Mahinda Mailagaha Kumbure

NOVEL FUZZY *K*-NEAREST NEIGHBOR METHODS FOR EFFECTIVE CLASSIFICATION AND REGRESSION

ACTA UNIVERSITATIS LAPPEENRANTAENSIS 1046

Mahinda Mailagaha Kumbure

NOVEL FUZZY *K*-NEAREST NEIGHBOR METHODS FOR EFFECTIVE CLASSIFICATION AND REGRESSION

Dissertation for the degree of Doctor of Philosophy to be presented with due permission for public examination and criticism in the Auditorium 1316 at Lappeenranta-Lahti University of Technology LUT, Lappeenranta, Finland on the 10th of November, 2022, at noon.

Acta Universitatis
Lappeenrantaensis 1046

Supervisors Professor Pasi Luukka
LUT School of Business and Management
Lappeenranta-Lahti University of Technology LUT
Finland

Professor Mikael Collan
LUT School of Business and Management
Lappeenranta-Lahti University of Technology LUT
Finland

Reviewers Professor Frank Chung-Hoon Rhee
School of Electrical Engineering
Hanyang University
South Korea

Associate Professor Sansanee Auephanwiriyakul
Computer Engineering Department
Chiang Mai University
Thailand

Opponent Professor Frank Chung-Hoon Rhee
School of Electrical Engineering
Hanyang University
South Korea

ISBN 978-952-335-868-3
ISBN 978-952-335-869-0 (PDF)
ISSN 1456-4491 (Print)
ISSN 2814-5518 (Online)

Lappeenranta-Lahti University of Technology LUT
LUT University Press 2022

Abstract

Mahinda Mailagaha Kumbure
Novel fuzzy *k*-nearest neighbor methods for effective classification and regression
Lappeenranta 2022
102 pages
Acta Universitatis Lappeenrantaensis 1046
Diss. Lappeenranta-Lahti University of Technology LUT
ISBN 978-952-335-868-3, ISBN 978-952-335-869-0 (PDF), ISSN 1456-4491 (Print), ISSN 2814-5518 (Online)

With the advancement of technology in many areas, an immense amount of data has currently become available, and discovering patterns and trends from this data is a core subject of interest in machine learning research. Machine learning, a form of artificial intelligence, provides a robust set of algorithms that iteratively learn from the data to understand and analyze data as well as predict future outcomes. The focus of this dissertation is on supervised machine learning techniques—classification and regression. In particular, the emphasis is on the fuzzy k-nearest neighbor (FKNN) algorithm that has received substantial attention in classification problems due to its efficacy and flexibility.

In the context of classification, learning from data can be considered challenging for many algorithms due to uncertainties and inconsistencies in the data. In particular, a typical issue associated with most classification problems is that class distributions in the data are imbalanced—meaning that data points do not equally represent the classes in class variable, which can significantly affect classification performance. Along with class imbalance, it is apparent that a level of class overlapping, class noise, and outliers may also cause the degradation of the classifier's performance. Given these issues, research has continued to make classification algorithms—particularly the nearest neighbor-based methods—more accurate and more competent. However, this has been a great challenge because the performance and efficiency of learning algorithms are heavily reliant on the correct choice of model features and data that often engages with many issues. In this context, this research seeks to develop solution techniques based on the FKNN algorithm, particularly for class imbalance problems.

The multi-local power mean fuzzy k-nearest neighbor (MLPM-FKNN), which uses class prototype local mean vectors instead of individuals for creating memberships, is the first approach presented in this dissertation. It is demonstrated that the proposed MLPM-FKNN classifier achieves better classification results than the classical methods in real-world data sets, often with high k (number of nearest neighbors chosen) values. In addition, the MLPM-FKNN classifier, in cooperation with feature selection, is applied to create a hybrid feature selection model to forecast the intraday return of the S&P index. Further, this work brings a feature selection and prediction (formed by classification) to a nexus wherein the feature selection can produce a significant impact with the help of MLPM-FKNN classification. The second approach proposed is the Bonferroni

mean-based fuzzy k-nearest neighbor (BM-FKNN) classifier, which is an extension of the MLPM-FKNN method by the use of the Bonferroni mean instead of the Power mean. The findings with one artificial and six real-world data sets stress the capability and effectiveness of this method in solving class imbalance problems as compared to the original and several other competitive classifiers. The next contribution of this dissertation is a novel regression approach called the Minkowski distance-based fuzzy k-nearest neighbor regression (Md-FKNNreg) method. This is motivated by the fact that no one has investigated the ability of the FKNN method in regression settings, although it has gained broader attention in the classification context. Moreover, the principal advantage of this algorithm is that it attributes importance to the nearest neighbors using fuzzy weights considering their distances to the test instance and hence makes a more accurate prediction across a weighted average. Experimental results using real-world data show that the Md-FKNNreg outperformed the benchmark models and thus highlight its potential in terms of linear and non-linear regression problems.

Keywords: classification, regression, feature selection, machine learning, prediction, class imbalance, fuzzy *k*-nearest neighbor, local means, performance.

Acknowledgements

The research presented in this dissertation was carried out at the School of Business and Management at LUT University, Lappeenranta, Finland, between November 2018 and May 2022. This work was performed within supports from many people. I am not able to refer names of everyone, but I deeply appreciate all your invaluable support.

First and foremost, I would like to express my deepest gratitude to my first supervisor, Professor Pasi Luukka, for his excellent guidance, continuous support, and encouragement throughout my doctoral and master's studies. I am extremely grateful that you took me on as a student and continued to have faith in me over the years. I am also indebted to my second supervisor, Professor Mikael Collan, for his support and guidance. Your feedback and assistance helped my research greatly, and your advice was priceless. Thank you both for trusting me, for your advice, and for helping me grow as an independent researcher—I feel incredibly privileged to have such motivating and supporting supervisors.

I would also like to thank the preliminary examiners, Professor Frank Chung-Hoon Rhee and Associate Professor Sansanee Auephanwiriyakul, for their careful review of the manuscript, insightful feedback, and suggestions for improving the manuscript. Your effort is highly appreciated.

I gratefully acknowledge the financial support that I received from the Research Foundation of Lappeenranta University of Technology, the Manufacturing 4.0 (MFG 4.0) project of the Finnish Strategic Research Council, and the Finnish Foundation for Share Promotion (Pörssisäätiö) during my doctoral studies.

I am also grateful to Associate Professor Jan Stoklasa and Post-doctoral Researcher Christoph Lohrmann for their help, insightful discussions, and moral support throughout my research. In addition, I would like to extend my thanks to other members of the Business Analytics team for their energy, understanding, and support during my research and other duties. I am truly blessed to be surrounded by such amazing colleagues and supporting staff at the School of Business and Management at LUT. Thank you to all who have either directly or indirectly facilitated this work. Further, I would like to thank the staff at the Doctoral School at LUT for their support during the studies and dissertation process.

Most importantly, I want to express my heartfelt gratitude to my family: my mother, Mrs. Kamalawathi, my father, Mr. Premarathna, and my brothers, Duminda and Malinda, and my loving sister, Dhammika, for their continuous support, care, and encouragement in every step to make me in the present stage. Further, my sincere appreciation goes to Veenavee for her unconditional love and support throughout the dissertation process and every day. I am very blessed and thankful for your understanding, never-ending encouragement, and always being there for me. Finally, thanks to all my friends near and

far, who helped me to strive towards my goals. Specifically, I want to thank Charlotte aunty and Markku uncle, who make Lappeenranta feel like home. I acknowledge all who have helped me during my studies and in all aspects of my life. Thank you!

Mahinda Mailagaha Kumbure
October 2022
Lappeenranta, Finland

To my Family

Contents

List of publications

This dissertation is based on the following papers. The rights have been granted by publishers to include the papers in dissertation.

I. Mailagaha-Kumbure, M., Luukka, P. and Collan, M. (2019). An enhancement of fuzzy k-nearest neighbor classifier using multi-local power means. In: *Proc. 11th Conference European Society for Fuzzy Logic and Technology (EUSFLAT 2019)*, pp. 83-90, Atlantis Press.

II. Mailagaha-Kumbure, M., Lohrmann, C. and Luukka, P. (2022). A study on relevant features for intraday S&P 500 prediction using a hybrid feature selection approach. In *Nicosia et al., (eds.) Machine Learning, Optimization, and Data Science (LOD 2021), Lecture Notes in Computer Science*, 13163, pp. 93-104 Springer, Cham.

III. Mailagaha-Kumbure, M., Luukka, P. and Collan, M. (2020). A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean. *Pattern Recognition Letters*, 140, pp. 172-178.

IV. Mailagaha-Kumbure, M. and Luukka, P. (2021). A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance. *Granular Computing*, 7, pp. 657-671.

Author's contribution

Mahinda Mailagaha Kumbure is the principal author and investigator in papers I–IV. In all publications, Mahinda Mailagaha Kumbure contributed to conceptualizing the research topics, developing proposed methods, implementing and testing them in MATLAB code, analyzing the results, visualization, and articulating the original drafts.

Nomenclature

Mathematical Notations

α	Significance level
$\hat{y}$	Predicted value of y
∞	Infinity
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathcal{B}$	Bonferroni mean
$\mathcal{D}$	Dimension of the feature space
$\mathcal{L}$	Local mean vector
$\mathcal{M}$	Power mean
$\mathcal{Y}$	Continuous output variable
ω	One class label
σ	Standard deviation
c	Class
$h(X)$	Predictor function of X
k	Number of nearest neighbors
L_1	1-norm space
$L_{\mathcal{P}}$	$\mathcal{P}$-norm space
m	Number of instances
max	Maximum
min	Minimum
N	Negative
n_i	Number of instances in i^{th} class
nn	Nearest neighbor
P	Positive
p	Power mean parameter
q	Bonferroni mean parameter
r	Fuzzy strength parameter
R^2	Coefficient of determination
S	Number of classes
u_{ij}	Membership of j^{th} nearest neighbor in the i^{th} class
X, Y	Data vector
x, y	One value of data vector

Abbreviations

CI	Confidence interval
DEFS	Differential evolution feature selection
FADPTKNN	Fuzzy adaptive *k*-nearest neighbor
FENN	Fuzzy edited nearest neighbor
FKNN	Fuzzy *k*-nearest neighbor

FKNNreg	Fuzzy k-nearest neighbor regression
FRKNNA	Fuzzy rough k-nearest neighbor approach
FRNN	Fuzzy rough nearest neighbor approach
FRNN-VQRS	Vaguely quantified rough set based on fuzzy rough nearest neighbor
FWKNN	Fuzzy weighted k-nearest neighbor
GAFKNN	Genetic algorithm based fuzzy k-nearest neighbor
GMDKNN	Generalized mean distance-based KNN classifier
IFROWANN	Imbalanced fuzzy-rough ordered weighted average nearest neighbor
IFSKNN	Intuitionistic fuzzy sets nearest neighbor
IFV-NP	Intuitionistic fuzzy version nearest prototype
IT2FKNN	Interval type-2 fuzzy k-nearest neighbor
IV-KNN	Interval-valued k-nearest neighbor
KNN	K-nearest neighbor
KNNreg	K-nearest neighbor regression
LASSO	Least absolute shrinkage and selection
LM-KNN	Local mean k-nearest neighbor
MLR	Multiple linear regression
MOGA-MFNN	Multi-objective genetic algorithm-modified fuzzy K-nearest neighbor
NB	Naive Bayes
OWA	Ordered weighted average
PCA	Principal component analysis
PIFW-kNN	Parameter independedent fuzzy k-nearest neighbor
PNN	Pseudo nearest neighbor
RMSE	Root mean square error
S&P	Standard and poor
SVM	Support vector machine
SVR	Support vector regression
TI	Technical indicator

1 Introduction

Over the years, advances in technology in an increasingly digital world and computerization of every part of our lives—from how we work, communicate, and access information—have made the world data-driven. Ultimately, extracting valuable information and knowledge from this data is at the forefront of recent research in academia and industry (Sarker, 2021; Aggarwal, 2015; Igual and Seguí, 2017). Here, it is noteworthy that advantages do not come from the data itself but from the capacity to obtain hidden patterns and information from the data. This is where the task of *machine learning* comes in (Kubat, 2017). Moreover, the availability of diverse and large-scale benchmark data sets has fueled the fast pace of expansion in machine learning research in recent years. While machine learning is about learning from the data (Kubat, 2017), it is not a simple process (Hurwitz and Kirsch, 2018). Specifically, it consists of various algorithms that iteratively learn from the data to understand and analyze data as well as forecast future outcomes (Hurwitz and Kirsch, 2018). Depending on the nature of the problem addressed, machine learning techniques are typically categorized into the following (i) supervised learning that learns from data containing both inputs and corresponding outputs; (ii) unsupervised learning that uses data involving only a set of inputs without corresponding outputs; (iii) semi-supervised learning, in which the data contains inputs and only a part of the outputs; and (iv) reinforcement learning, which is behavioral learning, indicating that it learns from trial and error instead of training from sample data (Sarker, 2021; Hurwitz and Kirsch, 2018; Igual and Seguí, 2017). This research is devoted to supervised learning techniques, and the other types are not discussed any further.

In supervised learning, there is a target variable that is categorical (classification problems) or numerical (regression problems), which is predicted. The model to predict this variable is fitted using training data. After the model is fitted, it can be used to predict new input observations. In the context of supervised learning, two types of machine learning techniques can be distinguished:

- Classification, which refers to assigning objects into a relatively small set of (discrete) categories (Bishop, 2006; Jung, 2022). These categories are often called "classes," "labels," or "targets." The primary goal in classification problems is to identify the class to which a new object belongs. For example, given a record of symptoms shown by a patient, the disease diagnosis system must classify whether the patient suffers from the disease or not. Additionally, the spam filtering model is another example of a classification task in which the goal is to classify emails as spam or non-spam. Moreover, regarding the number of classes in the output variable, a classification problem is typically called a binary class (when there are only two classes) or a multi-class (when there are more than two classes) problem (Tharwat, 2020). Both examples mentioned before are binary class problems. In the context of classification, the objects are often referred to as "instances" that describe different properties of associated categories. These properties, known as "features" (or variables), are used to make the expected classification. Meanwhile,

an instance with a corresponding class label (a labeled example) indicates its correct classification. An unlabeled example (also referred to as a "query sample" or "test sample") is an instance that needs to be classified. In this dissertation, the term "query sample" is used.

- Regression is a way to fit a learning algorithm to training data in which "continuous" numerical values characterize the outcome variable (Sarker, 2021; Jung, 2022). It is deployed to make predictions of numeric outcomes for new data based on historical data (Igual and Seguí, 2017; Witten et al., 2011)—for example, predicting a company's sales in the next month, the stock prices for the next day, and expected weather (e.g., temperature, wind speed). In such cases, a regression method approaches the problem by discovering a relationship between input features and the corresponding target variable. Generally, this relationship can be linear or non-linear, and basic techniques [e.g., multiple linear regression (Montgomery et al., 2012)] cannot learn non-linear relationships, which is a disadvantage of using those methods (Witten et al., 2011). However, some variants, such as the support vector regression (SVR) (Drucker et al., 1997) have been designed for data that has a non-linear dependency.

Meanwhile, it is noteworthy that the difference between classification and regression tasks in supervised context is somewhat ambiguous (Jung, 2022) because sometimes it is possible to solve classification problems by converting them into regression problems and vice versa. Nevertheless, the emphasis in this research is not only on enhanced classification and regression algorithms but also on applications. The starting point is the fuzzy k-nearest neighbor (FKNN) classifier by Keller et al. (1985).

The k-nearest neighbor (KNN) approach, a form of instance-based supervised learning (Kassani et al., 2017), is used to solve both regression and classification problems (Sarker, 2021). The KNN principle is straightforward and involves nothing more than a similarity calculation between a new instance and the training instances, plus votes cast for the majority class among the closest instances. It is recognized as one of the top ten algorithms applied in data mining and machine learning due to its simplicity of implementation and good performance (Derrac et al., 2015; Kassani et al., 2017; Biswas et al., 2018; Gou et al., 2014, 2019; Zeraatkar and Afsari, 2021). The KNN method is also referred to as a lazy algorithm because it requires no training phase (Hurwitz and Kirsch, 2018) and simply uses training data to yield outcomes. Since its initial proposal by Cover and Hart (1967), the KNN method is applied by many researchers and is still one of the most promising techniques for pattern classification (Chen et al., 2011; Derrac et al., 2014; Kassani et al., 2017). Additionally, it could also be a valuable tool for implementing predictions, as shown by Zhang et al. (2017a); Cao et al. (2019) where the KNN was used to identify historical patterns and subsequently forecast the future trends of a stock index price. Despite its influential role, it is noteworthy that the performance of the original KNN rule is degraded by several inherent limitations, such as assigning equal importance to the nearest neighbors (Keller et al., 1985; Kassani et al., 2017; Rhee and Hwang, 2003), high sensitivity to the value of k (Pan et al., 2017; Gou et al., 2014,

2019), and the distance metric that is used (Rastin et al., 2021; Pan et al., 2017). This method may also show poor performance when the cases have outlier effects (Gou et al., 2019) and the data sets are not balanced in terms of class distributions (Zeraatkar and Afsari, 2021). Given these issues, many studies have attempted to enhance the accuracy of the KNN method by inventing advanced new variants while solving associated issues with the original method (Pan et al., 2017; Rastin et al., 2021; Gou et al., 2014, 2019).

Using a degree of membership of each instance in the classes based on the fuzzy theory (Zadeh, 1965), Keller et al. (1985) introduced a fuzzy version of the original KNN method, namely the FKNN classifier, which is regarded as one of the most effective KNN variants (Derrac et al., 2015; Wu et al., 2021). This method solves the issue with the KNN method of giving equal importance to every neighboring instance regardless of the fact that different instances may have different impacts on the new instance (Derrac et al., 2015, 2016; Wu et al., 2021). Another interesting aspect of this method is that it uses a class specific membership for each instance for the decision rule, enabling the model to achieve higher performance while dealing with uncertainties associated with feature values and class distributions (Liao and Li, 1997). Consequently, for the aforementioned reasons, the FKNN algorithm has gained continuous interest in solving many real-world classification problems in, to mention a few, business (Chen et al., 2011), medicine (Cabello et al., 1991; Chen et al., 2013), manufacturing (Liao and Li, 1997), and bio-science (Huang and Li, 2004). The present research is particularly interested in the FKNN classifier and seeks to extend the original algorithm and improve its performance by introducing several changes to the learning part of it.

However, even though it improves the performance of the KNN, this algorithm increases the complexity of calculations since it introduces an additional parameter (Biswas et al., 2018). In this case, a proper optimization of the involved parameters for optimal values is required. Moreover, in terms of the class imbalance problems (also in cases that have outlier effects or class overlapping issues), the uncertainty associated with the class distributions and data could be higher, and the FKNN classifier may fail to account for them entirely. This is because its decision rule, as in the KNN, also depends on the individual samples in the neighborhood. Under this circumstance, an incorrect decision may prevail if the position of some individuals is unusual relative to others or fewer individuals represent some class(es). Specifically, in such cases, the membership assignment is the key issue in the FKNN method (Derrac et al., 2016), and if the observed nearest instances do not reflect accurate information, it may lead to inadequate membership values for training instances for each class. These issues cannot be avoided when the FKNN method is used and is one of the concerns in our research.

As already mentioned, the basic idea of the KNN method has been also used in the context of regression, which is referred to as the k-nearest neighbor regression (KNNreg) method (Turner, 1977; Benedetti, 1977; Stone, 1977). Here, the main focus is how to predict an outcome for a given new instance by averaging the outputs of the nearest neighbors of the new instance (Durbin et al., 2021). Moreover, the KNNreg method requires no

calculation of coefficients, which makes it computationally inexpensive and easier to interpret than other more refined regression approaches. Particularly, for situations where the output variable and input features may indicate a non-linear relationship, the KNNreg can be a useful choice of interest (Cai et al., 2020). Accordingly, a growing body of literature has examined the potential of the KNNreg in various fields (for example, see Hu et al., 2014; Huang and Li, 2004; Durbin et al., 2021; Yao and Ruzzo, 2006). In addition, the simplicity and potential of the KNNreg have led studies (Biau et al., 2012; Song et al., 2017; Nguyen et al., 2016) to create several improved variants. However, regarding the FKNN method, in the literature, there is no evidence of this method providing a clear indication in the regression setting. For this reason, the interest of this dissertation lies in examining the capability of the FKNN method as a regression method, which is the second emphasis of this research.

To this end, the field of KNN-based classification research is clearly active and provides new insights. This research is not about solving something with the FKNN algorithm—but about identifying new generalizations of it and in this way, expanding the capabilities of KNN methods. This is further clarified with the research objectives in the following sub-section.

1.1 Research Objectives

A machine learning algorithm does not remain fitting as it is required to be maintained occasionally, particularly due to the recent advancement of data sources and problem domains. The rationale of this research is that the FKNN method as a classifier as well as a regression model in real-world data can and should be performed effectively. From the classification perspective, the idea is to generalize the capability of the original method so that it can perform well, particularly in class imbalance problems. In addition, the focus is especially on the generalizability of the FKNN algorithm in terms of regression problems.

The objectives of the research in this dissertation can be summarized as follows:

- To examine the effect of using class prototype mean vectors instead of individual neighboring instances in the learning part of the FKNN algorithm on the classification performance in terms of imbalanced data sets.

- To examine the effect of using more general mean operator than the usual arithmetic operator in the calculation of the mean vector.

- To study the effect of combining KNN-based classifiers with feature selection on the classification performance.

- To generalize the FKNN algorithm so that it is able to address regression problems and examine its performance with respect to different alternative problems.

Overall, this research will focus on generalizing the original FKNN method. The area of theoretical aspects and strategies used to design new KNN-based methods or expand existing ones is vast (Derrac et al., 2014). First, several mean operators are focused on for this research, as they can be potential in modeling class representative vectors. These class representative mean vectors, which are theoretically observed by averaging the instances in each class, are used to provide local information about the represented class during the training. The concept of using local class representative mean vectors in the context of KNN classification is not new, as the same or similar ideas can be seen in previous studies (Mitani and Hamamoto, 2006; Gou et al., 2014, 2019; Pan et al., 2017). However, before this research and the associated proposals, no one had investigated such class prototype vectors in the FKNN classification. The underlying assumption in using such local mean vectors is that they may be more representative than individuals in terms of corresponding classes and can contribute to more accurate memberships. Moreover, it has already been observed that with the rationale of the KNN method, it is possible to characterize the problems in regression settings and obtain a high performance without relying on data-specific assumptions and parameters. In fact, in this research, an attempt is made to extend the FKNN method for regression problems and study its performance on some of the most commonly used data sets from the related fields in comparison to the KNNreg and several other competitive methods. Finally, several experiments that follow necessary principles in machine learning are conducted to judge whether the new method is a relevant contribution to the state-of-the-art classification and regression. In this regard, different feature selection methods and distance measures are deployed with and within classifiers to enhance their problem-solving capabilities. All the objectives of this dissertation are fulfilled across four main publications, which will be summarized in the next sub-section.

1.2 Background Information and Contributions

This research is considered to be contributing in two ways: enhancing the performance of the conventional FKNN algorithm (Keller et al., 1985) for various classification problems and generalizing its rationale so that it can also perform in terms of regression applications. To our knowledge, the second has not been adequately addressed or has been to a minimal degree. However, many studies have attempted to advance the efficacy of the FKNN method in the context of classification. Thus, this study will concentrate on the related work on enhanced FKNN classifier variants and the methods applied to modify the original algorithm.

Using different notions and principles, some authors have used the associated characteristics in the FKNN algorithm—particularly the nearest neighbor search, similarity measure, membership assignment, and the decision rule—to make the original approach more effective and practical (Derrac et al., 2014). The study by Hu and Xie (2005) applied the genetic algorithm to make the FKNN classifier less sensitive to its key parameters, and proposed a new version of the FKNN called the "genetic algorithm-based fuzzy k-nearest neighbor" (GAFKNN) approach. Similarly, Kassani et al. (2017) presented the

multi-objective genetic algorithm-modified fuzzy *k*-nearest neighbor (MOGA-MFNN) approach by introducing two new objective functions into the optimization part. A study by Biswas et al. (2018) also proposed a new parameter independent fuzzy class-specific feature weighted *k*-nearest neighbor (PIFW-kNN) method based on a specific feature weighting scheme. Moreover, some studies have proposed improved versions of the original methods, particularly focusing on class imbalance problems, such as the fuzzy weighted *k*-nearest neighbor (FWKNN) (Harshita and Singh, 2017) and the fuzzy adaptive *k*-nearest neighbor (FADPTKNN) (Patel and Thakur, 2019) methods.

Meanwhile, type-2 fuzzy sets, an extension of the type-1 fuzzy sets, can effectively deal with uncertainty and complexity and allow memberships to be varied in a broader range (Kassani et al., 2017). Using type-2 fuzzy sets, Rhee and Hwang (2003) proposed the interval type-2 fuzzy *k*-nearest neighbor (IT2FKNN) classifier as an extended version of the classical method. Additionally, interval-valued fuzzy sets, a specific type of type-2 fuzzy sets (Derrac et al., 2016), have also been utilized to design new FKNN variants, such as the interval-valued *k*-nearest neighbor (IV-KNN) (Derrac et al., 2015) and the evolutionary fuzzy *k*-nearest neighbors classifier using interval-valued fuzzy sets (EF-*k*NN-IVFS) (Derrac et al., 2016). Moreover, fuzzy rough sets and intuitionistic fuzzy sets have also gained attention among FKNN developers. The fuzzy rough *k*-nearest neighbor approach (FRKNNA) (Bian and Mazlack, 2003), the fuzzy rough-nearest neighbor (FRNN) (Sarkar, 2007), the fuzzy-rough *k*-nearest neighbor for imbalance learning (IM_FRknn) (Han and Mao, 2010), the vaguely quantified rough set based on fuzzy-rough nearest neighbor (FRNN-VQRS) (Jensen and Cornelis, 2011), and the imbalanced fuzzy-rough ordered weighted average nearest neighbor (IFROWANN) (Ramentol et al., 2015) are examples of enhanced FKNN variants based on the fuzzy rough set theory. Additionally, by introducing intuitionistic fuzzy sets to the classical algorithm, an intuitionistic version of the FKNN method [called the intuitionistic fuzzy *k*-nearest neighbor (IF-KNN)] has been established in a study by Kuncheva (1995). From this perspective, Hadjitodorov (1995) presented another variant, the intuitionistic fuzzy sets-nearest neighbor (IFSKNN) classifier, and later expanded this method to the intuitionistic fuzzy version-nearest prototype (IFV-NP) method in Hadjitodorov (2001). The objective of these methods is to improve the classification performance with the effect of combining notions of more advanced set theories with the fuzzy set theory. Moreover, a recent study by Zeraatkar and Afsari (2021) suggested two novel extensions of the FKNN based on the notions of interval-valued fuzzy (IVF) sets, intuitionistic fuzzy (IF) sets, and a resampling technique called SMOTE (Kotsiantis et al., 2006) for imbalanced data classification problems. With the new variants, SMOTE-IVF and SMOTE-IVIF, they mainly focused on improving the performance of minority class classification.

Another compelling method of improving the performance of the FKNN classification is to adopt reasonable data pre-processing methods (Derrac et al., 2014). Among them, the prototype generation and prototype selection (Triguero et al., 2012) techniques have been introduced in the data pre-processing steps to enhance the performance of the

classical FKNN method (Derrac et al., 2014). Regarding prototype generation, the study by Keller et al. (1985), besides the original FKNN method, also proposed the fuzzy nearest prototype classifier (FuzzyNPC) as the first prototypical variant of the original method. It functions by employing only one prototype per class, which is generated by taking the mean of the instances in each class. However, though this method is faster in terms of processing, it is less accurate than the original FKNN method (Keller et al., 1985; Derrac et al., 2014). Here, it should be noted that the term "prototype" is often used instead of "instance" (or neighbor) in the context of KNN-based classification. The fuzzy-edited nearest neighbor (FENN) rule (Yang and Chen, 1998) and the condensed fuzzy k-nearest neighbor (CFKNN) rule (Zhai et al., 2011) are example cases for prototype selection-based approaches.

Figure 1.1 proposes a taxonomy of the FKNN variants discussed above, in line with study by Derrac et al. (2014). The FKNN algorithms can generally be categorized into classification or regression methods. In the second level of the taxonomy, classification techniques are further divided into sub-categories depending on the central adopted concept. The methods in the same sub-category are further differentiated by introducing the third level. Detailed information on these categories and properties can be found in Derrac et al. (2014) who presented a comprehensive review of available FKNN-based methods. As indicated in Figure 1.1 (in yellow), the emphasis of our research is on enhanced FKNN variants using class prototype multi-local mean vectors to perform binary and multi-class classification. The use of multi-local mean vectors for the FKNN classification was motivated by previous studies (Mitani and Hamamoto, 2006; Gou et al., 2014; Pan et al., 2017) that have used the concept of local means to achieve reasonable results with the KNN method. Moreover, this dissertation focuses on generalizing the FKNN algorithm in the context of regression, which has not received considerable attention until now. It is noteworthy that the properties illustrated in Figure 1.1 might be very useful to understand how the existing methods work and how the contributions of this research can be distinguished from them as well as possible links to developments in future research.

The research underlying this dissertation focuses not only on developing enhanced fuzzy k-nearest neighbor classifiers (Publication **I**, **III**) but also on applying implemented methods to real-world data (Publication **II**). Regarding the application, it also addresses a part of feature selection (Publication **II**) because the methodology involved in this study is created by incorporating the new classifier with selected feature selection algorithms. In addition, this research has a theoretical contribution in demonstrating how the FKNN algorithm can be adapted in the regression setting (Publication **IV**). Table 1.1 provides a summary of the objectives and contents of publications related to this research.

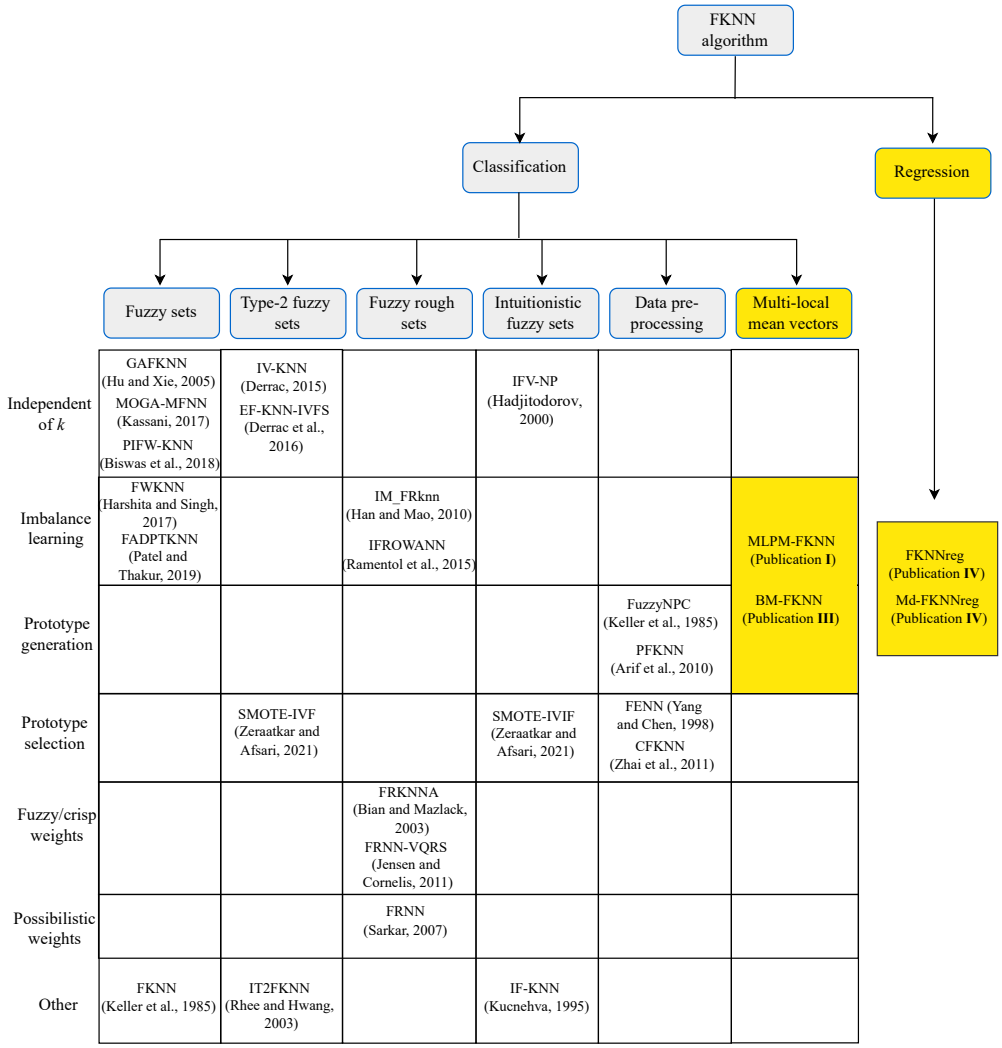


Figure 1.1: A taxonomy of the FKNN methods [modified from Derrac et al. (2014)].

Table 1.1: An overview of related publications.

	Publication **I**	Publication **II**	Publication **III**	Publication **IV**
Publication title	"An enhancement of fuzzy k-nearest neighbor classifier using multi-local power means"	"A study on relevant features for intraday S&P 500 prediction using a hybrid feature selection approach"	"A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean"	"A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance"
Objectives	To introduce a multi-local mean computation to the FKNN algorithm, to develop an enhanced FKNN variant based on the Power mean-based multi-local mean vectors, and to study the effectiveness of the proposed method on several class imbalance problems	To develop a hybrid approach that uses a filter+wrapper feature selection and the MLPM-FKNN method to predict the intraday return of the S&P 500 index and to find a feature subset that can yield a more accurate stock market prediction	To develop an extension to the MLPM-FKNN variant, which uses the Bonferroni mean-based local mean vectors and to demonstrate the capability of this extension on the class imbalance problems outperforming selected baseline models	To establish FKNN-based regression, to propose a novel FKNN regression method that uses the Minkowski distance, and to demonstrate the ability of the new approach to outperform well-known state-of-the-art regression methods in terms of real data
Problem type(s)	Classification	Feature selection and classification	Classification	Regression
Key approaches	KNN, FKNN, Power means	MLPM-FKNN, KNN, FKNN, DEFS	KNN, LM-KNN, FKNN, Bonferroni mean, SVM, Similarity classifier	KNNreg, FKNNreg, Minkowski distance, SVR, MLR, LASSO
Data	Four real-world data sets	Real-world stock market data set	One artificial data set, six real-world data sets	Eight real-world data sets
Contribution	A novel MLPM-FKNN classifier and its effectiveness regarding class imbalance problems	A hybrid feature selection method and a significant pattern of influential features	A novel BM-FKNN classifier and its effectiveness regarding class imbalance problems	A novel Md-FKNNreg approach and its effectiveness regarding regression problems

1.3 Dissertation Structure

This dissertation consists of six chapters, including this introductory chapter that describes the research topic, background information, scope of the research, and objectives. Next, the second chapter examines relevant theoretical contexts. The subsequent chapters (three to five) establish the new approaches for classification and regression tasks based on the FKNN algorithm, and examine their performances in different applications. Finally, the closing chapter summarizes the contribution of each publication, the limitations of the proposed methods and results, and discusses possible future research directions. Table 1.2 summarizes the structure of the dissertation along with the title and contents of each section as well as the related publication(s) number.

Table 1.2: An overview of the included sections and their contents

Section	Contents	Related publication(s)
1. Introduction	Contextualization of research study, scope of the research, and a summary of objectives and contributions	
2. Literature Review and Methodology	Theoretical background of related mean operators, distance functions, KNN-based classifiers, and evaluation settings machine learning	Publications **I**, **III**, **IV**
3. Proposed FKNN classifier based on the Power mean	Introduction of the novel MLPM-FKNN classification approach using the Power mean and its application to real-world data sets from machine learning repositories and a real stock market data set	Publications **I**–**II**
4. Proposed FKNN classifier based on the Bonferroni mean	Introduction of the novel BM-FKNN classifier using the Bonferroni mean and its application to one artificial and several real-world data sets	Publication **III**
5. Proposed FKNN regression with the Minkowski distance	Introduction of the Minkowski distance-based novel regression approach Md-FKNNreg and its application to real-world data sets	Publication **IV**
6. Conclusions, limitations, and future work	Concluding remarks on the contributions, limitations of the proposed methods and findings, and discussions of possible future research directions	Publications **I**–**IV**

2 Literature Review and Methodology

This chapter introduces basic notations and definitions of the theoretical concepts and methods involved in this dissertation. In particular, mean operators, distance measures, related classification techniques, and evaluation settings are discussed.

2.1 Mean Operators

Aggregation is a method of fusing a significant amount of data into a single representative value. Aggregation operators allow us to manifest how data need to be fused (Torra, 2016). Many different types of mean operators have been introduced in the literature for averaging information. In this research, the Power mean (Bullen, 2003) and Bonferroni mean (Bonferroni, 1950) operators, which are introduced below, have been applied.

2.1.1 Power Mean Operator

Power mean (also known as root-power mean or generalized mean) is a family of means generalized from the arithmetic mean (Beliakov et al., 2016). As a particular case of the Power mean, well-known expressions can be generated by changing parameter p to one specific value. These special cases include, for example, harmonic mean ($p = -1$), arithmetic mean ($p = 1$), quadratic mean ($p = 2$), and cubic mean ($p = 3$). Meanwhile, when $p \to 0$, it approximates the geometric mean. A formal definition of the Power mean is presented below, as indicated by Bullen (2003).

Definition 2.1 (Power mean) *Let $X = (x_1, x_2, .., x_m)$, $x_i \geq 0 \ \forall i \in \mathbb{N}$ be a vector with at least one $x_i \neq 0 \ \forall i = 1, 2, .., m$ and $p \in \mathbb{R}$ be a parameter. The Power mean of X is defined as:*

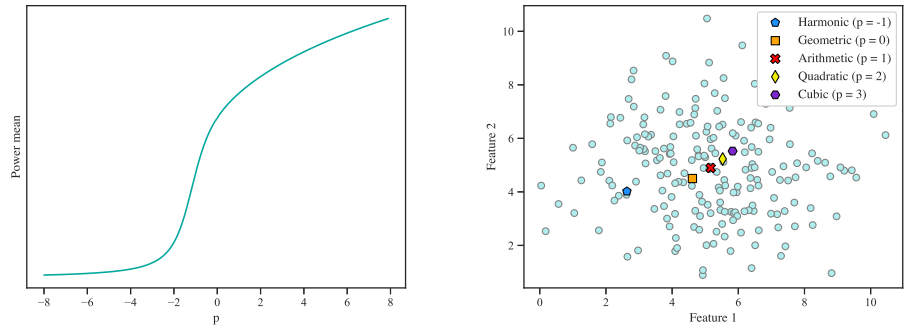
$$\mathcal{M}^p(X) = \begin{cases} \prod_{i=1}^{m} x_i^{1/m} & \text{if } \ p \to 0 \\ (\frac{1}{m} \sum_{i=1}^{m} x_i^p)^{1/p}, & \text{if } \ p \neq 0 \end{cases} \tag{2.1}$$

As seen above, an important property of the Power mean is that its value can be controlled by the exponent $p \in (-\infty, \infty)$.

Figure 2.1 displays the principal characteristics of the Power mean function for an example case. As depicted by Figure 2.1a, the Power mean operator can vary on a larger scale concerning the different values of p. Specifically, Figure 2.1b shows that the Power mean inherently changes its geometrical position with the different settings of p in the two-dimensional feature space.

2.1.2 Bonferroni Mean Operator

Bonferroni mean, another type of generalization of the arithmetic mean, offers a class of aggregation functions. This averaging operator was initially introduced by Bonferroni (1950), and later extended and discussed by several authors (see, for e.g., Beliakov et al.,



(a) (b)

Figure 2.1: A graphical illustration of the Power mean in an example case: (a) variation of the Power mean function vs. parameter p, (b) geometric positions of some special cases of the Power mean in two-dimensional space.

2016, 2007; Yager, 2009a,b). Additionally, Bonferroni mean and its extensions have been utilized in many practical applications (see, for e.g., Hait et al., 2022; Liua et al., 2019; Qin et al., 2020; Sun and Liu, 2013). The Bonferroni mean operator includes two parts—inner and outer. The product of one argument and the average of all the other remaining inner arguments make up each argument in the outer part, and this combination is what distinguishes it from others in terms of the aggregation (Beliakov et al., 2016). A formal definition of the Bonferroni mean can be presented as follows.

Definition 2.2 (Bonferroni mean) *Let* $X = (x_1, x_2, .., x_m)$, $x_i \geq 0 \; \forall i \in \mathbb{N}$ *be a vector with at least one* $x_i \neq 0 \; \forall i = 1, 2, .., m$ *and* $p, q \geq 0$ *be parameters. The Bonferroni mean of* X *is defined by Bonferroni (1950) as:*

$$\mathcal{B}^{p,q}(X) = \left(\frac{1}{m} \sum_{i=1}^{m} x_i^p \left(\frac{1}{m-1} \sum_{i,j=1, j \neq i}^{m} x_j^q \right) \right)^{\frac{1}{p+q}} \tag{2.2}$$

The Bonferroni mean offers a parameterized class of aggregation operators, including arithmetic, geometric, and harmonic means, to mention a few. In a certain sense, the Power mean can also be represented as a particular case in the Bonferroni mean. Moreover, the Bonferroni mean has gained significant attention among researchers and has been used in various applications (see. for example, Xu and Yager, 2011; Xia et al., 2013; Fabio et al., 2016; Kurama et al., 2016) due to its ability to allow multiple comparisons and perceive inter-relationships between input arguments (Wei et al., 2013). This dissertation uses the Power mean in Publication **I** and the Bonferroni mean in Publication **III**.

2.2 Distance Measures

Many machine learning algorithms require the quantification of similarity or dissimilarity between two objects (or instances) in the data (Mathisen et al., 2019; Aggarwal, 2015). In particular, measuring the similarity (in terms of "distance") is necessary for instance-based techniques[1] such as KNN. Here, it should be noted that a "similarity function" (where higher values indicate higher similarity) and a "distance function" (where smaller values indicate higher similarity) are distinct concepts for some domains (Aggarwal, 2015). Nevertheless, this chapter refers to the distance function as a methodological way of computing similarity.

Moreover, an ideal distance measure should be able to provide an accurate estimation of similarity between two data instances while allowing the researchers to comprehend how to compare, classify, or cluster those cases (Bergamasco and Nunes, 2019). The distance functions are particularly susceptible to the data type, distribution, and dimensionality (Aggarwal, 2015). Therefore, the distance measure can significantly affect the function of the method used, and a poor choice of distance measures can weaken the outcomes. Hence, this research is not limited to one distance function and instead, it investigates the performance of KNN-based regression methods with several distance functions, which are defined in detail below.

Definition 2.3 (Euclidean distance) *The Euclidean distance between two points,* $X = (x_1, x_2)$ *and* $Y = (y_1, y_2)$ *in a plane is defined as* $d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ *according to the Pythagorean theorem (Kubat, 2017). This expression is easy to extend to an* m*-dimensional space. For the two given points* $X = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ *and* $Y = (y_1, y_2, ..., y_m) \in \mathbb{R}^m$*, the Euclidean distance (*d_{EUC}*) is defined as follows (Kubat, 2017):*

$$d_{EUC}(X,Y) = \sqrt{\sum_{j=1}^{m}(x_j - y_j)^2} \tag{2.3}$$

The Euclidean distance typically indicates a good compromise (Witten et al., 2011) and is the most often used distance metric for computing distances (Li et al., 2022). If it is not explicitly specified, the Euclidean distance is the default distance measure under normal conditions. Moreover, this distance has the property of being rotation-invariant because the straight-line distance between two data points remains unchanged even when the axis system is oriented (Aggarwal, 2015). This indicates that some data transformation approaches (e.g., PCA) can be applied to the data without influencing the distance (Aggarwal, 2015).

Definition 2.4 (Manhattan distance) *The Manhattan distance (also referred to as* L_1 *norm or city block distance) is defined by the sum of the absolute differences between*

[1] In machine learning, instance-based learning refers to modeling new instances (for example, predicting the value) directly based on training instances (Keogh, 2011).

two points for each dimension. For the two given points $X = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ *and* $Y = (y_1, y_2, ..., y_m) \in \mathbb{R}^m$*, the Manhattan distance (*d_{Man}*) is defined as follows:*

$$d_{Man}(X, Y) = \sum_{j=1}^{m} |x_j - y_j| \tag{2.4}$$

Definition 2.5 (Minkowski distance) *For the two given points* $X = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ *and* $Y = (y_1, y_2, ..., y_m) \in \mathbb{R}^m$*, the Minkowski distance (*d_{Md}*) is defined as follows (Lu et al., 2015):*

$$d_{Md}(X, Y) = \big(\sum_{j=1}^{m} |x_j - y_j|^{\mathcal{P}}\big)^{1/\mathcal{P}} \text{ for } \mathcal{P} \geq 1 \tag{2.5}$$

The Minkowski distance (also referred to as $L_{\mathcal{P}}$ norm space) is a class of different distance measures formed by the parameter $\mathcal{P}$ (Igual and Seguí, 2017). By changing the value of $\mathcal{P}$, various distance measures can be generated; for example, when $\mathcal{P} = 1$, Minkowski distance equals to the Manhattan distance [as in Equation (2.4)] and when $\mathcal{P} = 2$, it corresponds to the Euclidean distance [as in Equation (2.3)]. Therefore, utilizing Minkowski distance allows greater flexibility for establishing various distances on the selected cases. However, there is no conscious rule on the accurate selection of $\mathcal{P}$ and it should be chosen according to the application and based on a reasonable assessment (for example, see Lu et al., 2015). Thus, this research uses the Minkowski distance to design a new regression approach based on the FKNN classifier (which is associated with Publication **IV**).

2.3 *K*-nearest Neighbor Method

In instance-based classification, the similarity of each new instance to training examples is measured using a distance metric, and the class of the closest instance is assigned to the new one. This is known as "nearest neighbor" classification (Witten et al., 2011). When more than one nearest neighbor is considered (say k), the new instance is assigned to the majority class among the k closest neighbors. This is known as the *k*-nearest neighbor classification (Witten et al., 2011). Fix and Hodges (1951) first introduced the rationale of the KNN, and later Cover and Hart (1967) presented a more evolved version of it for classification problems. In the following section, a step-by-step process underlying the KNN algorithm for classification and regression is separately introduced.

2.3.1 KNN Classification

Let $\{X_j, c_j\}_{j=1}^{N}$ be a training set, formed by N instances and each $X_j = \{x_j^1, x_j^2, .., x_j^{\mathcal{D}}\} \in \mathbb{R}^{\mathcal{D}}$ in $\mathcal{D}$-dimensional feature space from S classes. The class of X_j is c_j, where $c_j \in \{\omega_1, \omega_2, .., \omega_S\}$. For a given query sample $Y \in \mathbb{R}^{\mathcal{D}}$, the class ω_c is determined by the KNN rule according to the following steps:

Step I of the algorithm is choosing a value for k (i.e., number of nearest neighbors), which is the most challenging aspect of using the KNN classifier (Aggarwal, 2015).

However, it is possible that a minimal value of k may not offer accurate classification results due to some inconsistencies and noise of the data (Liu et al., 2013; Gou et al., 2014). In contrast, an increased value of k can lead to poor performance in the classification in situations where there are effects of outliers, especially when the training sample is small (Pan et al., 2017; Gou et al., 2014). In practice, a suitable value of k is determined heuristically. More often, the KNN with different values of k is tested for accuracy using training data, and when the accuracy is at its best, corresponding value of k is chosen (Aggarwal, 2015). This technique reflects the cross-validation, but several other methods are also available in the literature (see, for e.g., Ghosh, 2006).

Step II consists of measuring the similarity between Y and all training instances to identify the nearest neighbors of Y. The natural way of doing this is to calculate the Euclidean distances from Y to all training instances, and it can be presented as follows:

$$d_{EUC}(Y, X_j) = \sqrt{(Y - X_j)^T (Y - X_j)} \tag{2.6}$$

where $d_{EUC}(Y, X_j)$ is the Euclidean distance between Y and X_j for $j = 1, 2, .., N$. It is noteworthy that even though the Euclidean distance is the most commonly employed in KNN to measure the distance between two data instances, it is not always the best choice for every practical problem (Cai et al., 2020; Nguyen et al., 2016). Several studies (for examples, see Dettmann et al., 2011; Luukka, 2011) presented better classification results with different distance metrics, such as the Manhattan or Minkowski distances.

Step III is to find the k nearest neighbors for Y from the sorted training instances according to increasing Euclidean distances. In this step, the classifier searches for the nearest training instances using the user-defined (or optimized) value of k and their corresponding class labels.

Step IV assigns Y to the class ω_c that is represented by the majority of the k nearest neighbors. When $k = 1$, Y is classified into the class of its closest neighbor among all training instances.

Figure 2.2 illustrates a simple classification problem using the KNN classifier. In the figure, data instances in a two-dimensional space are represented by circles and squares belonging to two classes, A and B. A plus sign (+) represents the new instance (Y), and the KNN is used to predict the class (A or B) to which Y belongs. In this example, the KNN classification is considered with two cases. When $k = 3$, the new instance, Y, is classified into class A by identifying the classes of the three nearest neighbors. Similarly, the class of Y is determined using the five nearest neighbors and is predicted as class B when $k = 5$.

Moreover, the KNN algorithm is efficient in its simplicity of implementation and speed of the process (Pan et al., 2017; Gou et al., 2014; Wu et al., 2021). It is also flexible in terms of capability in dealing with complex data where the relationships may not be

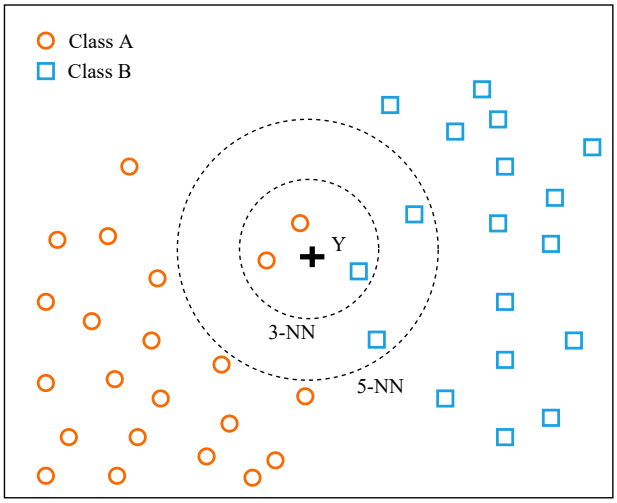


Figure 2.2: An illustrative example of the KNN classification

readily observable (Winters-Miner et al., 2015). Given these advantages, the rationale of the KNN classifier is also introduced for regression problems, and the *k*-nearest neighbor regression (KNNreg) method (Stone, 1977; Benedetti, 1977; Turner, 1977) is proposed. Unlike the other regression methods, the KNNreg method makes no assumptions about underlying data (Yao and Ruzzo, 2006; Winters-Miner et al., 2015) or model components; instead, it simply uses the training data for predictions. Now how it typically works for regression problems is presented below.

2.3.2 KNN Regression

The basic aim of the KNNreg method is to calculate an output value of the given input instance using the k number of its nearest neighbors, which is found from the input-output training instances (Biau et al., 2012; Hu et al., 2014; Song et al., 2017). In general, this method does not require an explicit training step but an initial dataset shaped by the values of input features and a response variable (Runkler, 2016). Even though the KNN rule is described for both classification and regression under the same umbrella, they differ from each other in that the classifier requires an output variable with discrete (categorical) values, while the regressor requires one with continuous values. Therefore, the step-wise process of the KNNreg algorithm is defined using a different notation as follows.

Let $\{X_j, y_j\}_{j=1}^{N}$ be a training set with N instances, where $X_j = \{x_j^1, x_j^2, .., x_j^{\mathcal{D}}\} \in \mathbb{R}^{\mathcal{D}}$ is the j^{th} instance from $\mathcal{D}$-dimensional input-feature space, and its output value is $y_j \in \mathcal{Y}$, where $\mathcal{Y} = \{y_1, y_2, .., y_N\}$ indicates the set of output values (i.e., response variable). Given a new data instance, say X, the goal of the KNNreg is to learn the predictor function $h(X)$ using the training data such that $\hat{y} \approx h(X)$, where $\hat{y}$ is the predicted value for the output y of X.

In the KNNreg algorithm, the set of nearest neighbors, $\{(X_j, y_j)\}_{j=1}^{k}$ of X is found as described in Steps **I**-**III** of the KNN classifier. As the last step (which is the modification of the classification rule), the output value y of X is estimated by taking the arithmetic average of the output values $(y_1, y_2, .., y_k)$ of the nearest neighbors (Györfi et al., 2002; Biau et al., 2012; Song et al., 2017) as follows:

$$\hat{y} = \frac{\sum_{j=1}^{k} y_j}{k} \tag{2.7}$$

This calculation is based on the assumptions that all training instances in the neighborhood have equal importance in the prediction (Cover and Hart, 1967) as well as similar output values to $h(x)$ (Kramer, 2011).

So far, all the essential steps of the KNN algorithm for classification and regression and its advantages have been explained in several different ways. Despite the breakthrough it has enabled, it is necessary to note that this method generally has several shortcomings, as mentioned in the introduction. One potential weakness of the KNN method is that it, by default, gives equal importance to all nearest neighbors in the classification (Keller et al., 1985; Pan et al., 2017), disregarding the fact that different points may have different effects on the query sample (Wu et al., 2021). For example, let us consider the case when $k = 5$ in Figure 2.2, in which the goal was to identify which class the instance Y belongs to by choosing five nearest data points. When $k = 5$, the KNN classifier identifies Y as a square because most of its nearest neighbors (three out of five) represent class B. However, this classification appears to be inaccurate. The three nearest circle points are relatively far away from Y, and thus they may not be entitled to the same level of influence as the other two circle points in Y's neighborhood. To address this problem, a weighted voting mechanism (Royall, 1966) is introduced to the KNN algorithm, and a weighted-KNN algorithm (Bailey and Jani, 1978) is proposed.

In the weighted-KNN algorithm, a weight for each nearest neighbor is calculated disproportionally to its distance from the query sample Y, accounting for the closer instances and the higher impacts (Kubat, 2017). The simplicity and the robustness of the weighted-KNN algorithm has motivated researchers to construct a variety of improved methods (see, for e.g., Li et al., 2022; Rastin et al., 2021; Biswas et al., 2018) and employ it for many applications (see, for e.g., Lei and Zuo, 2009; Bugata and Drotár, 2019; Kuang et al., 2019). However, it is clear that the performance of this method vastly depends on the applied distance metric (Rastin et al., 2021), class imbalance (Tan, 2005), and also noisy and redundant features (Biswas et al., 2018), which reflect the potential disadvantages of this method. Therefore, different weighting schemes need to be investigated. In this respect, the FKNN approach (Keller et al., 1985) can be introduced as an effective enhanced variant, which utilizes distance-based weights through the fuzzification. The FKNN approach assumes that a particular instance may belong to more than one class, which is opposite to the rule of non-fuzzy approaches. This way, the FKNN

aims to incorporate the uncertainties associated with the data points. For example, there can be situations where it is difficult to choose the class of an unknown sample due to the similarity of its feature information to each class information. In such cases, the fuzzy approach makes it easier to select a suitable class by allowing degrees of memberships to the unknown sample for each class. The FKNN approach is introduced in the next section.

2.4 Fuzzy *K*-nearest Neighbor Classifier

Keller et al. (1985) introduced the theory of fuzzy sets[2] into the original KNN rule to develop its fuzzy version and proposed the FKNN classifier. The underline idea of the FKNN method is that it assigns a membership degree to each class concerning the query sample (Y) and then predicts the class of Y based on the highest membership degree (Keller et al., 1985). Further, it allows partial memberships for a training instance to different classes and considers the relative importance of each nearest neighbor in terms of Y.

Implementing the FKNN algorithm for a given task involves five steps: parameter initialization, distance calculation, finding the k nearest neighbors, memberships computation, and class prediction. More specifically, the first three steps are essentially the same as Steps **I**-**III** in the KNN algorithm presented in Section 2.3. However, the FKNN algorithm uses an additional parameter called fuzzy strength value, which needs to be initialized along with k, the number of nearest neighbors in Step **I**. This parameter and the next steps are explained in detail below.

Step IV consists of assigning membership degrees, first for the nearest neighbors to each class and next for the query instance. This membership degree implies the proportion to the nearest neighbor or query instance belonging to each of the possible classes.
There are two types of methods that are often used for measuring the class memberships for nearest neighbors: crisp membership, in which a full membership is assigned to the known class and zero memberships to other classes; and fuzzy membership, in which the memberships are assigned according to the KNN rule (Keller et al., 1985; Chen et al., 2011; Huang and Li, 2004). If $u_{ij}(X_j)$ represents the membership of the j^{th} nearest neighbor $X_j \in \mathbb{R}^{\mathcal{D}}$ to the i^{th} class for $i = 1, 2, ..., S$, the crisp labeling approach can be expressed as:

$$u_{ij}(X_j) = \begin{cases} 1, & \textbf{if } \ c_i = c_j \\ 0, & \textbf{if } \ c_i \neq c_j \end{cases} \tag{2.8}$$

[2]Fuzzy set theory, an extension of the classical sets theory, was introduced by Zadeh (1965) to deal with associated uncertainties, impreciseness, and vagueness in the information in decision-making problems. A crisp set is typically expressed as a set of elements, and each element can either be, for example, 0 or 1. In contrast, the fuzzy set theory allows the element to have a degree of membership in a set, which is a value in a closed interval $[0, 1]$. Since 1965, many researchers have advanced this theory and used it to a wide range of application areas (see, e.g., Wong and Lai, 2011; Wang, 2009; Park et al., 2012; Mesiar et al., 2013; Derrac et al., 2014).

where c_j is the class of X_j for $j = 1, 2, ..., k$. The fuzzy labeling approach first finds the K closest neighbors to each instance X_j (the nearest neighbor of the query sample), and then allocates the degree of membership in each class as follows:

$$u_{ij}(X_j) = \begin{cases} 0.51 + (n_i/K) * 0.49, & \textbf{if } c_i = c_j \\ (n_i/K) * 0.49, & \textbf{if } c_i \neq c_j \end{cases} \tag{2.9}$$

where n_i indicates the number of observed neighboring instances of X_j belonging to the i^{th} class. This fuzzy approach attempts to grade the closer neighbors with higher memberships and the farther ones with low memberships for each class region (Chen et al., 2011; Derrac et al., 2015).

However, it should be noticed that in both approaches, each neighboring instance is considered not limited to one class but belonging to multiple classes with different memberships. In this way, considering each nearest neighbor has only one class, as in the KNN method, is avoided because in reality, it is not always the case (Wu et al., 2021). Once the class memberships of the nearest neighbors are generated, a membership degree $u_i(Y)$ to the query sample $Y \in \mathbb{R}^{\mathcal{D}}$ in class i is then calculated according to:

$$u_i(Y) = \frac{\sum_{j=1}^{K} u_{ij}(1/\left\|Y - X_j\right\|^{2/(r-1)})}{\sum_{j=1}^{K}(1/\left\|Y - X_j\right\|^{2/(r-1)})} \tag{2.10}$$

where $r \in (1, +\infty)$ is the fuzzy strength parameter, which is used to control the relative importance of the distance to be weighted, when determining the neighbors' contributions to the membership value. The most often used value is $r = 2$ (Derrac et al., 2015), but it should be noted that the performance of the FKNN classifier can be improved by optimizing the r through various settings (see, for e.g., Jiang et al., 2008). Moreover, $\|Y - X_j\|$ is the Euclidean distance between Y and its j^{th} neighbor X_j. As can be seen from Equation (2.10), the inverse of the distances assigns the membership of nearest neighbors to be small if it is far and high if it is close. Further, it is noteworthy that when compared to the KNN algorithm, **Step IV** of the FKNN is an extra step, which causes a slight increase in computational complexity.

Step V is the classification of Y into class ω_c with the highest membership degree among all other classes. This can be expressed as:

$$\omega_c = \arg \max_{\omega_i} u_i(Y), \ i = 1, 2, .., S \tag{2.11}$$

The FKNN method described in this section is the baseline of invented classification and regression methods in this research.

2.5 Local Mean *K*-nearest Neighbor Classifier

The local mean k-nearest neighbor (LM-KNN) (Mitani and Hamamoto, 2006) is a simple and reliable extension of the original KNN classifier. It is designed to overcome the problem of existing outliers, particularly in situations where the training-set sample size is small (Mitani and Hamamoto, 2006; Pan et al., 2017). This classifier uses a local mean vector of k nearest neighbors from each class to predict a correct class for the query sample. The step-by-step process underlying the LM-KNN algorithm when predicting the class for a given query sample $Y \in \mathcal{R}^{\mathcal{D}}$ can be presented as follows.

Step I and **Step II** of the LM-FKNN algorithm are similar to the corresponding steps of the KNN method presented in Section 2.3. However, this algorithm searches for k nearest neighbors of Y from each class.

Step III identifies the set of k nearest neighbors of Y from each class i by ordering the training instances corresponding to the increasing distances. Let $X_{i,j}^{nn}$ represents a set of k nearest neighbors (nn) in class ω_i for $i = 1, 2, .., S$.

Step IV consists of the calculation of a local mean vector $\mathcal{L}_i$ for each class ω_i for $i = 1, 2, .., S$ using the corresponding set of k nearest neighbors. It can be denoted as:

$$\mathcal{L}_i = \frac{1}{k} \sum_{j=1}^{k} X_{i,j}^{nn} \tag{2.12}$$

Step V classifies Y into the class ω_c, in which the representative local mean vector $\mathcal{L}_i$ has the lowest Euclidean distance to Y. In other words,

$$\omega_c = \arg \min_{\omega_i} d_{EUC}(Y, \mathcal{L}_i), \ i = 1, 2, .., S \tag{2.13}$$

Moreover, it has been demonstrated that the LM-KNN rule has the potential to overcome the adverse effects on the KNN-based classification from the existing outliers in the data (Mitani and Hamamoto, 2006). However, this method has several inherent weaknesses, such as choosing the neighborhood size k for each class (Pan et al., 2017) and assigning equal contribution for each neighbor in the classification (Gou et al., 2019). These issues have prompted researchers to develop a range of improved classifier versions (see, example, Gou et al., 2014; Chai et al., 2010; Sumet et al., 2018; Pan et al., 2017; Gou et al., 2019). Thus, this dissertation has employed the rationale of the LM-KNN method in Publications **I** and **III**, not the exact logic but a somewhat similar one.

2.6 Evaluation Settings

The key to making genuine progress in data mining and machine learning is "evaluation" (Witten et al., 2011). From this perspective, the goal of evaluation is to not only assess the model's performance but also justify the selection of a particular algorithm. There

are many ways to undertake data modeling and analyzing. However, different methods search for different patterns and tendencies, which implies that one method cannot be the best for all data sets. To achieve the best possible result, the best possible settings have to be set and followed at the time of model construction and evaluation. In this sense, the question to be answered is how to determine the most appropriate settings. It may indeed depend on various factors, including data properties, expected solutions, parameters, performance levels, and efficiency, to mention a few.

In this section, the evaluation of classifiers is referred to as a broader concept, which covers the whole process from the data processing step to the evaluation of the results. Figure 2.3 shows the applied process of developing and evaluating a supervised classifier in this research. It consists of four essential steps: "Data preparation," "Data sampling," "Model training," and "Performance evaluation." All these steps are discussed in detail in the following sections. Here it should noted that our problem has to obviously be first defined and the available data has to be identified before the process can be started.

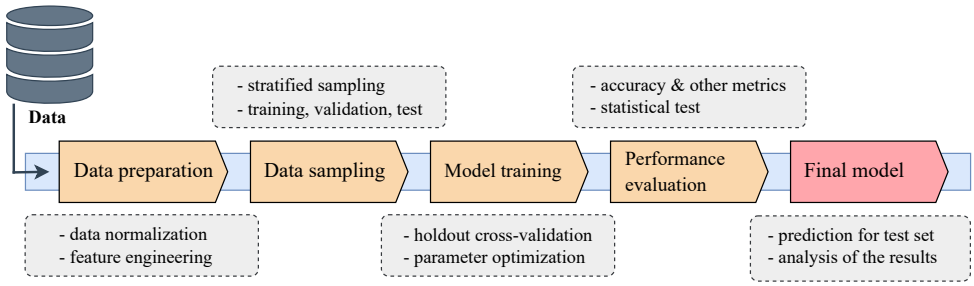


Figure 2.3: A flowchart of the development and evaluation of a supervised classification model.

2.6.1 Data Preparation

Data usually come from multiple sources with different characteristics. From this perspective, data may emerge with some issues such as inconsistencies, incompleteness, noise, and missing information (Trousse and Tanasa, 2004). Therefore, data preparation is required before applying machine learning techniques (Aggarwal, 2015). In particular, data pre-processing is an essential step in preparing the data in a usable and understandable form. It has several necessary steps, including data cleaning, data scaling, feature extraction or selection, or both (García et al., 2016). Specifically, data cleaning is the process of detecting and repairing (or removing) incorrect, incomplete, duplicate, irrelevant, or poorly formatted data (Han et al., 2012). Thus, data must be cleaned carefully to avoid such issues before starting any step in the mining and learning process. For example, if a data set contains missing values, it is necessary to either correct (or impute) them using a suitable approach or remove them entirely from the data (if there is sufficient data).

Data Scaling

In many situations, features in the data are described using different scales. For instance, a feature such as "salary" can be presented on a disparate scale as compared to one such as "age." This may result in the large-scale feature being considered with high weights while others are with low weights, which affects the performance of the applied machine learning technique, especially in distance-based learning methods such as KNN (Kubat, 2017). Consider a case where the similarity between two points, $A = (0.5, 342)$ and $B = (0.2, 450)$ and they are characterized by two continuous features: the first one holds the values from the interval $[0, 1]$ and the other from $[10, 1000]$. Using Equation (2.3), the distance between A and B can be obtained as $d_{EUC}(A, B) = \sqrt{(0.5 - 0.2)^2 + (342 - 450)^2}$. From this indication, it is clear that the second feature completely dominates, making the other irrelevant. To remedy this problem, there are several methods available, but it is common to use data normalization (Aggarwal, 2015; Witten et al., 2011). The idea behind the normalization is to transform data features in different scales into pre-defined intervals, usually the unit interval $[0, 1]$. In this context, min-max normalization (Shalabi et al., 2006) is one of the most frequently used approaches[3] in data scaling and assures a fast generalization through distance-based learning, especially for KNN-based models; it is thus used in this research. The formula of the min-max normalization technique can be defined as follows:

$$x_i^{scaled} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{2.14}$$

where x is the i^{th} value of vector $X \in \mathcal{R}^{\mathcal{D}}$ and its scaled value is x_i^{scaled}. The minimum and maximum of the vector X are indicated by $min(X)$ and $max(X)$. When $x_i = min(X)$ the scaled value is 0 (i.e., $x_i^{scaled} = 0$), and when $x_i = max(X)$, the numerator is 1; thus, this method always makes the scaled values fall within the interval $[0, 1]$. Moreover, min-max scaling is specifically helpful in algorithms, such as neural networks or KNN, which do not explicitly assume any distribution of the data used (Shalabi et al., 2006). However, it may be worth mentioning that this method is not effective when the minimum and maximum values are represented by extreme outliers (Aggarwal, 2015). In such cases, using further treatments on the data, it is essential to ensure that the data distribution is not influenced by the extreme outlier values (Namata et al., 2009). Here, it is noteworthy that scaling the class variable in terms of the classification problem is unnecessary.

Feature Extraction and Feature Selection

In parallel with the data cleaning and scaling, a dimensionality reduction step that commonly incorporates feature extraction or feature selection (Guyon and Elisseeff, 2003; Li et al., 2017; Kubat, 2017) is often performed. In the feature extraction process, original data features are projected into lower-dimensional feature space across a specific criterion, resulting in a new set of features that is less (in terms of the number of

[3]The most commonly applied data scaling methods in the literature are z-score standardization, decimal scaling, and min-max normalization [more information can be found in Han et al. (2012)]

features) than the initial one (Liu and Motoda, 1998). A standard and frequently applied feature extraction technique is the principal component analysis (PCA), in which a small-appropriate feature subset is extracted through a transformation of the original features (Liu and Motoda, 1998; Chen and Hao, 2018). In contrast, feature selection is a process of selecting a subset of available features based on a specific evaluation (e.g., by using classification accuracy or error rate) (Liu and Yu, 2005). Using the feature selection allows us to remove irrelevant and redundant features, which may cause model complexity, over-fitting issues, and poor performance (Runkler, 2016; Kubat, 2017). However, feature selection is a broader area of the research and may involve many advanced algorithms or methodologies in the form of filter, wrapper, or embedded methods; more information can be found in Liu and Motoda (1998); Guyon and Elisseeff (2003); Chandrashekar and Sahin (2014); Aggarwal (2015); Runkler (2016).

Notably, feature normalization, extraction, or selection is not always a part of data preparation because these processes depend highly on the particular problem being investigated and are also up to the analyst's choice. In this dissertation, a hybrid feature selection approach is performed in combination with filter and wrapper methods as in Publication **II** to discard linearly associated and irrelevant features from the data and achieve improved classification performance.

2.6.2 Data Sampling

In the developing phase of the model, original data are usually divided into two independent sets, often referred to as training and test sets (Bramer, 2016; Aggarwal, 2015). More specifically, a data set used to build the model is called the training data set. The performance of the developed model needs to be evaluated with another data set, called the test data set, which must have no part in developing and training the model (Witten et al., 2011; Bramer, 2016). In such a case, the model uses pre-defined parameters. However, that is not always the case as sometimes parameter values have to be selected through an iterative optimization process (Wendler and Gröttrup, 2016). In order to achieve optimal parameter values for the model, a third independent data set is required, which is called validation set (Witten et al., 2011). Therefore, the initial data set is split into three disjoint sets: training, validation, and test sets. This is the standard way of data sampling in machine learning (Igual and Seguí, 2017; Lee et al., 2019).

Particular to supervised classification, the training data are pre-classified examples, meaning that each instance in the data has been labeled with a specific class. Hence, the classifier uses the training data for learning to estimate the class label for a new record. Moreover, validation data is employed to choose the model parameters and structure with the best performance in terms of the accuracy or error rates. However, it should be noted that training and validation are often jointly performed in an iterative process that is often referred to as cross-validation, which will be further discussed in the next section. Once the model is found with optimal parameters and structure, its performance (i.e., capability of generalizing) is evaluated using the test data set.

Training, validation, and test sets are randomly created from the original data [in most cases, the major part of the data is used for training data, while the rest is used for validation and test data (Aggarwal, 2015)]. However, with random sampling, generated subsets might not be well-representative in terms of the classes available in the original data. In other words, there is no guarantee that each subset contains the same proportion of the classes as that of the original data set. This issue heavily affects situations where the classes presented in the data are not balanced (Kubat, 2017). For instance, suppose a data set consists of 100 instances, in which 90 cases belong to one class and 10 to another class. In this case, it is possible to originate a validation set of 20 instances without even a single data instance from the minority class. In such situations, a classifier may indeed yield inaccurate results, affecting the estimation of performance measures in the classification. Thus, it is necessary to perform random sampling to ensure that each class has an equal proportion in all training, validation, and test sets. The most popular way of dealing with this issue is using the so-called stratified sampling approach (Witten et al., 2011; Kubat, 2017), which always offers a rich coverage of the original data in divided data sets concerning class distributions.

2.6.3 Model Training and Validation

The previous section discussed model training with its core ideas of data sampling, model building, and parameter tuning. This section further explores how the data division and parameter estimation are performed with the cross-validation process. More specifically, cross-validation is a commonly used resampling technique to assess the generalization capability of the model being tested while mitigating the over-fitting issues (Berrar, 2019). There are several types of cross-validation techniques—the most widely used methods are holdout, k-fold, and leave-one-out cross-validation methods.

First, the holdout cross-validation is a simple approach in which a certain amount of a given data is held for validation (sometimes it is also referred to as test), while the rest is used for training, and the process is repeated multiple times to reduce the sample bias and variation of the performance (Arlot and Celisse, 2010; Han et al., 2012). The classification accuracies (and other performance measures) with all iterations are then averaged to achieve a final accuracy value. Using this method, variance (or standard deviation) can also be determined, which can be beneficial in making a statistical evaluation of the performance (Aggarwal, 2015).

Second, in k-fold cross-validation (Bramer, 2016; Runkler, 2016), a given data set is randomly split into k mutually exclusive and approximately equal-sized folds. The training and validation process is then repeated k times. Each time, one fold is used as the validation set, while the rest $(1 - k)$ are used as the training set. By simply taking an average of the set of k individual accuracy values, the cross-validation next produces an overall measure of the model performance (Bramer, 2016). five-fold and 10-fold cross-validations are the most common choices of this approach (Lever et al., 2016).

Third, the leave-one-out cross-validation (Bramer, 2016; Runkler, 2016), another common variant of cross-validation, effectively works in the same way as the *k*-fold method, but instead of dividing data into several subsets, it uses one instance of the data for validation and the remainder for model training. This approach, therefore, is also referred to as the *n*-fold method where *n* indicates the number of instances in the data set (Witten et al., 2011). However, this approach cannot be stratified because only a single instance is included in the validation set. In the present study, to tune the parameters of the classifier and assess its generalization capability, the holdout method with 30 runs was preferred. Example studies that have also used 30 runs are Luukka (2011); Koloseni et al. (2013); Kurama et al. (2016). Figure 2.4 shows a pictorial depiction of this cross-validation process along with the model testing on unseen (test) data.

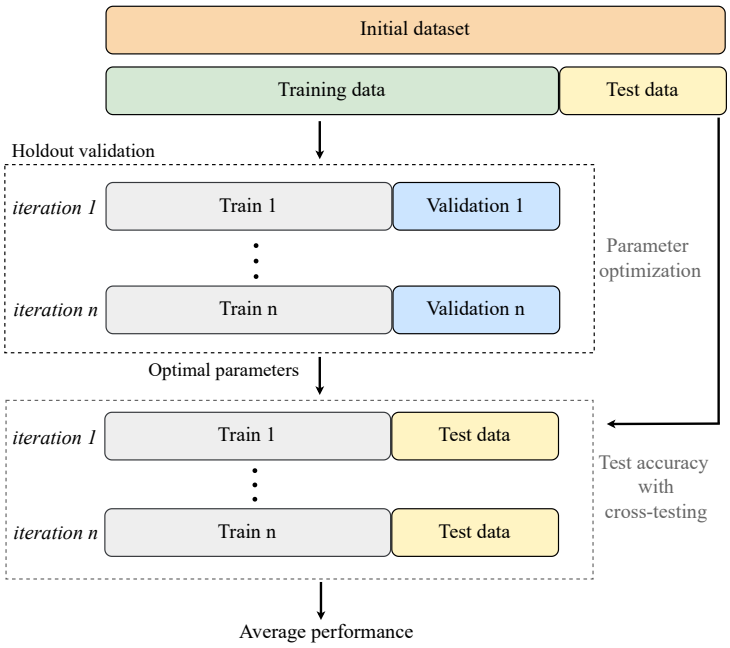


Figure 2.4: A taxonomy of the adapted validation process with the holdout method and model testing.

Using the empirical design shown in Figure 2.4, a particular model (an invented classifier or regression model for this research, or existing one) is validated and tested in the iterative process of $n = 30$ runs. At the start, the entire data set is partitioned with a typical $80/20$ split (Lever et al., 2016) for training and test data sets. In the holdout validation process, the initial training data set is further divided into training and validation (here, a $50/50$ split is often adopted). Moreover, the stratified random sampling approach is used to maintain the class balance of every sampling step. In each iteration, the classifier is tested with different parameter settings in the validation data, and its performance in

terms of accuracy and other measures are calculated. After the training and validation, the final model is set with the optimal parameter settings by evaluating the validation results in the average form.

In the testing phase, the predictive performance of the fitted model is evaluated using the test data that were initially separated from the original data set. Subsequently, stored training samples from the holdout cross-validation for the model test were used, effectively assessing the model performance in multiple testing times (which is precise to the number of training and validation runs). In this way, it is ensured that instances in the validation set are only used to tune the parameters and obtain an unbiased evaluation of the prediction in the test set by minimizing the model over-fitting/under-fitting issues.

It is noteworthy that parameter optimization in the training and validation step is conducted through the grid search method. Grid search requires a pre-defined set of values for each parameter involved and creates a class of trials by making every possible combination of those values (Bergstra and Bengio, 2012).

2.6.4 Performance Evaluation

Performance evaluation refers to how the results of the applied models are evaluated. It measures the predictions generated by the trained method on the test data. There are many performance measures, which are typically specified in different categories relying on the problem context, such as classification, clustering, and regression. In the present study, the evaluation metrics chosen in the classification and regression analysis are discussed in detail below.

Related performance measures for classifiers

As mentioned previously, the basic and the most frequently used performance measure for a classifier is accuracy (Rhee and Hwang, 2003; Chen et al., 2011; Derrac et al., 2015), which expresses the percentage of correct predictions concerning the total number of tested instances (Bramer, 2016). Specifically, the accuracy is the complement of the error rate that represents the misclassified instances from all classes. This metric can be computed as $1 - accuracy$. However, in practice, measuring the classification performance with accuracy (or error rate) alone is often not enough to verify a particular classifier is appropriate for a given problem. Further, these metrics do not explicitly describe the cost of wrong classifications or wrong decisions, which may lead to false conclusions (Witten et al., 2011). Another issue with the accuracy is that it challenges comparing the performance of several classifiers for a particular problem because they may produce the same accuracy but perform differently in terms of correct and incorrect classifications (García et al., 2010). Moreover, there might be cases where the number of correct predictions of a particular class differs from that of another class. In such cases, accuracy may not be a reasonable metric (Igual and Seguí, 2017) and additional performance measures and detail analysis are required. Given these issues, a confusion matrix al-

lows us to derive different essential measures to describe the performance of the classifier.

The confusion matrix is generated using the actual classes of test data and classifier predictions. It often engages with binary class and multi-class classification problems.

Binary class problems. There are only two classes in the binary classification, and suppose that one is a positive (P) and the other is a negative (N). In this case, there are four possible outcomes from the classification model, which can be represented by the elements of a 2×2 contingency table (confusion matrix) as in Table 2.1.

Table 2.1: A typical example of the 2×2 confusion matrix for a binary class problem.

		Actual class	
		Positive (P)	Negative (N)
Predicted class	Positive (P)	TP	FP
	Negative (N)	FN	TN

Here, it can be seen that true positive (TP), true negative (TN), false positive (FP), and false negative (FN) are elements of the table, while P and N in the rows are formed by the predicted class and P and N in columns are formed by the actual class. TP and TN represent the correct classifications (represented by a gray diagonal), while false positive (FP) represents the outcome when it is incorrectly determined as positive when it is truly negative. Meanwhile, false-negative (FN) represents the outcome when it is incorrectly determined as negative when it is truly positive. Using the elements of this confusion matrix, the performance measures, accuracy, sensitivity, specificity, and precision can be computed as (Tharwat, 2020; Wendler and Gröttrup, 2016):

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.15}$$

$$\text{Sensitivity (or Recall)} = \frac{TP}{TP + FN} \tag{2.16}$$

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{2.17}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.18}$$

These metrics enable us to assess how often a classifier predicts a particular class (Igual and Seguí, 2017) and to gain better understanding of the performance of the used classifier. This can be shown using a simple example: consider a 2×2 confusion matrix with the elements $TP = 0$, $TN = 95$, $FP = 0$, and $FN = 5$, where accuracy $= 95\%$ and specificity $= 100\%$ and both precision and sensitivity are zero. Undoubtedly, the model has high accuracy, but having zero sensitivity implies that the corresponding model is too weak to be applied in real-world applications. Moreover, when dealing with classification problems using imbalanced class data, it is essential to be employed these

performance metrics[4] along with the accuracy measure (Tharwat, 2020; Igual and Seguí, 2017).

Multi-class problems. Consider a confusion matrix with S (> 2) classes, represented by $\{a_{l,m}\}_{l=1,m=1}^{S}$, where $a_{l,m}$ is an element of a row l and a column m for $l, m = 1, 2, .., S$ in the matrix. When $l = m$, $a_{l,m}$ indicates the number of samples classified correctly to the corresponding class, and when $l \neq m$ indicates the number of misclassified samples of class ω_l as class ω_m. Then, the number of TP, TN, FP, and FN for each class ω_i for $i = 1, 2, ..., S$ can be measured according to Tharwat (2020) as follows:

$$TP(\omega_i) = a_{i,i} \tag{2.19}$$

$$TN(\omega_i) = \sum_{\substack{l=1 \\ l,m\neq i}}^{S} \sum_{m=1}^{S} (a_{l,m}) \tag{2.20}$$

$$FP(\omega_i) = \sum_{m=1,m\neq i}^{S} (a_{i,m}) \tag{2.21}$$

$$FN(\omega_i) = \sum_{l=1,l\neq i}^{S} (a_{l,i}) \tag{2.22}$$

Now, these estimates can be used to calculate the sensitivity, specificity, and precision with the help of formulas (2.16), (2.17), and (2.16) for each class. The average of each of these measures is considered the final performance measure for the classifier as stated in Ferri et al. (2009); Flach (2012). Here, consider an example case of a three-class classification problem. Table 2.2 shows the resultant confusion matrix of the classifier. In this example, suppose that there are three classes, ω_1, ω_2, and ω_3, and each class has 31, 30, and 39 test instances, respectively.

Table 2.2: The resultant confusion matrix of a multi-class classification problem: an example case with three classes, ω_1, ω_2, and ω_3.

		Actual class		
		ω_1	ω_2	ω_3
Predicted class	ω_1	24	3	9
	ω_2	5	26	0
	ω_3	2	1	30

From Table 2.2, it is clear that $TP(\omega_1) = 24$, $TP(\omega_2) = 26$, and $TP(\omega_3) = 30$. Thus, the accuracy of the classifier is $(24 + 26 + 45)/100 = 0.80$. Additionally, $TN(\omega_1) =$

[4]MATLAB codes created for sensitivity, specificity, and precision for binary and multi-class classification problems in this research can be found at: https://github.com/MahindaMK/Specificity-Sensitivity-and-Precision-for-multi-class-classification-problems.

57, $TN(\omega_2) = 65$, and $TN(\omega_3) = 58$. According to Equations (2.17) and (2.18), the false positive in the class ω_1 is $FP(\omega_1) = 3 + 9 = 12$. Similarly, $FP(\omega_2) = 5$ and $FP(\omega_3) = 3$. The false negatives for each class, $FN(\omega_1), FN(\omega_2)$, and $FN(\omega_3)$ are 7, 4, and 9, respectively. Using these values, the per-class sensitivity can be computed for classes ω_1, ω_2, and ω_3 as $24/31 = 0.77$, $26/30 = 0.87$, and $30/39 = 0.77$, respectively. In a similar way, per-class specificity and precision can be calculated, and a single value of each sensitivity, specificity, and precision for the classifier can be obtained by simply taking the average of these measures for all classes. For example, the final sensitivity score of the classifier is $(0.77 + 0.87 + 0.77)/3 = 0.80$.

Related performance measures for regression analysis

Root mean square error (RMSE) and coefficient of determination (R^2) are the most commonly used approaches to evaluate the prediction performance of regression methods. Generally, lower RMSE and higher R^2 values reflect the better performance of a regression model (Pham, 2019).

Specifically, RMSE calculates the square root of the average of the square of the differences between the predicted and actual values of each data point. It is typically applied with supervised learning techniques, as it requires the actual value of each predicted data point. The mathematical expression of RMSE can be defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2} \tag{2.23}$$

where n is the number of instances in the test data, $\hat{Y}_i$ is the predicted value, and Y_i is the true value of the i^{th} test instance.

Meanwhile, R^2 is a proportion of the variability in the response variable, which is "explained" by the regression model in comparison to the mean (Kurz-Kim and Loretan, 2014). It is a statistical measure that indicates how closely the values of the regression model fit the data points in the response variable. The general formula of R^2 can be defined as:

$$R^2 = \big(1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}\big) \times 100\% \tag{2.24}$$

As Equation 2.24 depicts, percentages are considered for R^2.

2.6.5 The Test of Statistical Significance

In machine learning applications, it is expected that the performances of methods applied for a given problem will be compared, after which the final model will be chosen based on the highest performance. However, when it is required to show that one method outperforms another for a particular task, it is necessary to employ a statistical test of

significance and validate the claim of improved performance (Borovicka et al., 2012). The mean performances of the classifiers across all iterations from the validation and testing provide an excellent advantage in identifying the performance (e.g., accuracy) of one classifier as significantly better than the performance of another. This task is usually achieved by using a statistical analysis approach called the paired t-test or t-test (Witten et al., 2011).

To evaluate whether there is a statistically significant difference between the mean accuracies, $\bar{Z}_1$ and $\bar{Z}_2$ of two classifiers, the mean of the accuracy differences, ($\bar{d}$) such as $\bar{d} = \bar{Z}_1 - \bar{Z}_2$, can be considered by assuming the differences as per the Student's distribution with $m - 1$ degrees of freedom (where m is the number of different validation or test sets). Accordingly, the t-test is performed to verify whether the mean difference is significantly different from zero, which is defined as the null hypothesis (H_0). The test statistic t_{m-1} (the subscript $m - 1$ indicates the value of degree of freedom) for H_0 at significance level α is computed according to:

$$t_{m-1} = \frac{\bar{d} - 0}{\mathcal{S}/\sqrt{m}} \tag{2.25}$$

where $\mathcal{S}$ is the standard deviation of the differences. If the corresponding P-value of the test statistic is less than 0.05, the test suggests rejecting the H_0 in favor of the alternative hypothesis (H_1), and if the P-value is higher than 0.05, vice versa. More specifically, the t-test applied in this research involves comparing the performances between two different classifiers used during the validation and test. In other words, the t-test is utilized to test whether the target classifier (usually the proposed one) significantly outperforms the benchmark methods in terms of the mean accuracy in the classification (Chen et al., 2011; Borovicka et al., 2012).

Additionally, confidence interval can also be analyzed to assess the significance of the mean difference. In this case, if the measured confidence interval (CI) does not include the observed statistic, then H_0 is rejected at the significance level α, indicating that there is a $100(1-\alpha)\%$ CI that delivers the exact conclusion as the t-test (Heumann and Shalabh, 2016). For some analyses in this research, the confidence intervals are used to understand the statistical significance of the outcomes.

3 Proposed Fuzzy *K*-nearest Neighbor Classifier Based on the Power Mean

A novel FKNN classifier is developed and tested in this chapter. To begin, the motivation for the new classifier is described with a brief literature review. Next, the new method and its step-by-step process is discussed in the implementation section. Finally, how the new classifier can be used to classify data from real-world applications is demonstrated.

3.1 Class Imbalance and Related Issues

A classification data set is called "imbalanced" if its observations do not represent the classes in the target variable equally (in other words, the number of observations of one class is significantly different from that of another class) (Liu and Chawla, 2011). Moreover, learning from imbalanced data has been one of the most challenging problems in machine learning research (Liu and Chawla, 2011; Johnson and Khoshgoftaar, 2019). If the training set is imbalanced, the models may tend to over-classify the majority class because of its higher prior probability. Consequently, instances in the minority class are incorrectly classified more frequently than those of the majority class (Johnson and Khoshgoftaar, 2019). More specifically, in these situations, some performance measures, such as accuracy, are not necessarily reasonable measures of the classifier's performance (Kubat and Matwin, 1997). For example, consider a binary classification task with a negative class distribution of 0.5%. In this case, if the model classifies all samples in the positive class for query instances, it will achieve an accuracy of 99.5%. This implies that even a classifier with accuracy not far from 100% may be useless, and more suitable measures thus need to be used in such situations.

Meanwhile, the class imbalance problem exists in many real-world applications (Zhang et al., 2017b; Sun and Chen, 2021)—a typical example is medical diagnoses [such as thyroid data in the UCI repository (Dheeru and Taniskidou, 2017)] where the task of interest was detecting disease, but most of the patients were healthy (Johnson and Khoshgoftaar, 2019). Additionally, imbalanced classification has often been a considerable challenge for existing classifiers, especially KNN-based methods. The voting principle that uses the majority class from the k neighborhood leads to the poor performance of KNN methods in imbalanced class problems. Therefore, enhancing the performance of KNN-based classifications on imbalanced data has been an important topic of interest in machine learning research (Sun and Chen, 2021).

The existent techniques that have been introduced to deal with class imbalance problems belong to three categories[5]: data-oriented[6], algorithm-oriented, and hybrid methods

[5]A detailed analysis of these methods with examples can be found in Kotsiantis et al. (2006) and Krawczyk (2016).

[6]The algorithm-oriented methods are further categorized into generality- and specificity-oriented algorithms (Zhang et al., 2017b). Generality-oriented methods include abstract classifiers derived based on training data (for example, support vector machine and decision tree methods). In contrast, specificity-

(Kotsiantis et al., 2006; Krawczyk, 2016). The focus of data-oriented approaches is to decrease the degree of class imbalance using different sampling techniques (Krawczyk, 2016), while algorithm-oriented methods are the extensions and enhancements of standard classification algorithms that can effectively handle imbalanced class problems (Liu and Chawla, 2011). Finally, the hybrid approaches are derived by integrating both data- and algorithm-oriented methods (Kotsiantis et al., 2006; Krawczyk, 2016).

There are many enhanced KNN classifiers proposed in the literature for imbalanced classification problems. Most can be understood from the study by Sun and Chen (2021), in which a comprehensive survey of KNN methods for solving class imbalance problems is presented. However, while all the examined approaches aimed to achieve improved performance with the KNN on the imbalanced data, none considered the issue of uncertainties that often emerge with most data sets. In this regard, the FKNN (Keller et al., 1985) method that was designed to tackle data uncertainty issues can be a great alternative. In fact, little attention has been paid to further improve the performance of the FKNN classifier in terms of the imbalanced class problems. For example, to learn from the imbalanced data and enhance the performance of minority class classification, the IM_FRknn (Han and Mao, 2010), IFROWANN (Ramentol et al., 2015), and FWKNN (Harshita and Singh, 2017) methods have been proposed by introducing different settings to the standard FKNN algorithm. Although these methods have demonstrated better performance in the selected imbalanced classification problems, they may not be suitable for data where the features of one class cannot be easily differentiated from those of another class (Patel and Thakur, 2019). Particularly, the FWKNN method (Harshita and Singh, 2017) itself encounters the problem of measuring accurate class memberships for training instances concerning class imbalance (Sun and Chen, 2021). In this context, by addressing these problems, Patel and Thakur (2019) introduced the FADPTKNN method that utilizes a self-adaptive k strategy for imbalanced binary class problems. However, since this approach requires different k values for different classes, its classification performance seems to be heavily influenced by different neighborhood sizes. It implies that if at least one neighborhood is represented by a noisy and imprecise instance, it can cause poor classification decisions.

Furthermore, it is apparent that class imbalance is not the only reason responsible for degrading the performance of classifiers. Kotsiantis et al. (2006) provided evidence that classifier performance is not influenced solely by class imbalance but also by the level of "class overlapping" in the data. According to Kubat and Matwin (1997), imbalanced data can also suffer from "class-label noise" that needs to be taken into consideration before any learning algorithm is utilized. As noted by some studies (for example, see Gou et al., 2014, 2019; Pan et al., 2017), existing outliers among classes can severely harm classification performance, especially for KNN classification. This issue can also arise due to imbalanced data and can consequently make classifier learning more challenging (García et al., 2006). Figure 3.1 attempts to exhibit these issues using example cases from

oriented methods are the classifiers that do not explicitly require training with training data. Instance-based learning methods such as KNN are examples of this method (Zhang et al., 2017b).

a binary class problem, where the classes are represented by (orange) circles and (blue) squares. As seen from the figure, class-label noise is indicated by three square points [see Figure 3.1(a)], which are in the circle class region and thus relatively far from the class boundary. The presence of outliers in both classes is depicted in Figure 3.1(b), while the nature of class overlapping is exhibited in Figure 3.1(c).

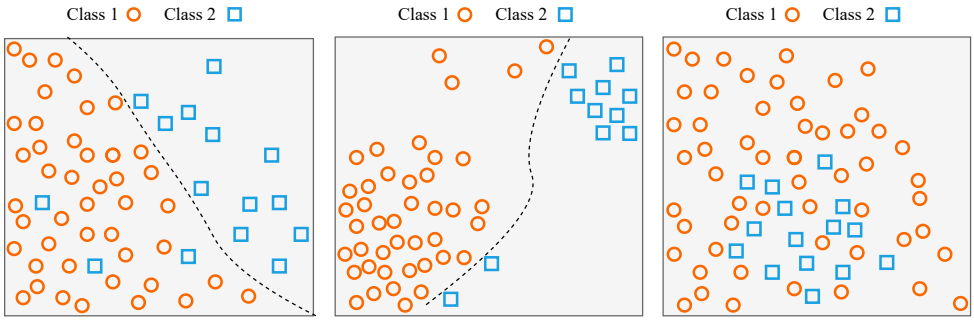


Figure 3.1: An illustration of imbalanced data with (a) class-label noise, (b) outliers, and (c) overlapping classes—the dashed line in sub-figures (a) and (b) indicates a class boundary.

Considering the above, it is clear that along with focusing on the class imbalance, it is necessary to pay attention to other issues (such as class-label noise, overlapping, and outliers) in the data distribution while performing model learning. However, this has been a great challenge since learning algorithms' performance and efficiency remain heavily dependent on the correct choice of model features and the data that often engage with many issues. Thus, there is a great demand for new ideas and practices that yield more effective and robust learning algorithms that can cope with the difficulties accompanied by the existing methods. In this context, this research seeks to develop solution techniques based on the FKNN algorithm (Keller et al., 1985) for the issues mentioned above while dealing with the class imbalance. To design the novel classifiers, the concept of the local mean—which was initially utilized by Mitani and Hamamoto (2006) and subsequently used in studies by Pan et al. (2017) and Gou et al. (2014, 2019)—is displayed. In Publications **I** and **III**, the FKNN algorithm was extended by deploying class representative local means in the learning part. The attempt in Publication **I** is discussed in detail in the next section.

3.2 Introduction

Here, a novel classification approach called "the multi-local power mean fuzzy k-nearest neighbor (MLPM-FKNN) method" is introduced. It is a supervised learning approach that theoretically works based on the previously described FKNN (Keller et al., 1985) classifier. The proposed method aims to advance the classification performance of the

FKNN classifier by using "multi-local" mean vectors[7] that can represent each class in the training data. To generalize these multi-local mean vectors further, the Power mean (Bullen, 2003) is used, and the resulting mean vectors are called "multi-local power mean vectors," which are deployed to measure the similarity of the query sample to each class. Because the Power mean is applied to calculate class representative mean vectors, it is necessary to determine the type of mean that is appropriate for a given data set by finding a proper value for the Power mean parameter. This adjustable mean-controlling parameter allows for the obtaining of an optimal mean value that can improve performance because a suitable mean type is data-dependent. As stressed in Section 2.1.1, it is also clear that by changing the Power mean parameter, several well-known means such as harmonic, geometric, and arithmetic means, can be selected.

The basic idea behind the use of local means is to disregard the class imbalance and associated inconsistencies in the training data without changing the class distribution and eliminating noisy instances and outliers. This allows prototypes of the minority class to have a significant contribution similar to those of the majority class in the neighborhood of the query sample. Moreover, class memberships are assigned to the query sample based on the distances between the query sample and class representative local mean vectors, unlike in the FKNN method. In this way, the local mean vectors are deployed instead of the individual instances in the neighborhood. Eventually, the classification of the query sample is accomplished by picking the class with the highest membership degree assigned by a representative multi-local mean vector.

3.3 Description

The MLPM-FKNN method consists of the following six steps, which are discussed in detail in this section. Before describing the algorithm as proposed in Publication **I**, it should be noted that the first steps until the search of k nearest neighbors are similar to what was discussed regarding the standard FKNN method. Thus, for simplicity of exposition, the second step of the new algorithm is described as a composition of steps two and three of the FKNN algorithm. The key steps of the proposed algorithm are the local mean computation and membership assignment steps.

Step I involves the settings of model parameters. The MLPM-FKNN requires initializing three parameters: the fuzzy strength parameter, $r \in (1, +\infty)$, the number of nearest neighbors, $k \in [1, N]$ (N is the number of instances in the training set), and the Power mean parameter, $p \in \mathbb{R}$. The values of p and k can vary along with the predefined ranges, and r is assigned a fixed[8] value. The key question here is what would be the most

[7]The concept of "multi-local" mean vectors was initially introduced by Pan et al. (2017) to the k-harmonic mean nearest neighbor (MLM-KHNN) classifier. However, the definition of "multi-local mean" in our study differs from what is used in Pan et al. (2017), in which it has been defined as a mean of the top nearest neighbors from 1 to k in each class. In the present work, a local mean vector is calculated by taking the average of nearest neighbors in a particular class by considering the whole set of k nearest neighbors.

[8]If needed, r can also be passed as an adjustable parameter, but for simplicity of computation, a fixed

suitable combination of parameters p and k values that can produce better accuracy in the classification.

Step II involves finding a set of k nearest neighbors denoted by $nn^k(Y)$ for a given query sample Y from the training set $\{X_j, c_j\}_{j=1}^{N}$, where $c_i \in \{\omega_1, \omega_2, .., \omega_S\}$. In this case, the Euclidean distances $d_{EUC}(Y, X_j)$ between Y and each instance $X_j \in \mathbb{R}^{\mathcal{D}}$ are calculated, and ordered training instances according to increasing distances are then used to search for the nearest neighbors.

Step III is the key step of the MLPM-FKNN algorithm, which encompasses calculating local mean vectors for each class among the k nearest neighbors. The idea here is that instances in $nn^k(Y)$ are categorized into each class first, and the resultant class-subsets $\{X_{ij}^{nn}\}_{j=1}^{n_i}$ for $i = 1, 2, .., S_k$ (where n_i is the number of instances that belong to the i^{th} class, and $S_k \in [1, S]$ is the number of classes available in $nn^k(Y)$) are then used to obtain the class representative local mean vectors. In this case, the Power mean operator (Bullen, 2003) is used to aggregate the neighboring instances over each class. The multi-local power mean vector $\mathcal{M}_i^p \in \mathbb{R}^{\mathcal{D}}$ from the i^{th} class can be defined for $p \neq 0$ as follows:

$$\mathcal{M}_i^p = \left(\frac{1}{n_i}\sum_{j=1}^{n_i}(X_{ij}^{nn})^p\right)^{1/p} \tag{3.1}$$

Here, it should be noted that when $p \to 0$, $\mathcal{M}_i^p$ approximates $\prod_{i=1}^{n_i}(X_{ij}^{nn})^{1/n_i}$ (the geometric mean). It is clear that altering parameter p pushes the local-mean prototype in the neighborhood closer to the query sample (this is clarified in Figure 2.1b).

To gain a clear perspective of local mean computation, Figure 3.2 displays an example case in which the local means for each class is calculated from the $k = 10$ neighborhood. It exhibits a classification problem with three classes: the (orange) circles (which appear to be the majority class), the (green) triangles (seemingly, the minority class), and the (blue) squares (the instances of the other class). The set of 10 nearest neighbors for a given query point Y (denoted by "+") consists of four instances (n_1, n_2, n_3, and n_4) from the circle class, three (n_5, n_6, and n_7) from the square class, and three (n_8, n_9, and n_{10}) from the triangular class. By aggregating them in each class using the Power mean, local mean vectors ($\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$) can be computed as depicted in the (right side) sub-figure. It is important to note that if the standard KNN method is applied in this example, the query point could be classified as a circle according to the majority voting rule, which seems to be an incorrect classification.

Further, it is also noteworthy that having class representative local means allow for mitigation of other problems caused by the distributions of data, including class noise, outlier effects, and overlapping class problems. For instance, as seen in Figure 3.2, n_4 and n_5 could be noisy examples of their corresponding classes. However, the local

value of 2 is used as per suggestions by Keller et al. (1985) and Derrac et al. (2015).

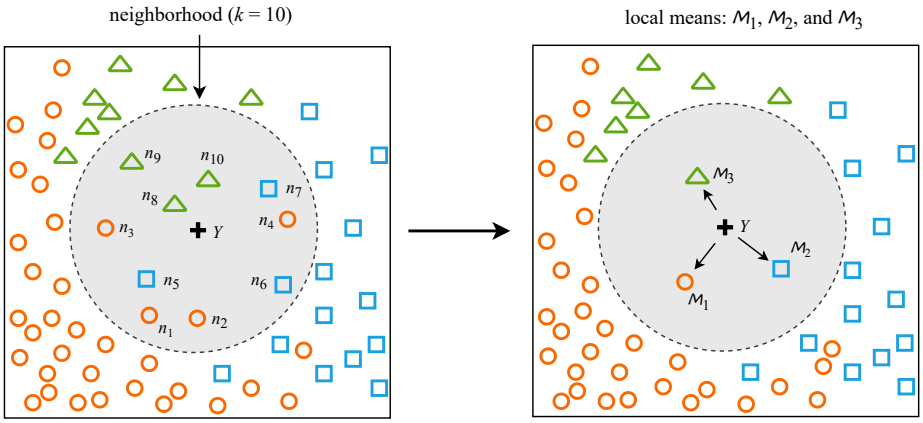


Figure 3.2: An illustrative example of local means computation for $k = 10$.

means $\mathcal{M}_1$ and $\mathcal{M}_2$ have treated them while disregarding their negative influences and representing the distributions of their local instances. In this way, multi-local means in the proposed approach effectively contribute to better learning.

Step IV computes the Euclidean distances $d_{EUC}(Y, \mathcal{M}_i^p)$ from Y to multi-local mean vectors $\mathcal{M}_i^p$ for $i = 1, 2, ..., S_k$ according to

$$d_{EUC}(Y, \mathcal{M}_i^p) = \sqrt{(Y - \mathcal{M}_i^p)^T (Y - \mathcal{M}_i^p)} \tag{3.2}$$

where T indicates the vector's transpose. This differs from the usual FKNN method in the sense that here distance is computed from multi-local mean vectors instead of the nearest neighbors, which is a unique property of the new method.

Step V is the assignment of membership (u_i) to the representing class i using the distances $d_{EUC}(Y, \mathcal{M}_j^p)$ for $j = 1, 2, ..., S_k$. This can be expressed as:

$$u_i(Y) = \frac{\sum_{j=1}^{S_k} u_{ij} (1/d_{EUC}(Y, \mathcal{M}_j^p)^{2/(r-1)})}{\sum_{j=1}^{S_k} (1/d_{EUC}(Y, \mathcal{M}_j^p)^{2/(r-1)})} \tag{3.3}$$

where, u_{ij} is the membership of the j^{th} multi-local mean vector in the i^{th} class, and it is measured using crisp membership as:

$$u_{ij} = \begin{cases} 1, & \textbf{if } \ c_i = c_j \\ 0, & \textbf{if } \ c_i \neq c_j \end{cases} \quad \text{for } j = 1, ..., S_k. \tag{3.4}$$

In the proposed method, membership degrees are assigned to the multi-local mean vectors instead of allocating memberships directly to the training instances in the neighborhood as per the FKNN rule.

Step VI classifies the query sample Y into class ω_c with the highest membership degree among all classes. It can be denoted as:

$$\omega_c = \arg \max_{\omega_i} u_i(Y) \tag{3.5}$$

Furthermore, the value selected for k is potentially important in generating local mean sub-samples and, ultimately, the classification accuracy. The performance of the proposed MLPM-FKNN classifier tends to be high when the higher values for k are chosen. If the class subset sizes rise with high k values, the multi-local mean vectors can be said to become more comprehensive representations or more realistic in terms of the classes they represent. This may lead to higher performances with the MLPM-FKNN classifier than the classical KNN and FKNN methods when the value of k increases. The best classification performance is obtained when the optimal values for k and p are found. A pseudo-code of the MLPM-FKNN method is presented below in Algorithm 1.

Algorithm 1: MLPM-FKNN Algorithm (adapted from Publication **I**)

Input: training data $\{X_j, c_j\}_{j=1}^{N}$, query sample Y, k, and p
Output: class label ω_c for Y

Begin
for $j \leftarrow 1$ **to** N **do**
  calculate $d_{EUC}(Y, X_j) \leftarrow \|Y - X_j\|$
  if $j < k$ **then**
    add X_j to $nn^k(Y)$
  else if X_j is closer to Y than any of neighbors in $nn^k(Y)$ **then**
    remove the farthest neighbor from the set $nn^k(Y)$ and add X_j
  end
end
for $i \leftarrow 1$ **to** S_k **do**
  find $\mathcal{M}_i^p \leftarrow (\frac{1}{n_i}\sum_{j=1}^{n_i}(X_{ij}^{nn})^p)^{1/p}$
  calculate $d_{EUC}(Y, \mathcal{M}_i^p) \leftarrow \|Y - \mathcal{M}_i^p\|$
end
for $i \leftarrow 1$ **to** S_k **do**
  compute class memberships for Y,

$$u_i(Y) \leftarrow \frac{\sum_{j=1}^{S_k} u_{ij}(1/d_{EUC}(Y, \mathcal{M}_j^p)^{2/(r-1)})}{\sum_{j=1}^{S_k}(1/d_{EUC}(Y, \mathcal{M}_j^p)^{2/(r-1)})}$$

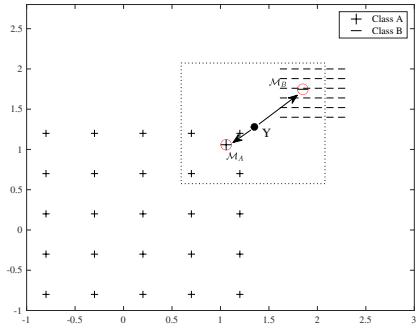
$$\text{where, } u_{ij} = \begin{cases} 1, & \textbf{if } c_i = c_j \\ 0, & \textbf{if } c_i \neq c_j \end{cases} \text{ for } j = 1, ..., S_k$$

end
return ω_c such that

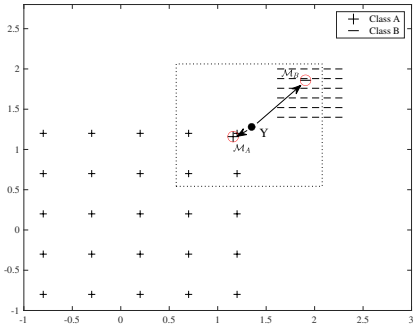
$$\omega_c = \arg\max_{\omega_i} u_i(Y)$$

Figure 3.3 demonstrates a simple example that depicts how the MLPM-FKNN algorithm finds reasonable class prototype vectors by varying the parameters k and p. In this example, the data set includes two classes [class A ($+$) and class B ($-$)], and the idea is to determine which class the query point (Y) belongs to. When the geometric position of the query sample is considered, class A might be the correct choice for Y. Figure 3.3a and 3.3b show the geometric positions of the computed multi-local mean vectors ($\mathcal{M}_A$ and $\mathcal{M}_B$) when the Power mean parameter $p = 5$ and $p = 20$, respectively. In this case, the value of neighborhood (k) is set to 28. As shown in Figure 3.3a and 3.3b, it is clear that when the p value is increased, the local mean of class A tends to be closer to Y, which might make degrees of membership more accurate and, finally, the correct class decision. The new classifier searches for more suitable class prototypes than the arithmetic mean-based local mean vectors, which can be observed when Figure 3.3d (that shows an example case $k = 9$ for the LM-KNN method) is compared with others.

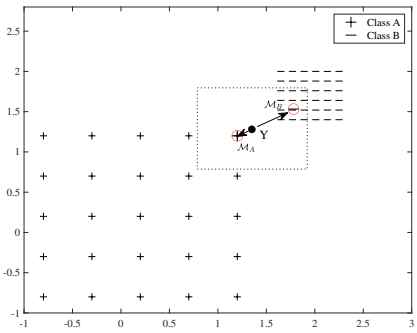
As Figure 3.3d shows, the LM-KNN algorithm results class B prototype being closer to Y than the class A prototype, leading to a choice that might not be the correct case. However, when the MLPM-FKNN classifier (with a similar k value) is applied (see Figure 3.3c), it shows this method can produce class prototypes that lead to an accurate classification. Moreover, the MLPM-FKNN classifier is also based on the distances between mean vectors and the query sample—therefore, the geometric positions of those mean vectors are essential. For this reason, the new approach is special, particularly compared to other local mean-based KNN methods, such as the LM-KNN.



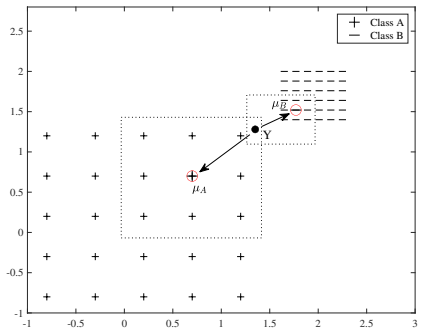
(a) when $p = 5$ (and $k = 28$)



(b) when $p = 20$ (and $k = 28$)



(c) when $p = 5$ (and $k = 10$)



(d) LM-KNN classifier with $k = 9$

Figure 3.3: Example cases of class prototypes around the query sample (Y) for different values of the parameters k and p.

The novelty of the proposed method is three-fold in nature:

- Introduction of a local mean computation in the learning part of the FKNN algorithm, which extracts the information of each class (represented by the instances in the k neighborhood) into a single vector (see Figure 3.2).
- Application of the Power mean in the local mean computation, which expands the

region for the mean, thus providing greater flexibility to obtain more reasonable class representative local mean vectors.

- Utilization of the inverse of the distance between the query sample and each multi-local power mean vector to calculate the class memberships for the query sample.

3.4 Application to Machine Learning Repositories Data

3.4.1 Data and Testing Methodology

To assess the performance of the proposed classifier in Publication **I**, data sets that are publicly available at the KEEL (knowledge extraction based on evolutionary learning) (Alcala-Fdez et al., 2011) and UCI (Dheeru and Taniskidou, 2017) machine learning repositories were utilized. Four data sets were chosen, including two binary class problems (Vehicle and Ionosphere) and two multi-class problems (Car and Thyroid). As Table 3.1 shows, each data set has different properties in terms of the number of instances, features, and classes. The corresponding field of each data set is described by "Domain" in the table.

Table 3.1: Properties of the data sets used—all features are continuous.

Data set	Database	Instances	Features	Classes	Domain
Car	KEEL	1728	7	4	Engineering
Vehicle	KEEL	846	18	2	Engineering
Ionosphere	UCI	351	34	2	Physics
Thyroid	UCI	215	6	3	Medicine

The study design to solve the selected classification problems included the following aspects: data normalization, test/validation/train splits, cross-validation, and a statistical significance test. All of these steps were performed as described in Section 2.6. It is necessary for our method that all features data fed in the analysis be in the unit interval when negative values appear. This is because the Power mean function with an odd value of p will produce the local mean vectors with complex numbers when some features include negative values. Furthermore, the performance of the proposed MLPM-FKNN classifier was benchmarked against the standard KNN (Cover and Hart, 1967) and FKNN (Keller et al., 1985) algorithms. Next, the number of nearest neighbors, k, was chosen from the set $\{1, 2, ..., 25\}$. Here, we assumed when the value of k increases, the classification performance of the MLPM-FKNN classifier also increases because more instances make local mean vectors better representative of the corresponding classes. This assumption was supported by the evidence in the study by Pan et al. (2017) that showed that the multi-local mean-based k-harmonic nearest neighbor (MLM-KHNN) achieved improved accuracy in the classification with higher k values. The Power mean parameter, p, was selected from the range $\{-8, -7, ..., 7, 8\}$. Here, it should be noted that the MLPM-FKNN algorithm

itself does not adjust its parameters—in which case, an external technique such as the grid search needs to be performed. To achieve statistically reliable results, a 30-fold hold-out cross-validation was applied (see Figure 2.4), whereby training and validation sets were repeatedly sampled and deployed to tune model parameters using the grid search and assess the validation performance. Next, the average validation performance of the proposed model (and the benchmarks) were evaluated and the optimal parameter values were obtained. Eventually, the classifier was tested by fitting it with the validation data to assess its generalization and predictive capability with the test data.

3.4.2 Results

Classification performance on the validation sets. First, the performance of this study's new method was compared to each of the benchmarks in the validation data. Table 3.2 summarizes the mean accuracies and related results achieved with each of classifiers on all four data sets.

Table 3.2: The average performance of the proposed approach and the benchmarks in the classification of validation data. The terms "Op. Parameters" and "CI" refer to optimal parameter values and confidence intervals, respectively [modified from Publication **I**].

Data	Measure	MLPM-FKNN	FKNN	KNN
Car	Mean accuracy	**0.9230**	0.8823	0.8740
	Variance	1.13e-04	6.19e-05	6.09e-05
	CI	[0.9191, 0.9270]	[0.8794, 0.8853]	[0.8711, 0.8769]
	Opt. Parameters	$k = 25, p = 0$	$k = 4$	$k = 3$
Vehicle	Mean accuracy	**0.9412**	0.9291	0.9274
	Variance	1.46e-04	1.92e-04	1.98e-04
	CI	[0.9367, 0.9457]	[0.9239, 0.9343]	[0.9222, 0.9327]
	Opt. Parameters	$k = 11, p = 1$	$k = 4$	$k = 1$
Ionosphere	Mean accuracy	**0.8881**	0.8443	0.8431
	Variance	5.57e-04	4.24e-04	4.66e-04
	CI	[0.8793, 0.8969]	[0.8366, 0.8520]	[0.8350, 0.8512]
	Opt. Parameters	$k = 16, p = 1$	$k = 4$	$k = 3$
Thyroid	Mean accuracy	**0.9229**	0.9167	0.9101
	Variance	5.92e-04	6.16e-04	6.05e-04
	CI	[0.9138, 0.9320]	[0.9074, 0.9259]	[0.9009, 0.9193]
	Opt. Parameters	$k = 7, p = 3$	$k = 3$	$k = 1$

The table results demonstrate that the novel MLPM-FKNN classifier achieved the highest mean accuracy (marked in bold), outperforming the KNN and FKNN methods in all cases. Moreover, the confidence interval results remained narrow, and the variances were relatively low—implying reasonable repeatability of the proposed approach.

Meanwhile, regarding parameter p, it is clear that arithmetic mean ($p = 1$) yielded the best performance with the proposed method for Vehicle and Ionosphere data sets, while geometric ($p = 0$) and cubic ($p = 3$) means reflected the optimum for Car and Thyroid data sets, respectively. Moreover, it is apparent from Table 3.2 that the MLPM-FKNN classifier performs considerably better when using a high value of parameter k. This is remarkable because low k values appear to be working better for the baseline methods, and specifically, $k = 1$ has been shown as optimum for the KNN with two data sets. This finding is consistent with what was obtained earlier by Derrac et al. (2015).

Moreover, Figure 3.4 makes it easier to understand the influence of different settings of p and k on the performance of the MLPM-FKNN classifier with the Vehicle data during training and validation. Publication **I** provides a detailed analysis of the proposed approach and its performance on the selected data.

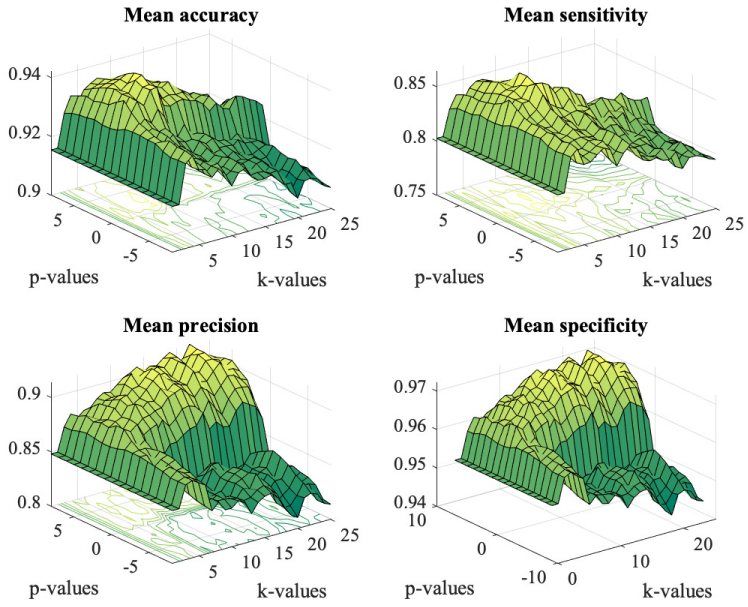


Figure 3.4: Classification performance of the MLPM-FKNN classifier for different combinations of parameters $p \in [-8, 8]$ and $k \in [1, 25]$ with the Vehicle data. [reproduced from Publication **I**]

Classification performance on the test sets. Regarding the results on the test sets (Table 3.3), essentially the same pattern was found in terms of the mean accuracy of the proposed method and benchmarks for all four data sets. As the table shows, the proposed MLPM-FKNN model achieved the highest accuracy and improved sensitivity and specificity results in all cases as compared to the KNN and FKNN classifiers. Here, the utilization of the local mean with the proposed approach appears to be reducing variances.

However, the proposed MLPM-FKNN approach takes a slightly longer time (average computation time in testing) for computation than the classical KNN and FKNN methods.

Table 3.3: Classification results of the proposed approach and the benchmarks with the test data (CI = confidence interval, Comp. time = computation time in seconds) [adapted from Publication **I**].

Data set	Measure	MLPM-FKNN	FKNN	KNN
Car	Mean accuracy	**0.9246**	0.8742	0.8632
	Mean sensitivity	0.8105	0.6704	0.6085
	Mean specificity	0.9667	0.9119	0.9014
	CI	[0.9204, 0.9288]	[0.8688, 0.8796]	[0.8577, 0.8687]
	Variance	1.26e-04	2.08e-04	2.15e-04
	Comp. time	0.0466	0.0174	0.0159
Vehicle	Mean accuracy	**0.9294**	0.9051	0.9006
	Mean sensitivity	0.8568	0.7907	0.7910
	Mean specificity	0.9521	0.9432	0.9355
	CI	[0.9241, 0.9347]	[0.8966, 0.9136]	[0.8932, 0.9080]
	Variance	2.03e-04	5.18e-04	3.95e-04
	Comp. time	0.0194	0.0059	0.0050
Ionosphere	Mean accuracy	**0.8381**	0.7957	0.7952
	Mean sensitivity	0.8212	0.7686	0.7694
	Mean specificity	0.8925	0.9215	0.9113
	CI	[0.8281, 0.8481]	[0.7875, 0.8039]	[0.7869, 0.8036]
	Variance	7.237e-04	4.81e-04	4.97e-04
	Comp. time	0.0100	0.0076	0.0026
Thyroid	Mean accuracy	**0.9434**	0.9279	0.9233
	Mean sensitivity	0.9256	0.8520	0.8449
	Mean specificity	0.9584	0.9254	0.9214
	CI	[0.9316, 0.9552]	[0.9171, 0.9387]	[0.9138, 0.9327]
	Variance	9.95e-04	8.34e-04	6.40e-04
	Comp. time	0.0037	0.0014	0.0009

Subsequently, a paired *t*-test (Section 2.6.5) was used to evaluate whether there is a significant difference between the mean accuracy of the proposed MLPM-FKNN and that of each benchmark method at a significance level of 0.05. Table 3.4 presents the results obtained from the t-test with the test data. The table results provide strong support to the claim that the accuracy of the MLPM-FKNN classifier is statistically significantly higher than the accuracies of traditional KNN methods (t-test $p < 0.05$ in all cases except for the Thyroid data with FKNN, but it also has a comparative significance value of p).

To further demonstrate the effectiveness of the proposed MLPM-FKNN classifier, we

Table 3.4: Results of the t-test at 0.05 significance on the performance of the MLPM-FKNN classifier vs. benchmarks for the test data [adapted from Publication **I**].

Dataset	Paired with	MLPM-FKNN (p-value)	Test-statistic
Car	FKNN	$1.64e^{-18}$	significant
	KNN	$2.88e^{-25}$	significant
Vehicle	FKNN	$6.73e^{-06}$	significant
	KNN	$2.41e^{-08}$	significant
Ionosphere	FKNN	$4.95e^{-08}$	significant
	KNN	$1.06e^{-12}$	significant
Thyroid	FKNN	0.051	not-significant
	KNN	0.008	significant

selected four additional data sets: Balance ($n = 625$), Landset ($n = 2000$), Monk-2 ($n = 432$), and Ringnorm ($n = 7400$) from UCI (Dheeru and Taniskidou, 2017) and KEEL(Alcala-Fdez et al., 2011) repositories and tested the classifiers. The classification results for each classifier over the test data sets are presented in Table 3.5. According to the results in the table, it is clear that the proposed MLPM-FKNN method has produced statistically significantly higher accuracies than the KNN and FKNN methods over all data sets. The new approach also has reasonable mean sensitivity and specificity values and small variances compared to the classical methods. The P value from t-test indicates the statistical significance of the higher performance of the MLPM-FKNN method in comparison to the classical methods. All in all, these results further confirm the potential of the proposed MLPM-FKNN method for classification problems.

Table 3.5: Test results of the proposed approach and the benchmarks over additional high-dimensional data.

Data set	Measure	MLPM-FKNN	FKNN	KNN
Balance	Mean accuracy	**0.8651**	0.8611	0.8500
	Mean sensitivity	0.7230	0.6236	0.6164
	Mean specificity	0.9293	0.9146	0.9074
	Variance	5.9947e-04	2.0743e-04	1.9266e-04
	P value (α = 0.05)		0.4473	0.0048
Satimage	Mean accuracy	**0.9089**	0.9013	0.8979
	Mean sensitivity	0.8828	0.8764	0.8720
	Mean specificity	0.9813	0.9799	0.9792
	Variance	7.0495e-06	1.5857e-05	2.9535e-05
	P value (α = 0.05)		1.7484e-08	6.6191e-10
Monk-2	Mean accuracy	**0.9812**	0.9670	0.9483
	Mean sensitivity	0.9810	0.9654	0.9453
	Mean specificity	0.9810	0.9654	0.9453
	Variance	2.5953e-04	3.0736e-04	4.1685e-04
	P value (α = 0.05)		0.0019	3.7048e-09
Ringnorm	Mean accuracy	**0.8176**	0.7245	0.7862
	Mean sensitivity	0.9994	0.9925	0.9874
	Mean specificity	0.7348	0.6476	0.7046
	Variance	3.5351e-05	8.6012e-05	4.7906e-05
	P value (α = 0.05)		1.0365e-20	8.2772e-13

3.5 Application to the S&P 500 Intraday Returns Forecast

3.5.1 Introduction and Objectives

In the previous section, it was demonstrated how the new MLPM-FKNN classifier could perform better than classical KNN and FKNN methods in terms of several real data sets from machine learning repositories. In Publication **II**, the capability of the proposed model for a challenging real-world problem was further explored—predicting the S&P 500 stock index return.

Stock market prediction, which often refers to forecasting the future price or return of a stock index (or stock), is typically a more challenging task due to non-stationary behavior and numerous uncertainties in the data. A precise prediction of the future pattern of a particular stock market variable allows investors to make effective decisions and investments (Barak et al., 2017). Choosing representative and relevant features is essential for accurate stock market prediction (Tsai and Hsiao, 2011; Zhang et al., 2014). Accordingly, many studies (for example, see Tsai and Hsiao, 2011; Zhang et al., 2014;

Barak et al., 2017; Lohrmann and Luukka, 2019) in stock market prediction research have focused on determining features that could produce improved accuracy while maintaining the model efficiency at a sufficient level. However, most of them have restricted their analysis to a lower number of features selected from a specific domain. In Publication **II**, the aim was to investigate potentially relevant features from a large input feature set (containing technical indicators (TIs), exchange rates, commodities, macro-economy, and stock indices as well as stocks features) for predicting the open-to-close (intraday) returns of the S&P 500 index. To deal with the curse of dimensionality while fitting classification models to high-dimensional data, a hybrid feature selection approach was developed in Publication **II** by integrating filter and wrapper methods to identify a relevant feature subset for stock market prediction. The idea was to first apply a simple filter method to discard highly correlated features before utilizing the wrapper method that is known to be more computationally expensive.

In the literature, many different prediction models using machine learning have been developed and applied for stock market forecasting (Cao et al., 2019; Kumbure et al., 2022). Among them, the KNN-based approaches have been demonstrated to perform well in several studies (for example, see Zhang et al., 2017a; Cao et al., 2019). The KNN-based classification methods generally consider all features equally when measuring the similarity between data points, which might not be optimal for all situations. In this study, a novel MLPM-FKNN classifier (and the classical KNN and FKNN methods) was integrated with a hybrid feature selection approach to mitigate the possibly harmful influence of irrelevant features on this form of classification.

3.5.2 Data

For this analysis, historical time series data was obtained free of charge from Federal Reserve Economic Data[9] and Yahoo Finance[10] websites, which covered the trading period from 10/10/2007 to 10/10/2020. The initial data set included daily prices of the S&P 500 stock index and a set of relevant expanding variables that were considered to be suitable for predicting this index return. These variables consisted of TIs, exchange rates, commodity prices, and other stocks and stock indices.

Input features. In our review research (Kumbure et al., 2022), we collected and examined the most commonly used input features for predicting the stock market from earlier studies. By taking advantage of it, 302 features that were at the top of the list, found by us in our study (Kumbure et al., 2022) in terms of the frequency of usage in stock market forecasting studies were chosen for the present study. An overview of the selected features, along with their category and the corresponding number of features ("No. of features") used from each category is provided in Table 3.6. The meanings of commonly used abbreviated forms of the features in Table 3.6 can be found in Kumbure

[9]FRED is a database of economic time series data, which is available at https://fred.stlouisfed.org.

[10]Yahoo Finance provides a vast range of financial data, news, and information, which are freely available at https://finance.yahoo.com.

et al. (2022). As shown in the table, TIs are further categorized into "Basic TIs" that represent Open, Low, High, and Close prices and Volume of the S&P 500 index, and "Other TIs" that represent all other types of TIs that had to be calculated using Closing prices of the S&P 500 and other variables. Next different variants of TIs were created by varying the time window (n in days) and some other parameters (for example, n_1 and n_2 for EMA, $slow, fast, sign$ for MACD). For other time series, all their basic TIs were considered.

Table 3.6: Information on the initial list of features [adapted from Publication **II**].

Category	**Feature names**	**No. of features**
Basic TIs	Open, High, Low, Close, Volume (n)	14
Other TIs	RSI(n), EMA(n), MACD($slow$,$fast$,$sign$), Bias (n), Disparity(n), SMA(n), OBV, Williams %R(n), Return(n), CCI(n), MFI(n), Stochastic %K(n), Momentum(n), Stochastic %D(n), Bollinger bands (H/M/ L), Chaikin Volatility, TMA(n), Price oscillator (n_1, n_2), TRIX, Typical price	140
Macro-economy	Treasury Bills, Treasury Constant Maturity Rate, Term Spread, Treasury Yields, AAA Corporate Bond, BAA Corporate, Bond, Default Spread	27
Commodities	Gold, Silver, Crude Oil	18
Exchange rates	USD/ NTD, USD/ GBP, USD/JPY, USD/ CAD, USD/ CNY	25
stock indices / Other stocks	Microsoft, Amazon, Apple Inc., General Electric, Hang Seng, DJIA, SSE, CAC40, JPM, NASDAQ, Wells Fargo, Exxon Mobil, JNJ	78

Target variable. The intraday (open-to-close) return of the S&P 500 index was selected as the target (i.e., class) variable and established as a multi-class variable across four classes ("1," "2," "3," and "4") by considering the daily magnitude of the return according to the study by Lohrmann and Luukka (2019). In this variable, class label "1" indicates the intraday returns that are larger than 0.5% (i.e., strong positive), "2" between 0.0% and 0.5% (i.e., slightly positive), "3" between -0.5% and 0.0% (i.e., slightly negative), and "4" smaller than -0.5% (i.e., strong negative). In this way, the problem was set up as a classification task to forecast the return class instead of predicting actual return values.

3.5.3 Hybrid Feature Selection

Feature selection to choose potential variables to forecast the intraday return of the S&P 500 included two parts: (1) a correlation coefficient-based analysis (filter method), which was performed with all features to identify variables that were highly linearly dependent

on other variables and then removing such variables; (2) DEFS (wrapper method) with KNN-based classifiers, which was applied to discard the remaining irrelevant features.

The Pearson correlation coefficient is a simple and popular filter approach in the context of feature selection, which is typically employed to estimate a linear dependence between an input feature and the response variable (Chandrashekar and Sahin, 2014). In this case, a high (absolute) correlation typically signifies the relevance of the input feature for the response (class variable). On the contrary, a high (absolute) correlation between two different input features may imply that using both features instead of a single one may not provide new information, implying that one of them may potentially be redundant. In the present analysis, the Pearson correlation coefficients between feature pairs (f_i, f_j) for $i, j \in \{1, 2, .., 302\}$ were first calculated. A threshold of 0.95 was used for a very high absolute correlation between two features, and one of the two features had to be eliminated. Next, the correlation coefficients between each feature and class variable (y) were considered and eventually, feature removal was performed in a way that if $corr(f_i, f_j) \geq 95$ and $corr(f_i, y) < corr(f_j, y)$ then f_i was removed, otherwise f_j was removed.

Meanwhile, differential evolution (DE), introduced by Price et al. (2005), is a population-based heuristic optimization technique (Yang et al., 2019). DE has been used in numerous applications due to its easy implementation, efficiency, fast convergence, and robustness (Khushaba et al., 2011; Bisoi et al., 2019). It iteratively optimizes a given problem across an evolutionary process that consists of four main phases: initialization, mutation, crossover, and evaluation of objective function (Yang et al., 2019). By introducing several advancements to its search process, Khushaba et al. (2011) extended the DE to be applied as a wrapper method for feature selection. This enhanced version is referred to as differential evolution feature selection (DEFS), which was implemented and applied with KNN-based classifiers in Publication **II**. A detailed process of the DEFS method can be found in Khushaba et al. (2011).

3.5.4 Testing Methodology and Parameter Settings

The empirical process conducted in this study included four main phases subject to data preparation, relevant feature selection, and the forecasting S&P 500 intraday stock return. The first step was the data pre-processing. Initially, the missing values that arose when the S&P 500 data variable was merged with other variables were replaced using linear interpolation. Thereafter, underlying distributions of input features were scaled into the unit interval. TIs are typically used to infer trading signals for when a market or stock is overbought or oversold, and corresponding buying and selling decisions might be profitable. Thus, continuous data of TIs were transformed into discrete data (trading signals) to represent accurate trading signals instead of the numeric values for the TIs that they could be deduced from. Subsequently, to convert the commodity channel index (CCI) and relative strength index (RSI) values into trading signals, the approaches introduced by Patel et al. (2015) were used.

In the feature selection step, the data set was first manually divided into two sets: one for training (10/01/2007–18/01/2018) and the other for testing (19/01/2018–10/10/2020). The training set was further repeatedly split into 80% for training and 20% for validation to tune the model parameters and perform feature selection. As for the DEFS model parameters, the crossover rate was set to 0.5, and both population size and number of iterations to 50. Here, $\{5, 10, 15, ..., 50\}$ was also specified as the desired number of features to be chosen (i.e., the number of features in the resulting feature subset). The rest of the parameters involved in the DEFS were provided in line with the study by Khushaba et al. (2011). Here, it should be noted that as a wrapper method, DEFS typically requires considerably high computation time since it uses an iterative process that includes a classifier for feature selection. Therefore, considering the computational simplicity, the parameters of the nearest neighbor classifiers in the DEFS were purposely fixed in this study, keeping the number of nearest neighbors (k), fuzzy strength parameter (r), and the Power mean parameter (p) constant at 20, 2, and 1.5, respectively.

The next phase of the analysis aimed to demonstrate whether the features selected in the previous step were relevant and thus helpful in predicting the intraday S&P 500 return. Here, the same classifiers and training and test sets were used, but they were included only the selected features. In this experimental setup, the parameter search, cross-validation, and model evaluation with the test set were performed as described in Section 2.6. During training and validation, the parameters of each classifier were optimized using 30 runs of holdout cross-validation. The optimal value for k was explored from the set $\{1, 2, ..., 30\}$ for each KNN method. The Power mean parameter p was selected and optimized from the range $\{0, 0.5, 1, ..., 5\}$ for the MLPM-FKNN method. The best values for parameters were chosen, considering the maximum validation accuracy. Finally, the classifiers with optimal parameters were tested on the test data set. Also, it is worth mentioning that the learning part of the original MLPM-FKNN method presented by Algorithm 1 was slightly updated before using it for this study. Initially, the training data was categorized into classes, and then the sets of k nearest neighbors of the query sample were found from each class. Next, the multi-local power mean vectors are calculated for each set of k nearest neighbors in each class. The rest of the steps were the same as presented in Algorithm 1. The methodology used here for model construction and prediction is discussed in detail in Publication **II**. A summary of the workflow process for the hybrid feature selection and KNN-based prediction is shown in Figure 3.5

3.5.5 Results and Discussion

Results in the feature selection. The findings from the correlation coefficient-based filter method suggested the removal of some features because of linear dependency effects. Notably, this analysis reduced the dimensionality of the data set by eliminating potentially redundant features, retaining 207 out of 302 initial features for the subsequent DEFS with classifier learning. As the second part of the hybrid feature selection approach, the wrapper DEFS was utilized along with each KNN-based classifier. Figure 3.6 shows the frequency of the 50 best features corresponding to the DEFS with each classifier within

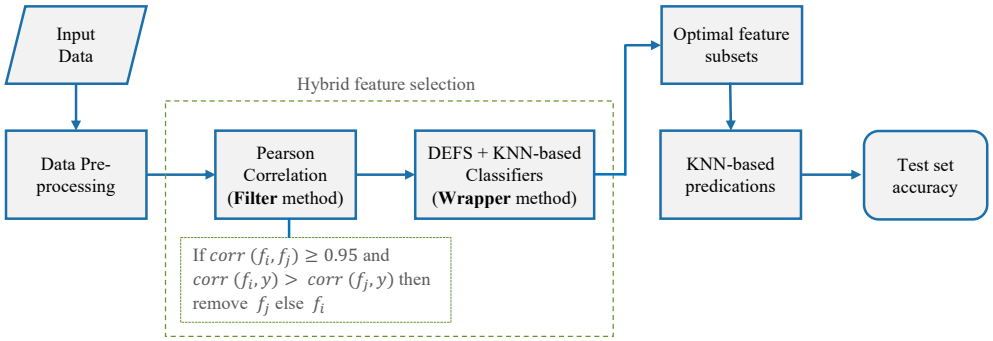


Figure 3.5: The process of hybrid feature selection and intraday S&P return prediction using KNN-based classifiers.

the 10-fold holdout validation.

Here, it is apparent that the majority of the selected features in the top 50 features for each classifier are TIs. The five-day moving average [SMA(5)] and the low price of Silver seem to be the most essential features through all three models, continuously ranking among the top 10 higher rated features. Other features that appeared at least twice in the top 10 contain TIs that were formed based on the last 5–15 days, such as SMA (15), Williams R(10), RSI(6), Disparity(10), and Chaikin Volatility(10). Finally, it is worth mentioning that no exchange rate or macro-economic variables were retained in any of the top 50 ranked features.

Figure 3.7 shows the mean classification error rates (%) of each classifier in the DEFS for each feature-subset size. It can be seen from the figure that when the size of the optimal feature subset with DEFS grows, the MLPM-FKNN classifier performs better than the classical KNN methods. This is to be expected since the instances with more features make local mean vectors explicitly representative and effective. Regarding the classification performance, it is clear that mean errors appear to be considerably high in this case as compared to other classification problems, with the accuracies for each classifier only being in the mid-thirties. However, it should be noted that this problem was approached as a four-class problem, and lower classification accuracies do not necessarily mean that a trading strategy based on these findings would not be able to yield excess returns (Lohrmann and Luukka, 2019; Teixeira and de Oliveira, 2010). When the four-class [strong negative (4), slightly negative (3), slightly positive (2), and strong positive (1)] prediction is aggregated to a binary class level by arranging predictions of class 3 and class 4 as “negative” and class 1 and class 2 as “positive,” the results tend to be informative and can be easily comparable with other binary tasks.

Results in the prediction. The aggregated test accuracies for the positive and negative classes for each classifier per feature subset size are provided in Table 3.7.
According to the results in Table 3.7, it can bee seen that the positive class has offered

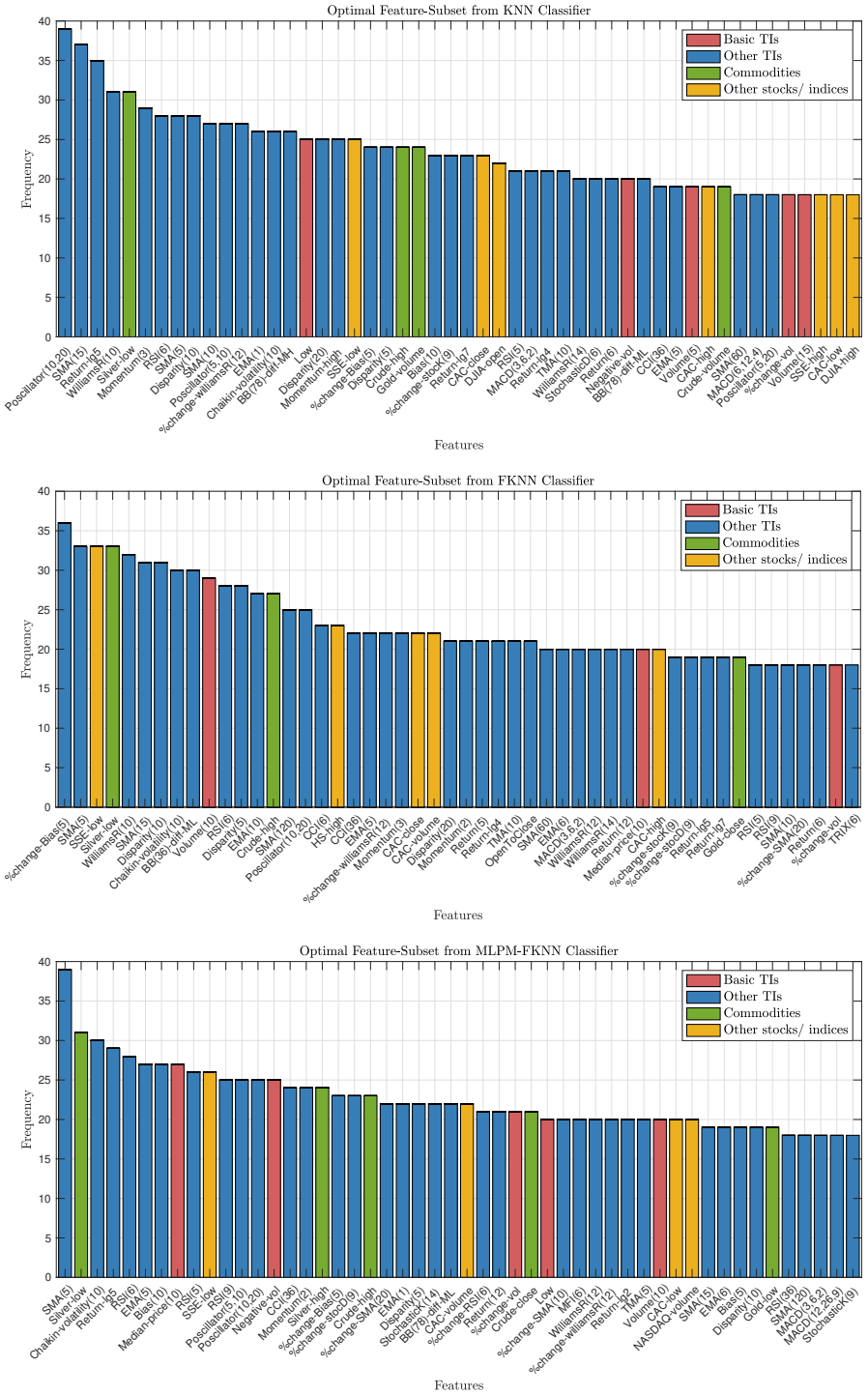


Figure 3.6: The frequency results of the top 50 features ranked (descending order) in the DEFS with each classifier [adapted from Publication **II**].

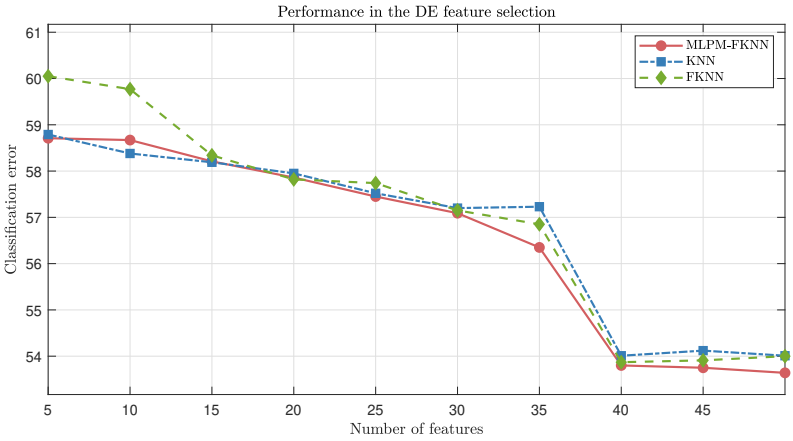


Figure 3.7: Average classification error (%) within a 10-fold holdout cross-validation for each feature subset size [adapted from Publication **II**].

more accurate predictions (about 70%) than the negative class (around 20–30%). This pattern appears to be consistent for all classification models when the feature subset size increases. This finding indicates that the prediction of the negative class is more challenging and therefore less accurate for the used KNN models. It is also noteworthy that the forecast of "positive" movement of the S&P 500 with higher accuracy than in the "negative" direction is consistent with the results reported in the study by Lohrmann and Luukka (2019). Moreover, test set accuracy with all features (see the last row in Table 3.7) appears to be the highest for all classifiers in the positive class but the lowest in the negative class. This confirms the efficacy of employing relevant features with DEFS because it enhances or, at a minimum, does not deteriorate the accuracy of classification of the negative intraday return of the S&P 500 even when a significant number of input features have been eliminated.

In summary, the generalizability of the study's proposed method across the feature selection has been demonstrated along with the possibility of performing it for the classification and prediction of a stock market index. The novelty of this work is the use of the MLPM-FKNN classifier with a (filter + wrapper) feature selection to identify potentially relevant features for forecasting the intraday returns of the S&P 500 index. Additionally, prediction performance of the applied classifiers were further investigated by using optimal feature subsets according to each classifier.

3.6 Conclusion and Limitations

A novel MLPM-FKNN classifier was introduced as an enhanced variant of the FKNN (Keller et al., 1985) method. The objective of the MLPM-FKNN classifier was to improve the performance of the FKNN in terms of imbalanced data sets. In this new variant, the Power mean was used to measure multi-local mean vectors that serve as

Table 3.7: Classification performance (mean accuracy % + standard deviation) in the positive and negative classes with the test set for all three classifiers and each feature subset [modified from Publication **II**].

	MLPM-FKNN		KNN		FKNN	
Subset	Positive	Negative	Positive	Negative	Positive	Negative
5	70.55 ± 4.99	30.14 ± 4.4	79.18 ± 3.89	21.71 ± 5.3	67.87 ±2.78	35.49 ±3.02
10	68.71 ±4.9	32.42 ±3.76	77.53 ±3.96	21.45 ±4.56	75.97 ±3.57	29.48 ±3.55
15	73.35 ±4.41	28.08 ±4.04	80.23 ±3.73	24.56 ±3.4	76.64 ±4.04	25.77 ±4.12
20	73.78 ±3.24	26.48 ±3.02	73.78 ±2.96	27.77 ±3.12	77.59 ±4.01	26.38 ±4.49
25	78.00 ±3.85	20.16 ±3.3	78.22 ±3.15	25.98± 3.45	74.85 ±4.2	28.48 ±4.2
30	75.31 ±3.86	21.05 ±4.34	73.71 ±3.45	27.77± 3.55	71.95 ±3.37	30.73 ±3.47
35	74.38 ±3.31	23.12 ±3.51	74.73±3.19	27.80 ±3.82	77.14 ±3.2	26.90 ±4.05
40	73.00 ±4.35	25.02 ±4.3	77.01 ±4.35	24.86± 4.43	75.97 ±3.49	28.66 ±3.64
45	70.60 ±3.68	26.60 ±4.07	77.26 ±4.48	24.56 ±4.75	73.83 ±4.27	28.15 ±4.2
50	72.08 ±3.82	24.81 ±3.72	76.91 ±4.25	26.43± 5.25	73.06 ±3.88	27.02 ±3.68
302	82.24 ±3.92	16.64 ±4.15	90.18 ±3.55	13.17 ±4.28	88.38 ±4.08	14.63 ±4.16

representatives for the existing classes around the query sample. Having the parametric Power mean allows the new classifier to be adapted to the context (of different situations) across a test to find an optimal parameter value that enhances classification accuracy. Here, the performance of the MLPM-FKNN model on four data sets selected from several different domains was evaluated, and empirical evidence that the new method can achieve significantly higher performance than the standard KNN and FKNN methods was presented. It was also highlighted that the proposed approach obtained the best accuracy with high k values; in contrast, the benchmark methods always produced better performances with relatively small k. Furthermore, a stock market prediction model for intraday S&P 500 index return was implemented using the MLPM-FKNN across the DEFS. Thus, this work brings a feature selection and prediction (formed by the classification) to a nexus in which the feature selection can produce a significant impact with the aid of MLPM-FKNN classification.

However, it should be emphasized that this approach comes with several limitations. It is clear that the newly defined classifier includes numerous calculations caused by the increase in parameters, thus testing more possible parameter combinations to obtain optimal values. Consequently, the computation time required for fitting the new method is longer than that of the baseline methods. However, it should be noted that using this method does not take much time, but fitting the model is more time-consuming. Moreover, the performance comparison was limited only to the standard KNN and FKNN methods. In response to this limitation, the performance of the proposed approach should be evaluated in comparison to some generalized versions of KNN [e.g., LM-KNN (Mitani and Hamamoto, 2006), IV-KNN (Derrac et al., 2015)] or other effective machine learning techniques [e.g., SVM (Cortes and Vapnik, 1995)]. Additionally, there is a

risk of poor performance on small samples of data because the generated multi-local means for each class may not be well-representative due to the lack of training samples. Moreover, particularly with the stock prediction study, it was observed that even though the MLPM-FKNN achieved a somewhat higher performance than the benchmarks in the feature selection part, the findings of the S&P 500 intraday return prediction showed comparatively similar performances for all classifiers. However, this does not necessarily mean that there is no difference in the performance between the proposed method and benchmarks; to show this difference comprehensively, a performance analysis with trading strategies derived from the results of each classifier is required.

Despite the limitations, it is believed that the results demonstrate the potential of the MLPM-FKNN classifier for binary- and multi-class classification tasks. Particularly, more challenging and imbalanced classification problems in machine learning can be modeled to extend the applicability of the used classifier. This work also provides a better understanding of the principles of the Power mean in terms of its applicability for fusing data in machine learning applications. In fact, possible future work can also focus on testing the impact of integrating the Power mean with other KNN variants, such as MLM-KHNN (Pan et al., 2017), IV-KNN (Derrac et al., 2015), and PNN (Gou et al., 2014).

4 Proposed Fuzzy *K*-nearest Neighbor Classifier Based on the Bonferroni Mean

This chapter presents a new generalized version of the FKNN classifier, which was introduced and tested in Publication **III**. It starts with a short introduction to the new classifier, its process, and then presents a performance analysis of the proposed classifier using artificial and real-life data sets that consist of binary and multi-class problems.

4.1 Introduction

This new classifier is an extension of the previously introduced MLPM-FKNN method (Publication **I**) that uses multi-local mean vectors and the Power mean. The importance of utilizing class representative local mean vectors in the FKNN (Keller et al., 1985) learning has emerged in the previous chapter. Typically, the classification in the FKNN method is based on the most common class and distance between the query sample and its nearest neighbors. The distance that can be regarded as imprecision in terms of the similarity of individual instances has a clear impact on the classification. The problems with "individual imprecision" can be overcome using averaging operators—this can also be interpreted as "wisdom of the crowd," and Aristotle (Aristotle, 4th century BC) was the first to discuss this problem. Later, this notion was made famous by Galton (1949) through a notable example of a country-fair contest of weight estimation. According to these forerunners, it can be expected that the use of local means should provide a better predictive capability than the individuals alone.

The focus of present work is to strengthen the local means-based learning in the original FKNN method by using a powerful averaging operator, the so-called Bonferroni mean (Bonferroni, 1950). Thus, the local mean is needed to be better than any of its components (i.e., training instances) in terms of the similarity to new instance. Accordingly, the MLPM-FKNN (Publication **I**) algorithm is expanded by the use of the Bonferroni mean instead of the Power mean for local mean computation and hence propose a new classifier called the "Bonferroni mean-based fuzzy *k*-nearest neighbor (BM-FKNN) classifier". The objective of the new classifier is to show that the Bonferroni mean vectors provide better learning than any other type of mean (e.g., the generalized mean) and improve classification accuracy.

The Bonferroni mean (Bonferroni, 1950) is probably the most advanced aggregation operator for fusing data within an extensive range of possibilities. Additionally, some studies have reported that the arithmetic mean may not always be ideal for producing the optimal results with classification models; instead, the model performance can be improved by using alternative mean operators, for instance, harmonic (Pan et al., 2017), generalized (Luukka et al., 2001), and ordered weighted average (OWA) (Luukka and Kurama, 2013) means. Moreover, as the arithmetic mean is a particular case of the Power mean, it can be observed that the results achieved with the Power mean are at least as good as with the arithmetic mean and often better. In the same way, as the Power mean is

a special case of the Bonferroni mean, the results achieved with the Bonferroni mean are expected to be at least as good as with the Power mean and can be even better. When the Bonferroni mean is utilized to create the class prototype local mean vectors, it should be noted that there is a possibility of optimizing the parameters to fit a particular situation (a data set). Here, adjusting the Bonferroni mean parameters allows for the search of optimal parameter values, which will lead to better classification performance.

The performance of the proposed BM-FKNN classifier was evaluated using one artificial data set to show how the proposed method performs with respect to the class imbalance ratio and then some real-world data sets chosen from machine learning repositories to assess its efficacy in the real-world applications. It was remarked in Publication **I** that the new MLPM-FKNN classifier could be a valuable alternative for dealing with class imbalance problems. However, how the local means-based FKNN method performs concerning the imbalance ratio was not shown explicitly. Given this issue, in Publication **III**, a special attention was paid to validate the ability of the BM-FKNN classifier in a complete range of class imbalance problems by using one artificial data set generated in controlled settings.

4.2 Description

In this section, the underlying process of the BM-FKNN algorithm is established. However, this process is similar (but not identical) to that of the MLPM-FKNN algorithm (Publication **I**). Therefore, every step of this algorithm is not discussed in detail, but in short, in line with Section 3.3, where a detailed procedure of the MLPM-FKNN algorithm is presented.

Formally, the MLPM-FKNN algorithm is modified to the present context, which comes in that class representative vectors in the learning part are computed using the Bonferroni mean. Consider a training set $\{X_j, c_j\}_{j=1}^{N}$, where $X_j = \{x_j^1, x_j^2, .., x_j^{\mathcal{D}}\} \in \mathbb{R}^{\mathcal{D}}$ and $c_i \in \{\omega_1, \omega_2, .., \omega_S\}$. Given a query sample $Y \in \mathbb{R}^{\mathcal{D}}$, corresponding class ω_c is determined in the BM-FKNN algorithm as in the following. In the first step, the distances $\{d_{EUC}(Y, X_j)\}_{j=1}^{N}$ between Y and each training instance $X_j \in \mathbb{R}^{\mathcal{D}}$ are estimated and next, a set of k nearest neighbors $nn^k(Y)$ is found. This is followed by grouping the set $nn^k(Y)$ into subsets based on the classes $\{\omega_i\}_{i=1}^{S_k}$, which is represented by the nearest neighbors. These subsets are then used to calculate the Bonferroni mean vectors $\{\mathcal{B}_i^{p,q}\}_{i=1}^{S_k}$. This can be expressed using the Equation (2.2) as:

$$\mathcal{B}_i^{p,q} = \left(\frac{1}{n_i}\sum_{j=1}^{n_i} X_j^p \left(\frac{1}{n_i - 1}\sum_{j,l=1,l\neq j}^{n_i} X_l^q\right)\right)^{\frac{1}{p+q}} \quad \text{for } i = 1, 2, ..., S_k \tag{4.1}$$

where n_i is the number of instances in the i^{th} class, and X_j and X_l represent the j^{th} and l^{th} instances, respectively. The parameters p and q matter greatly because these two key choices will typically push the class averaging vector to be fitted properly to the problem at hand. Moreover, it is clear that the number of the Bonferroni mean vectors depends on

the number of classes that are represented by the k nearest neighbors.

Next, the Euclidean distances $\{d_{EUC}(Y, B_j)\}_{j=1}^{S_k}$ between Y and each local Bonferroni mean vector $\mathcal{B}_j$ are calculated. Using these distances, membership degrees (u_i) to the query sample Y for the classes $\{\omega_i\}_{i=1}^{S_k}$ are assigned according to

$$u_i(Y) = \frac{\sum_{j=1}^{S_k} u_{ij}(1/d_{EUC}(Y, \mathcal{B}_j^{p,q})^{2/(r-1)})}{\sum_{j=1}^{S_k}(1/d_{EUC}(Y, \mathcal{B}_j^{p,q})^{2/(r-1)})} \tag{4.2}$$

where, $u_{ij} = \begin{cases} 1, & \textbf{if } c_i = c_j \\ 0, & \textbf{if } c_i \neq c_j \end{cases}$ for $j = 1, ..., S_k$ and $r \in (1, +\infty)$ is the fuzzy strength parameter. Practically, this step maps the set of responses to a membership degree by computing a dot product of the inverse of the distances and crisp memberships of the neighboring instances. In the final step, the query sample Y is assigned to the class ω_c with that the local mean vector has the highest membership degree. That is,

$$\omega_c = \arg \max_{\omega_i} u_i(Y) \tag{4.3}$$

This process is summarized as a pseudo-code in Algorithm 2. The main contribution of the proposed method is that it uses the local mean vectors inspired by the Bonferroni mean for all classes which are represented by the k nearest neighbors. The locally created prototype vectors for each class are expected to be well-positioned in perceiving class information in the similarity calculation to the query sample. In this way, the local mean computation was improved in the proposed method in Publication **III** to solve the class imbalance problems. Additionally, it also aims to remedy the issues that come with imprecise data in which the observations from different classes have very similar characteristics (Liu et al., 2013).

4.3 Data and Testing Methodology

The empirical study consisted of two separate phases: first, the performance of the proposed method was examined with "artificially generated data" to understand how well new classifier might perform on the imbalanced data. For the task, the imbalance ratio of the data (a binary problem) was progressively changed and the model's performance was assessed in each case. In the second phase, in addition to presenting performance with the artificial data, the proposed model was also tested with several real-life data sets acquired from the well-known machine learning repositories. More details of the data used and applicable study can be found in Publication **III**.

The generated data set consisted of two classes: class $1 \sim \mathcal{N}(9, 4^2)$ with 10 features and a sample size of 100, and class $2 \sim \mathcal{N}(10, 6^2)$ with 10 features and a sample size of n that was adapted from the set $\{100, 90, 80, ..., 20, 10\}$. In this way, the data was adjusted with imbalance ratio $\{1/1, 1/0.9, 1/0.8, ..., 1/0.1\}$ in increasing order and the performance of each classifier was evaluated for each case. In the subsequent analysis, six real-world data

Algorithm 2: BM-FKNN algorithm (adapted from Publication **III**)

Input: training data $\{X_j, c_j\}_{j=1}^{N}$, query sample Y, k, and p
Output: class label ω_c for Y

Begin
for $j \leftarrow 1$ **to** N **do**
  calculate $d_{EUC}(Y, X_j) \leftarrow \|Y - X_j\|$
  if $j < k$ **then**
    add X_j to $nn^k(Y)$
  else if X_j *is closer to* Y *than any of neighbors in* $nn^k(Y)$ **then**
    drop the farthest neighbor from the set $nn^k(Y)$ and add X_j
  end
end
for $i \leftarrow 1$ **to** S_k **do**
  find $\mathcal{B}_i^{p,q} \leftarrow (\frac{1}{n_i}\sum_{j=1}^{n_i}(X_{ij}^{nn})^p)^{1/p}$
  calculate $d_{EUC}(Y, \mathcal{B}_i^{p,q}) \leftarrow \|Y - \mathcal{B}_i^{p,q}\|$
end
for $i \leftarrow 1$ **to** S_k **do**
  calculate class memberships for Y according to:

$$u_i(Y) \leftarrow \frac{\sum_{j=1}^{S_k} u_{ij}(1/d_{EUC}(Y, \mathcal{B}_j^{p,q})^{2/(r-1)})}{\sum_{j=1}^{S_k}(1/d_{EUC}(Y, \mathcal{B}_j^{p,q})^{2/(r-1)})}$$

$$\text{where, } u_{ij} = \begin{cases} 1, & \textbf{if } c_i = c_j \\ 0, & \textbf{if } c_i \neq c_j \end{cases} \text{ for } j = 1, ..., S_k$$

end
return ω_c such that

$$\omega_c = \arg\max_{\omega_i} u_i(Y)$$

sets were obtained from the public UCI (Dheeru and Taniskidou, 2017) and KEEl (Alcala-Fdez et al., 2011) machine learning repositories to test whether the proposed classifier can effectively solve conventional classification problems. The properties of these data sets are summarized in a condensed way in Table 4.1 with a particular focus on the last column representing the relevant area of each data set used.

The evaluation was based on the standard performance measures, including accuracy, sensitivity, and specificity. In addition to them, variance and confidence interval (CI) were also reported. To ensure a fair comparison, the performances of the proposed approach was benchmarked with the classical KNN (Cover and Hart, 1967) and FKNN (Keller et al., 1985) methods and several more competitive methods, including LM-KNN (Mitani and Hamamoto, 2006), SVM (Cortes and Vapnik, 1995), Naive Bayes (Lewis, 1998), and similarity classifier (Luukka et al., 2001). In addition to them, an enhanced

Table 4.1: Information on the real-life data obtained from the machine learning repositories [modified from Publication **III**].

Data set	Database	Instances	Features	Classes	Domain
Car	KEEL	1728	6	4	Engineering
Ionosphere	UCI	351	34	2	Physics
Vehicle	KEEL	846	18	2	Engineering
Mammography	UCI	961	6	2	Medicine
Wine	UCI	178	13	3	Chemistry
Page Blocks	KEEL	548	10	5	Computer Science

variant of the LM-KNN (Mitani and Hamamoto, 2006) method was implemented based on the Bonferroni mean and its performance was also examined. This was the second new variant (referred as BM-KNN) introduced in this work (as requested by the reviewers for Publication **III**).

The number of nearest neighbors k was presented from the set $\{1, 2, ..., 25\}$ under the assumption that the performance of the new introduced methods would (relatively) increase when the k-value increases. The values of the parameters p and q for the Bonferroni mean were varied from the range $\{0, 1, ..., 10\}$. Moreover, all model parameters were tuned by cross-validated performance in terms of the accuracy across the grid search (for the complete process see Figure 2.4). Along the way, the best classifiers with the optimal parameters were set up for testing using the test sets in the final step.

4.4 Results

4.4.1 Artificial Data

An artificial data set was used to assess the sensitivity of classifiers' performance to class imbalance. Figure 4.1 illustrates the curves of mean accuracies among the six classifiers with class imbalance ratios. It is important to note that the mean accuracies shown in the figure were taken in the testing phase for all KNN-based classifiers. The imbalance ratio (X-label) given in Figure 4.1 indicates a sample percentage of the majority class with respect to the minority class. For example, "$1/0.5$" indicates a ratio that class 1 has a sample size of 100 and class 2 has a sample size of 50.

At first glance, it is apparent that both proposed BM-FKNN and BM-KNN classifiers yielded the same performance. Further, the performance of each classifier is at its best when the lowest number of instances represents one class as compared to the other class. It can also be seen the mean accuracy of each classifier gradually increases when the imbalance ratio rises. Additionally, it is apparent that both proposed methods achieved better performances than the benchmarks concerning all imbalance ratios. Altogether, this result implies that the new BM-FKNN and BM-KNN classifiers are less sensitive to the class imbalance than the benchmarks.

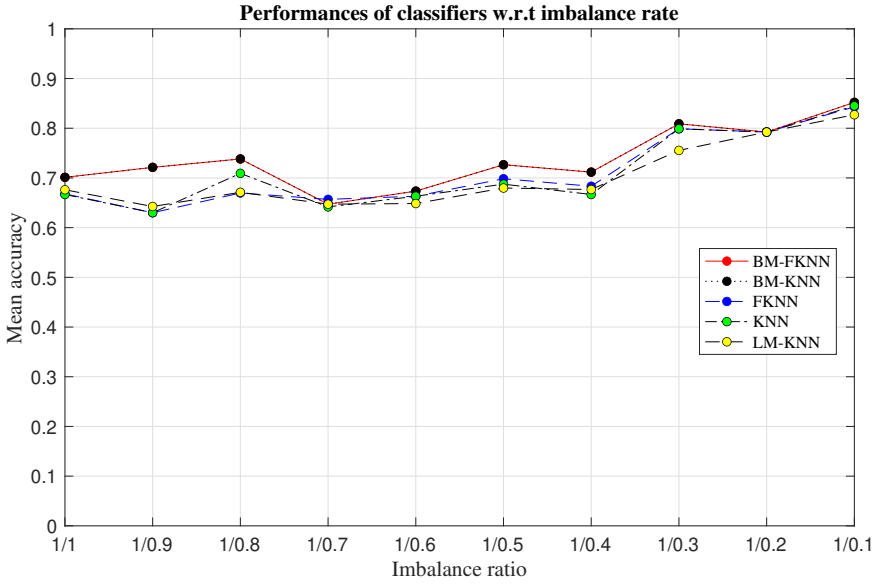


Figure 4.1: Classification accuracy with respect to the class imbalance ratio [adapted from Publication **III**].

4.4.2 Real-world Data

Now, the classification performances with the data sets from real-world applications are discussed. A 30-fold holdout cross-validation was performed to optimize the parameters of new classifiers and the benchmarks in the training and validation phase. The results in terms of the performance metrics and optimal parameter values are summarized as an example for the proposed methods in Table 4.2. In the table, the accuracy, sensitivity, and specificity represent their mean values from the 30-fold sample corresponding to the optimal parameter values. It should be noted that both proposed approaches produced the same results.

It is apparent that both proposed methods performed well in the validation sets (yielding the mean accuracy of 92.71%, 87.75%, 93.4%, 79.39%, 74.14%, and 93.58% for Car, Ionosphere, Vehicle, Mammogram, Wine, and Page Blocks data sets, respectively). The sensitivity and specificity scores also appear reasonable; these values matter greatly to ensure that classifications made by new approaches are meaningful and advocate in terms of each class available. Moreover, it is visible that the values of sensitivity are lower than the specificity for all cases except the Page Blocks data. As for the optimal parameter values, it can be seen that the proposed approach achieved the highest accuracy with the settings $p \in \{1, 2, 3\}$ and $q \in \{0, 1, 2\}$ in all data sets used.

Table 4.3 compares the classification results of all methods on the test sets that were initially separated from the original data. The performances are demonstrated in terms of

Table 4.2: Average classification performance of the BM-FKNN and BM-KNN classifiers in the training and validation step [modified from Publication **III**].

Data set	Accuracy	Sensitivity	Specificity	Optimal parameters		
Car	0.9271	0.8079	0.9637	$k = 3$	$p = 1$	$q = 1$
Ionosphere	0.8775	0.8611	0.9245	$k = 7$	$p = 1$	$q = 0$
Vehicle	0.9340	0.8557	0.9591	$k = 4$	$p = 3$	$q = 1$
Mammogram	0.7939	0.7901	0.7984	$k = 21$	$p = 2$	$q = 1$
Wine	0.7414	0.7388	0.8730	$k = 25$	$p = 2$	$q = 2$
Page Blocks	0.9358	0.9775	0.8679	$k = 3$	$p = 2$	$q = 2$

the mean values of accuracy, sensitivity, specificity, and the variances as well as CIs. The test results for the BM-FKNN and BM-KNN are shown in the same column because they obtained exact results for all data sets.

The results in Table 4.3 illustrate that the proposed classifiers outperformed all benchmarks with two data sets (Car and Mammogram) and achieved the second-best performance with three data sets (Vehicle, Ionosphere, and Page Blocks). The mean sensitivity and specificity values remained high for all data sets, showing good selectivity. Interestingly, the results for both BM-FKNN and BM-KNN methods were the same for all test sets. This indicates that using the Bonferroni mean within learning part of the classifier has a substantial impact in comparison to calculating the membership degree in the FKNN rule. Moreover, as compared to the FKNN, KNN, and LM-KNN approaches, the BM-KNN and BM-KNN classifiers significantly improved the classification accuracy. This exhibits that by introducing the notions of local mean vectors based on the Bonferroni mean as nearest prototypes instead of k nearest neighbors, reasonable class representative vectors can be created and the performance can be enhanced. Additionally, even though the SVM, NB, and Similarity classifiers obtained relatively higher accuracy in some cases, the BM-FKNN and BM-KNN still outperformed them on most data sets. Overall, it is clear based on the test results that newly proposed classifiers offered improved performance in comparison to the benchmarks. To empirically verify the significance of this improvement, a paired t-test at 0.05 level of significance was performed to compare their classifications with the best cases of the benchmarks. Table 4.4 shows the test results confirming a statistically significant difference between the performances in cases where the proposed approaches produced the best accuracy. Regarding computational complexity, the proposed BM-FKNN and BM-KNN classifiers required a longer time than the benchmarks. For example, the average computational time of BM-FKNN, BM-KNN, FKNN, KNN, LM-KNN, SVM, BM, and Similarity classifier on the test data of Page-blocks were $\{0.0812, 0.0756, 0.0043, 0.0041, 0.0099, 0.0072, 0.0181, 0.0354\}$ in seconds, respectively. Similar results were observed with the rest of data sets, indicating that the computational complexities of the new methods are higher than the standard classifiers.

Table 4.3: Comparison of the classifiers' performances with the test data sets.

Data set	Measure	BM-FKNN/ BM-KNN	FKNN	LM-KNN	KNN	SVM	NB	Similarity classifier
Car	Mean Accuracy	**0.9292**	0.8905	0.8956	0.8845	0.8506	0.8158	0.6984
	Mean Sensitivity	0.8251	0.6580	0.6345	0.6173	0.7000	0.4005	0.6599
	Mean Specificity	0.9659	0.9291	0.9197	0.9142	0.9133	0.8946	0.9017
	Variance	1.3673e-04	1.5642e-04	1.9778e-04	1.486e-04	1.6715e-04	2.5048e-04	2.2885e-04
	CI	[0.9237 0.9347]	[0.8846 0.8963]	[0.8890 0.9022]	[0.8795 0.8895]	[0.8445 0.8566]	[0.8083 0.8232]	[0.6913 0.7055]
Ionosphere	Mean Accuracy	0.8914	0.8529	0.8914	0.8529	0.8906	**0.9164**	0.8621
	Mean Sensitivity	0.8562	0.8240	0.8562	0.8240	0.8601	0.9446	0.8737
	Mean Specificity	0.9523	0.9541	0.9523	0.9541	0.9649	0.8762	0.8596
	Variance	4.5972e-04	0.0013	4.5972e-04	0.0013	4.8281e-04	9.2965e-04	0.0026
	CI	[0.8814 0.9015]	[0.8362 0.8695]	[0.8814 0.9015]	[0.8362 0.8695]	[0.8813 0.9009]	[0.9022 0.9307]	[0.8382 0.8861]
Vehicle	Mean Accuracy	0.9556	0.9456	0.9371	0.9456	**0.9562**	0.7015	0.6988
	Mean Sensitivity	0.8852	0.8790	0.8695	0.8790	0.9077	0.4351	0.4315
	Mean Specificity	0.9796	0.9670	0.9594	0.9670	0.9714	0.9444	0.9359
	Variance	1.3821e-04	7.9608e-05	2.7679e-04	7.9608e-05	8.0124e-04	2.2657e-04	8.8085e-05
	CI	[0.9501 0.9611]	[0.9414 0.9497]	[0.9309 0.9434]	[0.9414 0.9497]	[0.9430 0.9695]	[0.6944 0.7085]	[0.6944 0.7032]
Mammogram	Mean Accuracy	**0.7927**	0.7844	0.7909	0.7833	0.7906	0.7844	0.7789
	Mean Sensitivity	0.7528	0.7275	0.7437	0.7303	0.7819	0.7457	0.7054
	Mean Specificity	0.8343	0.8535	0.8434	0.8456	0.8058	0.8270	0.8850
	Variance	1.1593e-04	1.8675e-04	1.1786e-04	8.6806e-05	1.1593e-04	1.1536e-04	2.9911e-05
	CI	[0.7877 0.7977]	[0.7780 0.7908]	[0.7858 0.7960]	[0.7790 0.7877]	[0.7856 0.7957]	[0.7793 0.7894]	[0.7763 0.7815]
Wine	Mean Accuracy	0.8306	0.8097	0.8306	0.8069	0.8833	**0.9722**	0.9681
	Mean Sensitivity	0.8110	0.7897	0.8110	0.7897	0.8712	0.9722	0.9724
	Mean Specificity	0.9105	0.9012	0.9105	0.9012	0.9379	0.9848	0.9846
	Variance	0.0020	0.0028	0.0020	0.0028	0.0022	2.0759e-31	1.0356e-04
	CI	[0.8095 0.8516]	[0.7850 0.8344]	[0.8095 0.8516]	[0.7822 0.8317]	[0.8616 0.9051]	[0.9722 0.9722]	[0.9633 0.9728]
Page Blocks	Mean Accuracy	0.9255	0.9200	0.9150	0.9191	0.8918	**0.9259**	0.6586
	Mean Sensitivity	0.9597	0.9619	0.9597	0.9615	0.9337	0.9890	0.9954
	Mean Specificity	1	1	1	1	NaN	0.7366	0.4735
	Variance	3.7416e-05	2.7416e-05	3.7190e-05	2.5572e-04	2.5228e-05	3.0687e-04	0.0018
	CI	[0.9219 0.9290]	[0.9165 0.9235]	[0.9121 0.9179]	[0.9165 0.9235]	[0.8895 0.8942]	[0.9211 0.9307]	[0.7185 0.7579]
	Average (overall)	**0.8875**	0.8672	0.8767	0.8654	0.8772	0.8527	0.7775

4.5 Conclusion and Limitations

This chapter introduced a new variant of the FKNN method, called the BM-FKNN, which uses the local mean vectors based on the Bonferroni mean. Introducing new parameters with the Bonferroni mean may allow the model to maintain classification performance in a broader range; thus, the proposed classifier can be readily fitted to numerous contexts and applications. To demonstrate the capability of the BM-FKNN classifier, it was tested with one artificial and six real-world data sets in comparison to the selected state-of-the-art methods, including the FKNN, KNN, LM-KNN, SVM, NB, and similarity classifiers. Additionally, an improvement of the LM-KNN method based on the Bonferroni mean was also proposed and tested. This new version was called the BM-KNN classifier. Interestingly, both new BM-FKNN and BM-KNN methods showed exactly the same performance with all data sets used.

The findings with artificial data verified that new methods can effectively solve class imbalance problems as compared to the standard classifiers. Further, it was observed that the performance of each classifier was at its best when the imbalance ratio was at its possible highest level (i.e., when one class has the smallest number of instances in comparison to the other). In the analysis with real-world data sets, the BM-FKNN and BM-KNN algorithms illustrated improved performance (obtaining overall average accuracy of 88.75%) compared to the six benchmarks used. Another most notable finding engaged with the proposed classifiers was that the best accuracy was attained with a relatively high value of k.

In this work, even though several weaknesses of the MLPM-FKNN (Publication **I**) method were addressed, the proposed BM-FKNN approach still has several limitations. A key limitation of the proposed BM-FKNN method is the number of parameters that need to be optimized for proper values before using the model in an actual problem case. This may increase the computational complexity of the new model compared to the standard classifiers. In fact, the question is, can the proposed approach be robust and efficient for solving problems with the large data sets (e.g., Big data), which remains open for future research. However, it is possible that a suitable parameter search technique can be involved (or developed) instead of the grid search. Despite that, this study paves the way to explore new areas of research in the FKNN classification in terms of several different perspectives. For example, it would be interesting to examine how the Bonferroni means can be used in other local means-based nearest neighbor variants, for example, MLM-KHNN (Pan et al., 2017), a generalized mean distance-based KNN classifier (GMDKNN) (Gou et al., 2019), and the pseudo nearest neighbor (PNN) method (Gou et al., 2014).

Table 4.4: Results of the t-test at 0.05 significance on the performance of the proposed methods vs. baseline models for the test data [adapted from Publication **III**].

Data set	Paired-t with BM-FKNN / BM-KNN	P-value	Test-statistic
Car	FKNN	2.4770e-12	significant
	LM-KNN	6.3027e-10	significant
	KNN	3.8770e-14	significant
	SVM	6.7520e-22	significant
	NB	1.1173e-25	significant
	Similarity classifier	1.5717e-37	significant
Ionosphere	FKNN	3.4829e-04	significant
	LM-KNN	1	not significant
	KNN	3.4829e-04	significant
	SVM	0.0025	significant
	NB	5.6239e-07	significant
	Similarity classifier	0.4808	not significant
Vehicle	FKNN	0.0042	significant
	LM-KNN	1.2410e-05	significant
	KNN	0.0042	significant
	SVM	0.9317	not significant
	NB	5.6328e-41	significant
	Similarity classifier	4.3005e-42	significant
Mammogram	FKNN	0.0387	significant
	LM-KNN	0.5970	not significant
	KNN	0.0055	significant
	SVM	0.5443	not significant
	NB	0.0190	significant
	Similarity classifier	9.3774e-06	significant
Wine	FKNN	0.0871	not significant
	LM-KNN	1	not significant
	KNN	0.0839	not significant
	SVM	1.0465e-22	significant
	NB	3.3440e-34	significant
	Similarity classifier	2.3069e-32	significant
Page Blocks	FKNN	0.0096	significant
	LM-KNN	3.7653e-06	significant
	KNN	1.023e-04	significant
	SVM	6.5585e-20	significant
	NB	0.6917	not significant
	Similarity classifier	1.4920e-21	significant

5 Proposed Fuzzy *K*-nearest Neighbor Regression with the Minkowski Distance

As mentioned at the beginning, one objective of this dissertation is to view the FKNN method (Keller et al., 1985) from the regression perspective. Accordingly, this chapter focuses on the FKNN regression and hence proposes a novel regression approach, as presented in Publication **IV**.

5.1 Introduction

The KNNreg (Stone, 1977; Benedetti, 1977; Turner, 1977) is an old and simple but widely used regression method (Nguyen et al., 2016). This approach is suitable and effective for both linear and non-linear regression tasks (Cai et al., 2020). However, this method also suffers from the underlying weaknesses of the KNN classifier (Cover and Hart, 1967), questioning its effectiveness from one problem to another. Moreover, in terms of the enhanced variants of the KNN method, the FKNN classifier is known to be one of the best-performing classifiers to solve the problems of original method. Even though the FKNN method has been widely used in classification tasks, little attention has been paid to it in the context of regression. Nikoo et al. (2018) used the FKNN classifier to a regression problem without adjusting its original algorithm expressly (in other words, the experiment was conducted as a classification problem). Apart from that, as far as we know, no one had tried to use the FKNN approach in the context of regression. This encouraged to establish a regression version of the FKNN method in Publication **IV**, which is referred to as the fuzzy *k*-nearest neighbor regression (FKNNreg) method.

In this work, the emphasis is to examine the performance of the FKNNreg method in terms of different types of regression tasks, and the goal is to formally demonstrate that using the Minkowski distance in the FKNNreg algorithm offers better predictions than using the Euclidean distance. Meanwhile, the efficacy of combining the FKNNreg and Minkowski distance is examined and a new Minkowski distance-based fuzzy *k*-nearest neighbor regression (Md-FKNNreg) algorithm is introduced. The principal advantage of this algorithm is that it attributes the importance to the nearest neighbors using "fuzzy" weights considering their distances to the query sample and hence obtains a more accurate prediction across a weighted average.

Moreover, a distance metric is a pivotal component of distance-based algorithms, such as the FKNN and KNN methods (Rastin et al., 2021). Regarding these methods, the Euclidean distance measure is the most commonly used distance metric in calculating the similarity between two instances (Nguyen et al., 2016). However, using the Euclidean distance may not always be optimal for every problem because it naturally has several limitations (Cai et al., 2020; Nguyen et al., 2016). For example, even though two instances have no attribute values in common, they may have a shorter distance than other pairs of instances, including the exact attribute values (Shirkhorshidi et al., 2015). Moreover, several studies have achieved better results for their applications with a more general

choice of the distance measure (for example, see Chang et al., 2006; Koloseni et al., 2012; Dettmann et al., 2011). This study focuses on the Minkowski distance and examines its usefulness in the FKNN method from the regression perspective.

5.2 Description

The Md-FKNNreg algorithm can be divided into four steps: computing distances, finding nearest neighbors, observing fuzzy weights, and making predictions. All these steps are discussed in detail below.

Step I is the calculation of distances between a new instance and training instances using the Minkowski distance [Equation (2.5)]. Consider $X = \{x^1, x^2, .., x^{\mathcal{D}}\} \in \mathbb{R}^{\mathcal{D}}$ is a new instance and its output value (y) that needs to be estimated. The algorithm starts with calculating the Minkowski distances $d_{Md}(X, X_j)$ between X and X_j according to

$$d_{Md}(X, X_j) = \big(\sum_{t=1}^{\mathcal{D}} |x^t - x_j^t|^{\mathcal{P}}\big)^{1/\mathcal{P}} \tag{5.1}$$

where $X_j = \{x_j^1, x_j^2, .., x_j^{\mathcal{D}}\} \in \mathbb{R}^{\mathcal{D}}$ is the j^{th} instance of the training set $\{X_j, \mathrm{y}_j\}_{j=1}^N$ given and its output value is $\mathrm{y}_j \in \mathcal{Y} = \{\mathrm{y}_1, \mathrm{y}_2, .., \mathrm{y}_N\}$. $\mathcal{P} \geq 1$ is the parameter of the Minkowski distance metric.

Step II finds a set of k nearest neighbors $nn^k(X)$ of X from the ordered training instances according to increasing Minkowski distances. Here, a grid-based search is performed to determine optimal values for k and the Minkowski distance parameter $\mathcal{P}$ to a particular data set.

Step III consists of computing fuzzy weights (w_j) for each nearest neighbor X_j using $d_{Md}(X, X_j)$ for $j = 1, 2, ..., k$ as follows:

$$w_j = \frac{1}{\big(1/d_{Md}(X,\, X_j)\big)^{\frac{2}{r-1}}} \tag{5.2}$$

where $r \in (1, +\infty)$ is the fuzzy strength parameter and $(\frac{2}{r-1})$ is the fuzziness exponent. When the r is close to one, it produce larger weights. The intuition behind these weights is to define a proper linear predictor for the output value y such that $h(X) = W^T\mathcal{Y}$. Each weight value $w_j$indicates a relative importance of each nearest neighbor X_j to $\mathcal{Y}$.

Step IV is the prediction of output value for the new instance. By taking (fuzzy) weighted averaged of the outputs $\{\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_k\}$ of the nearest neighbors, the output value y of X is estimated as follows:

$$\hat{\mathrm{y}} = \frac{\sum_{j=1}^k w_j \mathrm{y}_j}{\sum_{j=1}^k w_j} \tag{5.3}$$

where $\hat{y}$ is the estimated value of y. In the KNNreg method, a uniform weighting scheme (Cheng, 1984) is used to perform the prediction. In other words, it predicts the output of the new instance by taking the arithmetic average of the outputs of the nearest neighbors selected. However, it does not consider the distances from the new instance to the nearest neighbors, giving equal contributions to all nearest instances, which is the main drawback of the KNNreg method (Kramer, 2011). On the contrary, an inverse weighting strategy that allocates higher weights to the nearest training instances is used in the Md-FKNNreg method. In this way, a fuzzy version allows closer instances more influence in the prediction.

Furthermore, the Minkowski distance in the Md-FKNNreg method is worked not only to calculate the distances between each training instance and new instance but also to observe the weights for nearest neighbors. Therefore, it is clear that the Minkowski distance measure plays a significant role in the proposed approach. The Md-FKNNreg method has two important parameters: k, the number of nearest neighbors and p, the parameter of the distance metric. These allow the model to extend the search area to a broader region and adapt to a particular situation with optimal conditions. A pseudo-code for the Md-FKNNreg process is provided in Algorithm 3. Additionally, Algorithm 4 presents the pseudo-code of applied grid search method for parameter optimization.

The main contributions of this work can be summarized as follows:

- The original FKNN (Keller et al., 1985) method is extended so that it can be applied in the regression context.
- A novel regression approach is proposed by introducing the Minkowski distance into the nearest neighbor search in the FKNN method.
- The efficiency and robustness of the proposed method are examined in terms of linear and non-linear regression problems using several low- and high-dimensional real-world data sets.

5.3 Data and Testing Methodology

To study the performance of the proposed Md-FKNNreg method, eight real-life data sets in low- and high-dimensional spaces were selected, which are publicly available at the KEEL (Alcala-Fdez et al., 2011) and UCI (Dheeru and Taniskidou, 2017) machine learning repositories. Table 5.1 summarizes the characteristics of each data set used, including the number of instances ("Instances"), the number of features ("Features"), and the related area of each data set ("Domain").

The experimental context involved in Publication **IV** to evaluate the performance of the proposed regression approach is summarized in Figure 5.1. All the aspects shown in the

Algorithm 3: Md-FKNNreg algorithm (adapted from Publication **IV**)

Input: $\{X_j, \mathrm{y}_j\}_{j=1}^{N}$ (training set), X (test instance), $k\,(1 \leq k \leq N)$, $\mathcal{P} \geq 1$
Output: predicted output value, $\hat{\mathrm{y}}$ for X

Begin
for $j \leftarrow 1$ **to** N **do**
  Compute $d_{Md}(X, X_j)$ between X and X_j according to
  $d_{Md}(X, X_j) = \left(\sum_{t=1}^{\mathcal{D}} |x^t - x_j^t|^{\mathcal{P}}\right)^{1/\mathcal{P}}$
  if $j < k$ **then**
    Add X_j to N_X^k
  else if X_j *is closer to* X *than any of neighbors in* $nn^k(X)$ **then**
    Drop the farthest neighbor from the set $nn^k(X)$ and add X_j
  end
end
for $j \leftarrow 1$ **to** k **do**
  Compute $w_j = \frac{1}{\left(1/d_{Md}(X, X_j)\right)^{\frac{2}{r-1}}}$
end
Estimate $\hat{\mathrm{y}}$ by taking weighted average such as $\hat{\mathrm{y}} = \frac{\sum_{j=1}^{k} w_j \mathrm{y}_j}{\sum_{j=1}^{k} w_j}$
return $\hat{\mathrm{y}}$

flow chart were performed as explained in Section 2.6. In particular, testing of the proposed method using the real-life data consisted of two phases—training and validation—and testing. The model was optimized for optimal parameter values in the training and validation step, and the generalizability of the finalized model was then assessed with the test data.

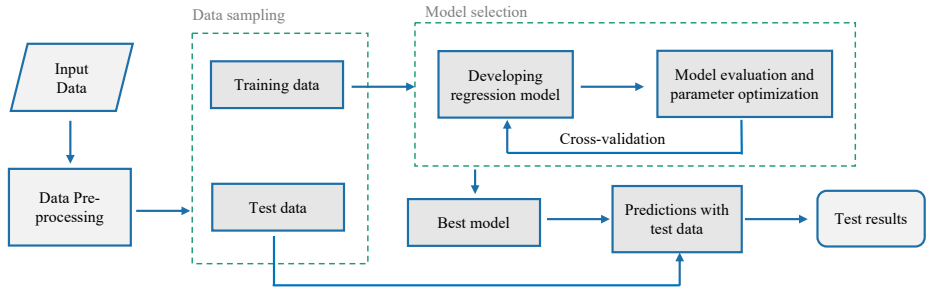


Figure 5.1: A flowchart of the Md-FKNNreg model development and evaluation.

The performance of the proposed Md-FKNNreg method was compared with the original KNNreg, as well as SVR (Drucker et al., 1997), least absolute shrinkage and selection (LASSO) (Tibshirani, 1996), and multiple linear regression (MLR) (Montgomery et al., 2012) methods. In addition to them, the Manhattan distance-based fuzzy *k*-nearest neighbor regression (Man-FKNNreg) and the Euclidean distance-based fuzzy *k*-nearest

Algorithm 4: The grid search algorithm (adapted from Publication **IV**)

Input: $\mathcal{P}$-values: $\{\mathcal{P}_i\}_{i=1}^{n_\mathcal{P}}$, k-values: $\{k_i\}_{i=1}^{n_k}$, n-runs for cross-validation
Output: Optimal $\mathcal{P}^*$ and k^*

```
Begin
for j ← 1 to n do
    Split the training data randomly into training and validation sets
    for j ← 1 to n_k do
        for s ← 1 to n_P do
            Perform Md-FKNNreg model with k(j) and P(s) and save the RMSE
        end
    end
end
for j ← 1 to n_k do
    for s ← 1 to n_P do
        Average the RMSE over n-runs
    end
end
Estimate the minimum RMSE average and corresponding P* and k* values
return P* and k*
```

neighbor regression (Euc-FKNNreg) methods were also developed and the results were compared to examine the effectiveness of using the Minkowski distance. The number of k nearest neighbors was chosen from the set $\{1, 2, ..., 15\}$ for all KNN-based regression methods. The Minkowski distance parameter $\mathcal{P}$ was selected from the range $\{1, 1.5, ..., 5\}$. For all Md-FKNNreg, Man-FKNNreg, and Euc-FKNNreg methods, fuzzy strength parameter r was remained constant at 1.5 as per suggestion by Arif et al. (2010). In addition, three different kernel functions (linear, radial basis function, polynomial) with the SVR model, five different values $\{0.001, 0.01, 1, 10, 100\}$ for the regularized parameter of the LASSO method, and four different regression types (linear, interaction, pure-quadratic, and quadratic) with the MLR model were tested to fit the best version of benchmarks. Finally, the evaluation of performances was based on RMSE and R^2 values. Detailed information about the experimental process and parameter settings can be found in Publication **IV**.

5.4 Results

5.4.1 Training and Validation Phase

Table 5.2 illustrates the regression results (in terms of the average RMSEs and standard deviations) of the proposed method and each benchmark for all data sets. From the error results in Table 5.2, it is apparent that the proposed Md-FKNNreg model outperformed all other state-of-the-art methods in six data sets (see the bold ones) and obtained the second-best results for two data sets (Servo and Laser). The standard deviation values

Table 5.1: Information on the data sets used.

Data set	Database	Instances	Features	Domain
Stock	KEEL	950	9	Business
Airfoil	UCI	1503	5	Physics
Laser	KEEL	993	4	Physics
AutoMPG	KEEL	392	6	Engineering
Baseball	KEEL	337	16	Sociology
Parkinson	UCI	5875	26	Medicine
Servo	UCI	167	4	IT
Qsar Fish	UCI	908	6	Biology

also remained reasonable, indicating consistency of the proposed method during the cross-validation. Additionally, it is visible that both Md-FKNNreg and Man-FKNNreg models yielded the same results in six data sets. It can be also seen that the Md-FKNNreg method performed better than the Euc-FKNNreg method in all data sets, though the results were comparable in some cases. This reflects the usefulness of using the Minkowski distance for distance calculation in the FKNN regression. Regarding R^2 values (see Figure 5.2), the same indications (as shown in Table 5.2) can be observed about the regression performance of all methods in all data sets. Notably, with higher R^2 values, the proposed method indicated a good overall fit to the data.

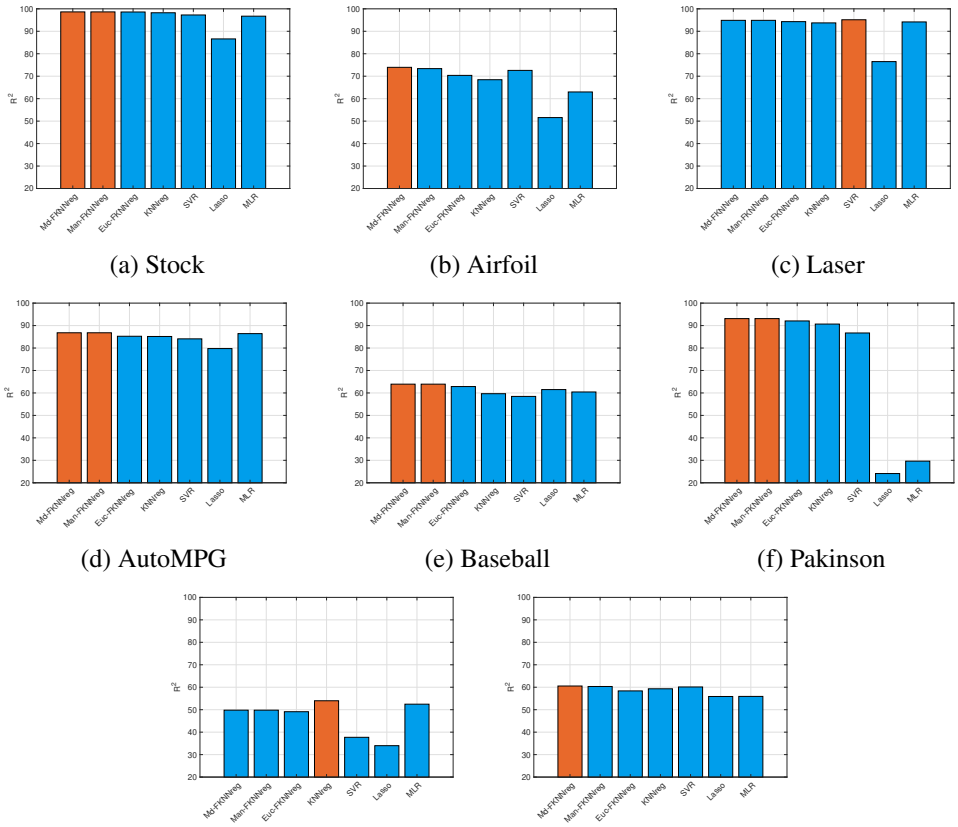
Considering optimal parameter values, it was observed that $\mathcal{P} = 1$ (i.e., Manhattan distance) produced a better performance with the proposed method for most data sets. This is clearly indicated by the results in Table 5.2 and Figure 5.2, in which the Man-FKNNreg shared the same results with the Md-FKNNreg method for six data sets. This finding is somewhat consistent with the study by Aggarwal et al. (2001) that showed the Manhattan distance could be the most suitable option for high-dimensional data. Moreover, relatively high k values (ranging from 2–25) were better suited to the FKNNreg versions, while low k values (ranging from 1–13) were better for the original KNNreg method. Meanwhile, the RBF kernel seemed to be the most suited for the SVR model with most data sets, and $\lambda = 0.001$ showed the optimum for LASSO model (see Publication **IV** for more information about the optimal parameter values resulted in each method).

5.4.2 Testing Phase

The models fitted with optimal parameter values were evaluated using test data sets in the testing phase. The results of average RMSEs and standard deviations (STDs) of each method are summarized for all data sets in Table 5.3. The average computational time ("Comp. time," in seconds) taken by each regression model during the testing is also reported.

Table 5.2: The average RMSE of each method and its corresponding standard deviation (STD) in the training and validation phase [adapted from Publication **IV**].

Data set	Measure	Md-FKNNreg	Man-FKNNreg	Euc-FKNNreg	KNNreg (Biau et al., 2012)	SVR (Drucker et al., 1997)	LASSO (Tibshirani, 1996)	MLR (Montgomery et al., 2012)
Stock	RMSE	**0.0272**	**0.0272**	0.0276	0.0308	0.0381	0.0854	0.0421
	STD	0.0022	0.0022	0.0023	0.0025	0.0062	0.0029	0.0026
Airfoil	RMSE	**0.0932**	0.0994	0.0942	0.1027	0.0957	0.1273	0.1113
	STD	0.0047	0.0049	0.0049	0.0046	0.0037	0.0031	0.0030
Laser	RMSE	0.0407	0.0407	0.0431	0.0451	**0.0396**	0.0885	0.0438
	STD	0.0083	0.0083	0.0078	0.0084	0.0076	0.0054	0.0050
AutoMPG	RMSE	**0.0769**	**0.0769**	0.0812	0.0815	0.0843	0.0952	0.0779
	STD	0.0039	0.0039	0.0050	0.0047	0.0075	0.0045	0.0043
Baseball	RMSE	**0.1235**	**0.1235**	0.1254	0.1307	0.1326	0.1276	0.1293
	STD	0.0096	0.0096	0.0092	0.0091	0.0083	0.0072	0.0071
Parkinson	RMSE	**0.0583**	**0.0583**	0.0626	0.0678	0.0810	0.1936	0.1865
	STD	0.0028	0.0028	0.0027	0.0022	0.0036	0.0016	0.0015
Servo	RMSE	0.1611	0.1611	0.1612	**0.1549**	0.1818	0.1861	0.1579
	STD	0.0221	0.0221	0.0185	0.0157	0.0296	0.0145	0.0150
Qsar Fish	RMSE	**0.0966**	0.0969	0.0993	0.0981	0.0971	0.1022	0.1021
	STD	0.0036	0.0036	0.0035	0.0039	0.0037	0.0042	0.0042



(a) Stock (b) Airfoil (c) Laser

(d) AutoMPG (e) Baseball (f) Pakinson

(g) Servo (h) Qsar Fish

Figure 5.2: R^2 values of each model for each data set. The bars in orange color represent the highest R^2, while other (blue) bars represent the rest of values [modified from Publication **IV**].

The results in Table 5.3 demonstrate that the proposed method outperformed all benchmarks in all data sets, proving the ability of the Md-FKNNreg model for regression applications. In the Airfoil and Qsar Fish data sets, regression error was lower with the Md-FKNNreg method than any other regression model, and with six data sets, the lowest error was shared with the Man-FKNNreg method. Moreover, it can be seen from the table results that the proposed method offered significantly higher performance than the Euc-FKNNreg model with all data sets. Altogether, this implies that using Minkowski distance in the part of learning can obtain more appropriate nearest neighbors and lead to a better performance than using the Manhattan or Euclidean distance. Moreover, though the KNNreg and SVR models showed the lowest error in several data sets during the training and validation, they were outperformed by the proposed Md-FKNNreg method in all test cases. Regarding the testing times, it is apparent that the proposed method has taken a relatively higher time than the other models. This might be because of an additional calculation part with the Minkowski distance in the learning part of the Md-FKNNreg.

Table 5.3: The average RMSEs and corresponding STDs as well as the computation time (Com. time) of each method in the testing phase [adapted from Publication **IV**].

Data set	Measure	Md-FKNNreg	Man-FKNNreg	Euc-FKNNreg	KNNreg (Biau et al., 2012)	SVR (Drucker et al., 1997)	LASSO (Tibshirani, 1996)	MLR (Montgomery et al., 2012)
Stock	RMSE mean	**0.0294**	**0.0294**	0.0302	0.0311	0.0406	0.0762	0.0407
	STD	0.0016	0.0016	0.0017	0.0017	0.0041	0.0007	0.0011
	Com. time	0.0230	0.0171	0.0200	0.0207	0.0009	0.0002	0.0082
Airfoil	RMSE mean	**0.0963**	0.0966	0.1002	0.1036	0.0986	0.1342	0.1182
	STD	0.0046	0.0039	0.0046	0.0048	0.0022	0.0010	0.0013
	Com. time	0.0428	0.0306	0.0264	0.0233	0.0009	0.0001	0.0030
Laser	RMSE mean	**0.0435**	**0.0435**	0.0441	0.0527	0.0483	0.0986	0.0509
	STD	0.0094	0.0094	0.0124	0.0106	0.0158	0.0026	0.0058
	Com. time	0.0237	0.0168	0.0192	0.0175	0.0004	0.0002	0.0030
AutoMPG	RMSE mean	**0.0687**	**0.0687**	0.0719	0.0728	0.0707	0.0824	0.0725
	STD	0.0024	0.0024	0.0023	0.0028	0.0036	0.0015	0.0020
	Com. time	0.0098	0.0066	0.0075	0.0066	0.0004	0.0002	0.0020
Baseball	RMSE mean	**0.1184**	**0.1184**	0.1239	0.1316	0.1448	0.1329	0.1350
	STD	0.0080	0.0080	0.0082	0.0070	0.0064	0.0043	0.0050
	Com. time	0.0068	0.0052	0.0060	0.0051	0.0004	0.0001	0.0003
Parkinson	RMSE mean	**0.0566**	**0.0566**	0.0608	0.0666	0.0786	0.1915	0.1844
	STD	0.0025	0.0025	0.0027	0.0027	0.0028	0.0005	0.0031
	Com. time	0.2296	0.2229	0.2971	0.2815	0.0178	0.0039	0.0101
Servo	RMSE mean	**0.1120**	**0.1120**	0.1231	0.1167	0.1577	0.1602	0.1197
	STD	0.0210	0.0210	0.0237	0.0164	0.0218	0.0038	0.0107
	Com. time	0.0062	0.0060	0.0062	0.0054	0.0014	0.0003	0.0051
Qsar Fish	RMSE mean	**0.0902**	0.0905	0.0917	0.0942	0.0943	0.0976	0.0973
	STD	0.0021	0.0021	0.0027	0.0020	0.0026	0.0010	0.0009
	Com. time	0.0249	0.0058	0.0201	0.0046	0.0005	0.0001	0.0003
Overall	RMSE mean	**0.0769**	0.0770	0.0807	0.0837	0.0917	0.1217	0.1023

Furthermore, a paired t-test was applied to validate the significance of the results in the testing, and the test results showed a statistically significant difference between the average RMSEs of the proposed method and each of the benchmark methods. This indicates that the proposed method achieved statistically significantly better performance than the other regression methods. However, t-test did not find evidence in favor of a significant difference between the test results of the Md-FKNNreg and Man-FKNNreg in all data sets. More information about the test results can be found in Publication **IV**.

5.5 Conclusion and Limitations

This chapter introduced a novel generalized regression approach based on the FKNN method. The generalization comes in that the Minkowski distance is used instead of Euclidean distance for the distance calculation. Using the Minkowski distance allows the new method to find an appropriate set of k nearest neighbors for the test sample because the Euclidean distance may not always be the optimum for every problem. The performance of the proposed method was examined using eight real-life data sets from different domains and benchmarked to the KNNreg, SVR, LASSO, and MLR models. In addition, the Euclidean distance- and Manhattan distance-based FKNNreg methods (referred to as Euc-FKNNreg and Man-FKNNreg) were also implemented, and the performances were compared.

The experimental results with the real-world data demonstrated that the proposed Md-FKNNreg method outperformed the baseline models and confirmed its effectiveness for various regression problems. Notably, the Md-FKNNreg method achieved the lowest overall average RMSE of 0.0769 in the testing in comparison to other regression models. Additionally, the Minkowski distance with $\mathcal{P} = 1$ appeared to be optimal in most cases for the Md-FKNNreg method, offering the best performance in the testing. In other words, the Man-FKNNreg model demonstrated outstanding performance for the regression at large, corroborating indications in the research by Aggarwal et al. (2001).

During the experiment, it was observed that the computational complexity of the proposed model was somewhat higher than the Man-FKNNreg, Euc-FKNNreg, and KNNreg methods. This can be expected because an extra computation part with the Minkowski distance is involved in the new algorithm. However, the results of the proposed method are encouraging, and the performance should be validated further by using large data sets (i.e., with different levels of complexities). In such cases, it would also be interesting to investigate the performance of the newly proposed method when it is incorporated with a feature selection or feature extraction method, which remains open for future research. Another exciting application of the Md-FKNNreg method could be an utilization of it in regression applications where the KNNreg method had been used (for example, see Yao and Ruzzo, 2006; Hu et al., 2014; Cai et al., 2020; Zhou et al., 2020).

6 Conclusion, Limitations, and Future Work

The objective of this dissertation was to develop enhanced FKNN methods to be applied in classification and regression problems. To achieve this, the effect of using class prototype mean vectors instead of individual neighbors in the learning part of the original FKNN algorithm was first examined. As a result, the novel MLPM-FKNN classifier (Publication **I**) was proposed. In particular, the influence of using a more general mean operator, the Power mean, instead of usual arithmetic mean for mean calculation was explored. As previously discussed in detail, the imbalance ratio of the classes has a significant influence on the performance of the classification algorithms, and in reality, classification problems are often imbalanced. Therefore, this dissertation primarily focused on several imbalanced data sets to study the capability of the new method for class imbalance problems. The observed results showed that the MLPM-FKNN classifier outperformed the KNN and FKNN methods, achieving the best accuracy in all data sets used. To further demonstrate the MLPM-FKNN algorithm's efficacy, it was successfully integrated with a (filter + wrapper) feature selection approach as a hybrid framework to predict S&P 500 intraday stock return (Publication **II**). However, even though the expected results were not found from this study in terms of the multi-class classification, it was observed that the MLPM-FKNN method could be valuable for wrapper feature selection approaches and produce improved results.

Additionally, this study has extended the MLPM-FKNN classifier based on the Bonferroni mean instead of Power mean in local mean computation and proposed the new BM-FKNN classifier (Publication **III**). The objective of this method was to demonstrate that Bonferroni mean-based class prototype vectors offer better learning than the Power mean and also improve classification accuracy. This method was then tested on one artificial data set generated in controlled settings and six real-world data sets compared to six competitive classifiers. The findings with both artificial and real-world data sets highlighted the capability and effectiveness of this method in terms of class imbalance problems. Furthermore, it is worth mentioning that the justification of the performance of classifiers was not based only on the accuracy measure but also on sensitivity and specificity values. From the theoretical and empirical perspectives, using these measures along with the accuracy (or error rate) is necessary because accuracy itself does not reveal the appropriateness of a particular model (which has already been discussed in Section 2.6.4). However, many studies (for example, see Gou et al., 2019; Pan et al., 2017; Biswas et al., 2018; Gou et al., 2014; Derrac et al., 2016, 2015) have not considered this factor.

In addition to the classification context, the rationale underlying the FKNN algorithm for regression problems was successfully generalized, and concurrently, a novel Md-FKNNreg method was proposed using the Minkowski distance instead of Euclidean distance for computing distances (Publication **IV**). This method was tested on several real-world data sets, and the results highlighted that it achieved a better performance than the several other regression methods.

Regarding the limitations, the shortcomings of each proposed method (and study) have already been discussed at the end of each related chapter. Overall, it should be highlighted that the computation complexity of the proposed methods seemed to be high in fitting to the data, but their execution on the test sets was not slow as compared to the applied benchmarks. Moreover, regardless of the limitations, this research paves a way to investigate new areas of research in FKNN classification from several different perspectives. From a methodological perspective, the findings of this study demonstrate the robustness and efficacy of the proposed approaches for class imbalance problems. In principle, any application that involves KNN and FKNN classifications can be reconsidered with the developed classifiers, particularly the BM-FKNN approach proposed in this study. For example, a diagnosis system presented in Chen et al. (2013) based on the FKNN classifier for detecting Parkinson's disease can be easily extended with the proposed BM-FKNN algorithm. Meanwhile, from a theoretical perspective, examining the impact of using different distance measures in the MLPM-FKNN (or BM-FKNN) algorithm on classification problems would be interesting. In this regard, it also remains an interesting open problem as to how the proposed classifiers perform when the "distance approximation" approach is adopted as in Maillo et al. (2020) instead of direct distance calculation, particularly in large data sets. Moreover, future work can also focus on examining new methods in type-2 fuzzy set settings. Finally, another direction for future research can also include testing the effect of integrating the Md-FKNNreg model with some other competitive methods, such as the SVM (Chen and Hao, 2018), in a hybrid framework.

References

Aggarwal, C. (2015). Data Classification. In: *Data Mining*. Springer, Cham.

Aggarwal, C.C., Hinneburg, A., and Keim, D.A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: *Database Theory — ICDT 2001*, pp. 420–434. Berlin: Springer.

Alcala-Fdez, J., et al. (2011). KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17, pp. 255–287.

Arif, M., Akram, M.U., and Minhas, F.A. (2010). Pruned fuzzy k-nearest neighbor classifier for beat classification. *Journal of Biomedical Science and Engineering*, 3, pp. 380–3899.

Aristotle (4th century BC). *Politics*.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4, pp. 40–79.

Bailey, T. and Jani, A. (1978). A note on distance-weighted k-nearest neighbor rules. *Transactions on Systems, Man, and Cybernetics*, 8, pp. 311–313.

Barak, S., Arjmand, A., and Ortobelli, S. (2017). Fusion of multiple diverse predictors in stock market. *Information Fusion*, 36, pp. 90–102.

Beliakov, G., Pradera, A., and Calvo, T. (2007). *Aggregation Functions: A guide for Practitioners*. Berlin, Heidelberg: Springer-Verlag.

Beliakov, G., Sola, B.H., and Sánchez, C. (2016). Classical Averaging Functions. In: *A practical guide to averaging functions*, vol. 329, pp. 55–99. Cham: Springer.

Benedetti, J.K. (1977). On the Nonparametric Estimation of Regression Functions. *Journal of the Royal Statistical Society*, 39, pp. 248–253.

Bergamasco, L.C.C. and Nunes, F.L.S. (2019). Intelligent retrieval and classification in three-dimensional biomedical images-A systematic mapping. *Computer Science Review*, 31, pp. 19–38.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10), pp. 281–305.

Berrar, D. (2019). Cross-Validation. In: Ranganathan, S., Gribskov, M., Nakai, K., and Schönbach, C., eds, *Encyclopedia of Bioinformatics and Computational Biology*, pp. 542–545. Oxford: Academic Press.

Bian, H. and Mazlack, L. (2003). Fuzzy-rough nearest-neighbor classification approach. In: *22nd International Conference of the North American Fuzzy Information Processing Society, NAFIPS 2003*, pp. 500–505.

Biau, G., Devroye, L., Dujmović, V., and Krzyżak, A. (2012). An affine invariant k-nearest neighbor regression estimate. *Journal of Multivariate Analysis*, 112, pp. 24–34.

Bishop, C.M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). Berlin, Heidelberg: Springer-Verlag.

Bisoi, R., Dash, P.K., and Parida, A.K. (2019). Hybrid Variational Mode Decomposition and evolutionary robust kernel extreme learning machine for stock price and movement prediction on daily basis. *Applied Soft Computing*, 74, pp. 652–676.

Biswas, N., Chakraborty, S., Mullick, S.S., and Das, S. (2018). A parameter independent fuzzy weighted k-Nearest neighbor classifier. *Pattern Recognition Letters*, 101, pp. 80–87. doi:https://doi.org/10.1016/j.patrec.2017.11.003.

Bonferroni, C. (1950). Sulle medie multiple di potenze. *Bolletino Mathematica Italiana*, 5, pp. 267–270.

Borovicka, T., Jirina, M.J., Kordik, P., and Jirina, M. (2012). Selecting representatives data sets. In: *Advances in Data Mining Knowledge Discovery and Applications*, pp. 43–70. Croatia: Rijeka.

Bramer, M. (2016). Estimating the Predictive Accuracy of a Classifier. In: *Principles of Data Mining*, pp. 79–92. London: Springer.

Bugata, P. and Drotár, P. (2019). Weighted nearest neighbors feature selection. *Knowledge-Based Systems*, 163, pp. 749–761.

Bullen, P.S. (2003). The Power Means. In: *Handbook of Means and Their Inequalities. Mathematics and Its Applications*, vol. 560, pp. 175–265. Dordrecht: Springer.

Cabello, D., et al. (1991). Fuzzy k-nearest neighbor classifiers for ventricular arrhytmia detection. *International Journal of Biomedical Computing*, 27, pp. 77–93.

Cai, L., et al. (2020). A Sample-Rebalanced Outlier-Rejected k-Nearest Neighbor Regression Model for Short-Term Traffic Flow Forecasting. *IEEE Access*, 8, pp. 22686–22696.

Cao, H., Lin, T., and Li, Y. (2019). Stock price pattern prediction based on complex network and machine learning. *Complexity*.

Chai, J., Liu, H., Chen, B., and Bao, Z. (2010). Large margin nearest local mean classifier. *Signal Processing*, 90, pp. 236–248.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40, pp. 16–28.

Chang, H., Yeung, D.Y., and Cheung, W.K. (2006). Relaxational metric adaptation and its application to semi-supervised clustering and content-based image retrieval. *Pattern Recognition*, 39, pp. 1905–1917.

Chen, H.L., et al. (2011). An adaptive fuzzy k-nearest neighbor method based on parallel particle swarm optimization for bankruptcy prediction. *Lecture Notes in Computer Science 6634 LNAI (PART 1)*, pp. 249–264.

Chen, H.L., et al. (2013). An efficient diagnosis system for detection of Parkinson's disease using fuzzy k-nearest neighbor approach. *Expert Systems with Applications*, 40(1), pp. 263–271.

Chen, Y. and Hao, Y. (2018). Integrating principle component analysis and weighted support vector machine for stock trading signals prediction. *Neurocomputing*, 321, pp. 381–402. doi:https://doi.org/10.1016/j.neucom.2018.08.077.

Cheng, P.E. (1984). Strong Consistency of Nearest Neighbor Regression Function Estimators. *Journal of Multivariate Analysis*, 15, pp. 63–72.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Maching Learning*, 20, pp. 273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, pp. 21–27.

Derrac, J., Chiclana, F., Garcia, S., and Herrera, F. (2015). An interval valued k-nearest neighbors classifiers. In: *Proceeding of the International Joint Conference IFSA-EUSFLAT-2015*.

Derrac, J., Chiclana, F., GarcÃa, S., and Herrera, F. (2016). Evolutionary fuzzy k-nearest neighbors algorithm using interval-valued fuzzy sets. *Information Sciences*, 329, pp. 144–163.

Derrac, J., García, S., and Herrera, F. (2014). Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. *Information Sciences*, 260, pp. 98–119. doi:https://doi.org/10.1016/j.ins.2013.10.038.

Dettmann, E., Becker, C., and Schmeiser, C. (2011). Distance functions for matching in sl samples. *Computational Statistics & Data Analysis*, 55, pp. 1942–1960.

Dheeru, D. and Taniskidou, E.K. (2017). *UCI machine learning repository.* url: `http://archive.ics.uci.edu/ml`.

Drucker, H., et al. (1997). Support vector regression machines. *Neural Information Processing Systems*, 9, pp. 155–161.

Durbin, M., Wonders, M.A., Flaska, M., and Lintereur, A.T. (2021). K-Nearest Neighbors regression for the discrimination of gamma rays and neutrons in organic scintillators. *Nuclear Instruments and Methods in Physics Research A*, 987, p. 164826.

Fabio, B.M., Jose, M.M., and Janusz, K. (2016). Bonferroni means with distance measures and the adequacy coefficient in entrepreneurial group theory. *Knowledge-Based Systems*, 111, pp. 217–227.

Ferri, C., Hernández-Orallo, J., and Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1), pp. 27–38.

Fix, E. and Hodges, J.L. (1951). Discriminatory analysis. Nonparametric discrimination: consistency properties. USAF School of Aviation Medicine, Randolph Field, Texas: Report Number 4, Project Number 21-49-004.

Flach, P. (2012). *Machine learning: The art and science of algorithms that make sense of data*. Cambridge: Cambridge University Press.

Galton, F. (1949). Vox populi. *Nature*, 75, pp. 450–451.

García, S., Luengo, J., and Herrera, F. (2016). Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowledge-Based Systems*, 98, pp. 1–29. doi:https://doi.org/10.1016/j.knosys.2015.12.006.

García, V., Mollineda, R.A., and Sánchez, J.S. (2010). Theoretical Analysis of a Performance Measure for Imbalanced Data. In: *2010 20th International Conference on Pattern Recognition*, pp. 617–620.

García, V., et al. (2006). Combined Effects of Class Imbalance and Class Overlap on Instance-Based Classification. In: Corchado, E., Yin, H., Botti, V., and Fyfe, C., eds, *Intelligent Data Engineering and Automated Learning – IDEAL 2006*, pp. 371–378. Berlin, Heidelberg: Springer.

Ghosh, A.K. (2006). On optimum choice of k in nearest neighbor classification. *Computational Statistics & Data Analysis*, 50, pp. 3113–3123.

Gou, J., et al. (2014). Improved pseudo nearest neighbor classification. *Knowledge-Based Systems*, 70, pp. 361–375. doi:https://doi.org/10.1016/j.knosys.2014.07.020.

Gou, J., et al. (2019). A generalized mean distance-based k-nearest neighbor classifier. *Expert Systems with Applications*, 115, pp. 356–372. doi: https://doi.org/10.1016/j.eswa.2018.08.021.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal Of Machine Learning Research*, 3, pp. 1157–1182.

Györfi, L., Kohler, M., Krzyzak, A., and Walk, H. (2002). *A distribution free theory of nonparametric regression*. New York: Springer-Verlag.

Hadjitodorov, S.T. (1995). An Intuitionistic fuzzy sets application to the k-NN method. *Notes on Intuitionistic Fuzzy Sets*, 1, pp. 66–69.

Hadjitodorov, S.T. (2001). An Intuitionistic fuzzy version of the nearest prototype classification method, based on a moving-of-pattern procedure. *International Journal of General Systems*, 30(2), pp. 155–165.

Hait, S.R., et al. (2022). The Bonferroni mean-type pre-aggregation operators construction and generalization: Application to edge detection. *Information Fusion*, 80, pp. 226–240. doi:https://doi.org/10.1016/j.inffus.2021.11.002.

Han, H. and Mao, B. (2010). Fuzzy-rough k-nearest neighbor algorithm for imbalanced data sets learning. In: *Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp. 1286–1290.

Han, J., Kamber, M., and Pei, J. (2012). *Data Mining (Third Edition)*. Boston: Morgan Kaufmann.

Harshita, P. and Singh, T. (2017). Classification of imbalanced data using a modified fuzzy-neighbor weighted approach. *International Journal of Intelligent Engineering and Systems*, 10(1), pp. 56–64.

Heumann, C. and Shalabh, M.S. (2016). Hypothesis Testing. In: *Introduction to Statistics and Data Analysis*. Springer, Cham.

Hu, C., et al. (2014). Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. *Applied Energy*, 129, pp. 49–55.

Hu, X. and Xie, C. (2005). Improving fuzzy k-nn by using genetic algorithm. *Journal of Computational Information Systems*, 1, pp. 203–213.

Huang, Y. and Li, Y. (2004). Prediction of protein subcellular locations using fuzzy k-nn method. *Bioinformatics*, 20(1), pp. 21–28.

Hurwitz, J. and Kirsch, D. (2018). *Machine Learning For Dummies, IBM Limited Edition*. John Wiley & Sons.

Igual, L. and Seguí, S. (2017). *Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications*. Cham: Springer International Publishing.

Jensen, R. and Cornelis, C. (2011). Fuzzy-rough nearest neighbour classification and prediction. *Theoretical Computer Science*, 412, pp. 5871–5884.

Jiang, X., Wei, R., Zhang, T., and Gu, Q. (2008). Using the concept of Chou's pseudo amino acid composition to predict apoptosis proteins subcellular location: an approach by approximate entropy. *Protein & Peptide Letters*, 15, pp. 392–396.

Johnson, J.M. and Khoshgoftaar, T.M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, 6(27).

Jung, A. (2022). *Components of ML*, pp. 19–56. Singapore: Springer Nature Singapore.

Kassani, P.H., Teoh, A.B.J., and Kim, E. (2017). Evolutionary-modified fuzzy nearest-neighbor rule for pattern classification. *Expert Systems with Applications*, 88, pp. 258–269.

Keller, J.M., Gray, M.R., and Givens, J.A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems*, 15, pp. 580–585.

Keogh, E. (2011). Instance-Based Learning. In: Sammut, C. and Webb, G.I., eds, *Encyclopedia of Machine Learning*. Boston, MA: Springer.

Khushaba, R.N., Al-Ani, A., and Al-Jumaily, A. (2011). Feature subset selection using differential evolution and a statistical repair mechanism. *Expert System with Applications*, 38, pp. 11515–11526.

Koloseni, D., Lampinen, J., and Luukka, P. (2012). Optimized distance metrics for differential evolution based nearest prototype classifier. *Expert Systems with Applications*, 39(12), pp. 10564–10570.

Koloseni, D., Lampinen, J., and Luukka, P. (2013). Differential evolution based nearest prototype classifier with optimized distance measures for the features in the data sets. *Expert Systems with Applications*, 40(10), pp. 4075–4082.

Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1), pp. 25–36.

Kramer, O. (2011). Dimensionality reduction by unsupervised K-nearest neighbor regression. In: *Proceedings of the 10th International Conference on Machine Learning and Applications, ICMLA*, pp. 275–278.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5, pp. 227–232.

Kuang, L., et al. (2019). Predicting duration of traffic accidents based on cost-sensitive Bayesian network and weighted K-nearest neighbor. *Journal of Intelligent Transportation Systems*, 23(2), pp. 161–174.

Kubat, M. (2017). *An Introduction to Machine Learning*. Cham: Springer International Publishing.

Kubat, M. and Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186. Morgan Kaufmann.

Kumbure, M.M., Lohrmann, C., Luukka, P., and Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197, p. 116659.

Kuncheva, L.I. (1995). An intuitionistic fuzzy k-nearest neighbors rule. *Notes on Intuitionistic Fuzzy Sets*, 1, pp. 56–60.

Kurama, O., Luukka, P., and Collan, M. (2016). A similarity classifier with Bonferroni mean operators. *Advances in Fuzzy Systems*, (2016), 7173054.

Kurz-Kim, J.R. and Loretan, M. (2014). On the properties of the coefficient of determination in regression models with infinite variance variables. *Journal of Economics*, 181, pp. 15–24.

Lee, S.B., Gui, X., Manquen, M., and Hamilton, E. (2019). Use of Training, Validation, and Test Sets for Developing Automated Classifiers in Quantitative Ethnography. In: Eagan, B., Misfeldt, M., and Siebert-Evenstone, A., eds, *Advances in Quantitative Ethnography. ICQE 2019. Communications in Computer and Information Science*, vol. 1112. Cham.: Springer.

Lei, Y. and Zuo, M.J. (2009). Gear crack level identification based on weighted K nearest neighbor classification algorithm. *Mechanical Systems and Signal Processing*, 23(5), pp. 1535–1547. doi:https://doi.org/10.1016/j.ymssp.2009.01.009.

Lever, J., Krzywinski, M., and Altman, N. (2016). Model selection and overfitting. *Nature Methods*, 13, pp. 703–704. doi:https://doi.org/10.1038/nmeth.3968.

Lewis, D.D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In: *Machine Learning: ECML-98*, pp. 4–15. Berlin, Heidelberg: Springer Berlin Heidelberg.

Li, D., et al. (2022). Continual learning classification method with the weighted k-nearest neighbor rule for time-varying data space based on the artificial immune system. *Knowledge-Based Systems*, 240, p. 108145.

Li, J., et al. (2017). Feature Selection: A Data Perspective. *ACM Computing Surveys (CSUR)*, 50, p. 94.

Liao, T.W. and Li, D. (1997). Two manufacturing applications of the fuzzy k-nn algorithm. *Fuzzy Sets and Systems*, 92, pp. 289–303.

Liu, H. and Motoda, H. (1998). Less Is More. In: Liu, H. and Motoda, H., eds, *Feature Extraction, Construction and Selection*. Boston, MA: The Springer International Series in Engineering and Computer Science, Springer.

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 4, pp. 491–502.

Liu, W. and Chawla, S. (2011). Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets. In: Huang, J.Z., Cao, L., and Srivastava, J., eds, *Advances in Knowledge Discovery and Data Mining*, pp. 345–356. Berlin, Heidelberg: Springer.

Liu, Z., Quan, P., and Jean, D. (2013). A new belief-based K-nearest neighbor classification method. *Pattern Recognition*, 46, pp. 834–844.

Liua, P., Gaoa, H., and Ma, J. (2019). Novel green supplier selection method by combining quality function deployment with partitioned Bonferroni mean operator in interval type-2 fuzzy environment. *Information Sciences*, 490, pp. 292–316.

Lohrmann, C. and Luukka, P. (2019). Classification of intraday S&P500 returns with a Random Forest. *International Journal of Forecasting*, 35, pp. 390–407.

Lu, B., Charlton, M., Brunsdon, C., and Harris, P. (2015). The minkowski approach for choosing the distance metric in geographically weighted regression. *International Journal of Geographcial Information Science*.

Luukka, P., Saastamoinen, K., and Kononen, V. (2001). A classifier based on the maximal fuzzy similarity in the generalized Lukasiewicz-structure. In: *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, vol. 1, pp. 195–198.

Luukka, P. (2011). Feature selection using fuzzy entropy measures with similarity classifier. *Expert Systems with Applications*, 38(4), pp. 4600–4607. doi: https://doi.org/10.1016/j.eswa.2010.09.133.

Luukka, P. and Kurama, O. (2013). Similarity classifier with ordered weighted averaging operators. *Expert Systems with Applications*, 40, pp. 995–1002.

Maillo, J., et al. (2020). Fast and scalable approaches to accelerate the fuzzy k-nearest neighbors classifier for Big data. *IEEE Transactions on Fuzzy Systems*, 28(5), pp. 874–886.

Mathisen, B.M., Aamodt, A., Bach, K., and Langseth, H. (2019). Learning similarity measures from data. *Progress in Artificial Intelligence*, 9(2), pp. 129–143.

Mesiar, R., Štěpnička, M., and Šostak, A. (2013). Fuzzy sets: Theory and applications (FSTA 2012). *Fuzzy Sets and Systems*, 232, pp. 1–2. doi: https://doi.org/10.1016/j.fss.2013.08.013.

Mitani, Y. and Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27, pp. 1151–1159.

Montgomery, D.C., Peck, E.A., and Vining, G.G. (2012). *Introduction to linear regression analysis*. John Wiley & Sons.

Namata, G., Sen, P., Bilgic, M., and Getoor, L. (2009). Chapter 3 - Collective classification for text classification. In: Srivastava, N.A. and Sahami, M., eds, *Text Mining: Classification, Clustering, and Applications*, pp. 51–66. Chapman and Hall, USA.

Nguyen, B., Morell, C., and Baets, B.D. (2016). Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing*, 214, pp. 805–814.

Nikoo, M.R., Kerachian, R., and Alizadeh, M.R. (2018). A fuzzy KNN-based model for significant wave height prediction in large lakes. *Oceanologia*, 60, pp. 153–168.

Pan, Z., Wang, Y., and Ku, W. (2017). A new k-harmonic nearest neighbor classifier based on the multi-local means. *Expert Systems with Applications*, 67, pp. 115–125.

Park, H.J., Um, J.G., Woo, I., and Kim, J.W. (2012). Application of fuzzy set theory to evaluate the probability of failure in rock slopes. *Engineering Geology*, 125, pp. 92–101. doi:https://doi.org/10.1016/j.enggeo.2011.11.008.

Patel, H. and Thakur, G.S. (2019). An improved fuzzy k-nearest neighbor algorithm for imbalanced data using adaptive approach. *IETE Journal of Research*, 65(6), pp. 780–789.

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4), pp. 2162–2172.

Pham, H. (2019). A new criterion for model selection. *Mathematics*, 7, p. 1215.

Price, K., Storn, R.M., and Lampinen, J.A. (2005). Differential evolution - a practical approach to global optimization. Berlin Heidelberg: Springer Vieweg.

Qin, Y., Qi, Q., Scott, P.J., and Jiang, X. (2020). An additive manufacturing process selection approach based on fuzzy Archimedean weighted power Bonferroni aggregation operators. *Robotics and Computer-Integrated Manufacturing*, 64, p. 101926.

Ramentol, E., et al. (2015). IFROWANN: Imbalanced Fuzzy-Rough Ordered Weighted Average Nearest Neighbor Classification. *IEEE Transactions on Fuzzy Systems*, 23(5), pp. 1622–1637.

Rastin, N., Taheri, M., and Jahromi, M.Z. (2021). A stacking weighted k-Nearest neighbour with thresholding. *Information Sciences*, 571, pp. 605–622. doi: https://doi.org/10.1016/j.ins.2021.05.030.

Rhee, F.C. and Hwang, H. (2003). An interval type-2 fuzzy k-nearest neighbor. In: *Proceeding of the 12th IEEE International Conference on Fuzzy Systems-2003. FUZZ'03*, pp. 802–807.

Royall, R. (1966). *A class of nonparametric estimators of a smooth regression function*. Ph.D. thesis. Stanford University, Stanford, CA.

Runkler, T. (2016). Regression. In: *Data Analytics*. Wiesbaden: Springer Vieweg.

Sarkar, M. (2007). Fuzzy-rough nearest neighbor algorithms in classification. *Fuzzy Sets and Systems*, 158(19), pp. 2134–2152. ISSN 0165-0114. Theme: Data Analysis.

Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(160).

Shalabi, L.A., Shaaban, Z., and Kasasbeh, A. (2006). Data Mining: A Preprocessing Engine. *Journal of Computer Science*, 2(9), pp. 735–739.

Shirkhorshidi, A.S., Aghabozorgi, S., and Wah, T.Y. (2015). A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. *PLOS ONE*, 10(12), pp. 1–20.

Song, Y., Liang, J., Lu, J., and Zhao, X. (2017). An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing*, 251, pp. 26–34.

Stone, C.J. (1977). Consistent Nonparametric Regression. *Annals of Statistics*, 5, pp. 595–645.

Sumet, M., Xiangjun, S., Jiangping, G., and Dejiao, N. (2018). A new nearest centroid neighbor classifier based on k local means using harmonic mean distance. *Information*, 9, p. 234.

Sun, B. and Chen, H. (2021). A Survey of Nearest Neighbor Algorithms for Solving the Class Imbalanced Problem. *Wireless Communications and Mobile Computing*, 2021. doi:https://doi.org/10.1155/2021/5520990.

Sun, M. and Liu, J. (2013). Normalized geometric Bonferroni operators of hesitant fuzzy sets and their application in multiple attribute decision making. *Journal of Information and Computational Science*, 10, pp. 2815–2822.

Tan, S. (2005). Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4), pp. 667–671.

Teixeira, L.A. and de Oliveira, A.L.I. (2010). A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, 37(10), pp. 6885–6890. ISSN 0957-4174.

Tharwat, A. (2020). Classification assessment methods. *Applied Computing and Informatics*, 17. doi:https://doi.org/10.1016/j.aci.2018.08.003.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistics Society: Series B Methodololgy*, 58, pp. 267–288.

Torra, V. (2016). Aggregation Operators. In: Seising, R., Trillas, E., Moraga, C., and Termini, S., eds, *On Fuzziness. Studies in Fuzziness and Soft Computing*, vol. 299, pp. 691–695. Berlin, Heidelberg: Springer.

Triguero, I., Derrac, J., Garcia, S., and Herrera, F. (2012). A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1), pp. 86–100.

Trousse, B. and Tanasa, D. (2004). Advanced Data Preprocessing for Intersites Web Usage Mining. *IEEE Intelligent Systems*, 19(02), pp. 59–65. doi: 10.1109/MIS.2004.1274912.

Tsai, C.F. and Hsiao, Y.C. (2011). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, 50, pp. 258–269.

Turner, T. (1977). *Exploratory data analysis*. Addison Wesley, Reading.

Wang, J.L. (2009). A supply chain application of fuzzy set theory to inventory control models–DRP system analysis. *Expert Systems with Applications*, 36(5), pp. 9229–9239. doi:https://doi.org/10.1016/j.eswa.2008.12.047.

Wei, G., Zhao, X., Lin, R., and Wang, H. (2013). Uncertain linguistic Bonferroni mean operators and their application to multiple attribute decision making. *Applied Mathematical Modelling*, 37, pp. 5277–5285.

Wendler, T. and Gröttrup, S. (2016). Classification Models. In: *Data Mining with SPSS Modeler*. Cham: Springer.

Winters-Miner, L.A., et al. (2015). Chapter 15 - Prediction in Medicine âFIXME“ The Data Mining Algorithms of Predictive Analytics. In: Winters-Miner, L.A., et al., eds, *Practical Predictive Analytics and Decisioning Systems for Medicine*, pp. 239–259. Academic Press.

Witten, I.H., Frank, E., and Hall, M.A. (2011). Chapter 3 - Output: Knowledge Representation. In: Witten, I.H., Frank, E., and Hall, M.A., eds, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edn, pp. 61–83. Boston: Morgan Kaufmann.

Wong, B.K. and Lai, V.S. (2011). A survey of the application of fuzzy set theory in production and operations management: 1998–2009. *International Journal of Production Economics*, 129(1), pp. 157–168. doi:https://doi.org/10.1016/j.ijpe.2010.09.013.

Wu, S., et al. (2021). Evolving fuzzy k-nearest neighbors using an enhanced sine cosine algorithm: Case study of lupus nephritis. *Computers in Biology and Medicine*, 135, p. 104582. doi:https://doi.org/10.1016/j.compbiomed.2021.104582.

Xia, M., Xu, Z., and Zhu, B. (2013). Geometric Bonferroni means with their application in multi-criteria decision making. *Knowledge-Based Systems*, 40, pp. 88–100.

Xu, Z. and Yager, R.R. (2011). Intuitionistic Fuzzy Bonferroni Means. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(2), pp. 568–578.

Yager, R.R. (2009a). On generalized Bonferroni mean operators for multi-criteria aggregation. *International Journal of Approximate Reasoning*, 50, pp. 1279–1286.

Yager, R.R. (2009b). On the dispersion measure of OWA operators. *Information Sciences*, 179, pp. 3908–3919.

Yang, F., Chen, Z., Li, J., and Tang, L. (2019). A novel hybrid stock selection method with stock prediction. *Applied Soft Computing*, 80, pp. 820–831. ISSN 1568-4946, doi: https://doi.org/10.1016/j.asoc.2019.03.028.

Yang, M.S. and Chen, C.H. (1998). On the edited fuzzy K-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3), pp. 461–466.

Yao, Z. and Ruzzo, W. (2006). A regression-based k nearest neighbor algorithm for gene function prediction from heterogeneous data. *BMC Bioinformatics*, 7, p. S11.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, 8, pp. 338–353.

Zeraatkar, S. and Afsari, F. (2021). Interval-valued fuzzy and intuitionistic fuzzy-KNN for imbalanced data classification. *Expert Systems with Applications*, 184, p. 115510.

Zhai, J.H., Li, N., and Zhai, M.Y. (2011). The condensed fuzzy k-nearest neighbor rule based on sample fuzzy entropy. In: *2011 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 282–286.

Zhang, N., Lin, A., and Shang, P. (2017a). Multidimensional k-nearest neighbor model based on EEMD for financial time series forecasting. *Physica A: Statistical Mechanics and its Applications*, 477, pp. 161–173.

Zhang, X., et al. (2014). A causal feature selection algorithm for stock prediction modeling. *Neurocomputing*, 142, pp. 48–59.

Zhang, X., et al. (2017b). KRNN: k Rare-class Nearest Neighbour classification. *Pattern Recognition*, 62, pp. 33–44. doi:https://doi.org/10.1016/j.patcog.2016.08.023.

Zhou, Y., Huang, M., and Pecht, M. (2020). Remaining useful life estimation of lithium-ion cells based on k- nearest neighbor regression with differential evolution optimization. *Journal of Cleaner Production*, 249, p. 119409.

Publication I

Mailagaha-Kumbure, M., Luukka, P., and Collan, M.
An enhancement of fuzzy k-nearest neighbor classifier using multi-local power means

Reprinted with permission from
Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)
Vol. 132, pp. 83–90, 2019.

ATLANTIS PRESS

Atlantis Studies in Uncertainty Modelling, volume 1

11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)

An Enhancement of Fuzzy K-Nearest Neighbor Classifier Using Multi-Local Power Means

Mahinda Mailagaha Kumbure[a] **and Pasi Luukka**[a] **and Mikael Collan**[a]
[a]School of Business and Management, LUT University, Skinnarilankatu 34, 53850
Lappeenranta, Finland, mahinda.mailagaha.kumbure@student.lut.fi; pasi.luukka@lut.fi; mikael.collan@lut.fi

Abstract

This study introduces a new method to the family of fuzzy k-nearest neighbor (FKNN) classifiers that is based on the use of power means in the calculation of multi-local means that are used in classification of samples. The proposed new classifier is called multi-local power means fuzzy k-nearest neighbor classifier (MLPM-FKNN). The proposed method can be adapted to the context (of different data sets), due to the power mean being parametric and thus allowing for testing to find the parameter value that can be optimized for the classification accuracy. Furthermore, we can find optimal value for the number of local observations used in calculation of the multi-local mean. The proposed method is usable for example in situations, where class distribution is significantly different and there is only few observations in some classes. The performance of the MLPM-FKNN classifier is studied by testing it with four datasets. The performance is benchmarked against that of the original k-nearest neighbor and the fuzzy k-nearest neighbor classifiers. We find that MLPM-FKNN classifier is able to reach a statistically significantly higher classification accuracy than the benchmarks used and has reasonable performance metrics.

Keywords: Accuracy, Classification, Fuzzy k-nearest neighbor, Performance, Power mean.

1 Introduction

The ability to classify samples to classes is a key property in a variety of scientific and engineering systems that range throughout human activities. Methods used in classification are typically divided into either supervised classification methods (with classes predefined) and unsupervised classification methods (with classes not pre-defined) [17]. In this research we concentrate on the k-nearest neighbor algorithm (KNN) [3] and the family of classification algorithms built around it. The KNN is a well-known supervised classification method that has been applied in many fields and is among the more popular techniques used in classification. The KNN is based on calculating the distance between already classified (labeled) samples and the sample to be classified (unlabeled) in determining the class membership [4]. Many enhancements to the original algorithm have been proposed, see, e.g., [2, 11].

One of the variants of the original KNN is the fuzzy k-nearest neighbor (FKNN) algorithm that was introduced by Keller in 1985 [8]. FKNN models the uncertainty in the data by introducing membership degrees to classes. In this research we develop the FKNN classifier further by using multi-local mean vectors (originally introduced by Pan et al. [11] to k-harmonic mean nearest neighbor classifier) to represent the known classes. These multi-local means are further generalized by using power mean and resulting mean vectors are used to calculate the distance of the unlabeled sample from the classes. Examples on such a prototype selection that was used to obtained better accuracy in the nearest classification have been reported in [14, 13, 9]. Moreover, since power mean is used in the calculation of the class representative mean vectors, we need to find what kind of mean is suitable for the particular data set by finding proper parameter value for power mean. By varying the mean controlling parameter in the power mean allows one to find the parameter value (and the mean) that can enhance the classification accuracy since suitable mean value is data dependent. We observe that by adjusting the parameter of the power mean one can arrive at several well-known means like the arithmetic, the geometric, and the harmonic means. The classification is done in

a way that an un-labeled sample is assigned membership in the class that has the representative multi-local power mean vector that has the highest membership degree to it.

The performance of the proposed method is studied with four data sets that include binary and multi-class data and compared with the performance of the classical KNN and the FKNN. We also test whether there is a statistically significant difference between the classification results from the MLPM-FKNN and the two benchmark classifiers. In addition to accuracy also performance measures, such as sensitivity and specificity are reported.

The rest of this paper is structured into five sections. Section 2 discusses the theoretical points of departure, the KNN, the FKNN, and the power mean, which all underlie this research study. The proposed MLPM-FKNN approach is described in section 3. Section 4 introduces the data and then goes through the testing of the proposed method and section 5 presents the results obtained with a comprehensive discussion. Section 6 shortly summarizes the findings and some conclusions are drawn.

2 Preliminaries

The FKNN and KNN classifiers, and the power mean are briefly described in this section.

2.1 The k-nearest neighbor and the fuzzy k-nearest neighbor classifiers

The k-nearest neighbor classification is one of the most well-known data mining techniques and has been widely applied for data classifications in many real-world applications. The KNN algorithm starts with the estimation of similarity (distance) of a new query sample (sample to be classified) with the samples in the training set (already classified). A set of k nearest neighbors for the query sample are identified, regardless of the class they belong to. Each neighbor "casts a vote" about to which class the query sample should belong. In the end of the process, the query sample is assigned to a class by way of counting the votes - the query sample is assigned to the class that has the most votes (from the k neighbors). To give a simple example, if $k = 1$, then the query sample is classified to the class of its closest neighbor.

A formal definition of the above procedure can be given as follows: Suppose a training set X shaped by N samples as $X = x_1, x_2, ..x_N$ and C classes. Each sample $x_j = x_j^1, x_j^2, .., x_j^S, x_t^S$ is described by S input variables and an output class variable t ($t \in C$). The nearest neighbor classifier finds the k nearest neighbors in X for a new query sample Q by using a distance function (Euclidean distance is widely used). Then, the class label for Q is assigned by taking into account the class labels t of the k nearest neighbors [4].

Studies have identified and addressed several weaknesses with the original KNN model. One identified problem is that the KNN model considers each of the k nearest neighbors to have the same importance in the classification [12]. Another observed issue with the KNN model is that once a sample is assigned to a particular class, the strength of the membership in the class is not indicated [8]. To remedy these issues Fuzzy k-nearest neighbor (FKNN) model has been introduced as an extension of the KNN [8]. Compared to the KNN, the range of k in FKNN is often wider and the classification results depend on the membership degree value of the unlabeled samples in the labeled classes in FKNN [16].

In the FKNN, for the query sample a degree of membership is assigned for each class and the decision is based on the highest membership degree [8]. In the membership function membership degrees are weighed by the inverse of distance between query sample and k nearest neighbors. In addition to this a fuzzy strength parameter m is used. The allocated degree of memberships of the query sample Q in the classes represented by the k nearest neighbors is measured as:

$$u_i(Q) = \frac{\sum_{j=1}^{K} u_{ij}(1/\left\|Q - x_j\right\|^{2/(m-1)})}{\sum_{j=1}^{K}(1/\left\|Q - x_j\right\|^{2/(m-1)})} \quad (1)$$

where, u_{ij} is the membership of the j^{th} sample in the i^{th} class in the training set, and $m > 1$ is the fuzzy strength parameter that influences to the membership degree. Often used value is $m = 2$. To define u_{ij}, there are two main approaches, one is through the crisp membership and other is through the fuzzy membership [2]. One way to define the fuzzy membership degree is introduced in [8], where k nearest neighbors are found for each training sample (x_j) and, the degree of membership of x_j in each represented class is computed as:

$$u_{ij}(x_j) = \begin{cases} 0.51 + (n_j/K) * 0.49, & \textbf{if } j = i \\ (n_j/K) * 0.49, & \textbf{if } j \neq i \end{cases} \quad (2)$$

where, n_j indicates the number of neighbors observed to belong in the j^{th} class. Once all the degrees of memberships are calculated for the query sample, the class that has the highest degree of membership is assigned to the query sample.

2.2 Power mean used in local mean computation

The power mean, also called generalized mean, is a function of a family of means. If $x_1, x_2, ..., x_n$ is a set of real numbers, and p is a parameter ($\in \Re$) then power mean (M_p) can be defined as follows.

$$M_p = \begin{cases} \prod_{i=1}^{n} x_i^{1/n}, & \textbf{if } p = 0 \\ (\frac{1}{n}\sum_{i=1}^{n} x_i^p)^{1/p}, & \textbf{if } p \neq 0 \end{cases} \quad (3)$$

The formula is defined generally for the two cases, one is when $p = 0$ (it is equal to geometric mean[1]) and other is $p \neq 0$. With the power mean we can get other well-known means, when parameter p is adjusted. These special cases can be defined as harmonic mean ($p = -1$), geometric mean ($p = 0$), arithmetic mean ($p = 1$), quadratic mean ($p = 2$), cubic mean ($p = 3$), and so on.

3 Fuzzy k-nearest neighbor classifier based on the multi-local power mean vectors

As in the FKNN method, the MLPM-FKNN determines first the distance from the labeled samples $\{X_i, C_i\}_{i=1}^{N_{tr}}$ to the query sample Q and the k nearest neighbors $nn^k(Q)$ are identified (where $C_i \in (\omega_1, \omega_2, .., \omega_T$: class labels)). Each sample X_i $(X_i^1, X_i^2, .., X_i^n)$ contains n attributes. The k nearest neighbors are grouped, based on the class they represent into sub-samples. Then power mean vectors for the sub-samples representing each class are computed, these are called "multi-local power mean vectors". That is, if the $nn^k(Q)$ is $\{X_i, C_i\}_{i=1}^{k}$ and $C_i \in (\omega_1, \omega_2, ..\omega_T)$, then the local power mean vectors for the corresponding classes are $\{M_{p,r}, \omega_r\}_{r=1}^{t}$, $1 \leq t \leq T$. This also means that the number of local mean vectors depends on the number of classes that can be found from among the k nearest neighbors. Then the Euclidean distance between the local power mean vectors, representing the corresponding classes, and the the query sample is calculated (for each local power mean vector). The distance $d_{EUC}\ (Q, \{M_{p,r}\}_{r=1}^{t})$ between each local power mean vector and the query sample is weighted by the degree of membership in each represented class $\{M_{p,r}, \omega_r\}_{r=1}^{t}$. Finally, the query sample is assigned membership in the class ω^*. Algorithm 1 summarizes the MLPM-FKNN method in pseudo-code.

One motivation for building the MLPM-FKNN is that it is well-known [10] that the KNN method encounters problems, when there is a clear imbalance in the quantity of labeled samples in the represented classes in the "proximity" of the query sample. The class with more samples tends to dominate the prediction of the query sample. This undesirable dominance can be guided by using the local mean vectors instead of using the k nearest neighbors directly. In the proposed method, the local sub-samples are used to create a local mean vector for all classes that are represented within the k nearest neighbors and thus one vector represents the whole neighborhood of the sub-sample that represents each class. The value chosen for k has potentially a great importance, when producing the local power mean samples, and ultimately in the accuracy of the classification. The classification results might be inadequate, if the chosen k is too small. In contrast, also a too high value of k may cause problems with classification, because then outliers that may be quite dissimilar to the nearest neighbors can become a part of the k nearest neighbors [7]. It seems that the performance of the proposed new model is quite high, even when high k values are used. When the sub-samples size increases with higher k the multi-local power mean vectors can be said to become more realistic or more comprehensive representations of the classes they represent, this accordingly leads to the

[1]It is well known that $M_p \to \prod_{i=1}^{n} x_i^{1/n}$ when $p \to 0$

Algorithm 1 MLPM-FKNN classifier

Inputs:
labeled(classified) samples: $\{X_i, C_i\}_{i=1}^{N_{tr}}$, unlabeled (query) sample: Q, k: number of neighbors and p: power mean scale.

Outputs:
predicted class: ω^* ($\in \omega_1, \omega_2, .., \omega_T$) for the query sample.

procedure

step 1: Compute the Euclidean distance between $\{X_i\}_{i=1}^{N_{tr}}$ and Q and sort them in an ascending order.

step 2: Set the k nearest neighbors: nn^k of Q considering sorted distances as:
$nn^k(Q) = \{X_i, C_i\}_{i=1}^{k}$.

step 3: Find the local power mean sub-samples: $\{M_{p,r}\}_{r=1}^{t}$ from among the k nearest neighbors: $nn^k(Q)$ for each class: $\{\omega_r\}_{r=1}^{t}$, $1 \leq t \leq T$ by using eq. (3).

step 4: Calculate the Euclidean distance between Q and $\{M_{p,r}\}_{r=1}^{t}$.

step 5: Assign the membership to $\{\omega_r\}_{r=1}^{t}$ by using the weighted distance with the help of eq. (1) and crisp method for computing u_{ij}.

step 6: Classify Q to the corresponding class (ω^*) with which it has the highest degree of membership.

MLPM-FKNN classifier acquiring higher classification accuracies compared with KNN and FKNN classifiers, when value of the parameter k grows. Best classification performance is achieved, when proper parameter values for k and p are search.

4 Data and testing the proposed model

In this section, the used data sets are shortly introduced and the testing methodology is presented.

4.1 Datasets used in testing

Testing the method is carried out by using four data sets all of which are freely available at the UCI Machine learning repository[5] and at the KEEL repository[1]. Table 1 summarizes the fundamental properties of the data sets.

Dataset	Database	Samples	Attributes	Classes
Car	KEEL	1728	7	4
Vehicle	KEEL	846	18	2
Ionosphere	UCI	351	34	2
Thyroid	UCI	215	6	3

Table 1: Information and properties of the data sets.

Before using the data we studied it for entry errors and quality issues and corrected any found problems. The data sets do not contain missing values or nominal variables.

4.2 Testing methodology and determining the parameter values

The proposed new method and the benchmark methods were coded and implemented using MATLAB 2018. In the code, basic calculations have been done using built-in MATLAB functions.

In the all data sets, the data were divided into 40% for training, 40% for validation, and 20% for testing. Stratified random sampling method was applied to ensure that all samples have the same proportions of units representing the different classes present as the whole data set. For cross validation we used hold out method where division between training samples and validation samples was randomly sampled 30 times. The testing (and the data) was divided into two main phases training & validation and testing. The training & validation in the context of this research means using the training and the validation data subsets to optimize the parameter values for the number of nearest neighbors used k and the mean used p. The value of the fuzzy strength parameter m was kept constant at $m = 2$ for both MLPM-FKNN and FKNN methods. This value was chosen based on the recommendations given in [8, 4].

The number of neighbors k was selected from the range $\{1, 2, .., 25\}$. The assumption here was when the k is increasing the performance of proposed MLPM-FKNN classifier would increase. Evidence in favor of this expectation can be found in the paper by Pan et al.[11], where a k-nearest neighbor based on the multi-local means reached a drop in the error rate of classification with a higher k. The parameter value for the power mean, p, was chosen (and optimized) from the set $\{-8, -7, .., 7, 8\}$. When the optimal parameter values were found through iteration with the training and validation subsets the performance was tested with the test subset. The results presented for the proposed new method and the benchmarks are the mean results from the thirty runs made, e.g., the mean accuracy ($(Mp)_{1\times 30}$) and the variance of accuracy of the thirty runs with the optimal p and k.

4.3 Performance measures used

The key measure we used for the evaluation is accuracy [2, 4, 12], but in practice, presenting classification results with accuracy alone is often not enough to be able to fully understand the suitability of a method for a given task. For this reason, we compute also additional performance measures such as sensitivity and specificity and we employ the following definitions [15].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6}$$

Where, TP (true positive), TN (true negative), FP (false positive), and FN (false negative) are on the 2×2 confusion matrix as follows.

$$\begin{array}{c} \\ True(T) \\ False(F) \end{array} \begin{array}{c} \begin{array}{cc} Positive(P) & Negative(N) \end{array} \\ \begin{pmatrix} TP & TN \\ FP & FN \end{pmatrix} \end{array}$$

These performance measures aid to further understand the performance of the the proposed method (or any method). Illustrating this issue, let us look at a simple example: suppose that a confusion matrix exists, where $TP = 0$, $TN = 95$, $FP = 0$, and $FN = 5$. In this matrix accuracy= 95% and specificity= 100%, but the precision and the sensitivity are zero. It is easy to understand that any model with zero precision is too weak to be used in any real-world purposes and this highlights the importance of including also other

performance measures than prediction accuracy, when model performance is reported. To compute these performance measures for the binary class problems (Vehicle and Ionosphere), equations (4), (5), and (6) were used. For the multi-class problems (Car and Thyroid) we applied performance measure computations, such as they are proposed in [15]. Accordingly, TN, TP FP, and FN first and then sensitivity and specificity were computed for each class. The average of sensitivities and specificities for each class were considered to the final performance measures of the classifier exactly as suggested in [6].

A paired t-test was used, in vein with [2] to test, whether the difference of the (best) mean accuracies achieved with 1) MLPM-FKNN and FKNN and 2) MLPM-FKNN and KNN significantly differs from zero under a 0.05 significance level. This is to say, we tested for whether a statistically significant difference in the classification accuracy of the different methods can be found.

5 Results and discussions

The first results presented here are from the training & validation step of testing the method. The values of the accuracy and performance measures achieved in the training & validation step are presented in Table 2. In the table "Max.Acc." refers to the the maximum of means of accuracy and "sensitivity" and "specificity" the means of these values. As mentioned earlier, optimal values for the parameters (Opt.param.val.) k and p were being picked for when the mean accuracy was at maximum.

Dataset	Max Acc.	Sensitivity	Specificity	Opt.param.val.
Car	0.9230	0.7903	0.9654	$p = 0$, $k = 25$
Vehicle	0.9412	0.8723	0.9628	$p = 1$, $k = 11$
Ionosphere	0.8881	0.8744	0.9265	$p = 1$, $k = 16$
Thyroid	0.9229	0.8813	0.9367	$p = 3$, $k = 7$

Table 2: Accuracy and related results for MLPM-FKNN in the training & validation step.

We can see from the Table 2, that using arithmetic mean, $p = 1$ has produced the maximum accuracy with the Vehicle and Ionosphere data sets, while geometric and cubic mean where optimal w.r.t. accuracy with the Car and Thyroid data sets respectively. It can be noted that the specificity is higher than the sensitivity in all cases considered.

By observing Figures 1 and 2 one can visually inspect the impact that different combinations of the parameters p and k have on selected performance measures for the Vehicle data set in the training & validation step.

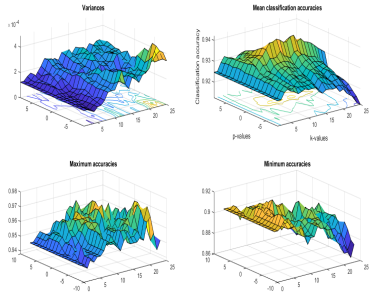


Figure 1: Variance and accuracies for different (p, k) parameter combinations with the Vehicle data

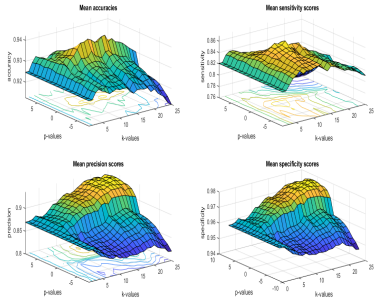


Figure 2: Mean accuracy and performance measures for different (p, k) parameter combinations with the Vehicle data

To understand how well the proposed new method performs in comparison with the two benchmark methods during the training & validation step, we show classification results for the three classifiers with all four data sets in Table 3. In addition to the maximum of the mean accuracy (Mean Acc.), variance, confidence interval (CI), and the optimal parameter k value (Op. K) is presented. As observed above, the mean accuracy and the variance are the result of the hold out method with 30 repetition.

It is visible from Table 3 that the proposed MLPM-FKNN method can outperform the FKNN and the KNN in mean classification accuracy, while the confidence intervals remain narrow. It seems that using high values for the parameter k significantly increases the performance of the classification. This is interesting, because low values of k seem to be working better

Data	Measure	MLPM-FKNN	FKNN	KNN
Car	Mean Acc.	0.9230	0.8823	0.8740
	Variance	1.13e-04	6.19e-05	6.09e-05
	CI	[0.9191, 0.9270]	[0.8794, 0.8853]	[0.8711, 0.8769]
	Op. K	$k = 25$	$k = 4$	$k = 3$
Vehicle	Mean Acc.	0.9412	0.9291	0.9274
	Variance	1.46e-04	1.92e-04	1.98e-04
	CI	[0.9367, 0.9457]	[0.9239, 0.9343]	[0.9222, 0.9327]
	Op. K	$k = 11$	$k = 4$	$k = 1$
Ionosphere	Mean Acc.	0.8881	0.8443	0.8431
	Variance	5.57e-04	4.24e-04	4.66e-04
	CI	[0.8793, 0.8969]	[0.8366, 0.8520]	[0.8350, 0.8512]
	Op. K	$k = 16$	$k = 4$	$k = 3$
Thyroid	Mean Acc.	0.9229	0.9167	0.9101
	Variance	5.92e-04	6.16e-04	6.05e-04
	CI	[0.9138, 0.9320]	[0.9074, 0.9259]	[0.9009, 0.9193]
	Op. K	$k = 7$	$k = 3$	$k = 1$

Table 3: Classification results for all three classifiers for the four datasets. CI = confidence interval

Dataset	Paired with	MLPM-FKNN (p-value)	test-statistic
Car	FKNN	$5.06e^{-24}$	significant
	KNN	$4.15e^{-28}$	significant
Vehicle	FKNN	$6.28e^{-04}$	significant
	KNN	$1.42e^{-04}$	significant
Ionosphere	FKNN	$2.26e^{-10}$	significant
	KNN	$1.91e^{-10}$	significant
Thyroid	FKNN	0.333	not-significant
	KNN	0.043	significant

Table 4: T-test results on the statistical significance in classification accuracy of the proposed new method.

for the benchmark methods and $k = 1$ has been the optimum for the KNN method with two of the data sets used. This finding corroborates what was found previously by Derrac at el.[4].

In Table 4, the results from paired t-test for testing the statistical significance of the classification accuracies between the proposed new classifier and the benchmarks are presented. The test results show that the claim that MLPM-FKNN has higher performance over the benchmarks finds statistical corroboration. One can observe that only in the case of the thyroid data the proposed new classifier cannot be said to produce statistically significantly better results in comparison with the result obtained from FKNN.

5.1 Performance with the testing data

This section presents the results obtained with the testing data (constituting of 20% of the data set size) on the classifiers, whose parameter values have been optimized in the training & validation phase. In this testing phase, the test sample set was evaluated with training set samples which were saved during the hold-out crossvalidation to get optimal parameter values from validation results. Then mean accuracies and other performance measures were computed. Table 5 shows the results obtained for the proposed new classifier and for the benchmark classifiers for mean classification accuracies (Mean Acc.), sensitivity (Mean Sen.), specificity (Mean Spe.) and confidence interval (CI) and variance for all the four data sets. The results from the test set show that MLPM-FKNN classifier has a higher classification accuracy in all cases compared to the benchmarks.

Data	Measure	MLPM-FKNN	FKNN	KNN
Car	Mean Acc.	0.9246	0.8742	0.8632
	Mean Sen.	0.8105	0.6704	0.6085
	Mean Spe.	0.9667	0.9119	0.9014
	CI	[0.9204, 0.9288]	[0.8688, 0.8796]	[0.8577, 0.8687]
	Variance	1.26e-04	2.08e-04	2.15e-04
Vehicle	Mean Acc.	0.9294	0.9051	0.9006
	Mean Sen.	0.8568	0.7907	0.7910
	Mean Spe.	0.9521	0.9432	0.9355
	CI	[0.9241, 0.9347]	[0.8966, 0.9136]	[0.8932, 0.9080]
	Variance	2.03e-04	5.18e-04	3.95e-04
Ionosphere	Mean Acc.	0.8381	0.7957	0.7952
	Mean Sen.	0.8212	0.7686	0.7694
	Mean Spe.	0.8925	0.9215	0.9113
	CI	[0.8281, 0.8481]	[0.7875, 0.8039]	[0.7869, 0.8036]
	Variance	7.237e-04	4.81e-04	4.97e-04
Thyroid	Mean Acc.	0.9434	0.9279	0.9233
	Mean Sen.	0.9256	0.8520	0.8449
	Mean Spe.	0.9584	0.9254	0.9214
	CI	[0.9316, 0.9552]	[0.9171, 0.9387]	[0.9138, 0.9327]
	Variance	9.95e-04	8.34e-04	6.40e-04

Table 5: Classification results for all three classifiers for the four datasets. CI = confidence interval

Also mean sensitivity and specificity values show improved results compared to FKNN and KNN. Using local means in this manner also seem to lower variances a bit compared to the benchmark methods. Table 6 illustrates the results of the statistical t-test on the accuracies of the proposed method and benchmarks over the test sample. From Table 6, it is also confirmed that the MLPM-FKNN method has produced significantly higher accuracies in classification for the test set than accuracies of FKNN and KNN classifiers.

Dataset	Paired with	MLPM-FKNN (p-value)	test-statistic
Car	FKNN	$1.64e^{-18}$	significant
	KNN	$2.88e^{-25}$	significant
Vehicle	FKNN	$6.73e^{-06}$	significant
	KNN	$2.41e^{-08}$	significant
Ionosphere	FKNN	$4.95e^{-08}$	significant
	KNN	$1.06e^{-12}$	significant
Thyroid	FKNN	0.051	not-significant
	KNN	0.008	significant

Table 6: T-test results on the statistical significance for test set in classification accuracy of the proposed new method with benchmarks.

6 Conclusions

The aim of this paper was to present a new classification method in the k-nearest neighbor family of classification algorithms that is based on using the power mean to calculate a multi-local mean that serves as a representative for the representative classes near the to-be-classified sample. The proposed new classifier includes also elements of the previously presented fuzzy KNN algorithm in that the strength of class membership of the to-be-classified sample is also taken into consideration in the classification. The results of this research show that the proposed MLPN-FKNN classifier offered improved classification accuracy compared to the KNN and the FKNN classifiers with the benchmark data sets. The performance was tested by using four different data sets and the improvement in classification accuracy was found to be statistically significant.

It can be observed in connection with the new proposed method that the best classification accuracy is found with, high k values (number of nearest neighbors considered). This we attribute to the use of multi-local power mean. We note that the computation time required for using the proposed MLPM-FKNN method is also longer than that of the benchmarks, it is clear that this is a result of the more numerous calculations needed caused by the increase in parameters and thus testing of more possible parameter combinations to find optimal parameters.

Future research possibilities include, e.g., testing the effect including power means would have together with other variants of KNN algorithms, such as IV-KNN [4], MLM-KHNN [11], and PTVPSO-FKNN [2].

7 Acknowledgment

This research was partly supported by the Academy of Finland project Manufacturing 4.0.

References

[1] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318. URL https://doi.org/10.1007/s00500-008-0323-y

[2] H. L. Chen, D. Y. Liu, B. Yang, J. Liu, G. Wang, S. J. Wang, An adaptive fuzzy k-nearest neighbor method based on parallel particle swarm optimization for bankruptcy prediction, Lecture Notes in Computer Science 6634 LNAI (PART 1) (2011) 249–264.

[3] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[4] J. Derrac, F. Chiclana, S. García, F. Herrera, An Interval Valued K-Nearest Neighbors Classifier, in: Proceedings of the International Joint Conference IFSA-EUSFLAT-2015, 2015.

[5] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017). URL http://archive.ics.uci.edu/ml

[6] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognition Letters 30 (1) (2009) 27–38. URL http://linkinghub.elsevier.com/retrieve/pii/ S0167865508002687

[7] J. Gou, Y. Zhan, Y. Rao, X. Shen, X. Wang, W. He, Improved pseudo nearest neighbor classification, Knowledge-Based Systems 70 (2014) 361375.

[8] J. M. Keller, Michal R. Gray, J. R. James A. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Transactions on Systems, Man, and Cybernetics 15 (4) (1985) 580–585.

[9] D. Koloseni, J. Lampinen, P. Luukka, Differential evolution based nearest prototype classifier with optimized distance for the features in the data sets, Expert Systems with Applications 40 (10) (2013) 4075–4082.

[10] C. Liu, L. Cao, P. S. Yu, A hybrid coupled k-nearest neighbor algorithm on imbalance data, in: Proceedings of the International Joint Conference on Neural Networks (2014) 2011–2018.

[11] Z. Pan, Y. Wang, W. Ku, A new k-harmonic nearest neighbor classifier based on the multi-local means, Expert Systems with Applications 67 (115-125) (2017) 21–27.

[12] F. C. Rhee, C. Hwang, An interval type-2 fuzzy k-nearest neighbor, in: The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03., Vol. 2, 2003, pp. 802–807 vol.2.

[13] J. S. Sanchez, R. Barandela, A. I. Marques, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (2003) 1015–1022.

[14] J. S. Sanchez, F. Pla, F. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recognition Letters 18 (6).

[15] A. Tharwat, Classification assessment methods, Applied Computing and Informatics, (2018) accepted.
URL https://doi.org/10.1016/j.aci.2018.08.003

[16] K. Toduka, Y. Endo, Fuzzy K-nearest neighbor and its application to recognize of the driving environment, in: IEEE International Conference on Fuzzy Systems, 2006.

[17] S. Watanabe, Pattern Recognition, Human and Mechanical., Trends in Logic, New York (N.Y.): Wiley, 1985.

Publication II

Mailagaha-Kumbure, M., Lohrmann, C., and Luukka, P.
A study on relevant features for intraday S&P 500 prediction using a hybrid feature selection approach

Reprinted with permission from
Lecture Notes in Computer Science
Vol. 13163, pp. 93–104, 2022.

A Study on Relevant Features for Intraday S&P 500 Prediction Using a Hybrid Feature Selection Approach

Mahinda Mailagaha Kumbure, Christoph Lohrmann, and Pasi Luukka

School of Business and Management, LUT University, Lappeenranta 53850, Finland
{mahinda.mailagaha.kumbure,christoph.lohrmann,pasi.luukka}@lut.fi

Abstract. This paper investigates relevant features for the prediction of intraday S&P 500 returns. In contrast to most previous research, the problem is approached as a four class classification problem to account for the magnitude of the returns and not only the direction of price movements. A novel framework for feature selection using a hybrid approach is developed that combines correlation as a fast filter method, with the wrapper method differential evolution feature selection (DEFS) that deploys distance-based classifiers (k-nearest neighbor, fuzzy k-nearest neighbor, and multi-local power mean fuzzy k-nearest neighbor) as evaluation criterion. The experimental results show that feature selection successfully discarded features for this application to improve the test set accuracies or, at a minimum, lead to similar accuracies than using the entire feature subset. Moreover, all setups in this study ranked technical indicators such as 5-day simple moving average as the most relevant features in this application. In contrast, the features based on other stock indices, commodities, and simple price and volume information were a minority within the top 10 and top 50 features. The prediction accuracies for the positive return class considerably higher than the negative class predictions with over 70% accuracy compared to 30%.

Keywords: Classification · Financial market · Machine learning · K-nearest neighbor · Supervised feature selection

1 Introduction

Forecasting future stock prices or returns is considered an essential subject in finance research, but it has been a challenging task due to the non-linear and non-stationary behavior of the stock market [1]. Having representative and informative input features is crucial for attempting to predict movements of the stock market in the prediction [2]. Many studies in the stock market literature have attempted to determine which features can be used to forecast the market (see eg., [2–4]). However, most of those studies have been limited to a small number of features. In this study, we focus on examining possibly relevant features from a large set of input features, including areas such as the macro-economy, technical

indicators, commodities, exchange rates and the stocks as well as stock indices. To address the case of dimensionality associated with fitting machine learning models to high-dimensional data, this paper develops an approach based on feature selection to identify a subset of relevant features for the stock market forecasting. Supervised "feature selection" refers to the process of selecting a subset of the relevant features from the set of the existing features [5], which will improve the classification accuracy or, at a minimum, not deteriorate it considerably [6]. Feature selection methods can be categorized into filter methods, which are used for pre-processing, wrapper methods that use a classifier for the selection, and embedded methods, which include feature selection already in the classifier itself [7]. Within this study, a hybrid approach is pursued where initially a filter method is applied to discard linearly associated features before a computationally more expensive wrapper method is provided with this subset to obtain a final feature subset.

Among the various classification methods in machine learning, k-nearest neighbor classifiers are adopted for this investigation due to their simplicity, easy implementation [8], and remarkable achievements in some stock market prediction applications [9, 10]. We selected k-nearest neighbor (KNN) [11], fuzzy k-nearest neighbor (FKNN) [12], and also the multi-local power means fuzzy k-nearest neighbor (MLPM-FKNN) [8] classifier, which was introduced recently [8]. We used this new FKNN method (MLPM-FKNN) since it has shown more robust to outliers and random variables than original ones according to [8]. Distance-based classifiers generally account for all features equally when calculating the distances between points. Thus, we combine the KNN classifiers with a hybrid feature selection approach to address the potentially detrimental impact of irrelevant features on this type of classification.

The remainder of the paper is organized as follows. Section 2 shortly discusses the theoretical concepts applied in this research. A detailed description of the data set used and the empirical process developed are provided in Section 3. The empirical results of the feature selection and stock return prediction are presented in Section 4 and the concluding remarks on our study are presented in Section 5.

2 Preliminaries

2.1 Differential Evolution Feature Selection

Differential evolution (DE) was originally proposed in [13] and is a well-known population-based heuristic optimization method [14]. It has been applied in many disciplines due to its simplicity in implementation, fast convergence, and robustness [15, 16]. In the DE method, optimal solutions are searched across four main steps: initialization, mutation, crossover, and evaluation of objective function [17]. To make DE more reliable as a wrapper method for feature selection (FS), the study in [15] has introduced several modifications to its search strategy, referring to the enhanced version as DEFS. This method is applied and implemented in our study, and the complete feature selection process can be found in [15].

2.2 K-Nearest Neighbor Classifier

The idea behind the k-nearest neighbor methods is simple and is based on measuring the distance of a query sample (the sample to be classified) to the labeled samples in the training data. Formally speaking, it starts with computing distances from the query sample (y) to the training set X that is composed of N samples $X = x_1, x_2, ..., x_N$ that belong to C classes. Also each sample $x_i = \{x_i^1, x_i^2, ..., x_i^M\}, x_i^c$ is characterized by M feature values and one class label c ($c \in C$). Several distance functions are available to be used for this step, but the one that is often used is the Euclidean distance metric. Next, a set of k nearest neighbors of y together with corresponding class labels is found. In the final step, the class for y is predicted by assigning the most common class among the set of k nearest neighbors.

2.3 Fuzzy K-Nearest Neighbor Classifier

In the FKNN, for the query sample y, a membership degree for each class is assigned, and the class decision is made based on the highest membership degree [12]. In particular, the allocated membership degree of y for each class indicated by the k nearest neighbors is calculated using the following membership function:

$$u_i(y) = \frac{\sum_{j=1}^{k} u_{ij}(1/\left\|y - x_j\right\|^{2/(m-1)})}{\sum_{j=1}^{k}(1/\left\|y - x_j\right\|^{2/(m-1)})} \tag{1}$$

where, $m \in (1, +\infty)$ is a fuzzy strength parameter and u_{ij} is the membership of the j^{th} sample in the i^{th} class of the training set X. To compute u_{ij}, we used a crisp labeling approach [12] in which full membership is assigned to a known class of each labeled sample and zero membership for all other classes. The fuzzy strength parameter m influences the membership degree by providing relative importance to the distance from y to k nearest neighbors to be weighted. For this parameter a common choice is a value of 2.

2.4 MLPM-FKNN Classifier

As another variant of the KNN, the multi-local power mean fuzzy k-nearest neighbor (MLPM-FKNN) classifier that was introduced in [8] is particularly preferable to the standard KNN and FKNN methods in situations where highly imbalanced classes are present (i.e., there are only a few samples in some classes). In the MLPM-FKNN, the obtained set of k nearest neighbors of y is further grouped into each of the classes they belong to, and then r ($\leq C$) number of multi-local power mean vectors ($M_{t=1}^{r}$) are computed (r is the number of classes presented in the set of nearest neighbors) using following formula:

$$M_t = \begin{cases} \prod_{i=1}^{n} x_i^{1/n}, & \textbf{if } p = 0 \\ (\frac{1}{n}\sum_{i=1}^{n} x_i^p)^{1/p}, & \textbf{if } p \neq 0 \end{cases} \tag{2}$$

where, p is a real-valued parameter.

In the next step, the distances of y to multi-local power mean vectors are measured using a suitable distance function (e.g., Euclidean distance). These distances are then used to find the class memberships for y, utilizing eq. (1), and a decision on the class is made according to the highest membership degree. In this study, however, we slightly updated the learning part of the MLPM-FKNN algorithm before applying it. First, the training data is grouped into classes, and then the sets of k nearest neighbors of y are found from each class. After this, for each k nearest neighbor set in each class, the multi-local power mean vectors are computed. The next steps are the same as in the MLPM-FKNN algorithm. We noticed that the updated version[1] of the MLPM-FKNN outperforms the original one in the classification of the intraday return of the S&P 500—thus, we report the results only for the updated classifier together with the benchmark algorithms in this study.

3 Data and Hybrid Feature Selection

3.1 Data Description

The data set employed in this study contains daily S&P 500 stock index prices and a set of expanding variables that are assumed to be relevant for the prediction of this index. Similar types of variables based on commodity prices, exchange rates, technical indicators, and other stock indices have also been used in previous studies [4, 18, 19]. Historical time series with 4182 samples were obtained from Yahoo Finance [20] and FRED Economic Data [21] during the period from January 10th, 2007 to October 10th, 2020.

Input Features. We considered and collected 302 features summarized by [22] as frequently used for stock market predictions in previous studies. These selected 302 features were at the top of the list concerning the frequency of usage in stock market forecasting applications according to the study of [22]. The list of features across their categories and the corresponding number of input features used in this study are summarized in Table 1. Technical indicators (TIs) are further categorized into "Basic TIs," which refers to Open, High, Low, and Close prices and Volume of the S&P 500 index and "Other TIs" refers to all other TIs that had to be computed based on the Closing price of the S&P 500, and their variants. From [22], one can find the definitions of commonly known abbreviated forms of the features in Table 1. The different variants of TIs were created by changing the time-period (n in days) and other parameters (n_1, n_2 for EMA, high(H), low(L), middle(M) for Bollinger bands, $slow, fast, sign$ in MACD). For the other time series where it was possible we collected all their basic TIs. Also, % changes of the selected TIs (e.g., Williams R, Stochastic K) and other time series were used.

[1] The MATLAB code of the updated MLPM-FKNN algorithm can be found from https://github.com/MahindaMK/Multi-local-Power-means-based-fuzzy-k-nearest-neighbor-algorithm-MLPM-FKNN

Table 1. Information of the input features.

Category	Feature list	No. of features
Basic TIs	Open, High, Low, Close, Volume (n)	14
Other TIs	RSI(n), MACD($slow$,$fast$,$sign$), EMA(n), Bias (n), SMA(n), Disparity(n), OBV, Return(n), Williams %R(n), CCI(n), Momentum(n), MFI(n), Stochastic %K(n), Stochastic %D(n), TMA(n), Bollinger bands ($H/M/\ L$), Chaikin Volatility, Price oscillator (n_1, n_2), Typical price, TRIX	140
Macro-economy	Treasury Bills, Term Spread, AAA Corporate Bond, Treasury Constant Maturity Rate, BAA Corporate Bond, Treasury Yields, Default Spread	27
Commodities	Crude Oil, Gold, Silver	18
Exchange rates	USD/ NTD, USD/JPY, USD/ GBP, USD/ CAD, USD/ CNY	25
Other stocks/ stock indices	Hang Seng, SSE, CAC40, DJIA, NASDAQ Microsoft, Amazon, JPM, General Electric, JNJ, Apple Inc., Wells Fargo, Exxon Mobil	78

In the data, other technical indicators ("Other TIs") were generated using closing prices of the S&P 500 stock index—for some indicators, high, low, and open prices were also used. Since some indicators require initial data before their value can be calculated, we started our data after such a period (e.g., 5-day moving average MA(5)). Next, the missing values that occurred when the time series data of the S&P 500 was concatenated with other time series were replaced using linear interpolation. All features in the data were scaled into the interval $[0, 1]$. Technical indicators are commonly used to infer trading signals for when a stock or market is overbought or oversold, and corresponding selling and buying decision may be profitable. Therefore, the continuous data of technical indicators was converted into discrete data (trading signals) to represent actual trading signals rather than the numeric values for the technical indicators they can be inferred from. For the relative strength index (RSI) and commodity channel index (CCI), we followed the techniques presented in [19] to convert them into trading signals. To create the signals from other technical indicators, a quartile-based approach was applied. That is, if the TI $> Q3$ then set to 1, if $Q3 >=$ TI $> Q2$ then set to $+0.33$, if $Q2 >=$ TI $> Q1$ then set to -0.33, and if the TI $=< Q1$ then set to -1. Thus, four discrete equally-spaced categories were created, which corresponds to the number of classes.

Output Variable. As in [4], in this paper, the intraday (open-to-close) return of the S&P 500 index was selected as target and set up as a multi-class variable within four classes ("1", "2", "3" and "4") according to daily magnitude of the return. In this variable, class label "4" represents the intraday returns that are smaller than -0.5% (i.e., strong negative), "3" between -0.5% and 0.0% (i.e., slightly negative), "2" between 0.0% and 0.5% (i.e., slightly positive) and "1"

larger than 0.5% (i.e., strong positive). The cut-off using +/- 0.5% was selected since it results in quite balanced classes (class 1: 27.42%, class 2: 28.08%, class 3: 23.92%, and class 4: 20.58%) and was also deployed in [4].

3.2 Hybrid Feature Selection

We first split the data set into two data sets, one for training (from January 10th, 2007 to January 18th, 2018) and the other for testing (from January 19th, 2018 to October 10th, 2020). The training sample was further repeatedly divided into 80% for the training and 20% for the validation sets, and these subsets were used in the feature selection and parameter optimization processes. The test sample was kept to evaluate the predictive performance of the trained classifiers with the selected features and optimized parameters in the last phase.

Feature Removal by Pearson Correlation Coefficients. Wrapper methods, especially those including optimization such as the DEFS, require comparably high computational time since they use an iterative process including a classifier for feature selection. Thus, the methodology in this paper includes Pearson correlation, which is used as a filter method before DEFS. The aim is to efficiently remove linearly dependent features initially in order to reduce the number of features and the computational complexity for the DEFS wrapper method.

The Pearson correlation coefficient is a simple filter method that is used in the context of feature selection to measure the linear dependence between a variable and the target (class label) [7], where higher (absolute) correlation is a sign of the relevance of a variable for the target. In contrast to that, a high (absolute) correlation between different explanatory variables may indicate that keeping both variables instead of a single one may not add information, thus potentially being redundant. Here, we followed the second approach and calculated the Pearson correlation coefficient between pairs of features f_i, f_j for $i, j \in \{1, .., 302\}$ to measure how much information they potentially have in common. A threshold of 0.95 for the absolute correlation between two features f_i and f_j was set so that for two variables that have a very high absolute correlation, one of these two variables can be removed. If $corr(f_i, y) > corr(f_j, y)$ then remove f_j, else remove f_i, where y is the class variable.

Parameter Settings and Exploring Relevant Features with DEFS. In this step, the DEFS with each selected classifier was deployed to find the most relevant features for forecasting the intraday return of the S&P 500 index. As parameters in the DEFS, we set both population size and the number of iterations to 50 and the crossover rate to 0.5. Also, we provided $5, 10, 15, ..., 50$ as the desired number of features to be selected (i.e., number of the features in the resulting feature subset). Other parameters involved in the DEFS algorithm were specified according to the previous study in [15]. For the nearest neighbor classification methods, the number of nearest neighbors k was kept constant at

20, the fuzzy strength parameter m was set to 2, and the power mean parameter p was 1.5, as in the previous studies [8, 12, 23]. We received the selected feature subsets (i.e., feature indices) and classification error rates across the different iterations as outputs from the DE function. This process was cross-validated using the holdout method where the training and the validation sets were generated 10 times randomly.

4 Experiment Results

4.1 Results in the Correlation Analysis

The evidence from the correlation coefficients based analysis suggests eliminating some features due to linear dependence effects. Some pairs of features, such as the low price of the Hang Seng and of the SSE composite index were even perfectly correlated. Overall, in this analysis, the dimensionality of the data set was reduced by discarding potentially redundant features, keeping 207 out of the 302 original features for the subsequent DEFS and classifier training.

4.2 Results in the DEFS

The wrapper DEFS that is the second component in the hybrid feature selection (filter+wrapper) was applied together with each nearest neighbor classifier. Fig. 1 illustrates the frequency of the top 50 features according to the DEFS algorithm in combination with each of the classifiers during the holdout cross-validation.

At first glance, it is apparent that for each of the classifiers, the majority of selected features are technical indicators in the list of the top 50 features. The most important features across all these classifiers appear to be the 5-day moving average (SMA (5)) and the Silver low price both being consistently within the top 10 higher ranked features. Other features that were at least contained twice in the top 10 include technical indicators representing model classification of the last 5-15 days such as the SMA (15), Williams R(10), Disparity(10), RSI(6), Chaikin Volatility(10). Finally, it is noteworthy that no macro-economic or exchange rate variations are contained in any of the top 50 features.

4.3 Prediction Performance

In the training and validation step, the parameters of the classification models were optimized by using 30 runs of cross-validation with the holdout method. The best value for k was searched from the range $\{1, 2, ..., 30\}$ during the training & validation. The value for the parameter p in power mean was chosen and optimized from the set $\{0, 0.5, 1, ..., 5\}$ for the MLPM-FKNN model. Accordingly, the optimal parameter values selected were those corresponding to the maximum validation accuracy. Fig. 2 illustrates the mean classification errors (%) of each model in the DEFS for each feature subset size.

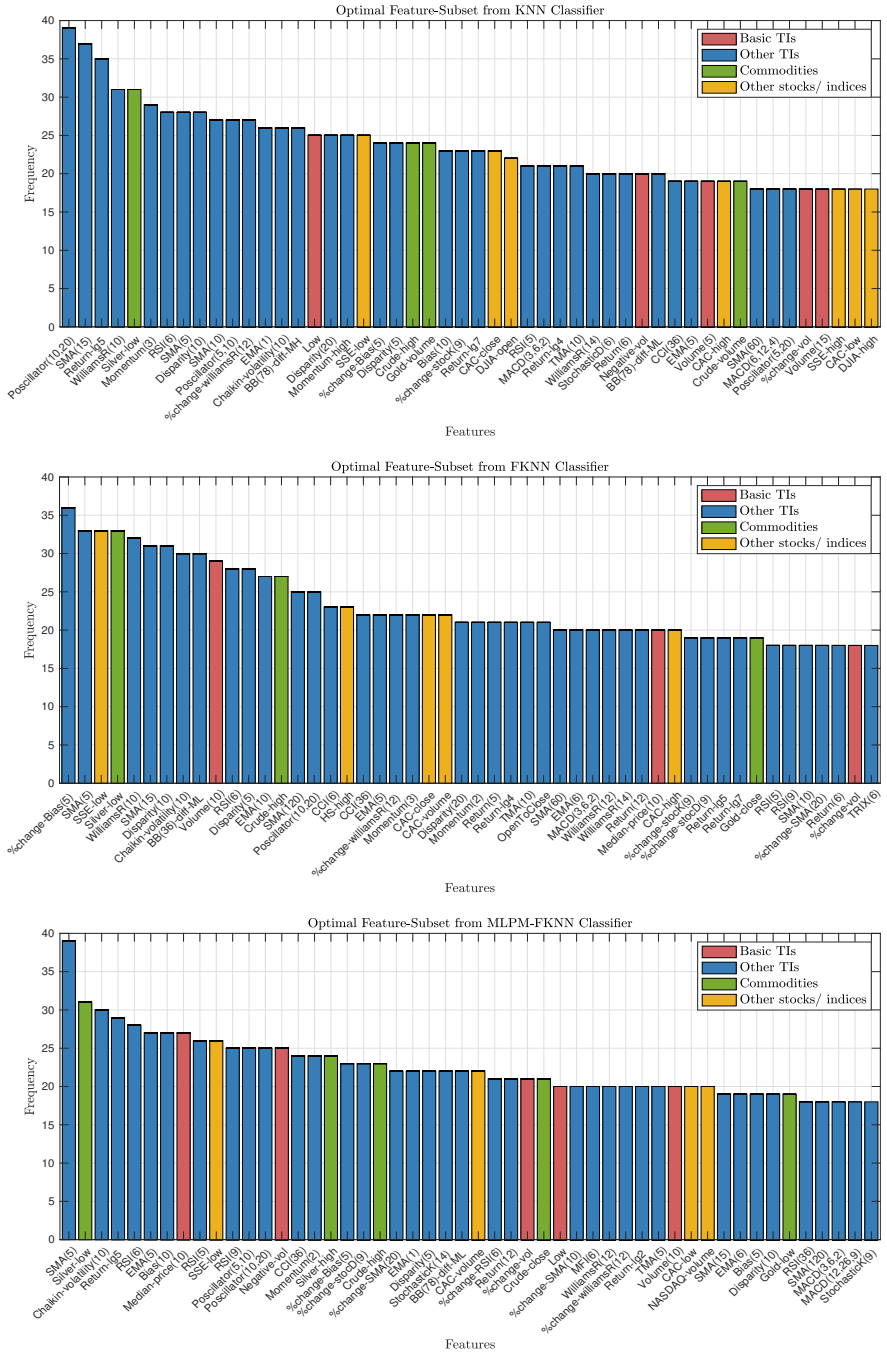


Fig. 1. The frequency of the 50 best features (descending order) based on the DE feature selection with each classifier.

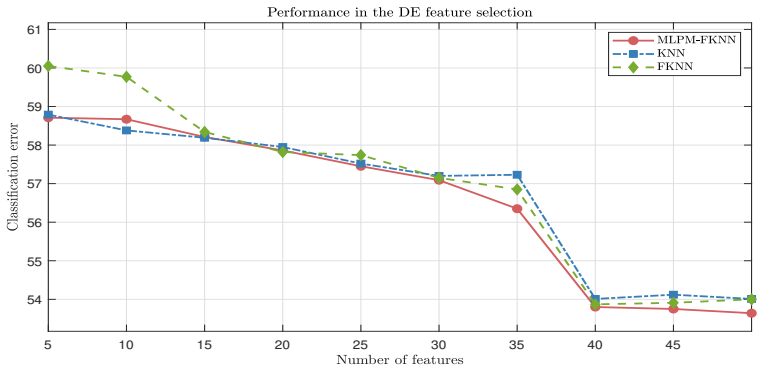


Fig. 2. Mean classification error (%) across ten runs of cross-validation for each feature subset size.

According to the results in Fig. 2, we can see that when the size of the optimal feature subsets with DEFS increases, the MLPM-FKNN model achieves better performance than the KNN and FKNN classifiers. This can be expected because the samples with more features make local power mean vectors more robust (and representative) in the learning. Looking at the classification performance, the mean errors appear quite high compared to other classification tasks with the real accuracies for each model only being in the mid-thirty accuracies. However, it should be kept in mind that this task is a four class problem and that low classification accuracies do not necessarily mean that a trading strategy based on these results would not be able to generate excess returns [4, 24]. When aggregating the four-class prediction (1: strong positive, 2: slightly positive, 3: slightly negative, and 4: strong negative) to a binary class level by counting class 1 and class 2 prediction as "positive" and class 3 and class 4 prediction as "negative", the results become clear and can more easily be compared with other binary studies.

Table 2 presents the aggregated accuracies on the positive and negative classes by classifier on subset size.

According to the results in Table 2, it is clear that the positive class can be predicted more accurately (around 70%) than the negative class (around $20 - 30\%$). This pattern appears consisted for all classifiers and feature subset rises. It is interesting to highlight that this result is not based on the class imbalance, which is not very high (57% positive, 43% negative observations). This result suggests that negative predictions are more challenging and, thus, less accurate for each of the KNN classifiers. This finding that the prediction of "negative" direction of the S&P 500 with less accuracy than "positive" upward movement appears consistent with the prediction results presented by the study of [4], which investigated the intraday S&P 500 return during the period (Oct 2010 to March 2018) using a different classifier (Random Forest).

Table 2. Prediction performance (%) on positive and negative classes with test sample for all classifiers and for each feature subset.

	MLPM-FKNN		KNN		FKNN	
Subset	Positive	Negative	Positive	Negative	Positive	Negative
5	70.55 (4.99)	30.14 (4.4)	79.18 (3.89)	21.71 (5.3)	67.87 (2.78)	35.49 (3.02)
10	68.71 (4.9)	32.42 (3.76)	77.53 (3.96)	21.45 (4.56)	75.97 (3.57)	29.48 (3.55)
15	73.35 (4.41)	28.08 (4.04)	80.23 (3.73)	24.56 (3.4)	76.64 (4.04)	25.77 (4.12)
20	73.78 (3.24)	26.48 (3.02)	73.78 (2.96)	27.77 (3.12)	77.59 (4.01)	26.38 (4.49)
25	78.00 (3.85)	20.16 (3.3)	78.22 (3.15)	25.98 (3.45)	74.85 (4.2)	28.48 (4.2)
30	75.31 (3.86)	21.05 (4.34)	73.71 (3.45)	27.77 (3.55)	71.95 (3.37)	30.73 (3.47)
35	74.38 (3.31)	23.12 (3.51)	74.73 (3.19)	27.80 (3.82)	77.14 (3.2)	26.90 (4.05)
40	73.00 (4.35)	25.02 (4.3)	77.01 (4.35)	24.86 (4.43)	75.97 (3.49)	28.66 (3.64)
45	70.60 (3.68)	26.60 (4.07)	77.26 (4.48)	24.56 (4.75)	73.83 (4.27)	28.15 (4.2)
50	72.08 (3.82)	24.81 (3.72)	76.91 (4.25)	26.43 (5.25)	73.06 (3.88)	27.02 (3.68)
302	82.24 (3.92)	16.64 (4.15)	90.18 (3.55)	13.17 (4.28)	88.38 (4.08)	14.63 (4.16)

Furthermore, we can see from Table 2, the last row that shows the test set accuracies with all features are the highest on the positive return but the lowest on the negative return for all classifiers. This indicates that the predictors with all features preferred more to predict positive return but not much on the negative return. This verifies the effectiveness of using relevant features with DEFS since it improves, or at a minimum, does not deteriorate the classification accuracy on the negative return of the intraday return of the S&P 500 even when a substantial amount of the input features is removed.

5 Conclusion

This paper examined potentially relevant features for the prediction of the intraday S&P 500 return in a four-class classification problem. A hybrid feature selection approach consisting of Pearson correlation (filter) and a DEFS algorithm (wrapper) was deployed to select only relevant features for the classification. Three distance-based classifiers (KNN, FKNN, and MLPM-KNN) were used as an evaluation criterion for the DEFS, and all three classifiers almost exclusively ranked technical indicators among the most relevant features for the intraday return S&P 500 predictions. The 5-day simple moving average (SMA (5)) and the Silver low price appear particularly relevant in this study given that all three classifiers include them in their top 10 features out of the more than 207 features included in the DE feature selection. Moreover, macro-economic features are weakly correlated to many technical indicators - however, those features do not appear in the top 50 feature subsets for all classifiers. This remains an open question in our study and is required for further investigations. The aggregated results in this study highlighted that prediction for "negative" movements was much less accurate than for "positive" movements of the S&P 500. This may suggest that the features in this study are related for the prediction of "positive" but other additional features, may be relevant for the predicting "negative"

development in the S&P 500. In the future, this finding can be further investigated for the S&P 500 index with different time horizons and classifiers and on different global stock markets.

Acknowledgment

This research was supported by the Finnish Foundation for Share Promotion (Pörssisäätiö).

References

1. Kazem, A., Sharifia, E., Hussainb, F. K., Saberic, M., Hussain, O. K.: Support vector regression with chaos-based firefly algorithm for stock market price forecasting. Applied Soft Computing**13**, 947-958 (2013). doi: 10.1016/j.asoc.2012.09.024
2. Zhang, X., Hu, Y., Xie, K., Wang, S., Ngai, E. W. T., Liu, M.: A causal feature selection algorithm for stock prediction modeling. Neurocomputing **142**, 48-59 (2014). doi: 10.1016/j.neucom.2014.01.057
3. Tsai, C. F., Hsiao, Y. C.: Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. Decision Support Systems **50**(1), 258-269 (2010). doi: 10.1016/j.dss.2010.08.028
4. Lohrmann, C., Luukka, P.: Classification of intraday S&P500 returns with a Random Forest. International Journal of Forecasting **35**, 390–407 (2019). doi: 10.1016/j.ijforecast.2018.08.004
5. Kittler, J., Mardia, K. V.: Statistical pattern recognition in image analysis. Journal of Applied Statistics **21**, 61–75 (1994)
6. Liang, J., Yang, S., Winstanley, A.: Invariant optimal feature selection: A distance discriminant and feature ranking based solution. Pattern Recognition, **41** , 1429–1439 (2008). doi: 10.1016/j.patcog.2007.10.018
7. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering. **40**, 16–28 (2014).
8. Kumbure, M. M., Luukka, P., Collan, M.: An Enhancement of Fuzzy K-Nearest Neighbor Classifier Using Multi-Local Power Means. In: Proceeding of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), pp. 83–90, Atlantis Press (2019)
9. Zhang, N., Lin, A., Shang, P.: Multidimensional k-nearest neighbor model based on EEMD for financial time series forecasting. IPhysica A: Statistical Mechanics and its Applications **477**, 161-173 (2017)
10. Cao, H., Lin, T., Li, Y., Zhang, H.: Stock Price Pattern Prediction Based on Complex Network and Machine Learning. Complexity **2019** (2019)
11. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21-27 (1967). doi: 10.1109/TIT.1967.1053964
12. Keller, J. M., Gray, M. R., Givens, J. A.: A Fuzzy K-Nearest Neighbor Algorithm. IEEE Transactions on Systems, Man, and Cybernetics **15**(4), 580-585 (1985). doi: 10.1109/TSMC.1985.6313426
13. Price, K., Storn, R. M., Lampinen, J. A.: Differential evolution - a practical approach to global optimization, Springer-Verlag Berlin Heidelberg (2005)
14. Yang, F., Chen, Z., Li, J., Tang, L.: A novel hybrid stock selection method with stock prediction. Applied Soft Computing Journal **142**, 820-831 (2019)

15. Khushaba, R. N., Al-Ani, A., Al-Jumaily, A.: Feature subset selection using differential evolution and a statistical repair mechanism. Expert System with Applications **38**, 11515–11526 (2011). doi: 10.1016/j.eswa.2011.03.028
16. Bisoi, R., Dash, P. K., Parida, A. K.: Hybrid Variational Mode Decomposition and evolutionary robust kernel extreme learning machine for stock price and movement prediction on daily basis. Applied Soft Computing **74**, 652-676 (2019)
17. Yang, F., Chen, Z., Li, J., Tang, L.: A novel hybrid stock selection method with stock prediction. Applied Soft Computing **80**, 820-831 (2019)
18. Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., Mosavi, A.: Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis. IEEE Access, **8**, 150199–150212 (2020). doi: 10.1109/ACCESS.2020.3015966
19. Patel, J., Shah, S., Thakkar, P., Kotecha, K.: Predicting stock market index using fusion of machine learning techniques. Expert System with Applications, **42**(4), 2162–2172 (2015). doi: 10.1016/j.eswa.2014.10.031
20. Yahoo Finance, https://finance.yahoo.com/. Last accessed 22 Oct 2020
21. FRED Economic Data, https://fred.stlouisfed.org. Last accessed 25 Oct 2020
22. Kumbure, M. M., Lohrmann, C., Luukka, P., Porras, J.: Machine Learning Techniques and Data for Stock Market Forecasting: A Literature Review. Expert System with Applications, submitted (2021)
23. Kumbure, M. M., Luukka, P., Collan, M.: A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean. Pattern Recognition Letters **140**, 172-178 (2020). doi: 10.1016/j.patrec.2020.10.005
24. Teixeira, L. A., De Oliveira, A. L. I.: A method for automatic stock trading combining technical analysis and nearest neighbor classification. Expert Systems with Applications, **37**(10), 6885–6890 (2010). doi: 10.1016/j.eswa.2010.03.033

Publication III

Mailagaha-Kumbure, M., Luukka, P., and Collan, M.
A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean

Reprinted with permission from
Pattern Recognition Letters
Vol. 140, pp. 172–178, 2020.

Pattern Recognition Letters 140 (2020) 172–178

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A new fuzzy *k*-nearest neighbor classifier based on the Bonferroni mean

Mahinda Mailagaha Kumbure*, Pasi Luukka, Mikael Collan

School of Business and Management, LUT University, Yliopistonkatu 34, Lappeenranta 53850, Finland

ARTICLE INFO

Article history:
Received 24 October 2019
Revised 10 May 2020
Accepted 10 October 2020
Available online 11 October 2020

Keywords:
Bonferroni mean
Classification
Fuzzy *k*-nearest neighbor
Performance measures
Local means

ABSTRACT

We present a new generalized version of the fuzzy *k*-nearest neighbor (FKNN) classifier that uses local mean vectors and utilizes the Bonferroni mean. We call the proposed new method Bonferroni-mean based fuzzy *k*-nearest neighbor (BM-FKNN) classifier. The BM-FKNN classifier can be easily fitted for various contexts and applications, because the parametric Bonferroni mean allows for problem-based parameter value fitting. The BM-FKNN classifier can perform well also in situations where clear imbalances in class distributions of data are found. The performance of the proposed classifier is tested with six real-world data sets and with one artificial data set. The results are benchmarked with classification results obtained with the classical *k*-nearest neighbor-, the local mean-based *k*-nearest neighbor-, the fuzzy *k*-nearest neighbor- and other three selected classifiers. In addition to this, an enhancement of the local mean-based *k*-nearest neighbor classifier by using the Bonferroni means is also proposed and tested. The results show that the proposed new BM-FKNN classifier has the potential to outperform the benchmarks in classification accuracy and confirm the usefulness of using the Bonferroni mean in the learning part of classifiers.

1. Introduction

In this paper, we focus on the *k*-nearest neighbor (KNN) classification method and its generalizations. The objective of classification (algorithms) is to identify the class to which a new unclassified object or sample belongs to. In supervised machine-learning based algorithms the classification is done based on previous training of the algorithm with pre-classified data. The KNN algorithm introduced in [1] is a well-known supervised machine-learning based classification technique that is used in a wide range of applications and is one of the most used methods in classification today. The KNN classifier confronts the classification problem by first measuring the similarity (distance) between a new to-be-classified sample and training samples, to observe the *k* nearest neighbors for the new sample, and then determines the membership of the new sample to the class that has the largest number of neighbors with the new sample [2].

The performance of the KNN classifier is generally good, however, it is well known that the prediction accuracy of the method can be negatively influenced by outliers, which are likely to distort the class-distribution [3]. To deal with this problem, a local mean-based *k*-nearest neighbor (LM-KNN) classifier was introduced in [4]. The LM-KNN variant utilizes the local mean vectors for each class to classify a query sample to a particular class. The LM-KNN algorithm first finds the local mean vectors in each class in terms of all *k* nearest neighbors and then allocates the query sample to the class represented by the local mean vector that has the lowest Euclidean distance from the query sample [5,6]. The robustness and the simplicity of the LM-KNN algorithm has invited researchers to develop a variety of enhanced method variants (see examples in [2,6–10]) and to construct variant-based classification systems [11].

One propellant for the development of new KNN variants has been the observation that the original method has weaknesses. For example, in the original KNN algorithm, the already classified samples are assumed to have the same importance in the classification process of a new sample [12]. This simplification can harm the classification performance especially in situations, where class distributions are not in balance [13]. Another difficulty with the KNN model is that once the new sample is allocated to a particular class, the "strength" of membership in the class of the classified sample is not considered [14]. To remedy these problems, Keller [14] applied idea of including membership degrees [15] in the KNN approach, to produce a fuzzy version of the algorithm.

* Corresponding author.
E-mail addresses: mahinda.mailagaha.kumbure@lut.fi, mahinda.mailagaha.kumbure@student.lut.fi (M. Mailagaha Kumbure), pasi.luukka@lut.fi (P. Luukka), mikael.collan@lut.fi (M. Collan).

https://doi.org/10.1016/j.patrec.2020.10.005
0167-8655/

Consequently, the fuzzy k-nearest neighbor (FKNN) classifier was created and is one of the most popular directions of the KNN developments. The FKNN technique performs the classification by introducing membership degrees to classes, while dealing with the uncertainty in the data. In this study, we extend the FKNN classifier further, by utilizing local mean vectors, which are formed by using the known classes of k nearest neighbor sample vectors. To generalize these local mean vectors, the Bonferroni mean operator is used and the resulting local Bonferroni mean vectors are used to measure the similarity of the new sample to the classes.

KNN is based on the majority voting principle, where the class of a new sample is based on nearest neighbors and their majority class. In the case that a data set is clearly imbalanced an observed drawback of majority voting principle is that the classified samples of the class or classes with a large number of samples tend to dominate the prediction of the new sample simply due to the fact that they often are more numerous among the k nearest neighbors [16]. A way to overcome this drawback is to use local means calculated from the classes that are represented within the nearest neighbors instead. Class assignment is then done based on the closest local mean vector rather than based on the number of nearest neighbors. In this way the classes with the highest number of samples will not have such domination over the less numerous classes. The FKNN bases the classification on the most frequent class and also the distance of the unclassified sample to the nearest neighbors. The distance that can be interpreted as imprecision with regards to similarity of individual samples also affects classification. Averaging operators also are able to overcome problems with "individual imprecision" - this can be understood also as the "wisdom of the crowd" and the first one to discuss this issue was Aristotle [17]. Later Francis Galton made this notion popular by his famous example of a country-fair contest of weight estimation [18]. Based on these precursors one can expect that using local means should have a better predictive power than individuals alone.

The Bonferroni mean is an aggregation operator that was originally introduced in [19] and further developments were discussed, e.g., in [20,21]. It can be defined as a function of means and it has been used as a very useful indicator in many applications [see [22,23]]) due to its capability to perceive inter-relationships and to allow multiple comparisons between input arguments [24].

Some previous studies have noted that using the arithmetic mean is not producing optimal results with classifiers, instead performance could be improved by using alternative averaging operators, for example, generalized means [25], ordered weighted means (OWA) [26], and harmonic mean [6]. We note that as the generalized mean is a special case of the Bonferroni mean and results gained with generalized mean are at least as good as with arithmetic mean and often better, we can expect that results gained using Bonferroni mean are at least as good as with generalized mean and in some cases better. As the Bonferroni mean operator is applied to compute the class-representative local-mean vectors, one must be aware of the possibility to optimize the parameters to fit the context (particular data sets). Changing the parameters of the Bonferroni mean allows us to find good (optimal) parameter values, which will enhance the classification accuracy. By altering the parameters, one can "choose" several well-known means through the Bonferroni mean operator, such as the geometric, arithmetic, quadratic, and power means.

We study the performance of the proposed variant by using both artificial and real-life data sets containing binary and multiclass classification problems. To compare the performance of the proposed BM-FKNN method we benchmark its performance with the performances of FKNN, LM-KNN, KNN, SVM, NB, and the similarity classifier. In addition to this, we also investigate the classification performance of an improved variant of the LM-KNN classier that uses the Bonferroni mean - this is the second new variant proposed in this research. To evaluate the performance we use accuracy, sensitivity, and specificity as our performance measures. Besides this we also test whether differences between the classification accuracy of the BM-FKNN and the benchmarks is statistically significant.

2. *K*-nearest neighbor classifier variants and the Bonferroni mean

In this section, we briefly present the theoretical underpinnings of the KNN, LM-KNN, FKNN classifiers, and the Bonferroni mean operator.

2.1. K-nearest neighbor, fuzzy KNN and local mean based k*-nearest neigbor classifiers*

A formal definition of the KNN method is presented below.

Let X $(x_1, x_2, \ldots, x_N)$ be a training set, formed by N samples, and C $(\omega_1, \omega_2, \ldots, \omega_C)$ classes (that is, $X = \{x_j, c_j\}_{j=1}^{N}$, where $c_j \in C$). Each sample x_j $(x_j^1, x_j^2, \ldots, x_j^S, x_{c_j})$ contains S features. If a new query sample y is given, then it is assigned into a class (ω^*) correctly by using the following steps:

1 Choose the number of k nearest neighbors ($1 \leq k \leq N$) to the new sample
2 Compute the Euclidean distances from y to x_j for all j. Also other distance measures can be used.
3 Find the set of k nearest neighbors from the X by using sorted distances in an ascending order.
4 Identify the classes represented by the k nearest neighbors.
5 Classify y into the class to which the largest number of k nearest neighbors belong to.

LM-KNN algorithm is a simple and robust extension of the KNN method [4]. In this method, a local mean vector of k nearest neighbors in each class is used to assign the correct class for the query sample. The process of the LM-KNN algorithm can be summarized as follows:

1 Find the k nearest neighbors from the training set X for each class ω_i by using the Euclidean distance in an ascending order.
2 Compute the local mean vector for each class using the k nearest neighbors found in the step 1.
3 Assign y into the class in which the local mean vector has the minimum Euclidean distance from y.

Underlying idea in the FKNN method is that a membership degree to each class is assigned to the new query sample and the highest membership degree dominates the decision about classification [14]. Membership degree indicates the proportion to which the query sample belongs to each one of the available classes. These membership degrees are weighted by the inverse of the distance of the query sample to its k nearest neighbors in the membership function. Along with this, a fuzzy strength parameter m is

employed to provide the relative importance to the distance to be weighted, when determining the contribution of the neighbors to the membership degree. The assigned membership degree of the query sample y in each class i is labeled by the k nearest neighbors and is measured as follows:

$$u_i(y) = \frac{\sum_{j=1}^{k} u_{ij}(1/\left\|y - x_j\right\|^{2/(m-1)})}{\sum_{j=1}^{k}(1/\left\|y - x_j\right\|^{2/(m-1)})} \quad (1)$$

where, u_{ij} is the membership of the jth sample in the ith class of the training set and $m \in (1, +\infty)$ ($m = 2$ is often used).

2.2. The Bonferroni mean operator

The Bonferroni aggregation operator was introduced by Bonferroni [19] in the1950's and later extended by other researchers (see [20,27–29], and [30]). The Bonferroni mean consists of two parts, outer and inner part. Each argument of the outer part is the product of one argument and the average of all the other remaining inner arguments, this combination is what makes it a unique in terms of aggregation, [27]. The Bonferroni mean is defined as:

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n), x_i \in [0, 1] \ \forall i \in \mathbb{N}$ be a vector with at least one $x_i \neq 0 \ \forall \ i = 1, 2, \ldots, n$ and $\alpha, \ q \geq 0$ be parameters. The general Bonferroni mean of x_i is defined by Bonferroni [19]:

$$B^{p,q}(X) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p \left(\frac{1}{n-1}\sum_{i,j=1, j\neq i}^{n} x_j^q\right)\right)^{\frac{1}{p+q}} \quad (2)$$

As an averaging operator Bonferroni mean satisfies all necessary axioms (see [20]) that an averaging operator is typically required to satisfy.

3. Proposed fuzzy *k*-nearest neighbor classifier, based on Bonferroni mean vectors

By adding a computation of the local Bonferroni mean vectors into the learning (training) part of the FKNN algorithm, we introduce the new BM-FKNN classifier. As in the FKNN method, the BM-FKNN classifier starts with the estimation of the distances from the query sample y to the labeled samples $\{x_j, c_j\}_{j=1}^{N}$ and the set of k nearest neighbors $nn^k(y)$ is observed. The idea is then to group the k nearest neighbors into sub-samples based on the classes they belong to. These sub-samples representing each class are used in the calculation of the Bonferroni mean vectors. That is, if the $nn^k(y)$ is $\{x_j, c_j\}_{j=1}^{k}$ and $c_j \in (\omega_1, \omega_2, \ldots \omega_C)$, then the local Bonferroni mean vectors with the corresponding classes are $\{B_r, \omega_r\}_{r=1}^{t}$, $1 \leq t \leq C$. This also implies that the number of local mean vectors relies on the number of classes that appear in the set of k nearest neighbors. Then the Euclidean distances (d_{EUC}) between the query sample y and the local Bonferroni mean vectors are computed. These distances $d_{EUC}(y, \{B_r\}_{r=1}^{t})$ are used to measure the membership degrees of the query sample with the classes the mean vectors represent $\{\omega_r\}_{r=1}^{t}$ by using the Eq. (1). Finally, the query sample y is classified to the class ω^* with which the sample has the highest membership degree with.

The pseudo code for the BM-FKNN algorithm is summarized as in Figure Algorithm 1:

The proposed method uses the local sub-samples to create local mean vectors for all classes that are represented by the k nearest neighbors. In other words, in BM-FKNN the locally created representative vectors for each class, well-positioned to perceive the class-information, are used instead of comparing the query sample directly to the original k nearest neighbors. The class imbalance-problems which have found to be difficult to original KNN, due to domination of majority class, can this way be overcomed by using the local means. Moreover, problems that appear when using imprecise data in situations, where the samples from different classes are very close to each other [31], can also be remedied.

Selection of the k value (number of nearest neighbors used) has typically a critical role in classification accuracy. A very low k may produce inadequate classification results, while a too high k may cause outliers to affect the classification [5]. In connection with the proposed method, the k values selected can be quite high, because this allows the method to capture larger class-representative sub-samples and to create more accurate local Bonferroni mean vectors.

3.1. LM-KNN classifier with Bonferroni means

In addition to the main contribution of introducing a new BM-FKNN classifier, we also investigated how using the Bonferroni mean influences the performance of the LM-KNN classifier, specifically the application to the computation of the local mean vectors. In other words, we present and test a new LM-KNN classifier variant with Bonferroni means. For the purpose of simplification, we address this method as BM-KNN in the following sections.

4. Data sets and testing methodology

This section briefly introduces the used data sets and presents the testing methodology of the proposed new methods.

4.1. Artificial data with imbalance rate modifications

In most of the classification problems, we have to deal with imbalanced classes that is, the number of samples per class is not the same or even similar [32]. Typically imbalance is defined as a ratio between the number of samples in the larger class and the smaller class(es) [33]. As already discussed, this can be a problem for the classical KNN and means that the more frequently present class may tend to dominate the prediction of the new samples, because they are often more common among the k nearest neighbors. Because of this we also test how imbalance between classes affects the performance of BM-FKNN and the benchmarks. The testing data included two classes: class 1 $\sim \mathcal{N}(9, 4^2)$ with 10 features and a sample size of 100, and class 2 $\sim \mathcal{N}(10, 6^2)$ with 10 features and a sample size of n that was variable from the set (100, 90, 80, 70, 60, 50, 40, 30, 20, 10). In this way, data with the imbalance ratio (1/1,1/0.9,....,1/0.1) was adjusted in ascending order and the tested classifiers' performance measured for each case.

4.2. Real-world data

In addition to the artificial data, this study uses also six real-world data sets: Car data, Vehicle data, Ionosphere data, Mammogram, Wine data, and Page Blocks data, all of which are freely available at the KEEL repository [34] and the UCI Machine Learning repository [35]. Vehicle, Ionosphere, and Mammogram data represent binary class problems and Car, Wine, and Page Blocks data multi-class problems. The entry errors and quality issues on the data were studied and fixed before using them. The characteristics of each of data set are summarized in Table 1.

Table 1
Information on the data sets used.

Data set	Database	Instances	Features	Classes
Car	KEEL	1728	6	4
Vehicle	KEEL	846	18	2
Ionosphere	UCI	351	34	2
Mammogram	UCI	961	6	2
Wine	UCI	178	13	3
Page Blocks	KEEL	548	10	5

4.3. Performance measures used

Next, we shortly go through the performance metrics we used in this study. Since we have multi-class classification problems in our study, we also shortly present their multi-class analogs. To evaluate classification methods the most common metric used is *accuracy* [2,12,36] as a percentage of correct predictions with respect to the total number of original tested samples. Reporting accuracy results alone is often not enough to conclude that that the performance of a classifier is useful for a given task. Hence, additional performance measures such as the *sensitivity* and the *specificity* are also needed to more comprehensively evaluate the performance of classifiers. Here we use all three measures to better understand the "goodness" of the proposed classifiers and their benchmarks.

4.3.1. Binary-class problem

In the binary classification, there are only two classes, one is a positive (P) and other is a negative (N) class. There are four possible outcomes from the classification model such as true positive (TP), true negative (TN), false positive (FP), and false negative (FN), and T and F are shaped by predicted class and P and N are shaped by the actual class. Using these metrics, the performance measures in the classification are defined as $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, $Sensitivity = \frac{TP}{TP+FN}$ and $Specificity = \frac{TN}{FP+TN}$.

4.3.2. Multi-class problems

Multi-class classification refers to classification tasks, where there are more than two classes. In this research, we utilize the performance measures computation for the multi-class problems proposed in [37]. General notation for performance metrics for the multi-class classification is defined in the following way:

Suppose a confusion matrix with C ($>$ 2) classes, represented by $\{a_{l,m}\}_{l=1,m=1}^{C}$, and $a_{l,m}$ is an element of a row l and a column m in a matrix. When $l = m$, $a_{l,m}$ indicates the number of samples classified correctly to the correspond class and when $l \neq m$ indicates the number of misclassified samples of class ω_l as class ω_m. Then the number of true positives, true negatives, false positives, and false negatives for each class ω_i ($i \in C$) can be measured as follows: $TP(\omega_i) = a_{i,i}, \ \forall\, i \in C$, $TN(\omega_i) = \sum_{l=1}^{C}\sum_{m=1l,m\neq i}^{C}(a_{l,m})$, $FP(\omega_i) = \sum_{l=1}^{C}(a_{l,i}) - TP(\omega_i)$ and $FN(\omega_i) = \sum_{m=1}^{C}(a_{i,m}) - TP(\omega_i)$. The accuracy, sensitivity, and specificity are computed for each class using above measures. The averages of these measures for all classes are considered as the final performance measures, in vein with [38].

4.4. Experimental setting and evaluation

In each selected data set (including artificial data set), the data sets were separated into a 40% training set, a 40% validation set, and a 20% testing set. The stratified random sampling technique was used in the sampling to ensure that class proportions in each of the divided sets are the same as they are in the whole data set. The hold out method was used for the cross-validation, in which 30 splits of the training and validation sets were randomly generated (30-fold cross validation).

We considered the number of neighbors k from the set {1, 2, . . . , 25}. This was due to our assumption that the performance of the proposed BM-FKNN method would increase (and relatively increase), when the value of k increases. Pan et al [6] had provided evidence in favor of this assumption by showing that a multi-local means based k harmonic nearest neighbor classifier achieved better performance in the classification with high k values. The values for the parameters p and q of the Bonferroni mean were chosen from the range $\{0, 1, \ldots, 9, 10\}$. We first optimized the parameter values with the training & validation step and the gained optimal values were then used to test the performance of the new method with the testing sample. Following the recommendations in [2,14], the fuzzy strength parameter m was kept at $m = 2$ for both BM-FKNN and FKNN classifiers. The results are presented in terms of mean values for all performance measures.

To validate the performance of the proposed new methods, we compare the classification results of BM-FKNN and BM-KNN classifiers with the original KNN, FKNN, and LM-KNN and also with support vector machines (SVM) [39], naive Bayes classifier (NB) [40], and similarity based classifier (Similarity) [41]. The same training and validation samples were used for these classifiers for all data sets and their classification performance was registered for the optimized model with the test samples. We carried out the comparative test essentially on the real-world data sets in terms of the accuracy and other performance measures discussed above.

A paired t-test, in vein with [36] was also performed to reveal whether the performance difference of the proposed methods is statistically significant when compared to the benchmarks, a 0.05 level of significance was used. For this analysis, the samples from the hold out method (size of 1 $\times$ 30) were considered for each classifier, when the optimal parameters were used. In addition, the confidence interval and variances were calculated.

5. Results and discussion

In this section we first present the findings obtained for the artificial case that was generated to investigate the difference between the new proposed classifiers and the benchmark classifiers. This is followed by a presentation of the results for the real-world data sets for the training & validation step and the testing step separately.

5.1. Results for the artificial data

The artificial data was used to test the class imbalance. For this data we present a mean classification accuracy plot, taken in the testing phase for the proposed two classifiers and the three KNN-based benchmarks. From Fig. 1 one can see how the mean classification accuracy develops with respect to the imbalance ratio. We point out that the performance of both the new proposed method is the same.

In Fig. 1, imbalance ratio is presented as the sample percentage of the larger class in terms of the smaller class. For example, "1/0.3" denotes a ratio that class 1 has the sample size of 100 and class 2 has the sample size of 30. It is evident from the Fig. 1 that classifier performance is at its best, when one class has the lowest number of instances in comparison to other class (in binary class problems). A gradual increase in the mean accuracy for all classifiers can be seen with the increase of the imbalance ratio. It is clearly visible that over all imbalance ratios the BM-FKNN and BM-KNN classifiers have achieved higher accuracies than the benchmarks, the difference is most pronounced with the high imbalances. In general, this result indicates that the proposed methods

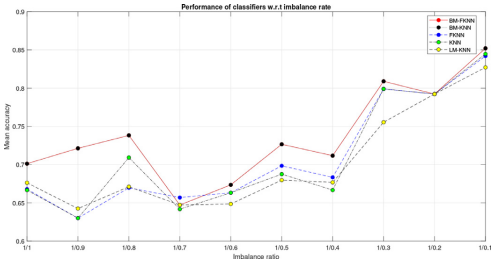


Fig. 1. Classifier performance with respect to imbalance ratio of classes.

Table 2
BM-FKNN and the BM-KNN classifier results in the validation part.

Data set	Mean Acc.	Sensitivity	Specificity	Opt. parameters
Car	0.9271	0.8079	0.9637	$k = 3, p = 1, q = 1$
Vehicle	0.9340	0.8557	0.9591	$k = 4, p = 3, q = 1$
Ionosphere	0.8775	0.8611	0.9245	$k = 7, p = 1, q = 0$
Mamm	0.7939	0.7901	0.7984	$k = 21, p = 2, q = 1$
Wine	0.7414	0.7388	0.8730	$k = 25, p = 2, q = 2$
Page Blocks	0.9358	0.9775	0.8679	$k = 3, p = 2, q = 2$

are less sensitive to the class imbalance problem than the benchmarks.

5.2. Results for the real-world data

The obtained results for accuracy, sensitivity, and specificity as well as for the difference between the classification accuracies of the new methods and the benchmarks are presented.

5.2.1. Performance with the training & validation data

The parameters of the proposed new approaches and the benchmarks were optimized with the training and validation steps by using the holdout method for ensuring sample similarity. A thirty-fold cross validation was performed with the data. The obtained results for performance measures and the optimal parameter values are presented as an example for the proposed BM-FKNN classifier in Table 2. In the table, "Mean Acc." indicates the mean accuracy from the 30 sample folds gained by using the optimal parameters. Sensitivity and specificity results are also reported with mean values accordingly. The highest mean accuracy was used to determine the optimal parameter values for p, q and k.

From the Table 2, we can see that the highest mean accuracy was reached with settings $p \in \{1, 2, 3\}$ and $q \in \{0, 1, 2\}$ for all data sets considered.

Also the sensitivity and specificity values indicate reasonable results. In addition, it is also apparent that for all cases, the specificity is higher than the sensitivity. Fig. 2 illustrates the impact of the different combinations of the parameters p and q (with the optimal k) on the selected performance measures for the Vehicle data in the training & validation step.

5.2.2. Performance with the test samples

In this sub-section, we present the classification results of the classifiers with the testing data samples, which were initially separated from the original data sets. We also include the comparison to other classifiers. Optimized parameter values and saved training samples in the validation step were used to test the classifiers with the previously unused test samples. Table 3 summarizes the results for mean classification accuracy, mean sensitivity, mean specificity, variance, and confidence interval (CI) obtained for the proposed

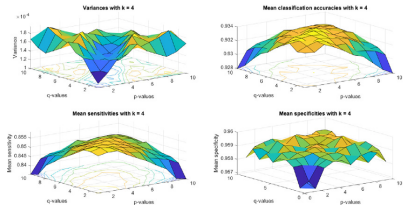


Fig. 2. Variance and performance measures for different parameter combinations (p, q) with Vehicle data for the BM-FKNN.

Algorithm 1 BM-FKNN.

Input: $\{x_j, c_j\}_{j=1}^{N}$ (labeled set), y (query sample), $k(1 \le k \le N)$, p, q $(p, q > 0)$
Output: The class label for y
Begin
for $j = 1$ to N **do**
  Compute $d_{EUC}(y, x_j)$ from y to x_j
  if $j < k$ **then**
    Add x_j to $nn^k(y)$
  else if x_j is closer to y than any of neighbors in $nn^k(y)$ **then**
    Drop the farthest neighbor from the set $nn^k(y)$ and add x_j
  end if
end for
for $r = 1$ to t **do**
  Find B_r in the set $nn^k(y)$ using using equation (2) and set the correspond class ω_r.
  Compute $d_{EUC}(y, B_r)$ from y to B_r.
  Assign membership $u_r(y)$ to ω_r in terms of weighed distanceaccording to:

$$u_r(y) = \frac{\sum_{r=1}^{t} u_{rr}(1/\|y - B_r\|^{2/(m-1)})}{\sum_{r=1}^{t}(1/\|y - B_r\|^{2/(m-1)})} \tag{3}$$

  where u_{rr} is 1 for known class and 0 for other classes.
end for
return ω^* (predicted class that has the highest membership degree) for y, $\omega^* \in (\omega_1, \omega_2, \ldots, \omega_t)$.
End

BM-FKNN and BM-KNN methods and for the benchmarks over all considered data sets. The results for the BM-FKNN and the BM-KNN are the same and they are presented in the same column.

The results from the test sets show that the proposed classifiers have high classification accuracy compared to the benchmarks.

From Table 3 one can observe that the proposed new methods outperform all benchmarks with two data sets and that the performance is second-best with three data sets. The mean sensitivity and specificity remains high for all data sets. Besides this, interestingly BM-KNN classifier obtained the exact results which were also obtained with BM-FKNN for all data sets. This reveals that the influence of Bonferroni mean inside the learning part of the classifier has dominating effect compared to membership degree computation in fuzzy KNN. In particular BM-KNN and BM-KNN classifiers significantly improved the accuracy compared to KNN, FKNN and LM-KNN methods. This indicates that introducing the concepts of Bonferroni mean local vectors as nearest representatives instead of k nearest samples one can generate more reasonable class repre-

Table 3
Classification results with the testing samples.

Data set	Measure	BM-FKNN/ BM-KNN	FKNN	LM-KNN	KNN	SVM	NB	Similarity classifier
Car	Mean Accuracy	**0.9292**	0.8905	0.8956	0.8845	0.8506	0.8158	0.6984
	Variance	1.37E-04	1.56E-04	1.98E-04	1.49E-04	1.67E-04	2.50E-04	2.29E-04
	CI	[0.9237 0.9347]	[0.8846 0.8963]	[0.8890 0.9022]	[0.8795 0.8895]	[0.8445 0.8566]	[0.8083 0.8232]	[0.6913 0.7055]
	Mean Sensitivity	0.8251	0.658	0.6345	0.6173	0.7	0.4005	0.6599
	Mean Specificity	0.9659	0.9291	0.9197	0.9142	0.9133	0.8946	0.9017
Vehicle	Mean Accuracy	0.9556	0.9456	0.9371	0.9456	**0.9562**	0.7015	0.6988
	Variance	1.38E-04	7.96E-05	2.77E-04	7.96E-05	8.01E-04	2.27E-04	8.81E-05
	CI	[0.9501 0.9611]	[0.9414 0.9497]	[0.9309 0.9434]	[0.9414 0.9497]	[0.9430 0.9695]	[0.6944 0.7085]	[0.6944 0.7032]
	Mean Sensitivity	0.8852	0.879	0.8695	0.879	0.9077	0.4351	0.4315
	Mean Specificity	0.9796	0.967	0.9594	0.967	0.9714	0.9444	0.9359
Ionosphere	Mean Accuracy	0.8914	0.8529	0.8914	0.8529	0.8906	**0.9164**	0.8621
	Variance	4.60E-04	0.0013	4.60E-04	0.0013	4.83E-04	9.30E-04	0.0026
	CI	[0.8814 0.9015]	[0.8362 0.8695]	[0.8814 0.9015]	[0.8362 0.8695]	[0.8813 0.9009]	[0.9022 0.9307]	[0.8382 0.8861]
	Mean Sensitivity	0.8562	0.824	0.8562	0.824	0.8601	0.9446	0.8737
	Mean Specificity	0.9523	0.9541	0.9523	0.9541	0.9649	0.8762	0.8596
Mammogram	Mean Accuracy	**0.7927**	0.7844	0.7909	0.7833	0.7906	0.7844	0.7789
	Variance	1.16E-04	1.87E-04	1.18E-04	8.68E-05	1.16E-04	1.15E-04	2.99E-05
	CI	[0.7877 0.7977]	[0.7780 0.7908]	[0.7858 0.7960]	[0.7790 0.7877]	[0.7856 0.7957]	[0.7793 0.7894]	[0.7763 0.7815]
	Mean Sensitivity	0.7528	0.7275	0.7437	0.7303	0.7819	0.7457	0.7054
	Mean Specificity	0.8343	0.8535	0.8434	0.8456	0.8058	0.827	0.885
Wine	Mean Accuracy	0.8306	0.8097	0.8306	0.8069	0.8833	**0.9722**	0.9681
	Variance	0.002	0.0028	0.002	0.0028	0.0022	2.08E-31	1.04E-04
	CI	[0.8095 0.8516]	[0.7850 0.8344]	[0.8095 0.8516]	[0.7822 0.8317]	[0.8616 0.9051]	[0.9722 0.9722]	[0.9633 0.9728]
	Mean Sensitivity	0.811	0.7897	0.811	0.7897	0.8712	0.9722	0.9724
	Mean Specificity	0.9105	0.9012	0.9105	0.9012	0.9379	0.9848	0.9846
Page Blocks	Mean Accuracy	0.9255	0.92	0.915	0.9191	0.8918	**0.9259**	0.6586
	Variance	3.74E-05	2.74E-05	3.72E-05	2.56E-04	2.52E-05	3.07E-04	0.0018
	CI	[0.9219 0.9290]	[0.9165 0.9235]	[0.9121 0.9179]	[0.9165 0.9235]	[0.8895 0.8942]	[0.9211 0.9307]	[0.7185 0.7579]
	Mean Sensitivity	0.9597	0.9619	0.9597	0.9615	0.9337	0.989	0.9954
	Mean Specificity	1	1	1	1	NaN	0.7366	0.4735
	Average (overall)	**0.8875**	0.8672	0.8767	0.8654	0.8772	0.8527	0.7775

Table 4
Results of the *t*-test on the performance of the proposed methods vs. the six benchmarks on the test sample data.

Data set	Paired-t with BM-FKNN / BM-KNN	P-value	test-statistic
Car	FKNN	2.4770e-12	significant
	LM-KNN	6.30E-10	significant
	KNN	3.88E-14	significant
	SVM	6.75E-22	significant
	NB	1.12E-25	significant
	Similarity classifier	1.57E-37	significant
Vehicle	FKNN	0.0042	significant
	LM-KNN	1.24E-05	significant
	KNN	0.0042	significant
	SVM	0.9317	not significant
	NB	5.63E-41	significant
	Similarity classifier	4.30E-42	significant
Ionosphere	FKNN	3.48E-04	significant
	LM-KNN	1	not significant
	KNN	3.48E-04	significant
	SVM	0.0025	significant
	NB	5.62E-07	significant
	Similarity classifier	0.4808	not significant
Mammogram	FKNN	0.0387	significant
	LM-KNN	0.597	not significant
	KNN	0.0055	significant
	SVM	0.5443	not significant
	NB	0.019	significant
	Similarity classifier	9.38E-06	significant
Wine	FKNN	0.0871	not significant
	LM-KNN	1	not significant
	KNN	0.0839	not significant
	SVM	1.05E-22	significant
	NB	3.34E-34	significant
	Similarity classifier	2.31E-32	significant
Page Blocks	FKNN	0.0096	significant
	LM-KNN	3.77E-06	significant
	KNN	1.02E-04	significant
	SVM	6.56E-20	significant
	NB	0.6917	not significant
	Similarity classifier	1.49E-21	significant

sentative vectors. Regarding SVM, NB and similarity classifiers even though in some cases they are able to achieve little higher accuracies BM-FKNN and BM-KNN classifiers still outperform them on majority of the data sets.

Moreover, it seems that the performance of the classification has been significantly increased by using the higher values for the parameter k with the proposed methods. Obviously, this is interesting since the low values of k are performing better for the benchmarks and it is also confirmed by showing that the KNN classifier worked well with $k = 1$ with two data sets considered. This finding is in agreement with previous findings by Derrac et al. in [2].

The preliminary conclusion that can be stated based on the results is that the proposed new classifiers outperform the KNN-based benchmarks and thus excels with data sets where KNN-based classification fits well.

Table 4 presents the paired *t*-test results for the BM-FKNN and BM-KNN and benchmark classifiers with the test samples. From the evidence on the table, it is visible that the BM-FKNN and BM-KNN methods have yielded statistically significantly higher classification accuracies in the cases where the accuracies produced were superior.

6. Conclusion

This paper introduced two new methods to the family of fuzzy k-nearest neighbor classifiers that are both developed by using the Bonferroni mean in the computation of local mean vectors, which are used in the classification of new query samples to know classes. The proposed BM-FKNN and BM-KNN methods differ from FKNN and LM-KNN methods in that they use the bonferroni mean in the computation of local mean vectors for the set of k nearest neighbors, where the difference with the FKNN is that no local mean vectors were previously calculated and with the LM-KNN mean operator was arithmetic mean.

To illustrate and study the performance of the proposed classifiers they were tested with an artificial data set and six real-world data sets. The obtained results show that the new methods can

give improved classification accuracy compared to the benchmarks used. Specifically it can be mentioned that the proposed new methods matched or outperformed all KNN-based benchmarks in all performance tests. The results were tested for statistical significance and it was found that the proposed methods had better classification accuracy than the benchmarks.

From the artificial data set experiment we found that the new methods are less sensitive to class imbalances, than its "original" counterparts. From the results with the real-world data, the most obvious finding to emerge for the new methods is that the best classification accuracy is achieved with a relatively high number (k) of nearest neighbors. This is reasonable, because when the sample size increases, the mean of the sample gets closer to a precise representation of the sample. However, we should note that due to the more complex calculations involved the execution of the proposed BM-FKNN method takes little more time than that of the benchmarks. Also finding a suitable parameters for Bonferroni mean requires a lot more classification runs since grid search is used and this takes time. In other words, computational complexity of the proposed approach is rather high in comparison to the classical methods.

Moreover, this study offers some insight into our understanding of the Bonferroni means and its usage in the classifiers and learning algorithms. In fact, further research directions include testing the effect of combining Bonferroni means together with other known variants of the KNN algorithm such as IV-KNN [2], kNN-TSC [42], and modified evidential KNN [10]. It also would be interesting to see how Bonferroni means can be employed in some other machine learning applications, e.g. in [43–46], where arithmetic mean has been extensively used.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by the Finnish Strategic Research Council project #313396, "Manufacturing 4.0".

References

[1] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27.
[2] J. Derrac, F. Chiclana, S. Garcia, F. Herrera, An interval valued k-nearest neighbors classifiers, in: in Proc. of the Int. Joint Conf. IFSA-EUSFLAT-2015, 2015.
[3] K. Fukunaga, Introduction to Statistical Pattern Recognition (2nd Ed.), Academic Press Prof., Inc., San Diego, CA, USA, 1990.
[4] Y. Mitania, Y. Hamamotob, A local mean-based nonparametric classifier, Pattern Recognit. Lett. 27 (2006) 1151–1159.
[5] J. Gou, Z. Yi, L. Du, T. Xiong, A local mean-based k-nearest centroid neighbor classifier, Comput. J. 55 (2012) 1058–1071.
[6] Z. Pan, Y. Wang, W. Ku, A new k-harmonic nearest neighbor classifier based on the multi-local means, Expert Syst. Appl. 67 (2017) 115–125.
[7] J. Chai, H. Liu, B. Chen, Z. Bao, Large margin nearest local mean classifier, Signal Process. 90 (2010) 236–248.
[8] G. Jianping, Z. Yongzhao, R. Yunbo, S. Xiangjun, W. Xiaoming, H. Wu, Improved pseudo nearest neighbor classification, IEEE Trans. Syst. 70 (2014) 361–375.
[9] M. Sumet, S. Xiangjun, G. Jiangping, N. Dejiao, A new nearest centroid neighbor classifier based on k local means using harmonic mean distance, Information 9 (2018) 234.
[10] T. Denœux, O. Kanjanatarakul, S. Sriboonchitta, A new evidential K-nearest neighbor rule based on contextual discounting with partially supervised learning, Int. J. Appr. Reason. 113 (2019) 287–302, doi:10.1016/j.ijar.2019.07.009.
[11] Z.G. Liu, Q. Pan, J. Dezert, G. Mercier, Hybrid classification system for uncertain data, IEEE Trans. Syst. Man Cybern.Syst. 47 (10) (2017) 2783–2790, doi:10.1109/TSMC.2016.2622247.
[12] F.C. Rhee, H. Hwang, An interval type-2 fuzzy k-nearest neighbor, in: Proc. of the 12th IEEE Int. Conf. on Fuzzy Systems-2003. FUZZ'03, 2003, pp. 802–807.
[13] M. Beckmann, N.F.F. Ebecken, B.S.L. Pires-de Lima, A KNN undersampling approach for data balancing, J. Intell. Learn. Syst.Appl. 7 (2015) 104–116.
[14] J.M. Keller, M.R. Gray, J.A. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Trans. Syst. 15 (1985) 580–585.
[15] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.
[16] D. Coomans, D.L. Massart, Alternative k-nearest neighbour rules in supervised pattern recognition : part 1. K-nearest neighbour classification by using alternative voting rules, Anal. Chim. Acta 136 (1982) 15–27.
[17] Aristotle, Politics, 4th century BC.
[18] F. Galton, Vox populi, Nature 75 (1949) 450–451.
[19] C. Bonferroni, Sulle medie multiple di potenze, Bolletino Math. Italiana 5 (1950) 267–270.
[20] R.R. Yager, On generalized Bonferroni mean operators for multi-criteria aggregation, Int. J. Approximate Reasoning 50 (2009) 1279–1286.
[21] S. Hongchun, S. Min, Generalized Bonferroni harmonic mean operators and their application to multiple attribute decision making, J. Comput. Inf. Syst. 8 (2012) 5717–5724.
[22] W. Guiwu, Z. Xiaofei, L. Rui, S. Hongjun, Uncertain linguistic Bonferroni mean operators and their application to multiple attribute decision making, Appl. Math. Model. 37 (2013) 5277–5285.
[23] O. Kurama, P. Luukka, M. Collan, A similarity classifier with Bonferroni mean operators, Advances in Fuzzy Systems (2016). ID 7173054
[24] B.M. Fabio, M.M. Jose, K. Janusz, Bonferroni means with distance measures and the adequacy coefficient in entrepreneurial group theory, Knowl. Based Syst. 111 (2016) 217–227.
[25] P. Luukka, T. Leppälampi, Similarity classifier with generalized mean applied to medical data, Comput. Biol. Med. 39 (2006) 1026–1040, doi:10.1016/j.compbiomed.2005.05.008.
[26] P. Luukka, O. Kurama, Similarity classifier with ordered weighted averaging operators, Expert Syst. Appl. 40 (2013) 995–1002, doi:10.1016/j.eswa.2012.08.014.
[27] G. Beliakov, S.H. Bustine, T. Calvo, A Practical Guide to Averaging Functions, Springer International Publishing, Switzerland, 2016.
[28] G. Beliakov, S. James, J. Mordelova, R.R. Yager, Generalized Bonferroni mean operators in multicriteria aggregation, Fuzzy Sets Syst. 161 (2010) 2227–2242.
[29] G. Beliakov, A. Pradera, T. Calvo, Aggregation Functions: A Guide for Practitioners, Springer-Verlag, Berlin, Heidelberg, 2007.
[30] R.R. Yager, On the dispersion measure of OWA operators, Inf. Sci. 179 (2009) 3908–3919.
[31] Z. Liu, P. Quan, D. Jean, A new belief-based k-nearest neighbor classification method, Pattern Recognit. 46 (2013) 834–844.
[32] S. Piri, D. Delen, T. Liu, A synthetic informative minority over-sampling (SIMO) algorithm leveraging support vector machine to enhance learning from imbalanced datasets, Decis. Support Syst. 106 (2018) 15–29.
[33] V. García, J.S. Sánchez, R.A. Mollineda, On the effectiveness of preprocessing methods when dealing with different levels of class imbalance, Knowl. Based Syst. 25 (1) (2012) 13–21.
[34] J. Alcala-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. Garc-a, L. Snchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2011) 255–287.
[35] D. Dheeru, E.K. Taniskidou, UCI machine learning repository, 2017.
[36] H.L. Chen, D.Y. Liu, B. Yang, J. Liu, G. Wang, S.J. Wang, An adaptive fuzzy k-nearest neighbor method based on parallel particle swarm optimization for bankruptcy prediction, in: Lecture Notes in Computer Science 6634 LNAI (PART 1), 2011, pp. 249–264.
[37] A. Tharwat, Classification assessment methods, Appl. Comput. Inf. xxx (2018) xxx–xxxx, doi:10.1016/j.aci.2018.08.003.
[38] C. Ferri, J. Hernandez-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognit. Lett. 30 (2009) 27–38.
[39] C. Cortes, V. Vapnik, Support-vector networks, Mach Learn 20 (1995) 273–297, doi:10.1023/A:1022627411411.
[40] D.D. Lewis, Naive (bayes) at forty: the independence assumption in information retrieval, in: Machine Learning: ECML-98, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 4–15.
[41] P. Luukka, K. Saastamoinen, V. Kononen, A classifier based on the maximal fuzzy similarity in the generalized Lukasiewicz-structure, in: 10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297), vol. 1, 2001, pp. 195–198.
[42] Y.H. Lee, C.P. Wei, T.H. Cheng, C.T. Yang, Nearest-neighbor-based approach to time-series classification, Decis. Support Syst. 53 (1) (2012) 207–217.
[43] H. Shi, S. Pan, J. Yang, C. Gong, Positive and unlabeled learning via loss decomposition and centroid estimation, in: IJCAI International Joint Conference on Artificial Intelligence 2018-July, 2018, pp. 2689–2695, doi:10.24963/ijcai.2018/373.
[44] Y.F. Li, J.T. Kwok, Z.H. Zhou, Semi-supervised learning using label mean, in: Proceedings of the 26th International Conference On Machine Learning, ICML 2009, 2009, pp. 633–640.
[45] C. Gong, H. Shi, T. Liu, C. Zhang, J. Yang, D. Tao, Loss decomposition and centroid estimation for positive and unlabeled learning, IEEE Trans. Pattern Anal. Mach. Intell. (2019), doi:10.1109/tpami.2019.2941684.
[46] F. Nie, J. Yuan, H. Huang, Optimal mean robust principal component analysis, in: 31st International Conference on Machine Learning, ICML 2014, 4, 2014, pp. 2755–2763.

Publication IV

Mailagaha-Kumbure, M., and Luukka, P.
A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance

Reprinted with permission from
Granular Computing
Vol. 7, pp. 657–671, 2022.

Granular Computing
https://doi.org/10.1007/s41066-021-00288-w

ORIGINAL PAPER

A generalized fuzzy *k*-nearest neighbor regression model based on Minkowski distance

Mahinda Mailagaha Kumbure[1] · Pasi Luukka[1]

Received: 30 June 2021 / Accepted: 5 September 2021

Abstract
The fuzzy k-nearest neighbor (FKNN) algorithm, one of the most well-known and effective supervised learning techniques, has often been used in data classification problems but rarely in regression settings. This paper introduces a new, more general fuzzy k-nearest neighbor regression model. Generalization is based on the usage of the Minkowski distance instead of the usual Euclidean distance. The Euclidean distance is often not the optimal choice for practical problems, and better results can be obtained by generalizing this. Using the Minkowski distance allows the proposed method to obtain more reasonable nearest neighbors to the target sample. Another key advantage of this method is that the nearest neighbors are weighted by fuzzy weights based on their similarity to the target sample, leading to the most accurate prediction through a weighted average. The performance of the proposed method is tested with eight real-world datasets from different fields and benchmarked to the *k*-nearest neighbor and three other state-of-the-art regression methods. The Manhattan distance- and Euclidean distance-based FKNNreg methods are also implemented, and the results are compared. The empirical results show that the proposed Minkowski distance-based fuzzy regression (Md-FKNNreg) method outperforms the benchmarks and can be a good algorithm for regression problems. In particular, the Md-FKNNreg model gave the significantly lowest overall average root mean square error (0.0769) of all other regression methods used. As a special case of the Minkowski distance, the Manhattan distance yielded the optimal conditions for Md-FKNNreg and achieved the best performance for most of the datasets.

Keywords Fuzzy *k*-nearest neighbor · Regression · Minkowski distance · Machine learning · Performance measures

1 Introduction

In machine learning, a regression problem refers to estimating a real-valued continuous response (output) based on the values of one or more input variables. By determining the relationships between output and input variables, a regression method numerically predicts a target value. In the literature, various regression techniques have been introduced for a wide range of machine learning problems. Among them, *k*-nearest neighbor regression (KNNreg) (Benedetti 1977; Stone 1977; Turner 1977) has become one of the most widely used regression techniques due to its simplicity and robustness (Buza et al. 2015). This method is an adapted version of the *k*-nearest neighbor (KNN) model that was initially introduced by Cover and Hart (1967) for applying classification problems. The main idea of the KNNreg is to predict the output value for a given test sample by averaging the output values of the nearest neighbor samples (Hu et al. 2014).

Though the KNN method has many significant advantages, it intuitively suffers from some weaknesses, for example, giving equal importance to all nearest neighbors (even if some of them are quite far from the test sample) in the classification process. To improve model and alleviate such issues, Keller et al. (1985) introduced the idea of using the degree of membership in the KNN method to propose its fuzzy version, called the fuzzy *k*-nearest neighbor (FKNN) classifier. Thanks to its capability of tackling uncertainty issues in the data, the FKNN model

✉ Mahinda Mailagaha Kumbure
mahinda.mailagaha.kumbure@lut.fi

Pasi Luukka
pasi.luukka@lut.fi

[1] School of Business and Management, LUT University, 53850 Lappeenranta, Finland

Published online: 25 September 2021

has proven promising for classification problems (Chen et al. 2013; Yu et al. 2002) compared to the classical KNN method. Although the FKNN classifier has received much attention in terms of classification, it has received less attention in the context of regression. This motivated us to establish the fuzzy *k*-nearest neighbor regression (FKNNreg) model in this research by modifying the original FKNN rule.

Typically, the distance metric is one of the main components of distance-based classifiers such as the KNN and FKNN methods (Rastin et al. 2021). Even though the Euclidean distance is the most common distance metric used in such methods to measure the similarity between two data samples, it is often not optimal for every problem domain (Cai et al. 2020; Nguyen et al. 2016). Several research papers have reported better results with a more general choice of distance metric (Chang et al. 2006; Dettmann et al. 2011; Jenicka and Suruliandi 2011; Kaski et al. 2001; Koloseni et al. 2012, 2013). Besides, the Euclidean distance has several drawbacks. For example, if two data samples have no feature values in common, they might have a shorter distance than the other sample pairs, including the same feature values (Shirkhorshidi et al. 2015). These facts encouraged us to examine the effectiveness of the Minkowski distance in the FKNN rule in the regression setting for low- and high-dimensional datasets.

The main goal of this study is to introduce the FKNNreg using the Minkowski distance metric and to examine its efficiency. The combination of the Minkowski distance metric and the FKNNreg has not been studied in the literature before. This led us to create the Minkowski distance-based fuzzy *k*-nearest neighbor regression (Md-FKNNreg) algorithm. A key advantage of this method is that the nearest neighbors are weighted by fuzzy weights considering their similarity to the test sample, leading to the most accurate prediction through a weighted average. Also, utilization of the Minkowski distance allows greater flexibility for obtaining more relevant neighboring samples close to the target sample.

Intuitively, most available regression models (e.g., multiple linear regression [MLR], least absolute shrinkage and selection operator [LASSO] regression) are based on assumptions regarding the distribution of the data. However, it is rarely confirmed that these assumptions apply to real-world problems. That is being said, an interesting fact about the KNNreg methods is that they do not explicitly make any assumptions about the underlying data (Yao and Ruzzo 2006) or model's components and simply use training data to make predictions. Another advantage is that they are, in general, relatively easy to implement and interpret and can potentially be applied even for non-linear problems (Hu et al. 2014). Moreover, support vector regression (SVR) is recognized as one of the well-known methods applied for non-linear regression problems. However, its utilization is restricted in various disciplines due to the difficulty of selecting suitable parameters for the model (Liu et al. 2013). In this regard, FKNNreg methods could be better alternatives in the regression context, and the proposed new KNNreg method is found to be significant for non-linear regression problems.

To study the performance of the proposed Md-FKNNreg model, we conducted an experiment using real-world data from various applications. We compared the regression performance of the proposed variant with the KNNreg, Lasso, SVR, and multiple linear regression models. In addition, the Manhattan distance-based fuzzy *k*-nearest neighbor regression (Man-FKNNreg) and Euclidean distance-based fuzzy *k*-nearest neighbor regression (Euc-FKNNreg) methods were also implemented, and the results were compared. To evaluate the regression performance, we used root mean square error (RMSE) and the coefficient of determination (R^2) values as the evaluation metrics. We also tested whether there was a statistically significant difference between the regression results for the Md-FKNNreg and baseline methods.

The main contributions of this paper can be summarized as follows:

(1) We propose a new regression approach based on the FKNN algorithm.
(2) We introduce the Minkowski distance into the nearest neighbors search in the proposed algorithm and investigate its efficiency and robustness.
(3) We demonstrate the performance of the proposed regression model on low- and high-dimensional real-world data coming from different domains.
(4) We analyze, compare, and benchmark the regression results of the proposed method with select well-known state-of-the-art regression methods.

The remainder of this paper is organized as follows. Section 2 discusses the background information related to the present study. Section 3 briefly provides the theoretical underpinning of the KNNreg and FKNNreg models and the Minkowski distance measure. Section 4 proposes the Md-FKNNreg method. Section 5 introduces the data used and the experiment setting for the proposed method and presents and discusses the empirical results obtained with the proposed method and benchmarks. Section 6 summarizes the main findings and provides concluding remarks.

2 Related work

The KNNreg model has the potential to tackle linear and non-linear problems in an effective way (Cai et al. 2020) and performs especially well in a high-dimensional space.

Accordingly, the growing popularity of the KNNreg method can be seen in various fields, including renewable energy (Hu et al. 2014; Huang and Perry 2016; Zhou et al. 2020), physics research (Durbin et al. 2021), biological studies (Yao and Ruzzo 2006), transportation (Cai et al. 2020; Dell'Acqua et al. 2015), robotics (Chen and Lau 2016), and telecommunication (Adege et al. 2018). In addition, some studies have also employed the KNNreg model with other approaches to develop effective hybrid models for specific applications. For example, Chen and Hao (2017) proposed an integrated framework by employing support vector machine (SVM) and KNNreg for stock market prediction. Salari et al. (2015) also presented a novel hybrid approach with a combination of a genetic algorithm (GA), the KNNreg method, and artificial neural network (ANN) for classification problems. Cheng et al. (2019) utilized the same idea as the KNNreg to introduce a novel approach for missing value imputations. Furthermore, the simplicity and strength of the KNNreg algorithm have encouraged researchers to develop different enhanced variants (for examples, see Buza et al. 2015; Guillen et al. 2010; Nguyen et al. 2016; Song et al. 2017) and to construct mathematical estimations (Biau et al. 2012).

An ideal distance measure must have the ability to precisely detect the similarity between two samples while allowing the researchers to understand how to compare, classify, or cluster those samples. Therefore, such metrics have great potential to influence the outcomes of the models used (Bergamasco and Nunes 2019). Accordingly, some previous studies focused only on which similarity measure best fit the particular situation (for examples, see Rodrigues 2018; Moghtadaiee and Dempster 2015; Huo et al. 2021). The Minkowski distance is the most investigated measure among the frequently applied techniques for measuring the similarity between instances in machine learning-based applications (Bergamasco and Nunes 2019; Cordeiro and Makarenkov 2016; Gueorguieva et al. 2017). The Minkowski distance is the main focus of this research because it offers the opportunity to compute the distance between two instances in several different ways and holds several well-known distances as special cases, e.g., the Manhattan and Euclidean distances.

The concept of the fuzzy theory, originally introduced by Zadeh (1965), can operate under uncertainty and has advanced in many different ways in various applications (for examples, see Chen et al. 1990; Chen and Hsiao 2005; Chen and Chen 2007; Chen and Chang 2010; Chen et al. 2009; Horng et al. 2005; Zeng et al. 2019). The FKNN classifier (Keller et al. 1985) was derived from fuzzy theory and has been one of the most effective techniques in supervised machine learning tasks. Nikoo et al. (2018) applied the FKNN classifier to a regression application without modifying its original algorithm explicitly (i.e., it was operated as a classification task). However, to the best of our knowledge, no one has attempted to utilize the FKNN model in the regression setting. Thus, the effectiveness of FKNNreg for machine learning applications requires further investigation.

3 Preliminaries

This section briefly discusses the KNNreg method, the FKNN method, and the Minkowski distance measure.

3.1 *K*-nearest neighbor regression

KNNreg (Benedetti 1977; Stone 1977; Turner 1977) is a simple, effective, and robust nonlinear regression method. The basic idea of KNNreg is to predict an output value to a given input sample based on a fixed number (k) of its nearest neighbors found from the input-output training samples. The k is a smoothing parameter, and its value controls the adaptability of the KNNreg method (Hu et al. 2014). KNNreg does not require an explicit training step besides the initial dataset's inputs and outputs, which represent a unique property. The notion of the KNNreg model can be formally defined as follows.

Let $T = \{(X_i, y_i)\}_{i=1}^{N}$ be a training dataset with N samples, where $X_i = \{x_1^i, x_2^i, \ldots, x_m^i\} \in \mathbb{R}^m$ is an input sample i from m-dimensional feature space, and its output value (response variable) is $y_i \in Y$, where $Y = \{y_1, y_2, \ldots, y_N\}$ denotes the set of output values. For a given new data sample X, the goal is to learn the predictor function $h(X)$ from the training dataset such that $\hat{y} \approx h(X)$, where $\hat{y}$ is the estimated value for the output y of X. The KNNreg starts with measuring the distance (d) between the test sample X and each sample X_i in T. In this case, the Euclidean distance is the most commonly adopted distance metric, and its formulation for the distance between $X = \{x_1, x_2, \ldots, x_m\}$ and X_i is presented by Eq. (1).

$$d(X, X_i) = \sqrt{\sum_{j=1}^{m} (x_j - x_j^i)^2}. \tag{1}$$

Next, the set of k nearest neighbors, $N_X^k = \{(X_i, y_i)\}_{i=1}^{k}$ for X, is found from the reordered training samples in T according to the increasing Euclidean distances. Finally, the output value y for X is estimated by taking the arithmetic mean of the output values $(y_1, y_2, \ldots, y_k)$ of the nearest neighbors (Song et al. 2017; Biau et al. 2012; Györfi et al. 2002) as follows:

$$\hat{y} = \frac{\sum_{j=1}^{k} y_j}{k}. \tag{2}$$

This is based on the assumptions that training samples in the N_X^k have similar output values to $h(X)$ (Kramer 2011) and also that all nearest neighbors in the N_X^k have equal importance in the prediction (Cover and Hart 1967).

3.2 Fuzzy *k*-nearest neighbor classification method

Unlike the KNN algorithm, the FKNN method uses the unbiased weighing scheme in the decision rule using the distances between the test sample and the nearest neighbor samples. Put it differently, the FKNN model computes a membership to the test sample for each class and makes the class decision according to the highest membership degree (Keller et al. 1985). These fuzzy memberships have excellent potential for accurate predictions (Kumbure et al. 2020). The membership degree of a given new sample X in a class i that is represented by the k nearest neighbors[1] is measured as follows:

$$u_i(y) = \frac{\sum_{j=1}^{k} u_{ij}(1/\|X - X_j\|^{2/(q-1)})}{\sum_{j=1}^{k}(1/\|X - X_j\|^{2/(q-1)})}, \tag{3}$$

where $q \in (1, +\infty)$ is the fuzzy strength parameter that controls the Euclidean distance $\|X - X_j\|^2$ between X and X_j to weigh the contribution of each nearest neighbor to the membership value. Also, u_{ij} is the membership of the sample X_j from the training data to the class i among the k nearest neighbors. Two methods are used to measure the u_{ij}: crisp memberships and fuzzy memberships. More details about these methods can be found in the work by Chen et al. (2011).

3.3 Minkowski distance

The Minkowski distance measure (also called L_p norm space) is a class of various distance functions that are formed by the parameter p. For two given samples X_i and X_j where $X_i = \{x_1^i, x_2^i, \ldots, x_m^i\} \in \mathbb{R}^m$ and $X_j = \{x_1^j, x_2^j, \ldots, x_m^j\} \in \mathbb{R}^m$, the Minkowski distance metric is defined as follows:

$$d_{Md}(X_i, X_j) = \left(\sum_{t=1}^{m} |x_t^i - x_t^j|^p\right)^{1/p} \quad \text{for} \quad p \geq 1. \tag{4}$$

From this metric, we can specify different distance functions by changing the value of p. For example, we can obtain the Manhattan distance (also known as the city block distance or L_1 norm) by setting $p = 1$ and the Euclidean distance, also referred to as L_2 norm (see also Eq. (1)) by setting $p = 2$.

[1] The set of nearest neighbors here is defined in terms of a classification problem, which means each nearest neighbor i contains m features values and a class label c_i (i.e., $X^i = \{x_1^i, x_2^i, \ldots, x_m^i, c_i\}$). X is also shaped by similar characteristics.

4 Proposed fuzzy *k*-nearest neighbor regression model using Minkowski distance

In this research, we focus on the fuzzy k-nearest neighbor regression. Given this, we define the FKNN method for regression together with the Minkowski distance. In this way, the novel regression method, Md-FKNNreg, is introduced. This method aims to achieve a reliable prediction for the predictor function by allowing the Minkowski distance to be adapted to the particular context with the optimal conditions. The procedure of the Md-FKNNreg method mainly includes four steps: measuring the distances, recognizing the nearest neighbors, computing the fuzzy weights, and making the prediction. The detailed process of this method is presented using the same notations in Sect. 3.1 as follows.

Step 1: Determine the Minkowski distance $d_{Md}(X, X_j)$ between X and X_i in T according to: $d_{Md}(X, X_j) = \left(\sum_{t=1}^{m} |x_t - x_t^j|^p\right)^{1/p}$.

Step 2: Find the set of k nearest neighbors N_X^k from the ranked training data samples according to increased Minkowski distances. Here, we used a grid-based search to find the optimal parameter p for the Minkowski distance and k that best fit a particular dataset.

Step 3: Calculate the fuzzy weight (w_i) for each nearest neighbor j using $d_{Md}(X, X_j)$ as follows:

$$w_j = \frac{1}{\left(1/d_{Md}(X, X_j)\right)^{\frac{2}{q-1}}}, \text{ for } j = 1, 2, \ldots, k, \tag{5}$$

where q is a fuzzy strength parameter, and $(\frac{2}{q-1})$ indicates the fuzziness exponent. The closer q is to 1, the larger the weights are. For distances over 1 unit, the larger q is, the smaller the weights are.

The purpose of these weights is to define a comprehensive linear predictor for the output value y such that $h(X) = W^T Y$, where $W = \{w_1, w_2, \ldots, w_k\} \in \mathbb{R}^k$. The weighted value w_j $(0 \leq w_j \leq 1)$ of the nearest neighbor X_j reflects its relative importance to Y.

Step 4: Predict the output value $\hat{y}$ for X by taking the weighted average (with the fuzzy weights) of the outputs y_j for $j = 1, 2, \ldots, k$ in the N_X^k according to the following equation:

$$\hat{y} = \frac{\sum_{j=1}^{k} w_j y_j}{\sum_{j=1}^{k} w_j}. \tag{6}$$

It is clear that in this method, the Minkowski distance is used not only to find the nearest neighbors but also to measure the weights. Accordingly, the Minkowski distance plays a critical role in the proposed framework. Besides, the Md-FKNNreg method is intuitively adaptive to the number of nearest neighbors k and the p of the distance function to vary with iterations throughout its search in a particular situation. This characteristic allows the method to expand the search area to a broader domain. The steps of the Md-FKNNreg method discussed above are summarized in Algorithm 1 by introducing a pseudo-code to it. In addition, the pseudo-code for the grid search method used is presented in Algorithm 2.

Algorithm 1: Md-FKNNreg

Input: $\{X_i, y_i\}_{i=1}^{N}$ (training data set), X (test data sample), $k\,(1 \le k \le N)$, $p\,(p \ge 1)$
Output: Predicted target value, $\hat{y}$ for X

Begin
for $i \leftarrow 1$ **to** N **do**
  Compute $d_{Md}(X, X_i)$ between X and X_i using Eq. (4),
  if $i < k$ **then**
    Add X_i to N_X^k
  else if *X_i is closer to X than any of neighbors in N_X^k* **then**
    Drop the farthest neighbor from the set N_X^k and add X_i
  end
end
for $j \leftarrow 1$ **to** k **do**
  Compute $w_j = \frac{1}{\left(1/d_{Md}(X, X_j)\right)^{\frac{2}{q-1}}}$
end
Estimate $\hat{y}$ by taking weighted average such as $\hat{y} = \frac{\sum_{j=1}^{k} w_j y_j}{\sum_{j=1}^{k} w_j}$
return $\hat{y}$

Algorithm 2: Grid search

Input: p-values: $\{p_i\}_{i=1}^{n_p}$, k-values: $\{k_i\}_{i=1}^{n_k}$, n-runs for cross-validation
Output: Optimal p^* and k^*

Begin
for $j \leftarrow 1$ **to** n **do**
  Split the training data randomly into training and validation sets
  for $j \leftarrow 1$ **to** n_k **do**
    for $s \leftarrow 1$ **to** n_p **do**
      Perform Md-FKNNreg model with $k(j)$ and $p(s)$ and save the RMSE
    end
  end
end
for $j \leftarrow 1$ **to** n_k **do**
  for $s \leftarrow 1$ **to** n_p **do**
    Average the RMSE over n-runs
  end
end
Estimate the minimum RMSE average and corresponding p^* and k^* values
return p^* and k^*

In the KNNreg method, the prediction of the output for a new sample is made through a uniform weighting scheme (Cheng 1984). This means it makes the prediction by taking the simple average of the outputs of the nearest neighbor samples and does not consider the distances between the new sample and its k nearest neighbors (i.e., all nearest neighbors have equal influence across the prediction) (Kramer 2011). In contrast, the FKNNreg uses an inverse weighting scheme that assigns higher weights to the closer training samples, allowing them more influence over the prediction. Moreover, the fuzzy strength parameter q controls how heavily the distance is weighted when determining the contribution of each nearest neighbor to the target sample (Keller et al. 1985). For example, when $q = 2$, the contributions of the nearest neighbors are weighted by the reciprocal of their distances to the target sample. Regarding the distance, the adopted distance metric plays a crucial role in achieving the best possible nearest neighbors and weighting them. Accordingly, the Minkowski distance is utilized in the proposed Md-FKNN method since it offers a more generalized nature[2] than the Euclidean distance and Manhattan distance. It has shown superior performance with supervised and unsupervised machine learning models compared to other distance measures (for examples, see Aggarwal et al. 2001; Ranmya and Sasikala 2019). Considering the above facts, overall, the proposed Md-FKNNreg is expected to produce significantly better results than the KNNreg and FKNNreg methods.

5 Experiment and results

This section presents the descriptions of the data sets used and the empirical procedures of the experiments conducted to investigate the regression performance of the proposed Md-FKNNreg model.

5.1 Data description

For our experiment, we selected eight real-world datasets that are freely available at the UCI Machine Learning repository (Dheeru and Taniskidou 2017) and at the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (Alcala-Fdez et al. 2011). As summarized in Table 1, each of these datasets holds different characteristics in terms of the number of instances and features. Also, the related area of each of the datasets is provided in the "Domain" in Table 1.

[2] This can be defined in terms of the "metric space": Minkowski metric space $\sim$ ($\mathbb{R}^m$, $\mathbb{L}^p$), Euclidean space $\sim$ ($\mathbb{R}^m$, $\mathbb{L}^2$), and Manhattan space $\sim$ ($\mathbb{R}^m$, $\mathbb{L}^1$).

Table 1 Summary of the datasets used in the experiment

Data set	Repository	Instances	Features	Domain
Stock	KEEL	950	9	Business
Airfoil	UCI	1503	5	Physics
AutoMPG	KEEL	392	6	Engineering
Baseball	KEEL	337	16	Sociology
Servo	UCI	167	4	IT
Laser	KEEL	993	4	Physics
Qsar Fish	UCI	908	6	Biology
Parkinson	UCI	5875	26	Medicine

5.2 Experimental setting

In each collected dataset, the data samples were divided into 40% for training, 40% for validation, and 20% for testing based on the works of Kumbure et al. (2019, 2020). Before data splitting, all the datasets were normalized into the unit interval of [0, 1] to avoid data differences between small and large ranges. For cross-validation, we adopted the holdout method (Arlot and Celisse 2010), in which the training and validation datasets were randomly sampled 30 times, and mean performance measures were computed from the results. The examination of the proposed method using the data can be categorized into two phases, training & validation and testing. In the training and validation step, we trained the model by optimizing the parameter values for p in the Minkowski distance and the number of k nearest neighbors. Here, we used mean regression error to determine the optimal parameter values. To find the best possible values for the parameters, we deployed a grid search technique during the training and validation. Then, we evaluated the performance of the model with optimal parameters in the testing phase. The steps of this process are summarized by the flowchart in Fig. 1.

The proposed Md-FKNNreg, Man-FKNNreg and Euc-FKNNreg models were implemented using MATLAB 2019b software. The KNNreg was implemented from scratch. The SVR, LASSO, and MLR models were developed using the Statistics and Machine Learning Toolbox in MATLAB. The computer used was an Intel® Core™ i5 1.8GHz, 16GB RAM with Microsoft Windows 10 operating system.

5.3 Baseline models

We compared the performance of the developed Md-FKNNreg method with the classical KNNreg, Man-FKNNreg, and Euc-FKNNreg methods. In addition, we also selected three more commonly used regression techniques, namely SVR (Drucker et al. 1997), LASSO regression (Tibshirani 1996; Wang et al. 2018), and MLR (Montgomery et al. 2012). The basic concepts of these methods are briefly presented.

SVR, a variant of SVM (Cortes and Vapnik 1995), is a non-linear kernel-based regression approach that performs the regression by constructing a hyperplane in a high-dimensional space. For a given test sample X, SVR develops a predictor function: $h(X) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)K(X, X_i) + b$ by mapping training samples onto the high-dimensional features space. Here, α_i and α_i^* are non-zero Lagrange multipliers, b is a bias constant, and K is the kernel function that represents the inner product of X and train sample X_i.

LASSO is a regularization-based[3] linear regression model. The model is selected to minimize the objective function: $\sum_{i=1}^{n}(w_0 + \sum_{j=1}^{m} w_j x_j^i - y_i)^2 + \lambda \sum_{j=1}^{m} w_j$, where λ is the regularization parameter that is used to control the empirical error. As the λ value increases, LASSO changes more coefficients to zero (Wang et al. 2018).

MLR (also referred to as ordinary least squares regression) is one of the oldest and most frequently employed techniques for analyzing the relationship between the response variable and multiple input variables. The general form of the MLR model can be given by $h(X_i) = w_0 + \sum_{j=1}^{m} w_j x_j^i + \epsilon$, where $h(X_i)$ is the predictor function for the sample $X_i = x_1^i, x_2^i, \ldots, x_m^i$, w_0 is the constant, w_j is the coefficient for the variable j, and ϵ is the error term ($\sim N(0, \sigma^2)$) of the model.

5.4 Parameter settings

The detailed parameter settings for the proposed method and benchmarks are presented in this sub-section. In the Md-FKNNreg algorithm, the value for p of the Minkowski distance was selected from $\{1, 1.5, \ldots, 5\}$. The number of nearest neighbors k was chosen from the range $\{1, 2, \ldots, 25\}$ for all nearest neighbor regression approaches. The value of the fuzzy strength parameter m was kept constant at $m = 1.5$ according to Arif et al. (2010) for both the Md-FKNNreg and FKNNreg models.

The kernel function is the most critical ingredient in the SVR model (Ali and Smith-Miles 2006) because it helps the model achieve robust mapping from training samples to the prediction. Accordingly, we tested the performance of the SVR model using three different kernels: *linear*, *Radial Basis Function (RBF)*, and *polynomial* based on Ali and Smith-Miles (2006). For the Lasso model, the regularization parameter λ was tuned from the range $\{0.001, 0.01, \ldots, 100\}$ by following the experiments of Saccoccio et al. (2014). Here, we attempted to create a

[3] In machine learning, regularization is a more sophisticated technique that is used to solve model overfitting problems.

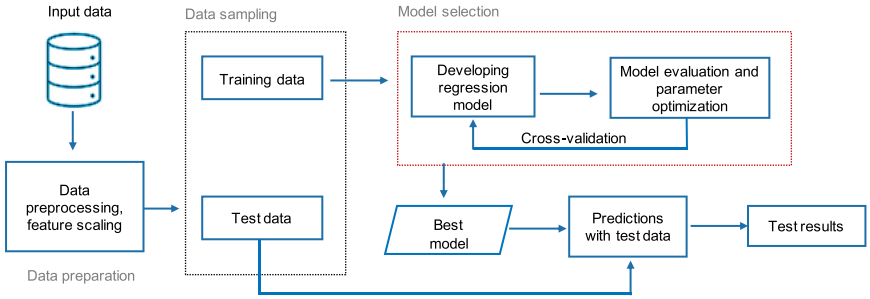


Fig. 1 A workflow of the development and evaluation of Md-FKNNreg model

balance for the λ values due to the fact that low values of λ prefer predictor functions that achieve a small training error while larger values tend to obtain simple prediction functions (Wang et al. 2018). With the multiple regression model, we tested four different model types from the toolbox: *linear*, *interaction*[4], *purequadratic*[5], and *quadratic*[6]. Using these different types of models, we were able to generate more sophisticated MLR versions for competition in the comparison even though high-order terms of the features were not deployed with the nearest neighbor methods. The rest of the parameter values in the SVR, LASSO, and MLR models were set to the default values according to the toolbox specifications.

5.5 Evaluation metrics

To evaluate the prediction performance of the proposed regression method and benchmarks, we adopted two frequently applied measures: RMSE and R^2. RMSE computes the square root of the average differences between the model's predictions and the true values. R^2 is the proportion of the variation in the response variable, which is "explained" by the regression model compared to the mean (Kurz-Kim and Loretan 2014). It is a statistical measure that implies how closely the data points in the response variable fit to the values of the regression model. In general, higher R^2 values and smaller *RMSE* values reflect better performance in the regression model (Pham 2019). The formulas used for both evaluation methods are defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2} \tag{7}$$

$$R^2 = \left(1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}\right) \times 100\% \tag{8}$$

where n is the number of samples in the test data, $\hat{Y}_i$ indicates the predicted value, Y_i indicates the true value of the ith test sample, and $\bar{Y}$ is the average of the true values. As shown in Eq. (8), the percentage values of R^2 are considered.

When it is necessary to apply several models to a particular problem and pick the best one, the usual method is to use several evaluation metrics to measure the models' performance and select the best model with the highest performance. However, when trying to prove that one model outperforms another for a particular problem, we must use a statistical test of significance and validate the claim of improved performance (Borovicka et al. 2012).

Following Chen et al. (2011), we adopted the paired t test, one of the most commonly used statistical tests in machine learning. This analysis tested the null hypothesis that there is no significant difference between two mean RMSE rates at the 0.05 significant level. Here we considered the error samples from the holdout method (size of 30×1) for each regression model when the optimal parameter values were employed. The standard deviations were also computed.

5.6 Results and discussion

In this subsection, we first present the results of the proposed method compared with the baseline methods from the training and validation step. After that, optimal parameter values observed for each model are discussed. Then, the performances of the fitted models for the training and validation data are evaluated with the test datasets.

[4] Includes an intercept, linear term for each variable, and all products of pairs of distinct variables.

[5] Includes an intercept term and linear and squared terms for each variable.

[6] Includes a constant term, linear and squared terms of each variable, and all products of pairs of distinct variables.

Table 2 The results obtained for all methods in the training & validation step

Data set	Measure	Md-FKNNreg	Man-FKNNreg	Euc-FKNNreg	KNNreg (Biau et al. 2012)	SVR (Drucker et al. 1997)	LASSO (Tibshirani 1996)	MLR (Montgomery et al. 2012)
Stock	RMSE	**0**.0272	**0**.0272	0.0276	0.0308	0.0381	0.0854	0.0421
	STD	0.0022	0.0022	0.0023	0.0025	0.0062	0.0029	0.0026
Airfoil	RMSE	**0**.0932	0.0994	0.0942	0.1027	0.0957	0.1273	0.1113
	STD	0.0047	0.0049	0.0049	0.0046	0.0037	0.0031	0.0030
AutoMPG	RMSE	**0**.0769	**0**.0769	0.0812	0.0815	0.0843	0.0952	0.0779
	STD	0.0039	0.0039	0.0050	0.0047	0.0075	0.0045	0.0043
Baseball	RMSE	**0**.1235	**0**.1235	0.1254	0.1307	0.1326	0.1276	0.1293
	STD	0.0096	0.0096	0.0092	0.0091	0.0083	0.0072	0.0071
Servo	RMSE	0.1611	0.1611	0.1612	**0**.1549	0.1818	0.1861	0.1579
	STD	0.0221	0.0221	0.0185	0.0157	0.0296	0.0145	0.0150
Laser	RMSE	0.0407	0.0407	0.0431	0.0451	**0**.0396	0.0885	0.0438
	STD	0.0083	0.0083	0.0078	0.0084	0.0076	0.0054	0.0050
Qsar Fish	RMSE	**0**.0966	0.0969	0.0993	0.0981	0.0971	0.1022	0.1021
	STD	0.0036	0.0036	0.0035	0.0039	0.0037	0.0042	0.0042
Parkinson	RMSE	**0**.0583	**0**.0583	0.0626	0.0678	0.0810	0.1936	0.1865
	STD	0.0028	0.0028	0.0027	0.0022	0.0036	0.0016	0.0015

5.6.1 Results with the validation data samples

Table 2 summarizes the results of all the methods for each of the datasets from the training and validation step. In the table, we report the minimum RMSE and standard deviation (STD) for the RMSE as the performance measures. Notice that these mean RMSEs and standard deviations are the result of the holdout method with 30 repetitions.

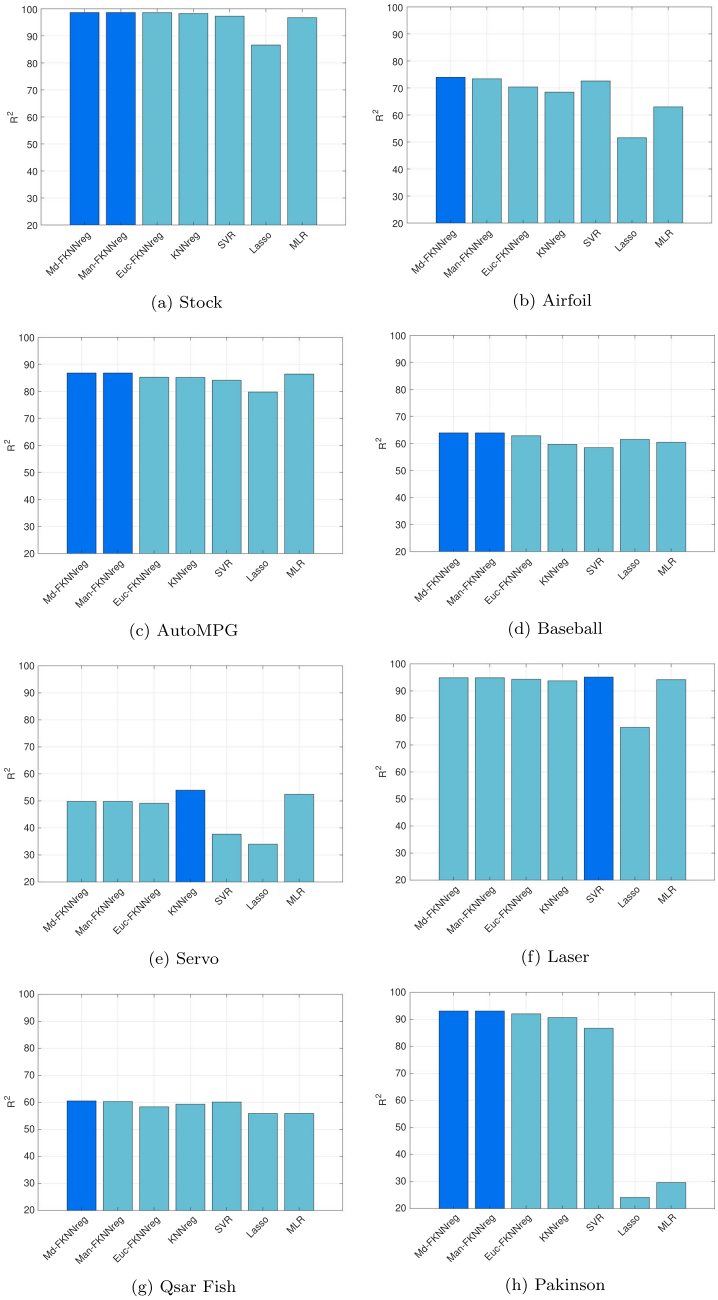
From Table 2, it is apparent that the proposed Md-FKNNreg method outperformed all benchmarks for six datasets and had the second-best performance for the rest of the datasets. Also, the standard deviations of the proposed method were reasonable for all cases. In particular, the Md-FKNNreg method achieved significantly improved performance compared with the Euc-FKNNreg method even though the results of the two methods were comparable in some cases, for example, in the cases of Servo and Stock. Additionally, the Md-FKNNreg and Man-FKNNreg models produced the same results over six datasets. Moreover, the KNNreg and SVR models achieved the lowest mean RMSE results in the cases of Servo and Laser, respectively. Finally, neither the LASSO and nor MLR models offered comparative results for the datasets compared to the other models.

Figure 2 illustrates the R^2values of each for the optimized regression models from the training and validation step for each dataset. Notice that the R^2 values revealed by the bar-heights in the graphs refer to the maximum of mean R^2 values obtained from the holdout cross-validation. The bar graphs in the figure are also displayed so that the blue bar represents the highest R^2 while the other bars indicate the rest of the values. From Fig. 2, one can observe a similar indications about the regression performance of the proposed Md-FKNNreg method and benchmarks as from Table 2.

5.6.2 Evaluation of the optimal parameter values

During the training and validation, we evaluated the regression results by tuning the parameters of the Md-FKNNreg and benchmark models. Figure 3 displays the impacts of different combinations of the parameters k and p in the Md-FKNNreg method on the RMSE (and R^2) with the Stock dataset. Figure 3 demonstrates how these parameters maintain the good performance of the Md-FKNNreg method for the Stock data set.

Corresponding to the results in Table 2, the optimal parameter values observed for the proposed Md-FKNNreg and benchmark models are presented in Table 3. The table shows that $p = 1$ (i.e., the Manhattan distance) obtained the best performance with the Md-FKNNreg method for the majority of the datasets. We also applied the Manhattan distance-based FKNNreg and received the same results as for the Md-FKNNreg method with $p = 1$. This finding is consistent with the implications of Aggarwal et al. (2001), who showed that the Manhattan distance (L_1 norm) is the best option for high-dimensional applications. Additionally, it seems that relatively low k values (varying from 1 to 13) are better suited to the original KNNreg method, while high k values (ranging from 2 to 25) work



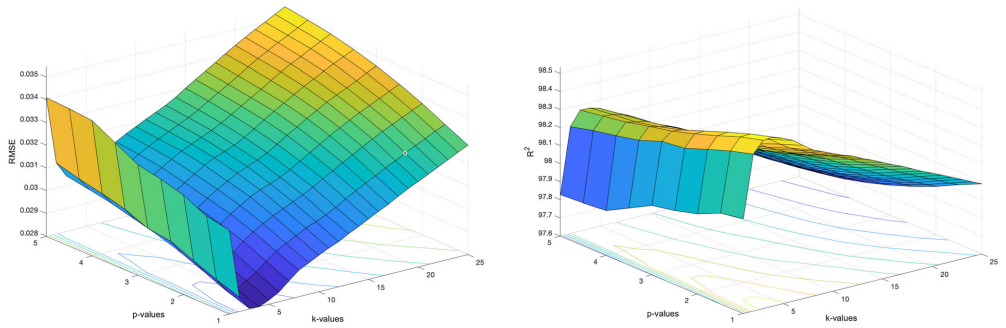
(a) Stock (b) Airfoil

(c) AutoMPG (d) Baseball

(e) Servo (f) Laser

(g) Qsar Fish (h) Pakinson

Fig. 2 Observed R^2 values of each model for each dataset



(a) (b)

Fig. 3 The averages of the RMSE and R^2 values for the Md-FKNNreg method for different parameter combinations (p, k) in the training and validation step with the Stock dataset

Table 3 Optimal parameter values of each model for each dataset

Model	Parameters	Stock	Airfoil	AutoMPG	Baseball	Servo	Laser	Qsar fish	Parkinson
Md-FKNNreg	(k, p)	(9, 1)	(5, 4.5)	(16, 1)	(8, 1)	(8, 1)	(4, 1)	(13, 1.5)	(4, 1)
Man-FKNNreg	k	9	2	16	8	8	4	15	4
Euc-FKNNreg	k	9	11	25	13	6	4	25	5
KNNreg (Biau et al. 2012)	k	2	1	7	13	9	3	10	3
SVR (Drucker et al. 1997)	kernel	RBF	RBF	RBF	linear	polynomial	RBF	RBf	RBF
LASSO (Tibshirani 1996)	λ	0.001	0.001	0.010	0.001	0.001	0.001	0.001	0.001
MLR (Montgomery et al. 2012)	Model	Quadratic	Quadratic	Interaction	Linear	Pure-quadratic	Quadratic	Linear	Linear

better for its fuzzy versions. Moreover, the RBF kernel appears to be the most suitable for the SVR model because it achieved high performance for all datasets except for Baseball and Servo data. Regarding the LASSO model having the most instances of the least-squares estimations, we believe that this is because of the lowest $\lambda = 0.001$ shows the optimum. The optimal type of MLR model varied depending on the particular datasets. By taking the parameters and other results together, it is evident that the fuzzy variants have more potential for linear and non-linear problems. For instance, the case with the Baseball data could be considered a linear problem since both SVR and MLR hold linear parameters, but high performance was achieved with the Md-FKNNreg, Man-FKNNreg, and Euc-FKNNreg methods.

5.6.3 Results with the test data samples

This subsection presents the regression results of the Md-FKNNreg and baseline models with the testing data samples that were initially split from the original datasets. In this testing phase, we used the optimized parameter values and training data samples stored from the cross-validation step to evaluate the models' performances with the previously unseen test data samples.

Table 4 summarizes the results with the mean RMSEs and the standard deviations (STD) of the proposed Md-FKNNreg and benchmarks models for the selected datasets. In addition, the average computational time (Com. time, in seconds) of each method in the testing phase is also reported.

The results of Table 4 show that the Md-FKNNreg method outperformed all benchmarks for all datasets, verifying the effectiveness of the proposed Md-FKNNreg

Table 4 The results obtained for all methods in the testing step

Data set	Measure	Md-FKNNreg	Man-FKNNreg	Euc-FKNNreg	KNNreg (Biau et al. 2012)	SVR (Drucker et al. 1997)	LASSO (Tibshirani 1996)	MLR (Montgomery et al. 2012)
Stock	RMSE mean	**0**.0294	**0**.0294	0.0302	0.0311	0.0406	0.0762	0.0407
	STD	0.0016	0.0016	0.0017	0.0017	0.0041	0.0007	0.0011
	Com. time	0.0230	0.0171	0.0200	0.0207	0.0009	0.0002	0.0082
Airfoil	RMSE mean	**0**.0963	0.0966	0.1002	0.1036	0.0986	0.1342	0.1182
	STD	0.0046	0.0039	0.0046	0.0048	0.0022	0.0010	0.0013
	Com. time	0.0428	0.0306	0.0264	0.0233	0.0009	0.0001	0.0030
AutoMPG	RMSE mean	**0**.0687	**0**.0687	0.0719	0.0728	0.0707	0.0824	0.0725
	STD	0.0024	0.0024	0.0023	0.0028	0.0036	0.0015	0.0020
	Com. time	0.0098	0.0066	0.0075	0.0066	0.0004	0.0002	0.0020
Baseball	RMSE mean	**0**.1184	**0**.1184	0.1239	0.1316	0.1448	0.1329	0.1350
	STD	0.0080	0.0080	0.0082	0.0070	0.0064	0.0043	0.0050
	Com. time	0.0068	0.0052	0.0060	0.0051	0.0004	0.0001	0.0003
Servo	RMSE mean	**0**.1120	**0**.1120	0.1231	0.1167	0.1577	0.1602	0.1197
	STD	0.0210	0.0210	0.0237	0.0164	0.0218	0.0038	0.0107
	Com. time	0.0062	0.0060	0.0062	0.0054	0.0014	0.0003	0.0051
Laser	RMSE mean	**0**.0435	**0**.0435	0.0441	0.0527	0.0483	0.0986	0.0509
	STD	0.0094	0.0094	0.0124	0.0106	0.0158	0.0026	0.0058
	Com. time	0.0237	0.0168	0.0192	0.0175	0.0004	0.0002	0.0030
Qsar Fish	RMSE mean	**0**.0902	0.0905	0.0917	0.0942	0.0943	0.0976	0.0973
	STD	0.0021	0.0021	0.0027	0.0020	0.0026	0.0010	0.0009
	Com. time	0.0249	0.0058	0.0201	0.0046	0.0005	0.0001	0.0003
Parkinson	RMSE mean	**0**.0566	**0**.0566	0.0608	0.0666	0.0786	0.1915	0.1844
	STD	0.0025	0.0025	0.0027	0.0027	0.0028	0.0005	0.0031
	Com. time	0.2296	0.2229	0.2971	0.2815	0.0178	0.0039	0.0101
Overall	RMSE mean	**0**.0769	0.0770	0.0807	0.0837	0.0917	0.1217	0.1023

method for regression problems. Compared with the Man-FKNNreg results, the Md-FKNNreg model achieved somewhat higher accuracy for the Airfoil and Qsar Fish datasets. Additionally, the Md-FKNNreg showed significantly better performance than the Euc-FKNNreg method across almost all datasets. This reveals that introducing the Minkowski distance in the learning part can result in finding more reasonable nearest neighbors, leading to better performance compared to the Manhattan and Euclidean distances. Considering the KNNreg and SVR models, even though these achieved the lowest errors in some cases during the training and validation, the Md-FKNNreg method outperformed them on all testing cases. This proves that the proposed Md-FKNNreg model has more power to overcome over-fitting issues than both KNNreg and SVR models. Moreover, the lowest regression performance for the testing data, similar to the validation data, occurred for both the LASSO and MLR models.

Based on the testing times, it is clear that the computational complexity of the FKNNreg methods was relatively high compared with the other methods used. This might be because of the inclusion of the weight generation process based on the inverse of the distances and the fuzzy strength parameter. Additionally, the proposed Md-FKNNreg method required more time (in seconds) than the Euc-FKNNreg and Man-FKNNreg methods to deliver the

results since it includes an additional computation with the parameter p.

To validate the test results statistically, a paired t test was applied, and the observed results for the P values and test statistics are presented in Table 5. The t test results demonstrate whether there is a statistically significant difference between mean RMSEs of the compared regression methods. From the evidence in the table, it is apparent that the Md-FKNNreg method yielded statistically significantly better performance than the benchmarks in terms of the lowest error. In particular, we observed that the proposed method could not produce statistically significant results for either the Servo dataset (compared with the Euc-FKNNreg and KNNreg methods) or the Laser dataset (compared with the Euc-FKNNreg and SVR methods). It should be noted here that we did not find a statistically

Table 5 Paired t test results on the performance of the Md-FKNNreg method vs. five benchmarks for the test datasets

Data set	Paired t with Md-FKNNreg	P value	Test statistic
Stock	Euc-FKNNreg	0.0463	Significant
	KNNreg (Biau et al. 2012)	1.0715e−04	Significant
	SVR (Drucker et al. 1997)	3.8055e−20	Significant
	LASSO (Tibshirani 1996)	1.7732e−76	Significant
	MLR (Montgomery et al. 2012)	1.8274e−38	Significant
Airfoil	Euc-FKNNreg	0.0014	Significant
	KNNreg (Biau et al. 2012)	4.2915e−08	Significant
	SVR (Drucker et al. 1997)	0.0131	Significant
	LASSO (Tibshirani 1996)	3.3463e−50	Significant
	MLR (Montgomery et al. 2012)	2.3700e−36	Significant
AutoMPG	Euc-FKNNreg	2.7216e−06	Significant
	KNNreg (Biau et al. 2012)	8.6347e−08	Significant
	SVR (Drucker et al. 1997)	0.0141	Significant
	LASSO (Tibshirani 1996)	2.1512e−34	Significant
	MLR (Montgomery et al. 2012)	1.6638e−08	Significant
Baseball	Euc-FKNNreg	0.0101	Significant
	KNNreg (Biau et al. 2012)	5.3335e−09	Significant
	SVR (Drucker et al. 1997)	1.9285e−20	Significant
	LASSO (Tibshirani 1996)	3.1952e−12	Significant
	MLR (Montgomery et al. 2012)	1.1241e−13	Significant
Servo	Euc-FKNNreg	0.0600	Not significant
	KNNreg (Biau et al. 2012)	0.3348	Not significant
	SVR (Drucker et al. 1997)	2.2466e−11	Significant
	LASSO (Tibshirani 1996)	6.2441e−18	significant
	MLR (Montgomery et al. 2012)	0.0778	Not significant
Laser	Euc-FKNNreg	0.8251	Not significant
	KNNreg (Biau et al. 2012)	7.6651e−04	Significant
	SVR (Drucker et al. 1997)	0.1544	Not significant
	LASSO (Tibshirani 1996)	8.6754e−38	Significant
	MLR (Montgomery et al. 2012)	4.9310e−04	Significant
Qsar Fish	Euc-FKNNreg	0.0155	Significant
	KNNreg (Biau et al. 2012)	3.0113e−10	Significant
	SVR (Drucker et al. 1997)	9.4806e−09	Significant
	SVR (Drucker et al. 1997)	1.3310e−24	significant
	MLR (Montgomery et al. 2012)	5.3260e−24	Significant
Parkinson	Euc-FKNNreg	3.7463e−08	Significant
	KNNreg (Biau et al. 2012)	1.9184e−21	Significant
	SVR (Drucker et al. 1997)	2.0103e−38	Significant
	LASSO (Tibshirani 1996)	2.2655e−93	Significant
	MLR (Montgomery et al. 2012)	6.3721e−81	Significant

significant difference between Md-FKNNreg and Man-FKNNreg results for any dataset. In addition to finding no difference between the test results of Md-FKNNreg and Man-FKNNreg for six datasets, the t test produced no evidence of a significant difference between these methods for the Airfoil and Qsar Fish cases.

6 Conclusions

This paper proposed a new generalized regression model based on the FKNN rule and investigated its effectiveness on different regression problems from various domains. The Minkowski distance metric was introduced into the nearest neighbor search in the proposed algorithm to examine how it improves accuracy. Accordingly, the proposed method was named the Md-FKNNreg model. The effectiveness of the Md-FKNNreg method was evaluated in comparative experiments with the standard nearest neighbor and three well-known regression methods, namely SVR, LASSO, and MLR. In addition, Man-FKNNreg and Euc-FKNNreg methods were implemented, and their results were compared. For the experiments, we used eight real-world datasets that are freely available from machine learning repositories. The regression performance of each model for those datasets were discussed in terms of the RMSE, R^2 and standard deviation. The results showed that the Md-FKNNreg method outperformed the benchmarks and is a suitable choice for regression problems. In our experiments, Md-FKNNreg gave the lowest overall average RMSE of 0.0769.

The results of the experiments showed that the proposed Md-FKNNreg method achieved statistically significantly higher performance than the other methods in almost all cases in terms of the RMSE means. Additionally, we found that the Minkowski distance with $p = 1$ yielded the optimal Md-FKNNreg model, which then achieved the best performance for the majority of the datasets. In other words, the Man-FKNNreg showed promising results for the regression at large, supporting the indications in the study by Aggarwal et al. (2001).

However, it should be noted that the computational complexity of the proposed method is relatively high compared with the Euc-FKNNreg and KNNreg methods because an additional calculation with the Minkowski distance is included in the learning algorithm. Despite that, this research has offered some insight into further investigations. For example, it would be interesting to see how the Md-FKNNreg method adapts and performs in regression applications in which KNNreg was previously utilized (e.g., Hu et al. 2014; Cai et al. 2020; Yao and Ruzzo 2006; Huang and Perry 2016; Zhou et al. 2020; Durbin et al. 2021; Dell'Acqua et al. 2015). Furthermore, future research directions may also test the effect of combining Md-FKNNreg with other efficient variants, such as SVM (Chen and Hao 2017) and ANN (Salari et al. 2015) in a hybrid framework.

Funding Open access funding provided by LUT University (previously Lappeenranta University of Technology (LUT)).

Availability of data and materials The data used in the manuscript are freely available from the UCI and KEEL repositories.

Code availability The code is available in https://github.com/MahindaMK/A-generalized-fuzzy-k-nearest-neighbor-regression-model-based-on-Minkowski-distance.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Adege AB, Yayeh Y, Berie G, Lin H, Yen L, Li YR (2018) Indoor localization using k-nearest neighbor and artificial neural network back propagation algorithms. In: 27th Wireless and Optical Communication Conference (WOCC), pp 1–2

Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional space. Database Theory ICDT 2001. Springer, Berlin, pp 420–434

Alcala-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Log Soft Comput 17:255–287, https://sci2s.ugr.es/keel/datasets.php

Ali S, Smith-Miles KA (2006) A meta-learning approach to automatic kernel selection for support vector machines. Neurocomputing 70:173–186

Arif M, Akram MU, Minhas FA (2010) Pruned fuzzy k-nearest neighbor classifier for beat classification. J Biomed Sci Eng 3:380–3899

Arlot S, Celisse A (2010) A survey of cross-validation procedures for model selection. Stat Surv 4:40–79

Benedetti JK (1977) On the nonparametric estimation of regression functions. J R Stat Soc Series B 39:248–253

Bergamasco LCC, Nunes FLS (2019) Intelligent retrieval and classification in three-dimensional biomedical images-a systematic mapping. Comput Sci Rev 31:19–38
Biau G, Devroye L, Dujmović V, Krzyżak A (2012) An affine invariant k-nearest neighbor regression estimate. J Multivar Anal 112:24–34
Borovicka T, Jirina MJ, Kordik P, Jirina M (2012) Selecting representatives data sets Advances in Data Mining Knowledge Discovery and Applications. Rijeka, Croatia, pp 43–70
Buza K, Nanopoulos A, Nagy G (2015) Nearest neighbor regression in the presence of bad hubs. Knowl Based Syst 86:250–260
Cai L, Yu Y, Zhang S, Song Y, Xiong Z, Zhou T (2020) A sample-rebalanced outlier-rejected k -nearest neighbor regression model for short-term traffic flow forecasting. IEEE Access 8:22686–22696
Chang H, Yeung DY, Cheung WK (2006) Relaxational metric adaptation and its application to semi-supervised clustering and content-based image retrieval. Pattern Recognit 39:1905–1917
Chen SM, Chang YC (2010) Multi-variable fuzzy forecasting based on fuzzy clustering and fuzzy rule interpolation techniques. Inf Sci 180:4772–4783
Chen S, Chen L (2007) A fuzzy hierarchical clustering method for clustering documents based on dynamic cluster centers. J Chin Inst Eng 30:169–172
Chen Y, Hao Y (2017) A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. Expert Syst Appl 80:340–355. https://doi.org/10.1016/j.eswa.2017.02.044
Chen SM, Hsiao HR (2005) A new method to estimate null values in relational database systems based on automatic clustering techniques. Inf Sci 169:47–69
Chen J, Lau HYK (2016) Learning the inverse kinematics of tendon-driven soft manipulators with k-nearest neighbors regression and gaussian mixture regression. In: 2nd International conference on control, automation and robotics (ICCAR), pp 103–107
Chen HL, Liu DY, Yang B, Wang SJ (2011) An adaptive fuzzy k-nearest neighbor method based on parallel particle swarm optimization for bankruptcy prediction. In: Lecture Notes in computer science 6634 LNAI (PART 1), pp 249–264
Chen SM, Ke JS, Chang JF (1990) Knowledge representation using fuzzy petri nets. IEEE Trans Knowl Data Eng 2:311–319
Chen SM, Wang NY, Pan JS (2009) Forecasting enrollments using automatic clustering techniques and fuzzy logical relationships. Expert Syst Appl 36:11070–11076
Chen HL, Huang CC, Yu XG, Xu X, Sun X, Wang G, Wang SJ (2013) An efficient diagnosis system for detection of parkinson's disease using fuzzy k-nearest neighbor approach. Expert Syst Appl 40(1):263–271
Cheng PE (1984) Strong consistency of nearest neighbor regression function estimators. J Multivar Anal 15:63–72
Cheng CH, Chan CP, Sheu YJ (2019) A novel purity-based k nearest neighbors imputation method and its application in financial distress prediction. Eng Appl Artif Intell 81:283–299
Cordeiro R, Makarenkov V (2016) Applying subclustering and lp distance in weighted k-means with distributed centroids. Neurocomputing 173:700–707. https://doi.org/10.1016/j.neucom.2015.08.018
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13:21–27
Dell'Acqua P, Bellotti F, Berta R, De Gloria A (2015) Time-aware multivar. nearest neighbor regression methods for traffic flow prediction. IEEE Trans Intell Transp Syst 16:3393–3402
Dettmann E, Becker C, Schmeiser C (2011) Distance functions for matching in small samples. Comput Stat Data Anal 55:1942–1960
Dheeru D, Taniskidou EK (2017) Uci machine learning repository. http://archive.ics.uci.edu/ml
Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V (1997) Support vector regression machines. Neural Inf Proc Syst 9:155–161
Durbin M, Wonders MA, Flaska M, Lintereur AT (2021) K-nearest neighbors regression for the discrimination of gamma rays and neutrons in organic scintillators. Nucl Instrum Methods Phys Res A 987:164826
Gueorguieva N, Valova I, Georgiev G (2017) M&mfcm: Fuzzy c-means clustering with mahalanobis and minkowski distance metrics. Proc Comput Sci 114:224–233. https://doi.org/10.1016/j.procs.2017.09.064
Guillen A, Herrera LJ, Rubio G, Pomares H, Lendasse A, Rojas I (2010) New method for instance or prototype selection using mutual information in time series prediction. Neurocomputing 73:2030–2038
Györfi L, Kohler M, Krzyzak A, Walk H (2002) A distribution free theory of nonparametric regression. Springer, New York
Horng YJ, Chen SM, Chang YC, Lee CH (2005) A new method for fuzzy information retrieval based on fuzzy hierarchical clustering and fuzzy inference techniques. IEEE Trans Fuzzy Syst 13:216–228
Hu C, Jain G, Zhang P, Schmidt C, Gomadam P, Gorka T (2014) Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. Appl Energy 129:49–55
Huang J, Perry M (2016) A semi-empirical approach using gradient boosting and k-nearest neighbors regression for GEFCom2014 probabilistic solar power forecasting. Int J Forecast 32:1081–1086
Huo J, Ma Y, Lu C, Li C, Duan K, Li H (2021) Mahalanobis distance based similarity regression learning of nirs for quality assurance of tobacco product with different variable selection methods. Spectrochimica Acta A Mol Biomol Spectrosc 251:119364
Jenicka S, Suruliandi A (2011) Empirical evaluation of distance measures for supervised classification of remotely sensed image with modified multivariate local binary pattern. In: International conference on emerging trends in electrical and computer technology (ICETECT), pp 762–767
Kaski S, Sinkkonen J, Peltonen J (2001) Bankruptcy analysis with self-organizing maps in learning metrics. IEEE Trans Neural Netw Learn Syst 12(4):936–947
Keller JM, Gray MR, Givens JA (1985) A fuzzy k-nearest neighbor algorithm. IEEE Trans Syst Man Cybern Syst 15:580–585
Koloseni D, Lampinen J, Luukka P (2012) Optimized distance metrics for differential evolution based nearest prototype classifier. Expert Syst Appl 39(12):10564–10570
Koloseni D, Lampinen J, Luukka P (2013) Differential evolution based nearest prototype classifier with optimized distance measures for the features in the data set. Expert Syst Appl 40(10):4075–4081
Kramer O (2011) Dimensionality reduction by unsupervised K-nearest neighbor regression. In: Proceedings of the 10th International Conference on Machine Learning and Applications, ICMLA, pp 275–278
Kumbure MM, Luukka P, Collan M (2019) An enhancement of fuzzy k-nearest neighbor classifier using multi-local power means. In: Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019), Atlantis Press, pp 83–90
Kumbure MM, Luukka P, Collan M (2020) A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean. Pattern

Recognit Lett 140:172–178. https://doi.org/10.1016/j.patrec.2020.10.005
Kurz-Kim JR, Loretan M (2014) On the properties of the coefficient of determination in regression models with infinite variance variables. J Econ 181:15–24
Liu X, Beyrend-Dur D, Dur G, Ban S (2013) Effects of temperature on life history traits of Eodiaptomus japonicus (copepoda: Calanoida) from lake biwa (japan). Limnology 15:85–97
Moghtadaiee V, Dempster AG (2015) Determining the best vector distance measure for use in location fingerprinting. Pervasive Mob Comput 23:59–79. https://doi.org/10.1016/j.pmcj.2014.11.002
Montgomery DC, Peck EA, Vining GG (2012) Introduction to linear regression analysis. John Wiley & Sons, Hoboken
Nguyen B, Morell C, Baets BD (2016) Large-scale distance metric learning for k-nearest neighbors regression. Neurocomputing 214:805–814
Nikoo MR, Kerachian R, Alizadeh MR (2018) A fuzzy knn-based model for significant wave height prediction in large lakes. Oceanologia 60:153–168
Pham H (2019) A new criterion for model selection. Mathematics 7:1215
Ranmya R, Sasikala T (2019) An efficient minkowski distance-based matching with merkle hash tree authentication for biometric recognition in cloud computing. Soft Comput 23:13423–13431
Rastin N, Jahromi MZ, Taheri M (2021) A generalized weighted distance k-nearest neighbor for multi-label problems. Pattern Recognit 114:107526. https://doi.org/10.1016/j.patcog.2020.107526
Rodrigues EO (2018) Combining minkowski and chebyshev: new distance proposal and survey of distance metrics using k-nearest neighbours classifier. Pattern Recognit Lett 110:66–71
Saccoccio M, Wan TH, Chen C, Ciucci F (2014) Optimal regularization in distribution of relaxation times applied to electrochemical impedance spectroscopy: ridge and lasso regression methods - a theoretical and experimental study. Electrochim Acta 147:470–482
Salari N, Shohaimi S, Najafi F, Nallappan M, Karishnarajah I (2015) Time-aware multivar. nearest neighbor regression methods for traffic flow prediction. IEEE Trans Intell Transp Syst 16:3393–3402
Shirkhorshidi AS, Aghabozorgi S, Wah TY (2015) A comparison study on similarity and dissimilarity measures in clustering continuous data. PLOS ONE 10(12):1–20
Song Y, Liang J, Lu J, Zhao X (2017) An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing 251:26–34
Stone CJ (1977) Consistent nonparametric regression. Ann Stat 5:595–645
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Methodol 58:267–288
Turner T (1977) Exploratory data analysis. Addison Wesley, Reading
Wang S, Ji B, Zhao J, Liu W, Xu T (2018) Predicting ship fuel consumption based on LASSO regression. Transp Res D Transp 65:817–824
Yao Z, Ruzzo W (2006) A regression-based k nearest neighbor algorithm for gene function prediction from heterogeneous data. BMC Bioinformatics 7:S11
Yu S, De Backer S, Scheunders P (2002) Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. Pattern Recognit Lett 23(1):183–190
Zadeh LA (1965) Fuzzy sets. Inf and Control 8:338–353
Zeng S, Chen SM, Teng MO (2019) Fuzzy forecasting based on linear combinations of independent variables, subtractive clustering algorithm and artificial bee colony algorithm. Inf Sci 484:350–366
Zhou Y, Huang M, Pecht M (2020) Remaining useful life estimation of lithium-ion cells based on k- nearest neighbor regression with differential evolution optimization. J Clean Prod 249:119409. https://doi.org/10.1016/j.jclepro.2019.119409

ACTA UNIVERSITATIS LAPPEENRANTAENSIS

1008. LEHTINEN, VESA. Organisaation emergentti itseohjautuvuus, case sinfoniaorkesteri: ”Miksi orkesteri soittaa hyvin, vaikka sitä johdettaisiin huonosti?”. 2022. Diss.

1009. KÄHKÖNEN, TIINA. Employee trust repair in the context of organizational change – identification and measurement of active trust repair practices. 2022. Diss.

1010. AHONEN, AILA. Challenges in sport entrepreneurship: cases in team sport business. 2022. Diss.

1011. LEVIKARI, SAKU. Acoustic emission testing of multilayer ceramic capacitors. 2022. Diss.

1012. ZAHEER, MINHAJ. Evaluation of open-source FEM software performance in analysing converter-fed induction machine losses. 2022. Diss.

1013. HAAPANIEMI, JOUNI. Power-based electricity distribution tariffs providing an incentive to enhance the capacity effectiveness of electricity distribution grids. 2022. Diss.

1014. BUAH, ERIC. Artificial intelligence technology acceptance framework for energy systems analysis. 2022. Diss.

1015. GIVIROVSKIY, GEORGY. In situ hydrogen production in power-to-food applications. 2022. Diss.

1016. SOMMARSTRÖM, KAARINA. Teachers’ practices of entrepreneurship education in cooperation with companies. 2022. Diss.

1017. KAN, YELENA. Coherent anti-stokes raman scattering spectromicroscopy in biomedical and climate research. 2022. Diss.

1018. MÄNDMAA, SIRLI. Financial literacy in perspective – evidence from Estonian and Finnish students. 2022. Diss.

1019. QORRI, ARDIAN. Measuring and managing sustainable development in supply chains. 2022. Diss.

1020. MARTIKAINEN, SUVI-JONNA. Meaningful work and eudaimonia: contributing to social sustainability in the workplace. 2022. Diss.

1021. MANNINEN, KAISA. Conducting sustainability target-driven business. 2022. Diss.

1022. LI, CHANGYAN. Design, development, and multi-objective optimization of robotic systems in a fusion reactor. 2022. Diss.

1023. CHOUDHURY, TUHIN. Simulation-based methods for fault estimation and parameter identification of rotating machines. 2022. Diss.

1024. DUKEOV, IGOR. On antecedents of organizational innovation: How the organizational learning, age and size of a firm impact its organizational innovation. 2022. Diss.

1025. BREIER, MATTHIAS. Business model innovation as crisis response strategy. 2022. Diss.

1026. FADEEV, EGOR. Magnetotransport properties of nanocomposites close to the percolation threshold. 2022. Diss.

1027. KEPSU, DARIA. Technology analysis of magnetically supported rotors applied to a centrifugal compressor of a high-temperature heat pump. 2022. Diss.

1028. CHAUHAN, VARDAAN. Optimizing design and process parameters for recycled thermoplastic natural fiber composites in automotive applications. 2022. Diss.

1029. RAM, MANISH. Socioeconomic impacts of cost optimised and climate compliant energy transitions across the world. 2022. Diss.

1030. AMADI, MIRACLE. Hybrid modelling methods for epidemiological studies. 2022. Diss.

1031. RAMÍREZ ANGEL, YENDERY. Water-energy nexus for waste minimisation in the mining industry. 2022. Diss.

1032. ZOLOTAREV, FEDOR. Computer vision for virtual sawing and timber tracing. 2022. Diss.

1033. NEPOVINNYKH, EKATERINA. Automatic image-based re-identification of ringed seals. 2022. Diss.

1034. ARAYA GÓMEZ, Natalia Andrea. Sustainable management of water and tailings in the mining industry. 2022. Diss.

1035. YAHYA, MANAL. Augmented reality based on human needs. 2022. Diss.

1036. KARUPPANNAN GOPALRAJ, SANKAR. Impacts of recycling carbon fibre and glass fibre as sustainable raw materials for thermosetting composites. 2022. Diss.

1037. UDOKWU, CHIBUZOR JOSEPH. A modelling approach for building blockchain applications that enables trustable inter-organizational collaborations. 2022. Diss.

1038. INGMAN, JONNY. Evaluation of failure mechanisms in electronics using X-ray imaging. 2022. Diss.

1039. LIPIÄINEN, SATU. The role of the forest industry in mitigating global change: towards energy efficient and low-carbon operation. 2022. Diss.

1040. AFKHAMI, SHAHRIAR. Laser powder-bed fusion of steels: case studies on microstructures, mechanical properties, and notch-load interactions. 2022. Diss.

1041. SHEVELEVA, NADEZHDA. NMR studies of functionalized peptide dendrimers. 2022. Diss.

1042. SOUSA DE SENA, ARTHUR. Intelligent reflecting surfaces and advanced multiple access techniques for multi-antenna wireless communication systems. 2022. Diss.

1043. MOLINARI, ANDREA. Integration between eLearning platforms and information systems : a new generation of tools for virtual communities. 2022. Diss.

1044. AGHAJANIAN MIANKOUH, SOHEIL. Reactive crystallisation studies of CaCO3 processing via a CO2 capture process : real-time crystallisation monitoring, fault detection, and hydrodynamic modelling. 2022. Diss.

1045. RYYNÄNEN, MARKO. A forecasting model of packaging costs : case plain packaging. 2022. Diss.

ISBN 978-952-335-868-3
ISBN 978-952-335-869-0 (PDF)
ISSN 1456-4491 (Print)
ISSN 2814-5518 (Online)
Lappeenranta 2022