RANDOMIZED AND EVOLUTIONARY APPROACHES TO DATASET CHARACTERIZATION, FEATURE WEIGHTING, AND SAMPLING IN K-NEAREST NEIGHBORS

by

SURYODAY BASAK

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
May 2020

To my Father and Mother,

who have always been the wind beneath my wings;

whose benevolence and kindness has always been the greatest enabler in my life.

ACKNOWLEDGEMENTS

I would like to thank Dr. Manfred Huber, my thesis advisor, for his invaluable guidance and support through my master thesis. I would also like to thank thank Dr. Farhad Kamangar and Dr. Won Hwa Kim, for accepting to be on my thesis committee.

I would sincerely like to thank all my research mentors whose support and advice is the reason I have been able to come this far: Dr. Snehanshu Saha, Dr. Amir Farbin, Dr. Ryan Curtin, Dr. Marcus Edel, Dr. Jayant Murthy, Dr. Saibal Kar, Dr. Margarita Safonova, Mrs. Archana Mathur, Mrs. Surbhi Agrawal and Mrs. Kakoli Bora. I would like to thank my friends in India, Simran, Madhav, Jeremy, Abinav, Sandesh, Aditi, Sampath, Aly, and Rakshit, and my friends and advisors in UTA, Sai Manoj, Sergio, Sanath, Aunik, Saket, Harshit, Andrew and Kathryn for always looking out for me and motivating me throughout my master's program.

I am grateful to my parents and sister for all their unconditional love and support. For me to earn a master's degree was as much of their dream as it was mine.

May 22, 2020

ABSTRACT

RANDOMIZED AND EVOLUTIONARY APPROACHES TO DATASET CHARACTERIZATION, FEATURE WEIGHTING, AND SAMPLING IN K-NEAREST NEIGHBORS

SURYODAY BASAK, M.S.

The University of Texas at Arlington, 2020

Supervising Professor: Manfred Huber

K-Nearest Neighbors (KNN) has remained one of the most popular methods for supervised machine learning tasks. However, its performance often depends on the characteristics of the dataset and on appropriate feature scaling. In this thesis, characteristics of a dataset that make it suitable for being used within KNN are explored. As part of this, two new measures for dataset dispersion, called mean neighborhood target variance (MNTV), and mean neighborhood target entropy (MNTE) are developed to help determine the performance we expect while using KNN regressors and classifiers, respectively. It is empirically demonstrated that these measures of dispersion can be indicative of the performance of KNN regression and classification. This idea is extended to learn feature weights that help improve the accuracy of KNN classification and regression. For this, it is argued that the MNTV and MNTE, when used to learn feature weights, cannot be optimized using traditional gradient-based optimization methods and we develop optimization strategies based on metaheuristic methods, namely genetic algorithms and particle swarm optimization. The feature-weighting method is tried in both regression and classification contexts on publicly available datasets, and the performance is compared to KNN without feature weighting. The results indicate that the performance of KNN with appropriate feature weighting leads to better performance. In a separate branch of the work, the ideas of MNTV and MNTE are used to develop a sample-weighting algorithm that assigns sampling probabilities to each instance in a training set. This too is tried in both regression and classification with subsamples drawn using the sampling probabilities, and the performance is compared to KNN without subsampling the training set.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT v
LIST OF ILLUSTRATIONS x
LIST OF TABLES xii

Chapter Page

1. INTRODUCTION 1
   - 1.1 A Novel Approach for Analyzing Data for Instance-Based Learners . 3
   - 1.2 Metaheuristic Optimization Approaches for Feature Scaling for Instance-Based Learners 3
   - 1.3 A Novel Randomized Approach to Data Sampling for Instance-Based Learners 4
2. BACKGROUND 5
   - 2.1 The Different Types of Machine Learning Tasks 5
   - 2.2 Representation Categories in Supervised Learning 6
   - 2.3 The K-Nearest Neighbors Approach 7
     - 2.3.1 Algorithm 7
     - 2.3.2 The Choice of k and the Bias-Variance Tradeoff in KNN . . . 8
     - 2.3.3 Advantages and Disadvantages of KNN 10
   - 2.4 Characteristics of Data 11
   - 2.5 Characteristics of Data Relevant to KNN 13
3. LOCALIZED PROPERTIES IN DATASETS 14
   - 3.1 Definitions 14

3.2 Mean Neighborhood Target Standard Deviation 15
3.3 Mean Neighborhood Target Entropy 16
3.4 Computing Localized Dataset Characteristics 16
3.5 MNTSD with Inverse-Distance Weighting of Samples 20
3.5.1 Average Leave-One-Out Variance 21
3.6 Empirical Results on Comparing the Performance of a Learner with Localized Dispersion Measures 22
3.6.1 Datasets 23
3.6.2 MNTSD for Regression 25
3.6.3 MNTSD and MNTE for Classification 31
4. OPTIMAL FEATURE SCALING 37
4.1 Selecting the Right Optimization Approach 38
4.2 Related Work on Feature Scaling for KNN 39
4.3 A Motivating Example 42
4.4 Methods for Feature Scaling Optimization 44
4.4.1 Genetic Algorithm (GA) 44
4.4.2 Particle Swarm Optimization (PSO) 46
4.5 Comparison with Principal Component Analysis 49
4.6 Results 50
4.6.1 Classification 51
4.6.2 Regression 52
5. DATA SUBSAMPLING 53
5.1 Related Work 53
5.2 Method 54
5.3 Results 55
6. CONCLUSION AND FUTURE WORK 59

6.1 Dataset Characterization 59
6.2 Feature Scaling and Feature Extraction 60
6.3 Data Subsampling 61
6.4 Source Code 62

Appendix

A. PROOF OF THEOREM 1: THE AVERAGE UNBIASED LEAVE-ONE-OUT VARIANCE IS THE SAME AS THE UNBIASED VARIANCE OF A SET 63
B. RESULTS OF OPTIMAL FEATURE SCALING 71
REFERENCES 90
BIOGRAPHICAL STATEMENT 100

LIST OF ILLUSTRATIONS

Figure Page

2.1 Effect of the decision boundary based on k 9

3.1 Example: Variation in decision surface while leaving one instance out at a time 21

3.2 Boxplots for the Airfoil dataset 27

3.3 Boxplots for the Auto MPG dataset 27

3.4 Boxplots for the Concrete dataset 28

3.5 Boxplots for the Energy (cool) dataset 28

3.6 Boxplots for the Energy (heat) dataset 29

3.7 Boxplots for the QSAR aqua toxicity dataset 29

3.8 Boxplots for the QSAR fish toxicity dataset 30

3.9 Boxplots for the Wine quality dataset 30

3.10 Boxplots for the Yacht hydrodynamics dataset 31

3.11 Boxplots for the Blood Transfusion dataset 32

3.12 Boxplots for the Breast Cancer dataset 33

3.13 Boxplots for the Cardioctography dataset 33

3.14 Boxplots for the C-Section dataset 34

3.15 Boxplots for the Diabetic Retinopathy dataset 34

3.16 Boxplots for the Haberman dataset 35

3.17 Boxplots for the ILPD dataset 35

3.18 Boxplots for the Immunotherapy dataset 36

3.19 Boxplots for the Wine dataset 36

4.1 Visualization of the effect of appropriate feature weights. 43

4.2 Working of GA on Feature Weights 46

LIST OF TABLES

Table Page

4.1 Results of classification. Metaheuristic optimization done here was based on MNTSD or MNTE. 51

4.2 Results of regression. All metaheuristic optimization done here was based on MNTSD. 52

5.1 MSE after sampling, Airfoil 56

5.2 MSE after sampling, Concrete 57

5.3 MSE after sampling, Energy (cool) 57

5.4 MSE after sampling, Energy (heat) 57

5.5 MSE after sampling, Auto MPG 58

B.1 Results of KNN Regression with all variants studied, on the Airfoil dataset. 73

B.2 Results of KNN Regression with all variants studied, on the Concrete Compressive Strength dataset. 73

B.3 Results of KNN Regression with all variants studied, on the Energy (cool) dataset. 74

B.4 Results of KNN Regression with all variants studied, on the Energy (heat) dataset. 75

B.5 Results of KNN Regression with all variants studied, on the Qsar Aqua toxicity dataset. 76

B.6 Results of KNN Regression with all variants studied, on the Qsar Fish toxicity dataset. 77

B.7 Results of KNN Regression with all variants studied, on the Wine (red). 77

B.8 Results of KNN Regression with all variants studied, on the Yacht. . . 78
B.9 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Blood Transfusion dataset. 81
B.10 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Blood Transfusion dataset. 81
B.11 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Breast Cancer dataset. . . . 82
B.12 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Breast Cancer dataset. 82
B.13 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Cardioctography dataset. . 83
B.14 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Cardioctography dataset. Multiple cases result in very low variance of accuracy. 83
B.15 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the C-Section dataset. 84
B.16 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the C-Section dataset. . . 84
B.17 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Diabetic Retinopathy dataset. 85
B.18 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Diabetic Retinopathy dataset. 85
B.19 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Haberman dataset. 86

B.20 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Haberman dataset. . . 86

B.21 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the ILPD dataset. 87

B.22 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the ILPD dataset. 87

B.23 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Immunotherapy dataset. . . 88

B.24 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Immunotherapy dataset. 88

B.25 Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Wine classification dataset. 89

B.26 Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Wine classification dataset. 89

CHAPTER 1

INTRODUCTION

In the past few decades, the amount of data that is gathered and organized in any domain has vastly increased [1]. Increasingly, people and organizations are interested in better quantitative assessments and predictions. This has given rise to opportunities to develop and deploy computational methods that use these massive amounts of data. Through a merging of cutting edge computational capabilities along with sophisticated methods of statistical analysis, the field of *machine learning* (ML) has become one of the fastest growing areas that is studied within the realm of computer science. The holy grail of machine learning is to be able to emulate human intelligence, in order to make accurate estimates and decisions automatically. In order to make any of this work within computers, statistical and mathematical models are used.

Part of the reason why ML has become immensely popular in the recent decades is because of strong applications in various areas, such as business, medicine, or anything that enriches the human experience. Today, ML is used within applications of fitness tracking [2, 3] , medical diagnosis [4, 5, 6], high-energy physics [7, 8], astronomy [9, 10], business [11, 12], graphics and rendering [13], human-computer interaction [14, 15], and almost anything where data can be collected. Increasingly, organizations and experts in all these domains are looking at results of advanced analytics that can provide data-driven inferences about such problems. Data-driven inferences are generally more informed, and can be explained with appropriate visualizations.

Naturally, the latest trends in ML have been to develop really sophisticated methods [16]. However, there still remains a plethora of tasks wherein simpler methods with clear explanations suffice, and where complex learning algorithms are an overkill [17]. With this in mind, questions arise about how suitable a certain dataset might be for a particular approach. From classical statistical analysis, popular measures of data dispersion such as histograms, distribution estimates, etc. are often used to develop a working intuition about a dataset that is being dealt with. However, such analyses fall short for ML approaches that are inherently unassuming.

K-nearest neighbors (KNN) is one of the simplest approaches that exist in machine learning. As an approch, the KNN model itself has very low bias [18], and does not assume or require that the data it is used on follow any particular distribution or exhibit any particular property. This simplicity is attractive for various computational and inferential reasons, but it comes at a cost. KNN is a non-parametric instance-based learner, which means that the predictive capability of KNN is dependent on individual samples in a dataset and not any other parameters learned from the data; KNN also does not require any sort of functional relationship to exist between the independent and dependent variables in a dataset. The lack of a functional relationship in KNN makes tasks like feature extraction or weighting inherently difficult (this is elaborated in Chapter 2). However, in order for it to perform well, it is important to address a number of shortcomings, including its interactions with characteristics and biases in the dataset, its dependence on feature magnitudes, and complexity and size of the dataset it is used in.

Traditionally, tasks such as feature extraction is done independently based on component analyses or data transformations. However, the independence of this step does not always guarantee an improvement in performance, and the absence of a functional relationship is one of the reasons for this. Thus, using KNN effectively has required careful analysis of the data being used, along with trying different approaches for understanding the best signals inherent to the data. This thesis investigates techniques to address these aspects by introducing data set characteristics, developing new approaches to feature weighting, and by investigating subsampling techniques.

1.1 A Novel Approach for Analyzing Data for Instance-Based Learners

Since instance based learners like KNN do not have any strict requirements related to distributions, separability, etc. in a dataset, an appropriate analysis of how well-suited a dataset might be for an approach would involve an analysis of the instances or sets of instances within a dataset that can lead to effective predictions. For example, within KNN, all combinations of k training points may not be a neighborhood that could be queried for any particular point in the domain of the data. For those sets that can be a nearest-neighborhood set, an analysis of the dispersion of their independent variables can give us an estimate of the error that we may expect to encounter.

In this thesis, two new measures of dataset dispersion, called *mean neighborhood target standard deviation* (MNTSD) and *mean neighborhood target entropy* (MNTE) are introduced. Their formulations are elaborated in Chapter 3 and their usefulness is empirically demonstrated by applying them to multiple freely-available datasets, and comparing these measures to actual performance metrics.

1.2 Metaheuristic Optimization Approaches for Feature Scaling for Instance-Based Learners

Based on the MNTSD and MNSE, feature weights are optimized using metaheuristic optimization methods. Feature weights that minimize the the MNTSD or

MNTE for a dataset are tested empirically for accuracy of the method. This is elaborated in Chapter 4. The results demonstrate that there is generally an improvement in accuracy when features in a dataset are scaled based on minimizing the MNTSD or MNTE.

1.3 A Novel Randomized Approach to Data Sampling for Instance-Based Learners

A randomized sampling approach is developed. Sample weights are optimized using randomized sampling and reweighing based on the MNTSD and MNSE. Sample sets that minimize the the MNTSD or MNTE for a dataset are tested empirically for accuracy of the method. This is elaborated in Chapter 6. The results demonstrate that there is generally an improvement in accuracy when sample weights are assigned based on MNTSD or MNTE, as compared to uniform sampling.

CHAPTER 2

BACKGROUND

2.1 The Different Types of Machine Learning Tasks

Within the purview of ML, different kinds of methods and approaches exist depending on the type of data that is available and the task that it is used for. The following is a list of the most popular machine learning tasks:

Supervised Learning: Supervised learning broadly refers to ML tasks wherein a mapping between two sets of variables is estimated. In supervised learning, the algorithm has access to an input set X, and a target set Y. The task is to learn a function $f : X \mapsto Y$ that has the lowest error. The most common tasks in supervised learning are those of classification and regression [19].

Unsupervised Learning: In unsupervised learning, the algorithm has access to an input set X, but there is usually no access to target set Y. Unsupervised learning methods learn patterns in the input set alone. Clustering is the most popular unsupervised learning task [20], followed by methods in component analysis.

Semi-Supervised Learning: Semi-supervised learning is, in essence, a combination of supervised and unsupervised learning, wherein a subset of the input set X has an associated set of target values, and the rest of the set does not. Semi-supervised learning is often used to prevent overfitting in ML models by allowing an often significantly larger amount of unlabeled data to be utilized during learning, making the resulting model more robust based on properties of unlabelled data [21], and in generative models [22] that can generate samples that are realistic.

Reinforcement Learning: Reinforcement learning (RL) deals with another set of tasks wherein the input is interpreted as a state in a Markov decision process. Very commonly, the input domain consists of states and the outputs consist of actions, that are not directly available. The goal of RL is to learn an action for each state such that some form of a performance measure is optimized. The difference to unsupervised learning is that the performance estimate comes with the data rather than being built into the algorithm.

2.2 Representation Categories in Supervised Learning

This thesis deals exclusively with supervised learning tasks. There are broadly two categories of representations in supervised learning approaches distinguished by the way the model is represented:

Parametric Approaches: Parametric learning algorithms employ a transformation of the input based on an assumed functional form, whose parameters are optimized [23]. Some popular examples of parametric approaches are linear regression, logistic regression, neural networks, etc.

Non-Parametric Approaches: Non-parametric approaches do not assume an explicit functional form, rather they use the aspects of the data itself to represent the structure of the learned model. As a result, there are usually no model parameters to be optimized [23]. The aspects that are learned in non-parametric approaches vary based on the particulars of the algorithm. K-nearest neighbors is one of the most popular non-parametric algorithms that exist and forms the focal point of this thesis.

2.3 The K-Nearest Neighbors Approach

K-nearest neighbors is a canonical learning method in machine learning and yet, with reasonable tuning of model parameters, can easily perform well in non-linear classification and regression. It is a form of *lazy learning* [24], which implies that a generalization of the properties predicted for a test instance is not made unless it is queried. Learning in KNN is based on a defined similarity metric which is used to select k training instances that are most similar to the test point, where k is a number that is set by the user. The target values of these k points are aggregated to make a prediction for the test point. For classification, KNN usually uses a majority-voting scheme, where the most occurring value is declared as the predicted value. In the case of regression, the k target values are usually averaged. Various sample-weighting schemes may also be used to consider a weighted-average, such as an inverse-distance weighting, where more similar samples are given a larger weight, and samples that are less similar to the test query are given a lower weight, based on the reciprocal of the similarity metric.

2.3.1 Algorithm

A simple implementation of the KNN algorithm using a for loop is described in Algorithm 1. In this description of the algorithm, the procedure *similarity_metric* refers to how the similarity between the test point and training samples is ascertained – popular similarity metrics are L1 and L2 norms, or a weighted L2 norm. However, KNN does not require that any particular metrics of similarity be used. The choice of the metric should depend on the application [25]; the procedure *aggregate* refers to how the target values of the k-most similar training instances are used to make a prediction. This may be majority voting in the case of classification, and average or weighted average for regression.

Algorithm 1 K Nearest Neighbors

```
1: input: $X_{\text{train}}$ is the training set
   $y_{\text{train}}$ is the set of associated target values of $X$
   $X_{\text{test}}$ is the test sample
   $k$ is the number of training samples used to make a prediction

2: output: $y_{\text{pred}}$ is the predicted target value of $X_{\text{test}}$

3: procedure KNN($X_{\text{train}}, y_{\text{train}}, X_{\text{test}}, k$)
4:     $n \leftarrow X_{\text{train}}$.length()
5:     initialize similarity as an array of length $n$
6:     for $i \leftarrow 0$, $i < n$, $i++$ do
7:         $similarity[i] \leftarrow$ similarity_metric($X_{\text{train}}, X_{\text{test}}$)
8:     $k_{\text{top}} \leftarrow$ argsort($similarity$)[0:$k$]
9:     $y_{\text{top}} \leftarrow y_{\text{train}}[k_{\text{top}}]$
10:    return aggregate($y_{\text{top}}$)
```

Although the complexity of Algorithm 1 is $O(nm+k)$, where m is the number of attributes or features in the data, in practice, much faster implementations of KNN exist that use datastructures such as K-D trees [26], and locality-sensitive hashing [27, 28] . This thesis, however, does not focus on speeding up KNN.

2.3.2 The Choice of k and the Bias-Variance Tradeoff in KNN

The bias-variance (BV) tradeoff of any predictive model is a property that indicates the tradeoff in errors that arise due to assumptions inherent to the model and errors that arise due to variability in a dataset [18]. In KNN, the sources of bias are in the selected k and the chosen similarity measure; the source of variance is the variability of individual points that constitute a k-nearest neighborhood of a test point.

KNN does not rely on any stringent assumptions about the data. Thus, the bias in the model is low, but the model suffers from high variance, that is related to the k selected points, and that changes from region to region within the feature space [18].

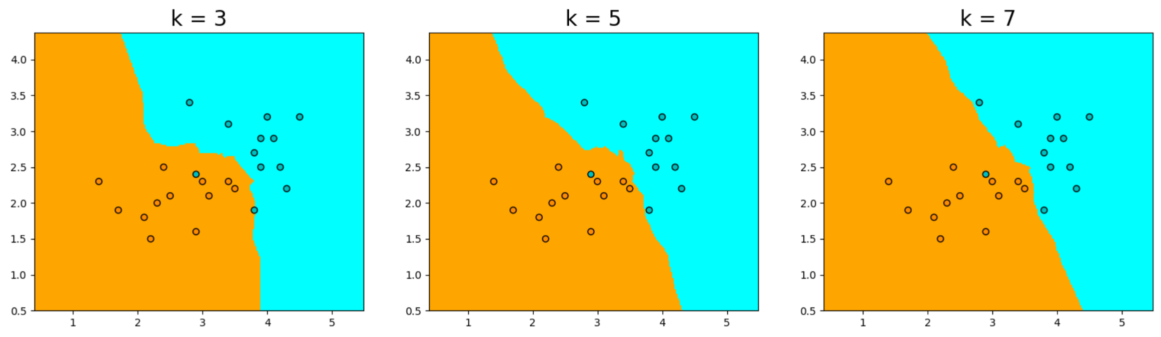

Figure 2.1. Effect of the decision boundary based on k.
Each of the above decision surfaces were created using the same dataset. However, the decision surface for each value of k is very different. Thus, selecting the right value of k greatly affects the predictive capabilities of KNN.

In Figure 2.1, all the plots are created based on the same two-dimensional dataset. This dataset, indicated by the points in the graphs, has two classes in it, with the colours of the points indicating the class each point belongs to. The color of the background indicates the class that a corresponding data point would belong to. The measure of similarity used here was the Euclidean distance, and the sample weighting was uniform. For different values of k, the decision surfaces turn out to be very different. Determining the best value of k to ensure an acceptable level of variance is thus a challenging task in KNN.

The interpretation of this in terms of the BV-tradeoff is that if the value of k is low, variability in predictions across regions in the feature space is expected to be high and the bias to the dataset is low, whereas if k is high, then the model generalizes more to the training data, with smaller fluctuations across regions, resulting in a high bias and low variance.

An extreme case of the BV-tradeoff is if $k = n$, where n is the size of the dataset. Assuming that the unweighted Euclidean distance is used as the measure of similarity, in the case of classification, any test point would be classified as the majority class in the dataset, and in the case of regression, any test point would have the prediction of the average of all the target values of the set: the variance in prediction is zero, whereas the method is extremely biased to the dataset. The other extreme case is that of 1-nearest neighbor, where the prediction of any test point depends on only one point in the dataset, resulting in high variance in predictability across the entire input domain, but low bias to the training set.

2.3.3 Advantages and Disadvantages of KNN

Despite an explosion in the amount of data that led to the development of new and more complex methods in machine learning, a lot of specialized applications still collect datasets that are small in size. For a lot of smaller datasets, powerful methods such as neural networks are known to overfit [29]. Due to this reason, classical methods in ML such as KNN are significant to this day.

KNN is a simple approach, it is generally a lot more transparent as a method in a practical setting: for example, imagine that KNN is employed in diagnosis of cancer. Data is collected and is classified into two classes: malignant and benign. If KNN is used, it gives the analyst an opportunity to take a look at the instances within the dataset that led to the prediction of a test sample. Since a misdiagnosis of a malignant case could potentially be a life-hazard, the use of KNN, in addition to providing a classification of a test sample, could point a doctor or an analyst towards similar cases as the test case. Based on all this knowledge an informed decision can be made about the diagnosis.

On the other hand, the lack of an explicit functional relationship between the input and target variables can sometimes be a disadvantage, especially in tasks such as feature extraction, etc., compared to parameterized models such as logistic regression where weights for each feature must be optimized [30]. For tasks such as feature extraction, the most popular approaches involve a cascading of methods such as principal component analysis (PCA), or independent component analysis (ICA) prior to using KNN [31, 32].

2.4 Characteristics of Data

Some characteristics of data that are generally analyzed prior to using ML algorithms are as follows:

1. **Size of Dataset**: The size of a dataset is one of the first attributes that is generally looked at. As a rule of thumb, the larger the number of representative and unbiased instances in a dataset, the better [33]. Often, the choice of a learner depends on the size of data: for instance, modern deep learning generally requires a large number of samples in order to not overfit on the training data [34].
2. **Dimensionality**: The number of dimensions in the data is indicative of the complexity of the learning task. The *curse of dimensionality* is a problem pertinent in all kinds of machine learning. Generally, localized methods such as KNN become less accurate with the increase in the number of dimensions: in high dimensions, all training instances sparseley populate the input domain [18].
3. **Number of Classes**: Often, like the dimensionality in the dataset, the number of classes is indicative of the complexity of the learning task at hand. With a larger number of classes, localized methods suffer. For example, a scenario

containing 100 classes would require a very high number of localized evaluations by localized methods.

4. **Correlation of Features**: Generally, decorrelated features are preferred in ML tasks as correlated features do not improve the quality of the information that is available in the dataset.

5. **Distribution of Features**: Certain methods implicitly assume that the data follows a certain distribution and hence, a confidence value of these assumptions can be representative of how well the data conforms to the assumptions of the method. For example, Gaussian Naïve Bayes (GNB) assumes that the features in the data are decorrelated and normally distributed [35], and the whole method is built on the basis of this assumption.

6. **Clusters**: Cluster analysis of a dataset can often tell us how different regions in the input domain vary in properties [36]. When this information is used along with information about the classes, we can get an idea about the distribution of classes in the input domain.

7. **Separability and Overlap of Classes**: Almost any trained ML model may be interpreted as a decision surface. In classification, the geometry of the decision surface is a result of the model parameters and their optimization. Some methods, however, require that classes be separable, either linearly or upon some transformation [37]. Support vector machines (SVM) work on the basis of this model assumption. In the vanilla version of SVM the support vectors cannot be discovered unless classes in the data are linearly separable; this requirement is relaxed in soft-margin SVM wherein an overlap between classes is compensated by introducing an error term [38]. In other popular variants, such as kernel SVM, the data is implicitly transformed into higher dimensions where the data may be linearly separable [39].

2.5 Characteristics of Data Relevant to KNN

Since KNN is a highly localized method, localized characteristics of a dataset would be relevant to gain an insight of how suitable a dataset might be to be used with KNN. This is traditionally a little elusive since KNN does not assume any particular distribution in the data, or use any explicit functional form between the input and target domains. Also, in practice, all sets of k-points in the input domain may not constitute a k-nearest neighborhood.

An appropriate empirical analysis would thus require an analysis of sets of points that can be reached as k-nearest neighborhoods to points in the input domain. Since KNN has low bias, within sets of points that constitute k-nearest neighborhoods, variance of the target values could provide us an estimate of the variance that we may expect for each neighborhood that exists. An aggregation of the localized properties for the whole dataset could tell, on average, how well-suited the whole dataset is for KNN.

In the next chapter, the rationale of these ideas are elaborated along with an implementation of the concepts and a coarse evaluation with results. These characteristics will then be used to introduce novel feature scaling and data set sampling approaches in subsequent chapters.

CHAPTER 3

LOCALIZED PROPERTIES IN DATASETS

As discussed in the previous chapter, characteristics of the dataset can be important in terms of the performance of KNN approaches. Due to its local behavior characteristics, properties important for KNN should generally capture localized characteristics of the data set. In this chapter, two measures of localized variance are introduced. These measures are aimed at capturing important aspects of the data in a local neighborhood and will be used in subsequent chapters to form techniques for optimizing KNN techniques.

3.1 Definitions

1. *k-Nearest Neighborhood Set*: A k-nearest neighborhood set is a set of k points in the training set that represents the k-nearest neighbors to at least one point in the input domain, for a fixed k.
2. *Target Variance*: The variance of the target values of a dataset.
3. *Target Standard Deviation*: The standard deviation of the target values of a dataset.
4. *Target Entropy*: The entropy [40] of the target values of a dataset.

Note that not all subsets of the points in the training set may constitute a k-nearest neighborhood set. In most practical problems only a small fraction of k-combinations of training instances constitute a k-nearest neighborhood, with some of the factors determining the fraction being the dimensionality of the underlying feature space compared to the size of the dataset as well as the density of the data points.

Thus, an appropriate analysis of a dataset to decide if KNN is a suitable supervised approach should take into consideration possible k-nearest neighborhood sets, or at most, points that are very close to each other. In addition to that, the accuracy of a KNN method entirely depends on the distribution of the target-values of k-nearest neighborhood sets: if the variance of the target values of a k-nearest neighborhood set is high, then it could indicate that the performance of the method could be prone to errors. In the case of classification, additionally, measures of dispersion appropriate for categorical values could be utilized. In the current work, we particularly use entropy, similar to the measure used to split nodes in decision trees [40].

3.2 Mean Neighborhood Target Standard Deviation

We define the mean neighborhood target standard deviation (MNTSD) as:

$$\frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(y_{n,i}-\mu_n)^2} \tag{3.1}$$

where N is the number of distinct sets of k points that form k-nearest neighborhood sets, k is the parameter of KNN, $y_{n,i}$ is the target value of the i^{th} instance in neighborhood n, μ_n is the mean of all the target values in neighborhood n.

This measures the average standard deviation of target values across all neighborhoods in a dataset. This measure of variation in target values is particularly useful when target values are continuous or ordinal. It was formulated with KNN regression in mind, since target values in regression are usually continuous valued. However, this can also be used within KNN classification if class labels are converted to numerical values.

3.3 Mean Neighborhood Target Entropy

We define the mean neighborhood target entropy (MNTE) as:

$$M = -\frac{1}{N}\sum_{n=1}^{N}\frac{1}{c}\sum_{i=1}^{c} p_i \log(p_i) \tag{3.2}$$

where N is the number of distinct sets of k points that form k-nearest neighborhood sets, c is the number of classes that are present in neighborhood n, p_i is the probability of encountering an instance of class i in neighborhood n.

This measures the average Shannon entropy of target values across all neighborhoods in a dataset. This measure of variation in target values is particularly useful when target values are discrete with no ordering in values. It was formulated with KNN classification in mind, since target values in classification are usually discrete and not ordinal. Since entropy does not care for ordering of values, this should not be used in the context of KNN regression.

3.4 Computing Localized Dataset Characteristics

Some geometrical aspects must be taken into consideration prior to developing a method to compute the MNTSD or MNTE of a dataset.

Lemma 1: *Given a training population of n samples in m dimensions and a fixed k, the number of unique sets of points that can constitute neighborhoods in KNN is upper bounded by by the number of combinations of k elements from a set of n elements, ${}^{n}C_k = \frac{n!}{(n-k)!k!}$.*

Proof: Given a set of n training instances, the number of unique sets of k points that can be formed is ${}^{n}C_k$. Condsider a training set that can be perfectly arranged on the surface of a hypersphere in m dimensions. From the center of this hypersphere, every point in the training set is equally far away, with the distance to each point corresponding to the radius of the hypersphere. Thus, at this point, any set of k training instances can be considered to be a neighborhood. The number of neighborhoods reachable from the center is thus ${}^nC_k$. Hence, the number of unique neighborhoods for any training set is upper bounded by ${}^nC_k$.

Lemma 2: *A set of k points A_i in a training sample T can form a k-nearest neighborhood set iff a convex hull can be constructed around A_i such that $\forall x \in T$ and $x \notin A_i$, x does not lie within the convex hull of A_i.*

Proof: Let there be some point $x_m \in T$ and $x_m \notin A_i$ that lies within the convex hull of the set A_i. Then, there exists at least one point $x_j \in A_i$, and some point x_t in the space of T such that the distance between x_t and x_j is greater than the distance between x_t and x_m, that is

$$D(x_t, x_j) > D(x_t, x_m) \tag{3.3}$$

For this point, the set $A_i - \{x_j\} + \{x_m\}$ is the k-nearest neighborhood. This contradicts that A_i is a k-nearest neighborhood set. Thus, this result is proven by contradiction.

Lemma 3:

$${}^nC_k \leq n^k \tag{3.4}$$

Proof:

$$
\begin{aligned}
{}^{n}C_k &= \frac{n!}{(n-k)!k!} \\
&= \frac{n \times (n-1) \times ... \times (n-k+1) \times (n-k)!}{(n-k)!k!} \\
&= \frac{n \times (n-1) \times ... \times (n-k+1)}{k!} \quad \leftarrow k \text{ terms} \\
&\leq \frac{n \times n \times ... \times n}{k!} \quad \leftarrow k \text{ terms} \\
&= \frac{n^k}{k!} \\
&\leq n^k
\end{aligned}
\tag{3.5}
$$

Based on the above lemmata, we see that in order to calculate the mean neighborhood target variance, first each neighborhood set in the dataset would have to be discovered. The most exhaustive way of doing this would be to find each of the ${}^{n}C_k$ unique sets of k instances (lemma 1), determine if they are a neighborhood set or not (based on lemma 2), and then find the target-variance if a k-set is a neighborhood. The cost of finding each k-set is upper-bounded by $O(n^k)$ (based on lemma 3). In order to determine if a k-set is a neighborhood, we may evaluate if any other point in the dataset lies within the convex hull of the neighborhood. From the definition of convex sets, a point $\mathbf{p}$ may lie inside the convex hull of n points $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ iff there exists a convex combination of $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ that equals $\mathbf{p}$. This may be set up as the following linear programming problem:

$$
\begin{aligned}
\min \quad & 1 \\
\text{s.t.} \quad & \mathrm{a}_1\mathbf{x}_1 + \mathrm{a}_2\mathbf{x}_2 + \ldots + \mathrm{a}_n\mathbf{x}_n \\
& \mathrm{a}_{ij} > 0 \\
-&\mathrm{a}_{ij} > 1 \\
& \mathrm{a}_{11} + \mathrm{a}_{21} + \ldots + \mathrm{a}_{n1} = 1 \\
& \mathrm{a}_{21} + \mathrm{a}_{22} + \ldots + \mathrm{a}_{n2} = 1 \\
& \vdots \\
& \mathrm{a}_{1m} + \mathrm{a}_{2m} + \ldots + \mathrm{a}_{nm} = 1
\end{aligned}
\tag{3.6}
$$

where $\mathrm{a}_1, \mathrm{a}_2, ..., \mathrm{a}_n$ are coefficient vectors of $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ and a_{ij} is the j^{th} element of a_i. The complexity of the current fastest algorithm for solving a linear programming problem is $O^*(n^{2.055})$ [41]. Thus, the overall complexity of finding k-sets that are possible k-nearest neighborhoods is a product of the complexity of finding unique k-item combinations and the cost to evaluate an LP for each k-set. This is upper bounded by $O({}^nC_k) + O^*({}^nC_k \cdot n^{2.055}) \leq O^*(n^k(1 + n^{3.055}))$. While this is polynomial in n and k, it is still computationally expensive even for smaller values of k and moderately-sized datasets.

In order to avoid this computational complexity, we use a sampling-based strategy to select fixed numbers of neighborhoods at random and use them for the computation of MNTV. In the current work, this is done by selecting a verification dataset from the training set and finding the k-nearest neighbors of each point in the verification set. The sample of the neighborhoods found in this way is used to calculate the MNTV.

3.5 MNTSD with Inverse-Distance Weighting of Samples

So far, the dispersion measures were considered for the Euclidean distance metric, with equally weighted samples. However, the same considerations fail to exist when samples themselves are unequally weighted.

Inverse-distance weighting is a popular sample-weighting scheme [42] that assigns larger weights to training points that are closer to a test point in a k-nearest neighborhood, and vice versa. The weights are proportional to the reciprocal of the distance, based on a chosen distance metric.

When all the instances in a neighborhood set are equally weighted, the decision surface does not vary within a neighborhood. This is especially relevant for regression. Equally weighted training samples result in a constant value across the points for which a particular neighborhood set is the k-nearest neighborhood. Inverse-distance weighting, however, results in a curved decision surface, parametric in the distance from each training instance – for every point for which a set of k instances is the k-nearest neighborhood, the predicted value is different. Thus, the variance of the target values of a k-nearest neighborhood would not measure the deviation of each point from a constant predicted value.

For example, in Figure 3.1, a neighborhood of 5 points is randomly created to illustrate the effect of leaving a singe point out in KNN regression with inverse-distance weighting. The subfigure in the top left consists of the decision surface created with all five points, and in every other subfigure, a single point is left out. From this figure, we can infer that KNN with inverse-distance weighting is extremely sensitive to the variance of the set of points as the geometry of the decision surface in each case is drastically different.

To simplify the formulation and interpretation of the MNTSD for inverse-weighted KNN, we measure the effect of each point in a k-nearest neighborhood to the

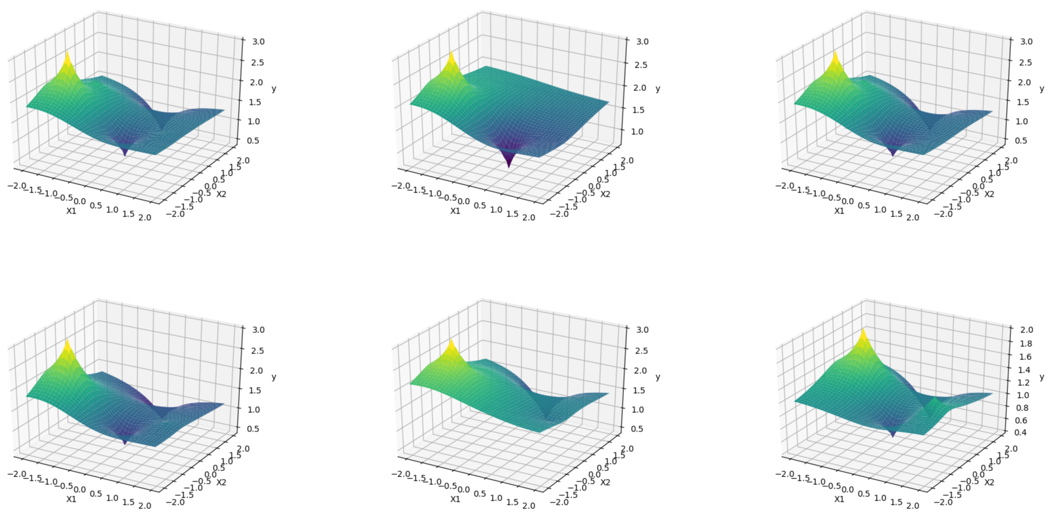

Figure 3.1. Example: Variation in decision surface while leaving one instance out at a time.
The $X1$ and $X2$ axes represent the features and the y axis represents the target value.

decision surface. This is done by calculating the variance in a leave-one-out manner: by excluding a point and computing the target variance of the remaining points, for each point in the k-nearest neighborhood. This is done by averaging the variance by leaving one-sample-out. The leave-one-out variance for a k-nearest neighborhood is described in Equation 3.7.

3.5.1 Average Leave-One-Out Variance

For a set of n points, $S = \{x_1, x_2, ..., x_n\}$ The average leave-one-out variance for a set is defined as:

$$\text{ALOOV} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n-1}\left[\left(\sum_{j=1}^{n}(x_j-\mu_i)^2\right)-(x_i-\mu_i)^2\right] \tag{3.7}$$

where n is the number of elements in the entire given set S, $n-1$ is the number of elements in a subset S_i that excludes element x_i, and μ_i is the mean of the subset S_i.
Unbiased Average Leave-One-Out Variance: The unbiased average leave-one-out variance for a set is defined as:

$$\text{ALOOV} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n-2}\left[\Big(\sum_{j=1}^{n}(x_j-\mu_i)^2\Big)-(x_i-\mu_i)^2\right] \tag{3.8}$$

The unbiased average leave-one-out variance has a nice property that it can be shown to equal the unbiased variance of the set.

Equations 3.7 and 3.8 present the definitions of ALOOV in a general context. In the context of target variance reduction, in the sets of points (represented using x) would be the k target values of a k-nearest neighborhood set.

Theorem 1: *The unbiased average leave-one-out variance is the same as the unbiased variance of the set.*

The proof of Theorem 1 is presented in Appendix A. Because of this result, the square root of the unbiased variance is used whenever the MNTSD is calculated. And because of this theorem, the unbiased variance can be used when training instances are also weighted.

3.6 Empirical Results on Comparing the Performance of a Learner with Localized Dispersion Measures

The effectiveness of using the MNTSD and MNTE as characteristics of a dataset are demonstrated in the following experimental results. Here the correlation between the proposed measures and the classification and regression performance of KNN are investigated using a set of common datasets. The datasets were acquired from the UCI repository.

3.6.1 Datasets

For the evaluation of the metrics, the following datasets from the UCI repository [43] were used. These datasets were selected to provide a representative set in terms of feature number and complexity: they are all small to medium sized datasets, with small to moderate number of features. For an initial exploration of the methods formulated in this thesis, small to medium sized datasets were chosen.

The following datasets were used to evaluate MNTSD for regression:

- *Airfoil Self Noise*: This dataset is obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel by NASA [44]. It has 1503 instances, 5 features, and one label, corresponding to self-noise.
- *Auto MPG*: This dataset contains data of city-cycle fuel consumption of automobiles. It is revisited from the CMY StatLib Library [45]. It has 398 instances, 7 features (excluding the model name of the automobile), and one label corresponding to miles per gallon (MPG) of an automobile.
- *Concrete Compressive Strength*: This dataset concerns different attributes of concrete [46]. It has 1030 instances, 8 features, and one label representing compressive strength.
- *Energy efficiency*: This dataset consists of requirements of heating and cooling loads of buildings as a function of building parameters [47]. Each instance has two labels: the heating load and the cooling load. In this thesis, predicting the heating and cooling loads are treated as two independent learning tasks. This dataset has 768 instances, 8 attributes, and two labels (corresponding to the heating and cooling loads).
- *QSAR aquatic toxicity*: This dataset contains 8 attributes (molecular descriptors) of 546 chemicals used to predict quantitative acute aquatic toxicity to-

wards Daphnia Magna, which is a small planktonic crustacean [48]. It has 546 instances (corresponding to 546 chemicals), 8 features, and one label representing a quantitative response to aquatic toxicity.

- *QSAR fish toxicity*: This dataset contains values for 6 attributes (molecular descriptors) of 908 chemicals used to predict quantitative acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow) [49]. It has 908 instances (corresponding to 908 chemicals), 6 attributes, and one label representing a quantitative response to aquatic toxicity.
- *Wine quality (red)*: This dataset contains quantitative attributes of the ingredients of wine [50]. It has 4898 instances, 11 features, and one label representing the quality of wine.
- *Yacht Hydrodynamics*: This dataset contains hydrodynamic data of sailing yachts, and can be used to model the performance of yachts [51]. It has 6 features and one label.

The following datasets were used to evaluate MNTSD and MNTE for classification:

- *Blood Transfusion Service Center*: This dataset contains attributes related to blood donation and can be used to model if a donor donated blood in March 2007 [52]. It has 4 features and one binary classification label.
- *Breast Cancer Wisconsin (Original)*: This is a popular dataset in the literature of clinical trials of breast cancer [53]. It has 10 features and one binary label representing a benign or malignant case.
- *Cardioctography*: This is a dataset of fetal cardioctograms [54]. It has 2126 instances, 22 features, and one label corresponding to three classes of fetal state.

- *Caesarian Section*: This dataset contains attributes related to cesarian section (c-section) birthing [55]. There are 80 instances, 4 features and one label representing whether a childbirth will be cesarian or not.
- *Diabetic Retinopathy Debrecen*: This dataset consists of features extracted from images, used for modeling diabetic retinopathy [56]. It contains 1151 instances, 19 features and one binary label indicating whether signs of diabetic retinopathy exist in a sample.
- *Haberman's Survival*: The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. It contains 306 instances, 3 features, and one binary attribute indicating survival (greater than 5 years or less than 5 years) after surgery.
- *Indian Liver Patient Dataset (ILPD)*: This dataset contains records of liver patients [57]. It has 583 instances, 10 attributes, and one binary label indicating the existance of a liver disease.
- *Immunotherapy*: This dataset contains information about wart treatment results of 90 patients using immunotherapy [58]. It consists of 90 instances, 7 features, and one binary label indicating the success of treatment.
- *Wine*: This dataset contains features from chemical analysis used to determine the origin of wines [59]. It has 178 instances, 13 attributes and one label representing three classes of wine.

3.6.2 MNTSD for Regression

The results below show both MNTSD and Mean-Squared Error (MSE) as a function of neighborhood size k for different regression problems. The results were obtained by randomly selecting multiple neighborhoods for varying k. For each case, the data was divided into training and test data with MNTSD calculated over the training set and the test set serving for the calculation of the MSE. Each case was repeated ten times with Monte-Carlo cross-validation [60]. Here, each set was divided into training and test sets using a $4 : 1$ ratio. Care was taken to ensure that computation of MNTSD and MSE were done on the same splits of the data to minimize random errors from skewing the results.

For a majority of the datasets, it can be observed that the MNTSD is somewhat indicative of the MSE performance of the method: we expect that a larger MNTSD would indicate a larger error due to variance, and thus a larger MNTSD should correspond to a larger MSE and vice versa. For the datasets Concrete and Qsar Aqua Tox, there is a close correlation between the two. This is less apparent in Energy-cool and Energy-heat, and somewhat apparent in Auto MPG. In the Airfoil dataset, the MNTSD fails to indicate anything about the MSE for small values of k. Upon close inspection, however, a correlation between MNTSD and MSE can be seen for larger values of k. In Wine quality and yacht hydrodynamics, the fluctuations in MNTSD and MSE correlate fairly well, albeit at different rates.

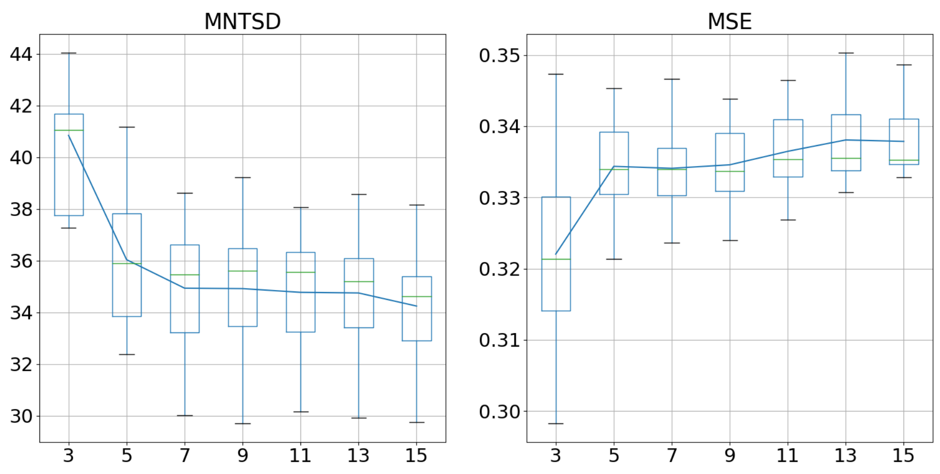

Figure 3.2. Boxplots for the Airfoil dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a function of Neighborhood Size k for the Airfoil dataset.

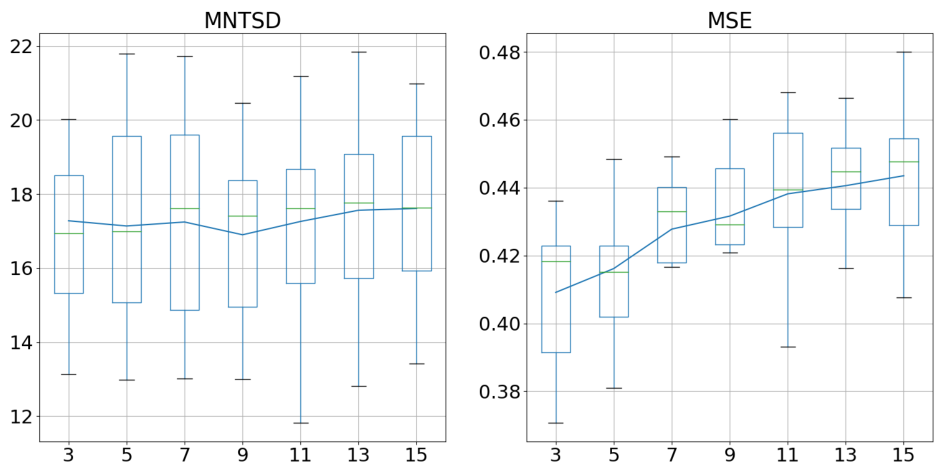

Figure 3.3. Boxplots for the Auto MPG dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a function of Neighborhood Size k for the Auto MPG dataset.

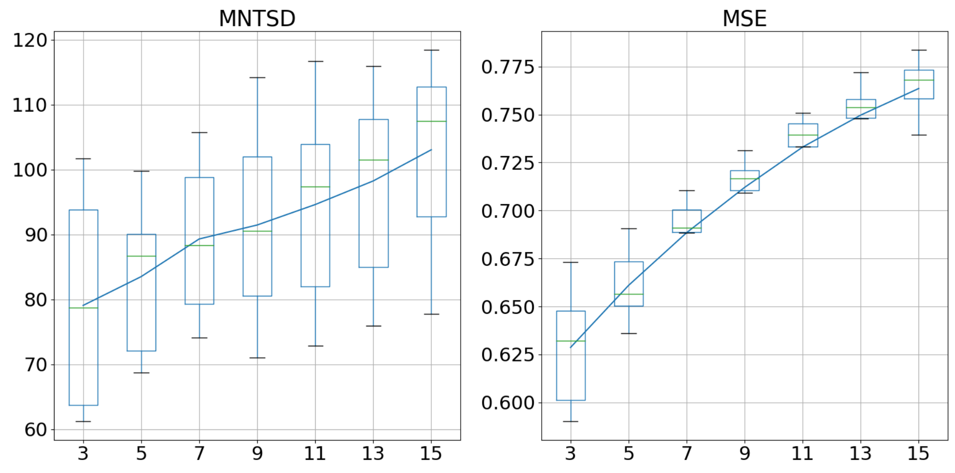

Figure 3.4. Boxplots for the Concrete dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a function of Neighborhood Size k for the Concrete compressive strength dataset.

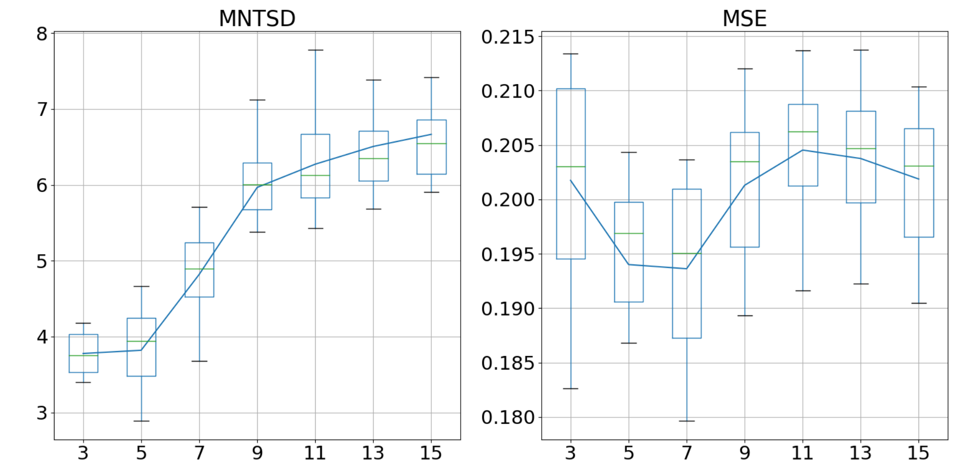

Figure 3.5. Boxplots for the Energy (cool) dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a function of Neighborhood Size k for the Energy (cool) dataset.

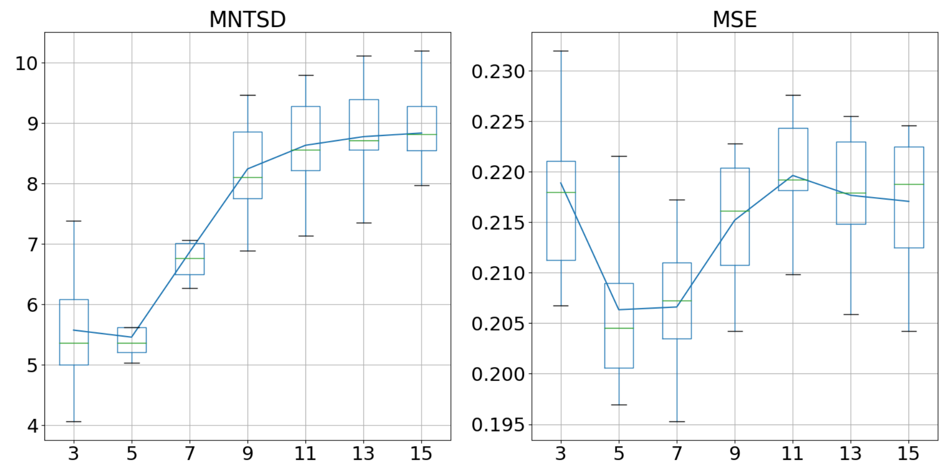

Figure 3.6. Boxplots for the Energy (heat) dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a function of Neighborhood Size k for the Energy (heat) dataset.

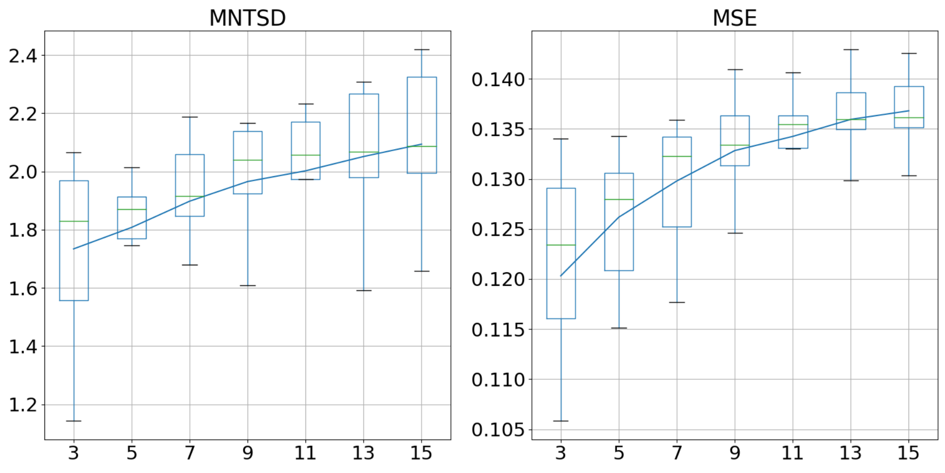

Figure 3.7. Boxplots for the QSAR aqua toxicity dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a Function of Neighborhood Size k for the QSAR aqua toxicity dataset.

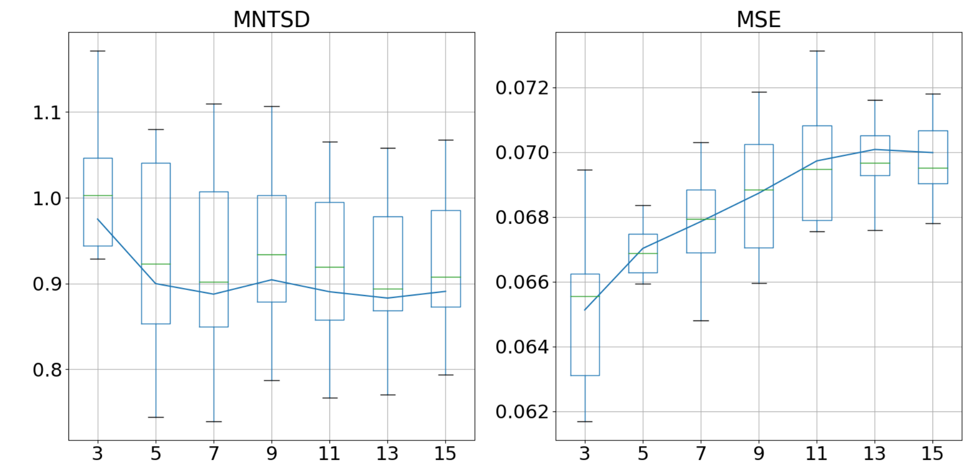

Figure 3.8. Boxplots for the QSAR fish toxicity dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a Function of Neighborhood Size k for the QSAR fish toxicity dataset.

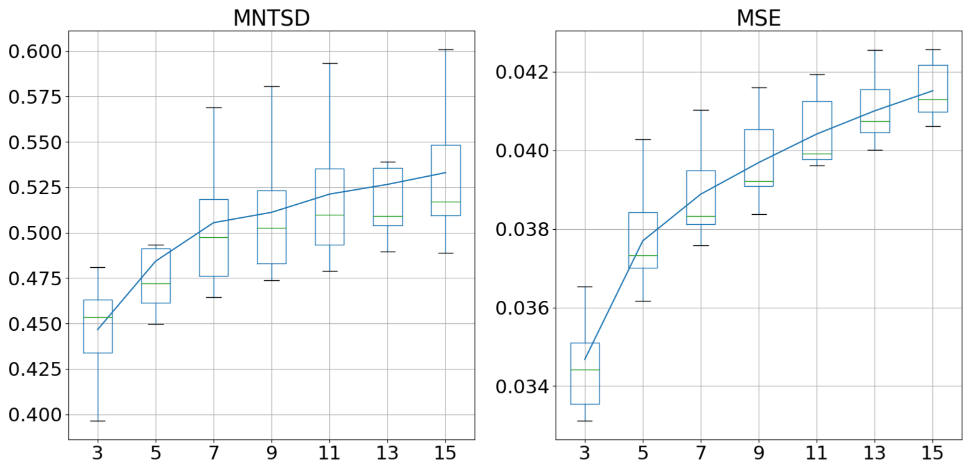

Figure 3.9. Boxplots for the Wine quality dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a Function of Neighborhood Size k for the Wine quality dataset.

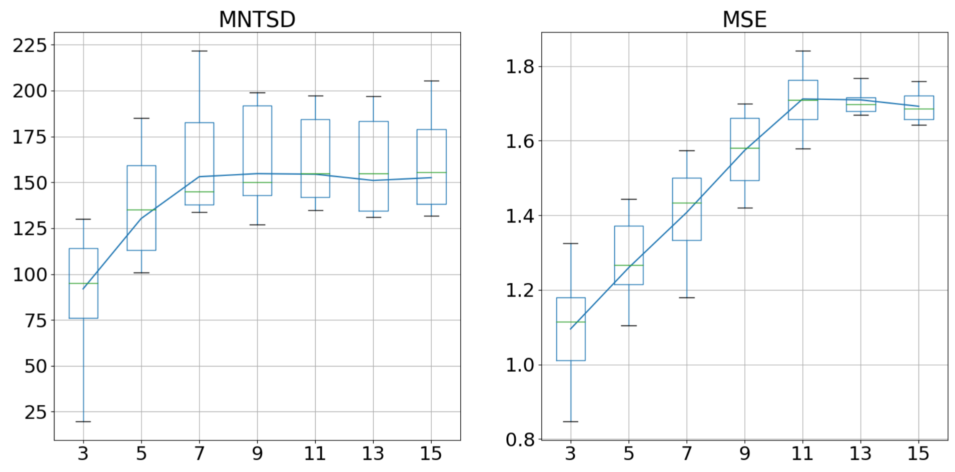

Figure 3.10. Boxplots for the Yacht hydrodynamics dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTSD (left) and MSE (right) as a Function of Neighborhood Size k for the Yacht hydrodynamics dataset.

3.6.3 MNTSD and MNTE for Classification

The results below show MNTE, MNTSD, and accuracy as a function of neighborhood size k for different classification problems. The results were obtained by randomly selecting multiple neighborhoods for varying k. Once again, for each case, the data was divided into training and test data with the MNTE and MNTSD calculated over the training set and the test set serving for the calculation of the accuracy. Each case was repeated ten times with Monte-Carlo cross-validation. Each set was divided into training and test sets in a $4 : 1$ ratio, and care was taken to ensure that computation of MNTSD and MSE were done on the same splits of the data to minimize random errors from skewing the results.

Here again, it can be observed that the MNTSD is somewhat indicative of the MSE performance of the method: we expect that a larger MNTSD would indicate a larger error due to variance, and thus a larger MNTSD should correspond to a lower accuracy and vice versa. For the datasets Cardioctography and Wine, there is a fairly good negative correlation between the MNTSD and accuracy. This is less apparent in the remaining datasets, but once again, for certain ranges of k, there is a good negative correlation between the MNTSD and the accuracy. The MNTE, on the other hand, is generally less expressive of the accuracy than the MNTSD: for most cases, we see that with an increase in MNTE, the accuracy also increases, which is not what we expect. However, there are a few cases and ranges of k where the MNTE is representative of accuracy: for example, in the C-Section dataset, with an increase in MNTE, the accuracy is seen to decrease; in Haberman, for $k \leq 7$, the MNTE and and accuracy show a negative correlation.

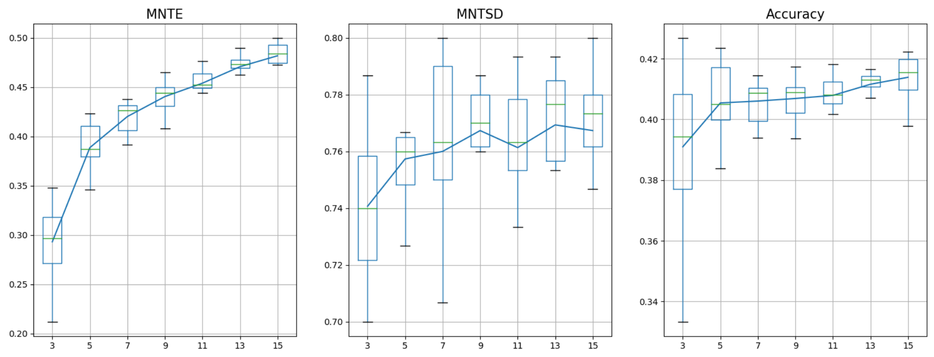

Figure 3.11. Boxplots for the Blood Transfusion dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Blood Transfusion dataset.

In the next chapter, we use the MNTE and MNTSD to optimally scale features to improve performance.

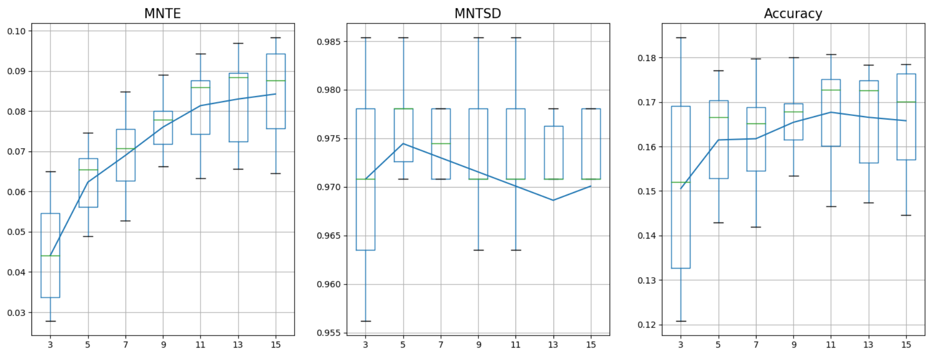

Figure 3.12. Boxplots for the Breast Cancer dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Breast Cancer dataset.

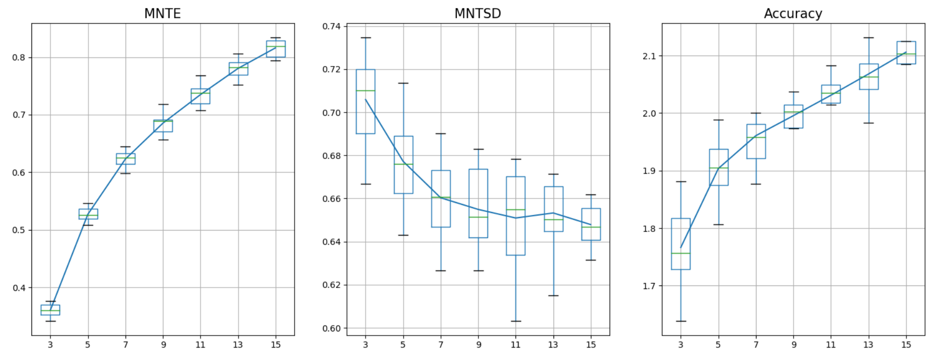

Figure 3.13. Boxplots for the Cardioctography dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Cardioctography.

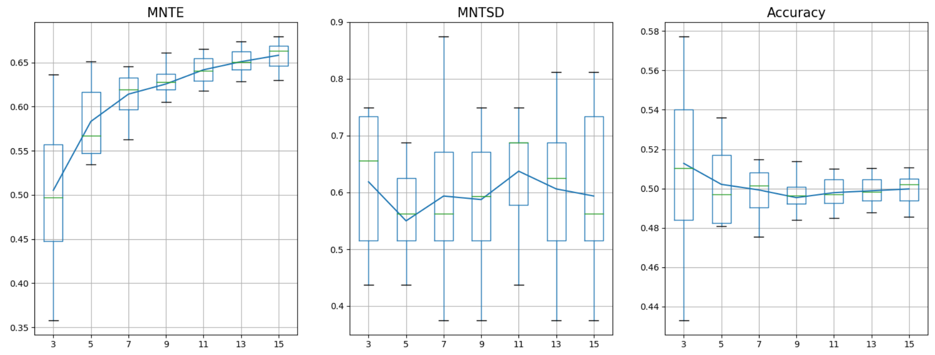

Figure 3.14. Boxplots for the C-Section dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the C-Section.

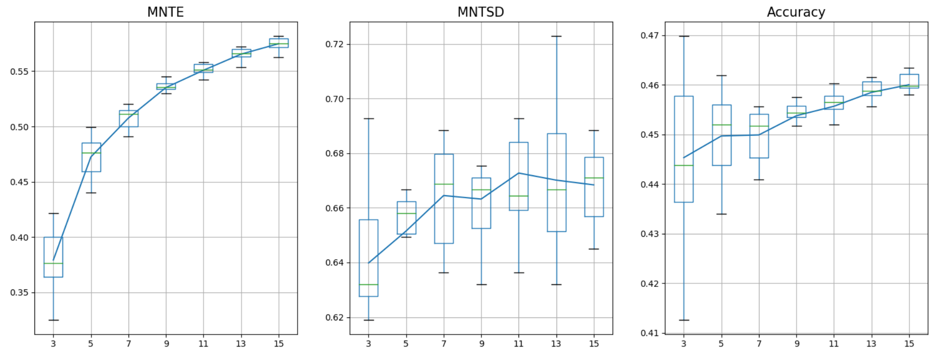

Figure 3.15. Boxplots for the Diabetic Retinopathy dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Diabetic Retinopathy dataset.

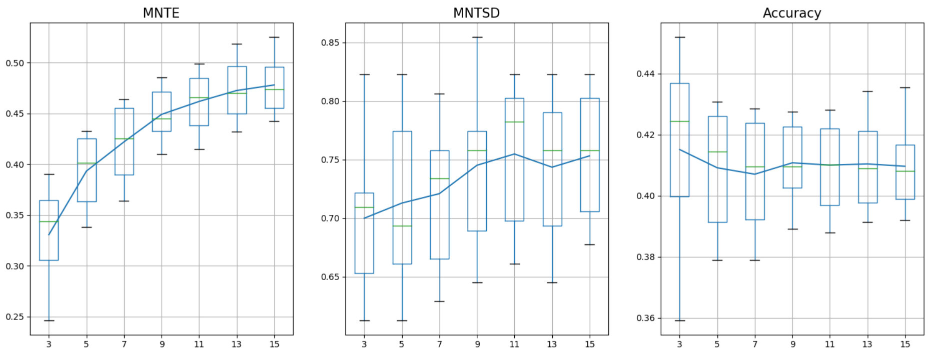

Figure 3.16. Boxplots for the Haberman dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Haberman dataset.

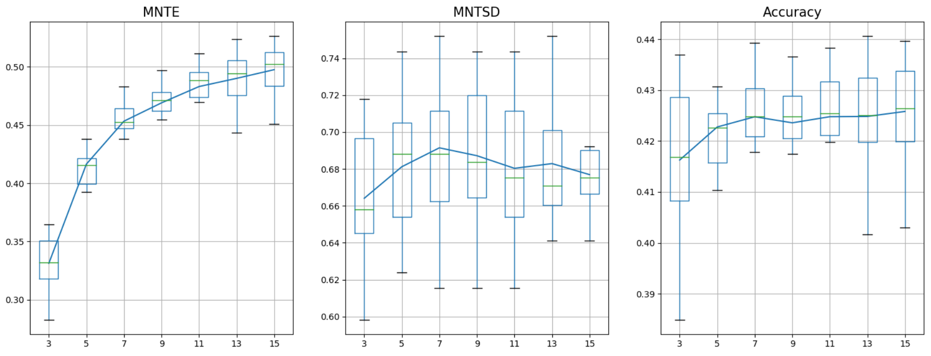

Figure 3.17. Boxplots for the ILPD dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the ILPD dataset.

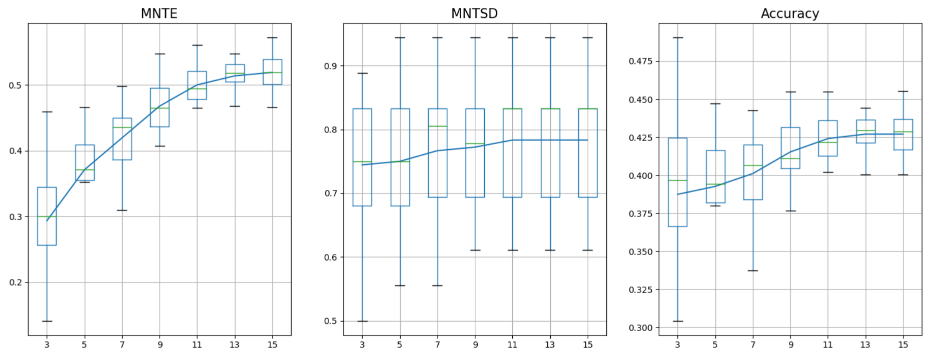

Figure 3.18. Boxplots for the Immunotherapy dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Immunotherapy dataset.

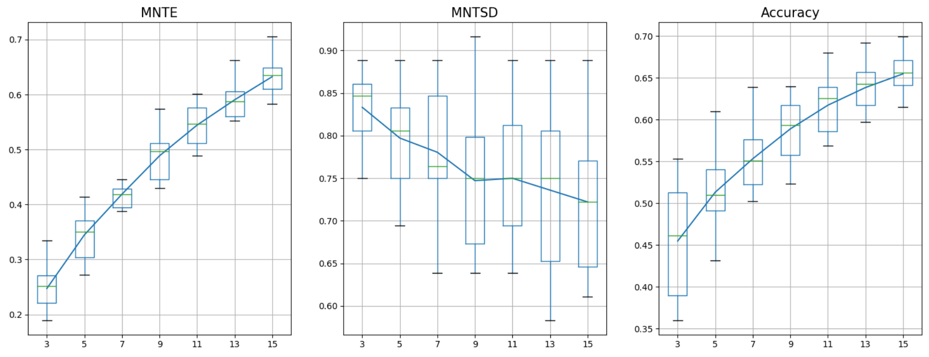

Figure 3.19. Boxplots for the Wine dataset.
Boxplots for 10-fold Monte-Carlo Cross-Validation showing MNTE (left), MNTSD (center) and MSE (right) as a function of Neighborhood Size k for the Wine dataset.

CHAPTER 4

OPTIMAL FEATURE SCALING

Some of the long-standing questions regarding KNN are related to the appropriate distance metrics that should be used, the weighting of features and the selection of appropriate features. An assumption that is made while building machine learning algorithms is that the training and testing data are drawn from the same distribution. With this assumption, we ask the question in the context of KNN: what if features could be scaled in a manner that reduces the variance in the output variable in every possible neighborhood? One particular approach is to reduce the error due to variance in a dataset. A minimization of model variance would lead to each neighborhood predicting a consistent value for a large range of feature sets, or inputs, to which they are the k-nearest neighborhoods. In the last section, two measures of label dispersion were introduced that have been shown to provide empirical estimates of errors arising due to variance of data.

In this chapter, we extend their usage from being an empirical estimate to being an objective function of an optimization problem: we introduce a new method of optimizing feature weights, which can be used in both classification as well as regression tasks called minimization of mean neighborhood target standard deviation (MMNTSD), and a new method of optimizing feature weights, which can be used in only classification called minimization of mean neighborhood target entropy (MMNTE).

The optimization objective of the MMNTSD model is described in Eq. 4.1.

$$\min \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(y_{n,i}-\mu_n)^2} \tag{4.1}$$

The optimization objective of the MNTE model is described in Eq. 4.2.

$$\min \; -\frac{1}{N}\sum_{n=1}^{N}\frac{1}{c}\sum_{i=1}^{c}p_i\log(p_i) \tag{4.2}$$

4.1 Selecting the Right Optimization Approach

Since KNN does not establish an explicit functional relationship between features and their target values, the functional relationship between the optimum weights and the best target values is essentially a black-box. An optimization of feature weights based on MMNTV cannot be done using a gradient-based approach. Hence *metaheuristic* approaches are used to optimize the weights. Another important aspect of this model is that the objective is inherently a property of the dataset – thus the optimization of this objective does not explicitly require a feedback based on the performance of the learner. This is especially advantageous in cases with less data, where a training-testing split could greatly alter the hypothesis space.

Most popular approaches to optimization utilize information about gradients to move towards a local-optimum. When a function is convex or concave and has a single global optimum, then gradient-based approaches are guaranteed to be able to discover the optimum point. However, a larger family of optimization objectives are non-convex in nature. A gradient-based algorithm such as gradient descent would get stuck in a local optimum when tried on a non-convex function. In order to deal with these objectives, various approaches such as relaxing the function to a convex objective, or using algorithms that are suited for non-convex objectives have been developed. Neither of these approaches are without caveats – optimization of non-convex functions is an NP-complete problem and therefore any computationally efficient approach of optimizing a non-convex function does not guarantee that a global optimum will be found. Nevertheless, these methods yield approximate solutions that are usable and are generally better than what is found using gradient-based methods.

The three algorithms that are developed here are based on genetic algorithm, local-best particle swarm optimization, and global-best particle swarm optimization. These algorithms concurrently evaluate multiple candidate solutions and use a feedback mechanism to evaluate how good each solution is – they are inherently a form of reinforcement learning used in the context of optimization. The details of their working in the context of the MMNTSD and MMNTE models is outlined in Section 4.4.

4.2 Related Work on Feature Scaling for KNN

Two new methods of selecting the nearest neighbors, namely axis-balanced KNN and box-KNN are reported in [61], wherein samples are directionally selected to maintain uniformity of sample selection from multiple directions around the test point. The performance of KNN with this modification is significantly higher than for the simple Euclidean-distance based case for many data sets were data was collected using non-uniform sampling and where thus significant variations in terms of sample density exist. In [62], the authors propose a fusion-distance-metric based on selecting the nearest neighbors of a point using two different distance metrics, whose importances are weighted. The accuracies are reported for seven datasets from the UCI repository and it is shown that their method achieves improvements over simple Euclidean-distance based KNN classification. In [63], gravitation-based potentials of influence of each point in a fixed search radius is used for classification – this is used in imbalanced datasets and is shown to significantly improve accuracy over KNN. A method based on similarity of neighborhood points using angular projections, called dependent KNN (dNN) is tried in [64] and tested on six synthetically generated, non-linearly separable datasets: dNN generally performs well with non-linearities. Linear feature weights, or equivalently, transformations of the distance metrics are learned in [65] and [66]. In [65], a metric is learned for KNN regression that is modeled using a convex optimization problem. In [66], methods of linear feature weighting are tried based on principal component analysis and independent component analysis. The results show that appropriately selecting linear feature weights can lead to lower error of KNN. KNN classification based on fuzzy sets is tried in [67]. In this work, the authors formulate a classifier based on intervals of values. The number of intervals and the boundaries of each interval used are the hyper-parameters of the system, which are optimized using the CHC genetic algorithm. The results show that this optimization outperforms KNN and many other variants of KNN. A similar optimization-oriented approach is tried in [68] where the parameter k is learned using evolutionary computation. In [69], an adaptive distance metric is learned based on modeling the non-linear spatial distribution of data. The metric is learned based on empirical risk minimization, and the authors describe it as risk-based weighted nearest neighbour (RBWNN) classification approach. Here, the risk associated with each neighbor (in a neighborhood) in each dimension is computed based on a leave-one-out cross-validation within the training process. The final risk is calculated by averaging the risk over all the neighbors, and based on this, all of the features are locally weighted. The method is tried on 10 datasets from the UCI repository and significantly outperforms KNN and many other variants of KNN.

A majority of the current work involves either computing feature weights or learning a distance metric. The contributions of this paper are thus concomitant to existing work that improves the performance of KNN using evolutionary computing. The particular contributions of this thesis are:

- *The MNTE and MNTSD are not measures of performance*: The optimization objectives in MMNTSD and MMNTE are not a feedback based on accuracy or MSE, or any other measure of performance. Since the MNTE and MNTSD are characteristics of a dataset, they can be calculated for any dataset (for a fixed k) and for any transformation of a dataset, and their optimization is explainable as an improvement of an estimate of the error due to variance in the dataset.
- *Flexibility inherent to the methods*: In this thesis, the methods used to optimize the MNTSD or MNTE are based on the sampling method described in Section 3.4. However, the computation can be done in various different ways: based on modeling the distribution of the input space, by exhaustive computation, etc. This is useful as different strategies can be developed for different datasets with varying sizes, distributions, etc.
- *Small Datasets*: Evaluation by training-testing splitting of smaller datasets can often skew the hypothesis space for KNN. In that case, with appropriate modeling and discovery of neighborhoods of in input space, one can preserve all the training samples in the data and then provide an empirical estimate of the error.
- *Novel optimization approaches*: The metaheuristic algorithms of genetic algorithms and particle swarm optimization are two very different models. To the best of the author's knowledeg, particle swarm optimization has hitherto been unexplored for feature optimization in KNN.

In this thesis, metaheuristic-driven approaches are used to optimize independent feature weights. Reiterating the aspect of flexibility of the new methods: the possible data transformations that can be used within MMNTSD and MMNTE are not restricted to independent feature weights. The reason for this restriction in the evaluations here is to build on a simple paradigm that can help improve the accuracy of KNN.

4.3 A Motivating Example

We begin with an illustration of the impact of good feature weights learned using MMNTSD. We create an artificial dataset X with two features, X_1 and X_2 from multivariate normal distributions, and a target y. The target value is linearly related to only X_1, and X_2 is essentially an unimportant attribute. The data generation model is as follows:

$$\begin{aligned}
\mu_1 &= (5, 2) \\
\Sigma_1 &= \begin{bmatrix} 1.0 & 0.8 \\ 0.8 & 1.0 \end{bmatrix} \\
\mu_2 &= (0, -1) \\
\Sigma_2 &= \begin{bmatrix} 1.0 & 0.1 \\ 0.1 & 1.0 \end{bmatrix} \\
y &= 5X_1 + \mathcal{N}(0, 1.0)
\end{aligned} \tag{4.3}$$

For this example, 50 instances are created using (μ_1, Σ_1), and 50 instances are created using (μ_2, Σ_2). The use of two distributions essentially leads to two clusters in the training set. The target variable y is calculated in the same way for independent variables of either distribution. The set of points generated using this method (without scaling) is visualized in the left graph in Fig. 4.1.

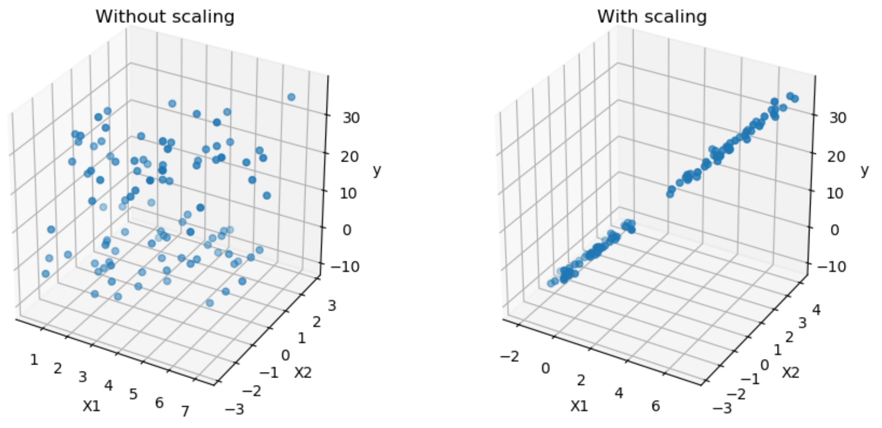

Figure 4.1. Visualization of the effect of appropriate feature weights..

Assuming linear feature scaling where the scaled features are $\widetilde{X_1} = w_1 X_1$ and $\widetilde{X_2} = w_2 X_2$, we use a genetic algorithm (as described in Section 4.4.1) to minimize the MMNTSD objective. This results in the following weights:

$$\begin{aligned} w_1 &= 0.646 \\ w_2 &= 0.049 \end{aligned} \tag{4.4}$$

where w_1 is the weight of X_1 and w_2 is the weight of X_2. What should be observed here is that the weight of X_2 is much lower than X_1. While feature weights by themselves do not necessarily indicate a relative importance of features, it is interesting to note that w_2 is much closer to zero than w_1. The dataset is visualized after these weights are applied to them (after scaling) in the right graph of Fig. 4.1. Here we see that for the most part, after the weights are applied, X_1 seems to have a linear relationship with y, and X_2 is largely neglected. We know that is indeed true because the data generation model is linear in X_1.

Likewise, a synthetic test was also performed. A test set with 50 points from the distribution (μ_1, Σ_1) and 50 points from the distribution (μ_2, Σ_2) was created based on the same simulation model as in Eq. 4.3. KNN regression was tested with scaling using the weights in Eq. 4.4, and without using them. Without scaling, the MSE was calculated to be 2.84 and with scaling, it was 1.74. Clearly, there is an improvement in performance after the weights were applied.

Although this is a simple example, it provides us with the insight that this method could improve the accuracy of KNN. The simplicity of the model helps us gain intuition about its working and effect on the performance of the model.

In the following we first introduce the approaches for localized weight optimization that were developed based on Genetic Algorithms and Particle Swarm Optimization, and then compare the resulting KNN performance with common KNN techniques in Section 4.6.

4.4 Methods for Feature Scaling Optimization

In this section, genetic algorithms and particle swarm optimization, and their use in the context of MMNTSD and MMNTE is explained.

4.4.1 Genetic Algorithm (GA)

Genetic algorithms [70, 71] are inspired by evolutionary biology and follow an approach that is similar based on the idea of *survival of the fittest.* The fundamental idea is drawn from the crossover of genes in living beings and the preservation of genes over successive generations. The important aspects of GAs and their relation to the problem of optimizing MNTSD or MNTE are described below.

1. *Gene*: A gene in a GA is a candidate solution of the optimization problem. In the current context, feature weights are genes.

2. *Gene Fitnesses:* The fitness of each gene is a measure of how good the gene is, and in the context of the current problem, how good a feature weight vector is. In the current work, it is calculated for each candidate solution by applying the weights to a training dataset and finding the MNTSD/MNTE – the MNSD/MNTE thus calculated is used as the fitness of each gene.

3. *Population Initialization:* An initial population of genes are drawn randomly from a specified distribution in the solution space. In the context of our problem, the solution space is that of the feature weights. The initial set of solutions is drawn from a uniform distribution in the space of the feature weights. The set of candidate solutions is also called the *gene pool*.

4. *Crossover:* In a GA, a crossover refers to the derivation of a candidate solution, called a *child* solution, from a set of other genes, called the *parent* solutions. In the current work, a child is derived by averaging two parents.

5. *Mutation:* A mutation operation refers to the alteration of a gene. Here, mutation is done by adding a uniform random variable in the range $[-1, 1]$ to each element in a child vector.

6. *Selection:* After a set of children are created, their fitness is calculated and a specified number of the best children are preserved as parents for the succeeding iteration in the GA. They form the next *generation* of genes and are in turn crossed over and mutated. The process is repeated for either a number of steps or until a better solution can't be found. In the current work, we repeat the GA for a fixed number of iterations.

The working of the GA as used in this work is illustrated in Figure 4.2 and the algorithm is elaborated in Algorithm 2. Two feature weights in an arbitrary problem are crossed-over by item-wise averaging and randomized mutation. The new gene has it's own fitness and the process is repeated many times.

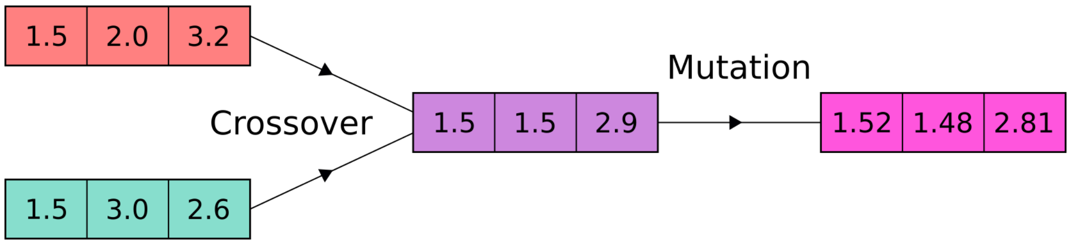

Figure 4.2. Working of GA on Feature Weights.

Algorithm 2 Genetic Algorithm

```
1:  input: $n\_dims$ is the number of dimensions in the solution space
    $n\_init$ is the size of the initial population
    $n\_cross$ is the number of individuals selected for crossover and mutation
    $n\_select$ is the number of individuals selected for the next generation

2:  output: the gene with the best fitness

3:  procedure GA($n\_dims, n\_init, n\_cross, n\_select$)
4:      $pool \leftarrow$ initialize($n\_init$, $n\_dims$)
5:      $iter\_max \leftarrow 100$

6:      for $n\_iter \leftarrow 0$, $n\_iter < iter\_max$, $n\_iter++$ do
7:          $children \leftarrow$ crossover($pool, n\_cross$)
8:          $children \leftarrow$ mutate($children, n\_cross$)
9:          $fitness \leftarrow$ MNX($children, n\_cross$)
10:         $best\_children \leftarrow$ argsort($fitness$)[$n\_select$]
11:         $pool \leftarrow children[best\_children]$
        return $children[0]$
```

4.4.2 Particle Swarm Optimization (PSO)

Particle swarm optimization [72, 73] is inspired by the mechanism by which swarms of birds or fishes find food. In this approach, a set of candidate solutions called *particles* are concurrently evaluated, and successively updated to find the best optimum of a function. All the particles can communicate with each other based on a star-topology or a ring-topology, leading to the global-best and local-best variants of PSO, respectively. Each step in the algorithm is described below.

1. *Particle:* A particle is a candidate solution. In the current context, feature weights are particles.
2. *Swarm Initialization:* The swarm of particles is initialized in a manner similar to GA: as a set of uniform random vectors in the solution space.
3. *Position:* The position of each particle is the point that it occupies in the solution space.
4. *Velocity:* Each particle has a corresponding velocity – in this context, that means the distance each particle moves in an iteration. It is initialized as a set of uniform random vectors with the same dimensions as the solution space.
5. *Fitness:* The fitness of a particle in PSO is the same as the fitness in a GA – a measure of how good the particle is. In the current work, the fitness of each particle is the MNTSD/MNTE.
6. *Updating the Particles:* The particles are updated in each iteration of the algorithm by adding the velocity component to it. The velocity is updated based on a star-topology or a ring-topology, leading to different variants of PSO.

4.4.2.1 Global-Best PSO:

The update rule in the *gbest* variant of PSO is described below:

$$\begin{aligned} p_{i+1} &= p_i + v_i \\ v_{i+1} &= v_i + c_1 r_1 (p_{\text{best}} - p_{\text{current}}) + c_1 r_1 (g_{\text{best}} - p_{\text{current}}) \end{aligned} \tag{4.5}$$

where p_i and v_i are the positions and velocities, respectively, of a particle at the i^{th} iteration, p_{best} is the best position of an individual particle, p_{current} is the current position of the particle, g_{best} is the global best position, r_1 and r_2 are standard uniform random variables, c_1 and c_2 are exploitation factor and exploration factor, respectively. Algorithm 3 shows the detailed operation of Global-Best PSO.

Algorithm 3 Global-Best PSO

```
 1: input: $n\_dims$ is the number of dimensions in the solution space
    $n\_pop$ is the number of particles in the swarm
    $c_1$ is the exploitation factor
    $c_2$ is the exploration factor

 2: output: the particle with the best fitness

 3: procedure GBestPSO($n\_dims, n\_pop, c_1, c_2$)
 4:     $p \leftarrow$ initialize($n\_pop$, $n\_dims$)
 5:     $p_{\text{best}} \leftarrow p$
 6:     $v \leftarrow$ uniform_random($n\_pop$, $n\_dims$, $[-1, 1]$)
 7:     $iter\_max \leftarrow 100$
 8:     $fitness \leftarrow$ MNSD($p$)
 9:     $g_{\text{best}} \leftarrow p$[argsort($fitness$)[0]]
10:     for $n\_iter \leftarrow 0$, $n\_iter < iter\_max$, $n\_iter++$ do
11:         $r_1 \leftarrow$ uniform_random($n\_pop$, $[-1, 1]$)
12:         $r_2 \leftarrow$ uniform_random($n\_pop$, $[-1, 1]$)
13:         $v \leftarrow v + c_1 r_1 (p_{\text{best}} - p) + c_2 r_2 (g_{\text{best}} - p)$
14:         $p \leftarrow p + v$
15:         $fitness \leftarrow$ MNSD($p$)
16:         $g_{\text{best}} \leftarrow p$[argsort($fitness$)[0]]
17:         for $i \leftarrow 0$, $i < n\_pop$, $i++$ do
18:             $p_{\text{best}}[i] \leftarrow p[i] : p_{\text{best}}[i] <$ MNSD($p[i]$) : MNSD($p_{best}[i]$)
    return $g_{\text{best}}$
```

4.4.2.2 Local-Best PSO:

The update rule in the *lbest* variant of PSO is described below:

$$\begin{aligned} p_{i+1} &= p_i + v_i \\ v_{i+1} &= v_1 + c_1 r_1 (p_{\text{best}} - p_{\text{current}}) + c_1 r_1 (l_{\text{best}} - p_{\text{current}}) \end{aligned} \tag{4.6}$$

where the symbols have the usual meanings, l_{best} is the best position that is local to a particle (local in terms of fitness). The only difference between the *gbest* and *lbest* variants are in the use of the global best and local best positions in the update rule.

In the current work, the heuristic used to refer to neighbors in the context of a ring-topology is that of indexes. For example, the topological neighbor of particle $p[i]$ is $p[i+1]$, with the exception of the neighbor of $p[n-1]$ being $p[0]$, where n is the number of particles in the swarm. Algorithm 4 shows the detailed operation of Local-Best PSO.

4.5 Comparison with Principal Component Analysis

Component analysis is a category of unsupervised learning task wherein important signals or components within a dataset are discovered. Principal component analysis (PCA) [66] is one of the most popular methods of component analysis and has been extensively studied and used in the literature for feature extraction before training a supervised learning model. It discovers orthogonal, or decorrelated features in a dataset, and decorrelated features provide more information to a supervised learner.

An important aspect of PCA is that it does not take into consideration the target values, if they exist, in a dataset. The features extracted in PCA are based on a learning task that is completely independent of reduction of error for any supervised learning algorithm. This is in contrast with the principles on which MMNTSD and MMNTE are built – MMNTE and MMNTSD scale features by considering the labels of the dataset. The learning of the weights in itself is a supervised learning problem.

Algorithm 4 Local-Best PSO

```
1: input: $n\_dims$ is the number of dimensions in the solution space
   $n\_pop$ is the number of particles in the swarm
   $c_1$ is the exploitation factor
   $c_2$ is the exploration factor

2: output: the particle with the best fitness

3: procedure LBestPSO($n\_dims, n\_pop, c_1, c_2$)
4:     $p \leftarrow$ initialize($n\_pop$, $n\_dims$)
5:     $p_{\text{best}} \leftarrow p$
6:     $v \leftarrow$ uniform_random($n\_pop$, $n\_dims$, $[-1, 1]$)
7:     $iter\_max \leftarrow 100$
8:     $fitness \leftarrow$ MNSD($p$)
9:     for $n\_iter \leftarrow 0$, $n\_iter < iter\_max$, $n\_iter++$ do
10:        $r_1 \leftarrow$ uniform_random($n\_pop$, $[-1, 1]$)
11:        $r_2 \leftarrow$ uniform_random($n\_pop$, $[-1, 1]$)
12:        $l_{\text{best}} \leftarrow$ right_rotate($p_{\text{best}}$)
13:        $v \leftarrow v + c_1 r_1 (p_{\text{best}} - p) + c_2 r_2 (l_{\text{best}} - p)$
14:        $p \leftarrow p + v$
15:        $fitness \leftarrow$ MNSD($p$)
16:        for $i \leftarrow 0$, $i < n\_pop$, $i++$ do
17:            $p_{best}[i] \leftarrow p[i] : p_{best}[i] <$ MNSD($p[i]$) : MNSD($p_{best}[i]$)
18:    $g_{\text{best}} \leftarrow p[\text{argsort}(fitness)[0]]$
   return $g_{\text{best}}$
```

In this thesis, without going into the details of the working of PCA, we use it along with KNN for a benchmark comparison. For each of the datasets, 50% of the principal components are used after transformation; using 100% of components results in a re-aligning of the feature space, but does not affect relative distances between points.

4.6 Results

All the results are reported after a 10-fold Monte Carlo cross validation. Each method was executed on the same training-testing sets in order to prevent random

Dataset	Existing Methods					New Methods					
	Variant	k	Prep.	Acc.	Var. of Acc.	Variant	k	Feat. Scaling	Measure	Acc.	Var. of Acc.
Blood Transfusion	KNN	9	None	74.87	0.10	KNN	7	GBest PSO	MNTSD	**75.93**	0.06
Breast Cancer	KNN	5	None	**97.96**	0.01	DKNN	3	GA	MNTE	97.81	0.02
Cardio Octography Fetal	KNN	3	PCA	90.26	0.0002	KNN	7	GBest PSO	MNTE	**92.49**	0.05
C-Section	KNN	13	None	60.62	0.44	KNN	9	GA	MNTE	**63.75**	1.37
Diabetic Retinopathy	KNN	7	None	67.19	0.04	KNN	11	LBest PSO	MNTSD	**68.57**	0.07
Haberman	KNN	9	PCA	**73.87**	0.17	KNN	9	GA	MNTE	73.23	0.13
ILPD	KNN	15	PCA	71.55	0.01	KNN	15	GA	MNTSD	**73.28**	0.13
Immunotherapy	DKNN	3	None	74.44	1.80	KNN	15	LBest PSO	MNTE	**77.78**	2.26
Wine	KNN	3	None	83.89	0.32	KNN	9	GA	MNTSD	**94.17**	0.08

Table 4.1. Results of classification. Metaheuristic optimization done here was based on MNTSD or MNTE.

errors. Classification accuracy and MSE are reported for classification and regression, respectively, and the variance in MSE and accuracy that is reported was the variance over 10 training-testing iterations.

The results shown here are a snapshot of the methods that resulted in the best average accuracy achieved over 10 iterations. An exhaustive set of results of all the methods is presented in Appendix B.

4.6.1 Classification

The classification experiments were performed on the datasets from the UCI repository introduced in Section 3.6.1. The results of classification are shown in Table 4.1. In this table, the best classical and best new method for each of the datasets are listed, together with the corresponding value for k, preprocessing method, and localized dispersion measure used. The best results for each dataset are highlighted in the table, showing that for most datasets the new feature scaling method outperforms the classical methods. In the few examples where performance is not increased, accuracy of the new method is very similar to the one achieved by the best classical method.

Dataset	Existing Methods					New Methods				
	Variant	k	Prep.	MSE	Var. of MSE	Variant	k	Feat. Scaling	MSE	Var. of MSE
Airfoil	KNN	15	None	33.92	5.74	DKNN	7	GBest PSO	**9.43**	7.67
Auto MPG	DKNN	9	None	15.78	5.36	DKNN	11	GBest PSO	**7.96**	1.02
Concrete	DKNN	5	None	71.96	222.92	DKNN	5	GBest PSO	**39.2**	59.22
Energy (cool)	KNN	5	None	3.61	0.46	KNN	15	LBest PSO	**3.26**	0.22
Energy (heat)	KNN	3	None	5.07	0.95	DKNN	7	LBest PSO	**0.39**	0.03
QSAR Aqua Toxicity	DKNN	5	None	1.64	0.07	DKNN	7	GBest PSO	**1.24**	0.01
QSAR Fish Toxicity	DKNN	7	None	0.91	0.01	DKNN	9	LBest PSO	**0.84**	0.01
Wine (red)	DKNN	9	PCA	0.32	0.002	DKNN	13	GBest PSO	**0.25**	0.002
Yacht	DKNN	3	PCA	75.44	936.41	DKNN	9	GBest PSO	**2.18**	1.19

Table 4.2. Results of regression. All metaheuristic optimization done here was based on MNTSD.

4.6.2 Regression

The regression experiments were performed on the datasets from the UCI repository introduced in Section 3.6.1. The results of regression are shown in Table 4.2. Again, the table lists the best performing classical and new method with the corresponding parameter settings. The results show that the new method significantly outperforms the classical method for all datasets, illustrating once more the benefit of the new methods for feature scaling.

In the next chapter, a subsampling algorithm based on MMNTSD is introduced and elaborated.

CHAPTER 5

DATA SUBSAMPLING

Since KNN is an instance-based learning approach, the smallest number of useful samples used to train a model would lead to the fastest querying times. For this purpose, being able to subsample the training dataset optimally to preserve only the most useful samples is imperative. However, since large sizes of datasets are recommended for most ML approaches, subsampling a dataset is in contrast to the best practices for data analysis and machine learning.

5.1 Related Work

The most common methods of data subsampling are to draw subsets uniformly from a dataset, or based on an estimate of the distribution of data, and sampling to overcome class imbalances. In [74], a method of undersampling specific to KNN is developed that filters instances in neighborhoods that belong to the majority class. There is an improvement in accuracy for unbalanced datasets. A novel hierarchical-clustering based approach for classification tasks, called CLUKER is developed in [75] wherein clusters of points are used as points in dense areas of points belonging to the same class. This approach results in better execution time of the KNN classification algorithm, while improving the accuracy. Another popular approach for solving the data imbalance problem is SMOTE [76] – this, however, is not an undersampling approach, but an oversampling approach, and it is not specific to KNN. The principle of its working is based on generating samples by utilizing nearest neighbor information. The samples generated by this method can be used within any learning algorithm. Subsampling methods for specific supervised learning algorithms have been developed. An optimal subsampling approach for softmax regression, based on the A-optimality and L-optimality criteria is formulated in [77]. The asymptotic normality is established, but the paper does not provide any experimental results on real datasets. The authors, however, state that they expect it to lead to good empirical error rates on real-world datasets. A similar approach has been developed in [78] for very large samples in logistic regression, and it is shown to work fast for large datasets. Two similar approaches based on ordinary-least squares and coefficient estimation are developed in [79]. Experimental results are provided and the methods are shown to outperform other popular methods of uniform subsampling, basic leveraging, approximate leveraging, and shrinkage leveraging.

Inspired by the methods that use optimality criteria to find subsamples that work well for specific learning approaches, we developed a randomized iterative subsampling method for KNN based on minimizing the MNTSD.

5.2 Method

The weight that is assigned to a sample in the training set is an estimate of how much it contributes to the MNTSD of a subsample – the lesser a training sample contributes to the label dispersion, the larger weight it is assigned and vice versa.

The implementation starts off with assigning equal weights to all points in the training set. Let the weight for training sample i be represented using w_i. The parameters that are set are those of k, and p, which is the desired proportion of the training set to be sampled. From the size of the training set and the value of p, the size of the resulting subsampled data set can be determined. Iteratively, training subsamples, and independent verification sets (for discovering k-nearest neighborhoods in the subsample) are drawn from the training set. Equal weights are initially assigned to all the training samples. Using the verification set, k-nearest neighborhood sets are discovered and their MNTSD is calculated. Points in a training subsample could belong to multiple k-nearest neighborhoods – for each point, the average MNTSD of all the k-nearest neighborhoods it belongs to, is calculated. Let this value be represented using $\tilde{v_i}$ for the i^{th} point in a training subsample. The weight updates are then assigned using the rule: $\tilde{w_i} = (1 - \tilde{v_i})$. If point i does not belong to any queried k-nearest neighborhood, the weights are updated as $\tilde{w_i} = -1$. In principle, this assigns lower weights to points that contribute to a large MNTSD and points that are not queried frequently, and larger weights to points that are frequently queried and contribute towards a lower MNTSD. The set containing all $\tilde{w_i}$ is then normalized between 0 and 1 and added to the sample weights w. The weights can be interpreted as probabilities using appropriate scaling.

The method is described in Algorithm 5.

5.3 Results

The sampling algorithm is used with 5-fold cross validation on the following regression datasets: Airfoil, Concrete, Energy (cool), Energy (heat) and Auto MPG. The results are provided in Tables 5.1 through 5.5. The MSE of different settings of the method are shown. Results are also provided for uniform sample weighting, for comparison.

In general, the results in this case do not show a significant improvement in the MSE. In most cases, uniform sampling and weighted sampling result in similar performance. Only in the Concrete dataset, there seems to be a fairly significant improvement for a subsampling proportion of 5% and for $k = 13$ and $k = 15$.

The next chapter concludes the thesis and discusses potential courses of future work.

Algorithm 5 Local-Best PSO

```
1: input: $X_{\text{train}}$ is the training set
   $Y_{\text{train}}$ is the set of training labels
   $X_{\text{verif}}$ is the verification set
   $n$ is the size of the training set
   $p$ is the proportion of the training set being sampled
   $w$ is the set of sample weights $k$ is the parameter of $KNN$
2: output: updated sample weights

3: procedure MNTSDSubsampling($X_{\text{train}}, Y_{\text{train}}, X_{\text{verif}}, Y_{\text{train}}, n, p, w, k$)
4:     Draw a subsample $X_{\text{sub}}, Y_{\text{sub}}$ of $X_{\text{train}}, Y_{\text{train}}$ containing $n \times p$ instances, using
   sample weights $w$
5:     $K \leftarrow$ the indexes of sets of neighborhoods that are found using $X_{\text{verif}}$
6:     $v \leftarrow \text{MNTSD}(K)$
7:     for $i \leftarrow 1$, $i < n$, $i++$ do
8:         $n\_hoods \leftarrow 0$
9:         $\tilde{w}_i \leftarrow 0$
10:        for $j \leftarrow 1$, $j \leq K.\text{len}()$, $j++$ do
11:            if $X_{\text{sub}}[i] \in K$ then
12:                $\tilde{w}_i \leftarrow \tilde{w}_i + (1 - v_i)$
13:                $n\_hoods \leftarrow n\_hoods + 1$
14:            if $n\_hoods == 0$ then
15:                $\tilde{w}_i \leftarrow -1$
16:     Normalize $\tilde{w}$ between 0 and 1.
17:     return $w + \tilde{w}$
```

Method	Proportion	k = 3	k = 5	k = 7	k = 9	k = 11	k = 13	k = 15
Weighted Sampling	p = 0.05	48.357	41.983	41.759	41.628	39.21	39.173	39.553
Uniform Sampling	p = 0.05	47.379	44.166	42.168	41.394	42.003	41.112	40.392
Weighted Sampling	p = 0.1	43.443	40.493	40.489	39.928	39.844	39.559	40.15
Uniform Sampling	p = 0.1	46.536	42.551	42.173	40.28	40.501	39.083	39.963
Weighted Sampling	p = 0.25	43.217	38.053	35.59	37.903	38.71	37.125	38.215
Uniform Sampling	p = 0.25	41.815	37.759	37.607	37.569	37.021	37.391	38.871
Weighted Sampling	p = 0.5	39.868	37.225	36.063	34.889	34.433	33.714	34.105
Uniform Sampling	p = 0.5	40.133	36.966	35.84	34.831	34.292	34.112	34.09

Table 5.1. MSE after sampling, Airfoil

Method	Proportion	k = 3	k = 5	k = 7	k = 9	k = 11	k = 13	k = 15
Weighted Sampling	p = 0.05	205.632	196.618	193.696	211.375	214.674	218.969	213.198
Uniform Sampling	p = 0.05	205.708	218.087	194.221	215.807	217.374	225.055	230.635
Weighted Sampling	p = 0.1	176.979	177.939	179.89	180.413	178.251	188.558	198.64
Uniform Sampling	p = 0.1	167.978	177.886	182.69	178.38	183.296	190.119	199.133
Weighted Sampling	p = 0.25	132.342	135.775	141.229	143.754	151.028	146.391	154.79
Uniform Sampling	p = 0.25	131.298	133.672	139.904	149.02	145.681	156.703	153.182
Weighted Sampling	p = 0.5	104.312	110.71	112.257	116.796	124.488	130.142	133.781
Uniform Sampling	p = 0.5	103.266	109.48	114.372	118.704	126.077	130.208	133.804

Table 5.2. MSE after sampling, Concrete

Method	Proportion	k = 3	k = 5	k = 7	k = 9	k = 11	k = 13	k = 15
Weighted Sampling	p = 0.05	11.696	14.393	16.87	16.533	16.744	17.484	20.918
Uniform Sampling	p = 0.05	9.551	13.827	14.737	16.424	16.964	17.18	18.57
Weighted Sampling	p = 0.1	9.506	9.329	8.866	10.963	13.902	15.632	14.662
Uniform Sampling	p = 0.1	9.017	10.18	9.294	10.615	13.572	14.496	15.335
Weighted Sampling	p = 0.25	7.304	7.831	7.973	7.423	7.618	7.251	7.19
Uniform Sampling	p = 0.25	7.74	7.639	7.638	7.39	7.096	7.144	7.571
Weighted Sampling	p = 0.5	4.824	6.024	6.696	6.927	7.416	7.166	7.272
Uniform Sampling	p = 0.5	5.013	5.801	6.576	6.915	7.21	7.33	7.316

Table 5.3. MSE after sampling, Energy (cool)

Method	Proportion	k = 3	k = 5	k = 7	k = 9	k = 11	k = 13	k = 15
Weighted Sampling	p = 0.05	16.252	15.237	19.461	18.323	24.563	22.69	25.765
Uniform Sampling	p = 0.05	15.557	19.266	19.041	20.939	19.284	20.196	23.548
Weighted Sampling	p = 0.1	11.796	11.077	13.626	14.54	14.844	16.746	18.042
Uniform Sampling	p = 0.1	11.709	12.487	13.572	15.636	14.639	17.241	18.475
Weighted Sampling	p = 0.25	9.414	9.895	10.433	9.959	10.076	10.354	10.583
Uniform Sampling	p = 0.25	9.39	9.593	9.94	10.093	10.117	10.087	10.706
Weighted Sampling	p = 0.5	6.265	7.783	8.646	8.904	9.112	9.378	9.427
Uniform Sampling	p = 0.5	6.197	7.577	8.611	8.976	9.413	9.414	9.483

Table 5.4. MSE after sampling, Energy (heat)

Method	Proportion	k = 3	k = 5	k = 7	k = 9	k = 11	k = 13	k = 15
Weighted Sampling	p = 0.05	24.815	20.098	27.914	28.814	32.277	41.112	43.515
Uniform Sampling	p = 0.05	20.42	21.512	23.271	25.521	30.625	37.848	43.228
Weighted Sampling	p = 0.1	22.087	19.018	19.911	22.966	19.648	22.02	21.014
Uniform Sampling	p = 0.1	24.298	20.535	19.312	19.194	20.292	21.296	21.358
Weighted Sampling	p = 0.25	21.878	18.055	18.905	17.878	17.505	17.211	17.7
Uniform Sampling	p = 0.25	21.715	18.192	18.244	17.111	17.12	17.479	17.892
Weighted Sampling	p = 0.5	19.052	18.024	17.713	17.339	17.609	17.44	17.612
Uniform Sampling	p = 0.5	19.573	18.188	17.178	17.976	17.877	17.405	17.508

Table 5.5. MSE after sampling, Auto MPG

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis introduces new methods of dataset characterization, methods of feature scaling, and data subsampling in the context of KNN regression and classification algorithms. Specific details of the implementation are theoretically justified. The new methods are implemented on various publicly available datasets and the results are compared with variants of KNN that are popular in the literature. The results from each section demonstrate the effectiveness of the ideas developed as a part of this research.

In this chapter, we conclude by mentioning the advantages and disadvantages of each new method, and potential directions for future work.

6.1 Dataset Characterization

The bias-variance tradeoff for any statistical model provides a framework to understand how different sets of parameters can lead to different levels of accuracy. Since KNN has low model bias, an empirical estimate of error due to variance in a dataset is a useful characteristic that could in-turn provide insights into how well KNN could perform when applied to a dataset for different values of k. The MNTE and MNTSD are estimates of error due to variance. These measures are easy to compute, have an intuitive explanation, and can be computed using different methods of sampling k-nearest neighborhoods. In this thesis, the MNTE and MNTSD are computed based on a randomized approach by using verification data from a dataset. However, the computation of these measures are not restricted to a randomized approach: they can be exhaustively calculated using powerful computers, and by using parallelization. Within a dataset, not all k-nearest neighborhoods are equally important, and thus, these measures may even be weighted based on how frequently a k-nearest neighborhood is queried.

On the other hand, there are certain disadvantages. As demonstrated in Chapter 3, the MNTSD and MNTE may not always be consistent. There could be special cases when their use could be uninformative. Hence, it is important to use them alongside other data characteristics in order to make well-informed guesses about what kinds of ML methods and model parameters could work best.

Thus, in the future, the MNTSD and MNTE should be investigated in various other contexts, and different methods of computing them should be studied.

6.2 Feature Scaling and Feature Extraction

Feature scaling approaches based on the minimization of MNTSD and MNTE are developed and introduced in Chapter 4. The motivation for using metaheuristic optimization approaches is explained in detail. The MNTSD and MNTE are used as objective functions within genetic algorithms and particle swarm optimization. The reason that independent feature weights were optimized was to keep the computational complexity low. The advantages of the methods introduced here are that they are simple and can provide insights into useful transformations of a dataset. In the future, these methods could be used to learn linear and non-linear transformations of a dataset.

The disadvantage is that these are slower than other popular algorithms and so in the future, efforts should be made to speed up execution using parallelism, or other kinds of metaheuristic approaches. Efforts should also be made to relax the black-box optimization problem based on approximate functions so that faster gradient-based algorithms can be used. Hybrid approaches combining gradient-based optimization and metaheuristics should also be studied.

6.3 Data Subsampling

A data subsampling approach based on the selection of a fixed number of random points that are weighted by their empirical likelihood of minimizing the MNTSD of a dataset is developed. This approach was used to draw smaller training sets using which KNN models were trained and tested. In this case, however, the results did not seem promising, with uniform subsampling resulting in similar error rates as our method of weighted subsampling.

This approach needs to be carefully studied in the future. There are a lot of aspects of this approach that can be potentially improved. For instance, determining the right number of iterations for which the weighting algorithm must be run, or using scaling based on MMNTSD along with the subsampling. Additionally, the distribution of the input space can be modeled and samples can initially be drawn based on a probability density function.

Since this is an iterative algorithm, for large datasets, learning the sample weights can take a long time. Approaches that can parallelize the computation of the MNTSD in each iteration could potentially accelerate this significantly. Also, in the method presented in this thesis, the sampling probabilities are for specific values of parameter k – for large datasets, recomputing sampling probabilities for different values of k could be very time consuming and thus, weighting for ranges of k should be explored.

6.4 Source Code

The source code for this research can be found on the author's GitHub page:

`https://github.com/SuryodayBasak`

The particular repository containing the code for feature scaling is:

`https://github.com/SuryodayBasak/mst-final-run`

The particular repository containing the code for data subsampling is:

`https://github.com/SuryodayBasak/mst-subsample-sel`

APPENDIX A

PROOF OF THEOREM 1: THE AVERAGE UNBIASED LEAVE-ONE-OUT VARIANCE IS THE SAME AS THE UNBIASED VARIANCE OF A SET

In this appendix, the proof of Theorem 1 is presented. This result holds true for single dimensional data.

Let μ refer to the mean of the entire set. The expression for any μ_i, without a loss of generality, is as follows:

$$\begin{aligned}
&\frac{x_1 + x_2 + ... + x_{n-1}}{n-1} = \mu_i \\
\Rightarrow &\sum_{i=1}^{n-1} x_i = (n-1)\mu_i \\
\Rightarrow &\sum_{i=1}^{n-1} x_i + x_n = (n-1)\mu_i + x_n \\
\Rightarrow &n \cdot \mu = (n-1)\mu_i + x_n \\
\Rightarrow &\mu = \frac{(n-1)\mu_i + x_n}{n} \\
\Rightarrow &\mu_i = \frac{n}{n-1} \cdot \mu - \frac{x_i}{n-1}
\end{aligned} \tag{A.1}$$

The expression of the ALOOV can be rewritten as:

$$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{n-2}\left[\Big(\sum_{j=1}^{n}(x_j - \mu_i)^2\Big) - (x_i - \mu_i)^2\right] = \frac{1}{n(n-2)}\left[\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}(x_j - \mu_i)^2\Big) - \Big(\sum_{i=1}^{n}(x_i - \mu_i)^2\Big)\right] \tag{A.2}$$

Substituting the expression of μ_i from Eq. A.1 in Eq. A.2, we get:

$$\frac{1}{n(n-2)}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\Big(x_j - \frac{n}{n-1}\cdot\mu + \frac{x_i}{n-1}\Big)^2 - \sum_{i=1}^{n}\Big(x_i - \frac{n}{n-1}\cdot\mu + \frac{x_i}{n-1}\Big)^2\right] \tag{A.3}$$

For simplicity, let:

$$
\begin{aligned}
T_1 &= \Big(x_j - \frac{n}{n-1}\cdot\mu + \frac{x_i}{n-1}\Big)^2 \\
T_2 &= \sum_{i=1}^{n}\sum_{j=1}^{n} T_1 \\
T_3 &= \sum_{i=1}^{n}\Big(x_i - \frac{n}{n-1}\cdot\mu + \frac{x_i}{n-1}\Big)^2
\end{aligned} \tag{A.4}
$$

Simplifying T_1,

$$
\begin{aligned}
T_1 &= \Big(x_j - \frac{n}{n-1}\cdot\mu + \frac{x_i}{n-1}\Big)^2 \\
&= \Big(\frac{(n-1)x_j - n\mu + x_i}{n-1}\Big)^2 \\
&= \frac{1}{(n-1)^2}(nx_j - x_j - n\mu + x_i)^2 \\
&= \frac{1}{(n-1)^2}\Big(n(x_j-\mu) + (x_i - x_j)\Big)^2 \\
&= \frac{1}{(n-1)^2}\Big(n^2(x_j-\mu)^2 + (x_i-x_j)^2 + 2n(x_j-\mu)(x_i-x_j)\Big)
\end{aligned} \tag{A.5}
$$

Substituting T_1 in T_2, we get:

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\frac{1}{(n-1)^2}\Big(n^2(x_j-\mu)^2 + (x_i-x_j)^2 + 2n(x_j-\mu)(x_i-x_j)\Big)\right] \\
=&\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\frac{n^2}{(n-1)^2}(x_j-\mu)^2 + \frac{1}{(n-1)^2}(x_i-x_j)^2 + \frac{2n}{(n-1)^2}(x_j-\mu)(x_i-x_j)\right] \\
=&\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{n^2(x_j-\mu)^2}{(n-1)^2} + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(x_i-x_j)^2}{(n-1)^2} + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{2n(x_j-\mu)(x_i-x_j)}{(n-1)^2}
\end{aligned} \tag{A.6}
$$

Simplifying T_3,

$$
\begin{aligned}
T_3 &= \sum_{i=1}^{n} \Big(x_i - \frac{n}{n-1} \cdot \mu + \frac{x_i}{n-1}\Big)^2 \\
&= \Big(\frac{(n-1)x_i + x_i}{n-1} - \frac{n\mu}{n-1}\Big)^2 \\
&= \Big(\frac{nx_i}{n-1} - \frac{n\mu}{n-1}\Big)^2 \\
&= \Big(\frac{n}{n-1}\Big)^2 (x_i - \mu)^2 \\
&= \frac{n^2\sigma^2}{n-1}
\end{aligned}
\tag{A.7}
$$

where σ^2 is the unbiased variance of the set. From Eq. A.6, let T_4, T_5 and T_6 be:

$$
\begin{aligned}
T_4 &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{n^2(x_j - \mu)^2}{(n-1)^2} \\
T_5 &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{(x_i - x_j)^2}{(n-1)^2} \\
T_6 &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{2n(x_j - \mu)(x_i - x_j)}{(n-1)^2}
\end{aligned}
\tag{A.8}
$$

Simplifying T_4,

$$
\begin{aligned}
T_4 &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{n^2(x_j - \mu)^2}{(n-1)^2} \\
&= \frac{n^3}{(n-1)^2} \sum_{j=1}^{n} (x_j - \mu)^2 \\
&= \frac{n^3}{n-1}\sigma^2
\end{aligned}
\tag{A.9}
$$

Simplifying T_5,

$$\begin{aligned}
T_5 &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(x_i - x_j)^2}{(n-1)^2} \\
&= \frac{1}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 \\
&= \frac{1}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i^2 - 2x_ix_j + x_j^2) \qquad \text{(A.10)} \\
&= \frac{1}{(n-1)^2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}x_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n}x_ix_j + \sum_{i=1}^{n}\sum_{j=1}^{n}x_j^2\Big) \\
&= \frac{1}{(n-1)^2}\Big(n\sum_{i=1}^{n}x_i^2 - 2\sum_{i=1}^{n}x_i\sum_{j=1}^{n}x_j + n\sum_{j=1}^{n}x_j^2\Big)
\end{aligned}$$

Since $n\sum_{i=1}^{n} x_i^2 = n\sum_{j=1}^{n} x_j^2$, with $x_i = x_j$ when $i = j$, we may replace the index j with i in $n\sum_{j=1}^{n} x_j^2$. We may also write $n\sum_{j=1}^{n} x_j = n\mu$. Therefore, upon making these substitutions, we get:

$$\begin{aligned}
T_5 &= \frac{1}{(n-1)^2}\Big(n\sum_{i=1}^{n}x_i^2 - 2\sum_{i=1}^{n}x_i \cdot n\mu + n\sum_{j=1}^{n}x_j^2\Big) \\
&= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n}x_i^2 - \sum_{i=1}^{n}\mu x_i\Big) \qquad \text{(A.11)} \\
&= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n}x_i^2 - 2\sum_{i=1}^{n}\mu x_i + \mu\sum_{i=1}^{n}x_i\Big)
\end{aligned}$$

Substituting $\sum_{i=1}^{n} x_i$ with $\sum_{i=1}^{n} \mu$, we get:

$$
\begin{aligned}
T_5 &= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} \mu x_i + \mu \sum_{i=1}^{n} \mu\Big) \\
&= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} \mu x_i + \sum_{i=1}^{n} \mu^2\Big) \\
&= \frac{2n}{(n-1)^2}\sum_{i=1}^{n}(x_i^2 - 2\mu x_i + \mu^2) \\
&= \frac{2n}{(n-1)^2}\sum_{i=1}^{n}(x_i - \mu)^2 \\
&= \frac{2n}{(n-1)}\sigma^2
\end{aligned}
\tag{A.12}
$$

Simplifying T_6,

$$
\begin{aligned}
T_6 &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{2n(x_j-\mu)(x_i-x_j)}{(n-1)^2} \\
&= \frac{2n}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_j-\mu)(x_i-x_j) \\
&= \frac{2n}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i x_j - x_j^2 - x_i\mu + x_j\mu) \\
&= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} x_j^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i\mu + \sum_{i=1}^{n}\sum_{j=1}^{n} x_j\mu\Big)
\end{aligned}
\tag{A.13}
$$

Note that $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i\mu = \sum_{i=1}^{n}\sum_{j=1}^{n} x_j\mu$, and cancel each other out. Therefore,

$$\begin{aligned}
T_6 &= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} x_j^2\Big)\\
&= \frac{2n}{(n-1)^2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j - n\sum_{j=1}^{n} x_j^2\Big)\\
&= -\frac{2n}{(n-1)^2}\Big(n\sum_{j=1}^{n} x_j^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j\Big)\\
&= -\frac{2n}{(n-1)^2}\Big(n\sum_{j=1}^{n} x_j^2 - \sum_{i=1}^{n} x_i\sum_{j=1}^{n} x_j\Big)\\
&= -\frac{2n}{(n-1)^2}\Big(n\sum_{j=1}^{n} x_j^2 - n\mu\sum_{j=1}^{n} x_j\Big)\\
&= -\frac{2n^2}{(n-1)^2}\Big(\sum_{j=1}^{n} x_j^2 - 2\mu\sum_{j=1}^{n} x_j + \mu\sum_{j=1}^{n} x_j\Big)\\
&= -\frac{2n^2}{(n-1)^2}\Big(\sum_{j=1}^{n} x_j^2 - 2\sum_{j=1}^{n} \mu x_j + \sum_{j=1}^{n} \mu^2\Big)\\
&= -\frac{2n^2}{(n-1)^2}\sum_{j=1}^{n}\Big(x_j^2 - 2\mu x_j + \mu^2\Big)\\
&= -\frac{2n^2}{(n-1)^2}\sum_{j=1}^{n}(x_j-\mu)^2\\
&= -\frac{2n^2}{(n-1)}\sigma^2
\end{aligned} \tag{A.14}$$

Substituting expressions of T_3, T_4, T_5 and T_6 in A.3, we get:

$$\begin{aligned}
&\frac{1}{n(n-2)}\left[\frac{n^3}{n-1}\sigma^2 + \frac{2n}{n-1}\sigma^2 - \frac{2n^2}{n-1}\sigma^2 - \frac{n^2\sigma^2}{n-1}\right]\\
=&\frac{\sigma^2}{n(n-1)(n-2)}(n^3 - 3n^2 + 2n)\\
=&\frac{\sigma^2}{n(n-1)(n-2)}\cdot n(n-1)(n-2)\\
=&\sigma^2
\end{aligned} \tag{A.15}$$

The above result is proven for a set of n points. When this is applied to a k-nearest neighborhood set, $k = n$, and the set of points it is applied to is the set of labels or target values of a k-nearest neighborhood.

APPENDIX B

RESULTS OF OPTIMAL FEATURE SCALING

B.1 Results of Regression

The results of regression are shown in Tables B.1 through B.8. The tables are exhaustive with regards to the results presented.

The tables present average MSE and the variance of MSE after 10-fold Monte-Carlo cross validation of every method studied, for different values of k. Below is a glossary of the terms used:

1. KNN: k-nearest neighbors without any preprocessing or feature extraction
2. D-KNN: inverse distance KNN
3. PCA-KNN: KNN with PCA for feature extraction
4. PCA-D-KNN: D-KNN with PCA for feature extraction
5. GA-KNN: KNN with feature weights optimized using GA and MMNTSD
6. GA-D-KNN: D-KNN with feature weights optimized using GA and MMNTSD
7. GPSO-KNN: KNN with feature weights optimized using GBest PSO and MMNTSD
8. GPSO-D-KNN: D-KNN with feature weights optimized using GBest PSO and MMNTSD
9. LPSO-KNN: KNN with feature weights optimized using LBest PSO and MMNTSD
10. LPSO-D-KNN: D-KNN with feature weights optimized using LBest PSO and MMNTSD

The entry in the tables in red indicates the method that resulted in the lowest average MSE; the entry in the tables in green indicates the method that resulted in the lowest MSE variance.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	41.18	36.71	35.3	34.97	34.53	34.19	33.92
	D-KNN	44.47	41.16	39.86	39.05	38.45	38.03	37.59
	PCA-KNN	43.5	38.88	36.77	35.98	35.71	35.14	34.71
	PCA-D-KNN	46.79	43.32	41.7	40.65	40.08	39.57	39.07
	GA-KNN	38.06	33.26	31.61	31.94	32.23	32.14	32.43
	GA-D-KNN	41.35	38.24	36.39	34.55	34.13	34.51	34.09
	GPSO-KNN	13.27	13.89	13.26	17.14	18.72	19.92	21.41
	GPSO-D-KNN	11.01	11.13	9.43	12.18	14.69	15.48	16.0
	LPSO-KNN	14.1	15.4	18.18	17.57	18.48	23.44	23.46
	LPSO-D-KNN	11.25	12.75	13.99	14.59	14.46	19.01	19.86
Variance of MSE	KNN	10.22	7.08	7.96	5.89	6.18	5.37	5.74
	D-KNN	9.8	7.43	6.2	5.49	4.38	4.58	4.33
	PCA-KNN	10.03	6.02	7.04	5.53	6.2	5.77	6.82
	PCA-D-KNN	8.87	6.92	5.98	5.33	4.38	4.48	4.38
	GA-KNN	11.98	5.3	3.12	2.11	3.45	2.74	4.83
	GA-D-KNN	9.26	7.23	4.4	58.93	66.9	45.94	50.5
	GPSO-KNN	6.06	3.7	12.29	38.8	24.0	11.29	7.32
	GPSO-D-KNN	8.16	3.6	7.67	14.41	33.0	21.14	17.71
	LPSO-KNN	17.13	5.95	32.48	16.12	23.06	11.13	12.87
	LPSO-D-KNN	10.37	2.35	20.8	16.41	19.72	22.0	25.84

Table B.1. Results of KNN Regression with all variants studied, on the Airfoil dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	84.17	91.69	97.95	100.01	104.74	107.85	111.21
	D-KNN	72.15	71.96	73.84	74.56	76.88	79.04	81.02
	PCA-KNN	91.92	103.17	112.0	118.7	125.56	129.68	134.04
	PCA-D-KNN	77.49	80.19	83.25	86.03	89.56	92.34	94.24
	GA-KNN	62.19	71.25	70.31	78.01	75.06	86.4	85.37
	GA-D-KNN	53.13	56.02	52.76	58.13	54.75	62.5	61.09
	GPSO-KNN	48.98	46.15	48.2	51.14	50.77	50.85	53.82
	GPSO-D-KNN	43.0	39.2	39.4	40.85	40.13	39.77	41.44
	LPSO-KNN	48.63	46.48	47.91	50.23	52.79	54.2	54.83
	LPSO-D-KNN	42.64	39.21	39.16	40.07	41.53	41.92	42.16
Variance of MSE	KNN	269.23	254.2	272.23	233.52	294.81	277.56	270.27
	D-KNN	248.8	222.92	239.2	211.19	240.84	240.42	219.58
	PCA-KNN	309.33	278.2	354.7	298.57	337.62	306.09	351.4
	PCA-D-KNN	273.71	299.93	323.07	284.0	295.08	274.38	274.93
	GA-KNN	88.59	170.87	129.77	205.69	133.24	157.09	184.99
	GA-D-KNN	94.08	180.1	103.76	184.99	128.64	129.6	132.85
	GPSO-KNN	54.85	34.5	45.07	49.94	36.44	15.57	24.38
	GPSO-D-KNN	50.15	59.22	59.67	53.41	50.54	34.8	44.61
	LPSO-KNN	49.04	32.86	44.88	35.54	84.13	40.19	40.9
	LPSO-D-KNN	64.02	49.67	67.5	49.11	97.78	51.06	59.26

Table B.2. Results of KNN Regression with all variants studied, on the Concrete Compressive Strength dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	3.91	3.61	4.85	5.75	6.22	6.35	6.51
	D-KNN	5.01	4.56	5.05	5.51	5.74	5.79	5.85
	PCA-KNN	8.53	6.88	6.03	6.31	6.77	7.07	7.29
	PCA-D-KNN	7.13	7.28	7.38	7.49	7.55	7.58	7.6
	GA-KNN	3.83	4.79	5.04	6.02	5.98	6.5	6.41
	GA-D-KNN	3.47	4.31	4.32	4.72	4.94	5.82	5.63
	GPSO-KNN	3.53	3.54	3.49	3.53	3.38	3.31	3.28
	GPSO-D-KNN	3.5	3.36	3.24	3.35	3.53	3.45	3.21
	LPSO-KNN	3.57	3.49	3.59	3.56	3.38	3.32	3.26
	LPSO-D-KNN	3.49	3.28	3.33	3.29	3.4	3.38	3.27
Variance of MSE	KNN	0.34	0.46	0.69	0.72	0.75	0.82	0.91
	D-KNN	0.26	0.37	0.5	0.52	0.58	0.61	0.64
	PCA-KNN	0.79	0.64	0.73	0.85	0.81	0.84	1.07
	PCA-D-KNN	0.5	0.49	0.51	0.49	0.55	0.52	0.52
	GA-KNN	0.36	1.8	2.56	1.69	0.78	1.0	0.73
	GA-D-KNN	0.33	0.8	1.56	0.68	1.13	0.65	0.48
	GPSO-KNN	0.32	0.32	0.3	0.19	0.2	0.19	0.22
	GPSO-D-KNN	0.38	0.44	0.26	0.39	0.35	0.39	0.3
	LPSO-KNN	0.33	0.42	0.3	0.19	0.22	0.17	0.22
	LPSO-D-KNN	0.33	0.53	0.27	0.33	0.41	0.38	0.41

Table B.3. Results of KNN Regression with all variants studied, on the Energy (cool) dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	5.07	4.85	6.24	7.34	7.81	7.93	7.96
	D-KNN	7.7	7.3	7.85	8.46	8.73	8.79	8.77
	PCA-KNN	12.18	9.69	8.09	8.27	8.77	9.06	9.1
	PCA-D-KNN	10.94	11.61	11.84	12.03	12.16	12.3	12.39
	GA-KNN	2.04	3.91	4.59	6.94	6.36	6.22	6.21
	GA-D-KNN	1.7	2.85	1.8	3.95	2.32	3.78	4.02
	GPSO-KNN	0.86	1.03	1.28	1.51	1.71	1.86	2.06
	GPSO-D-KNN	0.63	0.57	0.37	0.4	0.49	0.59	0.58
	LPSO-KNN	0.97	1.02	1.27	1.51	1.68	1.88	2.06
	LPSO-D-KNN	0.71	0.57	0.39	0.39	0.67	0.66	0.59
Variance of MSE	KNN	0.95	1.43	1.65	1.36	2.15	2.41	2.49
	D-KNN	1.01	1.49	1.7	1.5	2.01	2.27	2.34
	PCA-KNN	3.0	4.22	2.13	1.53	2.18	2.66	2.8
	PCA-D-KNN	1.04	1.68	1.82	1.83	2.16	2.42	2.52
	GA-KNN	2.13	3.22	3.56	1.52	1.44	1.79	4.48
	GA-D-KNN	3.18	8.75	1.9	5.96	2.66	5.01	9.64
	GPSO-KNN	0.35	0.14	0.22	0.3	0.36	0.44	0.45
	GPSO-D-KNN	0.32	0.08	0.03	0.02	0.09	0.11	0.06
	LPSO-KNN	0.41	0.14	0.22	0.3	0.35	0.43	0.46
	LPSO-D-KNN	0.3	0.09	0.03	0.05	0.17	0.08	0.07

Table B.4. Results of KNN Regression with all variants studied, on the Energy (heat) dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	1.77	1.83	1.87	1.96	1.97	2.0	2.04
	D-KNN	1.71	1.64	1.64	1.67	1.66	1.66	1.67
	PCA-KNN	1.76	1.82	1.88	1.97	1.97	2.0	2.06
	PCA-D-KNN	1.74	1.69	1.68	1.71	1.69	1.69	1.7
	GA-KNN	1.68	1.55	1.62	1.66	1.64	1.66	1.7
	GA-D-KNN	1.64	1.46	1.46	1.46	1.43	1.43	1.45
	GPSO-KNN	1.44	1.45	1.32	1.36	1.36	1.41	1.42
	GPSO-D-KNN	1.41	1.38	1.24*	1.25	1.25	1.29	1.29
	LPSO-KNN	1.58	1.42	1.34	1.39	1.38	1.37	1.4
	LPSO-D-KNN	1.52	1.34	1.26	1.26	1.26	1.25	1.27
Variance of MSE	KNN	0.091	0.0921	0.1065	0.1075	0.0955	0.0875	0.0898
	D-KNN	0.0819	0.0726	0.0835	0.0806	0.0732	0.0707	0.0769
	PCA-KNN	0.1003	0.0984	0.1041	0.0998	0.0927	0.0847	0.0872
	PCA-D-KNN	0.0856	0.0784	0.0846	0.0782	0.0721	0.0679	0.0718
	GA-KNN	0.0711	0.0453	0.0588	0.0669	0.0503	0.0309	0.0574
	GA-D-KNN	0.0557	0.0271	0.052	0.0464	0.0397	0.0352	0.0549
	GPSO-KNN	0.0516	0.0279	0.0073	0.0155	0.0144	0.0137	0.0166
	GPSO-D-KNN	0.0307	0.0226	0.0072*	0.0205	0.0132	0.0149	0.0153
	LPSO-KNN	0.0658	0.0444	0.0087	0.0112	0.0141	0.0168	0.0123
	LPSO-D-KNN	0.0688	0.0393	0.0138	0.0116	0.0133	0.0173	0.0181

Table B.5. Results of KNN Regression with all variants studied, on the Qsar Aqua toxicity dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	0.9908	0.9613	0.9388	0.9499	0.9614	0.9572	0.9634
	D-KNN	0.9757	0.942	0.9149	0.9176	0.9187	0.9135	0.9176
	PCA-KNN	1.2511	1.2009	1.1769	1.1555	1.1502	1.1472	1.155
	PCA-D-KNN	1.2329	1.158	1.1273	1.0986	1.0853	1.0755	1.0772
	GA-KNN	0.964	0.903	0.9091	0.9033	0.8955	0.8972	0.9039
	GA-D-KNN	0.9523	0.8943	0.8877	0.8802	0.8701	0.871	0.874
	GPSO-KNN	1.035	0.9273	0.8826	0.8828	0.8941	0.8653	0.8977
	GPSO-D-KNN	1.0201	0.9118	0.8707	0.8619	0.8681	0.844	0.8639
	LPSO-KNN	0.9968	0.8993	0.8661	0.8617	0.9054	0.8957	0.8995
	LPSO-D-KNN	0.9877	0.8802	0.8528	0.8418	0.8774	0.8713	0.8702
Variance of MSE	KNN	0.0194	0.0169	0.0111	0.0106	0.0083	0.0103	0.0099
	D-KNN	0.0163	0.0144	0.0103	0.0098	0.0077	0.0093	0.0086
	PCA-KNN	0.0069	0.0082	0.0076	0.0077	0.009	0.0109	0.0115
	PCA-D-KNN	0.0082	0.0068	0.0062	0.006	0.0067	0.0085	0.0095
	GA-KNN	0.0212	0.015	0.0038	0.0119	0.0091	0.0118	0.0089
	GA-D-KNN	0.0168	0.0142	0.004	0.0088	0.0072	0.0089	0.0068
	GPSO-KNN	0.025	0.0172	0.0091	0.0078	0.0036	0.005	0.0068
	GPSO-D-KNN	0.0316	0.0172	0.01	0.0061	0.0051	0.0052	0.005
	LPSO-KNN	0.0265	0.0107	0.0118	0.0086	0.0051	0.008	0.0099
	LPSO-D-KNN	0.025	0.009	0.0098	0.0079	0.0062	0.0064	0.0068

Table B.6. Results of KNN Regression with all variants studied, on the Qsar Fish toxicity dataset.

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	0.4742	0.4976	0.5042	0.5209	0.5283	0.5344	0.5438
	D-KNN	0.3664	0.3421	0.3239	0.3223	0.3177	0.3161	0.3185
	PCA-KNN	0.4735	0.4988	0.5065	0.5217	0.5304	0.5366	0.5448
	PCA-D-KNN	0.363	0.3413	0.3266	0.3228	0.3188	0.3171	0.3191
	GA-KNN	0.4225	0.4344	0.458	0.4613	0.4468	0.4454	0.4593
	GA-D-KNN	0.318	0.292	0.2936	0.285	0.2657	0.2644	0.2686
	GPSO-KNN	0.3979	0.424	0.4116	0.412	0.4223	0.4218	0.4203
	GPSO-D-KNN	0.2969	0.2912	0.2651	0.2585	0.263	0.255	0.2516
	LPSO-KNN	0.3934	0.4092	0.4151	0.4205	0.4113	0.4196	0.4199
	LPSO-D-KNN	0.3009	0.2771	0.2652	0.2675	0.2489	0.2497	0.2503
Variance of MSE	KNN	0.0034	0.0021	0.0018	0.002	0.0021	0.0023	0.0027
	D-KNN	0.0029	0.0021	0.0019	0.0022	0.0022	0.0023	0.0026
	PCA-KNN	0.003	0.002	0.002	0.002	0.0022	0.0022	0.0025
	PCA-D-KNN	0.0028	0.0022	0.002	0.0021	0.0022	0.0023	0.0025
	GA-KNN	0.0018	0.0016	0.002	0.0019	0.0010	0.0019	0.0013
	GA-D-KNN	0.002	0.0017	0.0016	0.0013	0.0013	0.0022	0.0013
	GPSO-KNN	0.0021	0.0029	0.0015	0.0019	0.0026	0.0019	0.0018
	GPSO-D-KNN	0.0023	0.0024	0.0014	0.0018	0.0019	0.0017	0.0015
	LPSO-KNN	0.0019	0.0036	0.0009	0.0028	0.0016	0.0014	0.0015
	LPSO-D-KNN	0.0017	0.0023	0.0012	0.0022	0.0014	0.0017	0.0012

Table B.7. Results of KNN Regression with all variants studied, on the Wine (red).

Performance Measure	Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
Average MSE	KNN	105.67	154.82	187.67	184.82	182.04	177.4	182.91
	D-KNN	75.15	95.71	107.89	108.61	110.12	113.03	117.81
	PCA-KNN	106.35	154.37	187.75	185.46	182.41	177.59	182.98
	PCA-D-KNN	75.44	95.55	107.92	108.69	110.19	113.13	117.89
	GA-KNN	69.57	62.84	73.84	100.23	114.98	109.36	116.04
	GA-D-KNN	50.79	46.42	52.85	67.2	74.0	74.1	81.6
	GPSO-KNN	2.32	2.82	2.91	2.48	2.78	3.42	2.99
	GPSO-D-KNN	2.17	2.25	2.5	2.18	2.51	2.78	2.57
	LPSO-KNN	3.44	2.9	2.73	4.43	3.16	3.35	3.11
	LPSO-D-KNN	3.21	2.58	2.66	2.47	2.97	2.97	2.79
Variance of MSE	KNN	1425.03	2224.76	2353.58	1721.45	1895.11	2030.05	2212.22
	D-KNN	951.42	1248.28	1276.27	1153.95	1211.78	1281.12	1370.78
	PCA-KNN	1387.97	2189.28	2328.49	1849.34	1869.54	2004.38	2198.82
	PCA-D-KNN	936.41	1243.8	1270.83	1170.86	1206.29	1275.44	1369.21
	GA-KNN	1341.5	381.39	948.46	1450.76	1392.45	1253.77	1499.26
	GA-D-KNN	722.24	235.11	426.13	725.67	699.8	559.32	774.96
	GPSO-KNN	2.02	3.48	2.21	1.65	1.35	8.4	2.03
	GPSO-D-KNN	1.42	1.08	1.04	1.19	1.22	4.3	1.42
	LPSO-KNN	8.49	3.07	2.01	23.66	2.1	3.94	1.88
	LPSO-D-KNN	4.74	1.7	1.71	1.13	2.0	2.77	1.31

Table B.8. Results of KNN Regression with all variants studied, on the Yacht.

B.2 Results of Classification

The results of classification are shown in Tables B.9 through B.26. The tables are exhaustive with regards to the results presented.

The tables present average accuracy and the variance of accuracy after 10-fold Monte-Carlo cross validation of every method studied, for different values of k. Below is a glossary of the terms used:

1. KNN: k-nearest neighbors without any preprocessing or feature extraction
2. D-KNN: inverse distance KNN
3. PCA-KNN: KNN with PCA for feature extraction
4. PCA-D-KNN: D-KNN with PCA for feature extraction
5. GA-KNN (MNTE): KNN with feature weights optimized using GA and MMNTE
6. GA-D-KNN (MNTE): D-KNN with feature weights optimized using GA and MMNTE
7. GA-KNN (MNTSD): KNN with feature weights optimized using GA and MMNTSD
8. GA-D-KNN (MNTSD): D-KNN with feature weights optimized using GA and MMNTSD
9. GPSO-KNN (MNTE): KNN with feature weights optimized using GBest PSO and MMNTE
10. GPSO-D-KNN (MNTE): D-KNN with feature weights optimized using GBest PSO and MMNTE
11. GPSO-KNN (MNTSD): KNN with feature weights optimized using GBest PSO and MMNTSD
12. GPSO-D-KNN (MNTSD): D-KNN with feature weights optimized using GBest PSO and MMNTSD

13. LPSO-KNN (MNTE): KNN with feature weights optimized using LBest PSO and MMNTE
14. LPSO-D-KNN (MNTE): D-KNN with feature weights optimized using LBest PSO and MMNTE
15. LPSO-KNN (MNTSD): KNN with feature weights optimized using LBest PSO and MMNTSD
16. LPSO-D-KNN (MNTSD): D-KNN with feature weights optimized using LBest PSO and MMNTSD

The entry in the tables in red indicates the method that resulted in the highest average accuracy; the entry in the tables in green indicates the method that resulted in the lowest variance in accuracy.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	72.2667	72.1333	73.1333	74.8667	74.0	74.4	74.4667
D-KNN	63.8667	56.9333	54.0	52.0667	50.4	48.9333	47.1333
PCA-KNN	71.9333	71.9333	73.4	73.2667	73.8	74.6667	74.6667
PCA-D-KNN	61.8667	56.6	54.6	52.1333	50.0	48.4667	47.4
GA-KNN (MNTE)	71.6	73.0667	74.4	74.5333	74.1333	74.4	74.6667
GA-KNN (MNTSD)	72.2	73.2	74.0667	75.0	73.9333	74.6667	74.8
GA-D-KNN (MNTE)	63.8	58.0667	55.8667	54.6	52.4667	50.4	48.9333
GA-D-KNN (MNTSD)	64.5333	58.2667	55.5333	54.4667	52.2	50.0667	49.4
GPSO-KNN (MNTE)	72.2	74.1333	75.5333	75.0667	75.4667	75.3333	75.0667
GPSO-KNN (MNTSD)	72.0	73.2	75.9333	74.7333	76.0	75.5333	75.3333
GPSO-D-KNN (MNTE)	63.0667	60.0667	56.1333	53.8	52.6	51.4667	50.2667
GPSO-D-KNN (MNTSD)	63.8	60.1333	57.2	54.2	52.0	50.8667	49.8
LPSO-KNN (MNTE)	72.4667	74.2	75.6	75.2	75.4	75.6	76.1333
LPSO-KNN (MNTSD)	72.6667	74.2	74.6667	75.0	75.6667	76.2	75.0
LPSO-D-KNN (MNTE)	64.6	58.9333	56.4667	53.8	52.1333	51.6	50.0667
LPSO-D-KNN (MNTSD)	64.7333	60.2	56.8	54.0667	51.9333	50.4	50.6667

Table B.9. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Blood Transfusion dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.1217	0.0996	0.085	0.0978	0.0533	0.0446	0.0386
D-KNN	0.0561	0.0743	0.0385	0.0953	0.1325	0.1177	0.1314
PCA-KNN	0.0725	0.0834	0.0933	0.0696	0.0613	0.0563	0.0356
PCA-D-KNN	0.1065	0.112	0.0923	0.1549	0.1452	0.1591	0.1525
GA-KNN (MNTE)	0.0901	0.0604	0.0437	0.0808	0.062	0.0456	0.0602
GA-KNN (MNTSD)	0.1008	0.0788	0.0725	0.0615	0.0656	0.0642	0.0581
GA-D-KNN (MNTE)	0.0731	0.1111	0.0926	0.1328	0.1719	0.1385	0.1197
GA-D-KNN (MNTSD)	0.0996	0.1325	0.0929	0.1492	0.1492	0.1674	0.118
GPSO-KNN (MNTE)	0.1304	0.1183	0.0959	0.0555	0.0571	0.0869	0.1424
GPSO-KNN (MNTSD)	0.0711	0.148	0.0617	0.0992	0.077	0.0642	0.0711
GPSO-D-KNN (MNTE)	0.096	0.0656	0.1687	0.1729	0.0962	0.1055	0.1069
GPSO-D-KNN (MNTSD)	0.1028	0.1381	0.1341	0.2628	0.1551	0.1512	0.1057
LPSO-KNN (MNTE)	0.1097	0.1442	0.1622	0.14	0.1061	0.098	0.0966
LPSO-KNN (MNTSD)	0.2035	0.1117	0.162	0.1336	0.0467	0.1156	0.0704
LPSO-D-KNN (MNTE)	0.0686	0.179	0.1117	0.1709	0.1252	0.097	0.0903
LPSO-D-KNN (MNTSD)	0.0725	0.1492	0.1242	0.1555	0.1802	0.1849	0.2311

Table B.10. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Blood Transfusion dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	97.7372	97.9562*	97.0803	96.8613	96.8613	96.9343	97.2263
D-KNN	97.5182	97.5912	97.2993	96.9343	96.6423	96.3504	96.2774
PCA-KNN	97.6642	97.8102	97.9562	97.8102	97.5182	97.5912	97.5912
PCA-D-KNN	97.6642	97.2263	96.5693	96.4964	96.2044	96.1314	96.1314
GA-KNN (MNTE)	97.5182	97.5182	97.3723	97.3723	97.0073	97.3723	96.6423
GA-KNN (MNTSD)	97.3723	97.3723	97.0803	97.2263	96.9343	97.0803	97.0073
GA-D-KNN (MNTE)	97.8102	97.1533	96.2774	96.1314	96.3504	95.9854	95.7664
GA-D-KNN (MNTSD)	97.5182	96.7883	96.7153	96.0584	96.1314	95.9854	95.9854
GPSO-KNN (MNTE)	96.4234	96.3504	96.7883	97.0073	96.8613	96.6423	96.2774
GPSO-KNN (MNTSD)	96.8613	96.8613	96.8613	96.7153	96.2044	96.7153	96.6423
GPSO-D-KNN (MNTE)	96.7153	95.8394	95.9124	95.7664	95.0365	94.6715	95.0365
GPSO-D-KNN (MNTSD)	96.7883	96.2774	95.9124	95.8394	95.7664	95.6204	94.8905
LPSO-KNN (MNTE)	96.7883	96.4234	96.4234	96.7153	95.7664	96.4964	96.2044
LPSO-KNN (MNTSD)	96.4964	96.6423	96.8613	96.4234	96.7883	96.0584	95.9854
LPSO-D-KNN (MNTE)	96.8613	96.3504	95.6204	95.3285	94.8175	94.8175	94.8175
LPSO-D-KNN (MNTSD)	96.3504	96.5693	95.9124	95.6204	95.4745	95.3285	94.5985

Table B.11. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Breast Cancer dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.0171	0.0057*	0.0118	0.0155	0.0202	0.0294	0.0211
D-KNN	0.0133	0.0166	0.0131	0.014	0.0144	0.0083	0.01
PCA-KNN	0.0234	0.0154	0.0092	0.0118	0.0133	0.0131	0.0131
PCA-D-KNN	0.0175	0.014	0.0155	0.0152	0.0199	0.0214	0.0214
GA-KNN (MNTE)	0.0097	0.0239	0.0156	0.018	0.01	0.0144	0.0085
GA-KNN (MNTSD)	0.0133	0.0133	0.0107	0.0128	0.0092	0.0095	0.0183
GA-D-KNN (MNTE)	0.0154	0.0088	0.0065	0.0095	0.013	0.0157	0.0116
GA-D-KNN (MNTSD)	0.0062	0.0097	0.0062	0.0121	0.0119	0.0169	0.0157
GPSO-KNN (MNTE)	0.0254	0.013	0.0133	0.0195	0.0214	0.0215	0.0337
GPSO-KNN (MNTSD)	0.0166	0.0261	0.0178	0.011	0.0092	0.0228	0.0263
GPSO-D-KNN (MNTE)	0.0062	0.0166	0.0121	0.0081	0.0211	0.0261	0.0092
GPSO-D-KNN (MNTSD)	0.0121	0.0171	0.0073	0.0415	0.0104	0.0107	0.0059
LPSO-KNN (MNTE)	0.0156	0.0136	0.0195	0.0228	0.0412	0.0388	0.0199
LPSO-KNN (MNTSD)	0.0282	0.018	0.0119	0.023	0.0121	0.0251	0.0157
LPSO-D-KNN (MNTE)	0.0083	0.0154	0.0213	0.0109	0.0076	0.0112	0.0112
LPSO-D-KNN (MNTSD)	0.0083	0.019	0.0168	0.0095	0.0199	0.0109	0.0204

Table B.12. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Breast Cancer dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	90.2582	89.7887	89.3192	88.8498	88.0282	87.6761	87.6761
D-KNN	87.9108	82.7465	79.4601	75.3521	72.4178	70.4225	68.1925
PCA-KNN	90.2582	89.6714	88.8498	88.9671	87.9108	87.5587	87.5587
PCA-D-KNN	88.1455	82.8638	79.2254	75.3521	72.3005	69.7183	67.9577
GA-KNN (MNTE)	90.9624	89.9061	90.2582	89.554	90.3756	88.8498	89.2019
GA-KNN (MNTSD)	90.3756	90.1408	89.9061	89.7887	89.7887	88.7324	88.3803
GA-D-KNN (MNTE)	88.8498	84.6244	81.338	78.2864	75.1174	72.6526	72.4178
GA-D-KNN (MNTSD)	87.6761	83.4507	78.2864	76.9953	74.5305	72.4178	70.7746
GPSO-KNN (MNTE)	90.2582	92.3709	92.4883	90.2582	91.3146	91.4319	90.6103
GPSO-KNN (MNTSD)	90.493	90.9624	90.6103	89.7887	90.0235	90.7277	90.0235
GPSO-D-KNN (MNTE)	88.1455	88.2629	85.5634	80.3991	80.5164	78.9906	76.4085
GPSO-D-KNN (MNTSD)	88.3803	85.3286	83.216	80.8685	75.5869	77.4648	73.9437
LPSO-KNN (MNTE)	91.0798	91.6667	91.784	91.784	90.9624	90.6103	90.0235
LPSO-KNN (MNTSD)	90.6103	91.1972	92.2535	91.784	90.3756	90.8451	91.5493
LPSO-D-KNN (MNTE)	88.0282	87.0892	83.5681	83.9202	79.2254	75.8216	73.7089
LPSO-D-KNN (MNTSD)	87.4413	87.4413	83.0986	83.5681	79.5775	77.23	75.1174

Table B.13. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Cardioctography dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.0001	0.0001	0.0034	0.0012	0.005	0.0068	0.0068
D-KNN	0.0068	0.0001	0.0001	0.0138	0.0068	0.0138	0.0233
PCA-KNN	0.0001	0.0006	0.0034	0.0022	0.0068	0.0088	0.005
PCA-D-KNN	0.0012	0.0006	0.0001	0.0088	0.0198	0.005	0.0167
GA-KNN (MNTE)	0.0167	0.027	0.0012	0.0034	0.0022	0.0068	0.0042
GA-KNN (MNTSD)	0.0138	0.0006	0.0006	0.0167	0.0001	0.0353	0.0112
GA-D-KNN (MNTE)	0.0112	0.0012	0.0167	0.0233	0.0138	0.0398	0.0167
GA-D-KNN (MNTSD)	0.0068	0.0012	0.0233	0.0551	0.0112	0.0233	0.0608
GPSO-KNN (MNTE)	0.0167	0.0034	0.0138	0.0112	0.0138	0.0112	0.027
GPSO-KNN (MNTSD)	0.0034	0.0012	0.0022	0.0012	0.0001	0.0034	0.0001
GPSO-D-KNN (MNTE)	0.0233	0.0551	0.0233	0.031	0.3174	0.0167	0.0729
GPSO-D-KNN (MNTSD)	0.0233	0.0001	0.0112	0.031	0.027	0.0353	0.0551
LPSO-KNN (MNTE)	0.0022	0.0012	0.0022	0.0006	0.0012	0.005	0.0012
LPSO-KNN (MNTSD)	0.0006	0.0012	0.0031	0.0012	0.0014	0.0030	0.0327
LPSO-D-KNN (MNTE)	0.005	0.0667	0.0138	0.0034	0.0497	0.0667	0.005
LPSO-D-KNN (MNTSD)	0.0012	0.0312	0.0271	0.0328	0.0555	0.0345	0.0618

Table B.14. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Cardioctography dataset. Multiple cases result in very low variance of accuracy.

Method	$k=3$	$k=5$	$k=7$	$k=9$	$k=11$	$k=13$	$k=15$
KNN	55.0	53.75	59.38	60.62	60.0	60.62	60.62
D-KNN	53.75	53.12	53.75	53.75	53.75	53.75	53.75
PCA-KNN	56.25	60.0	57.5	58.75	59.38	60.62	55.62
PCA-D-KNN	55.0	55.62	53.75	53.75	53.75	53.75	53.75
GA-KNN (MNTE)	58.13	59.38	60.0	63.75	63.75	60.62	57.5
GA-KNN (MNTSD)	58.75	60.0	63.12	61.25	62.5	60.0	57.5
GA-D-KNN (MNTE)	50.0	52.5	53.12	53.75	53.75	53.75	53.75
GA-D-KNN (MNTSD)	51.25	51.25	53.75	53.75	53.75	53.75	53.75
GPSO-KNN (MNTE)	58.13	60.62	58.13	59.38	55.62	54.37	57.5
GPSO-KNN (MNTSD)	59.38	60.62	58.13	58.75	56.88	56.25	56.88
GPSO-D-KNN (MNTE)	51.25	53.75	53.12	53.12	53.75	53.75	53.75
GPSO-D-KNN (MNTSD)	49.38	50.62	52.5	51.88	53.75	53.75	53.75
LPSO-KNN (MNTE)	54.37	58.13	60.62	63.12	55.62	56.88	58.13
LPSO-KNN (MNTSD)	58.13	59.38	58.13	62.5	57.5	53.12	58.13
LPSO-D-KNN (MNTE)	51.25	51.88	52.5	53.75	53.75	53.75	53.75
LPSO-D-KNN (MNTSD)	50.62	52.5	52.5	52.5	53.75	53.75	53.75

Table B.15. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the C-Section dataset.

Method	$k=3$	$k=5$	$k=7$	$k=9$	$k=11$	$k=13$	$k=15$
KNN	0.76	1.32	1.15	0.61	0.36	0.44	0.79
D-KNN	0.8	1.06	0.89	0.89	0.89	0.89	0.89
PCA-KNN	1.3	1.06	0.68	0.54	0.37	0.61	1.43
PCA-D-KNN	0.85	0.99	0.89	0.89	0.89	0.89	0.89
GA-KNN (MNTE)	2.17	1.06	1.67	1.37	1.28	0.53	1.63
GA-KNN (MNTSD)	1.93	1.06	1.78	1.2	1.39	2.1	1.72
GA-D-KNN (MNTE)	0.69	0.62	0.72	0.89	0.89	0.89	0.89
GA-D-KNN (MNTSD)	0.42	0.68	0.89	0.89	0.89	0.89	0.89
GPSO-KNN (MNTE)	2.17	0.61	1.31	1.93	1.08	0.87	1.2
GPSO-KNN (MNTSD)	1.93	1.22	1.31	2.27	1.08	0.95	1.43
GPSO-D-KNN (MNTE)	0.68	0.97	0.72	0.72	0.89	0.89	0.89
GPSO-D-KNN (MNTSD)	0.3	0.65	0.71	0.61	0.89	0.89	0.89
LPSO-KNN (MNTE)	1.57	0.79	1.48	1.6	0.91	1.25	1.22
LPSO-KNN (MNTSD)	1.65	0.98	1.48	1.48	1.46	1.32	1.57
LPSO-D-KNN (MNTE)	1.11	0.87	0.62	0.89	0.89	0.89	0.89
LPSO-D-KNN (MNTSD)	0.73	0.97	0.71	0.62	0.89	0.89	0.89

Table B.16. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the C-Section dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	62.5974	65.6277	67.1861	66.0173	66.1472	66.0606	66.8831
D-KNN	58.8312	55.1082	55.4113	54.8918	54.1558	53.5065	53.5065
PCA-KNN	62.7706	65.5844	67.0563	66.2771	66.3203	66.1472	66.7965
PCA-D-KNN	58.8312	55.1515	55.3247	54.8918	54.1991	53.5065	53.5065
GA-KNN (MNTE)	64.7186	66.9264	66.7532	67.0996	67.3593	66.9697	67.2294
GA-KNN (MNTSD)	64.8485	65.7143	68.0952	66.7965	67.316	68.4848	66.9697
GA-D-KNN (MNTE)	58.3117	55.8442	55.4978	54.7619	54.1991	53.7662	53.6797
GA-D-KNN (MNTSD)	58.1818	55.7576	55.2814	54.5455	53.9827	53.6797	53.7229
GPSO-KNN (MNTE)	65.4545	67.5758	66.4502	68.0952	67.0996	67.5758	66.4935
GPSO-KNN (MNTSD)	66.1039	66.8398	66.8831	67.013	67.8355	66.8831	67.316
GPSO-D-KNN (MNTE)	60.1732	56.9264	54.8485	54.4156	53.9394	53.5931	53.29
GPSO-D-KNN (MNTSD)	60.4329	57.4026	55.0649	54.3723	53.7229	53.3766	53.3766
LPSO-KNN (MNTE)	65.5844	66.1472	67.0563	65.7576	68.2684	66.3203	67.2294
LPSO-KNN (MNTSD)	65.1082	66.4502	66.2338	66.4935	68.5714	66.4069	66.9697
LPSO-D-KNN (MNTE)	59.4372	57.6623	55.1515	54.4156	53.7229	53.7229	53.4199
LPSO-D-KNN (MNTSD)	59.3506	57.1861	55.4545	54.1991	53.8961	53.29	53.5065

Table B.17. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Diabetic Retinopathy dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.0313	0.0471	0.0445	0.0526	0.0815	0.0359	0.038
D-KNN	0.1014	0.0925	0.0866	0.0753	0.0631	0.0696	0.0696
PCA-KNN	0.0296	0.0459	0.046	0.0593	0.0774	0.0349	0.0313
PCA-D-KNN	0.0935	0.0913	0.0907	0.0753	0.0636	0.0696	0.0696
GA-KNN (MNTE)	0.048	0.0601	0.0728	0.0479	0.0238	0.1379	0.0625
GA-KNN (MNTSD)	0.0561	0.0266	0.035	0.0542	0.0505	0.0212	0.0525
GA-D-KNN (MNTE)	0.1341	0.0883	0.0969	0.0771	0.0736	0.0636	0.0612
GA-D-KNN (MNTSD)	0.0846	0.1011	0.0783	0.0925	0.0704	0.0695	0.0631
GPSO-KNN (MNTE)	0.0845	0.0318	0.0288	0.1016	0.0504	0.0793	0.1554
GPSO-KNN (MNTSD)	0.0533	0.0388	0.0288	0.0953	0.0596	0.1159	0.1171
GPSO-D-KNN (MNTE)	0.1403	0.1013	0.1129	0.0483	0.065	0.0649	0.0606
GPSO-D-KNN (MNTSD)	0.1792	0.1425	0.1032	0.0775	0.0627	0.0558	0.0612
LPSO-KNN (MNTE)	0.0992	0.0449	0.0318	0.0485	0.0517	0.0886	0.0746
LPSO-KNN (MNTSD)	0.0471	0.0938	0.0879	0.0555	0.0734	0.0526	0.0408
LPSO-D-KNN (MNTE)	0.1245	0.1648	0.073	0.0779	0.0618	0.0585	0.0617
LPSO-D-KNN (MNTSD)	0.1222	0.1193	0.1384	0.0707	0.0622	0.0652	0.0671

Table B.18. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Diabetic Retinopathy dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	70.97	68.71	71.13	73.06	73.87	73.06	72.26
D-KNN	57.9	52.42	48.06	45.0	41.94	38.55	35.97
PCA-KNN	72.58	74.19	73.23	73.87	73.71	73.06	73.06
PCA-D-KNN	57.74	50.65	45.97	43.39	40.0	37.26	35.48
GA-KNN (MNTE)	71.77	71.61	71.94	73.23	71.29	71.45	71.61
GA-KNN (MNTSD)	71.45	70.97	72.42	73.06	71.13	71.61	72.26
GA-D-KNN (MNTE)	58.39	51.29	47.1	43.06	41.61	38.06	37.1
GA-D-KNN (MNTSD)	58.23	52.42	45.81	43.23	40.65	37.9	36.94
GPSO-KNN (MNTE)	69.19	71.29	69.35	71.45	71.29	72.1	70.97
GPSO-KNN (MNTSD)	69.35	70.97	71.13	73.23	71.61	72.42	70.48
GPSO-D-KNN (MNTE)	59.03	53.23	48.87	45.0	40.81	39.19	38.06
GPSO-D-KNN (MNTSD)	59.52	52.26	47.26	44.19	43.23	39.84	38.23
LPSO-KNN (MNTE)	69.68	70.0	68.71	72.26	70.65	72.1	70.97
LPSO-KNN (MNTSD)	69.03	70.16	70.0	71.45	70.81	71.94	71.29
LPSO-D-KNN (MNTE)	58.87	52.58	47.9	45.0	41.61	39.84	37.9
LPSO-D-KNN (MNTSD)	58.06	53.71	47.42	44.68	41.77	39.52	37.9

Table B.19. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Haberman dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.13	0.3	0.23	0.16	0.21	0.17	0.05
D-KNN	0.33	0.22	0.3	0.25	0.32	0.33	0.36
PCA-KNN	0.16	0.24	0.21	0.17	0.12	0.12	0.13
PCA-D-KNN	0.4	0.54	0.58	0.36	0.39	0.36	0.4
GA-KNN (MNTE)	0.17	0.31	0.2	0.13	0.2	0.17	0.14
GA-KNN (MNTSD)	0.08	0.31	0.27	0.13	0.16	0.16	0.1
GA-D-KNN (MNTE)	0.61	0.18	0.32	0.3	0.2	0.35	0.35
GA-D-KNN (MNTSD)	0.42	0.24	0.16	0.18	0.35	0.24	0.22
GPSO-KNN (MNTE)	0.36	0.41	0.28	0.27	0.32	0.2	0.21
GPSO-KNN (MNTSD)	0.4	0.37	0.38	0.19	0.26	0.25	0.12
GPSO-D-KNN (MNTE)	0.74	0.32	0.39	0.41	0.31	0.2	0.27
GPSO-D-KNN (MNTSD)	0.51	0.21	0.17	0.44	0.35	0.24	0.28
LPSO-KNN (MNTE)	0.4	0.46	0.25	0.31	0.23	0.15	0.19
LPSO-KNN (MNTSD)	0.36	0.34	0.24	0.27	0.27	0.23	0.12
LPSO-D-KNN (MNTE)	0.54	0.37	0.17	0.25	0.2	0.23	0.28
LPSO-D-KNN (MNTSD)	0.6	0.33	0.13	0.32	0.2	0.25	0.3

Table B.20. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Haberman dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	69.62	69.06	68.77	68.78	68.78	69.06	69.93
D-KNN	61.87	58.43	48.69	46.11	44.11	42.67	42.39
PCA-KNN	69.4	69.4	69.83	70.26	69.4	69.83	71.55
PCA-D-KNN	59.05	54.31	45.26	42.24	40.52	38.79	38.79
GA-KNN (MNTE)	70.69	70.69	71.98	72.84	69.4	73.71	73.28
GA-KNN (MNTSD)	70.26	71.12	69.83	68.1	71.12	72.41	73.28
GA-D-KNN (MNTE)	59.91	52.16	43.1	41.38	40.52	38.79	38.79
GA-D-KNN (MNTSD)	60.34	51.72	47.41	43.53	40.52	41.81	38.79
GPSO-KNN (MNTE)	67.67	72.84	67.24	66.81	70.69	71.55	72.41
GPSO-KNN (MNTSD)	72.84	67.67	66.81	66.38	68.97	72.84	70.69
GPSO-D-KNN (MNTE)	56.03	52.59	47.41	43.97	42.67	40.52	39.22
GPSO-D-KNN (MNTSD)	58.62	53.88	47.41	44.4	41.81	40.09	39.22
LPSO-KNN (MNTE)	70.69	69.4	67.24	67.67	71.55	71.55	71.98
LPSO-KNN (MNTSD)	71.12	73.71	67.24	70.26	66.38	69.83	67.24
LPSO-D-KNN (MNTE)	61.64	53.02	48.28	43.1	41.81	40.95	38.79
LPSO-D-KNN (MNTSD)	56.9	50.86	47.84	43.1	39.66	39.66	38.79

Table B.21. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the ILPD dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.1325	0.0135	0.0222	0.0928	0.0551	0.0299	0.0958
D-KNN	0.1706	0.2736	0.2468	0.2993	0.2575	0.3057	0.2632
PCA-KNN	0.1505	0.0167	0.0083	0.0019	0.0167	0.0074	0.0074
PCA-D-KNN	0.0167	0.0074	0.0167	0.0062	0.0047	0.0074	0.0074
GA-KNN (MNTE)	0.0042	0.1858	0.2248	0.1505	0.0464	0.091	0.2675
GA-KNN (MNTSD)	0.2248	0.2248	0.0669	0.0424	0.0019	0.1189	0.0669
GA-D-KNN (MNTE)	0.0019	0.0464	0.0074	0.0074	0.0031	0.0074	0.0074
GA-D-KNN (MNTSD)	0.0016	0.0297	0.1189	0.0464	0.0064	0.0167	0.0074
GPSO-KNN (MNTE)	0.091	0.0167	0.2675	0.091	0.0297	0.1189	0.1858
GPSO-KNN (MNTSD)	0.0019	0.091	0.1505	0.0297	0.1189	0.0464	0.1858
GPSO-D-KNN (MNTE)	0.0669	0.0297	0.0074	0.0074	0.0464	0.0297	0.0019
GPSO-D-KNN (MNTSD)	0.0074	0.0464	0.1189	0.0464	0.0464	0.0019	0.0167
LPSO-KNN (MNTE)	0.1858	0.091	0.1189	0.0167	0.0074	0.1189	0.091
LPSO-KNN (MNTSD)	0.0464	0.0019	0.1189	0.0019	0.0241	0.0138	0.0336
LPSO-D-KNN (MNTE)	0.0464	0.091	0.0669	0.0297	0.091	0.0167	0.0089
LPSO-D-KNN (MNTSD)	0.0074	0.0669	0.091	0.0033	0.0015	0.0018	0.0042

Table B.22. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the ILPD dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	68.89	71.11	73.33	73.89	74.44	74.44	74.44
D-KNN	74.44	74.44	74.44	74.44	74.44	74.44	74.44
PCA-KNN	66.11	70.0	74.44	73.89	74.44	74.44	74.44
PCA-D-KNN	74.44	74.44	74.44	74.44	74.44	74.44	74.44
GA-KNN (MNTE)	65.0	67.78	71.11	72.22	74.44	73.89	73.89
GA-KNN (MNTSD)	67.78	72.22	72.22	72.22	73.89	74.44	74.44
GA-D-KNN (MNTE)	73.89	74.44	74.44	74.44	74.44	74.44	74.44
GA-D-KNN (MNTSD)	74.44	74.44	74.44	74.44	74.44	74.44	74.44
GPSO-KNN (MNTE)	70.0	73.89	71.67	72.78	74.44	74.44	75.56
GPSO-KNN (MNTSD)	72.22	72.22	71.11	72.22	73.33	73.89	75.0
GPSO-D-KNN (MNTE)	72.22	74.44	75.0	74.44	74.44	74.44	74.44
GPSO-D-KNN (MNTSD)	76.11	74.44	74.44	74.44	74.44	74.44	74.44
LPSO-KNN (MNTE)	72.78	72.22	70.56	73.33	73.89	76.11	77.78
LPSO-KNN (MNTSD)	71.11	72.22	72.78	73.33	73.33	75.56	77.22
LPSO-D-KNN (MNTE)	74.44	74.44	75.56	74.44	74.44	74.44	74.44
LPSO-D-KNN (MNTSD)	75.56	74.44	74.44	74.44	74.44	74.44	74.44

Table B.23. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Immunotherapy dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	1.39	1.56	1.77	1.79	1.8	1.8	1.8
D-KNN	1.8	1.8	1.8	1.8	1.8	1.8	1.8
PCA-KNN	1.47	1.52	1.87	1.79	1.8	1.8	1.8
PCA-D-KNN	1.8	1.8	1.8	1.8	1.8	1.8	1.8
GA-KNN (MNTE)	1.17	1.29	1.5	1.65	1.8	1.79	1.79
GA-KNN (MNTSD)	1.43	1.44	1.51	1.71	1.79	1.8	1.8
GA-D-KNN (MNTE)	1.65	1.8	1.8	1.8	1.8	1.8	1.8
GA-D-KNN (MNTSD)	1.8	1.8	1.8	1.8	1.8	1.8	1.8
GPSO-KNN (MNTE)	1.73	1.99	2.16	2.16	2.0	2.28	2.07
GPSO-KNN (MNTSD)	2.19	1.71	2.18	2.19	1.91	2.06	2.21
GPSO-D-KNN (MNTE)	1.51	1.8	1.87	1.8	1.8	1.8	1.8
GPSO-D-KNN (MNTSD)	2.27	1.8	1.8	1.8	1.8	1.8	1.8
LPSO-KNN (MNTE)	1.88	2.19	1.86	2.04	2.06	2.34	2.26
LPSO-KNN (MNTSD)	0.6	2.26	1.54	1.91	2.11	2.07	2.29
LPSO-D-KNN (MNTE)	2.0	1.8	2.07	1.8	1.8	1.8	1.8
LPSO-D-KNN (MNTSD)	2.0	1.8	1.8	1.8	1.8	1.8	1.8

Table B.24. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Immunotherapy dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	83.89	82.22	81.39	78.33	76.67	77.22	76.11
D-KNN	81.94	75.56	68.06	56.94	49.17	45.56	41.94
PCA-KNN	77.5	78.06	77.78	76.39	75.28	74.44	74.44
PCA-D-KNN	71.39	62.22	56.11	48.06	45.0	41.94	39.44
GA-KNN (MNTE)	91.11	91.67	91.39	92.22	91.39	93.06	93.06
GA-KNN (MNTSD)	91.11	90.83	92.22	94.17	92.22	92.5	93.06
GA-D-KNN (MNTE)	91.39	88.61	87.5	86.39	81.67	79.17	79.72
GA-D-KNN (MNTSD)	93.06	87.5	87.5	85.0	82.22	79.17	75.28
GPSO-KNN (MNTE)	93.06	92.5	93.61	93.06	93.06	93.06	94.44
GPSO-KNN (MNTSD)	93.61	93.89	93.33	93.06	93.89	93.06	93.89
GPSO-D-KNN (MNTE)	94.17	88.06	88.06	87.5	88.33	84.72	83.89
GPSO-D-KNN (MNTSD)	94.44	92.5	92.5	88.61	85.83	84.72	86.67
LPSO-KNN (MNTE)	93.89	92.22	93.06	92.78	93.33	94.17	93.33
LPSO-KNN (MNTSD)	93.33	93.06	91.94	92.22	93.06	92.22	93.61
LPSO-D-KNN (MNTE)	93.33	90.0	88.89	88.89	86.11	86.67	84.17
LPSO-D-KNN (MNTSD)	93.06	91.94	88.61	86.39	86.94	84.17	85.0

Table B.25. Average accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Wine classification dataset.

Method	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$	$k = 13$	$k = 15$
KNN	0.32	0.72	0.62	0.48	0.47	0.41	0.5
D-KNN	0.31	0.43	0.47	0.66	0.7	0.59	0.45
PCA-KNN	0.49	0.61	0.41	0.43	0.49	0.6	0.51
PCA-D-KNN	0.84	0.26	0.51	0.62	0.61	0.32	0.29
GA-KNN (MNTE)	0.13	0.27	0.32	0.15	0.18	0.14	0.18
GA-KNN (MNTSD)	0.22	0.16	0.22	0.08	0.2	0.07	0.12
GA-D-KNN (MNTE)	0.2	0.47	0.31	0.3	0.69	0.91	0.98
GA-D-KNN (MNTSD)	0.21	0.55	0.43	0.62	1.07	0.76	0.66
GPSO-KNN (MNTE)	0.12	0.16	0.16	0.16	0.14	0.11	0.14
GPSO-KNN (MNTSD)	0.09	0.17	0.12	0.14	0.19	0.16	0.08
GPSO-D-KNN (MNTE)	0.11	0.62	0.22	0.4	0.27	0.48	0.51
GPSO-D-KNN (MNTSD)	0.15	0.19	0.16	0.32	0.25	0.4	0.31
LPSO-KNN (MNTE)	0.1	0.19	0.18	0.07	0.14	0.18	0.12
LPSO-KNN (MNTSD)	0.12	0.16	0.18	0.12	0.14	0.1	0.09
LPSO-D-KNN (MNTE)	0.09	0.48	0.26	0.36	0.45	0.24	0.41
LPSO-D-KNN (MNTSD)	0.19	0.25	0.38	0.56	0.48	1.12	1.0

Table B.26. Variance of accuracy of KNN Classification over 10-fold Monte-Carlo cross-validation with all variants studied, on the Wine classification dataset.

REFERENCES

[1] O. Obulesu, M. Mahendra, and M. ThrilokReddy, “Machine learning techniques and tools: A survey,” in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA).* IEEE, July 2018. [Online]. Available: https://doi.org/10.1109/icirca.2018.8597302

[2] Y. Jain, D. Chowdhury, and M. Chattopadhyay, “Machine learning based fitness tracker platform using MEMS accelerometer,” in *2017 International Conference on Computer, Electrical & Communication Engineering (ICCECE).* IEEE, Dec. 2017. [Online]. Available: https://doi.org/10.1109/iccece.2017.8526202

[3] S. Ghanta, S. Subramanian, L. Khermosh, S. Sundararaman, H. Shah, Y. Goldberg, D. Roselli, and N. Talagala, “Ml health: Fitness tracking for production models,” 2019.

[4] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, “Machine learning applications in cancer prognosis and prediction,” *Computational and Structural Biotechnology Journal*, vol. 13, pp. 8–17, 2015. [Online]. Available: https://doi.org/10.1016/j.csbj.2014.11.005

[5] C. Mahjoub, R. L. B. Jeannès, T. Lajnef, and A. Kachouri, “Epileptic seizure detection on EEG signals using machine learning techniques and advanced preprocessing methods,” *Biomedical Engineering / Biomedizinische Technik*, vol. 65, no. 1, pp. 33–50, Jan. 2020. [Online]. Available: https://doi.org/10.1515/bmt-2019-0001

[6] V. Grollemund, P.-F. Pradat, G. Querin, F. Delbot, G. L. Chat, J.-F. Pradat-Peyre, and P. Bede, “Machine learning in amyotrophic lateral sclerosis:

Achievements, pitfalls, and future directions," *Frontiers in Neuroscience*, vol. 13, Feb. 2019. [Online]. Available: https://doi.org/10.3389/fnins.2019.00135

[7] F. Carminati, G. Khattak, and M. Pierini, "Calorimetry with deep learning: particle classification, energy regression, and simulation for high-energy physics."

[8] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to LHC physics," *Annual Review of Nuclear and Particle Science*, vol. 68, no. 1, pp. 161–181, Oct. 2018. [Online]. Available: https://doi.org/10.1146/annurev-nucl-101917-021019

[9] S. Makhija, S. Saha, S. Basak, and M. Das, "Separating stars from quasars: Machine learning investigation using photometric data," *Astronomy and Computing*, vol. 29, p. 100313, Oct. 2019. [Online]. Available: https://doi.org/10.1016/j.ascom.2019.100313

[10] C. Lam and D. Kipping, "A machine learns to predict the stability of circumbinary planets," *Monthly Notices of the Royal Astronomical Society*, vol. 476, no. 4, pp. 5692–5697, Jan. 2018. [Online]. Available: https://doi.org/10.1093/mnras/sty022

[11] J. Wang, Y. Zhang, K. Tang, J. Wu, and Z. Xiong, "Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1900–1908. [Online]. Available: https://doi.org/10.1145/3292500.3330647

[12] B. Li, Y. Cheng, Y. Yuan, G. Wang, and L. Chen, "Three-dimensional stable matching problem for spatial crowdsourcing platforms," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining,*

ser. KDD ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1643–1653. [Online]. Available: https://doi.org/10.1145/3292500.3330879

[13] N. J. Mitra, I. Kokkinos, P. Guerrero, N. Thuerey, V. Kim, and L. Guibas, “Creativeai: Deep learning for graphics,” in *SIGGRAPH 2019 Courses*, ser. Siggraph 2019, 2019.

[14] S. Shen, M. Gowda, and R. R. Choudhury, “Closing the gaps in inertial motion tracking,” in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking - MobiCom 18.* ACM Press, 2018. [Online]. Available: https://doi.org/10.1145/3241539.3241582

[15] G. Devineau, F. Moutarde, W. Xi, and J. Yang, “Deep learning for hand gesture recognition on skeletal data,” in *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, 2018, pp. 106–113.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[17] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, “Deep learning vs. traditional computer vision,” in *Science and Information Conference.* Springer, 2019, pp. 128–144.

[18] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning.* Springer series in statistics New York, 2001, vol. 1, no. 10.

[19] C. Crisci, B. Ghattas, and G. Perera, “A review of supervised machine learning algorithms and their applications to ecological data,” *Ecological Modelling*, vol. 240, pp. 113–122, Aug. 2012. [Online]. Available: https://doi.org/10.1016/j.ecolmodel.2012.03.001

[20] C. Miller, Z. Nagy, and A. Schlueter, “A review of unsupervised statistical learning and visual analytics techniques applied to performance analysis of non-residential buildings,” *Renewable and Sustainable En-*

ergy Reviews, vol. 81, pp. 1365–1377, Jan. 2018. [Online]. Available: https://doi.org/10.1016/j.rser.2017.05.124

[21] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation."

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[24] C. Sammut, *Encyclopedia of machine learning.* New York: Springer, 2011.

[25] V. S. Prasatha, H. A. A. Alfeilate, A. B. Hassanate, O. Lasassmehe, A. S. Tarawnehf, M. B. Alhasanatg, and H. S. E. Salmane, "Effects of distance measure choice on knn classifier performance-a review," *arXiv preprint arXiv:1708.04321*, 2017.

[26] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[27] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal processing magazine*, vol. 25, no. 2, pp. 128–131, 2008.

[28] K. Zhao, H. Lu, and J. Mei, "Locality preserving hashing," 2014. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8357

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, Jan. 2014.

[30] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression.* Springer, 2002.

[31] H. Zhang and G. Chen, "The research of face recognition based on pca and k-nearest neighbor," in *2012 Symposium on Photonics and Optoelectronics*, 2012, pp. 1–4.

[32] F. Fink, K. Worle, P. Gruber, A. Tome, J. Gorriz-Saez, C. Puntonet, and E. Lang, "Ica analysis of retina images for glaucoma classification," in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2008, pp. 4664–4667.

[33] A. Jain and B. Chandrasekaran, "39 dimensionality and sample size considerations in pattern recognition practice," in *Handbook of Statistics*. Elsevier, 1982, pp. 835–855. [Online]. Available: https://doi.org/10.1016/s0169-7161(82)02042-2

[34] B. Liu, Y. Wei, Y. Zhang, and Q. Yang, "Deep neural networks for high dimension, low sample size data." in *IJCAI*, 2017, pp. 2287–2293.

[35] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," 2013.

[36] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster analysis*. John Wiley & Sons, 2011.

[37] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[38] C. Cortes and V. Vapnik, "Soft margin classifier," June 17 1997, uS Patent 5,640,492.

[39] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *Journal of Machine Learning Research*, vol. 13, no. Oct, pp. 3103–3131, 2012.

[40] Q. R. Wang and C. Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, no. 4, pp. 406–417, 1984.

[41] S. Jiang, Z. Song, O. Weinstein, and H. Zhang, "Faster dynamic matrix inverse for faster lps," 2020.

[42] T.-L. Pao, Y.-T. Chen, J.-H. Yeh, Y.-M. Cheng, and Y.-Y. Lin, "A comparative study of different weighting schemes on knn-based emotion recognition in mandarin speech," in *International Conference on Intelligent Computing.* Springer, 2007, pp. 997–1005.

[43] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[44] T. F. Brooks, D. S. Pope, and M. A. Marcolini, "Airfoil self-noise and prediction," 1989.

[45] J. R. Quinlan, "Combining instance-based and model-based learning," in *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ser. ICML'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, p. 236–243.

[46] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete research*, vol. 28, no. 12, pp. 1797–1808, 1998.

[47] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012.

[48] M. Cassotti, D. Ballabio, V. Consonni, A. Mauri, I. V. Tetko, and R. Todeschini, "Prediction of acute aquatic toxicity toward daphnia magna by using the GA-kNN method," *Alternatives to Laboratory Animals*, vol. 42, no. 1, pp. 31–41, Mar. 2014. [Online]. Available: https://doi.org/10.1177/026119291404200106

[49] M. Cassotti, D. Ballabio, R. Todeschini, and V. Consonni, "A similarity-based QSAR model for predicting acute toxicity towards the fathead minnow (pimephales promelas)," *SAR and QSAR in Environmental Research*, vol. 26, no. 3, pp. 217–243, Mar. 2015. [Online]. Available: https://doi.org/10.1080/1062936x.2015.1018938

[50] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[51] J. Gerritsma, R. Onnink, and A. Versluis, "Geometry, resistance and stability of the delft systematic yacht hull series," *International shipbuilding progress*, vol. 28, no. 328, pp. 276–297, 1981.

[52] I.-C. Yeh, K.-J. Yang, and T.-M. Ting, "Knowledge discovery on rfm model using bernoulli sequence," *Expert Syst. Appl.*, vol. 36, no. 3, p. 5866–5871, Apr. 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2008.07.018

[53] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology." *Proceedings of the National Academy of Sciences*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990. [Online]. Available: https://doi.org/10.1073/pnas.87.23.9193

[54] D. A. de Campos, J. Bernardes, A. Garrido, J. M. de S, and L. Pereira-Leite, "Sisporto 2.0: A program for automated analysis of cardiotocograms," *The Journal of Maternal-Fetal Medicine*, vol. 9, no. 5, pp. 311–318, 2000.

[55] M. Amin and A. Ali, "Performance evaluation of supervised machine learning classifiers for predicting healthcare operational decisions."

[56] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *Know.-Based Syst.*, vol. 60, p. 20–27, Apr. 2014. [Online]. Available: https://doi.org/10.1016/j.knosys.2013.12.023

[57] B. V. Ramana, M. S. P. Babu, N. Venkateswarlu, *et al.*, "A critical study of selected classification algorithms for liver disease diagnosis."

[58] F. Khozeimeh, R. Alizadehsani, M. Roshanzamir, A. Khosravi, P. Layegh, and S. Nahavandi, "An expert system for selecting wart treatment method," *Computers in Biology and Medicine*, vol. 81, pp. 167–175, feb 2017. [Online]. Available: https://doi.org/10.1016%2Fj.compbiomed.2017.01.001

[59] I. Fischer and J. Poland, "Amplifying the block matrix structure for spectral clustering," 2005.

[60] Q.-S. Xu and Y.-Z. Liang, "Monte carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1–11, Apr. 2001. [Online]. Available: https://doi.org/10.1016/s0169-7439(00)00122-2

[61] B. Cook and M. Huber, "Balanced k-nearest neighbors," 2019. [Online]. Available: https://aaai.org/ocs/index.php/FLAIRS/FLAIRS19/paper/view/18285

[62] Y. Lin, J. Li, M. Lin, and J. Chen, "A new nearest neighbor classifier via fusing neighborhood information," *Neurocomputing*, vol. 143, pp. 164–169, Nov. 2014. [Online]. Available: https://doi.org/10.1016/j.neucom.2014.06.009

[63] Y. Zhu, Z. Wang, and D. Gao, "Gravitational fixed radius nearest neighbor for imbalanced problem," *Knowledge-Based Systems*, vol. 90, pp. 224–238, Dec. 2015. [Online]. Available: https://doi.org/10.1016/j.knosys.2015.09.015

[64] Ömer Faruk Ertuğrul and M. E. Tağluk, "A novel version of k nearest neighbor: Dependent nearest neighbor," *Applied Soft Computing*, vol. 55, pp. 480–490, June 2017. [Online]. Available: https://doi.org/10.1016/j.asoc.2017.02.020

[65] B. Nguyen, C. Morell, and B. D. Baets, "Large-scale distance metric learning for k-nearest neighbors regression," *Neurocomputing*, vol. 214, pp. 805–814, Nov. 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2016.07.005

[66] J. Hocke and T. Martinetz, "Maximum distance minimization for feature weighting," *Pattern Recognition Letters*, vol. 52, pp. 48–52, Jan. 2015. [Online]. Available: https://doi.org/10.1016/j.patrec.2014.10.003

[67] J. Derrac, F. Chiclana, S. García, and F. Herrera, "Evolutionary fuzzy k-nearest neighbors algorithm using interval-valued fuzzy sets," *Information Sciences*, vol. 329, pp. 144–163, Feb. 2016. [Online]. Available: https://doi.org/10.1016/j.ins.2015.09.007

[68] N. Biswas, S. Chakraborty, S. S. Mullick, and S. Das, "A parameter independent fuzzy weighted k -nearest neighbor classifier," *Pattern Recognition Letters*, vol. 101, pp. 80–87, Jan. 2018. [Online]. Available: https://doi.org/10.1016/j.patrec.2017.11.003

[69] Y. Miao, X. Tao, Y. Sun, Y. Li, and J. Lu, "Risk-based adaptive metric learning for nearest neighbour classification," *Neurocomputing*, vol. 156, pp. 33–41, May 2015. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.01.009

[70] M. Mitchell, *An introduction to genetic algorithms.* Cambridge, Mass: MIT Press, 1996.

[71] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[72] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[73] A. Theophilus, S. Saha, S. Basak, and J. Murthy, "A novel exoplanetary habitability score via particle swarm optimization of CES production functions," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI).* IEEE, Nov. 2018. [Online]. Available: https://doi.org/10.1109/ssci.2018.8628669

[74] M. Beckmann, N. F. F. Ebecken, and B. S. L. P. de Lima, “A KNN undersampling approach for data balancing,” *Journal of Intelligent Learning Systems and Applications*, vol. 07, no. 04, pp. 104–116, 2015. [Online]. Available: https://doi.org/10.4236/jilsa.2015.74010

[75] Y. Xiang, Z. Cao, S. Yao, and J. He, “CW-kNN,” in *Proceedings of the 4th International Conference on Communication and Information Processing - ICCIP 18*. ACM Press, 2018. [Online]. Available: https://doi.org/10.1145/3290420.3290431

[76] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, June 2002. [Online]. Available: https://doi.org/10.1613/jair.953

[77] “Correction to: Erroneous pagination in volume 60, issue 2 and issue 3,” *Statistical Papers*, vol. 60, no. 5, pp. 1799–1802, Sept. 2019. [Online]. Available: https://doi.org/10.1007/s00362-019-01131-w

[78] H. Wang, R. Zhu, and P. Ma, “Optimal subsampling for large sample logistic regression,” *Journal of the American Statistical Association*, vol. 113, no. 522, pp. 829–844, Apr. 2018. [Online]. Available: https://doi.org/10.1080/01621459.2017.1292914

[79] R. Zhu, P. Ma, M. W. Mahoney, and B. Yu, “Optimal subsampling approaches for large sample linear regression,” 2015.

BIOGRAPHICAL STATEMENT

Suryoday Basak was born in Kolkata, India, in 1995. He received his Bachelor of Engineering degree from PESIT Bangalore South Campus (when it was affiliated to Visvesvaraya Technological University), Bangalore, India, in 2017, in Computer Science and Engineering, his Master of Science from The University of Texas at Arlington in 2020, in Computer Science. From 2017 to 2018, he was with the department of Computer Science and Engineering, PESIT Bangalore South Campus as a Research Associate, working with Dr. Snehanshu Saha. His current research interest is in the area of machine learning and cyberphysical systems.