TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
COMPUTER ENGINEERING DEPARTMENT

A NEW KNN CLASSIFIER BASED ON THE HARMONIC MEAN OF THE MAJORITY VOTES AND AVERAGE DISTANCES OF THE NEIGHBORS COMBINED WITH ADAPTIVE K-VALUE SELECTION

SELÇUK TOKGÖZ

M.Sc.

ADANA 2023

TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
COMPUTER ENGINEERING DEPARTMENT

A NEW KNN CLASSIFIER BASED ON THE HARMONIC MEAN OF THE MAJORITY VOTES AND AVERAGE DISTANCES OF THE NEIGHBORS COMBINED WITH ADAPTIVE K-VALUE SELECTION

SELÇUK TOKGÖZ

M.Sc.

THESIS ADVISOR
Asst. Prof. Dr. MUSTAFA AÇIKKAR

ADANA, 2023

DECLARATION OF CONFORMITY

In this thesis study, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations, and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all of the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

09/06/2023

Selçuk TOKGÖZ

ABSTRACT

A NEW KNN CLASSIFIER BASED ON THE HARMONIC MEAN OF THE MAJORITY VOTES AND AVERAGE DISTANCES OF THE NEIGHBORS COMBINED WITH ADAPTIVE K-VALUE SELECTION

Selçuk TOKGÖZ

M.Sc., Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Mustafa AÇIKKAR

June 2023, 54 pages

This study aims to improve the classification accuracy of traditional *KNN* by presenting an improved version of the *KNN* algorithm. This thesis proposes *HMAKNN*, an improved *KNN* classifier based on the harmonic mean of the majority voting and average distance phenomena with adaptive and incremental *k*-value selection. Within the purview of this study, two variants of *HMAKNN*, regular and weighted, were designed based on the presence or absence of a weighting mechanism. The names of these variants are $HMAKNN_R$ and $HMAKNN_W$, respectively. These *HMAKNN* classifiers were evaluated on a total of thirty-four data sets, of which eight were synthetic obtained from PRTools and twenty-six were real benchmark data sets from UCI and Kaggle repositories. To determine the classification effectiveness and success of the proposed approaches, they were compared with traditional *KNN* and four well-known weighted *KNN* models. In contrast to other weighting methods, both *HMAKNN* classifiers utilize the constructive interaction between majority voting and average distance, as well as the ability to adaptively modify the *k*-value, thereby substantially enhancing classification accuracy. Based on the results of the classification of real benchmark data sets, $HMAKNN_W$ produced the highest *ACC* value with 83.02%, and $HMAKNN_R$ reached the second-highest value with 82.83%. $HMAKNN_W$ and $HMAKNN_R$ provided an advantage of 2.69% and 2.50% over the other five methods compared, respectively. Furthermore, the classification results of real benchmark data sets indicate that both *HMAKNN* methods statistically outperform other weighted *KNN* methods.

Keywords: adaptive *k*-value selection, *k*-nearest neighbors, majority voting, weighted harmonic mean

ÖZET

UYARLANABİLİR K-DEĞERİ SEÇİMİ İLE BİRLEŞTİRİLMİŞ KOMŞULARIN ÇOĞUNLUK OYLAMASININ VE ORTALAMA UZAKLIKLARININ HARMONİK ORTALAMASINA BAĞLI YENİ BİR KNN SINIFLANDIRICI

Selçuk TOKGÖZ
Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Mustafa AÇIKKAR
Haziran 2023, 54 sayfa

Bu çalışma, *KNN* algoritmasının geliştirilmiş bir versiyonunu sunarak geleneksel *KNN*'nin sınıflandırma doğruluğunu iyileştirmeyi amaçlamaktadır. Bu tez, uyarlanabilir ve artımlı *k*-değeri seçimi ile çoğunluk oylamasının ve ortalama uzaklık harmonik ortalamasına dayanan geliştirilmiş bir *KNN* sınıflandırıcısı olan *HMAKNN*'ı önermektedir. Bu çalışma kapsamında, bir ağırlıklandırma mekanizmasının olup olmamasına bağlı olarak *HMAKNN*'nin sıradan ve ağırlıklı olmak üzere iki farklı türü tasarlanmıştır. Bu metotların adları sırasıyla $HMAKNN_R$ ve $HMAKNN_W$'dir. Bu *HMAKNN* sınıflandırıcıları, sekizi PRTools'tan elde edilen sentetik ve yirmi altısı UCI ve Kaggle depolarından alınan gerçek kıyaslama veri setleri olmak üzere toplam otuz dört veri seti üzerinde değerlendirilmiştir. Önerilen yaklaşımların sınıflandırma etkinliğini ve başarısını belirlemek için geleneksel *KNN* ve dört iyi bilinen ağırlıklı *KNN* modeli ile karşılaştırılmıştır. Diğer ağırlıklandırma yöntemlerinin aksine, her iki *HMAKNN* sınıflandırıcısı da çoğunluk oylaması ve ortalama uzaklık arasındaki sinerjinin yanı sıra *k*-değerini uyarlamalı olarak değiştirme becerisini kullanarak sınıflandırma doğruluğunu önemli ölçüde artırmıştır. Gerçek kıyaslama veri setlerinin sınıflandırılması sonuçlarına göre, $HMAKNN_W$ %83,02 ile en yüksek *ACC* değerini üretirken, $HMAKNN_R$ %82,83 ile ikinci en yüksek değere ulaşmıştır. Karşılaştırılan diğer beş yöntemden elde edilen ortalama sonuçlara göre $HMAKNN_W$ ve $HMAKNN_R$ sırasıyla %2,69 ve %2,50 üstünlük sağlamıştır. Ayrıca, gerçek kıyaslama veri setlerinin sınıflandırma sonuçları, her iki *HMAKNN* yönteminin de istatistiksel olarak diğer ağırlıklı *KNN* yöntemlerinden daha iyi performans gösterdiğini göstermektedir.

Anahtar Kelimeler: ağırlıklı harmonik ortalama, çoğunluk oylaması, *k*-en yakın komşular, uyarlanabilir *k*-değeri seçimi

Dedication

to my dear family

ACKNOWLEDGEMENTS

I would like to thank my esteemed supervisor Asst. Prof. Dr. Mustafa Açıkkar for his guidance, patience and help during this study.

I would also like to thank my precious family for their unwavering support and patience throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

K-nearest neighbors (*KNN*), introduced by (Fix & Hodges, 1951) and modified by (Cover & Hart, 1967) is one of the most preferred supervised machine learning classifiers. Many researchers usually use the *KNN* method to solve different problems in machine learning, data science and data mining. *KNN* can be used in classification (Arslan & Arslan, 2021; Deepa et al., 2022; Narayan, 2020; Singh, Sharma, & Singh, 2022) and regression (Gao, Lin, Sun, & Zeng, 2022; Ho & Yu, 2021; Hossny, Magdi, Soliman, & Hossny, 2020) problems, but it is generally preferred for classification, and we also used *KNN* as a classifier in this study. Among the classification algorithms, the *KNN* algorithm is popular for researchers because of its simplicity and efficiency. The *KNN*-based classification methods are implemented in many fields of practical classification tasks. From the past to the present, it has been used in many different areas, such as classifying various diseases in plant leaves (Hossain, Hossain, & Rahaman, 2019), visual category recognition (H. Zhang, Berg, Maire, & Malik, 2006), text categorization (Jiang, Pang, Wu, & Kuang, 2012), recommendation system (Adeniyi, Wei, & Yongquan, 2016), diagnosis of the diseases (Angel Viji & Rajesh, 2020; Chawla, Kumar, Aggarwal, & Swain, 2018) and scent classification (Müller et al., 2019).

KNN is a lazy learning and non-parametric algorithm. It makes no assumptions for fundamental data and requires learning all data. *KNN* does not learn any distinctive information about the data set and use it to build a generalization model from the training data samples. The training phase simply stores all the data in the training set, and they act as a classifier in the testing phase. When predicting which class the query's label belongs to, *KNN* decides by calculating the nearest neighbors to this query across all training data. Essentially, *KNN* considers points close to each other as belonging to the same class. *KNN* operates in an inductive behavior, assuming that the data set is equally distributed. Therefore, while a reasonable prediction can be made if the data set is balanced, prediction accuracy can be decreased if the training set is unbalanced.

KNN takes only two essential hyperparameters. One of them is the *k* hyperparameter, which specifies how many nearest neighbors to look at. The other hyperparameter is the distance function to be used to measure the proximity of training data points and a query point. After determining the proximity of the *k*-nearest neighbors by the preferred distance function, *KNN*

predicts the class label of the test samples using a simple majority vote among *k*-nearest neighbors. Thus, it is examined which class the new query is more similar to, and an assignment is made accordingly. Although the working principle of *KNN* is straightforward, it is a remarkably effective algorithm.

There are various distance metrics used for proximity measurement in *KNN*. Some of the most preferred ones are Euclidean distance, Minkowski distance, Manhattan distance and Chebychev distance. Among these distance metrics, the most widely used one is the Euclidian distance, which we also use in this study. The Euclidean metric is a basic measure of distance. So, it assumes that all samples are equally essential and measures the distance by giving equal weight to each sample when calculating distances.

The selected *k*-value has different effects on the performance and the label value to be assigned at the classification stage. In other words, a modification in the *k*-value may change the success of the classification results. Traditional *KNN* applies the same *k*-value during the evaluation phase of each query in the testing set. The constant *k*-value may prevent the increase in classification success. Deciding the optimal *k*-value for a better result on the tag of the query is a big challenge in *KNN*. The sensitivity of the *k*-value can easily affect the accuracy of the classification and may lead to degradation of the classification performance of *KNN*. The predicted class label may also change according to the *k*-value in some classifications. Therefore, it is essential to choose the optimum *k*-value to produce the correct output during the estimation phase.

So far, many studies have been proposed to improve the prediction of *KNN*. For this improvement, approaches such as choosing the changeable *k*-value, weighting the neighbors in the voting stage and different versions of distance metrics were applied.

In the literature, various improved algorithms were proposed and utilized to increase the model accuracy of the *KNN* classifier, and some of these suggested studies are as follows:

In classification problems where the samples are not normally distributed and the training set is not large, it is also important to reduce the error rate and improve performance. A new classifier of *KNN* based on local mean vectors was proposed to reduce the negative impact of

KNN's performance especially on small training data in the study (Mitani & Hamamoto, 2006). It relies on locally averaged vectors to make the proposed classifier robust and compared with four different nonparametric classifiers in terms of error rate.

(Pan, Wang, & Ku, 2017) represented an enhanced version of the study in (Mitani & Hamamoto, 2006). After finding the nearest neighbors in the classification problem, local mean vectors are computed for each class label. Then, the harmonic mean is calculated between the query data and the local mean vectors for each class label. The final stage involves assigning the class label with the minimum harmonic mean to the query class. The results on the twenty real-world data sets showed improvement in classification performance when compared to *KNN* and the 8 *KNN* approaches presented.

In the study (Ertuğrul & Tağluk, 2017), a new version of *KNN* was developed to improve the classification performance. Both synthetic data sets and real benchmark data sets were used in the study. The data to be classified were processed both by calculating their distances and by including the dependency information determined by the angle measure between them in the sample space. To make the classification more accurate, mapping was applied in the study and the angles of the neighbors in the feature space were considered. The presented method was shown to improve accuracy and computational cost compared to other well-known methods.

In (Mateos-García, García-Gutiérrez, & Riquelme-Santos, 2019), an evolutionary method is proposed for the *KNN* classifier that weights both nearest neighbors and features. The optimization phase focused on finding two vectors that represent the weight of nearest neighbors and features. Although the weighting of the nearest neighbors was independent of the distance, the nearest neighbor generally had a higher weight. In other words, restrictions were applied in the coding of individuals. The presented method has been assessed on thirty-five data sets from UCI and improvement in classification has been observed.

In the study (Karabulut, Arslan, & Ünver, 2019), it was proposed a new distance metric as weighted similarity to improve the prediction performance of *KNN*. In the presented method, each attribute is weighted, and similarities between samples are weighted accordingly. A weighted similarity matrix was created and used as a measure of proximity. The presented method, evaluated on ten different real data sets, was compared with the standard *KNN* using

Euclidean distance. This algorithm gave satisfactory results on half of the data sets, and it seemed that it could be used as an alternative distance measure for some classification problems.

In (Wang, Wang, Wei, Chen, & Zhang, 2021), a *KNN* model based on weighting multidimensional features is proposed for fault diagnosis of bearings. In the study, first, while creating the feature set, the characteristics of the entropy, frequency and time domains of the signal were extracted. Then, the weights of the features were calculated with the *ReliefF* feature scanning method to create a high-dimensional feature set, and unnecessary features were removed. Finally, bearing failure modes are classified with the weighted *KNN* model. When the classification results are examined, it has been determined that the error recognition rate of the presented model is better than the efficient previous methods.

Several studies have been conducted to change the *k*-value adaptively in order to select the optimal *k*-value for each query in the most effective and correct way, and some of the studies are mentioned below:

In (Bulut & Amasyali, 2017), a hybrid *KNN* model with changeable *k*-value is proposed. The optimal *k*-value was found for each test sample using the *k*-means clustering method. That is, a dynamic *k* was applied to each test sample at the classification stage. The proposed model has been tested on 36 real data sets and classification prediction improved compared to the method in which the same *k*-value was applied to each test sample, but an increase in runtime has been observed as the hybrid model also includes an unsupervised method.

In (Zhong, Guo, Gao, Shan, & Zheng, 2017), a dynamic *k*-value approach is proposed instead of a fixed *k*-value to improve the classification performance of *KNN*. In the presented method, a minimum and maximum interval for dynamic *k* was determined by a proposed algorithm. The class percentages were calculated for each test sample within this interval. It was decided to classify them according to the changes in the trends of percentage curves. The presented method has been tested on a real binary data set and it has been observed that it gives better results than the method using fixed *k*-value.

In (S. Zhang, Li, Zong, Zhu, & Cheng, 2017), it is proposed to create a correlation matrix for *KNN* to reconstruct the test data with the training data to assign different *k*-values to each test sample. The Least-Squares Loss was also used to minimize the error in the rebuilding phase. A graph Laplacian regularization is preferred to preserve the structure of the data. The presented method has learned different *k*-values for each test data and is also robust for data sets with noisy samples. The presented *KNN* classifier model was evaluated on a total of ten binary and three-class data sets, and an improvement was observed when the accuracy values were examined.

S. Zhang, Li, Zong, Zhu, & Wang, 2018 used changeable *k*-value for *KNN*. Instead of using a fixed *k*-value in *KNN*, the most appropriate *k*-value was found and applied. In the training phase, the most optimal *k*-value is found by using the decision tree. In the test phase, it provides a fast operation for the output by using the appropriate *k*-value for each test data. As a result, the presented method provides higher classification accuracy than traditional *KNN*.

In addition to these studies that assign a variable *k*-value, some studies have been proposed in the literature that weight the neighbors to ensure that the effects of neighbors are optimally ameliorated, and some recommended studies are as follows:

Distance-weighted *KNN* (*WKNN*) was introduced by Dudani in (Dudani, 1976). In this study, the nearest neighbor is given a high weight, while the farthest neighbor is given the lowest weight. Thus, those closest to the controlled point have a greater impact on the prediction phase. Also, one of the similarly weighted *KNN*s is Uniform Weighted *KNN* (*UWKNN*). Weights in *UWKNN* are assigned by dividing one by the distances in (Dudani, 1976). The weights are determined inversely with the proximity of neighbors to the query. As the distance of the neighbor increases, its weight approaches zero. This process, which is referred to as the uniform function, is also specified in (Kang & Cho, 2008).

In the study (Gou, Xiong, & Kuang, 2011), a new dual-weighted *KNN* classifier named was conducted by reconciling *WKNN* and *UWKNN*. The presented method was assessed on both artificial data sets and real data sets. In this study, the weights are calculated based on both the proximity of the neighbors and the order of the neighbors. The weight value of the nearest neighbor is determined as one, while the value of the farthest neighbor is zero. Especially, when

the *k*-value is large, the presented method gives better results than previous weighted *KNN* classifiers, but this classifier has only been tested on numerical data.
In (Gou, Du, Zhang, & Xiong, 2012), a novel distance weighted *KNN* is proposed a similar method used in *WKNN*. In the study, it was aimed to improve the classification performance by focusing on the sensitivity of different *k*-values. The difference of proposed algorithm from *WKNN* is that the determined weights have been improved numerically. A dual weight is created in this algorithm and the first one is the same as the weight in *WKNN*, while the second one is newly defined in this study. The presented algorithm had more accurate results compared to similar weighted *KNN*s.

In (Yigit, 2013), it is presented to weight the neighbors of the *KNN* to increase the classification accuracy. The proposed method consists of *KNN* and Artificial Bee Colony (*ABC*) combination model. The step of weighting the neighbors is provided by *ABC* algorithm. The nearest neighbors were given random weights. Then, these neighbors were updated to the optimum weights by *ABC* algorithm. An improvement in classification performance was observed in three data sets of the combined model tested with four data sets from UCI.

The objective of this study is to present a modified weighted *KNN* classifier, abbreviated as *HMAKNN*, based on the harmonic mean of the reciprocal of majority voting and average distance phenomena with adaptive *k*-value selection for each query. The harmonic mean is simply the reciprocal of the arithmetic mean of the reciprocal values of the unit values of the data, and it provides a balanced result by giving less weight to large values and more weight to small values. For each query with a given value of *k*, we compute the harmonic mean of the majority votes, and the average distances of each class to determine as accurately as possible the label to which the query should be assigned. Then the percentage difference between the calculated harmonic means of the top two candidate classes is calculated. In cases where the difference between these two classes is less than a certain threshold value in percentage terms, the label will not be assigned, and the *k*-value is increased by one. The same procedure is repeated until the percentage is greater than the threshold value.

In this study, two different versions of *HMAKNN* are developed depending on whether there is a weighting mechanism. The regular and weighted versions of the *HMAKNN* are named $HMAKNN_R$ and $HMAKNN_W$, respectively. This proposed method was assessed on various

synthetic and real-world benchmark data sets. To exhibit the effectiveness and performance of the proposed *HMAKNN* algorithm on classification problems, it is first compared with its constituent classifier *KNN*, and with other four weighted *KNN* classifiers, *WKNN*, $DWKNN_1$, $DWKNN_2$ and *UWKNN* presented in the literature.

This paper is divided into four sections, and the rest of the paper is organized as follows. Section 2 presents the details of the utilized data sets and gives the methodology along with details of the proposed classification method. In Section 3, the experimental results of all *KNN*-based models are given and compared with each other. Finally, Section 4 summarizes the results of the paper.

2. MATERIALS AND METHODS

2.1. Overview of the Data Sets

34 data sets, eight synthetic and 26 real benchmark data sets, were used to prove the performance of the $HMAKNN_R$ and $HMAKNN_W$ methods presented in this study. The synthetic data sets were obtained by using PRTools. The real-world data sets were taken from the Machine Learning Repository UCI and the Machine Learning and Data Science Community Kaggle. The data sets' names, number of samples, number of features, and number of classes are shown in Table 2.1 for synthetic data sets and Table 2.2 for real data sets. Additionally, Table 2.2 contains the source of each real-world data sets. On the other hand, to demonstrate the generalization skill of the presented method in various areas, attention was paid to the fact that the real data sets were from various fields and consisted of different numbers of samples, features, and classes.

The visualization of the synthetic data sets is shown in Figure 2.1, and the distributions, statistical properties, and geometric complexity (Ertuğrul & Tağluk, 2017) of each synthetic data set are different from each other. The number of instances of all synthetic data sets was determined as 600. The first seven data sets consist of two classes, each containing 300 instances. The MultiClass data set was created by combining the Banana, Lithuanian, Highleyman and GaussianCircle data sets and has eight classes with 75 instances for each class.

Table 2.1. Description of the synthetic data sets

Name	Number of Samples	Number of Features	Number of Classes
Banana	600	2	2
Lithuanian	600	2	2
Highleyman	600	2	2
Gaussian	600	2	2
Spherical	600	2	2
GaussianCircle	600	2	2
Difficult	600	2	2
MultiClass	600	2	8

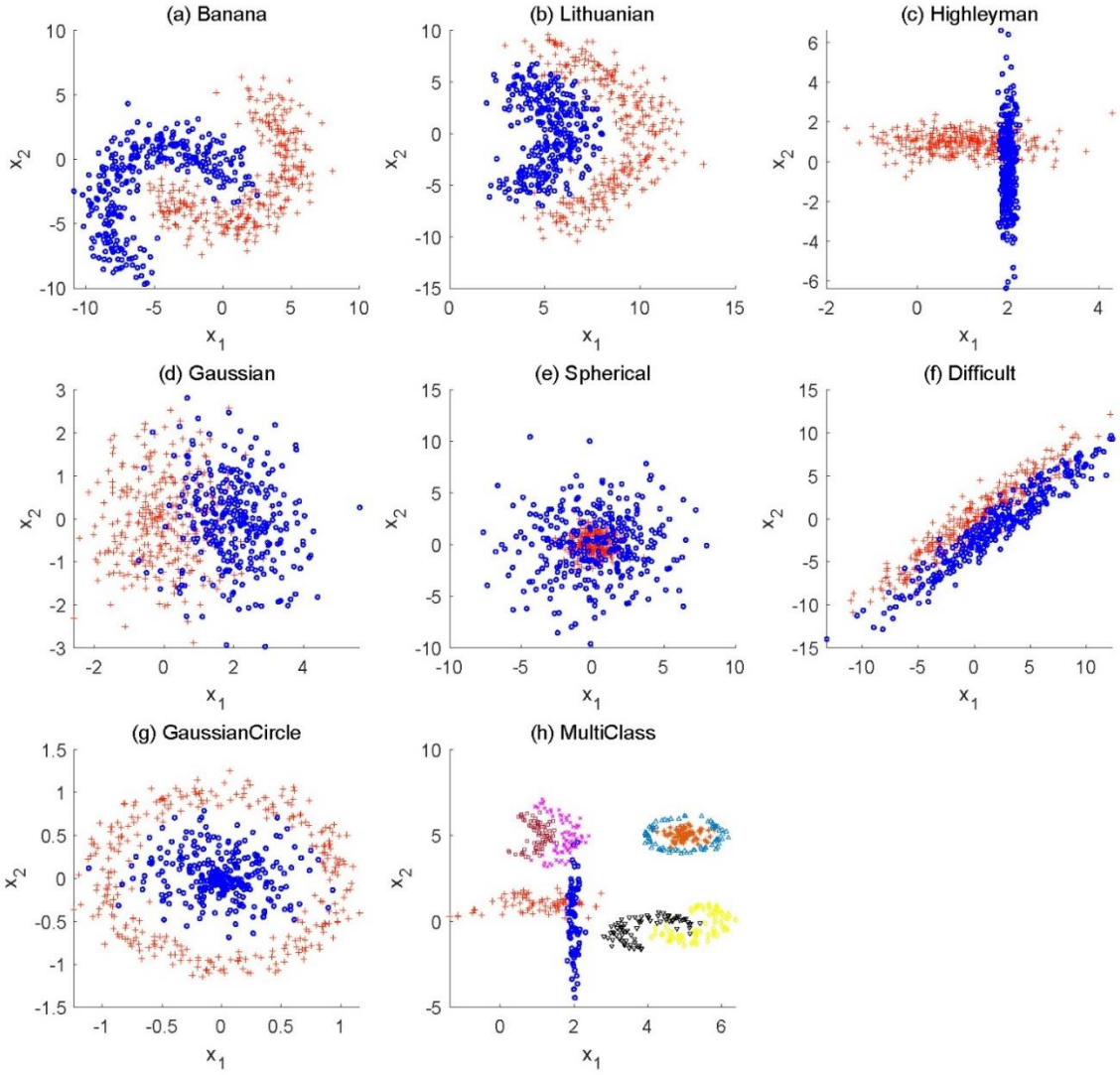

Figure 2.1. Synthetic data sets generated by PRTools

Table 2.2. Description of the real data sets

Name	Number of Samples	Number of Features	Number of Classes	Source
Blood	748	4	2	UCI
BreastCancer	683	9	2	UCI
Diabetes	2000	8	2	Kaggle
DiabeticRetinopathy	1151	19	2	UCI
Habermans	306	3	2	UCI
Ionosphere	351	34	2	UCI
MammographicMass	830	5	2	UCI
PimaIndian	768	8	2	UCI
StatlogHeart	270	13	2	UCI
BalanceScale	625	4	3	UCI
HayesRoth	160	4	3	UCI
Iris	150	4	3	UCI
HealthRisk	1014	6	3	UCI
TeachingAssistant	151	5	3	UCI
WebsitePhishing	1353	9	3	UCI
Wine	178	13	3	UCI
Lymphography	148	18	4	UCI
Drug	200	5	5	Kaggle
BMI	500	3	6	Kaggle
Dermatology	358	34	6	UCI
Glass	214	9	7	UCI
Image	2310	19	7	UCI
Landsat	6435	36	7	UCI
Ecoli	336	7	8	UCI
OptDigits	5620	64	10	UCI
ConnectionistBench	990	12	11	UCI

2.2. Performance Metrics and *k*-fold Cross-validation

Confusion matrix, which is a special layout, is preferred for visualizing the classification performances of the studies in the field of machine learning and displaying them in tabular format. It shows the actual and predicted classes in the classification problem and allows the numbers of true positives (*TP*), true negatives (*TN*), false positives (*FP*) and false negatives (*FN*) to be seen in detail. The correct predictions of the positive samples are represented by *TP*, and the correct predictions of the negative samples are represented by *TN*. On the other hand, incorrect prediction of positive samples is indicated by *FP*, and incorrect prediction of negative samples is indicated by *FN*. Confusion matrix for M-class classifiers is included in Table 2.3.

Table 2.3. Confusion matrix for *M*-class classifier

	C_1	C_2	C_3	…	C_{M-2}	C_{M-1}	C_M
C_1	TN	TN	TN	…	FP	TN	TN
C_2	TN	TN	TN	…	FP	TN	TN
C_3	TN	TN	TN	…	FP	TN	TN
…	…	…	…	…	…	…	…
C_{M-2}	FN	FN	FN	…	TP	FN	FN
C_{M-1}	TN	TN	TN	…	FP	TN	TN
C_M	TN	TN	TN	…	FP	TN	TN

To prove the suitability of our method and compare it with similar *KNN* methods, the prediction performances of all models are calculated with accuracy (*ACC*) metric. For real data sets, F-Score (*FScore*) metric values were also calculated in addition to *ACC*. Although *ACC* is a useful metric, it can be misleading in classification problems with unbalanced classes. Therefore, in addition to *ACC*, when evaluating the performance of models, it would be more reliable to include the *FScore* metric, which is the harmonic mean of Precision (*P*) and Recall (*R*). *P* measures how many of the model's predicted positives are actually positive and *R* measures how many of the positives in the data set are actually predicted positive by the model. The formulas *ACC, R, P* and *FScore* are given in Eqs. (2.1), (2.2), (2.3) and (2.4), respectively.

$$ACC\ (\%) = \frac{TP + TN}{TP + FP + FN + TN} \times 100 \tag{2.1}$$

$$R\ (\%) = \frac{TP}{TP + FN} \times 100 \tag{2.2}$$

$$P\ (\%) = \frac{TP}{TP + FP} \times 100 \tag{2.3}$$

$$FScore\ (\%) = 2 \times \frac{P\ \times R}{P + R} \times 100 \tag{2.4}$$

In machine learning studies, the data set is usually randomly divided into training and testing subsets. To avoid the negative effects of randomness and to obtain more reliable results, using cross-validation increases the generalization skill of the prediction model. Using *k*-fold cross-validation, the raw data set is randomly divided into *k*-equal size subsets. Only one of the subsets is kept as testing data to validate the model. The remaining *k*-1 subsets are used to train the proposed model. This process is done *k* times, using each of the *k* subsets separately as testing data. The best advantage of this method is that all samples in the data set are used as both training and testing data, and each data is used only once as test data.

In our presented study, 10-fold cross-validation was performed to split the data set. The *ACC* and *FScore* values of each model were averaged to obtain an accurate outcome value. In addition to the cross-validation process, the models were run independently 30 times to reduce the negative effects of randomness and obtain the most trustworthy result, and average values of 30 runs were taken for final performance of models.

2.3. Implementation of the *HMAKNN*-based Prediction Model

The working principle of the *HMAKNN* model is shown step by step in Figure 2.2. The first step was to apply a 10-fold cross-validation to obtain new training and test sets from the data set. The optimal parameter values of the *HMAKNN* classifier were obtained by running the *BO* algorithm on the training set in the optimization phase. At this stage, sub-training and sub-testing sets were obtained by applying 5-fold cross-validation to the training set to reduce the negative effects of randomness and increase the generalization ability. For each of the sub-folds, the error rates of the sub-models were calculated and averaged over all to find the final

value of prediction error. The parameters with the lowest prediction error rate were selected and used to develop the training set that predicts the output class label of the testing set. Finally, *ACC*, *R*, *P*, and *FScore* metrics were calculated by comparing the predicted and actual labels to determine the performance of the classifier. These processes shown in Figure 2.2 were applied separately for each data set and *k*-value.

The steps given in Figure 2.2 were applied in the same way for all other compared *KNN*-based models, but the optimization step was skipped because there were no parameters to be optimized for them.

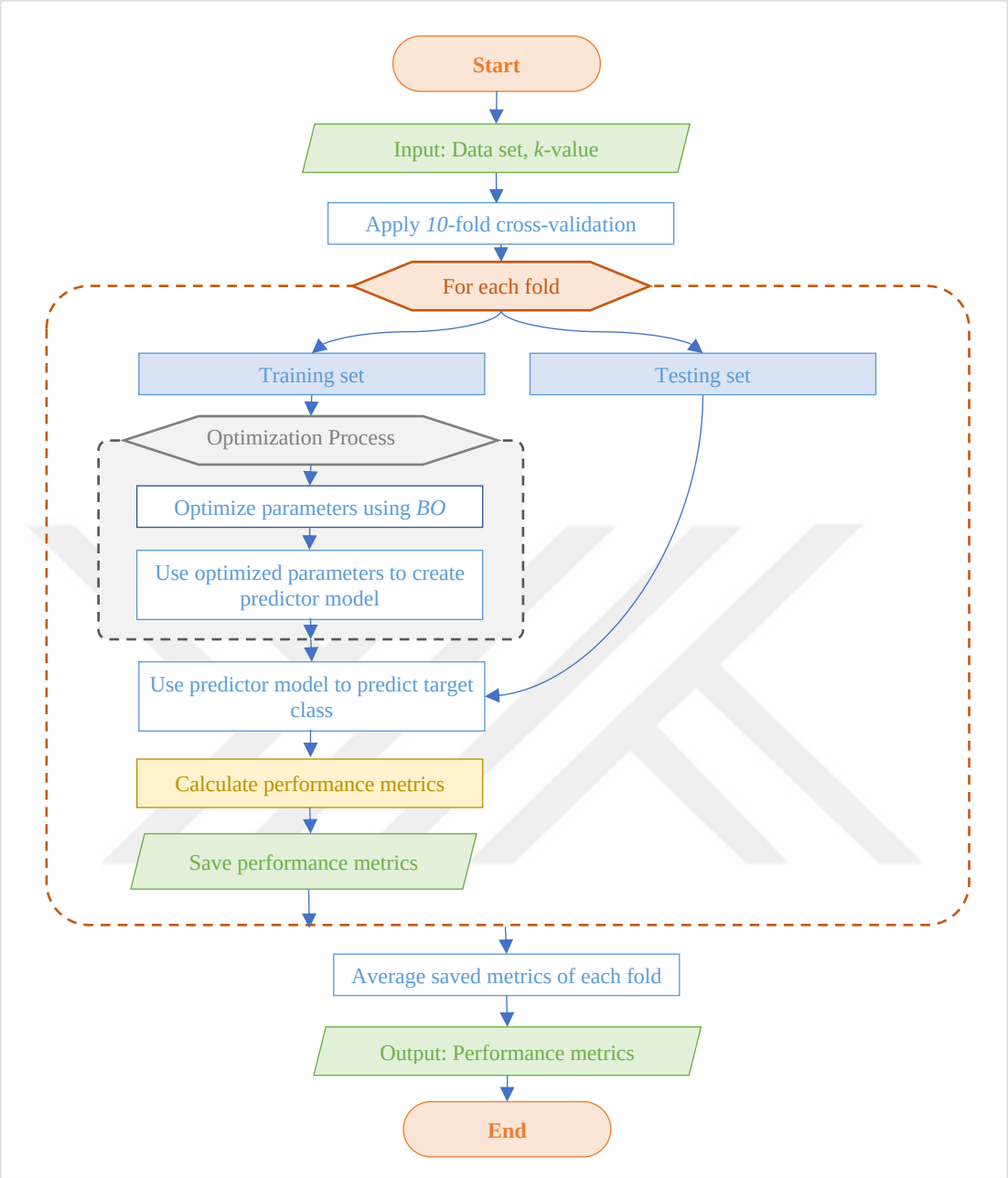

Figure 2.2. Implementation steps of the *HMAKNN*-based prediction models for a single run

2.4. Bayesian Optimization (*BO*)

In recent times, the *BO* algorithm has been utilized as a robust, flexible, and rapid optimization tool that provides optimal and practical solutions for objective functions that are computationally complex, noisy, and costly to evaluate (Elsayad, Nassef, & Al-Dhaifallah, 2022). Consequently, it has also been effectively applied to the parameter optimization of various machine learning methods (Kim, Kwon, Pham, Oh, & Choi, 2022; Lahmiri, Bekiros, & Avdoulas, 2023; Lee et al., 2022; Sultana et al., 2022). Based on this, *BO* was utilized in this

study to determine the optimal values for the *HMAKNN* model parameters. It can be said that *BO*, which is a common method, works quicker than other methods and aims to make fewer mistakes by using more data. In Section 2.7, where *HMAKNN* is introduced, more information is provided about these parameters. The following equation is expressed as a general representation of the *BO* algorithm under the assumption that the solution space for the potential parameters is X and the objective function is f to be utilized to minimize the amount of validation errors.

$$x^* = arg\ min_{x \epsilon X} f(x) \tag{2.5}$$

where x^* is the set of parameters that produces the minimum value of the objective function and x is any value in the solution space X. The objective function is composed of an unidentified structure known as a black box. It is only $f(x)$ that is obtained; none of its derivatives are evaluated. Standard Bayesian optimization demands that every x in X is straightforward to evaluate. Problems that contradict this assumption are referred to as exotic Bayesian optimization problems, and they typically involve noise, parallel evaluation, and evaluation of derivatives.

The *BO* algorithm employs Bayes' Theorem to determine the objective function's minimum or maximum. Since the identity of the objective function is unknown, the Bayesian approach treats it as a random function and assigns precedence. This priority provides information regarding the function's previous behavior. To determine where the next objective function will be evaluated in the solution space, this algorithm employs a surrogate probability model and follows an iterative procedure. At each iteration of this procedure, extant prior knowledge is utilized to update the proxy model (learning phase). The objective function has several prior and post distribution methods. One of them uses Gaussian processes named kriging. Less expensive Parzen-Tree Estimator is the other prevalent technique. *BO* then employs a second function known as the acquisition function to evaluate the objective function, which can analyze the surrogate model to predict the next potential optimal solution. The algorithm then incorporates the new result into the surrogate model for the subsequent iteration. Acquisition functions are very convenient to use for costly problems as they minimize the number of queries. Finally, the algorithm iterates between learning and optimization stages until sufficient data is generated, the surrogate model evolves, and the global optimum is reached (Elsayad et

al., 2022). This strategy allows the *BO* algorithm to generate the optimal solution more efficiently than random selection with a restricted number of evaluations. This theorem can be expressed using the equation below:

$$p(w \mid D) = \frac{p(D \mid w)\, p(w)}{p(D)} \tag{2.6}$$

$p(D \mid w)$ defines the likelihood, $p(w)$ represents the marginal probability, $p(D)$ indicates the prior probability (evidence), and $p(w \mid D)$ shows the posterior probability. This calculation can be simplified by eliminating the normalizing value of *P(D)* and describing the conditional probability as a proportionate quantity below:

$$p(w \mid D) = p(D \mid w)\, p(w) \tag{2.7}$$

2.5. K-Nearest Neighbors (*KNN*)

While *KNN* is a supervised machine learning technique for solving classification and regression problems, it is primarily used for classification problems like the one in this study. In contrast to many machine learning methods, *KNN* is a non-parametric and lazy learning method. Since no training is performed on the training data, *KNN* is referred to as a lazy learning technique. In lieu of conducting calculations during the training phase, *KNN* simply retains the training data for later use. Therefore, a prediction model is not constructed until a query point is performed on the training data. *KNN* is a non-parametric method because it makes no assertions about the distribution of the underlying data.

KNN classification is straightforward to comprehend and implement. Simply put, *KNN* attempts to determine which class label a query point pertains to by examining its neighboring data points. This is why they are referred to as nearest neighbours. In basis, *KNN* uses a voting mechanism to decide the query point's class category. The class label with the greatest number of votes is assigned as the class label of the query point.

Let's describe the implementation steps of the *KNN* algorithm's described logic. Let us first designate the training set as $D_T = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \Re^m$ is the input vector specified in

the *m*-dimensional feature space, y_i is the class label corresponding to x_i, and *N* is the number of training set samples.

The future stages are applied to the given query x^* in order to determine its unknown class label y^*. Before assigning the class label for the query x^*, the number *k* of target neighbors must be detected. For this goal, the Euclidean distance function is used to calculate the distance between the stated query x^* and each x_i in D_T and the formula is below:

$$d(x^*, x_i) = \sqrt{(x^* - x_i)^T (x^* - x_i)} \tag{2.8}$$

$D_T^* = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{k}$ is indicated by ordering the *N* calculated distances in ascending order and then selecting the *k*-nearest to the query x^* (neighbors). Then, the class category of the query x^* is estimated based on the majority voting of its closest neighbors, as shown in the following equation:

$$y^* = \arg max_y \sum_{(\hat{x}_i, \hat{y}_i) \in D_T^*} \delta(y == \hat{y}_i) \tag{2.9}$$

where *y* is the class label, $\hat{y}_i$ is the label of the class of *i*-th closest neighbor, and $\delta = (y == \hat{y}_i)$ is the Dirac delta function that returns one if the class $\hat{y}_i$ of the neighbor $\hat{x}_i$ is equal to y, and zero otherwise.

2.6. Distance-weighted *KNN* Algorithms

KNN assigns an unclassified query the class category with the greatest number of votes among its *k*-nearest neighbors, as is generally acknowledged. But this approach automatically presumes that the *k*-nearest neighbors at various distances from the query point have the same importance in the classification decision and disregards the possibility that the neighbors closest to the query point may have a greater influence on the classification's accuracy. In other words, neighbors that are very close to the query point and neighbors that are significantly further away have the same effect during the classification phase, which can lead to incorrect label assignment. In the classification phase, there are some studies that give weights to the *k*-nearest neighbors to determine the most accurate label to assign to the query point. Although the

weighting functions presented vary slightly from study to study, they are similar and are based on mathematical ratios of significant factors like the distance between neighbors or the quantity of the closest neighbors. The common characteristic of these studies in general is that the neighbors are weighted at a decreasing rate from the nearest neighbor to the farthest neighbor. Through this weighting, among the *k*-nearest neighbors, the closest neighbors have a greater impact on the query label's class decision than the farthest neighbors. In this study, four well-known distance-weighted *KNN* algorithms, called *WKNN*, *UWKNN*, *DWKNN*$_1$, and *DWKNN*$_2$, were utilized to demonstrate our model's classification superiority. These four different algorithms can be examined in more detail in related studies (Dudani, 1976; Gou et al., 2012, 2011).

2.6.1. *WKNN* and *UWKNN*

The concept of weighted neighbors in *KNN* was introduced in (Dudani, 1976). In this study, a certain weight is assigned to the *k*-nearest neighbors according to their distance from the query point. According to this method, which is called *WKNN*, among the *k*-nearest neighbors, the closest ones are given more weight than the furthest ones, making them more important. The following describes the weight of the *i*-th nearest neighbor (w_i) of the query point x^*. The $d(x^*, \hat{x}_i)$ section of the formula can be shortened to $\hat{d}_i$ in order to make w_i simpler to comprehend.

$$w_i = \begin{cases} 1, & if(\hat{d}_k = \hat{d}_1) \\ \dfrac{\hat{d}_k - \hat{d}_i}{\hat{d}_k - \hat{d}_1}, & if(\hat{d}_k \neq \hat{d}_1) \end{cases} \tag{2.10}$$

where d_i^*, d_1^*, and d_k^* denote the distance of the *i*-th nearest neighbor, the closest neighbor, and the *k*-furthest neighbor, respectively, from the query point. The weight is 1 for the nearest neighbor and 0 for the furthest, as the formula indicates.

After the finish of the nearest neighbors weighting procedure, the query point x^*'s label is found and assigned by majority vote using the following equation:

$$y^* = \arg max_y \sum_{(\hat{x}_i, \hat{y}_i) \in D_T^*} w_i \times \delta(y == \hat{y}_i) \tag{2.11}$$

In (Dudani, 1976), in addition to *WKNN*, Dudani presented another *KNN* method called *UWKNN*, in which neighbors are weighted by utilizing the uniform function. In the *UWKNN* approach, the weight w_i is calculated according to the order of proximity to the query point of the *k*-nearest neighbors, regardless of the distance d_i^* between the query instance and the neighbors. In fact, this procedure weights the *k*-nearest neighbors from closest to farthest in decreasing order of significance, consecutively. The expression for the weight w_i for the query point x^* is as follows:

$$w_i = \frac{1}{i}, i = 1, \dots, k \tag{2.12}$$

2.6.2. DWKNN₁

In (Gou et al., 2011), a dual-weighted *KNN* technique, named *DWKNN₁*, is presented to enhance the efficiency and sensitiveness of selecting *k*-values and calculating the voting of the *k*-nearest neighbors. Since the significance of the nearest neighbors plays a very decisive role in the label estimation of the query, a method combining the distances of the neighbors and the uniform function is developed. This dual-weighted method is derived by combining the distances of the *k*-nearest neighbors from the query point and their proximity order; consequently, the importance weights of the nearest neighbors decrease from closest to farthest. The following formula illustrates the closest neighbors $\widehat{w}_i$ dual-weighted method's mathematical calculation.

$$\widehat{w}_i = \begin{cases} 1, & if(\hat{d}_k = \hat{d}_1) \\ \frac{\hat{d}_k - \hat{d}_i}{\hat{d}_k - \hat{d}_1} \times \frac{1}{i}, & if(\hat{d}_k \neq \hat{d}_1) \end{cases} \tag{2.13}$$

After the nearest neighbors are weighted by Eq. (2.13), the label of the query point x^* is assigned to the class label with the majority votes calculated by Eq. (2.11). According to Eq.

(2.13), the *DWKNN₁* method assigns the farthest neighbor among the *k*-nearest neighbors a weight of 0, so it has no impact on the label assignment.

2.6.3. DWKNN₂

In (Gou et al., 2012), named *DWKNN₂*, a redesigned *KNN* algorithm that weights closest neighbors with a new dual distance-weighted function, is proposed to reduce the sensitivity caused by the selection of *k*-value and to improve the influence of neighbors on the determination of the query point's label. *DWKNN₂* has a new dual weight that was calculated by multiplying the weight used in the *WKNN* by a different weight. In *DWKNN₂*, while the weights of the nearest and farthest neighbors have the same weighting as those in *WKNN*, a reduction in the weights of other neighbors takes place. That is, *DWKNN₂* is constructed in such a way that the nearest neighbor has a weight of 1 and the farthest has a weight of 0. Among the nearest neighbors, reducing their weights can minimize the errors caused by outliers on deciding the correct label of the query. As a result, the effects of noisy points, which have a negative impact on the prediction label, can be lessened in terms of classification performance. In the dual distance-weighted function, $\widehat{w}_i$ is defined as in the formula below:

$$\widehat{w}_i = \begin{cases} 1, & if\left(\hat{d}_k = \hat{d}_1\right) \\ \dfrac{\hat{d}_k - \hat{d}_i}{\hat{d}_k - \hat{d}_1} \times \dfrac{\hat{d}_k + \hat{d}_1}{\hat{d}_k + \hat{d}_i}, & if\left(\hat{d}_k \neq \hat{d}_1\right) \end{cases} \tag{2.14}$$

After weighting the nearest neighbors, the label of the x^* query point is assigned by Eq. (2.11) utilizing the weight $\widehat{w}_i$ found from Eq. (2.14).

2.7. Proposed Method (*HMAKNN*)

Before introducing the specifics of our *HMAKNN* classifiers, we explain why we decided to conduct this study in this section. In *KNN*, the *k*-value is an important parameter which defines how many neighbors applied to categorize or estimate the label of a new data point. The advantages and benefits, disadvantages and restrictions of the *k*-value determination and the majority voting procedure are discussed in the following:

The *k*-value allows modeling flexibility by enabling the number of neighbors considered for classification to be modified. This versatility makes *KNN* applicable to a wide variety of

problem areas, and this strategy can be beneficial in situations where the region of decision is complex or nonlinear. By setting a suitable *k*-value, it is possible to manage the trade-off between generalization and complexity of the model, thereby customizing the algorithm to the particular needs of the problem at given.

In addition, the *k*-value provides the *KNN* algorithm strike a balance between overfitting and underfitting. The *KNN* method is easily prone to overfitting when the *k*-value is too small. It may become overly sensitive to regional distinctions in the training data and neglect to detect the fundamental structures in the wider context. In contrast, when the k-value is excessively large, the approach may underfit the data, leading to poor accuracy in prediction. In addition, the value of *k* influences the potential of the *KNN* algorithm to generalize. A smaller *k*-value appears to generate a more complex decision limit, which may overfit the training data and underperform on unknown data. A larger *k*-value, on the other hand, generates a smoother selection boundary, which facilitates generalization and improves performance on unknown data. By selecting a suitable *k*-value, the method can generalize effectively and make accurate estimates on new data samples.

The *k*-value is also essential for dealing with noisy or outlier-prone data sets. *KNN* can minimize the negative effect of noisy or incorrectly labeled instances on the final classification determination by considering multiple neighbors. This durability can contribute to more accurate predictions, especially when working with real-world data sets that frequently contain defective or wrong data. In addition, the efficacy of *KNN* is excessively dependent on the selection of the *k*-value. When the value of *k* is too small, the method becomes more susceptible to data sparsity, noisy, ambiguous, or incorrectly labeled points, resulting in a reduction in classification accuracy. In contrast, when the value of *k* is too large, the method may include anomalies from different categories in the neighborhood, which may also negatively impact the accuracy of classification.

On the other side, the majority voting strategy used in *KNN* can be problematic if the distances of the nearest neighbors differ considerably and the closer ones more accurately indicate the query data's class. In such situations, a straightforward majority vote may not adequately convey the underlying patterns, resulting in suboptimal outcomes. To overcome this limitation,

effective methods such as weighted voting, in which neighbors with closer proximity have greater influence, have been employed.

Aside from this, the *k*-value may cause difficulties when working with unbalanced data sets in which the number of instances in various classes is substantially unequal. A larger *k*-value can lead to the majority class dominating the decision-making process, resulting in biased predictions. In such instances, approaches like weighted *KNN* or resampling methods have been utilized to avoid this constraint.

In summation, the *k*-value in *KNN* provides benefits in terms of modeling versatility, localized decision-making, and noise and anomalous resistance. Nonetheless, it offers difficulties relating to the constraints of the majority voting process and the susceptibility of parameter selection. To take advantage of these improvements and overcome these restrictions, we have developed a new approach that combines several modifications and enhancements to *KNN* in an effort to improve its performance in a variety of situations. The modifications and enhancements to our suggested technique fall under two categories: a new weighted voting function and an adapted *k*-value selection.

In majority voting, *KNN* allocates an uncategorized query the class label with the most votes from among its *k*-nearest neighbors. The new voting mechanism we present computes the vote and average distance of the nearest neighbors equivalent to the class label for each new query point. The harmonic mean of these two values is then calculated to produce a new revised voting value for each class label.

In addition, this study presents a novel adaptive and incremental *k*-value selection method. Using the updated voting values obtained for each class label as described previously, the two top class labels with the lowest voting value are picked. If the percentage difference between the updated voting values of these two class labels is less than or equal to a threshold value (i.e., 5%), the default *k*-value is increased by one, and this process is repeated until the percentage difference exceeds the threshold. When the percentage change is less than or equal to the threshold value, the mechanism aims to reevaluate the query by setting a new *k*-value without determining the query's class label at that time. In order to determine the query's class label with greater confidence and surety, a smoother decision boundary is obtained that allows for a less

jagged or irregular, more continuous and progressive transition between classes, thereby improving classification accuracy.

Moreover, regular, and weighted variants of the *HMAKNN* classifier are presented based on whether the weighting strategy is used in the above-described voting rule.

Before detailing the implementation steps for the *HMAKNN* models, let's review the harmonic mean utilized in our suggested voting rule. The harmonic mean is advantageous because it provides each value in the set equal weight, regardless of its size, and is especially sensitive to small values. In situations where tiny values are crucial to the overall calculation, such as in this study, this can be useful. The harmonic mean is always less than or equal to the arithmetic mean, which is a significant property. This indicates that if there are extreme values in the set, the harmonic mean is less impacted by them than the arithmetic mean, thereby averting undue influence on the outcome.

Calculating the harmonic mean involves taking the reciprocal of each value, computing the arithmetic mean of these reciprocals, and then taking the reciprocal of the resulting value. In terms of mathematics, the harmonic mean of a set of values $X = \{x_1, x_2, \dots, x_n\}$ is given by:

$$H_x = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} \tag{2.15}$$

where *n* is the number of values in the set.

In order to present suggested algorithm, let us represent the votes, the sum of the distances, and the arithmetic means of the distances corresponding to the class labels as $v^* = \{v_1, v_2, \dots, v_M\}$, $s^* = \{s_1, s_2, \dots, s_M\}$ and $a^* = \{a_1, a_2, \dots, a_M\}$ utilizing the sets D_T^* and $\hat{d}^*$, respectively, where M is the number of class labels.

$$v^* = \sum_{(\hat{x}_i, \hat{y}_i) \in D_T^*} \delta(c = \hat{y}_i) \tag{2.16}$$

If, after specifying the set v^*, only a single element of v^* is greater than zero, i.e., all nearest neighbors belong to a single class, then, without additional processing, the category of the query point is assigned to the class label with the majority of votes.

$$s^* = \sum_{(\hat{x}_i, \hat{y}_i) \in D_T^*} \hat{d}_i \times \delta(c = \hat{y}_i) \tag{2.17}$$

To compute each element of the set a^*, the vectors generated by Eqs. (2.16) and (2.17) are incorporated into Eq (2.18). If v_j is equal to zero, indicating a class with no votes, then a_j is not available (*NA*) and no computation is required.

$$a_j = \begin{cases} \frac{s_j}{v_j}, if(v_j \neq 0) \\ NA, \quad if(v_j = 0) \end{cases} \tag{2.18}$$

The proposed algorithm then attempts to calculate the harmonic mean of each value of v^* and a^* to acquire new voting values, denoted by $H^* = \{h_1, h_2, \ldots, h_M, \}$. Nevertheless, for each class label, the model's accuracy improves for large values of v_j and minor values of a_j. In addition, as stated previously, lesser values within a set have a substantial effect on the overall harmonic mean. In accordance with the essence of the harmonic mean, the reciprocal of v_j is used instead of v_j itself to contribute positively to the model's accuracy. Each element of H^* is determined using Eq. (2.19).

$$h_j = \begin{cases} \frac{2}{\frac{1}{1/v_j} + \frac{1}{a_j}}, if(v_j \neq 0) \\ NA, \quad if(v_j = 0) \end{cases} \tag{2.19}$$

As indicated previously, this study presents two variants of *HMAKNN*: *HMAKNN*$_R$ and *HMAKNN*$_W$. Eq. (2.19) is used for *HMAKNN*$_R$ and requires to be updated as follows for *HMAKNN*$_W$:

$$h_j = \begin{cases} \dfrac{2}{\dfrac{w_1}{1/v_j} + \dfrac{w_2}{a_j}}, if(v_j \neq 0) \\ NA, \quad if(v_j = 0) \end{cases} \tag{2.20}$$

where w_1 and w_2 are the respective weights of $1/v_j$ and a_j. The weight w_2 is dependent on the weight w_1 and is calculated using the formula $w_2 = 1 - w_1$.

Then, using Eqs. (2.21) and (2.22), the two class labels with the lowest voting values, h_{min_1} and h_{min_2}, are obtained, respectively. Using Eq. (2.23), the percentage difference h_{pd} between these two voting values is calculated.

$$h_{min_1} = min\{H^* \backslash \{NA\}\} \tag{2.21}$$

$$h_{min_2} = min\left\{H^* \backslash \left\{NA, h_{min_1}\right\}\right\} \tag{2.22}$$

$$h_{pd} = \frac{\left|h_{min_2} - h_{min_1}\right|}{h_{min_1}} \times 100 \tag{2.23}$$

If h_{pd} is under or equal to the value of the threshold t_p, the default *k*-value is increased by one, and the process is repeated until h_{pd} becomes higher than t_p. If h_{pd} exceeds t_p, the query point x^* is classified using Eq. (2.24).

$$y^* = arg\ find_c\{h_j \in H^* : h_j = h_{min_1}\} \tag{2.24}$$

3. RESULTS AND DISCUSSIONS

Based on the harmonic mean of the majority vote and average distance with adaptive and incremental *k*-value selection, *HMAKNN*, an enhanced version of the traditional *KNN* algorithm, was suggested in the current study. The classification performance of the presented approach was primarily compared to that of the *KNN* technique to determine the performance of the proposed method's predictions. In addition, for comparative purposes, the performances of the *HMAKNN* models were compared to those of models constructed using other weighted *KNN* classifiers, namely *WKNN*, $DWKNN_1$, $DWKNN_2$, and *UWKNN*, on synthetic and real-world data sets.

In this study, data sets were subjected to 10-fold cross-validation to ensure the generalizability of the provided results. In addition to this process, the flowchart shown in Figure 2.2 was executed 30 times separately for each data set to reduce random interference to maximize the stability and reliability of the offered findings. Furthermore, random data segments of all folds (test/train division) were generated for each data set and run before the development of the prediction models. Consequently, it was verified that all models were run on exactly identical data segments, and more credible results were generated for comparison purposes. The last value of each model's performance metrics was determined by averaging the outcomes of each fold and then each run.

3.1. Parameter Settings

The parameters utilized by all *KNN*-based algorithms in this study are detailed in Table 3.1. Through trial-and-error, the *k*-values of all *KNN* approaches were optimized by comparing the performance of models with varying *k*-values. Furthermore, the Section 2.4 described *BO* algorithm was used to optimize the parameters of the suggested *HMAKNN* techniques.

Table 3.1. Overview of the parameters for the *KNN*-based prediction models

Method	**Parameter Description**	**Value**
For all *KNN*-based methods	Distance function	Euclidian
	The number of nearest neighbors (*k*)	[3-15]
Both *HMAKNN* methods	Percent threshold (t_p)	[0.01,5]
$HMAKNN_w$	Weights (w_1, w_2)	[eps*, 1]

*eps is the smallest positive real number, i.e. 2.2×10^{-10}, that can be used as a constant in MATLAB

3.2. Experimental Results

Tables Table **3.2**, Table **3.3**, and Table **3.4** show the mean and standard deviation of 30 runs for *ACC*, *R*, and *P* values of the *KNN*-based models on the synthetic data sets, whereas Tables Table **3.7**, Table **3.8**, and Table **3.9** show the same for the real-world data sets. The *FScore* values for the real data set are also in Table 3.10.

In this study, the performances of all *KNN*-based classifiers used were assessed using eight synthetic and twenty-six real-world data sets described in Section 2.1. The average and standard deviations of the 30-runs for *ACC*, *R*, and *P* values of the *KNN*-based models are given in Tables Table **3.2**, Table **3.3**, and Table **3.4** for the synthetic data sets and in Tables Table **3.7**, Table **3.8**, Table **3.9**, and Table **3.10** (*FScore*) for the real-world data sets, respectively. In addition, the *k*-value that attained the highest *ACC* value is shown within parenthesis in Tables Table **3.2** and Table **3.7**. Aside from this, it is sometimes inadequate to directly compare classification performance results with one another to determine the model's efficiency. Also, the outcomes of each technique should be statistically checked. In accordance with this purpose, to examine if the performance of the *KNN*-based classifiers differed substantially from one another, we preferred to utilize a non-parametric Friedman's test. The use of non-parametric tests is necessitated by the high sensitivity of the conditions required for applying parametric tests, particularly the sphericity condition (Mateos-García et al., 2019). Using Friedman's test yielded the ranking and statistics values of each model, which are shown in Tables Table **3.5** and Table **3.11** for synthetic and real data sets, respectively. In addition, the average ranking findings for each method are computed and shown in increasing order in Tables Table **3.6** and Table **3.12**, respectively, for synthetic and real data sets. It is essential to keep in mind that 1 and 7 are,

respectively, the best and worst ranking values.

For comparison of results from all models, the best value for each data set is specified in bold in Tables Table **3.2**, Table **3.3**, Table **3.4**, Table **3.7**, Table **3.8**, Table **3.9**, and Table **3.10**. Moreover, two best models according to the averages across data sets are highlighted in bold. The model with the best average value in Tables Table **3.7**, Table **3.8**, Table **3.9**, and Table **3.10** is underlined.

While comparing the efficacy of the proposed technique to that of the other five models, it is important to consider some criteria. Higher values of *ACC*, *R*, *P*, and *FScore*, which are used to measure the prediction performance of the model, indicate that the model performs well, whereas lower values of the average ranking from Friedman's test show that the model works statistically well.

In the next sub-sections, the outcomes produced from the synthetic and real data sets are examined and discussed in detail.

3.2.1. Synthetic data sets

With regard to the outcomes of the trials conducted on synthetic data sets (shown in Table 3.2), the traditional *KNN* technique yielded more accurate predictions in three of the eight data sets. Likewise, the variants of the *HMAKNN* approach that we suggested yielded superior results for three of the eight data sets. Considering the average model accuracies obtained by *KNN* approaches on all synthetic data sets, the traditional *KNN*-based models are the most accurate, while the suggested $HMAKNN_W$-based models are the second-most accurate. Following these two models were the *WKNN*, $DWKNN_2$, *UWKNN*, $HMAKNN_R$, and $DWKNN_1$ approaches, in that order. There is less than 1% difference between the *ACC* values of the traditional *KNN* with the best value and the $DWKNN_1$ with the worst value. Based on the minor differences in accuracies, it can be concluded that the prediction performances of the classifiers utilized in this study are closely similar for synthetic data sets.

Tables Table **3.3** and Table **3.4** display the values of the *R* and *P* metrics calculated with the outcomes of synthetic data sets. The standard *KNN* model performs better in 6 of 8 data sets for the *R* metric and in 4 of 8 data sets for the *P* metrics, based on the results in these tables.

Examining the average *R*, and *P* values of the models, the standard *KNN* yields the highest value, while $HMAKNN_W$ yields the second-highest value. There were no major differences between the two best average values. For *R* values, traditional *KNN* and $HMAKNN_W$ models are followed by *WKNN*, $HMAKNN_R$, $DWKNN_2$, *UWKNN*, and $DWKNN_1$. For *P* values, following traditional *KNN* and $HMAKNN_W$ are *WKNN*, $DWKNN_2$, $HMAKNN_R$, *UWKNN*, and $DWKNN_1$ methods, in that sequence.

In addition to the evaluations, Friedman's test was utilized to statistically analyze the effectiveness of the two *HMAKNN* models with the results of five additional *KNN*-based approaches. According to the chi-square statistics and *p*-values for each data set, the null hypothesis (no statistical difference between competing classifiers) could be rejected with 6 degrees of freedom at $\alpha = 0.05$ when Tables Table **3.5** and Table **3.6** are examined together. According to the non-parametric Friedman's test assessment results for synthetic data sets, the performances of all *KNN*-based models were statistically different. In addition, based on the average rankings of all *KNN*-based classifiers, the $HMAKNN_W$ classifier we presented was the second-best performing model, behind traditional *KNN*. The following classifiers became *UWKNN*, *WKNN*, $DWKNN_2$, $HMAKNN_R$, and $DWKNN_1$. The average rankings result in a similar order to performance metrics, although the *UWKNN* model appears to have ascended to third position according to the ranking results. Our $HMAKNN_R$ model scored sixth, while the $HMAKNN_W$ model we presented ranked second, demonstrating its efficacy.

By evaluating synthetic data sets, it is possible to conclude that, despite statistically distinct findings, algorithms work comparably and achieve model accuracy that differs only slightly.

Table 3.2. Achieved *ACC* (%) for synthetic data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Banana	98.81 ± 0.13 (10)	**98.89 ± 0.19 (11)**	98.84 ± 0.10 (7)	98.80 ± 0.13 (4)	98.82 ± 0.22 (5)	98.80 ± 0.22 (3)	98.80 ± 0.08 (8)
Lithuanian	**97.02 ± 0.33 (3)**	96.74 ± 0.34 (12)	96.82 ± 0.27 (13)	96.88 ± 0.25 (13)	96.89 ± 0.29 (15)	96.87 ± 0.26 (13)	96.97 ± 0.20 (16)
Highleyman	89.76 ± 0.64 (14)	89.94 ± 0.47 (9)	**90.44 ± 0.48 (14)**	90.28 ± 0.69 (15)	90.03 ± 0.53 (11)	90.22 ± 0.62 (15)	90.38 ± 0.45 (15)
Gaussian	82.59 ± 0.51 (15)	82.72 ± 0.49 (15)	**82.92 ± 0.43 (13)**	82.23 ± 0.60 (15)	80.74 ± 0.73 (15)	81.84 ± 0.63 (15)	81.91 ± 0.53 (12)
Spherical	84.07 ± 0.48 (15)	**84.09 ± 0.42 (14)**	84.06 ± 0.38 (15)	83.71 ± 0.44 (15)	82.07 ± 0.58 (15)	83.74 ± 0.50 (15)	83.36 ± 0.61 (14)
Difficult	83.83 ± 0.44 (14)	83.76 ± 0.51 (14)	**84.06 ± 0.47 (15)**	83.45 ± 0.44 (15)	82.95 ± 0.66 (15)	83.58 ± 0.50 (12)	84.01 ± 0.65 (14)
GaussianCircle	95.94 ± 0.27 (13)	96.36 ± 0.25 (3)	96.47 ± 0.21 (3)	96.47 ± 0.21 (7)	96.49 ± 0.22 (13)	96.47 ± 0.18 (8)	**96.61 ± 0.23 (5)**
MultiClass	94.98 ± 0.24 (7)	95.42 ± 0.31 (3)	95.55 ± 0.27 (3)	95.53 ± 0.29 (6)	95.10 ± 0.28 (9)	**95.63 ± 0.27 (6)**	95.13 ± 0.30 (5)
Average *ACC*	90.876 ± 0.382	**90.988 ± 0.372**	**91.145 ± 0.327**	90.919 ± 0.384	90.385 ± 0.440	90.894 ± 0.397	90.896 ± 0.381

Table 3.3. Achieved *R* (%) for synthetic data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Banana	99,59 ± 0,15	99,52 ± 0,19	**99,65 ± 0,09**	99,59 ± 0,16	99,31 ± 0,09	99,55 ± 0,18	99,32 ± 0,05
Lithuanian	96,65 ± 0,40	96,26 ± 0,46	**96,71 ± 0,36**	96,47 ± 0,24	96,31 ± 0,44	96,36 ± 0,26	96,38 ± 0,27
Highleyman	86,82 ± 0,66	88,37 ± 0,66	87,93 ± 1,02	**88,71 ± 0,53**	88,34 ± 0,54	88,40 ± 0,54	86,84 ± 0,55
Gaussian	80,85 ± 0,50	81,13 ± 0,62	**83,24 ± 0,79**	80,76 ± 0,87	79,03 ± 0,72	80,41 ± 0,94	80,22 ± 0,60
Spherical	92,59 ± 0,70	92,33 ± 0,82	**94,04 ± 0,68**	91,54 ± 0,67	89,41 ± 0,84	91,53 ± 0,76	91,29 ± 0,70
Difficult	84,37 ± 0,73	84,36 ± 0,76	**86,53 ± 0,71**	84,03 ± 0,65	82,90 ± 0,75	83,64 ± 0,66	84,26 ± 0,67
GaussianCircle	99,11 ± 0,35	99,03 ± 0,45	**99,62 ± 0,14**	99,07 ± 0,35	98,40 ± 0,27	98,91 ± 0,29	98,79 ± 0,31
MultiClass	95,06 ± 0,37	95,48 ± 0,36	95,63 ± 0,35	95,63 ± 0,35	95,17 ± 0,41	**95,72 ± 0,36**	95,23 ± 0,38
Average *R*	91,880 ± 0,483	**92,060 ± 0,540**	**92,918 ± 0,516**	91,976 ± 0,476	91,107 ± 0,508	91,814 ± 0,497	91,542 ± 0,442

Table 3.4. Achieved *P* (%) for synthetic data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Banana	98.46 ± 0.25	98.42 ± 0.35	98.36 ± 0.19	**98.69 ± 0.25**	98.68 ± 0.24	**98.69 ± 0.25**	98.36 ± 0.19
Lithuanian	97.56 ± 0.39	97.33 ± 0.49	97.57 ± 0.39	97.26 ± 0.40	**97.80 ± 0.39**	97.33 ± 0.32	97.54 ± 0.37
Highleyman	95.93 ± 0.89	95.68 ± 0.81	**96.66 ± 0.87**	95.93 ± 1.09	93.75 ± 0.96	95.80 ± 0.97	95.87 ± 0.74
Gaussian	84.05 ± 0.78	84.25 ± 0.84	**84.56 ± 0.78**	83.24 ± 0.87	81.86 ± 1.15	82.83 ± 0.87	83.08 ± 0.81
Spherical	79.24 ± 0.60	79.47 ± 0.53	**79.51 ± 0.46**	79.36 ± 0.59	78.07 ± 0.50	79.32 ± 0.45	79.00 ± 0.61
Difficult	83.43 ± 0.74	83.77 ± 0.89	**84.18 ± 0.83**	83.72 ± 0.63	82.93 ± 0.82	83.78 ± 0.81	83.81 ± 0.86
GaussianCircle	94.31 ± 0.37	94.82 ± 0.31	94.86 ± 0.3	94.87 ± 0.28	94.81 ± 0.36	94.85 ± 0.24	**94.98 ± 0.38**
MultiClass	95.13 ± 0.46	95.57 ± 0.46	95.70 ± 0.46	95.74 ± 0.51	95.31 ± 0.38	**95.81 ± 0.51**	95.37 ± 0.48
Average *P*	91.014 ± 0.559	**91.163 ± 0.584**	**91.425 ± 0.535**	91.100 ± 0.578	90.401 ± 0.601	91.052 ± 0.552	91.000 ± 0.553

Table 3.5. Achieved ranking and statistics for synthetic data sets

Name	Ranking							Statistics	
	$HMAKNN_R$	$HMAKNN_W$	KNN	$WKNN$	$DWKNN_1$	$DWKNN_2$	$UWKNN$	Chi-Square	p-value
Banana	4.07	3.03	3.67	4.45	4.05	4.38	4.35	13.27	3.89×10^{-2}
Lithuanian	3.28	5.00	4.45	3.98	3.97	4.15	3.17	17.41	7.90×10^{-3}
Highleyman	5.58	4.88	2.82	3.50	4.58	3.85	2.78	45.60	3.66×10^{-8}
Gaussian	2.95	2.38	1.92	3.82	6.83	5.23	4.87	121.10	9.55×10^{-24}
Spherical	2.78	2.60	2.40	4.18	7.00	3.85	5.18	109.71	2.34×10^{-21}
Difficult	3.40	4.03	2.02	5.15	6.25	4.45	2.70	84.39	4.42×10^{-16}
GaussianCircle	6.62	4.58	3.58	3.70	3.43	3.83	2.25	79.75	4.02×10^{-15}
MultiClass	6.13	3.50	2.50	2.85	5.58	1.95	5.48	120.35	1.38×10^{-23}

Table 3.6. Average ranking of each *KNN*-based methods for all synthetic data sets

Method Name	Average Ranking*
KNN	2.919
$HMAKNN_W$	3.752
UWKNN	3.848
WKNN	3.954
$DWKNN_2$	3.963
$HMAKNN_R$	4.352
$DWKNN_1$	5.213

*Sorted in ascending order

3.2.2. Real data sets

As stated previously, 26 real-world benchmark data sets were utilized to evaluate the performance of the two proposed *HMAKNN* models against that of standard *KNN* and four other weighted *KNN* models.

The findings obtained because of the comparisons of the *ACC* values of all *KNN*-based models shown in Table 3.7 are listed below:

- The presented $HMAKNN_R$ and $HMAKNN_W$ classifiers achieved the highest prediction rate in 9 and 4 data sets, respectively, and showed their superiority by giving better results in 13 of 26 data sets than the other five models compared.
- In five of the 26 data sets, called Blood, Diabetes, HayesRoth, HealthRisk, and TeachingAssistant, the performance of the suggested classifiers was significantly superior to that of the other four weighted *KNN* predictors. Both *HMAKNN* classifiers exhibited an average *ACC* advantage of 6.6%, 16.7%, 10.3%, 13.3%, and 20.9% over the other four competing classifiers, *WKNN*, $DWKNN_1$, $DWKNN_2$, and *UWKNN*, for the aforementioned five data sets.
- Also, it was considerably superior to traditional *KNN* for the four data sets except Blood.
- In 4 of the 26 data sets, named Diabetes, HayesRoth, HealthRisk, and TeachingAssistant, the average *R*, *P* and *FScore* values of the *HMAKNN* models outscored the averages of the other four competing methods (*WKNN*, $DWKNN_1$, $DWKNN_2$, and *UWKNN)* by 11.08 %, 7.92 %, 13.57 %, and 21.19 % for *R*, 29.76%,

6.24%, 13.88%, and 21.97% for *P*, 22.16%, 9.36%, 14.38%, and 21.37% for *FScore*, respectively.

- With an average *ACC* of 83.023%, the $HMAKNN_W$ classifier was the best model, while the $HMAKNN_R$ classifier was its nearest follower, with an average *ACC* of 82.826%. *KNN*, $DWKNN_2$, *WKNN*, $DWKNN_1$, and *UWKNN* followed with average *ACC*s of 80.979%, 80.694%, 80.632%, 80.021%, and 79.322%, respectively.

The average *ACC* of the $HMAKNN_R$ classifier, which generates the highest *ACC* value in 9 data sets, is predicted to be higher than that of the $HMAKNN_W$ classifier. However, the average *ACC* of the $HMAKNN_W$ classifier is higher.

Examining Table 3.8, which contains the *R* values, reveals that the two models we presented had the best two averages among the *KNN*-based models' average scores. $HMAKNN_W$ reached the highest average *R* value with 81.697%, followed by $HMAKNN_R$ with 81.550%.

Similarly, in Table 3.9, which includes *P* values, the highest one among the best average *P* values of the classifiers was $HMAKNN_W$ with 81.271%, while the second highest was $HMAKNN_R$ with 81.060%.

In Table 3.10, which shows the *FScore* values of all *KNN*-based models, the proposed *HMAKNN* classifiers are positioned in the top two rankings. The $HMAKNN_R$ and $HMAKNN_W$ models achieved the highest *FScore* values with 9 and 4 out of 26 data sets, respectively. Additionally, in the evaluation of the average *FScore* values of all *KNN*-based classifiers, $HMAKNN_W$ had the highest value with 80.317%, followed by $HMAKNN_R$ with 80.099%. *KNN*, $DWKNN_2$, *WKNN*, $DWKNN_1$, and *UWKNN* had average *FScores* of 78.039%, 77.386%, 77.279%, 76.497%, and 75.299%, respectively. In addition, the *FScores* of $HMAKNN_W$ and $HMAKNN_R$ averaged 80.21%, while the average of the other five classifiers compared was 76.90%. Comparing the reported performances of the *KNN*-based models in terms of the measure *FScore*, it can be shown that *FScore* results are equivalent to and coincide with *ACC* values. Based on these findings, it is possible to assert that both proposed approaches produce accurate prediction results even for data sets with an imbalanced class distribution.

In addition to the analysis of *ACC*, *P*, *R*, and *FScore* metrics, a non-parametric Friedman's test was also performed on real data sets to establish whether the performance of *KNN*-based models differed considerably and to illustrate and validate the superiority of both presented *HMAKNN* methods over other compared methods. Based on the rankings, chi-square statistics, and *p*-values for each data set in Table 3.11, the null hypothesis may be rejected at $\alpha = 0.001$ with 6 degrees of freedom. Examining the average rankings of the classifiers listed in ascending order in Table 3.12 indicated that the *HMAKNN* classifiers, $HMAKNN_W$ which achieved an average ranking of 3.219 and $HMAKNN_R$ which achieved an average ranking of 3.465, likewise obtained statistically superior performance compared to the other competing classifiers, respectively. Following both *HMAKNN* classifiers were the competing classifiers $DWKNN_2$, *WKNN*, *KNN*, $DWKNN_1$, and *UWKNN*, with average rankings of 3.749, 3.901, 4.262, 4.681, and 4.724, respectively.

Table 3.7. Achieved *ACC* (%) values for real data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Blood	74.32 ±0.70 (5)	74.07 ±0.47 (3)	**76.59 ±0.60 (5)**	67.44 ±0.72 (5)	66.15 ±0.70 (5)	67.37 ±0.63 (5)	69.58 ±0.86 (4)
BreastCancer	**97.29 ±0.21 (5)**	97.20 ±0.21 (6)	97.28 ±0.23 (5)	97.19 ±0.16 (10)	96.63 ±0.24 (15)	97.24 ±0.22 (9)	97.10 ±0.17 (13)
Diabetes	**98.74 ±0.32 (3)**	98.73 ±0.29 (3)	87.96 ±0.55 (3)	85.30 ±0.29 (3)	85.30 ±0.29 (3)	85.30 ±0.29 (3)	72.40 ±0.38 (15)
DiabeticRetinopathy	66.31 ±0.66 (9)	66.28 ±0.48 (15)	**66.43 ±0.60 (8)**	66.28 ±0.54 (15)	63.39 ±0.65 (15)	66.26 ±0.52 (15)	65.55 ±0.58 (15)
Habermans	**75.57 ±0.89 (13)**	75.41 ±0.85 (13)	75.56 ±0.93 (13)	74.90 ±0.93 (15)	73.23 ±1.05 (15)	74.60 ±1.02 (15)	74.89 ±1.04 (15)
Ionosphere	84.80 ±0.54 (3)	86.00 ±0.96 (3)	84.51 ±0.53 (3)	86.53 ±0.48 (3)	**86.89 ±0.57 (13)**	86.53 ±0.48 (3)	85.96 ±0.52 (3)
MammographicMass	78.30 ±0.49 (14)	78.20 ±0.49 (14)	80.78 ±0.53 (7)	79.33 ±0.69 (15)	79.66 ±0.56 (15)	79.48 ±0.65 (15)	**81.05 ±0.58 (13)**
PimaIndian	**74.55 ±0.56 (15)**	74.54 ±0.58 (15)	**74.55 ±0.56 (15)**	73.43 ±0.66 (15)	71.10 ±0.54 (15)	73.26 ±0.71 (15)	73.08 ±0.70 (15)
StatlogHeart	66.77 ±1.56 (7)	66.46 ±1.73 (7)	66.77 ±1.56 (7)	**66.96 ±1.35 (10)**	63.30 ±1.14 (15)	66.46 ±1.43 (15)	66.80 ±1.18 (11)
BalanceScale	89.09 ±0.43 (9)	**89.19 ±0.40 (13)**	88.42 ±0.40 (13)	87.37 ±0.63 (15)	81.23 ±0.58 (15)	87.35 ±0.63 (15)	84.91 ±0.49 (15)
HayesRoth	**76.04 ±1.50 (3)**	75.42 ±1.87 (3)	69.23 ±2.21 (3)	65.46 ±1.79 (3)	66.83 ±2.01 (3)	65.46 ±1.79 (3)	64.13 ±2.38 (3)
Iris	**97.07 ±0.75 (13)**	96.87 ±0.84 (12)	97.02 ±0.92 (15)	96.78 ±0.84 (14)	95.93 ±0.27 (13)	96.76 ±0.69 (15)	96.02 ±0.21 (3)
HealthRisk	**85.02 ±0.47 (3)**	84.82 ±0.45 (3)	73.51 ±0.68 (3)	73.49 ±0.75 (3)	73.90 ±0.72 (3)	73.50 ±0.76 (3)	65.39 ±0.76 (3)
TeachingAssistant	**59.18 ±2.23 (4)**	58.77 ±2.72 (4)	43.15 ±1.59 (5)	38.57 ±1.97 (8)	38.99 ±1.86 (3)	39.03 ±1.79 (3)	35.58 ±1.52 (6)
WebsitePhishing	87.07 ±0.37 (10)	86.86 ±0.45 (10)	**88.09 ±0.41 (6)**	86.55 ±0.30 (3)	85.57 ±0.30 (3)	86.56 ±0.30 (3)	86.92 ±0.36 (11)
Wine	72.37 ±1.36 (3)	74.67 ±1.88 (3)	71.75 ±1.23 (3)	74.82 ±1.24 (3)	**76.33 ±1.39 (3)**	76.22 ±1.34 (3)	72.92 ±1.49 (3)
Lymphography	77.77 ±1.54 (6)	77.98 ±1.58 (6)	77.18 ±1.34 (7)	79.27 ±1.29 (3)	78.81 ±1.53 (10)	79.27 ±1.29 (3)	**79.45 ±1.28 (8)**

Table 3.7. Achieved *ACC* (%) values for real data sets (Continued)

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Drug	73.07 ±1.54 (3)	**73.38 ±1.66 (3)**	70.87 ±1.50 (4)	72.63 ±1.41 (4)	73.22 ±1.08 (8)	73.03 ±1.17 (4)	72.65 ±1.09 (4)
BMI	**90.75 ±0.60 (4)**	90.51 ±0.62 (4)	90.14 ±0.56 (3)	89.69 ±0.59 (6)	90.30 ±0.61 (9)	89.95 ±0.55 (6)	90.07 ±0.53 (8)
Dermatology	89.71 ±0.66 (3)	89.55 ±0.77 (3)	88.78 ±0.70 (3)	**90.41 ±0.81 (4)**	90.28 ±0.75 (13)	90.20 ±0.79 (4)	90.26 ±0.71 (6)
Glass	70.08 ±1.32 (3)	71.39 ±1.28 (3)	67.51 ±1.17 (3)	71.69 ±1.03 (3)	**72.71 ±1.18 (9)**	71.69 ±1.03 (3)	68.27 ±1.08 (3)
Image	95.89 ±0.17 (4)	**96.16 ±0.19 (3)**	95.33 ±0.18 (3)	95.74 ±0.19 (5)	95.79 ±0.13 (3)	95.83 ±0.19 (5)	94.72 ±0.14 (3)
Landsat	91.22 ±0.18 (4)	91.21 ±0.15 (4)	91.01 ±0.14 (3)	91.37 ±0.13 (6)	91.26 ±0.13 (15)	91.36 ±0.14 (6)	**91.39 ±0.13 (4)**
Ecoli	85.81 ±0.44 (15)	86.68 ±0.64 (7)	86.79 ±0.54 (7)	87.06 ±0.57 (11)	85.57 ±0.68 (15)	**87.16 ±0.57 (12)**	86.55 ±0.74 (10)
OptDigits	98.89 ±0.07 (4)	98.92 ±0.07 (4)	98.84 ±0.05 (3)	98.94 ±0.05 (6)	98.92 ±0.05 (14)	**98.96 ±0.06 (6)**	98.92 ±0.05 (8)
ConnectionistBench	97.82 ±0.27 (3)	**99.33 ±0.29 (3)**	97.40 ±0.38 (3)	99.23 ±0.23 (3)	99.26 ±0.23 (3)	99.25 ±0.23 (3)	97.82 ±0.35 (3)
Average *ACC*	**82.826 ±0.763**	**83.023 ±0.844**	80.979 ±0.772	80.632 ±0.755	80.021 ±0.739	80.694 ±0.741	79.322 ± 0.743

Table 3.8. Achieved *R* (%) values for real data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Blood	35,12 ± 1,63	34,97 ± 1,65	39,61 ± 3,38	46,42 ± 2,06	**49,53 ± 2,49**	46,24 ± 2,08	42,28 ± 1,91
BreastCancer	97,88 ± 0,18	97,81 ± 0,26	**98,06 ± 0,14**	97,71 ± 0,13	97,35 ± 0,18	97,73 ± 0,12	97,77 ± 0,15
Diabetes	**98,26 ± 0,65**	98,21 ± 0,59	86,43 ± 1,02	96,79 ± 0,80	96,79 ± 0,80	96,79 ± 0,80	58,26 ± 0,56
DiabeticRetinopathy	71,63 ± 1,05	71,45 ± 0,99	**78,66 ± 1,10**	71,11 ± 0,79	65,22 ± 1,00	70,91 ± 0,90	69,51 ± 0,95
Habermans	92,67 ± 0,76	92,52 ± 0,74	**93,57 ± 0,86**	92,37 ± 0,62	88,69 ± 0,90	92,15 ± 0,76	91,86 ± 0,87
Ionosphere	**98,61 ± 0,31**	97,62 ± 0,41	98,40 ± 0,35	98,18 ± 0,38	97,89 ± 0,42	98,14 ± 0,37	98,18 ± 0,28
MammographicMass	82,42 ± 0,66	81,96 ± 0,75	85,67 ± 0,69	**87,92 ± 0,56**	83,84 ± 0,76	**87,92 ± 0,56**	83,61 ± 0,53
PimaIndian	54,22 ± 1,41	54,10 ± 1,39	**59,25 ± 2,90**	53,09 ± 1,39	54,26 ± 1,38	53,36 ± 1,58	53,90 ± 1,43
StatlogHeart	58,73 ± 1,79	58,55 ± 1,50	**67,24 ± 3,25**	59,48 ± 2,65	54,40 ± 1,87	59,04 ± 2,53	58,58 ± 2,21
BalanceScale	95,48 ± 1,65	**95,79 ± 1,90**	93,59 ± 1,89	70,31 ± 4,56	59,29 ± 1,32	70,35 ± 4,64	69,57 ± 3,71
HayesRoth	**76,02 ± 2,20**	75,75 ± 2,25	70,67 ± 2,34	67,83 ± 2,33	69,23 ± 2,13	67,83 ± 2,33	67,00 ± 2,39
Iris	**97,19 ± 0,65**	97,01 ± 0,75	97,14 ± 0,67	96,92 ± 0,78	96,05 ± 0,59	96,90 ± 0,66	96,15 ± 0,64
HealthRisk	**85,29 ± 0,49**	85,11 ± 0,47	73,92 ± 0,65	73,59 ± 0,78	73,92 ± 0,76	73,60 ± 0,79	65,42 ± 0,84
TeachingAssistant	**60,03 ± 2,48**	59,60 ± 2,83	43,40 ± 2,55	38,83 ± 2,87	39,69 ± 2,90	39,76 ± 2,86	36,23 ± 2,78
WebsitePhishing	83,02 ± 1,04	83,51 ± 0,92	83,59 ± 0,85	**86,50 ± 0,84**	85,10 ± 0,91	**86,50 ± 0,84**	78,60 ± 1,10
Wine	72,24 ± 1,90	74,77 ± 2,30	71,72 ± 2,23	74,55 ± 1,70	**76,29 ± 1,63**	76,20 ± 1,60	73,22 ± 1,88
Lymphography	76,03 ± 3,46	75,96 ± 4,13	74,82 ± 3,77	**80,27 ± 2,38**	76,70 ± 2,82	**80,27 ± 2,38**	77,99 ± 3,27

Table 3.8. Achieved *R* (%) values for real data sets (Continued)

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Drug	66,80 ± 3,72	67,84 ± 3,46	67,22 ± 5,20	67,35 ± 2,97	**68,90 ± 3,16**	67,75 ± 3,38	68,39 ± 4,02
BMI	**90,56 ± 0,94**	89,62 ± 1,21	90,05 ± 1,00	89,32 ± 1,02	89,47 ± 1,24	89,66 ± 1,03	89,21 ± 0,81
Dermatology	88,85 ± 0,97	88,55 ± 1,00	88,15 ± 1,09	**89,57 ± 1,00**	89,29 ± 0,97	89,37 ± 0,98	89,24 ± 0,95
Glass	72,80 ± 2,71	74,47 ± 3,15	69,89 ± 2,53	74,54 ± 2,51	**75,17 ± 2,49**	74,54 ± 2,51	70,09 ± 2,36
Image	95,91 ± 0,20	**96,18 ± 0,22**	95,35 ± 0,19	95,76 ± 0,15	95,82 ± 0,15	95,85 ± 0,20	94,75 ± 0,14
Landsat	89,67 ± 0,21	89,65 ± 0,20	89,48 ± 0,18	89,86 ± 0,18	89,76 ± 0,14	**89,89 ± 0,17**	**89,89 ± 0,13**
Ecoli	84,10 ± 1,28	84,84 ± 1,66	84,81 ± 1,53	85,16 ± 1,46	83,28 ± 1,54	**85,36 ± 1,29**	84,57 ± 1,66
OptDigits	98,89 ± 0,07	98,89 ± 0,07	98,83 ± 0,05	98,94 ± 0,05	98,91 ± 0,05	**98,96 ± 0,06**	98,92 ± 0,05
ConnectionistBench	97,95 ± 0,31	**99,39 ± 0,26**	97,55 ± 0,41	99,28 ± 0,22	99,31 ± 0,21	99,29 ± 0,22	97,94 ± 0,36
Average *R*	**81,550 ± 1,258**	**81,697 ± 1,347**	80,655 ± 1,570	80,063 ± 1,353	79,006 ± 1,263	80,166 ± 1,371	76,966 ± 1,383

Table 3.9. Achieved *P* (%) values for real data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Blood	45,27 ± 1,70	44,91 ± 1,59	**51,95 ± 2,61**	36,15 ± 1,33	35,11 ± 1,10	36,06 ± 1,16	37,21 ± 1,40
BreastCancer	**98,20 ± 0,29**	98,13 ± 0,33	**98,20 ± 0,29**	97,96 ± 0,24	97,46 ± 0,32	98,05 ± 0,33	97,78 ± 0,32
Diabetes	98,10 ± 0,61	**98,14 ± 0,56**	82,03 ± 0,92	70,90 ± 0,26	70,90 ± 0,26	70,90 ± 0,26	60,72 ± 0,71
DiabeticRetinopathy	**62,51 ± 0,46**	62,38 ± 0,49	62,49 ± 0,46	62,46 ± 0,53	60,23 ± 0,71	62,49 ± 0,48	61,96 ± 0,52
Habermans	78,16 ± 0,64	78,12 ± 0,60	78,16 ± 0,64	78,16 ± 0,57	78,17 ± 0,67	78,18 ± 0,58	**78,61 ± 0,79**
Ionosphere	81,62 ± 0,55	83,48 ± 1,06	81,50 ± 0,44	84,04 ± 0,48	**84,17 ± 0,55**	84,04 ± 0,48	83,62 ± 0,48
MammographicMass	75,60 ± 0,59	75,61 ± 0,56	79,29 ± 0,61	75,77 ± 0,83	77,55 ± 0,68	76,00 ± 0,78	**79,57 ± 0,72**
PimaIndian	67,53 ± 1,17	**67,55 ± 1,21**	67,53 ± 1,17	64,99 ± 1,23	59,66 ± 1,08	64,54 ± 1,40	64,11 ± 1,30
StatlogHeart	64,68 ± 2,18	64,29 ± 2,36	64,68 ± 2,18	**65,05 ± 1,63**	60,17 ± 2,26	64,09 ± 1,82	64,67 ± 1,80
BalanceScale	**89,66 ± 1,43**	88,87 ± 1,74	88,79 ± 1,67	66,57 ± 3,92	59,68 ± 1,10	66,49 ± 3,87	66,67 ± 3,28
HayesRoth	**80,26 ± 1,86**	79,75 ± 2,06	75,12 ± 2,22	74,83 ± 2,89	75,16 ± 2,91	74,83 ± 2,89	70,27 ± 2,80
Iris	**97,14 ± 0,95**	96,99 ± 1,06	97,07 ± 0,96	96,90 ± 0,90	96,14 ± 0,60	96,85 ± 0,83	96,22 ± 0,64
HealthRisk	**86,38 ± 0,45**	86,24 ± 0,46	74,20 ± 0,66	74,44 ± 0,96	74,98 ± 0,87	74,44 ± 0,96	65,86 ± 0,93
TeachingAssistant	**60,53 ± 2,76**	60,13 ± 2,77	44,91 ± 3,66	38,58 ± 2,98	39,20 ± 2,45	39,28 ± 2,46	36,38 ± 2,76
WebsitePhishing	**84,65 ± 0,63**	84,64 ± 0,65	84,32 ± 0,97	80,13 ± 0,61	78,89 ± 0,72	80,13 ± 0,61	82,38 ± 1,59
Wine	72,05 ± 2,14	74,49 ± 2,39	71,38 ± 1,82	74,75 ± 1,93	**76,15 ± 2,01**	76,04 ± 1,98	72,88 ± 2,01
Lymphography	78,29 ± 2,64	78,75 ± 3,37	77,97 ± 3,20	79,53 ± 2,89	**80,78 ± 3,06**	79,80 ± 2,82	80,14 ± 3,25

Table 3.9. Achieved *P* (%) values for real data sets (Continued)

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Drug	67,70 ± 3,26	68,58 ± 3,41	64,50 ± 3,36	67,56 ± 2,61	**68,74 ± 3,01**	68,02 ± 3,23	67,87 ± 3,00
BMI	**90,26 ± 0,90**	89,78 ± 1,02	90,03 ± 0,88	89,59 ± 0,93	89,36 ± 1,11	89,81 ± 0,85	89,42 ± 1,04
Dermatology	87,94 ± 0,99	87,79 ± 1,00	87,42 ± 0,94	**89,04 ± 1,14**	88,88 ± 0,92	88,85 ± 1,05	88,38 ± 1,02
Glass	73,73 ± 3,07	74,61 ± 3,66	70,94 ± 2,96	74,73 ± 3,14	**75,63 ± 2,96**	74,73 ± 3,14	70,62 ± 2,74
Image	95,93 ± 0,19	**96,20 ± 0,20**	95,37 ± 0,19	95,79 ± 0,13	95,85 ± 0,14	95,88 ± 0,19	94,75 ± 0,14
Landsat	89,83 ± 0,22	89,81 ± 0,19	89,65 ± 0,17	**90,04 ± 0,16**	89,93 ± 0,15	90,03 ± 0,17	**90,04 ± 0,16**
Ecoli	84,84 ± 1,56	85,56 ± 1,28	85,67 ± 1,52	85,98 ± 1,24	84,11 ± 1,68	**86,10 ± 1,18**	84,96 ± 1,60
OptDigits	98,90 ± 0,07	98,91 ± 0,07	98,85 ± 0,05	98,96 ± 0,05	98,93 ± 0,06	**98,98 ± 0,06**	98,93 ± 0,05
ConnectionistBench	97,83 ± 0,32	**99,34 ± 0,32**	97,41 ± 0,40	99,24 ± 0,28	99,27 ± 0,27	99,25 ± 0,28	97,83 ± 0,36
Average *P*	**81,060 ± 1,216**	**81,271 ± 1,323**	79,208 ± 1,344	77,389 ± 1,302	76,733 ± 1,217	77,456 ± 1,302	76,225 ± 1,363

Table 3.10. Achieved *FScore* (%) values for real data sets

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Blood	38.69 ± 1.61 (3)	38.48 ± 1.58 (3)	**40.93 ± 2.94 (4)**	39.79 ± 1.38 (5)	40.15 ± 1.59 (6)	39.66 ± 1.27 (5)	38.74 ± 1.53 (3)
BreastCancer	**97.87 ± 0.18 (5)**	97.81 ± 0.18 (6)	**97.87 ± 0.18 (5)**	97.81 ± 0.13 (10)	97.37 ± 0.20 (15)	97.84 ± 0.17 (9)	97.73 ± 0.15 (13)
Diabetes	**98.15 ± 0.48 (3)**	**98.15 ± 0.43 (3)**	82.43 ± 0.82 (3)	81.76 ± 0.41 (3)	81.76 ± 0.41 (3)	81.76 ± 0.41 (3)	58.66 ± 0.44 (3)
DiabeticRetinopathy	66.42 ± 0.65 (15)	66.36 ± 0.58 (15)	**67.81 ± 0.75 (6)**	66.26 ± 0.55 (15)	62.39 ± 0.71 (15)	66.19 ± 0.58 (15)	65.27 ± 0.62 (15)
Habermans	**84.66 ± 0.56 (13)**	84.45 ± 0.52 (13)	84.61 ± 0.65 (12)	84.15 ± 0.55 (15)	82.68 ± 0.71 (15)	83.96 ± 0.62 (15)	84.07 ± 0.75 (14)
Ionosphere	89.04 ± 0.39 (3)	89.72 ± 0.70 (3)	88.80 ± 0.40 (3)	90.04 ± 0.30 (3)	**90.31 ± 0.38 (13)**	90.04 ± 0.30 (3)	89.61 ± 0.35 (3)
MammographicMass	78.38 ± 0.51 (15)	78.26 ± 0.55 (15)	80.38 ± 0.59 (7)	79.73 ± 0.66 (15)	79.53 ± 0.51 (15)	79.84 ± 0.64 (15)	**80.81 ± 0.61 (13)**
PimaIndian	59.14 ± 1.11 (13)	59.07 ± 1.14 (13)	**59.84 ± 1.42 (12)**	57.67 ± 1.18 (15)	56.13 ± 0.93 (15)	57.60 ± 1.24 (15)	57.76 ± 1.21 (14)
StatlogHeart	59.86 ± 2.19 (7)	59.68 ± 1.55 (5)	**62.12 ± 2.18 (6)**	60.33 ± 2.00 (15)	55.87 ± 1.34 (15)	60.14 ± 1.92 (15)	60.17 ± 1.64 (11)
BalanceScale	91.62 ± 1.53 (11)	**91.89 ± 1.81 (13)**	89.89 ± 1.76 (13)	68.12 ± 4.17 (15)	58.59 ± 1.21 (15)	68.14 ± 4.22 (15)	67.35 ± 3.48 (15)
HayesRoth	**75.38 ± 2.01 (3)**	74.94 ± 2.08 (4)	69.50 ± 2.33 (3)	65.42 ± 1.85 (3)	66.95 ± 2.41 (3)	65.42 ± 1.85 (3)	65.39 ± 2.38 (3)
Iris	**96.77 ± 0.86 (13)**	96.58 ± 0.99 (12)	96.73 ± 1.06 (15)	96.50 ± 0.90 (14)	95.61 ± 0.48 (13)	96.46 ± 0.79 (15)	95.70 ± 0.47 (4)
HealthRisk	**85.04 ± 0.48 (3)**	84.86 ± 0.46 (3)	73.62 ± 0.72 (3)	72.23 ± 0.88 (3)	72.59 ± 0.85 (3)	72.23 ± 0.89 (3)	65.22 ± 0.88 (3)
TeachingAssistant	**57.27 ± 2.31 (4)**	56.76 ± 2.82 (4)	39.78 ± 1.91 (5)	36.00 ± 2.23 (3)	36.57 ± 2.00 (3)	36.82 ± 1.95 (3)	33.40 ± 1.90 (6)
WebsitePhishing	**83.36 ± 0.74 (4)**	83.26 ± 0.87 (4)	83.05 ± 0.70 (4)	82.20 ± 0.59 (3)	80.90 ± 0.71 (3)	82.20 ± 0.59 (3)	77.24 ± 1.02 (7)
Wine	70.09 ± 1.80 (3)	72.65 ± 2.35 (3)	69.35 ± 1.49 (3)	72.60 ± 1.75 (3)	**74.34 ± 1.82 (3)**	74.22 ± 1.77 (3)	70.92 ± 1.67 (3)
Lymphography	75.07 ± 2.50 (6)	75.31 ± 2.75 (6)	74.04 ± 2.51 (7)	**78.03 ± 1.56 (3)**	76.67 ± 2.26 (8)	**78.03 ± 1.56 (3)**	77.38 ± 2.67 (8)

Table 3.10. Achieved *FScore* (%) values for real data sets (Continued)

Name	***HMAKNN$_R$***	***HMAKNN$_W$***	***KNN***	***WKNN***	***DWKNN$_1$***	***DWKNN$_2$***	***UWKNN***
Drug	64.03 ± 3.25 (3)	65.08 ± 3.08 (3)	60.52 ± 3.01 (4)	64.30 ± 2.03 (3)	**65.57 ± 2.25 (8)**	64.74 ± 2.81 (4)	64.53 ± 3.36 (10)
BMI	**89.44 ± 0.87 (3)**	88.63 ± 1.16 (4)	89.05 ± 0.89 (3)	88.40 ± 0.96 (6)	88.40 ± 1.17 (10)	88.73 ± 0.89 (6)	88.22 ± 0.85 (5)
Dermatology	87.10 ± 0.99 (3)	86.87 ± 1.00 (3)	86.43 ± 1.03 (3)	**88.09 ± 1.01 (4)**	87.88 ± 0.84 (13)	87.87 ± 0.97 (4)	87.60 ± 0.98 (5)
Glass	70.63 ± 2.41 (3)	71.90 ± 2.84 (3)	67.74 ± 2.52 (3)	72.15 ± 2.33 (3)	**72.81 ± 2.24 (9)**	72.15 ± 2.33 (3)	67.87 ± 2.18 (3)
Image	95.84 ± 0.18 (4)	**96.12 ± 0.21 (3)**	95.26 ± 0.19 (3)	95.69 ± 0.13 (3)	95.76 ± 0.14 (3)	95.78 ± 0.20 (5)	94.65 ± 0.13 (3)
Landsat	89.69 ± 0.21 (4)	89.67 ± 0.19 (4)	89.50 ± 0.17 (3)	**89.89 ± 0.15 (6)**	89.79 ± 0.14 (15)	89.88 ± 0.18 (5)	**89.89 ± 0.13 (5)**
Ecoli	82.55 ± 0.98 (15)	83.55 ± 1.29 (7)	83.58 ± 1.27 (7)	83.96 ± 1.01 (13)	81.97 ± 1.35 (15)	**84.14 ± 0.96 (13)**	82.94 ± 1.43 (10)
OptDigits	98.88 ± 0.07 (4)	98.89 ± 0.07 (4)	98.83 ± 0.05 (3)	98.93 ± 0.05 (6)	98.91 ± 0.05 (14)	**98.95 ± 0.06 (6)**	98.91 ± 0.05 (8)
ConnectionistBench	97.72 ± 0.32 (3)	**99.32 ± 0.30 (3)**	97.27 ± 0.43 (3)	99.21 ± 0.26 (3)	99.24 ± 0.25 (3)	99.22 ± 0.26 (3)	97.72 ± 0.39 (3)
Average *FScore*	**80.099 ± 1.123**	**80.317 ± 1.212**	78.039 ± 1.229	77.279 ± 1.117	76.497 ± 1.038	77.386 ± 1.132	75.299 ± 1.203

Table 3.11. Achieved ranking and statistics for real data sets

Name	Ranking							Statistics	
	$HMAKNN_R$	$HMAKNN_W$	*KNN*	*WKNN*	$DWKNN_1$	$DWKNN_2$	*UWKNN*	Chi-Square	*p*-value
Blood	2.40	2.60	1.00	5.42	7.00	5.58	4.00	173.89	6.72×10^{-35}
BreastCancer	2.72	3.62	2.88	3.88	6.80	3.42	4.68	80.55	2.75×10^{-15}
Diabetes	1.50	1.50	3.00	5.00	5.00	5.00	7.00	178.83	6.01×10^{-36}
DiabeticRetinopathy	3.18	3.07	2.98	3.25	7.00	3.15	5.37	95.00	2.77×10^{-18}
Habermans	2.68	3.30	2.53	3.87	6.62	5.13	3.87	85.51	2.59×10^{-16}
Ionosphere	5.77	4.02	6.87	2.62	1.53	2.62	4.58	143.23	2.08×10^{-28}
MammographicMass	6.25	6.65	1.77	4.60	3.55	3.88	1.30	162.84	1.48×10^{-32}
PimaIndian	2.08	2.17	2.08	4.55	7.00	4.90	5.22	149.13	1.18×10^{-29}
StatlogHeart	3.33	4.33	3.33	3.02	6.97	3.92	3.10	81.72	1.57×10^{-15}
BalanceScale	1.63	1.53	3.10	4.32	7.00	4.42	6.00	169.39	6.05×10^{-34}
HayesRoth	1.28	1.72	3.27	5.57	4.18	5.57	6.42	159.04	9.47×10^{-32}
Iris	2.78	3.37	2.98	3.52	5.98	3.58	5.78	84.11	5.04×10^{-16}
HealthRisk	1.13	1.87	4.43	5.08	3.47	5.02	7.00	160.12	5.58×10^{-32}
TeachingAssistant	1.42	1.58	3.07	5.23	4.95	4.95	6.80	159.92	6.15×10^{-32}
WebsitePhishing	2.80	3.80	1.03	4.88	7.00	4.88	3.60	137.72	3.04×10^{-27}
Wine	5.73	3.55	6.50	3.63	1.57	1.80	5.22	143.06	1.74×10^{-28}

Table 3.11. Achieved ranking and statistics for real data sets (Continued)

Name	Ranking							Statistics	
	$HMAKNN_R$	$HMAKNN_W$	*KNN*	*WKNN*	$DWKNN_1$	$DWKNN_2$	*UWKNN*	**Chi-Square**	***p*-value**
Lymphography	5.08	4.85	6.08	2.78	3.88	2.78	2.53	76.03	2.36×10^{-14}
Drug	3.35	2.83	6.40	4.40	3.25	3.50	4.27	59.11	6.83×10^{-11}
BMI	1.73	2.82	4.12	6.10	3.68	4.92	4.63	83.05	8.37×10^{-16}
Dermatology	4.52	5.27	6.93	2.17	3.02	3.17	2.93	109.09	3.16×10^{-21}
Glass	4.77	3.40	6.75	2.85	1.23	2.85	6.15	154.31	9.84×10^{-31}
Image	2.78	1.22	6.00	4.27	3.55	3.18	7.00	150.73	5.42×10^{-30}
Landsat	4.72	5.15	6.83	2.28	4.47	2.62	1.93	124.33	2.00×10^{-24}
Ecoli	5.98	3.55	3.20	2.45	6.50	2.12	4.20	111.37	1.05×10^{-21}
OptDigits	4.88	4.17	6.73	2.72	3.95	1.82	3.73	98.75	5.58×10^{-19}
ConnectionistBench	5.57	1.78	6.92	2.95	2.55	2.72	5.52	160.86	3.89×10^{-32}

Table 3.12. Average ranking of each *KNN*-based methods for all real data sets

Method Name	Average Ranking*
$HMAKNN_W$	3.219
$HMAKNN_R$	3.465
$DWKNN_2$	3.749
WKNN	3.901
KNN	4.262
$DWKNN_1$	4.681
UWKNN	4.724

*Sorted in ascending order

As there is no parameter optimization step in the traditional *KNN* classifier and the other weighted *KNN* classifiers introduced to this study, they are often quite fast, particularly for small to medium-sized data sets. Nevertheless, for the suggested $HMAKNN_R$ and $HMAKNN_W$ classifiers, as opposed to the competing *KNN* classifiers, parameters specified in Table 3.1 must be optimized. When the number of parameters to be optimized rises, the execution time of the model increases naturally owing to the time required for optimization. Even though the *BO* technique used to optimize these parameters is rapid and has a good convergence rate, *HMAKNN* classifiers might be viewed as more time-consuming than competing *KNN* classifiers. This study found that the maximum number of evaluations for the *BO* algorithm is 20 for $HMAKNN_R$ and 30 for $HMAKNN_W$. To determine the optimal parameters, each *HMAKNN*-based model should be run for the maximum number of evaluations, rather than just one. In terms of execution time, the suggested classifiers are inferior to the competitive *KNN* classifiers employed in this investigation.

Consequently, the accuracies achieved for the real data sets with statistically distinct distributions demonstrated that the *HMAKNN* classifiers can be successfully used to classification tasks in the real world, independent of the domain.

3.2.3. Comparison on synthetic and real data sets

Taking into consideration all of the results generated by using synthetic and real data sets, the suggested *HMAKNN* methods show comparable prediction accuracies to the competing *KNN* algorithms on synthetic data sets, while they produce better outcomes on real data sets, with several factors contributing to this enhancement. Listed below are the primary reasons why the suggested method performs more accurate on real data sets than on synthetic data sets:

- **Improved management of noisy or irrelevant data**: Real-world data sets frequently contain noisy or irrelevant data, which may negatively impact the performance of standard *KNN* model. Although every data of a particular class has similar properties, some exceptional data may be close to property values of different classes. The presented algorithm enhances the classification accuracy of real-world data sets by eliminating noise. By altering the particular characteristics of real-world data sets, the newly developed algorithm can significantly better process the relevant data.
- **Controlling imbalanced data sets**: Class imbalance is a common characteristic of real-world data sets, in which the number of instances in various classes is substantially unequal. Each class of data may have a distinct quantity of samples than the others, or significantly more than the others. The suggested technique uses strategies for overcoming class discrimination, including an improved weighted voting procedure, and *k*-value-based revising processes. The algorithm increases its ability to correctly classify samples belonging to underrepresented classes, that produce a general rise for accuracy.
- **Local adaptation or instance-based learning**: The suggested method contains strategies that alter the neighborhood number and thus, the group of closest neighbors utilized for classification on an instance-by-instance basis. Real-world data sets frequently have varying densities or complexity in different locations of the feature space. By dynamically adjusting the neighborhood size for each instance (query point), the algorithm is able to better detect local patterns and provide more accurate predictions on real data sets.

It has been demonstrated that the *HMAKNN* methods is able to predict with effective accuracies in the classification problem regardless of the area, even when the data points in the real data sets are distributed at different densities in various areas.

4. CONCLUSIONS

In this study, two improved versions of *KNN* named $HMAKNN_R$ and $HMAKNN_W$ were presented to increase the prediction performance of the traditional *KNN* classifier. The objective of the study was to demonstrate the proposed classifiers' effectivity by comparison with other *KNN* classifiers. In this context, firstly, eight synthetic and twenty-six real data sets were used to compare the prediction performance of the presented *HMAKNN* classifiers and the traditional *KNN* classifier. In addition, to prove the generalizable ability of accurate prediction of our models, the *HMAKNN* classifiers and the four weighted *KNN* classifiers *WKNN*, $DWKNN_1$, $DWKNN_2$ and *UWKNN* were compared in terms of prediction rates on the data sets.

The performance of all prediction models was assessed by calculating *ACC*, *R*, *P* and *FScore* (for real data sets) values through 10-fold cross-validation. Furthermore, Friedman's test, which is a statistical test, was used to determine the performance of the models in the most accurate way, and the average ranking values of all the models were found. To both reduce the negative effects of randomness and to ensure that results are more dependable, each of the *KNN*-based models was run independently 30 times and the average of the 30 results was taken as the evaluation value.

The following conclusions were reached regarding the results obtained from the synthetic data sets:

- Considering the average *ACC*, *R*, and *P* values of each model, the two best models were traditional *KNN* and $HMAKNN_W$, respectively.
- Following the best two models in terms of average *ACC* were *WKNN*, $DWKNN_2$, *UWKNN*, $HMAKNN_R$, and $DWKNN_1$, in that order. The average values of the classifiers are remarkably close to each other and the difference between the highest and lowest average *ACC* values is only 0.76 %.
- $HMAKNN_W$ has proven its robustness as the first model to follow *KNN* in each sorting, even though the ranking of the $HMAKNN_R$ model differs across metrics.
- The statistical rankings also produced comparable results to the performance metrics.

Traditional *KNN* topped the *ACC*, *R*, and *P* measures as well as the average ranking of all the models, whereas $HMAKNN_W$ was always second. $HMAKNN_W$ has proven its robustness as the

first model to follow *KNN* in each sorting, even though the ranking of the $HMAKNN_R$ model differs across metrics.

In the presented study, although the prediction performances of all the *KNN*-based models for synthetic data sets are statistically different from each other, it is noted that the differences are small, and the values are comparable. Since $HMAKNN_W$ we proposed is the most successful classifier after traditional *KNN* in terms of *ACC*, *R*, and *P* metrics and ranking results on synthetic data sets, we evaluated our models on real data sets and obtained the significant findings:

- In the average *ACC*, *R*, *P*, and *FScore* values of all classifiers, the best two models were $HMAKNN_W$ and $HMAKNN_R$, followed by *KNN*, $DWKNN_2$, *WKNN*, $DWKNN_1$, and *UWKNN*.
- In the average ranking values of each classifier, among the models we presented, $HMAKNN_W$ was first and $HMAKNN_R$ was second, followed by $DWKNN_2$, *WKNN*, *KNN*, $DWKNN_1$, and *UWKNN*.
- The $HMAKNN_R$ and $HMAKNN_W$ classifiers scored the greatest prediction rate in nine and four data sets, respectively, and demonstrated their superiority by outperforming the other five models in 13 of 26 data sets.
- In five of the 26 data sets, namely Blood, Diabetes, HayesRoth, HealthRisk, and TeachingAssistant, the recommended classifiers performed considerably better than the other five *KNN* predictors. For the aforementioned five data sets, both *HMAKNN* classifiers had an average *ACC* advantage of 4.77%, 15.49%, 9.51%, 12.96%, and 19.92% over the other five competing classifiers, *KNN*, *WKNN*, $DWKNN_1$, $DWKNN_2$, and *UWKNN*, respectively.

On the other side, when the classification results on real data sets were examined, both proposed *HMAKNN* models provided statistically more accurate values than all other competing methods compared in terms of average *ACC*, *R*, *P*, *FScore* and ranking.

In conclusion, both *HMAKNN* models presented in this study have proven their accuracy and reliability by achieving correct prediction rates. Therefore, with this remarkable efficient performance on real-world data sets, we present the *HMAKNN* algorithms as a safe and robust classifier for classification problems.

REFERENCES

Adeniyi, D. A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. *Applied Computing and Informatics*, *12*(1), 90–108. https://doi.org/10.1016/j.aci.2014.10.001

Angel Viji, K. S., & Rajesh, D. H. (2020). An Efficient Technique to Segment the Tumor and Abnormality Detection in the Brain MRI Images Using KNN Classifier. *Materials Today: Proceedings*, *24*, 1944–1954.

Arslan, H., & Arslan, H. (2021). A new COVID-19 detection method from human genome sequences using CpG island features and KNN classifier. *Engineering Science and Technology, an International Journal*, *24*(4), 839–847. https://doi.org/10.1016/j.jestch.2020.12.026

Bulut, F., & Amasyali, M. F. (2017). Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster. *Pattern Analysis and Applications*, *20*(2), 415–425. https://doi.org/10.1007/s10044-015-0504-0

Chawla, S., Kumar, R., Aggarwal, E., & Swain, S. (2018). *Breast Cancer Detection Using K-Nearest Neighbour Algorithm*.

Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Deepa, G., Mary, G. L. R., Karthikeyan, A., Rajalakshmi, P., Hemavathi, K., & Dharanisri, M. (2022). Detection of brain tumor using modified particle swarm optimization (MPSO) segmentation via haralick features extraction and subsequent classification by KNN algorithm. *Materials Today: Proceedings*, *56*, 1820–1826. https://doi.org/10.1016/j.matpr.2021.10.475

Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(4), 325–327. https://doi.org/10.1109/TSMC.1976.5408784

Elsayad, A. M., Nassef, A. M., & Al-Dhaifallah, M. (2022). Bayesian optimization of multiclass SVM for efficient diagnosis of erythemato-squamous diseases. *Biomedical Signal Processing and Control*, *71*. https://doi.org/10.1016/j.bspc.2021.103223

Ertuğrul, Ö. F., & Tağluk, M. E. (2017). A novel version of k nearest neighbor: Dependent nearest neighbor. *Applied Soft Computing Journal*, *55*, 480–490. https://doi.org/10.1016/j.asoc.2017.02.020

Fix, E., & Hodges, J. L. (1951). An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation. *International Statistical Review*, *57*(3), 233–247.

Gao, Z., Lin, Y., Sun, X., & Zeng, X. (2022). A reduced order method for nonlinear parameterized partial differential equations using dynamic mode decomposition coupled with k-nearest-neighbors regression. *Journal of Computational Physics*, *452*. https://doi.org/10.1016/j.jcp.2021.110907

Gou, J., Du, L., Zhang, Y., & Xiong, T. (2012). A New Distance-weighted k-nearest Neighbor Classifier. *Article in Journal of Information and Computational Science*, *17*(6), 1429–1436.

Gou, J., Xiong, T., & Kuang, Y. (2011). A novel weighted voting for k-nearest neighbor rule. *Journal of Computers*, *6*(5), 833–840. https://doi.org/10.4304/jcp.6.5.833-840

Ho, W. T., & Yu, F. W. (2021). Chiller system optimization using k nearest neighbour regression. *Journal of Cleaner Production*, *303*. https://doi.org/10.1016/j.jclepro.2021.127050

Hossain, E., Hossain, M. F., & Rahaman, M. A. (2019). A Color and Texture Based Approach for the Detection and Classification of Plant Leaf Disease Using KNN Classifier. *2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ECACE.2019.8679247

Hossny, K., Magdi, S., Soliman, A. Y., & Hossny, A. H. (2020). Detecting explosives by PGNAA using KNN Regressors and decision tree classifier: A proof of concept. *Progress in Nuclear Energy*, *124*. https://doi.org/10.1016/j.pnucene.2020.103332

Jiang, S., Pang, G., Wu, M., & Kuang, L. (2012). An improved K-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, *39*(1), 1503–1509. https://doi.org/10.1016/j.eswa.2011.08.040

Kang, P., & Cho, S. (2008). Locally linear reconstruction for instance-based learning. *Pattern Recognition*, *41*(11), 3507–3518. https://doi.org/10.1016/j.patcog.2008.04.009

Karabulut, B., Arslan, G., & Ünver, H. M. (2019). A Weighted Similarity Measure for k-Nearest Neighbors Algorithm. *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, *15*(4), 393–400. https://doi.org/10.18466/cbayarfbe.618964

Kim, D., Kwon, K., Pham, K., Oh, J. Y., & Choi, H. (2022). Surface settlement prediction for urban tunneling using machine learning algorithms with Bayesian optimization. *Automation in Construction*, *140*. https://doi.org/10.1016/j.autcon.2022.104331

Lahmiri, S., Bekiros, S., & Avdoulas, C. (2023). A comparative assessment of machine learning methods for predicting housing prices using Bayesian optimization. *Decision Analytics Journal, 6*. https://doi.org/10.1016/j.dajour.2023.100166

Lee, S., Bae, J. H., Hong, J., Yang, D., Panagos, P., Borrelli, P., … Lim, K. J. (2022). Estimation of rainfall erosivity factor in Italy and Switzerland using Bayesian optimization based machine learning models. *Catena, 211*. https://doi.org/10.1016/j.catena.2021.105957

Mateos-García, D., García-Gutiérrez, J., & Riquelme-Santos, J. C. (2019). On the evolutionary weighting of neighbours and features in the k-nearest neighbour rule. *Neurocomputing, 326–327*, 54–60. https://doi.org/10.1016/j.neucom.2016.08.159

Mitani, Y., & Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recognition Letters, 27*(10), 1151–1159. https://doi.org/10.1016/j.patrec.2005.12.016

Müller, P., Salminen, K., Nieminen, V., Kontunen, A., Karjalainen, M., Isokoski, P., … Surakka, V. (2019). Scent classification by K nearest neighbors using ion-mobility spectrometry measurements. *Expert Systems with Applications, 115*, 593–606. https://doi.org/10.1016/j.eswa.2018.08.042

Narayan, Y. (2020). SEMG signal classification using KNN classifier with FD and TFD features. *Materials Today: Proceedings, 37*(Part 2), 3219–3225. https://doi.org/10.1016/j.matpr.2020.09.089

Pan, Z., Wang, Y., & Ku, W. (2017). A new k-harmonic nearest neighbor classifier based on the multi-local means. *Expert Systems with Applications, 67*, 115–125. https://doi.org/10.1016/j.eswa.2016.09.031

Singh, H., Sharma, V., & Singh, D. (2022). Comparative analysis of proficiencies of various textures and geometric features in breast mass classification using k-nearest neighbor. *Visual Computing for Industry, Biomedicine, and Art, 5*(1). https://doi.org/10.1186/s42492-021-00100-1

Sultana, N., Hossain, S. M. Z., Abusaad, M., Alanbar, N., Senan, Y., & Razzak, S. A. (2022). Prediction of biodiesel production from microalgal oil using Bayesian optimization algorithm-based machine learning approaches. *Fuel, 309*. https://doi.org/10.1016/j.fuel.2021.122184

Wang, Q., Wang, S., Wei, B., Chen, W., & Zhang, Y. (2021). Weighted K-NN Classification Method of Bearings Fault Diagnosis with Multi-Dimensional Sensitive Features. *IEEE Access, 9*, 45428–45440. https://doi.org/10.1109/ACCESS.2021.3066489

Yigit, H. (2013). A weighting approach for KNN classifier. *2013 International Conference on Electronics, Computer and Computation, ICECCO 2013*, 228–231. IEEE Computer Society. https://doi.org/10.1109/ICECCO.2013.6718270

Zhang, H., Berg, A. C., Maire, M., & Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2*, 2129–2136. IEEE Computer Society. https://doi.org/10.1109/CVPR.2006.301

Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for kNN Classification. *ACM Transactions on Intelligent Systems and Technology*, *8*(3). https://doi.org/10.1145/2990508

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

Zhong, X. F., Guo, S. Z., Gao, L., Shan, H., & Zheng, J. H. (2017). An improved k-NN classification with Dynamic K. *ACM International Conference Proceeding Series*, *Part F128357*, 211–216. Association for Computing Machinery. https://doi.org/10.1145/3055635.3056604