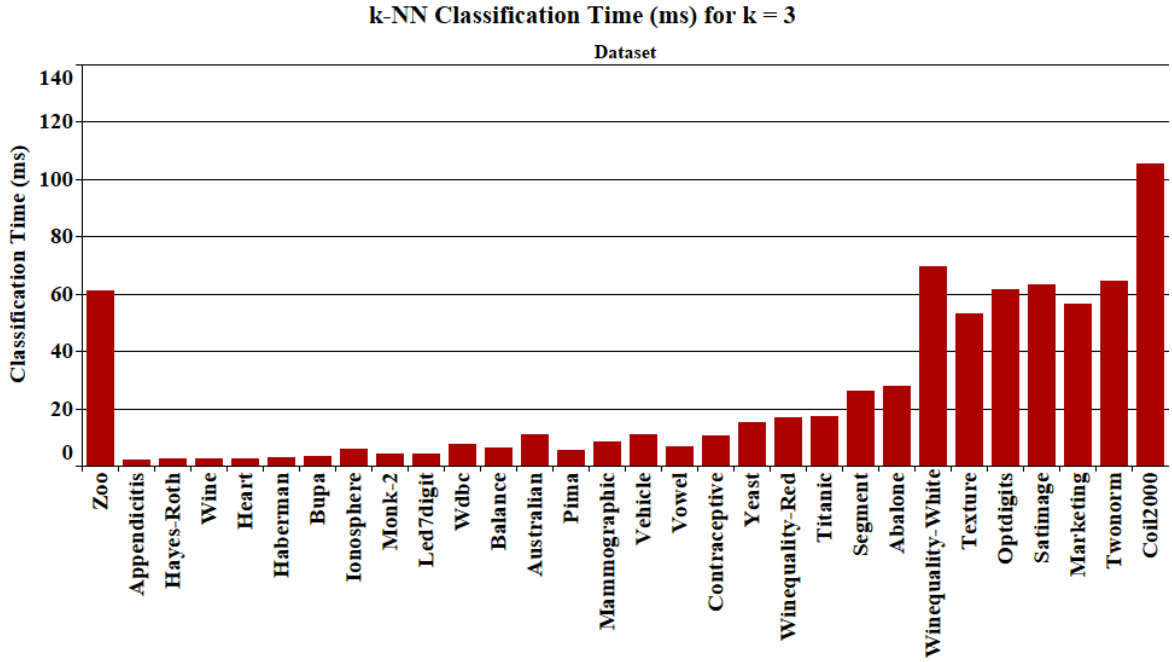


**Figure 4.17:** k-NN Classification Time (ms) for $k = 3$

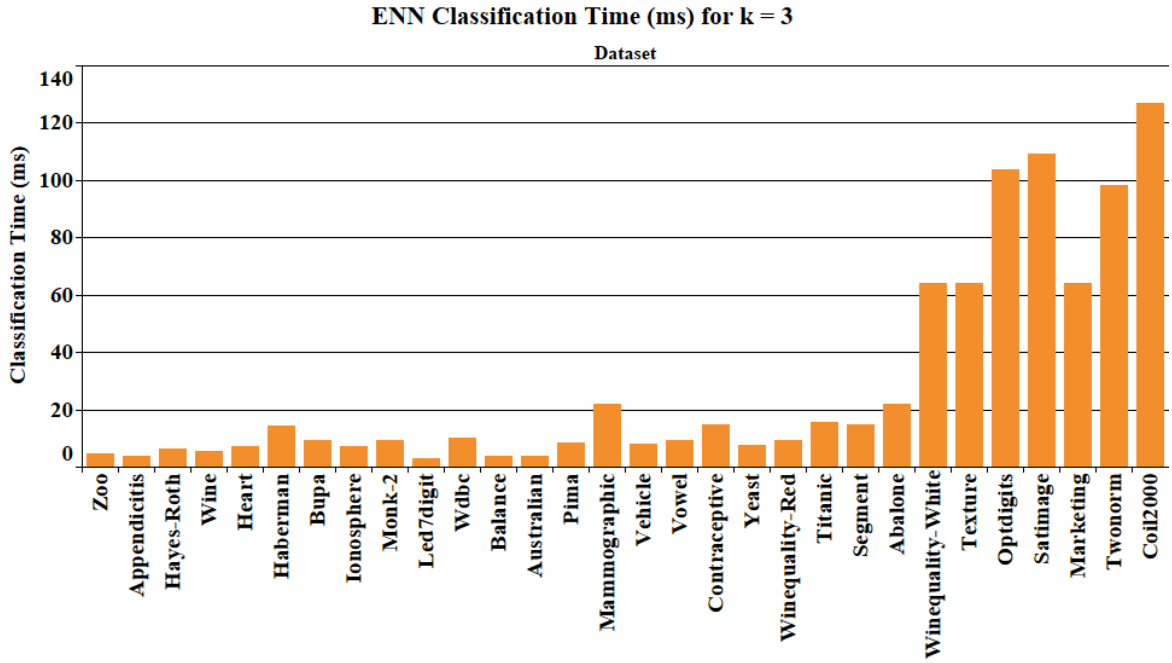


**Figure 4.18:** ENN Classification Time (ms) for $k = 3$

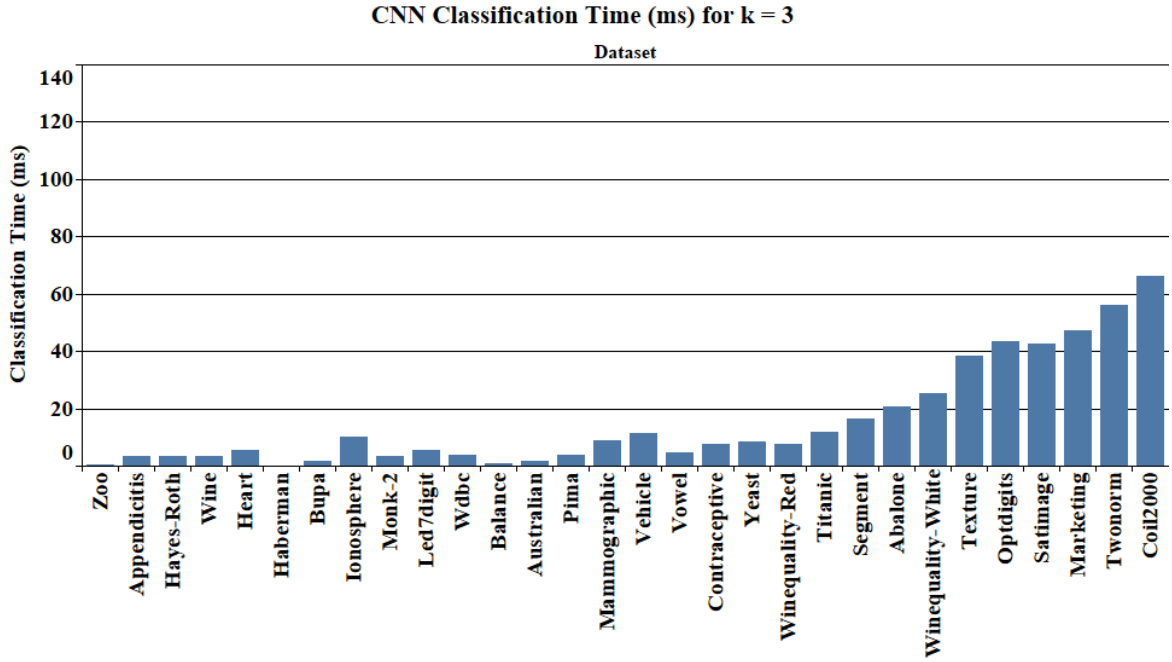


**Figure 4.19:** CNN Classification Time (ms) for $k = 3$

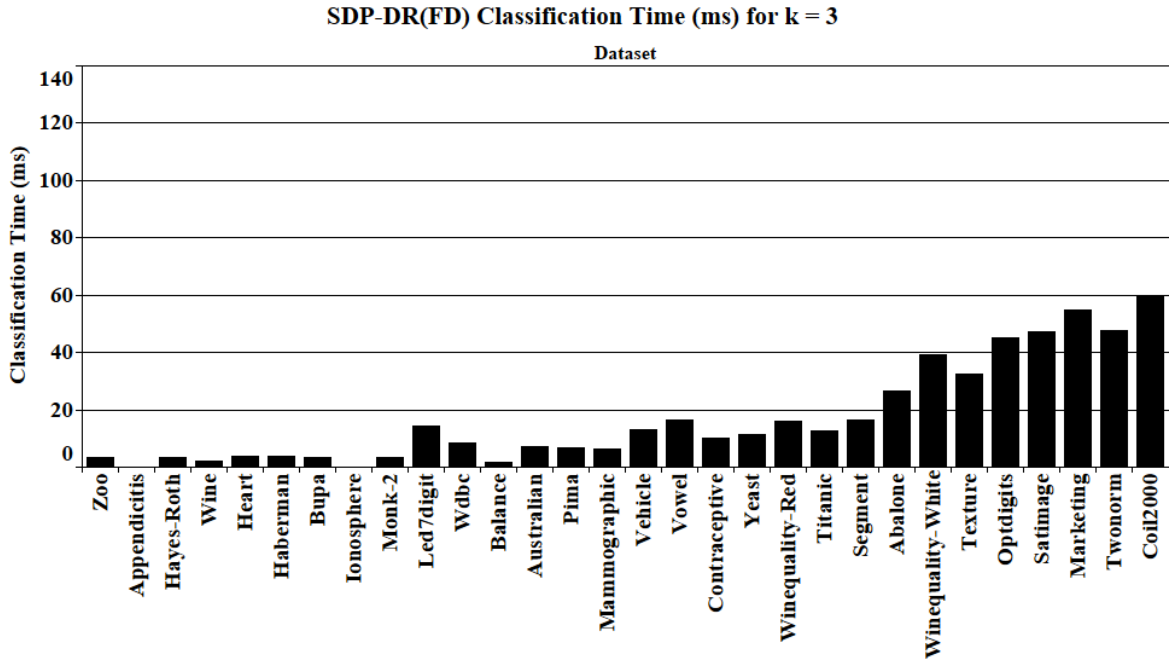


**Figure 4.20:** SDP-DR (FD) Classification Time (ms) for $k = 3$

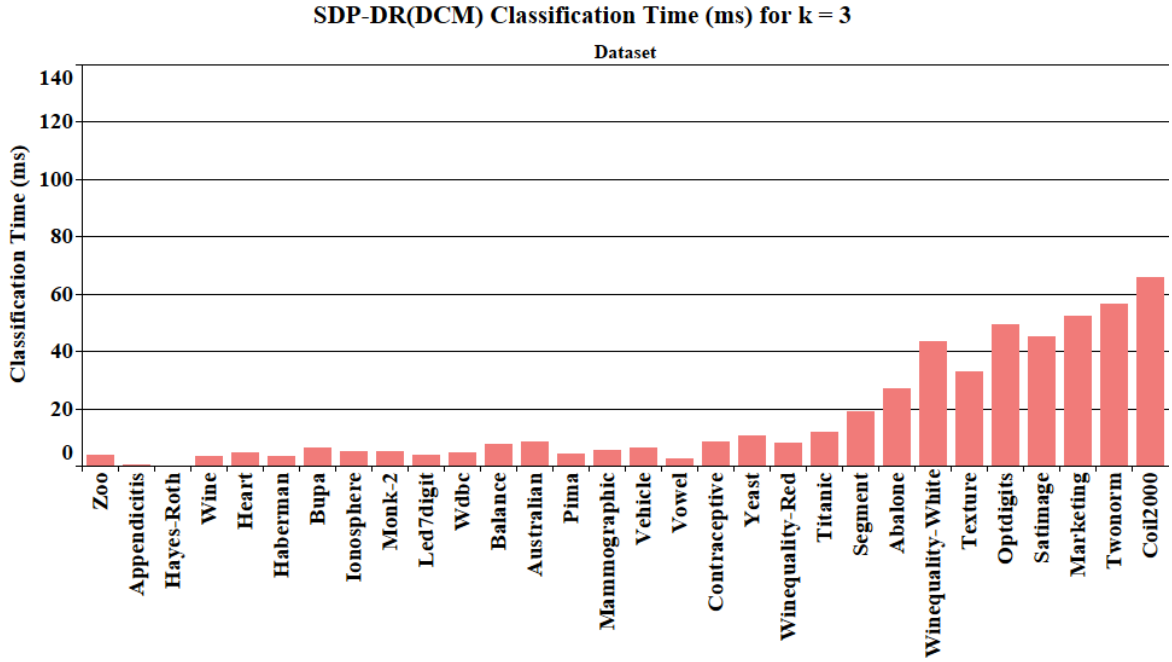


**Figure 4.21:** SDP-DR (DCM) Classification Time (ms) for $k = 3$

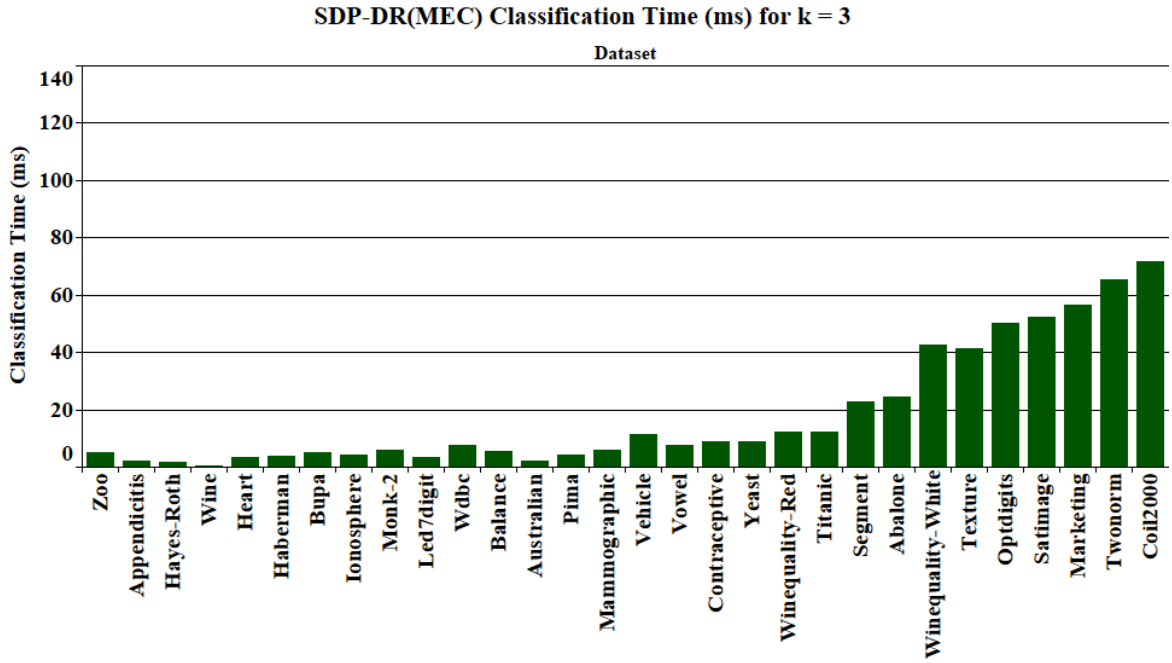


**Figure 4.22:** SDP-DR (MEC) Classification Time (ms) for $k = 3$

Table 4.16 presents the mean classification time across 32 datasets for k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC). The results indicate that k-NN continues

to have the highest average classification time (23.39 ms), highlighting its computational inefficiency when processing the full dataset. CNN achieves the lowest classification time (14.78 ms), confirming its efficiency in reducing computational cost but at the risk of sacrificing classification accuracy. Among the SDP-DR methods, SDP-DR (DCM) and SDP-DR (FD) record classification times of 16.20 ms and 16.38 ms, respectively, suggesting that these methods effectively optimise computational cost while maintaining classification performance. Meanwhile, SDP-DR (MEC) has a classification time of 17.35 ms, slightly higher than the other SDP-DR variants but still significantly lower than k-NN and ENN. Table 4.16 summarizes the full results presented in Appendix B, Table B.8. Table 4.17 provides the statistical significance analysis for classification time using the Sign Test (*ps*) and Wilcoxon Test (*pw*), where significance is confirmed if both values are below 0.05. The results confirm that k-NN is significantly slower than CNN (*ps* = 0.01333, *pw* = 0.00016), reinforcing CNN's computational efficiency in term of classification. Additionally, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) significantly reduce classification time compared to k-NN (*ps* = 0.01333, *pw* = 0.00214 for FD; *ps* = 0.00146, *pw* = 0.00022 for DCM; *ps* = 0.00146, *pw* = 0.00016 for MEC), validating their effectiveness in optimising computational performance. The statistical tests further reveal that ENN is significantly slower than all SDP-DR variants and CNN ($ps < 0.05$, $pw < 0.05$), confirming its inefficiency in classification time. These findings establish that the SDP-DR variants are efficient for reducing classification time, making them optimal choices for improving k-NN classification performance.

**Table 4.16:** Average of Mean Classification Time for k-NN (k = 3)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 23.39 | 26.98 | **14.78** | 16.38 | 16.20 | 17.35 |

**Table 4.17:** Sign and Wilcoxon Tests for Mean Classification Time of k-NN (k = 3)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | 11/00/21 | **23/00/09 | **23/00/09 | **25/00/07 | **25/00/07 |
| | | *ps* | 0.07710 | 0.01333 | 0.01333 | 0.00146 | 0.00146 |
| | | *pw* | 0.13104 | 0.00016 | 0.00214 | 0.00022 | 0.00016 |
| | ENN | *ss* | | **27/00/05 | **24/00/08 | **25/00/007 | **24/00/08 |
| | | *ps* | | 0.0001 | 0.00468 | 0.00146 | 0.00468 |
| | | *pw* | | < .00001 | 0.00932 | 0.00100 | 0.00030 |
| | CNN | *ss* | | | 11/00/21 | 13/00/019 | *09/00/23 |
| | | *ps* | | | 0.07710 | 0.28884 | 0.01333 |
| | | *pw* | | | 0.08914 | 0.12602 | 0.00424 |
| | FD | *ss* | | | | 14/00/18 | 16/00/016 |
| | | *ps* | | | | 0.47950 | 1.00000 |
| | | *pw* | | | | 0.77948 | 0.38978 |
| | DCM | *ss* | | | | | 13/00/19 |
| | | *ps* | | | | | 0.28884 |
| | | *pw* | | | | | 0.38978 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

Figure 4.23 to Figure 4.28 present the classification time (ms) for $k = 5$ across k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC), providing insights into the computational efficiency of different prototype selection techniques. The results indicate that CNN continues to have the lowest classification time due to its aggressive instance reduction, but as observed in previous analyses, this comes at the cost of reduced classification accuracy. SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) maintain lower classification times than k-NN while retaining useful data, confirming their ability to balance efficiency and classification performance. Meanwhile, ENN exhibits variability, reducing classification time in some cases but not as effectively as CNN or the SDP-DR variants. k-NN (Figure 4.23) exhibits the highest classification time, reinforcing that direct classification without prototype selection results in increased computational cost.

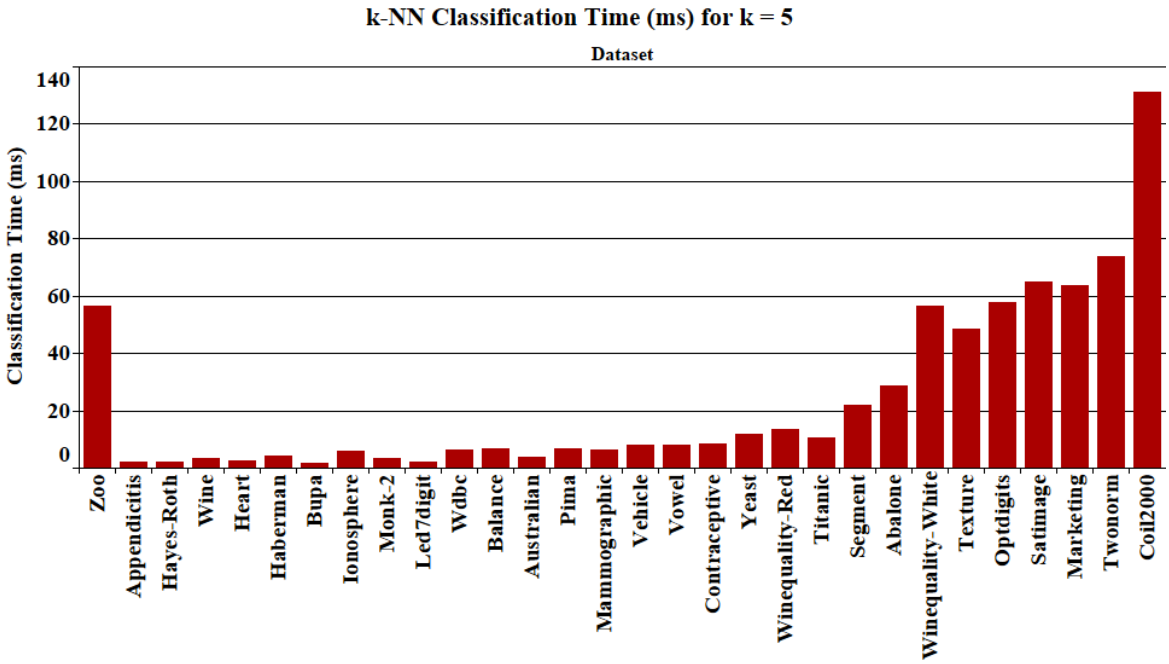


**Figure 4.23:** k-NN Classification Time (ms) for $k = 5$

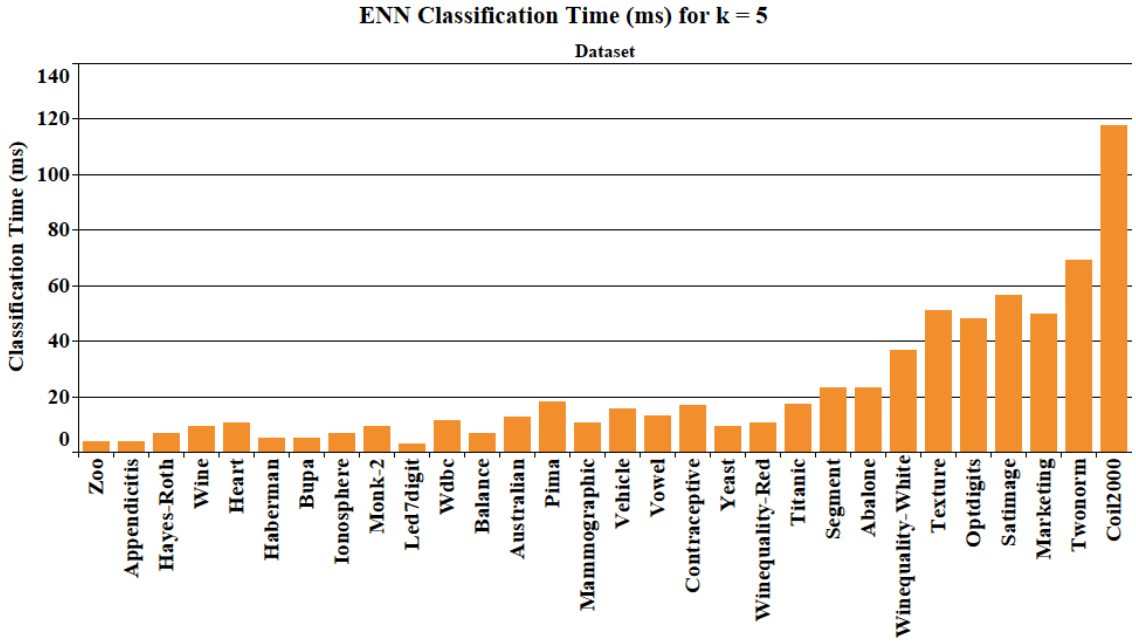


**Figure 4.24:** ENN Classification Time (ms) for $k = 5$

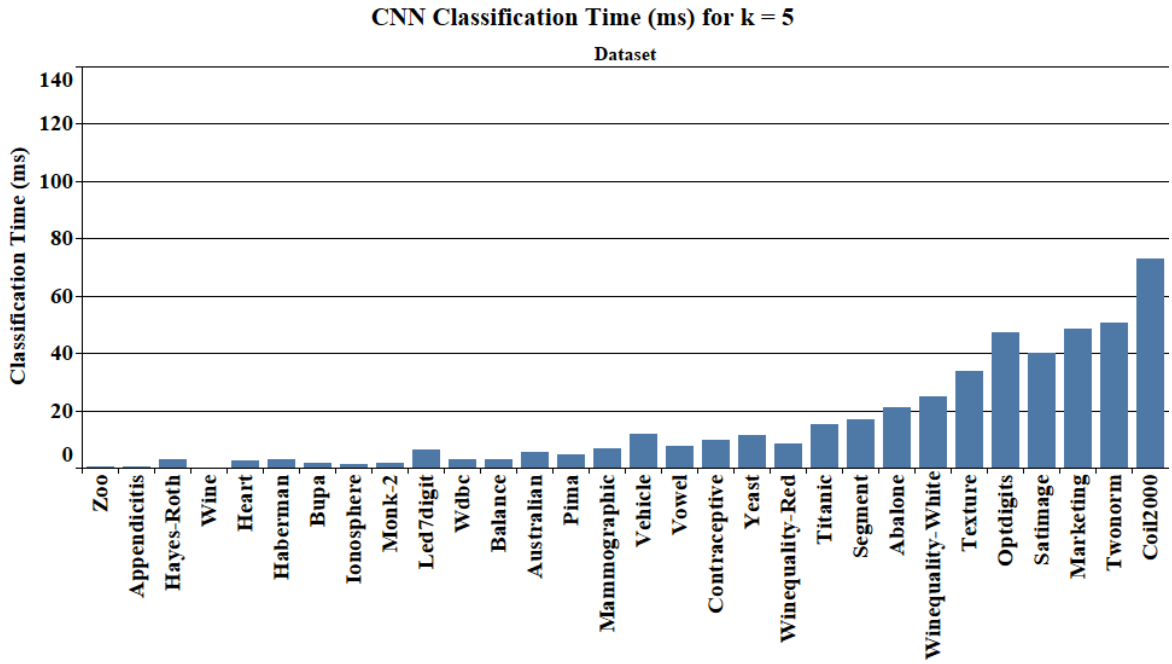


**Figure 4.25:** CNN Classification Time (ms) for $k = 5$

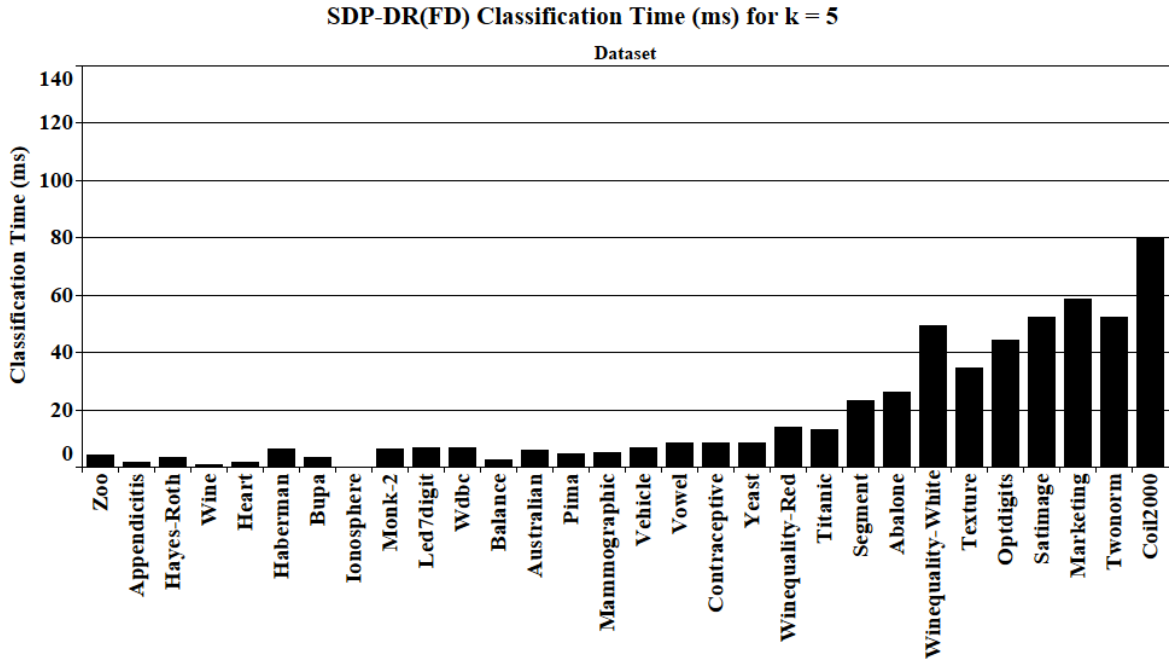


**Figure 4.26:** SDP-DR (FD) Classification Time (ms) for $k = 5$

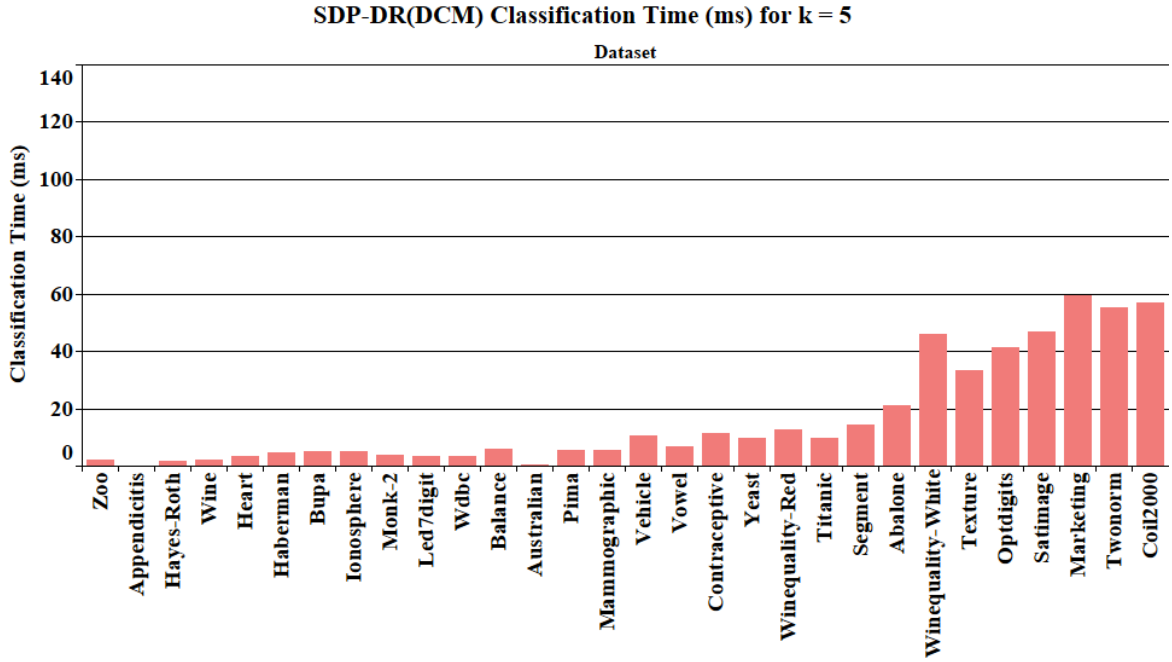


**Figure 4.27:** SDP-DR (DCM) Classification Time (ms) for $k = 5$

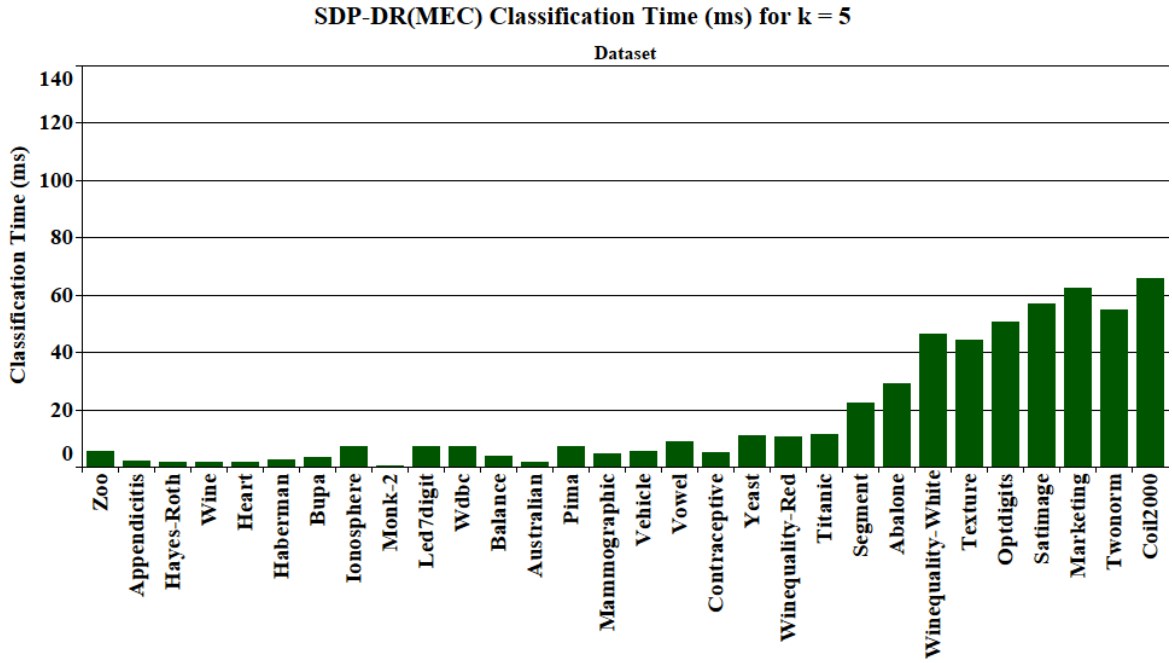


**Figure 4.28:** SDP-DR (MEC) Classification Time (ms) for $k = 5$

Table 4.18 presents the mean classification time across 32 datasets for k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC) for $k = 5$. The results indicate that k-NN continues to have the highest average classification time (22.88 ms), highlighting its computational inefficiency when processing the full dataset. CNN achieves the lowest classification time (14.76 ms), demonstrating its ability to reduce computational cost but at the risk of losing critical classification information. Among the SDP-DR methods, SDP-DR (DCM) and SDP-DR (FD) record classification times of 15.51 ms and 17.19 ms, respectively, suggesting that these methods effectively optimise computational cost while maintaining classification performance. Meanwhile, SDP-DR (MEC) has a classification time of 17.33 ms, slightly higher than the other SDP-DR variants but still significantly lower than k-NN and ENN. Table 4.18 summarizes the full results presented in Appendix B, Table B.9.

Table 4.19 provides the statistical significance analysis for classification time using the Sign Test (*ps*) and Wilcoxon Test (*pw*), where significance is confirmed if both values are below 0.05. The results confirm that k-NN is significantly slower than CNN ($ps = 0.01333$, $pw = 0.00128$), reinforcing CNN's computational efficiency. Additionally, SDP-DR (DCM) significantly reduces classification time compared to k-NN ($ps = 0.00146$, $pw = 0.00094$), confirming its effectiveness in optimising computational performance. Meanwhile, SDP-DR (FD) does not show a statistically significant difference from k-NN ($ps = 0.15730$, $pw = 0.02260$), indicating that it maintains a classification time similar to k-NN while still benefiting from dataset reduction. The statistical tests also reveal that ENN is significantly slower than all SDP-DR variants and CNN ($ps < 0.05$, $pw < 0.05$), reinforcing its inefficiency in classification speed. These findings confirm that SDP-DR (DCM) and SDP-DR (MEC) are the most efficient SDP-DR variants in reducing classification time,

making them optimal choices for improving k-NN classification efficiency while preserving classification accuracy.

**Table 4.18:** Average of Mean Classification Time for k-NN ($k$ = 5)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 22.88 | 21.66 | **14.76** | 17.19 | 15.51 | 17.33 |

**Table 4.19:** Sign and Wilcoxon Tests for Mean Classification Time of k-NN ($k$ = 5)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | 12/00/20 | ****23/00/09** | 20/00/12 | ****25/00/07** | **21/00/11** |
| | | *ps* | 0.15730 | 0.01333 | 0.15730 | 0.00146 | 0.07710 |
| | | *pw* | 0.56192 | 0.00128 | 0.02260 | 0.00094 | 0.00528 |
| | ENN | *ss* | | ****29/00/03** | ****24/00/08** | ****26/00/06** | ****22/00/10** |
| | | *ps* | | < .00001 | 0.00468 | 0.00041 | 0.03389 |
| | | *pw* | | < .00001 | 0.00180 | 0.00044 | 0.01108 |
| | CNN | *ss* | | | 19/00/13 | 19/00/13 | 11/00/21 |
| | | *ps* | | | 0.28884 | 0.28884 | 0.07710 |
| | | *pw* | | | 0.05238 | 0.05238 | 0.03486 |
| | FD | *ss* | | | | 19/00/13 | 13/00/19 |
| | | *ps* | | | | 0.28884 | 0.28884 |
| | | *pw* | | | | 0.05238 | 0.53526 |
| | DCM | *ss* | | | | | 11/00/21 |
| | | *ps* | | | | | 0.07710 |
| | | *pw* | | | | | 0.04338 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

Figure 4.29 to Figure 4.34 present the classification time (ms) for $k$ = 7 across k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC), offering insights into computational efficiency as $k$ increases. The results indicate that CNN maintains the lowest classification time due to its aggressive prototype selection, but as previously noted, this comes at the expense of reduced classification accuracy. SDP-DR (FD) and SDP-DR (DCM) continue to exhibit lower classification times. Meanwhile, ENN displays inconsistent performance, reducing classification time in some cases but failing to match the efficiency of CNN and the SDP-DR methods. k-NN (Figure 4.29) has the highest classification time,

reinforcing its computational inefficiency when processing the full dataset without prototype selection.

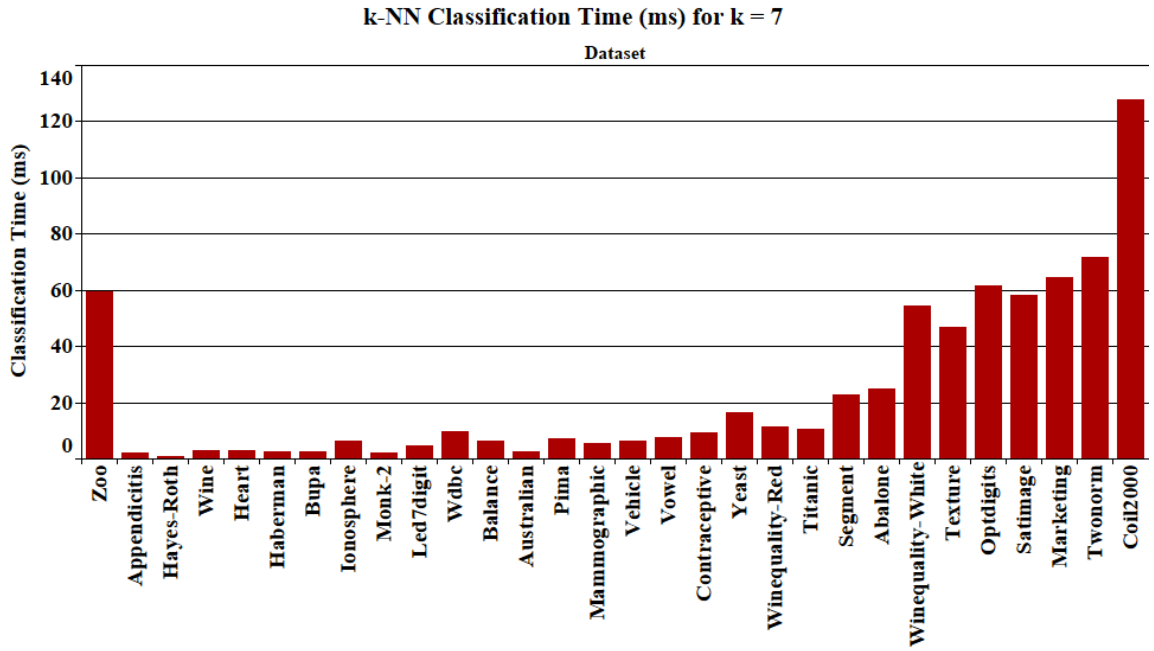


**Figure 4.29:** k-NN Classification Time (ms) for $k = 7$

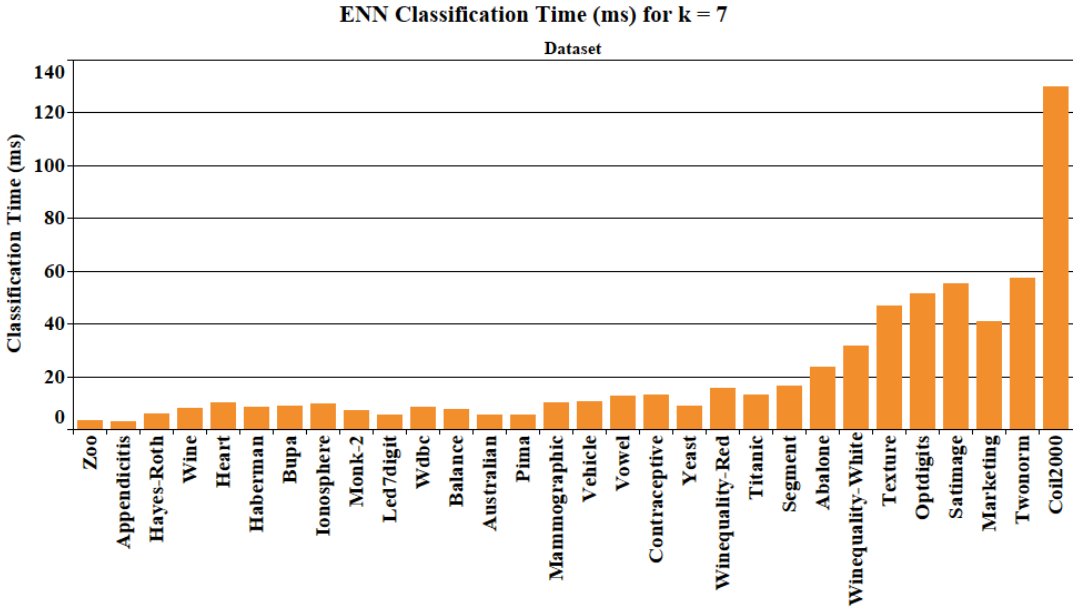


**Figure 4.30:** ENN Classification Time (ms) for $k = 7$

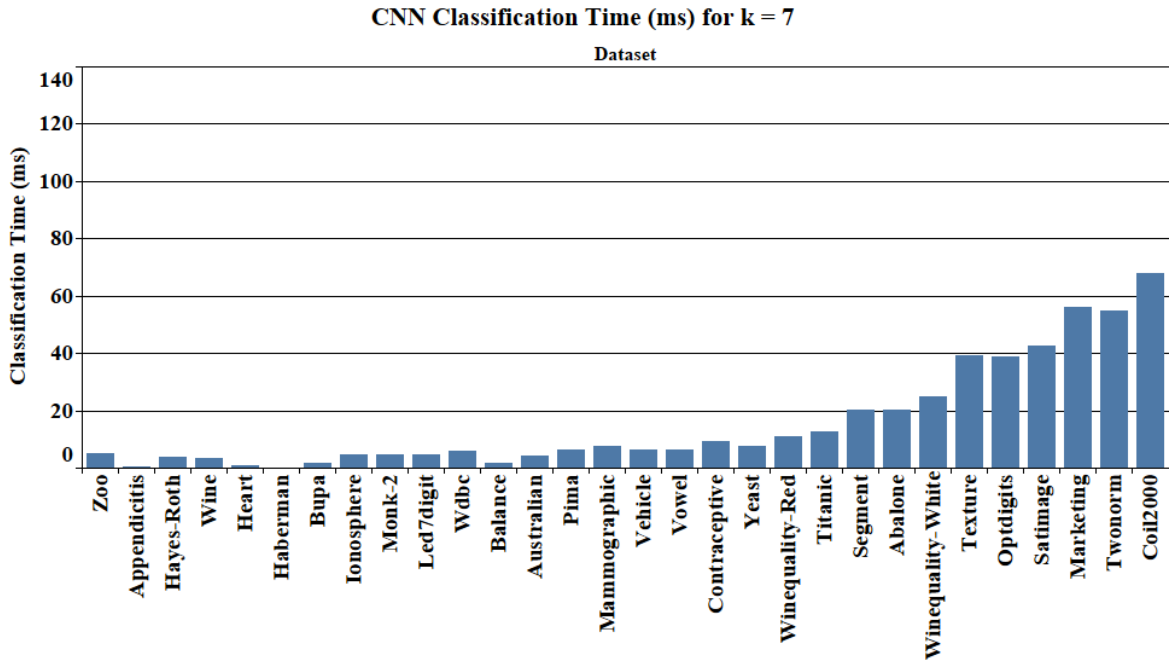


**Figure 4.31:** CNN Classification Time (ms) for $k = 7$

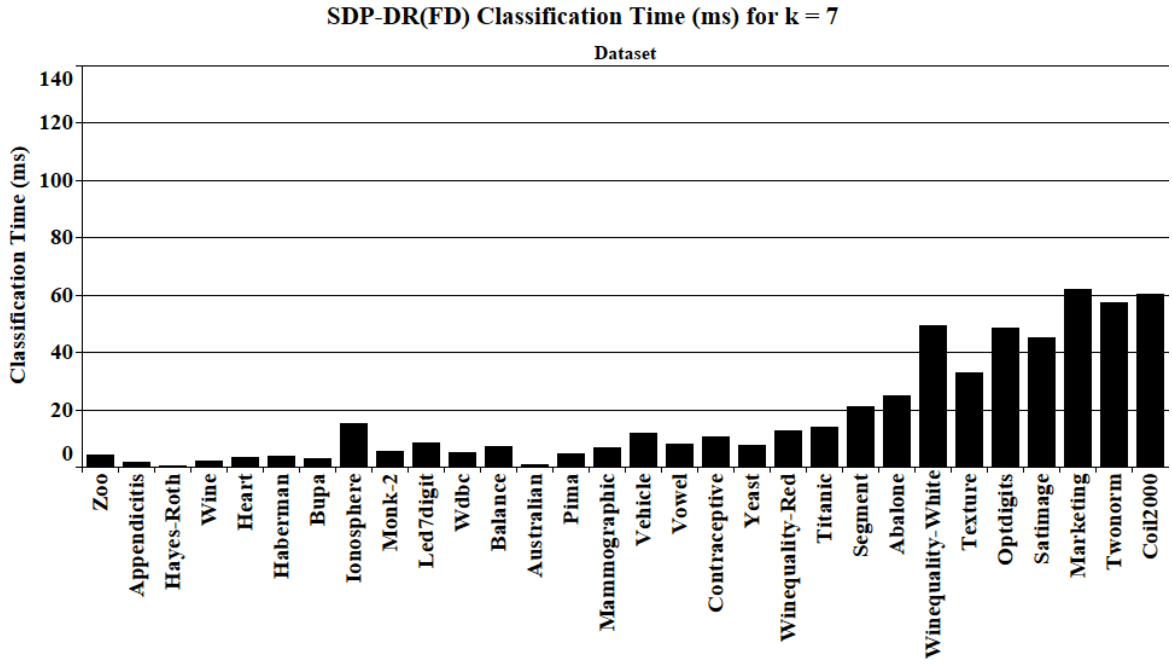


**Figure 4.32:** SDP-DR (FD) Classification Time (ms) for $k = 7$

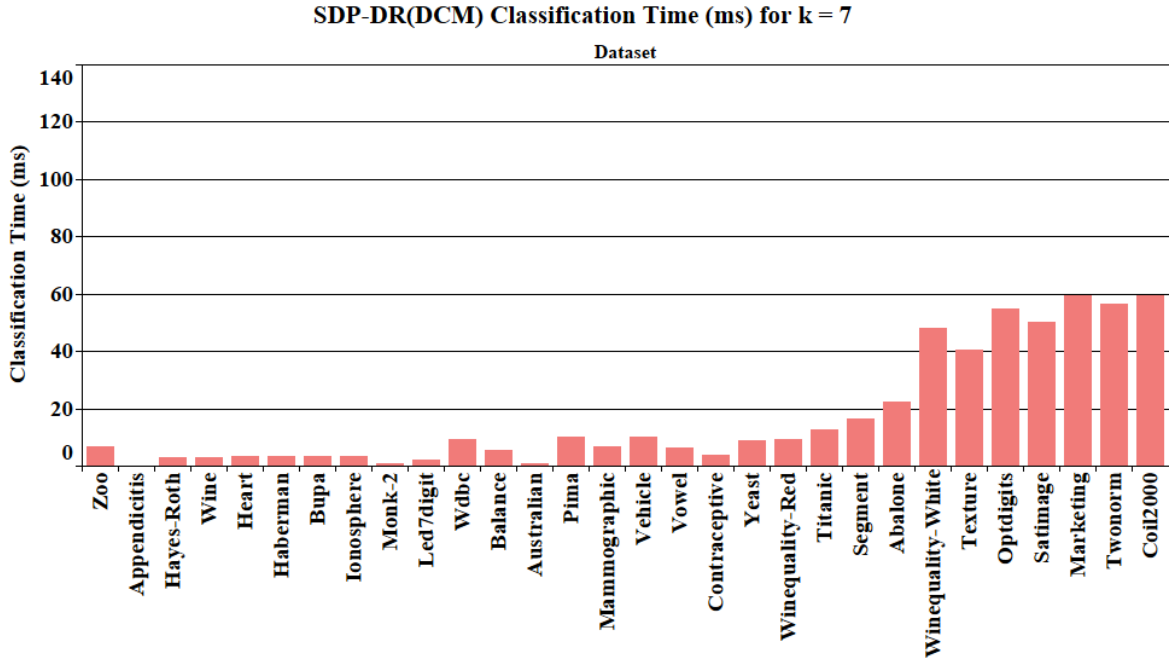


**Figure 4.33:** SDP-DR (DCM) Classification Time (ms) for $k = 7$

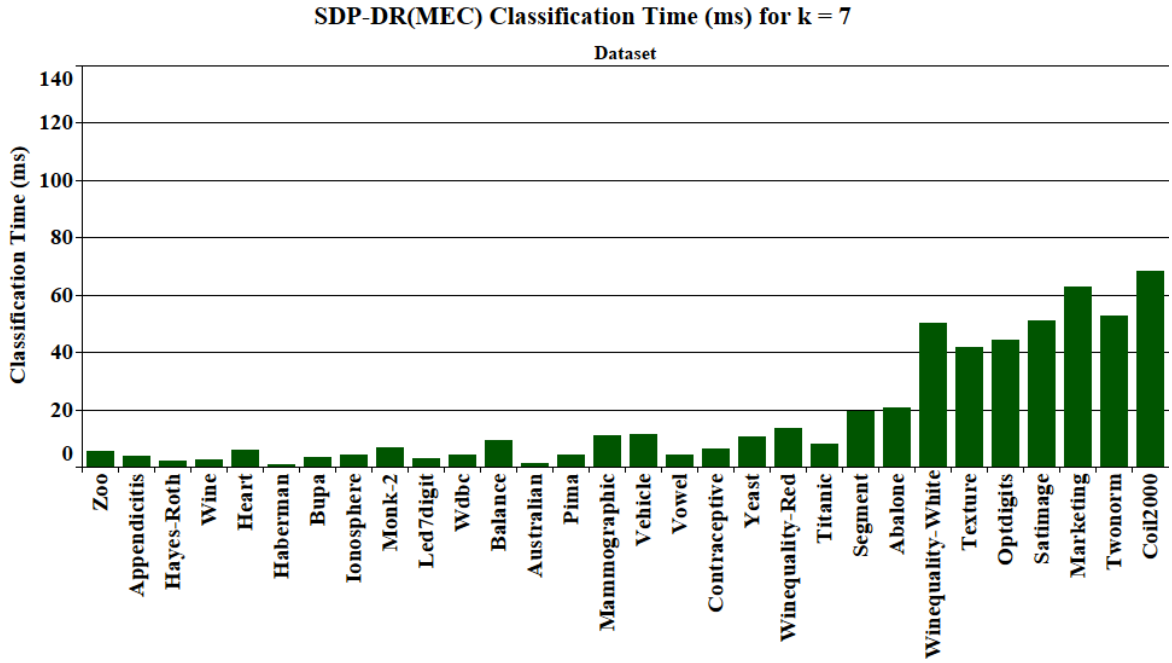


**Figure 4.34:** SDP-DR (MEC) Classification Time (ms) for $k = 7$

Table 4.20 details the mean classification time across 32 datasets for k-NN, ENN, CNN, and the SDP-DR variants (FD, DCM, MEC) for $k = 7$. The results indicate that k-NN continues

to have the highest average classification time (22.48 ms), further highlighting its computational inefficiency when processing large datasets. CNN achieves the lowest classification time (15.01 ms), confirming its ability to reduce computational cost at the expense of accuracy. Among the SDP-DR methods, SDP-DR (DCM), SDP-DR (MEC), and SDP-DR (FD) record classification times of 16.60 ms, 16.97 ms, and 17.19 ms, respectively, suggesting that these methods effectively balance computational cost and classification accuracy, while maintaining significantly lower times than k-NN. Table 4.20 summarizes the full results presented in Appendix B, Table B.10. Table 4.21 provides the statistical significance analysis for classification time using the Sign Test (*ps*) and Wilcoxon Test (*pw*), where significance is confirmed if both values are below 0.05. The results confirm that k-NN is significantly slower than CNN ($ps = 0.03389$, $pw = 0.00200$), reinforcing CNN's computational advantage. Additionally, SDP-DR (MEC) significantly reduces classification time compared to k-NN ($ps = 0.03389$, $pw = 0.02382$), validating its efficiency in optimising computational performance. Meanwhile, SDP-DR (FD) and SDP-DR (DCM) do not show statistically significant differences from k-NN ($ps > 0.05$, $pw > 0.05$), indicating that they maintain classification times similar to k-NN while benefiting from dataset reduction. The statistical tests further reveal that ENN is significantly slower than all SDP-DR variants and CNN ($ps < 0.05$, $pw < 0.05$), reinforcing its inefficiency in classification speed. These findings confirm that the SDP-DR variants are effective in reducing classification time while maintaining classification performance, making them optimal choices for improving k-NN classification efficiency. Table 4.22 shows the summarization of overal result for benchmark algorithms compared to SDP-DR variants.

**Table 4.20:** Average of Mean Classification Time for k-NN ($k$ = 7)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| **AVERAGE** | 22.48 | 20.29 | **15.01** | 17.19 | 16.60 | 16.97 |

**Table 4.21:** Sign and Wilcoxon Tests for Mean Classification Time of k-NN ($k$ = 7)

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | ENN | CNN | FD | DCM | MEC |
| Mean Accuracy | k-NN | *ss* | 12/00/20 | ****22/00/10** | 16/00/16 | 21/00/11 | ****22/00/10** |
| | | *ps* | 0.15730 | 0.03389 | 1.00000 | 0.07710 | 0.03389 |
| | | *pw* | 0.52218 | 0.00200 | 0.23014 | 0.00480 | 0.02382 |
| | ENN | *ss* | | ****28/00/04** | ****22/00/10** | ****24/00/08** | ****22/00/10** |
| | | *ps* | | 0.00002 | 0.03389 | 0.00468 | 0.03389 |
| | | *pw* | | 0.00008 | 0.03662 | 0.00672 | 0.01428 |
| | CNN | *ss* | | | 11/00/21 | 14/00/18 | 13/00/19 |
| | | *ps* | | | 0.07710 | 0.47950 | 0.28884 |
| | | *pw* | | | 0.02144 | 0.19020 | 0.07840 |
| | FD | *ss* | | | | 21/00/11 | 14/00/18 |
| | | *ps* | | | | 0.07710 | 0.47950 |
| | | *pw* | | | | 0.26272 | 0.91240 |
| | DCM | *ss* | | | | | 12/00/20 |
| | | *ps* | | | | | 0.15730 |
| | | *pw* | | | | | 0.41222 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, pw – Wilcoxon test.

**Table 4.22:** Overall Result of Benchmark Algorithms Compared to SDP-DR Variants

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Average Mean Reduction Time | - | 106.76 | 11230.63 | **69.41** | 101.20 | 87.59 |
| Average Mean Reduction Rate | - | 33.90 | **63.66** | 42.17 | 40.53 | 37.47 |
| Average Mean Classification Accuracy for k = 1 | 68.02 | 63.80 | 64.46 | 69.84 | **70.28** | 70.26 |
| Average Mean Classification Accuracy for k = 3 | 69.54 | 64.86 | 61.43 | 69.97 | 70.62 | **71.17** |
| Average Mean Classification Accuracy for k = 5 | 68.40 | 64.95 | 58.82 | 69.55 | 69.72 | **70.74** |
| Average Mean Classification Accuracy for k = 7 | 67.92 | 64.53 | 57.47 | 69.38 | 69.72 | **70.79** |
| Average Mean Classification Time for k = 1 | 20.46 | 17.84 | **13.79** | 15.03 | 14.50 | 17.64 |
| Average Mean Classification Time for k = 3 | 23.39 | 26.98 | **14.78** | 16.38 | 16.20 | 17.35 |
| Average Mean Classification Time for k = 5 | 22.88 | 21.66 | **14.76** | 17.19 | 15.51 | 17.33 |
| Average Mean Classification Time for k = 7 | 22.48 | 20.29 | **15.01** | 17.19 | 16.60 | 16.97 |

### 4.2.7 Comparison between SDP-DR with Current Data Reduction Techniques

Based on all of the previous results, we selected SDP-DR (MEC) with $k$ = 5 to compete with other reduction techniques such as RIS, ATISA1, and DROP3. This decision was made because SDP-DR (MEC) demonstrated overall significant improvement compared to k-NN, CNN, ENN, and the other SDP-DR variants. Additionally, we specifically chose $k$ = 5 because, in prior studies (Cavalcanti & Soares, 2020), the comparison was conducted for $k$ values of 1, 3, and 5, with the best-performing $k$ value being used for the results. However, study done by Cavalcanti and Soares (2020) did not explicitly mention which $k$ value was selected for the final evaluation. Therefore, to ensure clarity and consistency in our comparison, we opted for SDP-DR (MEC) with $k$ = 5 as the representative model for benchmarking against existing reduction techniques. This selection was based on its overall classification performance, as it achieved the highest mean accuracy compared to k-NN, CNN, ENN, and the other SDP-DR variants. Moreover, SDP-DR (MEC) demonstrated one of the best performances in terms of reduction time, reduction rate, and classification time when compared to CNN, ENN, and k-NN, further validating its efficiency. However, in this comparison, we can only evaluate mean accuracy, as the referenced study reports only mean accuracy without providing details on reduction rate, classification time, or reduction time.

Table 4.23 presents the mean classification accuracy across 22 datasets for SDP-DR (MEC) at $k$ = 5, compared to k-NN, ENN, and other existing data reduction techniques, including RIS, ATISA, and DROP3. The results highlight SDP-DR (MEC) achieving an average accuracy of 66.96%, which is higher than ENN (57.25%), DROP3 (62.58%), and ATISA (63.73%), but slightly lower than RIS (67.45%). SDP-DR (MEC) outperforms k-NN

(62.43%), suggesting that the proposed method not only reduces dataset size but also enhances classification accuracy. The effectiveness of SDP-DR (MEC) is evident in datasets such as Appendicitis (88.64%), Segment (95.54%), and Satimage (88.41%), where it achieves the highest accuracy compared to all other methods. In contrast, RIS achieves the highest overall accuracy (67.45%), particularly in datasets like Coil2000 (94.03%) and Titanic (69.06%). However, RIS underperforms in some datasets, such as Marketing (18.28%) and Yeast (41.77%), where SDP-DR (MEC) outperforms it. ATISA and DROP3 maintain competitive accuracy but fall behind SDP-DR (MEC) in key datasets such as Yeast and Winequality-White, confirming SDP-DR (MEC)'s superior classification performance and data reduction.

**Table 4.23:** Mean Accuracy Comparison of SDP-DR Variants ($k = 5$) and Existing Data Reduction Techniques

| Dataset | k-NN | RIS | ENN | DROP3 | ATISA1 | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Appendicitis | 74.08 | 80.79 | 77.27 | 78.33 | 78.33 | **88.64** |
| Balance | 77.81 | **86.72** | 66.10 | 85.24 | 84.92 | 75.38 |
| Bupa | **58.40** | 57.97 | 53.35 | 56.00 | 56.86 | 58.22 |
| Coil2000 | 90.05 | **94.03** | 84.37 | 93.96 | 93.94 | 75.77 |
| Contraceptive | 43.04 | 46.91 | 34.49 | 48.66 | **50.34** | 41.48 |
| Haberman | 62.81 | 64.98 | 58.85 | **70.00** | 69.69 | 66.98 |
| Hayes-roth | 58.56 | **66.42** | 42.22 | 44.44 | 36.11 | 41.88 |
| Heart | 74.81 | 74.44 | 72.59 | 76.30 | **81.11** | 80.00 |
| Ionosphere | 84.42 | **90.62** | 82.22 | 82.22 | 83.89 | 87.74 |
| Led7digit | 61.27 | **72.37** | 39.82 | 39.45 | 41.27 | 70.20 |
| Marketing | 21.80 | 18.28 | 19.88 | 21.32 | 21.14 | **28.26** |
| Monk-2 | 69.93 | 92.11 | 70.35 | 79.32 | **95.45** | 79.15 |
| Movement-libras | 65.60 | **78.11** | 60.83 | 65.27 | 68.05 | 57.78 |
| Pima | 69.22 | 63.40 | 67.18 | **71.95** | 71.69 | 69.79 |
| Satimage | 21.49 | 25.05 | 21.07 | 21.07 | 21.07 | **88.41** |
| Segment | 94.52 | 92.12 | 90.74 | 91.73 | 92.47 | **95.54** |
| Titanic | 59.14 | **69.06** | 32.17 | 32.17 | 32.17 | 59.61 |
| Vowel | 65.35 | 87.88 | 53.74 | 87.07 | **88.89** | 57.07 |
| Wine | 89.79 | 93.39 | 89.47 | 89.47 | 90.53 | **93.79** |
| Winequality-red | 48.03 | 45.83 | 52.16 | 52.16 | **53.40** | 52.35 |

**Table 4.23** continued

| | | | | | | |
|---|---|---|---|---|---|---|
| Winequality-white | 41.19 | 41.55 | 42.02 | 43.34 | 42.08 | **49.10** |
| Yeast | 42.10 | 41.77 | 48.69 | 47.32 | 48.63 | **56.06** |
| **AVERAGE** | 62.43 | **67.45** | 57.25 | 62.58 | 63.73 | 66.96 |

Table 4.24 presents the statistical significance analysis for classification accuracy using the Sign Test (*ps*) and Wilcoxon Test (*pw*), where significance is confirmed if both values are below 0.05. The proposed method, SDP-DR (MEC), is compared against k-NN, ENN, RIS, DROP3, and ATISA1 to assess its effectiveness in classification performance. When comparing SDP-DR (MEC) to k-NN, the win/draw/loss record is 07/00/15, indicating that SDP-DR (MEC) outperforms k-NN in 15 datasets, while k-NN performs better in only seven datasets. The statistical tests show that this difference is not statistically significant (*ps* = 0.08808, *pw* = 0.13104), suggesting that while SDP-DR (MEC) achieves superior accuracy compared to k-NN, the improvement is not statistically conclusive. Additionally, the method benefits from dataset reduction, further validating its potential practical utility.

When compared to RIS, SDP-DR (MEC) records a win/draw/loss of 10/00/12, meaning RIS performs better in ten datasets, while SDP-DR (MEC) outperforms RIS in 12 datasets. However, the statistical tests (*ps* = 0.66982, *pw* = 0.71138) indicate no significant difference between RIS and SDP-DR (MEC), reinforcing that both methods achieve comparable classification accuracy, making SDP-DR (MEC) a practical alternative to RIS. Furthermore, SDP-DR (MEC) significantly outperforms ENN (win/draw/loss: 03/00/19, *ps* = 0.00065, *pw* = 0.00070), confirming its effectiveness in maintaining classification accuracy while optimising data reduction. These results strongly validate SDP-DR (MEC) as the best-performing method, surpassing k-NN and ENN, while achieving comparable accuracy to RIS, DROP3, and ATISA1, with a more efficient prototype selection approach.

**Table 4.24:** Sign and Wilcoxon Tests for Mean Accuracy of SDP-DR Variant ($k$ = 5) and Existing Data Reduction Techniques

| Evaluation Measure | Models (Row) | Analysis | Model (Column) | | | | |
|---|---|---|---|---|---|---|---|
| | | | RIS | ENN | DROP3 | ATISA1 | MEC |
| Mean Accuracy | k-NN | *ss* | 07/00/15 | ****17/00/05** | 10/00/12 | 08/00/14 | 07/00/15 |
| | | *ps* | 0.08808 | 0.01052 | 0.66982 | 0.20083 | 0.08808 |
| | | *pw* | 0.00614 | 0.00714 | 0.33706 | 0.11642 | 0.13104 |
| | RIS | *ss* | | ****17/00/05** | 14/00/08 | 11/00/11 | 10/00/12 |
| | | *ps* | | 0.01052 | 0.20083 | 1.00000 | 0.66982 |
| | | *pw* | | 0.00236 | 0.20054 | 0.74896 | 0.71138 |
| | ENN | *ss* | | | ***02/05/15** | ***02/02/18** | ***03/00/19** |
| | | *ps* | | | 0.00162 | 0.00035 | 0.00065 |
| | | *pw* | | | 0.00084 | 0.00062 | 0.00070 |
| | DROP3 | *ss* | | | | 07/03/12 | 09/00/13 |
| | | *ps* | | | | 0.25135 | 0.39377 |
| | | *pw* | | | | 0.03662 | 0.32218 |
| | ATISA1 | *ss* | | | | | 10/00/12 |
| | | *ps* | | | | | 0.66982 |
| | | *pw* | | | | | 0.58920 |

**Note:** *- the column model has significant difference with the row model for both statistical test. **-the row model has significant difference with the column model for both statistical tests. ss – win/draw/lose record, *ps* – Sign test, *pw* – Wilcoxon test.

## 4.3 Performance Measurement Evaluation for Big Datasets

The second evaluation analysis presents the results for big datasets, specifically focusing on the KDDCUP as a case study. The KDDCUP dataset, although originally introduced in 1999, remains one of the most widely used benchmark datasets in classification research. It was selected in this study for two main reasons. First, it is among the smaller datasets within the category of big data benchmarks, making it suitable as a starting point for evaluating computationally intensive algorithms such as CNN, ENN, and k-NN, which otherwise require substantial processing resources. Second, its continued use in the literature provides a basis for comparison with prior studies, thereby ensuring that the experimental results remain relevant and comparable. It discusses key performance metrics such as reduction time, reduction rate, and classification time, emphasising how big data influences these metrics. Subsequently, a comparison of mean accuracy for six large datasets between

our proposed method and other k-NN methods for big data will be presented. The evaluated algorithms include:

i. ENN: Edited Nearest Neighbour
ii. CNN: Condensed Nearest Neighbour
iii. KNN-IS: K-Nearest Neighbour Instance Selection
iv. GHAHS-FKNN: Global Approximate Hybrid Spill Tree
v. LHS-FKNN: Local Hybrid Spill Tree FKNN
vi. CQ-KNN: Composite Quantization of k-NN
vii. CQ-EKNN: Composite Quantization EK-NN
viii. DEKNN: Distributed EK-NN Classification
ix. GEK-NN: Global Exact EK-NN
x. LEK-NN: Local Approximate EK-NN
xi. SDP-DR FD: Similarity Distance Plot-Data Reduction (First Data as Anchor)
xii. SDP-DR DCM: Similarity Distance Plot-Data Reduction (Data Closed to Mean as Anchor)
xiii. SDP-DR MEC: Similarity Distance Plot-Data Reduction (Mean of Each Column as Anchor)

### 4.3.1 Datasets

Table 4.25 presents the big datasets used in the experiment, including KDDCUP, Epsilon, Poker, SUSY, and Higgs. These datasets vary in size, complexity, and number of attributes, providing a robust benchmark for evaluating the efficiency of prototype selection techniques in large-scale classification problems. The KDDCUP dataset consists of 494,020 instances with 41 attributes and 23 classes. The Epsilon dataset contains 500,000 instances with 2,000 attributes and 2 classes. The Poker dataset comprises 1,025,009 instances with

10 attributes and 10 classes. The SUSY dataset includes 5,000,000 instances with 18 attributes and 2 classes. Finally, the Higgs dataset, the largest in this study, consists of 11,000,000 instances, 28 attributes, and 3 classes. These datasets were chosen to evaluate the scalability and efficiency of the SDP-DR variants in big dataset classification tasks, offering diverse challenges in terms of dataset size and computational complexity by competing with other data reduction techniques for big dataset from previous study done by Gong et al. (2023b).

**Table 4.25:** Dataset Used for Experiment of Big Data

| Dataset | Attribute | Instances | Classes |
|---|---|---|---|
| KDDCUP | 41 | 494020 | 23 |
| Epsilon | 2000 | 500000 | 2 |
| Poker | 10 | 1025009 | 10 |
| SUSY | 18 | 5000000 | 2 |
| Higgs | 28 | 11000000 | 3 |

### 4.3.2 Case Study: KDDCUP

This case study aims to demonstrate the advantages of the proposed SDP-DR method over traditional techniques such as k-NN, CNN, and ENN, particularly in terms of reduction time, reduction rate, and classification time. The KDDCUP dataset was chosen for this analysis as it is the smallest among the large datasets used, making it more practical for executing detailed comparisons. By using KDDCUP, this study effectively illustrates the substantial computational demands of conventional prototype selection methods and emphasises the efficiency gains achieved through the proposed SDP-DR approach.

Figure 4.35 illustrates the reduction time (ms) comparison for CNN, ENN, and SDP-DR variants on the KDDCUP dataset. The results indicate that ENN has the highest reduction time, followed by CNN, while all SDP-DR variants (FD, DCM, MEC) exhibit

significantly lower reduction times. Specifically, ENN requires the longest processing time due to its iterative nearest neighbour comparisons, making it inefficient for large datasets. CNN, while faster than ENN, still demands substantial processing resources. In contrast, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) demonstrate significantly lower reduction times, reinforcing their computational efficiency in handling big data. This highlights that SDP-DR methods offer a more practical and scalable approach for large-scale data reduction, outperforming traditional prototype selection techniques in terms of speed.

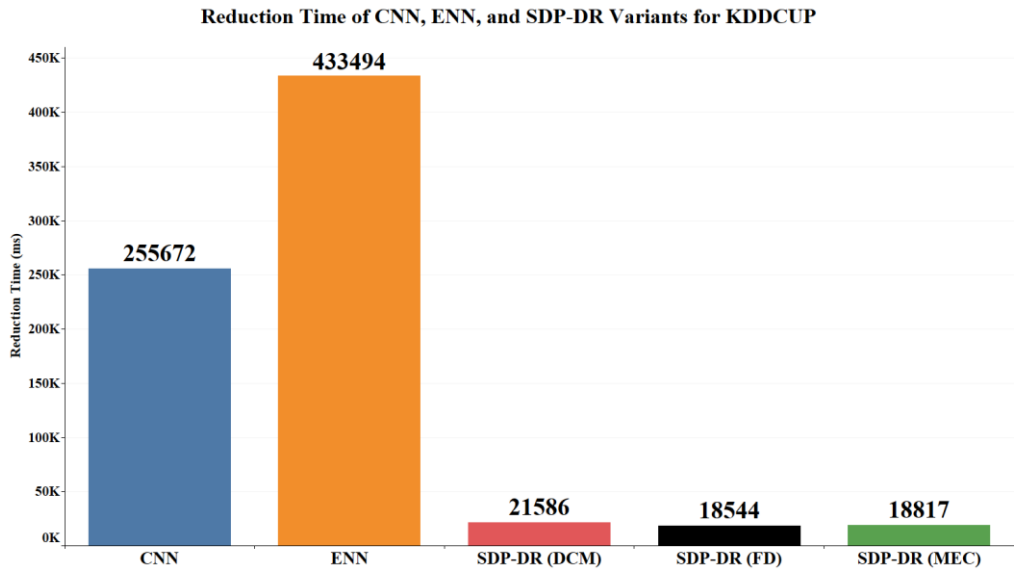


**Figure 4.35:** Reduction Time (ms) Comparison of CNN, ENN, and SDP-DR Variants on the KDDCUP

Figure 4.36 presents the reduction rate (%) comparison for CNN, ENN, and SDP-DR variants on the KDDCUP dataset. The results show that CNN achieves the highest reduction rate, indicating that it removes the largest proportion of instances. However, this aggressive reduction may lead to a loss of valuable classification information. ENN exhibits the lowest reduction rate but incurs a much higher computational cost, making it less favourable for big data applications. Among the SDP-DR variants, SDP-DR (MEC) achieves

a well-balanced reduction rate and reduction time, followed by SDP-DR (FD) and SDP-DR (DCM). These findings suggest that SDP-DR methods effectively reduce dataset size while preserving critical classification information, making them superior to CNN and ENN in large-scale applications.

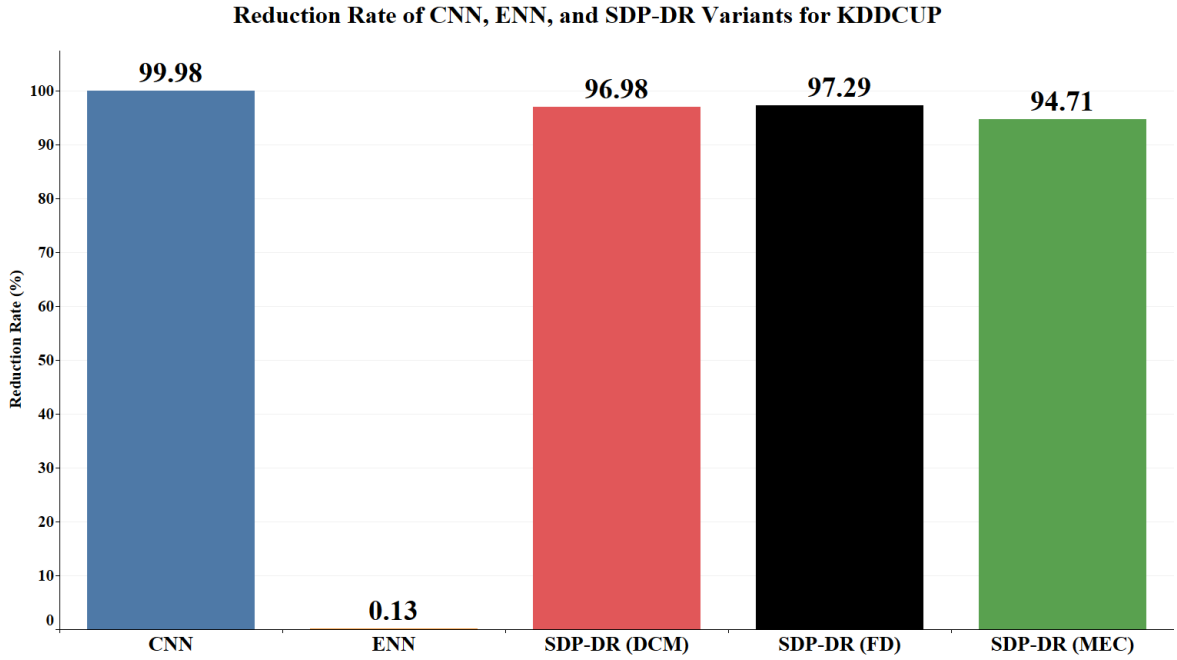


**Figure 4.36:** Reduction Rate (%) Comparison of CNN, ENN, and SDP-DR Variants on the KDDCUP

Figure 4.37 compares the classification time (ms) for k-NN, CNN, ENN, and SDP-DR variants on the KDDCUP dataset, where $k = 1$. The results indicate that k-NN exhibits the highest classification time, reinforcing the need for prototype selection techniques when handling large datasets. CNN, despite achieving a high reduction rate, still requires considerable classification time. ENN also exhibits a high classification time due to insufficient data reduction. In contrast, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) significantly reduce classification time compared to k-NN and ENN. Among the SDP-DR variants, SDP-DR (FD) achieves the lowest classification time, confirming its

computational efficiency. These results validate that SDP-DR methods provide an optimal balance between dataset reduction and classification efficiency, making them highly suitable for large-scale datasets like KDDCUP.

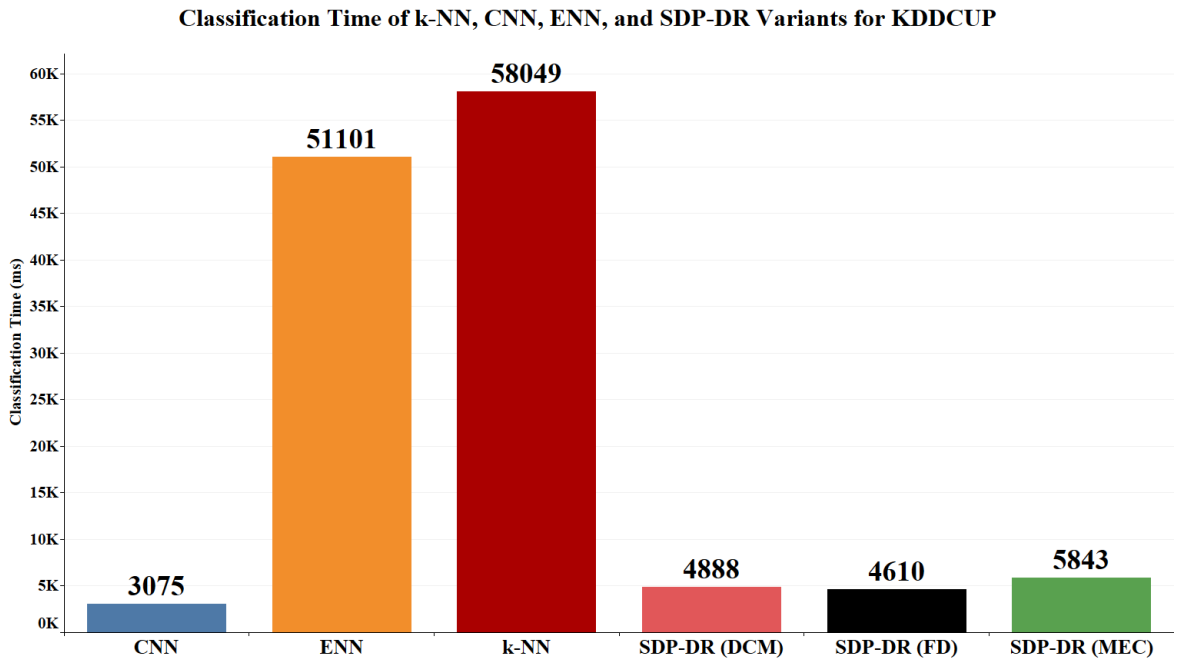


**Figure 4.37:** Classification Time (ms) Comparison of k-NN, CNN, ENN, and SDP-DR Variants on the KDDCUP

The findings from this case study reveal that SDP-DR variants (FD, DCM, MEC) significantly outperform CNN and ENN in reduction time, making them computationally efficient for big data applications. While CNN achieves the highest reduction rate, it risks losing critical classification accuracy, whereas SDP-DR methods provide a more balanced approach. Additionally, SDP-DR (FD) achieves the lowest classification time, demonstrating its superiority in efficiently handling large datasets. These results reinforce that k-NN suffers from excessive classification time without prototype selection, emphasising the necessity of data reduction techniques like SDP-DR. Overall, the analysis of KDDCUP confirms SDP-DR as a powerful prototype selection approach for large-scale

classification tasks, proving its advantage over traditional methods like CNN and ENN in terms of computational efficiency.

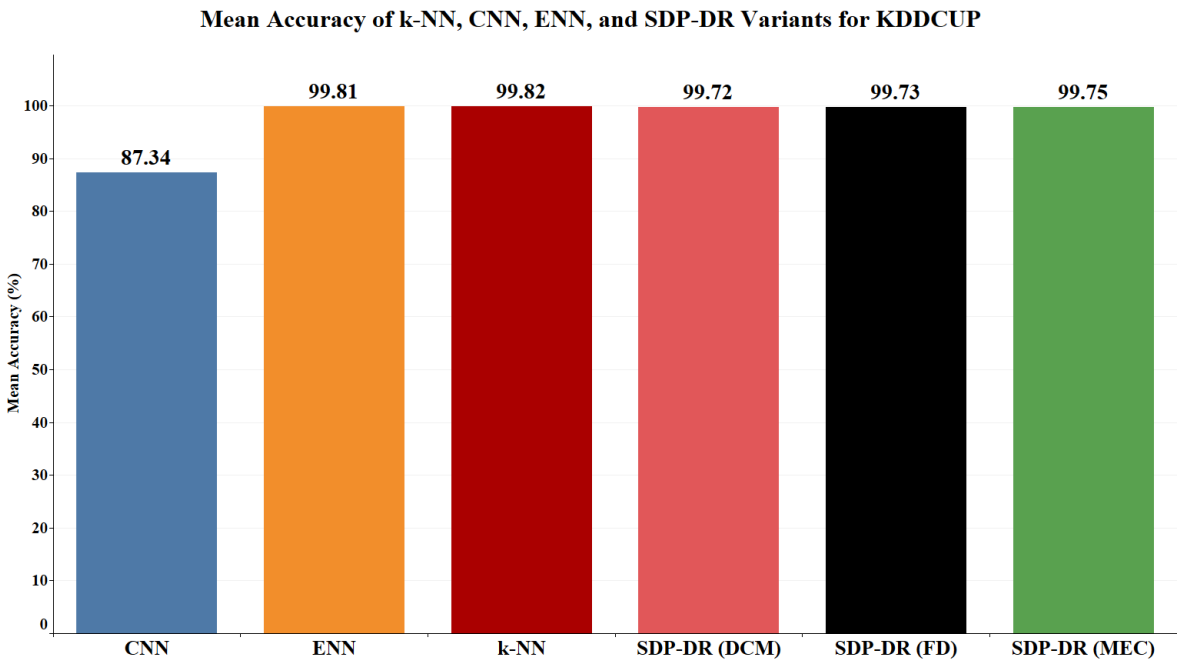


**Figure 4.38:** Mean Accuracy Comparison of k-NN, CNN, ENN, and SDP-DR Variants on the KDDCUP

Figure 4.38 illustrates the mean classification accuracy of k-NN, CNN, ENN, and SDP-DR variants on the KDDCUP dataset. The results reveal that all SDP-DR variants (FD, DCM, and MEC) substantially outperform CNN, which records the lowest accuracy at 87.34%. Although k-NN achieves 99.82% accuracy and ENN achieves 99.81%, their slight advantage in accuracy comes at the cost of significantly higher classification time and reduced computational efficiency. In comparison, SDP-DR (MEC), SDP-DR (FD), and SDP-DR (DCM) achieve competitive accuracies of 99.75%, 99.73%, and 99.72%, respectively, while delivering notable improvements in reduction rate, reduction time and classification time. These findings confirm that SDP-DR methods are capable of maintaining high classification accuracy while significantly reducing computational overhead,

reinforcing their value as effective and scalable prototype selection approaches for large-scale classification tasks such as KDDCUP.

### 4.3.3 Comparison between SDP-DR with State-of-the-Art Data Reduction Techniques for Big Data

To ensure a comprehensive evaluation of our proposed method, we compared our results with an existing experimental study that closely aligns with our setup. It was essential to select a study with a similar experimental configuration, including stratified 10-fold cross-validation and the use of the Euclidean distance as the similarity measure for k-NN classification.

After an extensive review of related works, we identified a study by Gong et al. (2023b) that closely matches our experimental conditions. Their study evaluates multiple k-NN-based prototype selection and classification methods using the same datasets as ours (KDDCUP, Epsilon, Poker, SUSY, and Higgs), ensuring a fair and relevant comparison. The methods tested in their study include Global Exact EK-NN (GEK-NN), Local Approximate EK-NN (LEK-NN), K-Nearest Neighbour Instance Selection (kNN-IS) (Maillo et al., 2017), Global Approximate Hybrid Spill Tree (GHAHS-FKNN) (Maillo et al., 2019), Local Hybrid Spill Tree FKNN (LHS-FKNN) (Maillo et al., 2019), Composite Quantization of k-NN (CQ-KNN) (Wang & Zhang, 2018), Composite Quantization EK-NN (CQ-EKNN) (Wang & Zhang, 2018), and Distributed EK-NN Classification (DEKNN) (Gong et al., 2022).

For performance evaluation, we specifically compare classification accuracy, as it is the primary metric reported in both our study and Gong et al. (2023b). This ensures a fair and meaningful comparison of how well each method performs on large-scale datasets. By

benchmarking our SDP-DR variants (FD, DCM, MEC) against these established algorithms, we validate the effectiveness of our approach in enhancing classification accuracy.

To further analyse the comparative performance of prototype selection techniques, Table 4.26 presents the classification accuracy of various k-NN-based algorithms across five large-scale datasets: KDDCUP, Epsilon, Poker, SUSY, and Higgs. The comparison is conducted using three different $k$ values ($k$ = 3, 5, 7) to evaluate how our proposed algorithm performs across diverse datasets. The algorithms tested include KNN-IS, DEKNN, GEK-NN, LEK-NN, RIS, ATISA, DROP3, and the proposed SDP-DR variants (FD, DCM, MEC).

For the KDDCUP dataset, SDP-DR (MEC) records the highest accuracy across all $k$ values, demonstrating its robustness in handling large-scale classification problems. At $k$ = 3, SDP-DR (MEC) achieves 0.9974, surpassing all other methods, including SDP-DR (FD) (0.9970) and SDP-DR (DCM) (0.9969). The highest accuracy among benchmark algorithm methods is achieved by LEK-NN (0.6994), which is significantly lower than that of SDP-DR (MEC). When $k$ increases to 5 and 7, SDP-DR (MEC) maintains its dominance with accuracies of 0.9973 and 0.9970, respectively, while LEK-NN remains the highest-performing k-NN variant at 0.6994 and 0.6993. These results confirm that SDP-DR (MEC) is highly effective for large datasets like KDDCUP, significantly outperforming other k-NN-based algorithms.

For the Epsilon dataset, the results indicate that all SDP-DR variants (FD, DCM, MEC) consistently achieve high classification accuracy across $k$ = 5 and 7, demonstrating their effectiveness in dataset reduction without compromising classification performance. At $k$ = 3, SDP-DR (FD) achieves the highest accuracy among the SDP-DR variants at 0.5909, closely followed by SDP-DR (MEC) (0.5904) and SDP-DR (DCM) (0.5896). Although

these SDP-DR variants exhibit slightly lower accuracy compared to traditional k-NN methods such as LEK-NN (0.5999) and KNN-IS (0.5864), their performance remains competitive, indicating their capability to handle large-scale datasets like Epsilon effectively.

When *k* is increased to 5, all SDP-DR variants demonstrate significant improvements in accuracy, with SDP-DR (FD) achieving 0.6060, followed closely by SDP-DR (MEC) (0.6043) and SDP-DR (DCM) (0.6030). At this *k*-value, the SDP-DR variants outperform traditional k-NN-based methods, including LEK-NN (0.6007) and KNN-IS (0.6006), confirming the superior effectiveness of SDP-DR at higher *k*-values. When *k* is further increased to 7, all SDP-DR variants show additional improvements, with SDP-DR (FD) attaining the highest accuracy at 0.6152, closely followed by SDP-DR (DCM) (0.6127) and SDP-DR (MEC) (0.6120). At this stage, all SDP-DR variants surpass the highest-performing non-SDP-DR method, LEK-NN (0.6118). This trend demonstrates that the effectiveness of SDP-DR methods increases with higher *k*-values, consistently outperforming traditional k-NN-based algorithms. Overall, these results clearly indicate that all SDP-DR variants effectively handle the Epsilon dataset by consistently achieving strong and stable classification performance, particularly at higher *k*-values, underscoring their robustness and suitability for large-scale datasets.

In the Poker dataset, the accuracy trends differ slightly. At $k = 3$, SDP-DR (MEC) achieves 0.5149, which is lower than several k-NN variants, such as GEK-NN (0.5571) and DEKNN (0.5571), but slightly higher than SDP-DR (FD) (0.5067) and SDP-DR (DCM) (0.5063). When *k* is increased to 5, the accuracy of SDP-DR (MEC) improves to 0.5283, surpassing SDP-DR (FD) (0.5223) and SDP-DR (DCM) (0.5220), but still lower than GEK-

NN (0.5998) and DEKNN (0.5948). At $k = 7$, SDP-DR (MEC) further improves to 0.5360, but remains below GEK-NN (0.6107) and DEKNN (0.6102). These results suggest that while SDP-DR methods benefit from increasing $k$, certain datasets, such as Poker, may require alternative reduction strategies for optimal performance.

For the SUSY dataset, the results indicate that all SDP-DR variants (FD, DCM, MEC) consistently achieve high classification accuracy across $k = 3$, 5, and 7, demonstrating their effectiveness in reducing dataset size while maintaining strong classification performance. At $k = 3$, SDP-DR (DCM) achieves the highest accuracy among all methods at 0.7253, tied with LHS-FKNN, followed closely by SDP-DR (FD) (0.7251) and SDP-DR (MEC) (0.7244). These results indicate that the SDP-DR variants perform competitively with the best k-NN-based algorithms and effectively handle large datasets like SUSY. When $k$ is increased to 5, all SDP-DR variants further improve in classification accuracy, reinforcing the trend that larger $k$ values enhance the effectiveness of prototype selection. SDP-DR (FD) achieves 0.7410, SDP-DR (DCM) records 0.7411, and SDP-DR (MEC) reaches 0.7402. These values confirm that all SDP-DR variants outperform other k-NN methods such as LEK-NN (0.7362) and DEKNN (0.7360), demonstrating that SDP-DR provides a superior approach to prototype selection for large datasets like SUSY. At $k = 7$, all SDP-DR variants maintain their classification strength, further improving their accuracy. SDP-DR (DCM) achieves the highest accuracy among the three SDP-DR methods at 0.7501, followed by SDP-DR (FD) at 0.7499 and SDP-DR (MEC) at 0.7491. These results indicate that increasing $k$ further enhances the performance of SDP-DR methods, making them the most effective prototype selection approaches for SUSY. Notably, all three SDP-DR variants outperform traditional k-NN methods, as the highest-performing non-SDP-DR method (LEK-NN) records 0.7440, which is lower than all SDP-DR variants. These findings confirm

that all SDP-DR variants (FD, DCM, MEC) perform exceptionally well on the SUSY dataset across all $k$ values, consistently achieving higher classification accuracy than traditional k-NN-based methods. SDP-DR (DCM) achieves the highest accuracy at $k$ = 7 (0.7501), making it the most effective among the three SDP-DR variants. However, all SDP-DR methods maintain stable and strong classification performance, proving their robustness in large-scale datasets like SUSY.

For the Higgs dataset, the results indicate that all SDP-DR variants (FD, DCM, MEC) consistently achieve competitive classification accuracy across $k$ = 3, 5, and 7, demonstrating their effectiveness in handling large-scale classification problems. At $k$ = 3, SDP-DR (DCM) achieves the highest accuracy among the three SDP-DR variants at 0.6079, followed closely by SDP-DR (FD) (0.6076) and SDP-DR (MEC) (0.6076). Among traditional k-NN-based methods, LHS-FKNN records the highest accuracy (0.6426), surpassing all SDP-DR variants and other algorithms. These results suggest that while SDP-DR methods are highly efficient, certain k-NN variations like LHS-FKNN can sometimes achieve better performance. When $k$ is increased to 5, the classification accuracy of all SDP-DR variants improves, demonstrating the benefits of higher $k$ values in prototype selection. SDP-DR (MEC) achieves 0.6159, while SDP-DR (FD) and SDP-DR (DCM) both record 0.6158 and 0.6157, respectively. Despite these improvements, other k-NN methods such as LHS-FKNN (0.6615) and LEK-NN (0.6532) still outperform all SDP-DR variants, confirming that some dataset structures may favour other k-NN approaches over our proposed prototype selection techniques. However, SDP-DR methods continue to maintain their advantage in achieving competitive accuracy. At $k$ = 7, all SDP-DR variants maintain stable classification performance. SDP-DR (DCM) achieves the highest accuracy among the SDP-DR methods at 0.6227, followed by SDP-DR (FD) (0.6221) and SDP-DR (MEC) (0.6220). However,

LEK-NN (0.6673) and GHAHS-FKNN (0.6593) outperform all SDP-DR variants, suggesting that other k-NN approaches remain strong contenders in datasets like Higgs. While SDP-DR methods provide dataset reduction benefits, their classification accuracy does not surpass that of the best-performing k-NN-based methods in this dataset. These findings confirm that while all SDP-DR variants (FD, DCM, MEC) perform well on the Higgs dataset, they do not achieve the highest accuracy compared to certain k-NN-based methods, particularly LHS-FKNN, LEK-NN, and GHAHS-FKNN. SDP-DR (DCM) achieves the highest accuracy among the SDP-DR methods at $k = 7$ (0.6227), demonstrating its effectiveness in handling large datasets. However, all SDP-DR methods maintain stable classification performance while benefiting from dataset reduction, proving their efficiency in large-scale classification tasks like Higgs.

**Table 4.26:** Accuracy of Different Algorithms on Five Datasets

| *k* value | Algorithm | KDDCUP | Epsilon | Poker | SUSY | Higgs | Average |
|---|---|---|---|---|---|---|---|
| $k = 3$ | KNN-IS | 0.6854 | 0.5864 | 0.5180 | 0.7114 | 0.6402 | 0.6283 |
| | GHAHS-FKNN | 0.6864 | 0.5859 | 0.5270 | 0.7210 | 0.6401 | 0.6321 |
| | LHS-FKNN | 0.6864 | 0.5889 | 0.5365 | **0.7253** | **0.6426** | 0.6359 |
| | CQ-KNN | 0.6724 | 0.5648 | 0.5293 | 0.6910 | 0.6053 | 0.6126 |
| | CQ-EKNN | 0.6741 | 0.5612 | 0.5291 | 0.6932 | 0.6121 | 0.6139 |
| | DEKNN | 0.6890 | 0.5854 | **0.5571** | 0.7185 | 0.6410 | 0.6382 |
| | GEK-NN | 0.6968 | 0.5967 | **0.5571** | 0.7187 | 0.6413 | 0.6421 |
| | LEK-NN | 0.6994 | **0.5999** | 0.5530 | 0.7214 | 0.6414 | 0.6430 |
| | SDP-DR (FD) | 0.9970 | 0.5909 | 0.5067 | 0.7251 | 0.6076 | 0.6855 |
| | SDP-DR(DCM) | 0.9969 | 0.5896 | 0.5063 | **0.7253** | 0.6079 | 0.6852 |
| | SDP-DR(MEC) | **0.9974** | 0.5904 | 0.5149 | 0.7244 | 0.6076 | **0.6869** |
| $k = 5$ | KNN-IS | 0.6837 | 0.6006 | 0.5314 | 0.7262 | 0.6401 | 0.6364 |
| | GHAHS-FKNN | 0.6864 | 0.5868 | 0.5379 | 0.7266 | 0.6413 | 0.6358 |
| | LHS-FKNN | 0.6864 | 0.5898 | 0.5489 | 0.7254 | **0.6615** | 0.6424 |
| | CQ-KNN | 0.6731 | 0.5609 | 0.5295 | 0.6900 | 0.6092 | 0.6125 |
| | CQ-EKNN | 0.6747 | 0.5653 | 0.5295 | 0.6996 | 0.6114 | 0.6161 |
| | DEKNN | 0.6878 | 0.5990 | 0.5948 | 0.7360 | 0.6445 | 0.6524 |
| | GEK-NN | 0.6968 | 0.5982 | **0.5998** | 0.7346 | 0.6448 | 0.6548 |
| | LEK-NN | 0.6994 | 0.6007 | 0.5524 | 0.7362 | 0.6532 | 0.6484 |
| | SDP-DR (FD) | 0.9969 | **0.6060** | 0.5223 | 0.7410 | 0.6158 | 0.6964 |
| | SDP-DR(DCM) | 0.9965 | 0.6030 | 0.5220 | **0.7411** | 0.6157 | 0.6957 |

**Table 4.26** continued

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | SDP-DR(MEC) | **0.9973** | 0.6043 | 0.5283 | 0.7402 | 0.6159 | **0.6972** |
| $k = 7$ | KNN-IS | 0.6823 | 0.6110 | 0.5408 | 0.7303 | 0.6420 | 0.6413 |
| | GHAHS-FKNN | 0.6842 | 0.5879 | 0.5487 | 0.7218 | 0.6593 | 0.6404 |
| | LHS-FKNN | 0.6843 | 0.5808 | 0.5526 | 0.7295 | 0.6586 | 0.6412 |
| | CQ-KNN | 0.6727 | 0.5639 | 0.5291 | 0.7010 | 0.6193 | 0.6172 |
| | CQ-EKNN | 0.6741 | 0.5689 | 0.5276 | 0.7084 | 0.6207 | 0.6199 |
| | DEKNN | 0.6870 | 0.5871 | 0.6102 | 0.7386 | 0.6580 | 0.6562 |
| | GEK-NN | 0.6968 | 0.5991 | **0.6107** | 0.7388 | 0.6588 | 0.6608 |
| | LEK-NN | 0.6993 | 0.6118 | 0.5516 | 0.7440 | **0.6673** | 0.6548 |
| | SDP-DR (FD) | 0.9966 | **0.6152** | 0.5311 | 0.7499 | 0.6221 | 0.7030 |
| | SDP-DR(DCM) | 0.9960 | 0.6120 | 0.5294 | **0.7501** | 0.6227 | 0.7020 |
| | SDP-DR(MEC) | **0.9970** | 0.6127 | 0.5360 | 0.7491 | 0.6220 | **0.7034** |

## 4.4 Discussions

The evaluation of SDP-DR variants (FD, DCM, MEC) on small and medium-sized datasets demonstrates their effectiveness in enhancing classification accuracy while reducing computational costs. For benchmarking, this study includes Condensed Nearest Neighbour (CNN) and Edited Nearest Neighbour (ENN) as comparison methods. CNN and ENN are among the earliest and most widely cited prototype selection algorithms specifically designed to address the main limitations of k-NN: excessive storage requirements and sensitivity to noisy instances. Although k-NN itself is an older algorithm (dating back to the 1950s), CNN and ENN are historically considered its foundational extensions and are frequently adopted as baseline methods in the prototype selection literature. Their inclusion in this study is therefore not intended to position them as state-of-the-art, but rather as classical reference points. By comparing against CNN and ENN, we are able to highlight the extent to which the proposed SDP-DR method improves upon these foundational approaches in terms of accuracy, reduction time, and reduction rate. Across multiple datasets, the results show that SDP-DR consistently outperforms CNN and ENN in

both accuracy and reduction time. While CNN and ENN serve as classical baselines, more recent prototype selection methods such as RIS, ATISA, and DROP3 represent state-of-the-art approaches in the literature. Comparing SDP-DR against both categories ensures a fair evaluation of its performance, showing improvements over early foundational methods while also demonstrating competitiveness with modern techniques. Comparisons with recent prototype selection algorithms such as RIS, ATISA, and DROP3 further highlight that SDP-DR methods provide a balanced approach, achieving classification accuracy comparable to or exceeding that of the best-performing methods while ensuring efficient data reduction.

In terms of reduction time, the results indicate that CNN and ENN exhibit significantly higher processing times compared to SDP-DR variants, making them less suitable for large datasets. SDP-DR (MEC), in particular, achieves one of the lowest reduction times while maintaining high classification performance, demonstrating its suitability for real-time classification tasks. The reduction rate analysis reveals that CNN achieves the highest instance reduction, albeit at the cost of classification performance, whereas SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) achieve more balanced reductions, ensuring that important classification information is retained.

The classification accuracy results confirm that SDP-DR methods consistently outperform traditional k-NN and prototype selection techniques across small and medium datasets. SDP-DR (MEC) emerges as the best-performing variant in most cases, achieving the highest accuracy in datasets such as Segment, Appendicitis, and Yeast. The statistical significance tests further confirm that the SDP-DR variants achieve results that are significantly better than CNN and ENN, while providing comparable or superior performance to RIS, ATISA, k-NN, and DROP3.

As the $k$ value increases, the performance of SDP-DR methods improves, demonstrating their scalability and adaptability. For instance, at $k$ = 3, SDP-DR (MEC) achieves strong results, and as $k$ increases to 5 and 7, its accuracy further improves, confirming that the effectiveness of prototype selection techniques is influenced by the choice of $k$ value. This trend highlights the importance of selecting an appropriate $k$ in k-NN classification, as a higher $k$ allows for more stable decision boundaries while leveraging the benefits of prototype selection. This finding is also supported by Zhang (2021), who noted that increasing the $k$ parameter can improve the accuracy of k-NN, although the improvement is not always substantial and may not apply to all datasets.

Overall, the results for small and medium-sized datasets confirm that the SDP-DR variants are highly effective at balancing classification accuracy and data reduction. SDP-DR (MEC) consistently outperforms other methods, achieving the highest overall classification accuracy, while SDP-DR (FD) and SDP-DR (DCM) provide alternative approaches that perform well across different datasets. These findings suggest that SDP-DR is a competitive prototype selection method that enhances k-NN classification by reducing computational complexity without sacrificing classification performance.

The evaluation of SDP-DR variants (FD, DCM, MEC) on large-scale datasets (KDDCUP, Epsilon, Poker, SUSY, and Higgs) demonstrates their efficiency in reducing dataset size while maintaining high classification performance. The comparison with traditional prototype selection methods such as CNN and ENN highlights that SDP-DR significantly improves computational efficiency, as shown in the results for small and medium datasets, making it a viable solution for big data classification tasks. Additionally, the benchmarking against advanced prototype selection techniques like RIS, ATISA1, and

DROP3 further confirms that SDP-DR methods are capable of achieving comparable or superior classification accuracy while optimising data reduction.

In the KDDCUP case study dataset, the results indicate that ENN requires the longest processing time due to its iterative nearest neighbour comparisons, making it computationally expensive for large-scale datasets. CNN, while faster than ENN, still exhibits high reduction times compared to the proposed SDP-DR variants. In contrast, SDP-DR (FD), SDP-DR (DCM), and SDP-DR (MEC) significantly reduce the computational time required for the prototype selection process, proving their efficiency in handling big data. Among the three SDP-DR variants, SDP-DR (FD) achieves the lowest reduction time, reinforcing its capability for scalable prototype selection in large datasets like KDDCUP.

The reduction rate analysis on the KDDCUP dataset further reveals that CNN achieves the highest reduction rate by aggressively removing instances, but this comes at the cost of lower classification accuracy and higher classification time, as shown in previous small and medium dataset examples, making it unsuitable for big datasets. ENN maintains a moderate reduction rate but remains computationally inefficient, making it unsuitable for large datasets due to its extended processing time. The SDP-DR variants, on the other hand, achieve a balanced reduction rate, effectively reducing dataset size while retaining critical classification information. This balance ensures that SDP-DR methods do not compromise classification accuracy while still significantly reducing computational complexity, making them a more practical choice for large-scale data reduction compared to CNN and ENN.

In terms of classification accuracy, the SDP-DR variants consistently outperform CNN and perform competitively with ENN and k-NN. CNN records the lowest mean accuracy at 87.34%, highlighting the downside of its aggressive reduction strategy. While

ENN and k-NN achieve slightly higher accuracies of 99.81% and 99.82% respectively, they do so with significant computational costs. In contrast, SDP-DR (MEC), SDP-DR (FD), and SDP-DR (DCM) maintain high accuracies of 99.75%, 99.73%, and 99.72%, respectively. These results confirm that the SDP-DR methods successfully balance accuracy and efficiency, making them robust and scalable prototype selection techniques for large-scale classification tasks such as KDDCUP.

Beyond the KDDCUP case study, analysis across other big data datasets further supports the effectiveness of the SDP-DR variants. In datasets such as SUSY and KDDCUP when comparing with state-of-art algorithm, all SDP-DR variants (FD, DCM, and MEC) consistently outperform most traditional prototype selection techniques, with SDP-DR (MEC) achieving the highest accuracy across all tested $k$ values in KDDCUP. For the Epsilon dataset, the SDP-DR variants demonstrate superior accuracy at specific $k$ values, particularly at $k = 5$ and 7. In contrast, for the Poker dataset, other k-NN based methods such as GEK-NN and DEKNN achieve better classification accuracy than SDP-DR. Similarly, for the Higgs dataset, while SDP-DR methods maintain stable performance across various $k$ values, they do not outperform the top-performing methods like LEK-NN and GEK-NN. These results suggest that although SDP-DR is highly competitive and efficient, its effectiveness can vary depending on dataset characteristics, and some datasets may benefit more from alternative reduction strategies.

Additionally, the positive impact of increasing $k$ values is evident across all big data datasets. As $k$ increases from 3 to 7, the classification accuracy of the SDP-DR variants improves consistently, reinforcing that larger $k$ values enhance the effectiveness of prototype selection. SDP-DR (MEC), in particular, shows the greatest accuracy improvement as $k$

increases, confirming that it benefits from more stable decision boundaries while maintaining the advantages of dataset reduction. However based on studies done by Gong et al. (2023b) highlights that the impact of increasing *k* is not universally positive; while some datasets show improved accuracy, others experience a decline.

**Table 4.27:** Average Accuracy of Different Algorithms on Epsilon, Poker, SUSY, Higgs Datasets

| Algorithm | Average Accuracy for *k* =3 | Average Accuracy for *k* =5 | Average Accuracy for *k* =7 |
|---|---|---|---|
| KNN-IS | 0.61686 | 0.62694 | 0.63308 |
| GHAHS-FKNN | 0.62122 | 0.62568 | 0.63162 |
| LHS-FKNN | 0.62584 | 0.6336 | 0.63254 |
| CQ-KNN | 0.6006 | 0.60042 | 0.6061 |
| CQ-EKNN | 0.6019 | 0.60438 | 0.6091 |
| DEKNN | 0.62804 | 0.64534 | 0.65002 |
| GEK-NN | 0.63118 | **0.64644** | **0.65364** |
| LEK-NN | **0.63174** | 0.63818 | 0.6459 |
| SDP-DR (FD) | 0.62316 | 0.6363 | 0.64426 |
| SDP-DR(DCM) | 0.62286 | 0.6355 | 0.64324 |
| SDP-DR(MEC) | 0.62484 | 0.63718 | 0.64464 |

Table 4.27 reports the average classification accuracy of different algorithms on the Epsilon, Poker, SUSY, and Higgs datasets (k = 3, 5, 7). The KDDCUP dataset is excluded from this comparison, as the performance of our proposed method on that dataset was significantly superior to the state-of-the-art methods, and its inclusion would overshadow the comparative results. As shown, the proposed SDP-DR variants (FD, DCM, MEC) consistently outperform the baseline kNN-IS method across all k values and also achieve higher accuracy than several established prototype selection approaches such as CQ-KNN, CQ-EKNN, and GHAHS-FKNN. Although methods such as LEK-NN and GEK-NN report slightly higher average accuracies, it is important to note that these algorithms incorporate additional learning mechanisms, making them significantly more complex and computationally demanding. In contrast, SDP-DR does not rely on learning-based

enhancements, yet it delivers accuracy levels that are very close to these advanced methods. This finding highlights a key advantage of SDP-DR: it achieves a strong balance between accuracy and simplicity. The ability of SDP-DR to remain competitive with learning-based algorithms, while retaining a straightforward and efficient reduction mechanism, demonstrates its practical value for large-scale classification tasks where both accuracy and computational efficiency are critical.

Based on comparative algorithms, hyperparameter tuning is essential, particularly for methods such as LHS-FkNN and GAHS-FkNN. These algorithms require manual intervention for optimal parameter tuning, which can impact their efficiency and ease of use. In contrast, our proposed method significantly simplifies the parameter selection process, requiring users only to select the type of anchor for data reduction. This makes it considerably easier to implement than the benchmark algorithms.

When using Apache Spark, users must carefully tune the number of map and reduce tasks to achieve optimal classification speed. Determining the appropriate number of these tasks can be complex, as it heavily depends on hardware capabilities such as the amount of Random Access Memory (RAM) and the number of available processing cores. Consequently, conducting big data experiments with Apache Spark usually requires at least two high-performance personal computers, significantly increasing costs. Conversely, our experiments with Dask utilised only parallel computing on a single personal computer, considerably reducing expenses. Although Dask supports distributed computing similar to Apache Spark and Hadoop, parallel computing alone was sufficient in our case due to the simplicity of our reduction method and Dask's efficient RAM management during data reduction and classification processes.

Moreover, in Apache Spark, a higher number of map tasks required to handle large datasets substantially increases the computational load during the reduce phase. Larger datasets, such as Higgs, exacerbate this issue, necessitating multiple reducers for efficient processing. Hence, balancing the number of map and reduce tasks according to dataset size and available hardware resources is crucial to maintain performance and prevent degradation (Gong et al., 2023b; Maillo et al., 2017).

Our findings reveal that the classification accuracy achieved on the KDDCUP dataset is significantly higher compared to several state-of-the-art methods, as summarised in Table 4.&. Specifically, while the accuracy reported for the standard kNN-IS approach is 68.54%, our proposed method achieves a mean accuracy of 99.82%. This substantial difference may be attributed to variations in data distribution strategies and preprocessing techniques, particularly in how the dataset is partitioned in the Spark framework used by previous studies. In our case, we employed the k-NN implementation from the well-established scikit-learn library and utilised the Dask framework to ensure scalability. Given the reliability of these tools, we can reasonably assume the correctness of our implementation.

Moreover, it is important to note that both experiments used the same version of the KDDCUP dataset obtained from the UCI Machine Learning Repository, which eliminates dataset inconsistency as a possible cause of the discrepancy. Additionally, the KDDCUP dataset is known to exhibit a highly imbalanced class distribution, as indicated in Table 3.3. This imbalance may have further influenced the classification results, especially considering the differences in data handling between the Spark and Dask frameworks.

Our findings also indicate that using the mean of each column as an anchor point (MEC) produces superior outcomes compared to other anchoring strategies. This is

demonstrated by the stable performance in small, medium, and large datasets. The SDP-DR (MEC) method effectively plots distances from each data point to the mean attribute of each class, differing notably from strategies such as First Data (FD) and Data Close to Mean (DCM), which use existing data points as anchors. Ideally, data points from the same class cluster closely, while points from different classes maintain greater distances from FD and DCM anchor points. Between FD and DCM, the latter consistently delivers better performance due to its proximity to the mean attribute point. Conversely, FD anchor positioning can vary significantly, affecting both storage sets and classification outcomes. Both DCM and MEC demonstrate stable and consistent results across multiple trials, underlining their robustness and reliability.

The consistent performance of our SDP-DR models highlights their robustness and reliability, particularly when compared to prototype selection (PS) approaches that rely on random sampling (Cavalcanti & Soares, 2020; Garcia et al., 2012). Techniques based on randomness often produce variable prototype sets, which can undermine their practical effectiveness. In contrast, the stability of our approach offers a key advantage for real-world applications, where consistent and dependable outcomes are essential. While the literature highlights dimensionality reduction as a common strategy for improving the general classification performance of the k-NN algorithm, the findings of this study demonstrate that such improvements can still be achieved without relying on dimensionality reduction techniques. These results further reinforce that dimensionality reduction is not essential for enhancing the overall performance of k-NN.

## 4.5 Summary

This chapter presented the experimental results and analysis of the proposed method. The evaluation was conducted using multiple benchmark datasets from the UCI Machine Learning Repository and the KEEL dataset repository. The experiments aimed to assess the effectiveness of the proposed method in terms of classification accuracy, classification time, reduction rate, and reduction time. Comparative analyses were performed between the proposed SDP-DR variants and several state-of-the-art data reduction techniques, including Edited Nearest Neighbour (ENN) and Condensed Nearest Neighbour (CNN). The k-Nearest Neighbour (k-NN) classifier was used as the baseline model for performance evaluation.

The findings demonstrated that the proposed method consistently outperformed state-of-the-art approaches in both efficiency and effectiveness across several datasets, notably Appendicitis, KDDCUP, and SUSY. The proposed method also consistently achieved lower reduction time and classification time compared to ENN and CNN. Furthermore, the results highlighted the importance of Dask framework, which contributed to improved scalability when handling large-scale datasets. Overall, the experimental results validate the robustness and effectiveness of the proposed method (SDP-DR), particularly in enhancing classification performance while reducing computational overhead.

# CHAPTER 5

# CONCLUSION AND FUTURE WORKS

## 5.1 Introduction

This chapter concludes the study by summarizing the main findings and highlighting how the stated research objectives have been achieved. It also outlines the contributions of the study to the field of data reduction and classification, and finally, it presents directions for future research.

## 5.2 Achievement Discussion

**Objective i:** To investigate existing prototype selection techniques, parallel and distributed computing approaches, and similarity distance measures used to enhance k-NN classification methods particularly in the context of big dataset.

This objective was successfully achieved. A comprehensive investigation was conducted on existing prototype selection and data reduction techniques such as Condensed Nearest Neighbor (CNN), Edited Nearest Neighbor (ENN), DROP3, ATISA, and RIS. Their advantages, limitations, and suitability for large-scale data were analyzed to identify gaps in accuracy, reduction rate, and computational efficiency. In addition, various similarity distance measures (e.g., Euclidean, Manhattan, and Minkowski) were examined to determine their impact on k-NN performance. The study also explored parallel and distributed computing frameworks such as Apache Spark and Dask to address scalability issues in big data classification. The insights gained from this investigation formed the foundation for designing the proposed SDP-DR method, which integrates data reduction with parallel computing to enhance k-NN efficiency and generalization.

**Objective ii:** To develop a Similarity Distance Plot–Data Reduction technique integrated with Dask, incorporating the nearest neighbour rule

This objective was successfully achieved. The SDP-DR method was designed as a novel prototype selection technique that leverages similarity distance plotting to reduce training data before classification. The method incorporated three anchor strategies such as First Data (FD), Data Close to Mean (DCM), and Mean of Each Column (MEC) to execute prototype selection. The integration of the Dask framework enabled parallel processing, reducing computational time and allowing the method to handle both small-scale and large-scale datasets efficiently.

**Objective iii:** To evaluate the effectiveness and efficiency of the proposed method in comparison with the original kNN-IS algorithm and several state-of-the-art data reduction methods in terms of classification performance, prototype storage, classification time, and reduction time.

This objective was successfully achieved. Experimental evaluations on small, medium, and big datasets demonstrated that the proposed SDP-DR method achieved competitive or superior classification accuracy compared to existing data reduction algorithms. SDP-DR consistently outperformed traditional methods such as CNN and ENN in accuracy and performed comparably or better than recent approaches, including RIS, ATISA1, and DROP3. On large-scale datasets such as KDDCUP and Epsilon, SDP-DR achieved remarkable accuracy, with the MEC variant reaching 0.9974 on KDDCUP, surpassing LEK-NN, which achieved only 0.6994. In terms of efficiency, SDP-DR substantially reduced computation time compared to the original k-NN algorithm, particularly when integrated with the Dask framework. Although CNN achieved a higher

reduction rate, SDP-DR consistently recorded faster reduction times than both CNN and ENN while maintaining a balanced trade-off between accuracy and efficiency. Classification time was slightly slower than ENN but remained competitive overall, with the MEC variant consistently demonstrating the best balance between performance and computational efficiency across datasets.

## 5.3 Contributions of the Study

The main contributions of this research can be summarised as follows:

i. Methodological Contribution:
    a. Proposed a new prototype selection technique, SDP-DR, that applies similarity distance plotting as a basis for data reduction. Concept not previously utilised for this purpose.
    b. Introduced three anchor-based strategies (FD, DCM, MEC) to guide prototype selection, with MEC demonstrating the most consistent general classification performance.

ii. Experimental Contribution:
    a. Conducted extensive evaluation across small, medium, and big datasets, comparing SDP-DR with classical (CNN, ENN, DROP3) and modern (RIS, ATISA1, LEK-NN, kNN-IS) data reduction algorithms.
    b. Demonstrated the statistical significance of SDP-DR's performance improvements using the Sign Test and Wilcoxon Test.

iii. Practical Contribution:
    a. Implemented SDP-DR in Python with Dask integration, providing an efficient yet lightweight alternative to distributed frameworks like Apache Spark and Apache Hadoop.

b. Demonstrated that SDP-DR can handle datasets with millions of instances on a single PC, making it more accessible to users without high-performance computing resources.

iv. Theoretical Contribution:

a. Highlighted the critical role of anchor selection in prototype selection.

b. Provided evidence that SDP-DR generalises well across datasets without requiring hyperparameter tuning beyond the k value of k-NN.

## 5.4 Conclusion

In conclusion, this study identified and addressed two key challenges associated with k-NN: high memory requirements and reduced classification speed when applied to large datasets. The proposed SDP-DR method, enhanced with Dask parallelism, successfully reduced dataset size while maintaining classification accuracy, achieving competitive performance against both classical and state-of-the-art data reduction techniques. The findings confirmed the importance of anchor selection, with the MEC variant consistently delivering the best balance of accuracy and efficiency.

## 5.5 Future Works

During the evaluation and analysis stages of this study, several limitations and opportunities for future research were identified. These can be categorised into the following four key directions:

**Exploring other Classification Algorithms:** In Chapter 4, the k-NN algorithm was employed for classification because of the specific limitations associated with it, as discussed in Chapter 1. Our findings demonstrated that the proposed SDP-DR method effectively

addresses these limitations. Therefore, a promising direction for future research is to investigate whether SDP-DR can also enhance the performance of other classification algorithms. This could provide broader insights into the generalisability and versatility of the proposed data reduction method.

**Investigating Alternative Similarity Distances:** This study employed only the Euclidean distance as the similarity measure for both the k-NN classification and the SDP-DR reduction process. However, other distance measures were not explored in this study. These include members of the Minkowski family, such as Manhattan and Chebyshev distances, as well as similarity measures like Hassanat and Jaccard distances that were discussed in the literature review (Alfeilat et al., 2019; Muniswamaiah et al., 2023). Future work should examine the use of these alternative similarity distances to determine whether they can further improve classification performance when integrated with SDP-DR.

**Integration of Dimensionality Reduction in SDP-DR:** As discussed in Chapters 1 and 2, dimensionality reduction is another approach commonly used to address the challenges faced by k-NN, particularly in high-dimensional datasets where the curse of dimensionality becomes prominent. However, this study did not incorporate any dimensionality reduction techniques within the proposed SDP-DR framework. Future work is encouraged to explore the potential benefits of integrating such techniques to further enhance overall classification performance.

**Utilising Distributed Computing for Big Data:** In this study, we used the Dask framework to enable parallel computing for handling big datasets. Although effective, Dask is limited compared to full-scale distributed computing systems. As noted in Chapter 2, many researchers have utilised distributed frameworks such as Apache Spark for big data

classification tasks. Due to hardware constraints, this study could not implement distributed computing. Therefore, future research should investigate the integration of SDP-DR with distributed computing platforms like Apache Spark. This would enable a more robust comparison with other distributed k-NN methods, as presented in Chapter 4, Phase 2, to evaluate both the efficiency and effectiveness of the proposed method in distributed environments.

## REFERENCES


Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL Data-Mining Software Tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*, 255-287. https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/pr/2010-JMVLSC-Alcala_Fdez-KEEL-dataset.pdf

Alexandropoulos, S. N., Kotsiantis, S. B., & Vrahatis, M. N. (2019). Data preprocessing in predictive data mining. *The Knowledge Engineering Review*, *34*. https://doi.org/10.1017/s026988891800036x

Alfeilat, H. A. A., Hassanat, A. B. A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Salman, H. S. E., & Prasath, V. B. S. (2019). Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data*, *7*(4), 221-248. https://doi.org/10.1089/big.2018.0175

Ali, M., Jung, L. T., Abdel-Aty, A. H., Abubakar, M. Y., Elhoseny, M., & Ali, I. (2020). Semantic-k-NN algorithm: An enhanced version of traditional k-NN algorithm. *Expert Systems with Applications*, *151*, 18, Article 113374. https://doi.org/10.1016/j.eswa.2020.113374

Arora, I., Khanduja, N., & Bansal, M. (2022). Effect of Distance Metric and Feature Scaling on KNN Algorithm while Classifying X-rays. *In RIF*, 61-75.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. *Symposium on Discrete Algorithms*, 1027–1035. https://doi.org/10.5555/1283383.1283494

Aslani, M., & Seipel, S. (2020). A fast instance selection method for support vector machines in building extraction. *Applied Soft Computing*, *97*, 106716. https://doi.org/10.1016/j.asoc.2020.106716

Aslani, M., & Seipel, S. (2021). Efficient and decision boundary aware instance selection for support vector machines. *Information Sciences*, *577*, 579-598. https://doi.org/10.1016/j.ins.2021.07.015

Asuncion, A. (2007). UCI Machine Learning Repository. *CiNii Research*. https://doi.org/http://ci.nii.ac.jp/naid/20001247967

Bellman, R. (1962). Adaptive control processes: a guided tour. *Journal of the Royal Statistical Society Series A (General)*, *125*(1), 161. https://doi.org/10.2307/2343225

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is "nearest neighbor" meaningful? In *Lecture notes in computer science* (pp. 217-235). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49257-7_15

Bharill, N., Tiwari, A., Malviya, A., Patel, O. P., Gupta, A., Puthal, D., Saxena, A., & Prasad, M. (2020). Fuzzy knowledge based performance analysis on big data. *Neurocomputing*, *389*, 218-228. https://doi.org/10.1016/j.neucom.2018.10.088

Bi, Y., Xue, B., & Zhang, M. (2021). Instance selection-based surrogate-assisted genetic programming for feature learning in image classification. *Ieee Transactions on Cybernetics*, *53*(2), 1118-1132. https://doi.org/10.1109/tcyb.2021.3105696

Castellanos, F. J., Valero-Mas, J. J., & Calvo-Zaragoza, J. (2021). Prototype generation in the string space via approximate median for data reduction in nearest neighbor classification. *Soft Computing*, *25*(24), 15403-15415. https://doi.org/10.1007/s00500-021-06178-2

Cavalcanti, G. D., & Soares, R. J. (2020). Ranking-based instance selection for pattern classification. *Expert Systems with Applications*, *150*, 113269. https://doi.org/10.1016/j.eswa.2020.113269

Cavalcanti, G. D. C., Ren, T. I., & Pereira, C. L. (2013). ATISA: adaptive threshold-based instance selection algorithm. *Expert systems with applications*, *40*(17), 6894-6900. https://doi.org/10.1016/j.eswa.2013.06.053

Cha, S. H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, *1*. https://tcs.ah-epos.eu/eprints/1372/

Chi, M., Plaza, A., Benediktsson, J. A., Sun, Z., Shen, J., & Zhu, Y. (2016). Big data for remote sensing: challenges and opportunities. *Proceedings of the IEEE*, *104*(11), 2207-2219, Article 7565634. https://doi.org/10.1109/jproc.2016.2598228

Chi, Z., Yan, H., & Pham, T. (1996). *Fuzzy algorithms: with applications to image processing and pattern recognition* (Vol. 10). World Scientific. https://doi.org/10.1142/3132

Chuang, L., Tsai, S., & Yang, C. (2011). Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications*, *38*(10), 12699-12707. https://doi.org/10.1016/j.eswa.2011.04.057

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT Press.

Cunha, W., Viegas, F., França, C., Rosa, T., Rocha, L., & Gonçalves, M. A. (2023). A comparative survey of instance selection methods applied to non-neural and transformer-based text classification. *ACM Computing Surveys*, *55*(13s), 1-52. https://doi.org/10.1145/3582000

Cunningham, P., & Delany, S. J. (2021). K-nearest neighbour classifiers - a tutorial. *ACM Computing Surveys*, *54*(6), 1-25, Article 128. https://doi.org/10.1145/3459665

Elkano, M., Galar, M., Sanz, J., & Bustince, H. (2018). CHI-PG: A fast prototype generation algorithm for big data classification problems. *Neurocomputing*, *287*, 22-33. https://doi.org/10.1016/j.neucom.2018.01.056

Escalante, H. J., Graff, M., & Morales-Reyes, A. (2015). PGGP: prototype generation via genetic programming. *Applied Soft Computing*, *40*, 569-580. https://doi.org/10.1016/j.asoc.2015.12.015

Fuangkhon, P. (2021). Effect of the distance functions on the distance-based instance selection for the feed-forward neural network. *Evolutionary Intelligence*, *15*(3), 1991-2015. https://doi.org/10.1007/s12065-021-00607-9

Garcia, S., Derrac, J., Cano, J. R., & Herrera, F. (2012). Prototype selection for nearest neighbor classification: taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(3), 417-435. https://doi.org/10.1109/tpami.2011.142

Gautam, A., & Chatterjee, I. (2020). Big data and cloud computing. *International Journal of Operations Research and Information Systems*, *11*(3), 19-38. https://doi.org/10.4018/ijoris.2020070102

Gong, C., Demmel, J., & You, Y. (2023a). Distributed and joint evidential k-nearest neighbor classification. *IEEE Transactions on Knowledge and Data Engineering*, *36*(11), 5972-5985. https://doi.org/10.1109/tkde.2023.3341098

Gong, C., Demmel, J., & You, Y. (2023b). Scalable evidential k-nearest neighbor classification on big data. *IEEE Transactions on Big Data*, *10*(3), 226-237. https://doi.org/10.1109/tbdata.2023.3327220

Gong, C., Su, Z., Wang, P., Wang, Q., & You, Y. (2021). Evidential instance selection for k-nearest neighbor classification of big data. *International Journal of Approximate Reasoning*, *138*, 123-144. https://doi.org/10.1016/j.ijar.2021.08.006

Gong, C., Su, Z., Wang, Q., & You, Y. (2022). Distributed EK-NN classification. In *Lecture notes in computer science* (pp. 99-108). https://doi.org/10.1007/978-3-031-17801-6_10

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

He, N. H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263-1284. https://doi.org/10.1109/tkde.2008.239

Hernandez-Leal, P., Carrasco-Ochoa, J. A., Martínez-Trinidad, J., & Olvera-Lopez, J. A. (2013). InstanceRank based on borders for instance selection. *Pattern Recognition*, *46*(1), 365-375. https://doi.org/10.1016/j.patcog.2012.07.007

Hu, L., Huang, M., Ke, S., & Tsai, C. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *Springerplus*, *5*(1). https://doi.org/10.1186/s40064-016-2941-7

Jankowski, N., & Orliński, M. (2019). Fast algorithm for prototypes Selection—Trust-Margin prototypes. *Lecture notes in computer science*, 583-594. https://doi.org/10.1007/978-3-030-20912-4_53

Jaruenpunyasak, J., & Duangsoithong, R. (2021). Empirical analysis of feature reduction in deep learning and conventional methods for foot image classification. *IEEE Access*, *9*, 53133-53145. https://doi.org/10.1109/access.2021.3069625

Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, *8*(3), 2663-2693. https://doi.org/10.1007/s40747-021-00637-x

Jiménez, F., Sánchez, G., Palma, J., & Sciavicco, G. (2022). Three-objective constrained evolutionary instance selection for classification: wrapper and filter approaches. *Engineering Applications of Artificial Intelligence*, *107*, 104531. https://doi.org/10.1016/j.engappai.2021.104531

Juez-Gil, M., Arnaiz-González, A., Rodríguez, J. J., López-Nozal, C., & García-Osorio, C. (2021). Approx-SMOTE: fast SMOTE for big data on apache spark. *Neurocomputing*, *464*, 432-437. https://doi.org/10.1016/j.neucom.2021.08.086

Kalra, V., Kashyap, I., & Harmeet, K. (2022). Effect of distance measures on k-nearest neighbour classifier. *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 1-7. https://doi.org/10.1109/iccsea54677.2022.9936314

Kasemtaweechok, C., & Suwannik, W. (2016). Prototype selection for k-nearest neighbors classification using geometric median. *Proceedings of the Fifth International Conference on Network, Communication and Computing*, 140-144. https://doi.org/10.1145/3033288.3033301

Kumar, M. (2023). Distributed execution of dask on HPC: a case study. *World Conference on Communication & Computing (WCONF)*, 1-4. https://doi.org/10.1109/wconf58270.2023.10234994

Kumar, R., & Singla, S. (2021). Classification rule discovery for software bug severity using knn with different distance metric. *Indian Journal of Computer Science and Engineering*, *12*(4), 841-847. https://doi.org/10.21817/indjcse/2021/v12i4/211204092

Kumar, S., & Mohbey, K. K. (2019). A review on big data based parallel and distributed approaches of pattern mining. *Journal of King Saud University-Computer and Information Sciences*, *34*(5), 1639-1662. https://doi.org/10.1016/j.jksuci.2019.09.006

Laney, D. (2001). *3D data management: controlling data volume, velocity and variety* (META group research note, Issue. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85136685710&partnerID=40&md5=fd848f021150f0279ef0cd4c30c6cbd5

Li, J., & Dai, C. (2022). Fast prototype selection algorithm based on adjacent neighbourhood and boundary approximation. *Scientific Reports*, *12*(1), 20108, Article 20108. https://doi.org/10.1038/s41598-022-23036-9

Li, M., Wang, H., Yang, L., Liang, Y., Shang, Z., & Wan, H. (2020). Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, *150*, 10, Article 113277. https://doi.org/10.1016/j.eswa.2020.113277

Li, S., Dragicevic, S., Castro, F. A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., & Cheng, T. (2015). Geospatial big data handling theory and methods: A review and research challenges. *Isprs Journal of Photogrammetry and Remote Sensing*, *115*, 119-133. https://doi.org/10.1016/j.isprsjprs.2015.10.012

Liu, H., Zhang, S., Zhao, J., Zhao, X., & Mo, Y. (2010). A new classification algorithm using mutual nearest neighbors. *International Conference on Grid and Cloud Computing*, 52-57. https://doi.org/10.1109/gcc.2010.23

Liu, P. (2015). A survey of remote-sensing big data. *Frontiers in Environmental Science*, *3*. https://doi.org/10.3389/fenvs.2015.00045

Ma, J., Cheng, J. C. P., Jiang, F., Chen, W., & Zhang, J. (2020). Analyzing driving factors of land values in urban scale based on big data and non-linear machine learning techniques. *Land Use Policy*, *94*, 104537. https://doi.org/10.1016/j.landusepol.2020.104537

Mailagaha Kumbure, M., & Luukka, P. (2024). Local means-based fuzzy k-nearest neighbor classifier with Minkowski distance and relevance-complementarity feature weighting. *Granular Computing*, *9*(4), 73. https://doi.org/10.1007/s41066-024-00496-0

Maillo, J., García, S., Luengo, J., Herrera, F., & Triguero, I. (2019). Fast and scalable approaches to accelerate the fuzzy k-nearest neighbors classifier for big data. *IEEE Transactions on Fuzzy Systems*, *28*(5), 874-886. https://doi.org/10.1109/tfuzz.2019.2936356

Maillo, J., Raamírez, S., Triguero, I., & Herrera, F. (2017). Knn-is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, *117*, 3-15. https://doi.org/10.1016/j.knosys.2016.06.012

Malhat, M., Menshawy El, M., Mousa, H., & El Sisi, A. (2020). A new approach for instance selection: Algorithms, evaluation, and comparisons. *Expert Systems with Applications*, *149*, 113297. https://doi.org/10.1016/j.eswa.2020.113297

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the Python in Science Conferences*, 56-61. https://doi.org/10.25080/majora-92bf1922-00a

Mostafaeipour, A., Rafsanjani, A. J., Ahmadi, M., & Dhanraj, J. A. (2021). Investigating the performance of Hadoop and Spark platforms on machine learning algorithms. *The Journal of Supercomputing*, *77*(2), 1273-1300. https://doi.org/10.1007/s11227-020-03328-5

Muniswamaiah, M., Agerwala, T., & Tappert, C. C. (2023). Applications of binary similarity and distance measures. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2307.00411

Nalepa, J., & Kawulok, M. (2019). Selecting training sets for support vector machines: a review. *Artificial Intelligence Review*, *52*(2), 857-900. https://doi.org/10.1007/s10462-017-9611-1

Nanni, L., & Lumini, A. (2011). Prototype reduction techniques: A comparison among different approaches. *Expert Systems with Applications*, *38*(9), 11820-11828. https://doi.org/10.1016/j.eswa.2011.03.070

Narisetty, N. N. (2020). Bayesian model selection for high-dimensional data. In *Handbook of statistics* (Vol. 43, pp. 207-248). Elsevier. https://doi.org/10.1016/bs.host.2019.08.001

Nguyen, S., Tran, B., & Alahakoon, D. (2020). Dynamic self-organising swarm for unsupervised prototype generation. *2022 IEEE Congress on Evolutionary Computation (CEC)*, 1-8. https://doi.org/10.1109/cec48606.2020.9185816

Nobre, J., & Neves, R. F. (2019). Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. *Expert Systems with Applications*, *125*, 181-194. https://doi.org/10.1016/j.eswa.2019.01.083

Ougiaroglou, S., Filippakis, P., & Evangelidis, G. (2021). Prototype generation for multi-label nearest neighbours classification. In *Lecture notes in computer science* (pp. 172-183). Springer. https://doi.org/10.1007/978-3-030-86271-8_15

Ougiaroglou, S., Filippakis, P., Fotiadou, G., & Evangelidis, G. (2023). Data reduction via multi-label prototype generation. *Neurocomputing*, *526*, 1-8. https://doi.org/10.1016/j.neucom.2023.01.004

Papanikolaou, M., Evangelidis, G., & Ougiaroglou, S. (2021). Dynamic k determination in k-NN classifier: A literature review. *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1-8. https://doi.org/10.1109/iisa52424.2021.9555525

Pedregosa, Fabian, Gaël, V., Alexandre, G., Vincent, M., Bertrand, T., Olivier, G., Mathieu, B., Peter, P., Ron, W., Vincent, D., & others. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825-2830. https://doi.org/10.5555/1953048.2078195

Pizer, S. M., Hong, J., Vicory, J., Liu, Z., Marron, J. S., Choi, H.-y., Damon, J., Jung, S., Paniagua, B., & Schulz, J. (2020). Object shape representation via skeletal models (s-reps) and statistical analysis. In *Riemannian geometric statistics in medical image analysis* (pp. 233-271). Elsevier. https://doi.org/10.1016/b978-0-12-814725-2.00014-5

Rani, R., Singh, A. P., & Kumar, R. (2019). Impact of reduction in descriptor size on object detection and classification. *Multimedia Tools and Applications*, *78*(7), 8965-8979. https://doi.org/10.1007/s11042-018-6911-7

Rezaei, M., & Nezamabadi-pour, H. (2022). A hybridization method of prototype generation and prototype selection for k-nn rule based on GSA. *10*, 257-268. https://doi.org/10.22044/jadm.2021.10159.2154

Rezanejad, A., & Danesh, A. S. (2024). Improving the performance of the k-nearest neighbors algorithm with parallelization in Dask. *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, 1-5. https://doi.org/10.1109/aisp61396.2024.10475304

Rico-Juan, J. R., & Iñesta, J. M. (2012). New rank methods for reducing the size of the training set using the nearest neighbor rule. *Pattern Recognition Letters*, *33*(5), 654-660. https://doi.org/10.1016/j.patrec.2011.07.019

Rico-Juan, J. R., Valero-Mas, J. J., & Calvo-Zaragoza, J. (2019). Extensions to rank-based prototype selection in k-Nearest Neighbour classification. *Applied Soft Computing*, *85*, 11, Article 105803. https://doi.org/10.1016/j.asoc.2019.105803

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. *Proceedings of the Python in Science Conferences*, 126-132. https://doi.org/10.25080/majora-7b98e3ed-013

Rodriguez-Martinez, C., & Torres-Sospedra, J. (2021). Revisiting the analysis of hyperparameters in k-nn for wi-fi and BLE fingerprinting: Current status and general results. *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, *1*, 1-8. https://doi.org/10.1109/ipin51156.2021.9662542

Rong, M., Gong, D., & Gao, X. (2019). Feature selection and Its use in big data: challenges, methods, and trends. *IEEE Access*, *7*, 19709-19725. https://doi.org/10.1109/access.2019.2894366

Shirkhorshidi, A. S., Aghabozorgi, S., & Wah, T. Y. (2015). A comparison study on similarity and dissimilarity measures in clustering continuous data. *PLoS One*, *10*(12). https://doi.org/10.1371/journal.pone.0144059

Sisodia, D., & Sisodia, D. S. (2022). Quad division prototype selection-based k-nearest neighbor classifier for click fraud detection from highly skewed user click dataset. *Engineering Science and Technology an International Journal*, *28*, 12. https://doi.org/10.1016/j.jestch.2021.05.015

Solorio-Fernández, S., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2019). A review of unsupervised feature selection methods. *Artificial Intelligence Review*, *53*(2), 907-948. https://doi.org/10.1007/s10462-019-09682-y

Srinivas, S., & Kancharla, G. R. (2019). Feature selection in big data using filter based techniques. *4th MEC International Conference on Big Data and Smart City (ICBDSC)*, 1-7. https://doi.org/10.1109/icbdsc.2019.8645573

Su, Z.-g., Hu, Q., & Denoeux, T. (2021). A distributed rough evidential k-nn classifier: integrating feature reduction and classification. *IEEE Transactions on Fuzzy Systems*, *29*(8), 2322-2335. https://doi.org/10.1109/tfuzz.2020.2998502

Suárez, J. L., García, S., & Herrera, F. (2021). Distance metric learning with prototype selection for imbalanced classification. In *Hybrid Artificial Intelligent Systems* (pp. 391-402). Springer International Publishing. https://doi.org/10.1007/978-3-030-86271-8_33

Syriopoulos, P. K., Kalampalikis, N. G., Kotsiantis, S. B., & Vrahatis, M. N. (2025). k NN Classification: a review. *Annals of Mathematics and Artificial Intelligence*, *93*(1), 43-75. https://doi.org/10.1007/s10472-023-09882-x

Triguero, I., Derrac, J., Garcia, S., & Herrera, F. (2012). A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(1), 86-100. https://doi.org/10.1109/tsmcc.2010.2103939

Triguero, I., Peralta, D., Bacardit, J., García, S., & Herrera, F. (2015). MRPR: A MapReduce solution for prototype reduction in big data classification. *Neurocomputing*, *150*, 331-345. https://doi.org/10.1016/j.neucom.2014.04.078

Valero-Mas, J., J., Gallego, A. J., Alonso-Jiménez, P., & Serra, X. (2023). Multilabel prototype generation for data reduction in k-nearest neighbour classification. *Pattern Recognition*, *135*, Article 109190. https://doi.org/10.1016/j.patcog.2022.109190

Wang, H., Yang, M., & Stufken, J. (2018). Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association*, *114*(525), 393-405. https://doi.org/10.1080/01621459.2017.1408468

Wang, J., & Zhang, T. (2018). Composite quantization. *IEEE transactions on pattern analysis and machine intelligence*, *41*(6), 1308-1322. https://doi.org/10.1109/tpami.2018.2835468

Wang, Q., Ouyang, X., & Zhan, J. (2019). A classification algorithm based on data clustering and data reduction for intrusion detection system over big data. *KSII Transactions on Internet and Information Systems*, *13*(7), 3714-3732. https://doi.org/10.3837/tiis.2019.07.021

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*(3), 257-286. https://doi.org/10.1023/A:1007626913721

Xing, W., & Bei, Y. (2020). Medical health big data classification based on knn classification algorithm. *IEEE Access*, *8*, 28808-28819. https://doi.org/10.1109/access.2019.2955754

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

Zhang, S., Li, X., Zong, M., Zhu, X., & Wang, R. (2018). Efficient knn classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1774-1785. https://doi.org/10.1109/tnnls.2017.2673241

Zhang, S. (2021). Challenges in knn classification. *IEEE Transactions on Knowledge and Data Engineering*, *34*(10), 4663-4675. https://doi.org/10.1109/tkde.2021.3049250

Zu, B., Xia, K., Li, T., He, Z., Li, Y., Hou, J., & Du, W. (2019). SLIC superpixel-based l2,1-norm robust principal component analysis for hyperspectral image classification. *Sensors (Basel)*, *19*(3), 479. https://doi.org/10.3390/s19030479

# APPENDICES

**Appendix A:** List of Publications

1. Rushdi, A. M., & Hossin, M. (n.d.). Innovative Application of Similarity Distance Measure for Data Reduction. *International Conference on Informatics, Computing and Applied Mathematics*. - Scopus (Accepted and Waiting for Publication).
2. Rushdi, A. M., Hossin, M., & Norwawi, N. (n.d.). Improving Support Vector Machine Classification Training using Modified Similarity Distance Plotting-Data Reduction. *International Journal of Computing and Digital Systems*. -Scopus (Accepted with Correction)
3. Hossin, M., & Rushdi, A. M. (n.d.). Anchor-Point Based Euclidean Reduction for Enhanced Instance-based Classification. *The 14th International Conference on Information Technology in Asia 2025*. - Scopus (Accepted and Waiting for Presentation and Publication)

**Appendix B:** Details of Results

**Table B.1:** Mean Reduction Time Across 32 Datasets

| Dataset | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|
| Abalone | 100.57 | 1625.18 | **88.32** | 136.08 | 112.63 |
| Appendicitis | **11.81** | 154.02 | 17.19 | 41.52 | 31.94 |
| Australian | 40.2 | 134.48 | **19.18** | 40.31 | 36.84 |
| Balance | **14.4** | 2141.41 | 24.44 | 51.58 | 40.34 |
| Bupa | **8.63** | 1065.55 | 23.42 | 50.03 | 36.08 |
| Coil2000 | 525.12 | 118682.4 | **411.54** | 474.75 | 440.26 |
| Contraceptive | **31.52** | 15488.99 | 43.19 | 66.04 | 60.4 |
| Haberman | **12.04** | 1025.49 | 20.54 | 49.93 | 36.19 |
| Hayes-Roth | **7.64** | 171.45 | 17.19 | 39.44 | 32.45 |
| Heart | **7.98** | 441.38 | 22.69 | 50.06 | 33.74 |
| Housevotes | **10.2** | 141.9 | 20.57 | 48.23 | 34.79 |
| Ionosphere | **11.46** | 139.35 | 16.68 | 52.58 | 35.13 |
| Led7digit | 18.75 | 568.46 | **18.02** | 54.01 | 36.77 |
| Mammographic | **20.75** | 3203.3 | 30.27 | 56.29 | 41.76 |
| Marketing | 415.53 | 95302.26 | **140.85** | 187.17 | 165.75 |
| Monk-2 | **9.71** | 1206.42 | 24.65 | 52.01 | 38.25 |
| Movement_Libras | 48.98 | 298.53 | **33.16** | 51.97 | 46.27 |
| Optdigits | 478.36 | 944.62 | **230.45** | 278 | 255.59 |
| Pima | **20.21** | 4241.49 | 30.24 | 53.56 | 42.23 |
| Satimage | 378.04 | 6623.82 | **188.41** | 236.85 | 213.37 |
| Segment | **59.76** | 1086.81 | 77.96 | 103.73 | 95.82 |
| Texture | 343.67 | 2075.37 | **162.64** | 211.9 | 191 |
| Titanic | **32.81** | 55968.83 | 47.87 | 74.69 | 68.77 |
| Twonorm | **129.66** | 35141.68 | 177.62 | 222.52 | 209.92 |
| Vehicle | **18.27** | 654.85 | 42.14 | 66.96 | 61.08 |
| Vowel | **29.04** | 471.07 | 33.36 | 59.64 | 50.32 |
| Wdbc | **9.07** | 256.94 | 30.18 | 57.48 | 44.76 |
| Wine | **13.39** | 116.86 | 18.23 | 41.98 | 35.33 |
| Winequality-Red | 60.02 | 2306.59 | **46.07** | 68.58 | 59.71 |
| Winequality-White | 404.59 | 6504.06 | **106.95** | 150.15 | 124.99 |
| Yeast | 59.36 | 994.39 | **38.7** | 70.91 | 55 |
| Zoo | 84.72 | 202.2 | **18.5** | 39.49 | 35.42 |
| **AVERAGE** | 106.76 | 11230.63 | **69.41** | 101.20 | 87.59 |

**Table B.2:** Mean Reduction Rate Across 32 Datasets

| Dataset | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|
| Abalone | 9.13 | **83.94** | 21.34 | 21.01 | 19.03 |
| Appendicitis | 21.70 | 57.65 | 75.68 | 75.47 | **75.69** |
| Australian | 57.85 | **61.32** | 25.97 | 33.96 | 29.72 |
| Balance | 9.30 | 65.86 | 76.96 | **80.53** | 74.28 |
| Bupa | 7.85 | 40.23 | **59.96** | 59.10 | 50.72 |
| Coil2000 | 23.95 | **36.87** | 34.86 | 28.64 | 30.74 |
| Contraceptive | 32.72 | **45.93** | 39.11 | 36.06 | 37.11 |
| Haberman | **41.61** | 31.98 | 35.17 | 31.14 | 33.33 |
| Hayes-Roth | 2.72 | 57.11 | **58.47** | 55.59 | 56.06 |
| Heart | 41.42 | **78.06** | 57.53 | 59.48 | 45.49 |
| Housevotes | 18.90 | 30.89 | 54.86 | **62.22** | 54.55 |
| Ionosphere | 47.76 | **71.20** | 37.33 | 36.84 | 31.69 |
| Led7digit | 5.00 | 53.70 | **64.48** | 52.96 | 52.61 |
| Mammographic | 28.16 | **67.79** | 48.32 | 47.64 | 34.90 |
| Marketing | 62.61 | **82.95** | 7.62 | 6.64 | 5.66 |
| Monk-2 | 28.94 | **41.07** | 33.46 | 37.50 | 34.77 |
| Movement_Libras | 21.74 | 31.33 | **51.02** | 43.82 | 36.04 |
| Optdigits | 48.14 | **61.03** | 13.69 | 11.01 | 9.97 |
| Pima | 11.91 | **83.46** | 14.85 | 14.75 | 13.13 |
| Satimage | 66.01 | 34.90 | **72.94** | 72.84 | 72.45 |
| Segment | 77.88 | **98.60** | 12.60 | 12.92 | 11.54 |
| Texture | 71.79 | **91.90** | 31.88 | 29.90 | 30.32 |
| Titanic | 46.21 | 27.07 | **92.07** | **92.07** | 91.72 |
| Twonorm | 8.20 | **82.67** | 24.91 | 29.70 | 26.58 |
| Vehicle | 98.27 | **98.65** | 3.80 | 2.74 | 3.00 |
| Vowel | 72.39 | **96.60** | 31.24 | 29.75 | 30.25 |
| Wdbc | 2.49 | **87.56** | 82.57 | 68.09 | 60.37 |
| Wine | 3.03 | **87.99** | 8.66 | 6.37 | 3.45 |
| Winequality-Red | 14.85 | **78.40** | 43.59 | 35.60 | 31.21 |
| Winequality-White | **83.09** | 53.53 | 6.36 | 5.84 | 4.72 |
| Yeast | 6.20 | 40.73 | **45.00** | 33.40 | 25.01 |
| Zoo | 13.13 | 76.00 | 83.04 | **83.23** | 82.84 |
| **AVERAGE** | 33.90 | **63.66** | 42.17 | 40.53 | 37.47 |

**Table B.3:** Mean Classification Accuracy Across 32 Datasets for k-NN ($k$ = 1)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 20.20 | **23.14** | 8.05 | 19.84 | 20.08 | 20.03 |
| Appendicitis | 72.08 | 77.27 | 79.18 | **81.09** | 74.55 | 76.55 |
| Australian | **78.04** | 51.54 | 49.13 | 76.79 | 76.79 | 77.42 |
| Balance | 73.30 | 71.86 | 59.85 | 66.74 | **73.48** | 72.52 |
| Bupa | 58.00 | 53.35 | 60.28 | **60.52** | 58.50 | 57.34 |
| Coil2000 | **90.05** | 84.37 | 87.47 | 68.94 | 68.75 | 69.88 |
| Contraceptive | **42.97** | 34.49 | 40.93 | 39.18 | 40.19 | 40.13 |
| Haberman | 61.81 | 58.85 | 62.73 | 61.39 | **65.00** | 64.68 |
| Hayes-Roth | 58.36 | 40.52 | 48.75 | **68.13** | 63.13 | 61.88 |
| Heart | 74.81 | 72.59 | **75.93** | 72.59 | 73.70 | 74.07 |
| Housevotes | **92.23** | 90.54 | 87.48 | 90.94 | 91.39 | 91.83 |
| Ionosphere | 84.89 | 80.92 | 85.74 | 86.90 | 86.06 | **87.19** |
| Led7digit | 60.27 | 36.80 | 49.80 | **65.80** | 63.40 | 63.80 |
| Mammographic | 71.69 | **74.58** | 71.08 | 67.11 | 69.76 | 71.81 |
| Marketing | 20.20 | 18.28 | 22.05 | 27.12 | 27.34 | **27.50** |
| Monk-2 | 69.46 | 71.74 | 80.11 | **80.57** | 79.64 | 78.94 |
| Movement_Libras | 66.20 | 60.83 | 66.67 | 67.50 | 71.39 | **75.28** |
| Optdigits | **98.40** | 98.33 | 88.67 | 98.36 | **98.40** | 98.35 |
| Pima | **67.63** | 67.18 | 65.10 | 67.57 | 66.78 | 67.44 |
| Satimage | 21.39 | 21.37 | 82.22 | 86.84 | 87.89 | **88.28** |
| Segment | 96.30 | 92.07 | 70.95 | 96.75 | **97.06** | 96.97 |
| Texture | **99.09** | 98.64 | 89.73 | 87.16 | 92.87 | 93.36 |
| Titanic | 60.10 | 30.43 | **69.15** | 51.57 | 53.70 | 45.71 |
| Twonorm | 94.73 | **95.27** | 91.73 | 94.03 | 94.45 | 94.51 |
| Vehicle | 68.43 | 63.96 | 62.88 | **68.56** | 67.26 | 67.96 |
| Vowel | 65.35 | 53.74 | 49.49 | **65.76** | 65.66 | 64.75 |
| Wdbc | **95.78** | 95.43 | 93.32 | 93.15 | 94.90 | 94.91 |
| Wine | 90.01 | 87.97 | 87.03 | 89.35 | **91.57** | 90.49 |
| Winequality-Red | 48.03 | **52.16** | 31.08 | 46.53 | 46.78 | 46.78 |
| Winequality-White | 40.02 | 40.13 | 24.46 | **44.08** | 43.67 | 43.63 |
| Yeast | 40.89 | 47.19 | 26.69 | 47.91 | **48.85** | 48.25 |
| Zoo | 96.00 | **96.09** | 95.09 | 96.00 | 96.00 | 96.00 |
| **AVERAGE** | 68.02 | 63.80 | 64.46 | 69.84 | **70.28** | 70.26 |

**Table B.4:** Mean Classification Accuracy Across 32 Datasets for k-NN ($k$ = 3)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 21.06 | **22.59** | 12.46 | 21.25 | 21.16 | 21.16 |
| Appendicitis | 73.08 | 77.93 | 78.91 | 79.18 | 83.91 | **84.00** |
| Australian | **65.42** | 45.25 | 34.08 | 63.50 | 62.88 | 64.13 |
| Balance | 72.81 | **73.29** | 61.61 | 70.58 | 71.06 | 72.02 |
| Bupa | **58.60** | 52.18 | 56.16 | 57.62 | 57.67 | 55.57 |
| Coil2000 | **90.44** | 90.26 | 89.56 | 74.14 | 72.58 | 74.14 |
| Contraceptive | **44.87** | 37.41 | 41.82 | 40.46 | 42.16 | 40.67 |
| Haberman | 62.51 | 60.77 | 60.14 | 61.72 | 62.05 | **64.68** |
| Hayes-Roth | **58.56** | 41.21 | 50.63 | 50.63 | 49.38 | 51.88 |
| Heart | 75.26 | 76.67 | 73.70 | 75.93 | **79.26** | 78.52 |
| Housevotes | 91.83 | 89.26 | 84.93 | 90.94 | **92.70** | **92.70** |
| Ionosphere | 84.61 | 86.88 | 86.34 | 85.18 | 86.90 | **88.03** |
| Led7digit | 61.15 | 34.12 | 44.60 | **68.00** | 63.40 | 67.40 |
| Mammographic | 75.66 | **76.87** | **76.87** | 70.36 | 75.18 | 76.27 |
| Marketing | 21.10 | 19.01 | 21.09 | 27.88 | **28.19** | 27.89 |
| Monk-2 | **94.23** | 82.40 | 90.74 | 84.24 | 78.90 | 80.08 |
| Movement_Libras | **66.00** | 56.67 | 39.17 | 56.94 | 55.28 | 64.44 |
| Optdigits | **98.42** | 98.10 | 83.86 | 98.36 | 98.36 | 98.40 |
| Pima | 68.12 | 67.57 | 66.27 | 70.45 | 69.92 | **70.58** |
| Satimage | 21.69 | 21.32 | 82.56 | 87.75 | 88.56 | **88.76** |
| Segment | 95.32 | 91.89 | 44.29 | 95.89 | 95.93 | **96.06** |
| Texture | **98.78** | 98.38 | 74.80 | 84.76 | 87.40 | 89.55 |
| Titanic | 61.20 | 31.10 | **71.74** | 50.57 | 55.76 | 48.02 |
| Twonorm | **96.61** | 96.59 | 94.05 | 95.92 | 96.23 | 96.27 |
| Vehicle | **69.16** | 63.48 | 59.23 | 67.27 | 69.04 | 68.21 |
| Vowel | **62.32** | 50.20 | 27.68 | 61.62 | 60.40 | 61.72 |
| Wdbc | **97.01** | 96.48 | 95.08 | 95.43 | 95.78 | 95.78 |
| Wine | 91.01 | 88.07 | 82.03 | 90.98 | 94.38 | **94.97** |
| Winequality-Red | 48.60 | **51.41** | 28.08 | 45.72 | 47.47 | 46.47 |
| Winequality-White | 40.49 | 41.22 | 18.29 | 43.90 | 44.47 | **44.92** |
| Yeast | 41.30 | 49.83 | 24.53 | 53.03 | **53.57** | 52.96 |
| Zoo | 93.09 | **95.09** | 62.36 | 93.09 | 93.09 | 93.09 |
| **AVERAGE** | 69.54 | 64.86 | 61.43 | 69.97 | 70.62 | **71.17** |

**Table B.5:** Mean Classification Accuracy Across 32 Datasets for k-NN ($k$ = 5)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 22.79 | 22.81 | 14.21 | 22.71 | 22.67 | **22.83** |
| Appendicitis | 74.08 | 78.33 | 79.09 | 86.82 | 83.91 | **88.64** |
| Australian | **54.71** | 44.00 | 25.13 | 51.54 | 50.92 | 53.42 |
| Balance | 74.89 | **78.26** | 66.40 | 74.58 | 74.74 | 75.38 |
| Bupa | **58.40** | 49.24 | 49.80 | 56.51 | 56.80 | 58.22 |
| Coil2000 | 90.84 | **91.81** | 91.27 | 76.50 | 75.03 | 75.77 |
| Contraceptive | **44.95** | 36.12 | 44.88 | 40.12 | 41.28 | 41.48 |
| Haberman | 62.81 | 65.34 | 58.52 | 65.37 | 61.09 | **66.98** |
| Hayes-Roth | 48.75 | 42.22 | **53.75** | 38.75 | 44.38 | 41.88 |
| Heart | 76.26 | 78.89 | 74.07 | 80.37 | **81.48** | 80.00 |
| Housevotes | 91.83 | 90.54 | 86.63 | 91.81 | 91.81 | **93.12** |
| Ionosphere | 83.18 | 82.22 | 82.33 | 85.45 | 86.88 | **87.74** |
| Led7digit | 61.27 | 39.82 | 48.60 | 68.00 | **71.40** | 70.20 |
| Mammographic | **78.92** | 78.55 | 77.35 | 73.01 | 76.87 | 77.83 |
| Marketing | 21.80 | 19.88 | 23.08 | 28.20 | 28.20 | **28.26** |
| Monk-2 | **86.14** | 79.39 | 84.70 | 80.33 | 77.08 | 79.15 |
| Movement_Libras | **65.60** | 50.28 | 36.94 | 51.67 | 48.89 | 57.78 |
| Optdigits | 98.35 | 98.01 | 81.55 | 98.35 | **98.40** | 98.33 |
| Pima | 69.22 | 68.87 | 68.09 | **71.49** | 70.83 | 69.79 |
| Satimage | 21.49 | 21.07 | 81.91 | 87.65 | 88.28 | **88.41** |
| Segment | 94.52 | 90.74 | 29.74 | 95.15 | 95.11 | **95.54** |
| Texture | **98.55** | 98.29 | 63.55 | 83.20 | 85.40 | 88.40 |
| Titanic | 63.79 | 32.17 | **73.74** | 53.93 | 57.98 | 59.61 |
| Twonorm | **96.99** | 96.80 | 94.65 | 96.58 | 96.62 | 96.80 |
| Vehicle | **70.45** | 62.77 | 62.29 | 70.34 | 69.15 | 69.62 |
| Vowel | **61.92** | 48.69 | 18.79 | 56.46 | 57.98 | 57.07 |
| Wdbc | **96.84** | 96.13 | 95.42 | 95.25 | 96.14 | 96.13 |
| Wine | 89.79 | 89.47 | 75.23 | **94.90** | 93.27 | 93.79 |
| Winequality-Red | 51.47 | **52.96** | 40.34 | 51.34 | 50.97 | 52.35 |
| Winequality-White | 41.19 | 42.02 | 25.64 | 48.61 | 47.47 | **49.10** |
| Yeast | 42.10 | **57.74** | 28.90 | 56.53 | 55.86 | 56.06 |
| Zoo | **95.09** | **95.09** | 45.73 | 94.09 | 94.09 | 94.09 |
| **AVERAGE** | 68.40 | 64.95 | 58.82 | 69.55 | 69.72 | **70.74** |

**Table B.6:** Mean Classification Accuracy Across 32 Datasets for k-NN ($k$ = 7)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 23.14 | **23.48** | 14.23 | 23.22 | 22.81 | 23.12 |
| Appendicitis | 74.28 | 75.10 | 84.82 | 86.00 | 84.91 | **87.73** |
| Australian | **55.92** | 47.13 | 25.88 | 50.88 | 52.79 | 53.42 |
| Balance | 81.78 | **82.08** | 70.56 | 80.34 | 79.84 | 79.38 |
| Bupa | 58.90 | 48.37 | 51.85 | 57.41 | 55.92 | **60.28** |
| Coil2000 | 90.94 | **92.73** | 92.51 | 78.26 | 76.52 | 77.39 |
| Contraceptive | 44.95 | 35.30 | **45.29** | 39.65 | 41.61 | 41.21 |
| Haberman | 63.01 | 65.02 | 61.46 | 67.38 | 65.02 | **67.99** |
| Hayes-Roth | 36.25 | 42.01 | **52.50** | 31.88 | 41.25 | 39.38 |
| Heart | 76.56 | 78.15 | 75.93 | **80.74** | 80.00 | **80.74** |
| Housevotes | 92.26 | 90.11 | 88.37 | 92.68 | 92.68 | **93.12** |
| Ionosphere | 81.48 | 81.27 | 75.74 | 85.75 | **86.02** | 85.74 |
| Led7digit | 62.37 | 40.92 | 55.80 | 70.20 | **70.60** | 69.80 |
| Mammographic | **79.64** | **79.64** | 78.07 | 75.18 | 78.19 | 79.28 |
| Marketing | 22.50 | 20.38 | 23.21 | **29.38** | 29.31 | 29.16 |
| Monk-2 | 78.25 | 80.78 | **80.98** | 78.25 | 78.69 | 75.91 |
| Movement_Libras | **64.90** | 47.22 | 29.17 | 48.89 | 42.22 | 55.56 |
| Optdigits | 98.35 | 97.95 | 79.63 | 98.35 | **98.36** | 98.29 |
| Pima | 69.72 | 69.78 | 69.78 | **72.40** | 71.48 | 71.49 |
| Satimage | 21.29 | 20.84 | 81.83 | 87.44 | 88.13 | **88.62** |
| Segment | 93.20 | 90.04 | 27.66 | 94.46 | 94.76 | **94.81** |
| Texture | **98.25** | 97.93 | 56.93 | 81.24 | 83.85 | 87.75 |
| Titanic | 63.00 | 34.15 | **76.01** | 48.85 | 56.12 | 60.92 |
| Twonorm | 97.01 | **97.14** | 94.86 | 96.78 | 96.93 | 97.05 |
| Vehicle | **70.58** | 61.47 | 55.09 | 68.81 | 69.74 | 69.51 |
| Vowel | **56.46** | 47.88 | 14.95 | 52.02 | 51.21 | 52.42 |
| Wdbc | **96.66** | 96.48 | 94.90 | 95.96 | 95.78 | 95.78 |
| Wine | 90.10 | 87.27 | 58.33 | 93.79 | 92.71 | **93.86** |
| Winequality-Red | 52.72 | 51.78 | 41.65 | 52.66 | 52.60 | **54.91** |
| Winequality-White | 42.19 | 42.62 | 27.69 | **49.88** | 49.04 | 49.04 |
| Yeast | 42.67 | 48.69 | 29.98 | 57.27 | **57.81** | 57.47 |
| Zoo | **94.09** | 91.18 | 23.45 | **94.09** | **94.09** | **94.09** |
| **AVERAGE** | 67.92 | 64.53 | 57.47 | 69.38 | 69.72 | **70.79** |

**Table B.7:** Mean Classification Time Across 32 Datasets for k-NN ($k$ = 1)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 22.54 | **17.60** | 18.87 | 25.20 | 21.27 | 24.89 |
| Appendicitis | 0.20 | 3.69 | 0.37 | 1.34 | 1.47 | **0.00** |
| Australian | 5.75 | 4.67 | 1.20 | 6.06 | **0.70** | 14.98 |
| Balance | 3.87 | 7.04 | **0.30** | 4.49 | 5.27 | 1.20 |
| Bupa | 3.44 | **2.92** | 3.17 | 3.43 | 6.37 | 3.83 |
| Coil2000 | 113.62 | 112.51 | 70.00 | 59.36 | **57.41** | 68.21 |
| Contraceptive | 9.71 | 13.89 | 10.34 | **8.35** | 8.61 | 12.58 |
| Haberman | 3.13 | 3.28 | 3.65 | 0.25 | **0.10** | 3.63 |
| Hayes-Roth | 2.30 | 3.53 | **0.20** | 3.46 | 2.06 | **0.20** |
| Heart | 3.52 | 3.38 | **0.70** | 3.13 | 3.24 | 5.10 |
| Housevotes | 4.58 | 5.53 | 8.41 | 6.54 | 5.13 | **2.47** |
| Ionosphere | 6.25 | 6.66 | 2.08 | **0.00** | 5.42 | 6.94 |
| Led7digit | **3.06** | 3.99 | 3.58 | 4.72 | 3.85 | 4.10 |
| Mammographic | 5.83 | 4.29 | 5.18 | 5.01 | **1.50** | 5.36 |
| Marketing | 51.39 | **38.20** | 41.79 | 48.99 | 48.79 | 50.91 |
| Monk-2 | 3.41 | 7.14 | 4.94 | 3.45 | **1.75** | 2.65 |
| Movement_Libras | 6.71 | 5.25 | **4.12** | 8.24 | 7.22 | 6.68 |
| Optdigits | 52.17 | 46.93 | **37.15** | 47.59 | 42.98 | 58.43 |
| Pima | 3.06 | 7.00 | **1.56** | 4.98 | 2.56 | 3.57 |
| Satimage | 57.10 | 48.02 | 43.07 | **37.91** | 45.82 | 52.15 |
| Segment | 19.74 | **15.23** | 15.74 | 15.56 | 20.79 | 20.49 |
| Texture | 48.62 | 41.88 | 31.87 | 35.19 | **28.11** | 36.39 |
| Titanic | **10.65** | 15.06 | 17.10 | 11.88 | 10.94 | 16.23 |
| Twonorm | 68.62 | 56.24 | **50.57** | 52.70 | 51.26 | 78.53 |
| Vehicle | 10.87 | 9.07 | 14.08 | **4.51** | 4.80 | 8.83 |
| Vowel | **3.12** | 7.52 | 3.72 | 6.29 | 7.87 | 7.14 |
| Wdbc | 6.25 | 15.12 | **4.72** | 6.81 | 4.91 | 7.07 |
| Wine | 2.58 | 5.49 | **0.51** | 1.64 | 2.02 | 1.76 |
| Winequality-Red | 9.61 | 13.81 | **6.25** | 10.28 | 12.31 | 11.84 |
| Winequality-White | 42.63 | 31.48 | **22.35** | 40.55 | 38.97 | 33.19 |
| Yeast | 10.80 | 10.11 | **8.27** | 8.98 | 8.98 | 8.32 |
| Zoo | 59.68 | 4.31 | 5.28 | 4.18 | **1.66** | 6.69 |
| **AVERAGE** | 20.46 | 17.84 | **13.79** | 15.03 | 14.50 | 17.64 |

**Table B.8:** Mean Classification Time Across 32 Datasets for k-NN (k = 3)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 27.97 | 21.92 | **20.64** | 26.53 | 26.92 | 24.36 |
| Appendicitis | 2.28 | 3.74 | 3.52 | **0.00** | 0.30 | 2.06 |
| Australian | 11.09 | 3.69 | **1.86** | 6.99 | 8.34 | 2.05 |
| Balance | 6.28 | 3.85 | **0.90** | 1.70 | 7.39 | 5.45 |
| Bupa | 3.19 | 9.27 | **1.84** | 3.35 | 6.48 | 5.09 |
| Coil2000 | 105.34 | 126.84 | 66.14 | **59.92** | 65.71 | 71.75 |
| Contraceptive | 10.65 | 14.77 | **7.54** | 10.26 | 8.25 | 9.04 |
| Haberman | 3.02 | 14.34 | **0.00** | 3.62 | 3.20 | 3.86 |
| Hayes-Roth | 2.73 | 6.17 | 3.53 | 3.16 | **0.20** | 1.67 |
| Heart | **2.68** | 6.96 | 5.51 | 3.84 | 4.83 | 3.55 |
| Housevotes | 5.21 | 8.52 | 5.28 | **1.68** | 3.63 | 2.66 |
| Ionosphere | 5.88 | 7.18 | 10.08 | **0.00** | 5.15 | 4.02 |
| Led7digit | 4.09 | **2.85** | 5.63 | 14.50 | 3.80 | 3.29 |
| Mammographic | 8.31 | 21.76 | 8.91 | 6.50 | **5.28** | 5.79 |
| Marketing | 56.45 | 64.09 | **47.19** | 54.68 | 52.29 | 56.33 |
| Monk-2 | 4.17 | 9.16 | 3.57 | **3.26** | 5.13 | 5.70 |
| Movement_Libras | 5.68 | 5.84 | **3.23** | 5.23 | 10.07 | 4.50 |
| Optdigits | 61.62 | 103.70 | **43.44** | 45.14 | 49.22 | 50.34 |
| Pima | 5.69 | 8.49 | **3.96** | 6.60 | 4.13 | 4.13 |
| Satimage | 63.44 | 109.35 | **42.45** | 47.39 | 44.96 | 52.24 |
| Segment | 26.15 | **14.60** | 16.45 | 16.52 | 19.02 | 22.84 |
| Texture | 53.21 | 64.04 | 38.23 | **32.68** | 32.83 | 41.23 |
| Titanic | 17.18 | 15.73 | 11.99 | 12.51 | **11.83** | 12.19 |
| Twonorm | 64.69 | 98.42 | 56.28 | **47.74** | 56.67 | 65.32 |
| Vehicle | 11.17 | 7.87 | 11.53 | 12.88 | **6.25** | 11.37 |
| Vowel | 6.78 | 9.16 | 4.75 | 16.47 | **2.62** | 7.56 |
| Wdbc | 7.76 | 10.01 | **3.62** | 8.25 | 4.62 | 7.45 |
| Wine | 2.42 | 5.38 | 3.51 | 2.06 | 3.24 | **0.40** |
| Winequality-Red | 17.07 | 9.42 | **7.40** | 16.21 | 8.22 | 12.28 |
| Winequality-White | 69.73 | 64.04 | **25.23** | 39.41 | 43.31 | 42.71 |
| Yeast | 15.05 | **7.69** | 8.46 | 11.57 | 10.69 | 8.92 |
| Zoo | 61.33 | 4.67 | **0.30** | 3.52 | 4.00 | 5.20 |
| **AVERAGE** | 23.39 | 26.98 | **14.78** | 16.38 | 16.20 | 17.35 |

**Table B.9:** Mean Classification Time Across 32 Datasets for k-NN ($k$ = 5)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 28.70 | 23.22 | 21.10 | 26.04 | **21.03** | 28.91 |
| Appendicitis | 1.96 | 3.86 | 0.40 | 1.87 | **0.20** | 2.00 |
| Australian | 3.72 | 12.53 | 5.32 | 5.86 | **0.60** | 1.59 |
| Balance | 6.60 | 6.59 | 3.12 | **2.72** | 5.71 | 3.62 |
| Bupa | **1.50** | 5.10 | 1.56 | 3.56 | 5.06 | 3.53 |
| Coil2000 | 131.17 | 117.48 | 72.93 | 80.09 | **56.83** | 65.59 |
| Contraceptive | 8.47 | 16.96 | 9.81 | 8.55 | 11.31 | **5.20** |
| Haberman | 4.01 | 5.09 | 2.86 | 6.46 | 4.80 | **2.36** |
| Hayes-Roth | 2.08 | 6.81 | 3.13 | 3.56 | **1.56** | 1.83 |
| Heart | 2.56 | 10.42 | 2.71 | **1.78** | 3.32 | 1.86 |
| Housevotes | 3.70 | 8.39 | **2.16** | 3.39 | 3.39 | 5.48 |
| Ionosphere | 5.78 | 6.85 | 1.08 | **0.00** | 5.04 | 7.14 |
| Led7digit | **2.30** | 2.84 | 6.38 | 6.69 | 3.57 | 6.97 |
| Mammographic | 6.25 | 10.37 | 6.60 | 5.22 | 5.48 | **4.69** |
| Marketing | 63.75 | 49.95 | **48.60** | 58.75 | 59.36 | 62.56 |
| Monk-2 | 3.37 | 9.43 | 1.88 | 6.26 | 3.66 | **0.30** |
| Movement_Libras | 7.81 | **4.19** | 6.37 | 6.64 | 6.74 | 7.80 |
| Optdigits | 57.69 | 48.10 | 47.08 | 44.36 | **41.16** | 50.42 |
| Pima | 6.70 | 18.20 | **4.63** | 4.67 | 5.42 | 7.09 |
| Satimage | 65.00 | 56.60 | **40.06** | 52.44 | 46.61 | 56.74 |
| Segment | 21.81 | 23.18 | 16.71 | 22.99 | **14.44** | 22.43 |
| Texture | 48.37 | 51.23 | 33.66 | 34.57 | **33.14** | 44.23 |
| Titanic | 10.48 | 17.38 | 15.27 | 12.94 | **9.90** | 11.18 |
| Twonorm | 73.87 | 69.16 | **50.81** | 52.49 | 55.05 | 55.03 |
| Vehicle | 8.22 | 15.76 | 11.90 | 6.54 | 10.70 | **5.50** |
| Vowel | 7.81 | 12.98 | 7.56 | 8.62 | **6.74** | 8.76 |
| Wdbc | 6.25 | 11.32 | **3.12** | 6.76 | 3.18 | 7.34 |
| Wine | 3.44 | 9.10 | **0.20** | 0.70 | 1.96 | 1.56 |
| Winequality-Red | 13.62 | 10.67 | **8.64** | 13.72 | 12.67 | 10.56 |
| Winequality-White | 56.69 | 36.54 | **24.67** | 49.22 | 46.00 | 46.26 |
| Yeast | 12.01 | 9.15 | 11.38 | **8.46** | 9.67 | 10.79 |
| Zoo | 56.43 | 3.77 | **0.60** | 4.06 | 1.97 | 5.31 |
| **AVERAGE** | 22.88 | 21.66 | **14.76** | 17.19 | 15.51 | 17.33 |

**Table B.10:** Mean Classification Time Across 32 Datasets for k-NN ($k$ = 7)

| Dataset | K-NN | ENN | CNN | SDP-DR (FD) | SDP-DR (DCM) | SDP-DR (MEC) |
|---|---|---|---|---|---|---|
| Abalone | 24.89 | 23.56 | **20.33** | 25.08 | 22.22 | 20.45 |
| Appendicitis | 2.21 | 3.12 | 0.30 | 1.67 | **0.00** | 3.72 |
| Australian | 2.59 | 5.67 | 4.02 | 0.99 | **0.70** | 1.40 |
| Balance | 6.24 | 7.78 | **1.56** | 7.15 | 5.29 | 9.16 |
| Bupa | 2.40 | 8.77 | **1.86** | 3.06 | 3.26 | 3.41 |
| Coil2000 | 127.97 | 130.07 | 67.97 | 60.40 | **59.30** | 68.27 |
| Contraceptive | 9.37 | 13.17 | 9.37 | 10.58 | **3.83** | 6.43 |
| Haberman | 2.47 | 8.43 | **0.08** | 3.72 | 3.24 | 0.89 |
| Hayes-Roth | 0.93 | 6.08 | 3.92 | **0.42** | 3.13 | 2.17 |
| Heart | 2.99 | 10.20 | **0.89** | 3.42 | 3.23 | 5.80 |
| Housevotes | 3.57 | 9.38 | 5.49 | 5.00 | 4.91 | **1.96** |
| Ionosphere | 6.25 | 9.88 | 4.69 | 15.14 | **3.48** | 4.02 |
| Led7digit | 4.69 | 5.40 | 4.75 | 8.40 | **2.06** | 3.11 |
| Mammographic | **5.62** | 9.96 | 7.69 | 6.62 | 6.78 | 10.81 |
| Marketing | 64.41 | **40.81** | 56.24 | 61.94 | 59.47 | 62.77 |
| Monk-2 | 2.16 | 7.05 | 4.84 | 5.37 | **0.74** | 6.75 |
| Movement_Libras | 3.13 | 5.44 | **1.40** | 6.69 | 6.25 | 8.49 |
| Optdigits | 61.60 | 51.38 | **38.79** | 48.45 | 54.94 | 44.43 |
| Pima | 7.00 | 5.40 | 6.30 | 4.70 | 10.26 | **4.40** |
| Satimage | 58.35 | 55.26 | **42.61** | 45.04 | 50.16 | 51.10 |
| Segment | 22.68 | 16.60 | 20.31 | 21.20 | **16.29** | 19.22 |
| Texture | 46.92 | 46.86 | 39.05 | **33.08** | 40.61 | 41.59 |
| Titanic | 10.62 | 12.90 | 12.50 | 13.72 | 12.56 | **7.90** |
| Twonorm | 71.58 | 57.35 | 54.67 | 57.35 | 56.40 | **52.56** |
| Vehicle | **6.25** | 10.42 | 6.28 | 11.92 | 10.28 | 11.19 |
| Vowel | 7.76 | 12.85 | 6.36 | 8.15 | 6.24 | **4.33** |
| Wdbc | 9.62 | 8.31 | 5.83 | 5.15 | 9.11 | **4.33** |
| Wine | 3.12 | 8.16 | 3.32 | **2.02** | 3.12 | 2.66 |
| Winequality-Red | 11.34 | 15.42 | 10.94 | 12.54 | **9.37** | 13.64 |
| Winequality-White | 54.54 | 31.50 | **24.99** | 49.34 | 48.07 | 50.03 |
| Yeast | 16.55 | 8.94 | 7.80 | **7.49** | 9.06 | 10.46 |
| Zoo | 59.64 | **3.31** | 5.25 | 4.12 | 6.95 | 5.52 |
| **AVERAGE** | 22.48 | 20.29 | **15.01** | 17.19 | 16.60 | 16.97 |