Amer *et al. Journal of Big Data* (2025) 12:86
https://doi.org/10.1186/s40537-025-01137-2

Journal of Big Data

RESEARCH Open Access

Effective k-nearest neighbor models for data classification enhancement

Ali A. Amer[1], Sri Devi Ravana[1*] and Riyaz Ahamed Ariyaluran Habeeb[1]

*Correspondence:
sdevi@um.edu.my

[1] Department of Information Systems, Faculty of Computer Science & Information Technology, Universiti Malaya, Kuala Lumpur, Malaysia

Abstract

Imbalanced class distributions and class overlaps are major problems that are often associated with the classifier's decreased performance. To lessen their impact, the literature continues to introduce methods, including kNN extensions. Nevertheless, as far as we know, the kNN's performance has not yet attained satisfactory levels. Therefore, our study presents kNN models aimed at considerably alleviating these problems' negative effects. Compared to previous research that frequently considered one problem, each of our models considers more than one problem. **Firstly**, we propose a straightforward yet effective technique to simply identify irrelevant "noisy" points, the proximal ratio (PR), using which these overlapped points have the minimum possible contribution regarding the classification decision. PR explicitly identifies points with low PR scores as outliers and gives a direct indication of how consistent a point is with its neighbors. It flags points as outliers when they are surrounded by neighbors from multiple classes. **Further**, PR evaluates the local consistency of a point with its neighbors, considering the distribution of classes within the local neighborhood. **Secondly**, a weighting computation is introduced to reduce the imbalanced class impact. **Finally**, PR and weighting equations are integrated with the kNN algorithm to develop PRkNN models, whose performance outperforms their rival kNNs and competes fiercely with popular machine learning models. Extensive evaluation analysis is made across fifty-two datasets in six experimental phases using six evaluation metrics. The results, supported by statistical tests, demonstrate that, on average, our proposed models are extremely competitive without being appreciably slower than their competitors.

Keywords: k-nearest neighbor, kNN, Data classification, Data mining, Machine learning, Information retrieval

Introduction

One of the most popular approaches for pattern clustering, classification, and regression is the nearest neighbor algorithm [1, 2]. For data classification, to simply label the target test data, the k-nearest neighbor (kNN) must traverse the entire training set. Because of this, classical kNN is considered to be lazy and, as a result, performs poorly with high computational complexity, especially when dealing with imbalanced and high-dimensional data. It struggles with class imbalance, as the majority class tends to dominate, leading to biased predictions towards the more

frequent class. Further, KNN is sensitive to outliers, as noisy or distant data points can disproportionately affect distance calculations and predictions.

However, our extensive investigation of the kNN literature has shown that there is still considerable room for improvement in the computational efficiency and classification performance of the kNN model. It has been demonstrated that additional data complexity, such as class overlap, noisy, and k value selection, greatly affects classification performance in addition to class imbalance. Unfortunately, this complexity coexists with the class imbalance problem. It is worth indicating that throughout this work, irrelevant points will be used interchangeably with noisy points, sometimes known in literature as "outliers."

Therefore, the literature keeps enhancing kNN to address this issue by introducing kNN extensions that have been able to effectively compete with other sophisticated ML models, over numerical datasets in particular, in many applications related to several fields. Among these fields are: text mining [3], text clustering and classification [4, 5], business and financial modelling using regression [6], medicine [7–9], disease prediction [10], data classification [11], pattern recognition [12–14], industrial applications [15], recommendation-related applications [2, 16], etc. Nevertheless, the performance of kNN has long been noted to be negatively affected by several problems (see Fig. 1). Among these problems are the imbalanced class distributions in the image and data [17–19], class overlaps [20, 21], high-dimensional [22], outliers [23], data similarity [8], certainty degree [9], high-computation costs [12], power plan applications [9], and prediction [10, 24]. Furthermore, the "k" value and the similarity metric employed have a significant impact on the kNN performance. The "k" selection is a difficult problem when data are formally distributed and used for retrieval purposes [25]. Overall, the larger k is preferable to the smaller one because it will smooth out the class boundaries and resist noise. As a result, it is hard to determine the optimal k in many applications. Thus, several academics attempted to resolve this conundrum by repeatedly applying kNN to identify the ideal k value. They also used the averaged performance to track the overall effectiveness of their classifiers.

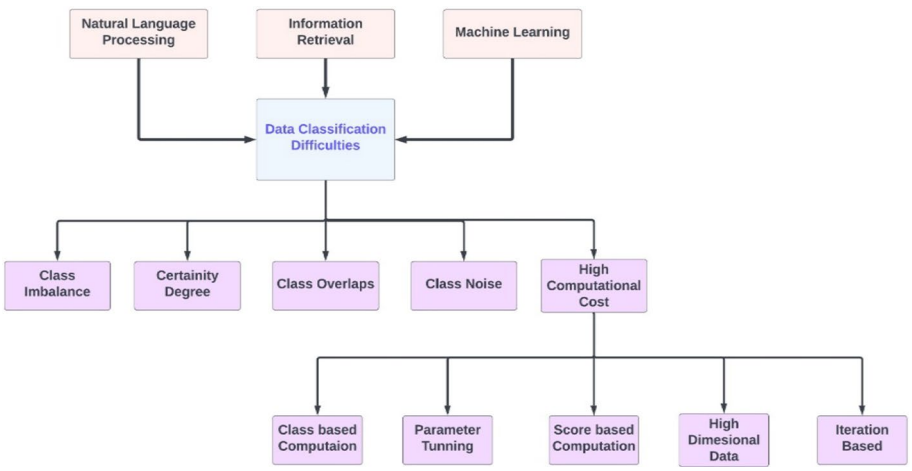

Fig. 1 Data classification difficulties

On the other hand, while the kNN is fairly fast and has incredible performance when used over balanced datasets [5], its performance is frequently degraded when the datasets are imbalanced. For these reasons, the literature has been developing kNN extensions to improve kNN efficacy and efficiency. Some researchers attempted to considerably tackle some of the kNN difficulties and thus improve the performance of kNN by incorporating prior knowledge (i.e., the data distribution). ExNRule KNN [13], CKNN [19], the discriminant kNN [26], fuzzy kNN [27], weight-adjusted kNN (WEkNN) [28], degree of certainty-based kNN [20], distance-based kNN [29], local DMCKNN [30], modified kNN [31], distance-weighted kNN [32], Ensemble kNN [33], and LMKNN [37] are a few examples of these studies. Despite this progress, these kNN models still suffer from computational efficiency [13, 19, 26, 42], noise sensitivity [13, 31, 33], k-value selection [19, 27, 28, 30], and class imbalance negative impact [31, 37]. A detailed discussion over the problems from which each kNN model still suffers has been drawn in the next section.

As discussed in the next section and seen by the results section's findings, the data classification enhancement using the kNN algorithm has not yet achieved the intended level. Therefore, our research's ultimate aim is to introduce unique, straightforward, and effective kNN models that have the power to lessen the negative impact of some of these problems from which the state-of-the-art (SOTA) models still suffer or do not satisfy the intended performance. Concisely, the following is a list of our major contributions:

1. Developing the proximity ratio (PR) technique to simply identify overlapping "noisy" points. As an explicit outlier detection, PR clearly identifies points with low PR scores as outliers and gives a direct indication of how consistent a point is with its neighbors.
2. Proposing new weighting schemes to alleviate the detrimental impact of the class imbalance on the classifier's performance. Further, PR evaluates the local consistency of a point with its neighbors, considering the distribution of classes within the local neighborhood.
3. Combining the PR technique with weighting schemes to introduce three kNN versions (namely, PRkNN, EPRkNN, and WPRkNN) of straightforward yet highly competitive designs to significantly improve the data classification.
4. Conducting an extensive experimental study, supported by statistical tests, using fifty-two real-world to demonstrate the considerable competitiveness of the proposed kNN models. According to the results secured, our WPRkNN model outperform its kNN rivals in terms of stability and sensitivity to the number of nearest neighbors.
5. Providing a more in-depth analysis using imbalanced, noisy, and time series datasets to illustrate how the models perform under different circumstances. Overall, it is seen that the proposed PRkNN models' classification accuracy gets better as k grows, adding a new feature by becoming less sensitive to the selection of k value.

The rest of the paper is structured as follows: In Sect. "Related work", the relevant work on data classification using kNN extensions is critically analyzed. Sect. "Methodology" presents the proposed approach, which includes the proposed kNN designs. Sect. "Experimental setup" describes the experimental setup, including

the dataset description, all models compared, and the evaluation metrics. Sect. "Results" draws the experimental results in four phases, supported by the significance tests. A concise discussion of the behavior of the model is provided in Sect. "Discussion". Sect. "Conclusions and future work" concludes this paper and outlines the next possibilities for research.

Related work

The literature keeps introducing kNN variants to relieve the effects of the class imbalance and overlapping problems by which the data classification is severely affected. For example, to address the class imbalance, some variations attempted to assign various weights to the k closest neighboring points according to the distances of the points from the test point P_{test}. The weighting scheme would assign the neighbors with closer distances heavier weights, alleviating the class imbalance and outlier consequences [34]. Some examples of these variations are the weighted local mean representation-based KNN [35], CRC-based kNN [36], local mean-based k closest neighbor (LMkNN) [37]. and the weighted discriminative collaborative competitive representation [38].

On the flip side, based on the collaborative representation-based classification (CRC) concept, a CRC-based kNN was presented in [36] based on the weighting used in [38] to increase pattern discrimination. However, the model's ability was limited to classifying images only. In [11], the two-layer nearest neighbor (TLNN) was developed based on a neighbor selection technique used to build the two-layer neighborhood data. Simultaneous consideration was given to the neighborhoods of the query and all selected training instances inside this neighborhood. The distance, distribution relationship, and backward nearest neighbor relationship between the query and each of the chosen training instances in the previously stated neighborhoods were then calculated to get the two-layer closest neighbors of the query. However, the model must first go through a parameter tuning process to perform competitively, restricting the model's generalization. Furthermore, as reported in the author's results, the proposed kTLNN classifier had a larger error rate than the kNN classifier, primarily for small values of k (e.g., 1, 3, and 5). Not to mention that all of the rivals used in the comparison study were not well established, with the exception of one kNN variation.

In order to solve the problem of k value selection, the authors [35] presented two locality-constrained representation-based kNNs by utilizing the representation coefficients and the k-local mean vectors of k-nearest neighbors per class. The weighted local mean representation-based k-nearest neighbor rule (WLMRKNN) and the weighted representation-based k-nearest neighbor rule (WRKNN) were the two methods adopted. The study conducted an experimental comparison between the models suggested and seven competitors, utilizing only the least classification error metric. The results indicated that one of the proposed kNNs outperformed the other somewhat on certain datasets. However, the performance was not very competitive considering that the designs of these variants are complex. The weighted extended nearest neighbor (WENN) method was recently developed [28] to boost the speed of ENN [39] which, came to deal with noise and class imbalance. In order to increase the efficiency of ENN, WENN adopted weighted voting to determine the class label of test data. This means that each neighbor associated with its inverse distance assigns a

value between 0 and 1 to a class label consensus level, whereas ENN used hard voting, which means that each neighbor can only assign 0 or 1 to a class label consensus level. In comparison to even standard SkNN, WENN's test phase takes a very long time due to the weighting process, even if it lacks a training step. In this regard, both our WPRkNN and WENN employ a soft weighting scheme; however, our WPRkNN is theoretically superior to both ENN and WENN in the following ways: (1) **Explicit Outlier Detection:** PR clearly identifies points with low PR scores as outliers and gives a direct indication of how consistent a point is with its neighbors. ENN and WENN, in contrast, are primarily focused on classification and implicitly handle noise through point removal (ENN) or weighting (WENN), (2) **Handling Overlapping Classes:** PR is designed to handle cases where classes overlap in the feature space. It flags points as outliers when they are surrounded by neighbors from multiple classes, making it better suited for datasets with class overlap or complex structures. ENN and WENN are less equipped to detect outliers in such cases because they don't evaluate neighborhood consistency in the same pattern, and (3) **Local Consistency Evaluation:** PR evaluates the local consistency of a point with its neighbors, considering the distribution of classes within the local neighborhood. In contrast, ENN removes misclassified points based solely on KNN, and WENN modifies weights without directly evaluating local class distributions. Most importantly, compared to WPRkNN, which has no parameter to tune, WENN suffers in finding the optimal performance over each dataset due to its fine-tuning process needed to tune the alpha parameter; otherwise, its optimal performance is not guaranteed, and thus no generalization is secured.

Meanwhile, the conditional kNN (CkNN) has been presented in [19] as a solution to the overlapping-class dilemma. Our experimental analysis demonstrates that CkNN performance has typically been either equivalent to or occasionally worse than SkNN, with the exception of the scenario where k is of very low value (i.e., 1, 3, 5). Our results are in line with the assertions made by the authors themselves that the CkNN performs well for these low values of k. Additionally, they implied that both CkNN and SkNN perform similarly for bigger k values (i.e., $k > 5$). Another kNN variation used class local techniques to improve the performance of the kNN. One of the best-known examples of this kind is the local mean-based k closest neighbor (LMkNN) [37]. The local mean vector of these neighboring points is calculated for the test point Ptest by LMkNN after it finds its k nearest neighbors in each class. The distance between each local mean and the Ptest is found, and the class with the shortest distance is given the Ptest. Our experimental findings, however, show that LMkNN is only more resistant to outliers (widely known as irrelevant points) than SkNN when the training set is small. The evidence gained in [19] and our empirical observations are both consistent regarding this fact.

The query sample in LMKNN is allocated to the class of the category that the nearest mean vector falls into. Sometimes, due to the short number of training samples, outliers "irrelevant points" can more readily impact KNN's classification performance, producing inaccurate outcomes. The Local Means *K*-Nearest Centroid Neighbor Classification (LMKNCN) was developed in [38] as an effective substitute for LMkNN in solving this issue to classify the test data in order to take advantage of each centroid neighbor's contribution to the classification. On LMKNCN, however, the default contribution of

each sample point to the classification result is the same, failing to adequately account for the fact that distinct sample points ought to contribute differently to the classification of the samples that are to be evaluated. To address this deficiency, the Attention-Based Local Mean N-Nearest Centroid Neighbor Classifier (ALMKNCN) was proposed [4]. Nevertheless, the ALMKNCN model had a complex design, making it highly prone for overfiting. Further, it did not work well with overlapping classes, and its performance is constrained due to the fine tuning process of its parameters.

Meanwhile, Parvin et al. (2010) in [31] aimed to introduce a modified kNN version (MkNN) of kNN. By weighting the query's neighbors, MKNN approximates the query label, making it a type of weighted KNN. The process increases the overall number of neighbors by adding the frequencies of the same labeled neighbors. But when we experimented with the MkNN, we discovered that its performance is worse with higher k values. The modified KNN approach can also be sensitive to noise and outliers, as the changes introduced may amplify the impact of these data points on classification results. Finally, like standard k-NN, it may face challenges with class imbalance, where the classifier tends to be biased toward the majority class. In order to address the majority vote difficulties, a different weighted kNN model (WLMkNN) was presented in [32]. Only the accuracy obtained from the original k-NN approach was used to compare the accuracy of the findings, though. Although the accuracy has been enhanced, the model is susceptible to the outliers' impact on the classification tasks due to the local mean computations. Our results indicate that, in comparison to our proposed PRkNN models, the WLMkNN model performs well on a small number of datasets but poorly on the remainder.

Further, the literature frequently makes use of several clustering-based kNNs to speed up kNN performance. The training data are initially grouped in these kNNs [39, 40], and only a tiny subset of the data is used as the representative of each group or cluster as the new training set to determine the class label of a test point in the test stage. Consequently, substantial informative points would be missed in the test phase. On the same page, a density-based adaptive k closest neighbor technique called DBANN was presented by [21] to address imbalanced and overlapping problems at the same time. Besides being a clustering-based kNN, only one k value—3—was used to assess the model's performance, raising concerns about its efficacy. Additionally, the model's complexity and the process of fine-tuning its parameters point to the model's inefficiency. More recently, in [33], the centroid displacement-based *k*-NN algorithm (CDkNN) was proposed to tackle the class imbalance problem. Instead of Euclidean, the model used centroid displacement for class determination. Even though CDkNN had a short runtime compared to its competitors, its precision was not highly competitive. Comparing to CDkNN, our experimental results assert the superiority of our PRkNNs in the vast majority.

Recently, the idea of minimizing the distances of multiple kNNs in each class was used to provide a multi-voter multi-commission nearest neighbor (MVMCNN) as a KNN extension for LMkNN [41]. To determine the overall distance between the provided Ptest and the k pseudo-closest neighbors, MVMCNN used what were called commissions. Results indicated that MVMCNN could somewhat enhance both the original LMkNN and SkNN. Our experimental results showed that across small datasets,

MVMCNN enhanced SkNN performance in the vast majority of situations. However, in the majority of target datasets, it often performed worse than multiple SOTA kNNs, including the ones we presented, due to its clustering-based mechanism. As a matter of fact, some kNNs, such as MVMCNN, which depends on clustering, have short testing times because of the large reduction in training set size; nevertheless, the speed comes at the expense of their accuracy, which is sometimes worse than that of the regular kNN. Our experimental study reveals that even though clustering-based kNNs are efficient, the SkNN often outperforms them regarding classification accuracy.

Finally, the extended neighborhood rule (ExNRule) and the representation coefficient-based k-nearest centroid neighbor (RCKNCN) were proposed in [13, 42] as the last and most recent kNN extensions to improve data classification. The outcomes of both studies demonstrated how successful these kNN extensions were in outperforming their competitors. The RCKNCN can be computationally expensive due to the need to calculate the matrix inversion for each test point, which can become inefficient for large datasets or high-dimensional feature spaces. The regularization term (lmbda) is sensitive to its choice, as improper tuning can lead to underfitting or overfitting. On the other hand, ExNRule KNN can be computationally expensive due to bootstrapping and multiple distance calculations for each test point, especially with large datasets. It is also sensitive to noisy data and outliers, which can distort the bootstrapping process and lead to inaccurate predictions. Additionally, the method's performance may degrade with high-dimensional data due to the curse of dimensionality. Fortunately, our experimental results also show our models' superiority over these models, indicating that the kNN is still an actively appealing model whose performance is not yet at an adequate level.

Methodology

Proposed proximal ratio-based technique

Our work introduces the proximal ratio (PR) technique as a ratio to reflect the degree of differentiation and high or low correlation between the classes in the target dataset. Meanwhile, the irrelevant points (or what can be called outliers or noisy points) are those highly overlapped points in the drawn hyper-sphere region (see Fig. 2). The PR would assign these noisy points the maximum distance from each test point. As a result, these points are given the minimum contributions during the classification process, preventing them from making any substantial contributions that would negatively impact the data classification. As such, we can say that the PR technique functions similarly to the outlier detection methods since the outlier identification problem centers on figuring out the parameter value—which in our case is the minimum PR value as the outlier threshold value—to indicate whether or not the value is an outlier. Not to mention that the PRkNN reduces the potential impact of these overlapped points (outliers) by only using the k neighbor points in the radius drawn to calculate the PR scores. Most importantly, the PR scores of each neighbor are used as its weight, and distances and labels of neighbors are also used to compute double neighbors' weights, as will be explained below. Specifically speaking, based on the class affiliation of its nearest neighbors, the Proximal Ratio Score (PR) provides a straightforward method to calculate the target point's weight. Consequently, these scores are used to generate the predictions after being computed using the available data for each training data point. In actuality,

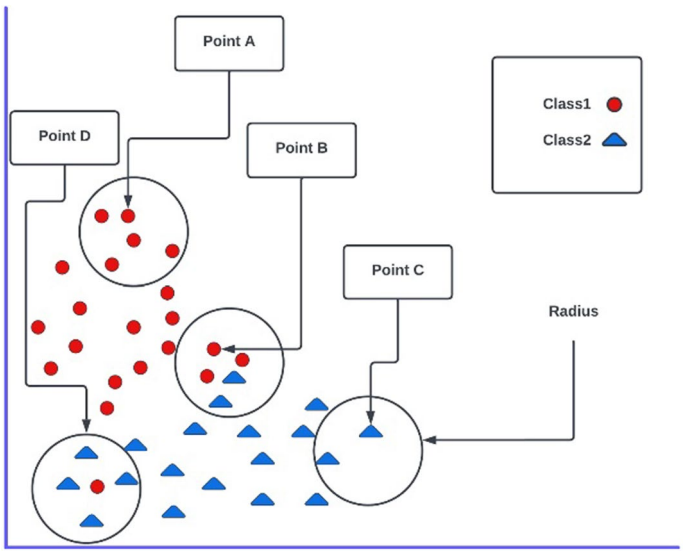

Fig. 2 Proximal ratio computation illustration

the PR score computation makes use of the information found in a dataset's structure. These score values are then used to distinguish the pertinent neighbors that should be taken into account throughout the prediction phase.

Therefore, it can be concluded that the PR technique is precisely designed with the ultimate aim of simultaneously relieving the detrimental impact of class imbalance and overlapping. The PR is then combined with further weights to make the weighted PR, which is finally integrated with the proposed kNNs to successfully construct effective PRkNN models (see Figs. 3, 4, 5). We have conducted an extensive experimental study to validate the effectiveness of PRkNNs versus the twelve state-of-the-art leading kNN models. Further, besides being robust to the class imbalance negative impact, the experimental investigation reveals that the PRkNN models are also effective over noisy and time series datasets as well as less sensitive to the k value selection as their performance keeps stable with growing k values (see Figs. 7 and 8).

Proximal ratio computation (PR)

In its simple definition, PR is a technique to establish a measure (ratio) for capturing the proximity of the target point with regard to its neighboring points by taking each point "i" and using a drawn hyper-sphere region to determine exactly how many other points of its class lie inside that area (e.g., radius). In the PR computation, a class-wise radius is computed (see Eq. 1) using the mean pairwise distance of all the points in the class.

$$Radius_c = \frac{\sum_{i=1}^{N_c} \sum_{j=1}^{N_c} dist_{i,j}}{N_c} \tag{1}$$

Here, N_c is the number of training data points in class c, $dist_{i,j}$ is the distance between class data point i and data point j. Now, for every training point, every other training data point has a distance less than or equal to the values of $Radius_c$ is inside its radius, and the Proximal Ratio is computed as follows (see Eq. 2).

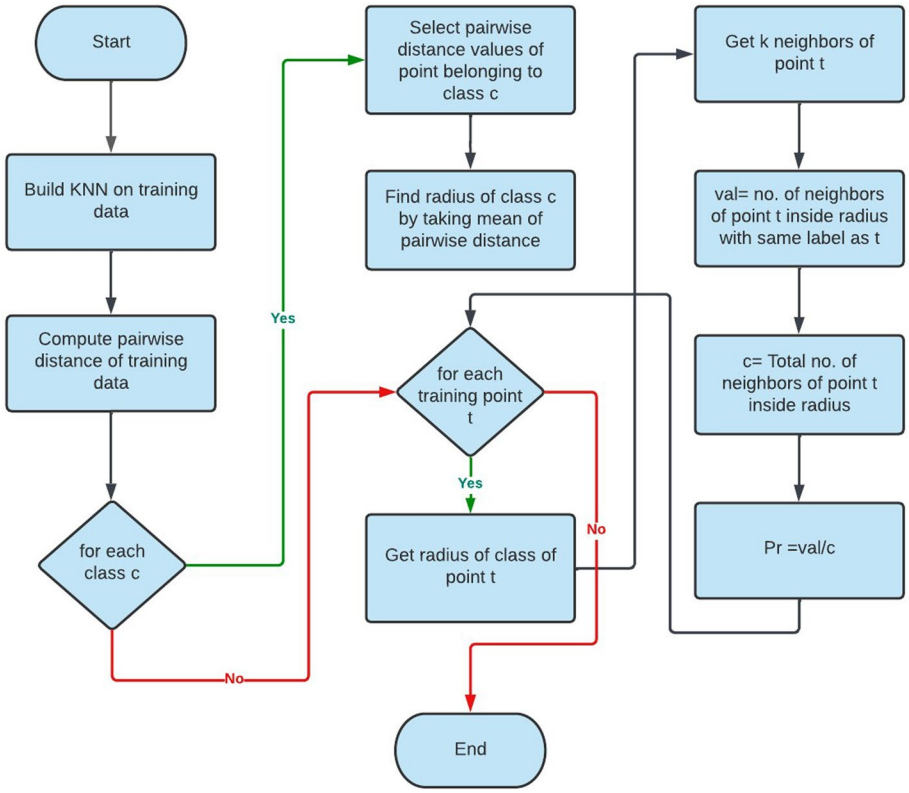

Fig. 3 Proximal ratio computation algorithm

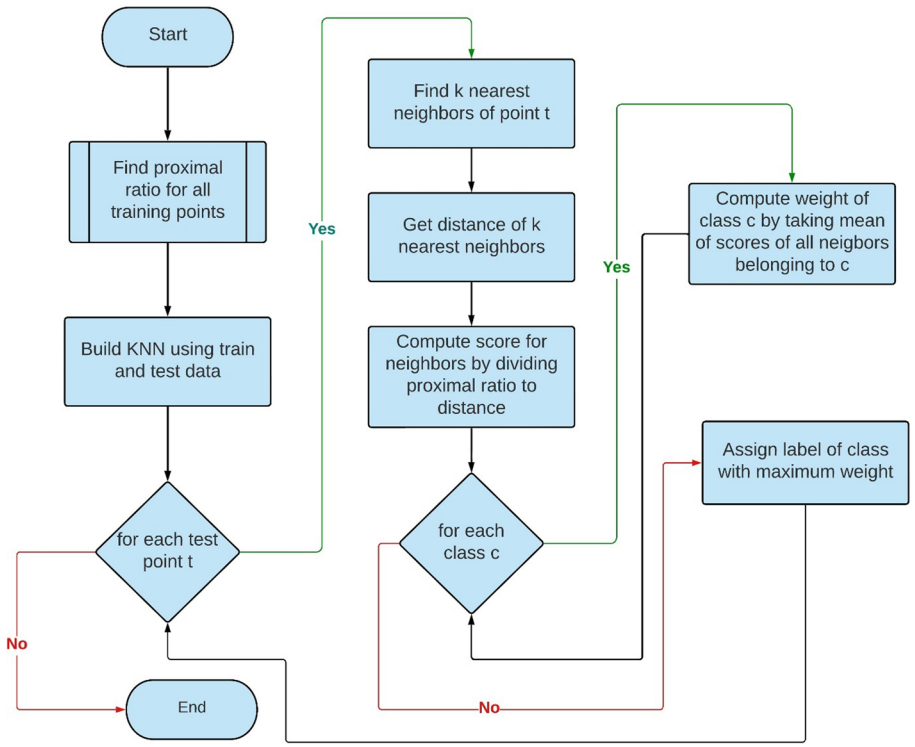

Fig. 4 PRkNN algorithm

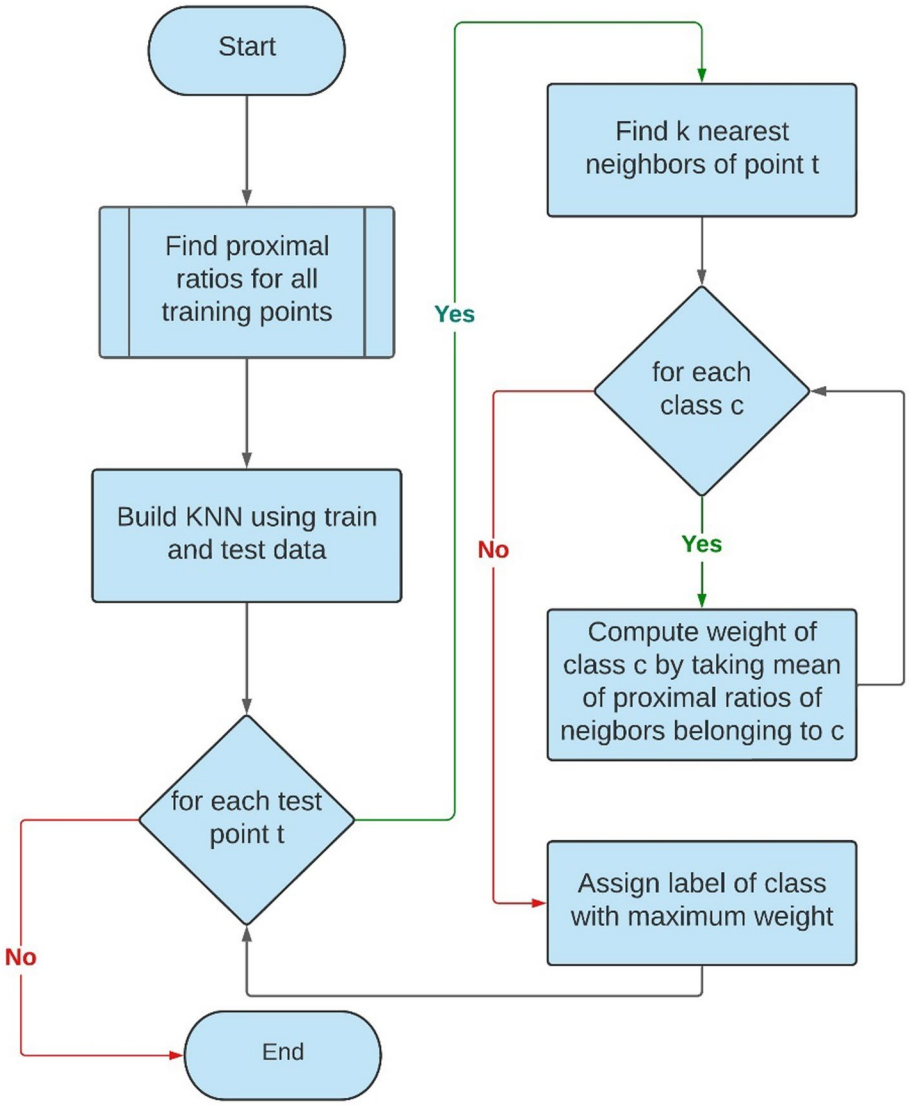

Fig. 5 EPRkNN algorithm

$$ProximalRatio(Pr_t) = \frac{val}{c} \tag{2}$$

where, Pr_t ithe s proximal ratio of t^{th} data point, val is the number of neighbors among k nearest neighbors within a radius having the same label as the target data point. C is the total number of neighbors within the radius. Considering Fig. 2 below, which is drawn for illustration purposes, to simply clarify the mechanism of PR score computation, keeping in mind k = 3. In Fig. 2, Point A belongs to class 1, and there are 4 points inside its radius, but we will consider only the three nearest neighbors. All these three nearest points belong to the same class as point A, so its proximal ratio is $\mathbf{3/3 = 1}$. Point B belongs to class 1, and there are 5 points inside its radius, but again, we will consider the three nearest points. Among these 3 points, 2 points belong to the same class as point B, while 1 point belongs to class 2, so its proximal ratio is $\mathbf{2/3 = 0.666}$. Point C

belongs to class 2; it has only one point inside the radius belonging to the same class, so its proximal ratio is $\mathbf{1/1=1}$. Finally, Point D belongs to class 2. It has a zero point inside the radius belonging to the same class, so its proximal ratio is $\mathbf{0/3=0}$, which means this point is an outlier and will have no contribution when the classification task is made.

On one hand, as we can see in Fig. 2, points A and C are awarded a PR ratio of one because their nearest neighbor is in the same class as them, indicating that they are non-overlapping points. Point B, on the other hand, is regarded as an overlapping point as two of its closest neighbors are in different classes. Equations 1 and 2, therefore, allow us to ascertain the significance of PR in relation to the proximity ratio when identifying the observations. Overall, based on this example, the PR can help separate between overlapping and non-overlapping points. Additionally, as shown by the experimental results and discussion sections, the proposed PR-based kNNs have behaved well over overlapped datasets by combining the classification decisions from multiple test points across different regions. Technically speaking, we can assume that if the observations or points are in the overlap region, they should be assigned lesser PR scores than those assigned to observations in non-overlapping regions, where the PR score range falls into the interval [0..1].

The PR score falls into three categories, which are: (1) The zero value is the score used to identify the actual outliers, as is the case with point D. (2) The one value is the score (as the heaviest weight on all non-overlapping class points) used to identify the actual non-overlapping points, as is the case with points A and C. (3) the value between 0 and 1 as heavily-varied weights assigned to any "over-lapping" point whose nearest neighbors belong to different classes, and each value varies based on the ratio of the nearest neighbor around the target point, as is the case with point B. Mathematically speaking, assuming we have the point P and its nearest neighbors $NP(P) = \{P_1, P_2, P_3, \ldots, P_k\}$ that belong to many classes $C = \{C_1, C_2, C_3, \ldots, C_m\}$ where k and m are the number of nearest neighbors and their classes, the PR score of point (P) can be expressed as given in Eq. 3:

$$PR(P) = \begin{cases} 1, PandNP \in Class_i \\ 1 > PR > 0, PandNP \in C_i, C_{i+1}, .., Class_m \\ 0, Pisirrelevant"anouttlier" \end{cases} \tag{3}$$

Stated differently, the relative weight of observations within the overlapping region(s) varies based on how many of their closest neighbors are affiliated with classes other than the target observation's class. The PR Score Computation Algorithm steps are listed as follows (see Fig. 3):

1. Find the pairwise distance $d_{x,y}$ of all training points, $x, y \in X_{train}$.
2. Build KNN on training data using k nearest neighbors.
3. Extract class-wise data points X_c from training data X_{train}.
4. Find the radius r_c of each class by taking the average of the distances of its points, as given in Eq. 4.

$$r_c = \frac{\sum_{i=1}^{N_c} dist_i}{N_c} \tag{4}$$

where N_c is the number of training point belonging to class c and $dist_i$ is the sum of pairwise distance of point i. The pairwise distance is computed using the Euclidean distance between each training point and every other one. Suppose the training dataset contains n points distributed across two classes, C_1 and C_2. Out of these n points, m_1 points belong to class C_1 and m_2 points belong to class C_2. The pairwise distance matrix for class C_1 will have dimensions $m_1 \times m_1$, while for class C_2, it will be $m_2 \times m_2$. To compute the radius for class C_1, we sum the entries of its $m_1 \times m_1$ matrix, and similarly, the radius for class C_2 is obtained by summing the values in its $m_2 \times m_2$ matrix. Then, for every training point t, do the following:

4.1 Extract neighbors Nb_t of training point t.
4.2 Select all the neighbors with in the radius of class of training point t.
4.3 Compute the proximal ratio for t using Eq. 2.

First version—PRkNN model

In this variation, we calculate the radius of each class as the mean of the distance of all data points in that class. This radius is then used as the radius of every observation belonging to that class, and the nearest neighbors in this radius are used to compute the proximity ratio for that observation. Here are the steps of the PR-kNN algorithm (see Fig. 4), including proximal ratio computation.

1. Compute the proximal ratio $\mathrm{Pr}_1, \mathrm{Pr}_2 \ldots, \mathrm{Pr}_n$ for every point $x_1, x_2 \ldots, x_n$ in training data X_{train}.
2. Make a KNN classifier using k neighbors.
3. Train the classifier on training data X_{train}, labels y_{train}

For every query q in the test data X_{test}, do the following:

4. Find k neighbors N_q and their distance values $dist_q$.
5. Compute weight of each neighbor $n_i \in N_q$ by dividing its Pr value with its distance value using Eq. 5.

$$W_{n_i} = \frac{Pr_{n_i}}{d_{n_i}} \tag{5}$$

6. Using Eq. 6, compute class-wise weights by taking the mean of weights of neighbors belonging to that class.

$$W_c = \frac{\sum_{i=1}^{N_c} W_{n_i}}{N_c} \tag{6}$$

where N_c are neighbors of q belonging to class c.
7. Make predictions by assigning label of class with the highest weight to query q, as given in Eq. 7.

$$\widehat{y} = \text{argmax}(W_c) \tag{7}$$

Overall, the higher the PR score for the point, the less distance there will be between the target point and its nearest neighbors, which should also be smaller than the radius. In consequence, the region's points regarding the decision-making process are stronger the less overlapping they are. As a result, we can expect to achieve a more robust classification decision by integrating the PR notion with kNN and using the most informative points.

Second version—enhanced PRKNN model (EPRkNN)

In this variation, we tried to make weight computation simple and more effective. To achieve this, we only used the average proximal ratios to compute the class weights and did not consider the distance. Here are the steps of the PRkNN algorithm (see Fig. 5), including proximal ratio computation.

8. Compute the proximal ratio $\text{Pr}_1, \text{Pr}_2 \ldots, \text{Pr}_n$ for every point $x_1, x_2 \ldots, x_n$ in training data X_{train}.
9. Make a KNN classifier using k neighbors.
10. Train the classifier on training data X_{train}, labels y_{train}

For every query q in the test data X_{test}, do the following:

11. Find k neighbors N_q and their distance values $dist_q$.
12. Compute class-wise weights by taking the mean of the proximal ratios of neighbors belonging to that class as drawn in Eq. 8.

$$W_c = \frac{\sum_{i=1}^{N_c} Pr_i}{N_c} \tag{8}$$

13. Make predictions by assigning label of class with the highest weight to query q using Eq. 7.

Third version—weighted PRkNN (WPRkNN)

In this variation, we introduced weighted KNN instead of normal KNN to make the algorithm more effective. After the proximal ratios are computed for each training data point, we use these scores as weights in the weighted KNN algorithm. These scores serve as weights in finding the k nearest neighbors, and then these neighbors are used in further processing. We also introduced three weighting factors in this variation, which are as follows:

1. The weights of each class are directly proportional to the number of points in the neighborhood belonging to it (e.g., 1/distance of each neighbor as given in Eq. 9). In other words, the count of each label is the number of neighbors (a class with more labels will have a higher weight).

$$w_{1i} = \frac{1}{D_{i,t}} \tag{9}$$

where, $D_{i,t}$ is the distance of neighbor i concerning query t.

2. The weight of a neighbor is inversely proportional to its distance from the test point. By taking the inverse distance (e.g., 1/distance) of each neighbor, we give the nearest neighbor the heaviest weight. It means that we will not only consider the frequency of class labels in the neighborhood to decide the target labels but will also consider the distance of neighbors. A class with the most labels and nearest neighbors will be assigned as the target class. In other words, the class with more labels in its neighbor will have a higher weight, as given in Eq. 10.

$$w_c = \frac{n_c}{k} \tag{10}$$

where, w_c is the weight of class c, n_c is the number of neighbors belonging to class c, and k is the total number of neighbors. We further enhanced this variation by integrating the weight computation of [38] into our algorithm. In total, three weights are now introduced to significantly improve our algorithm's performance.

3. The third weight is computed using Eq. 11.

$$w_i = \begin{cases} \frac{dist_{xk} - dist_{xi}}{dist_{xk} - dist_{x1}} * \frac{dist_{xk} + dist_{xi}}{dist_{xk} + dist_{x1}} if dist_{xk} \neq dist_{x1} \\ 1 if dist_{xk} = dist_{x1} \end{cases} \tag{11}$$

where,$dist_{xk}$ is the distance of k^{th}(last) neighbor,$dist_{x1}$ is the distance of 1^{st} neighbor, and $dist_{xi}$ is the distance of i^{th} neighbor. The complete procedure for WPRkNN is drawn in Fig. 6 and described in the next steps:

1. Compute the proximal ratio $Pr_1, Pr_2 \ldots, Pr_n$ for every point $x_1, x_2 \ldots, x_n$ in training dataX_{train}.
2. Build a weighted KNN classifier using proximal ratio values as weights for training data points.
3. Train the classifier on training data X_{train}, labels y_{train} and weights Pr.

For every query q in the test data X_{test}, do the following:

a. Find k neighbors N_q and their distance values $dist_q$.
b. Find w1 based on the inverse of each training data point's distance using the Eq. 12.

$$w1_c = \sum_{i=1}^{N_{qc}} \frac{1}{dist_i} \tag{12}$$

where $w1_c$ is the weight of class c and $dist_i$ is the distance of i^{th} neighbor with query q and N_{qc} are the neighbors of q belonging to class c.

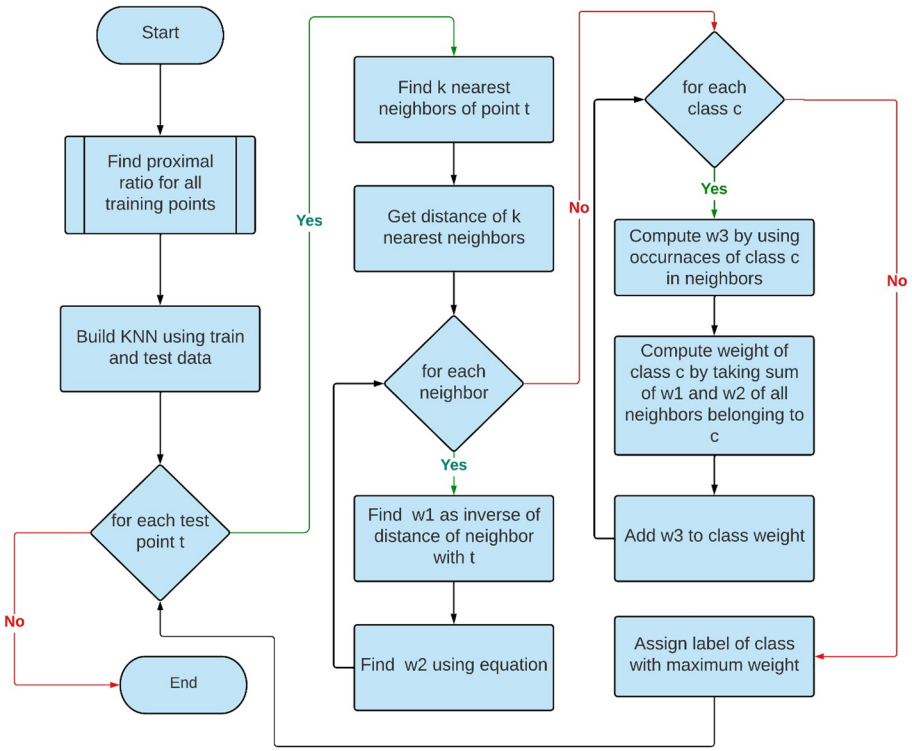

Fig. 6 WPRkNN algorithm

iii. Find w2 based on the occurrence of each class in neighbors as giben in Eq. 13.

$$w2_c = \frac{N_{qc}}{k} \tag{13}$$

where N_{qc} are the number of neighbors of q belonging to class c and k is the total number of neighbours.

d. Find w3 for each class based on Eq. 14, where weights are set to be computed using Eq. 6.

$$w3_c = \sum_{i=1}^{N_{qc}} w_i \tag{14}$$

e. Find the sum of all three weights as Wc in Eq. (15), where w_c is the total weight from total weight for class c

$$w_c = w1_c + w2_c + w3_c \tag{15}$$

f. Assign the class with the highest weight sum to the test data point using Eq. 7.

Table 1 Datasets description

Dataset	# Instances	#Attributes	# Classes	Imbalanced Ratio (IR)	Binary/Multi class	Domain
Banknote	1372	5	2	1.249	Binary	Business
Glass	214	9	7	8.44	Multi	Physics and Chemistry
Ionosphere	351	34	2	1.78	Binary	Physics and Chemistry
Wine	178	13	3	1.479	Multi	Physics and Chemistry
Parkinson	197	22	2	3.06	Binary	Health and Medicine
Sonar	208	60	2	1.144	Binary	Health and Medicine
PLRX	182	13	2	2.48	Binary	EEG signals
Haberman	306	3	2	2.78	Binary	Health and Medicine
Page Blocks	5473	10	5	175.46	Multi	Computer Science
Letter Recognition	20,000	16	26	1.108	Multi	Computer Science
Ecoli	336	8	8	71.5	Multi	Biology
Optical digits	5620	64	10	1.034	Multi	Science
Vowel	990	13	11	1.0	Multi	Computer Science
Robot-Navigation	5456	64	10	6.72	Multi	Engineering
Pen_digits	10,992	16	10	1.08	Multi	Science
Seeds	210	7	3	1.0	Multi	Biology
Transfusion	748	3	2	3.20	Binary	Business
Hill_Valley	606	101	2	1.058	Binary	Science
Musk2	6598	167	2	5.49	Binary	Physics and Chemistry
WPbc	197	34	2	3.19	Binary	Health and Medicine
Wdbc	698	9	2	1.90	Binary	Health and Medicine

It is worth mentioning that the space complexity of proposed models is provided in appendix A.

Experimental setup

We describe the implementation of the proposed models and assess their potential versus twelve leading kNNs using fifty-two datasets across six evaluation metrics. A comparative analysis of the PRkNNs with other state-of-the-art machine learning models is also provided.

All models, including the proposed ones as well as the kNN rivals and other machine learning models, are implemented using Python. For all experiments, we have used an Intel Core i7 (2.7 GHz) processor with 32 GB of RAM and Windows 10 Pro (64-bit). To verify the effectiveness of the proposed models, we have conducted extensive evaluation in six experimental phases. In the first three experiments, evaluation is conducted on

21 real-world numerical[1] datasets (see Table 1) to compare our models against 10 kNN rivals and 5 machine learning models.

The stratified tenfold cross-validation is used for data sampling with four k values (5, 15, 30, and 45) for evaluation phases 1, 2, and 3, respectively. Then, in each experimental phase, the average of the ten runs is used to determine the overall evaluation metrics' values. It is worth mentioning that these datasets were chosen because they are frequently utilized in the assessment of data classification. Furthermore, they are generated to be as close to real-world data as possible, according to the UCI and KEEL repositories [43]. As such, they are perfect for assessing both our models' and their rival models' performance.

Evaluation metrics

To perform the first three experimental phases, we have used three metrics for performance evaluation. The metrics are: accuracy, F1, and ROC which are described by Eqs. 16–18.

Accuracy: Out of the entire collection, it verifies the total number of samples that are correctly classified.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (16)$$

F1-Measure (*F1):* Using F1, each category's classification results are computed, and the averaged results are subsequently determined.

$$F - measure(F1) = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recal}} \quad (17)$$

Area under Roc Curve (AUROC): ROC can be obtained by plotting the false positive rate versus the true positive rate to determine how effectively the model can distinguish between different groups in classification. This curve plots TPR vs. FPR as follows:

$$TPR = TP/(TP + FN) \quad (18)$$

where $TPR = \frac{TP}{TP+FN}$, $FPR = \frac{FP}{FP+TN}$

where the symbols TP, FN, FP, and TN, respectively, stand for the values of true positive, false negative, false positive, and true negative. Notably, we adapt AUROC for multiclass classification by evaluating each class simultaneously against all others, since this metric is primarily intended for binary classes.

Baseline models compared

We investigated ten kNN models, which are: SkNN [3], ExNRule [13], CkNN [19], discriminant DkNN [26], MkNN [31], WLMkNN [32], CDkNN [33], LMkNN [38], MVMCNN [41], ENN [39], and RCKNCN [42]. The default parameter values (if any) of each model have been used as provided by the authors. Since all competing models utilize the Euclidean distance as their benchmark, we adopt it as well.

[1] https://archive.ics.uci.edu/.

Results

In the first experiment, we evaluate the difference in kNN performance between proposed PRkNNs (Figs. 7, 8). In the second experiment, we conduct a thorough comparison between our top model and its related kNN rivals (Tables 2, 3, 4) supported by the significance tests in Tables 1–3 (Appendix B). In the third experiment, we assess our model against five popular ML models (Tables 5, 6, 7). In the fourth experiment, using the leading kNNs out of experiment 2, we made

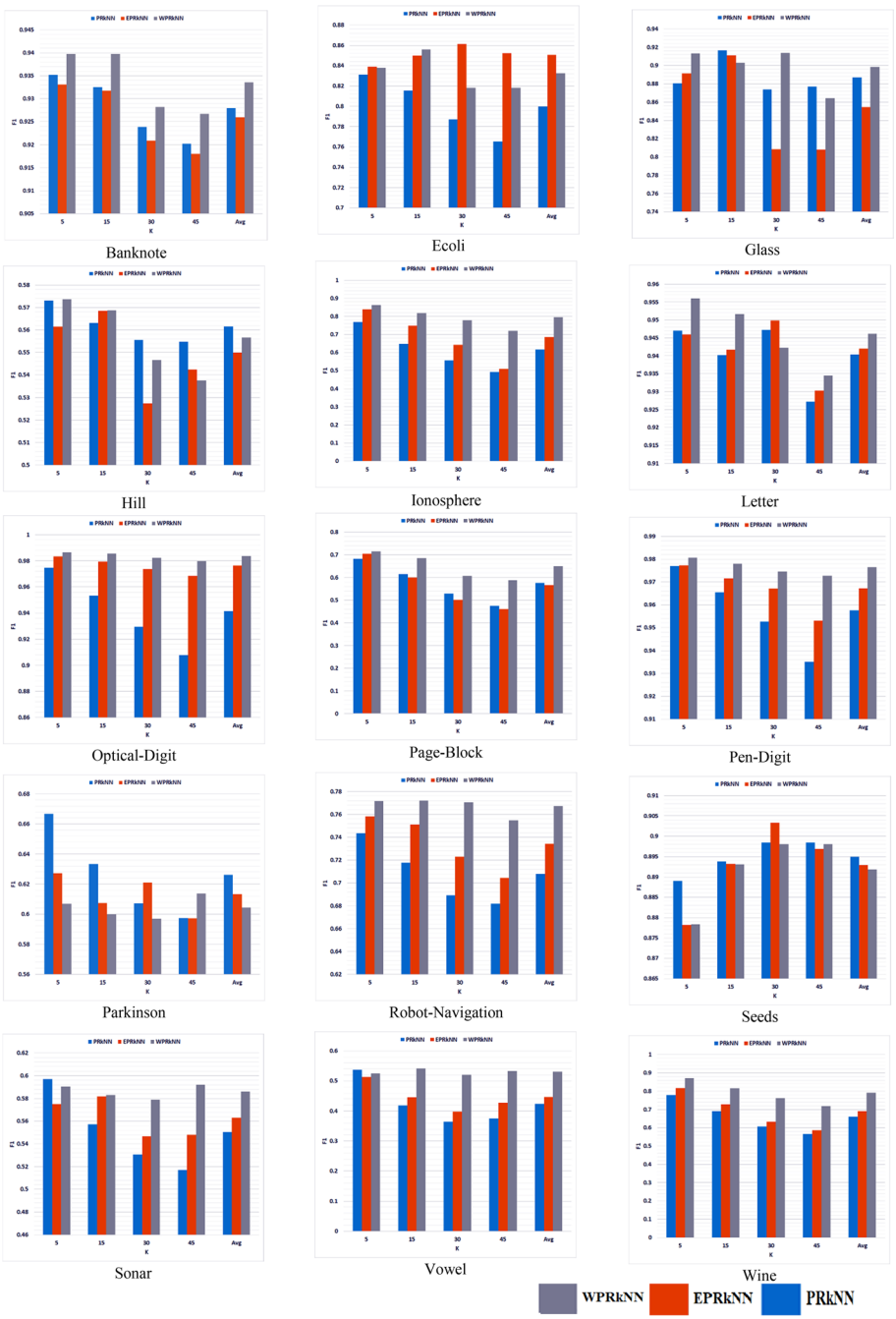

Fig. 7 F1 results in details for all models over all k values

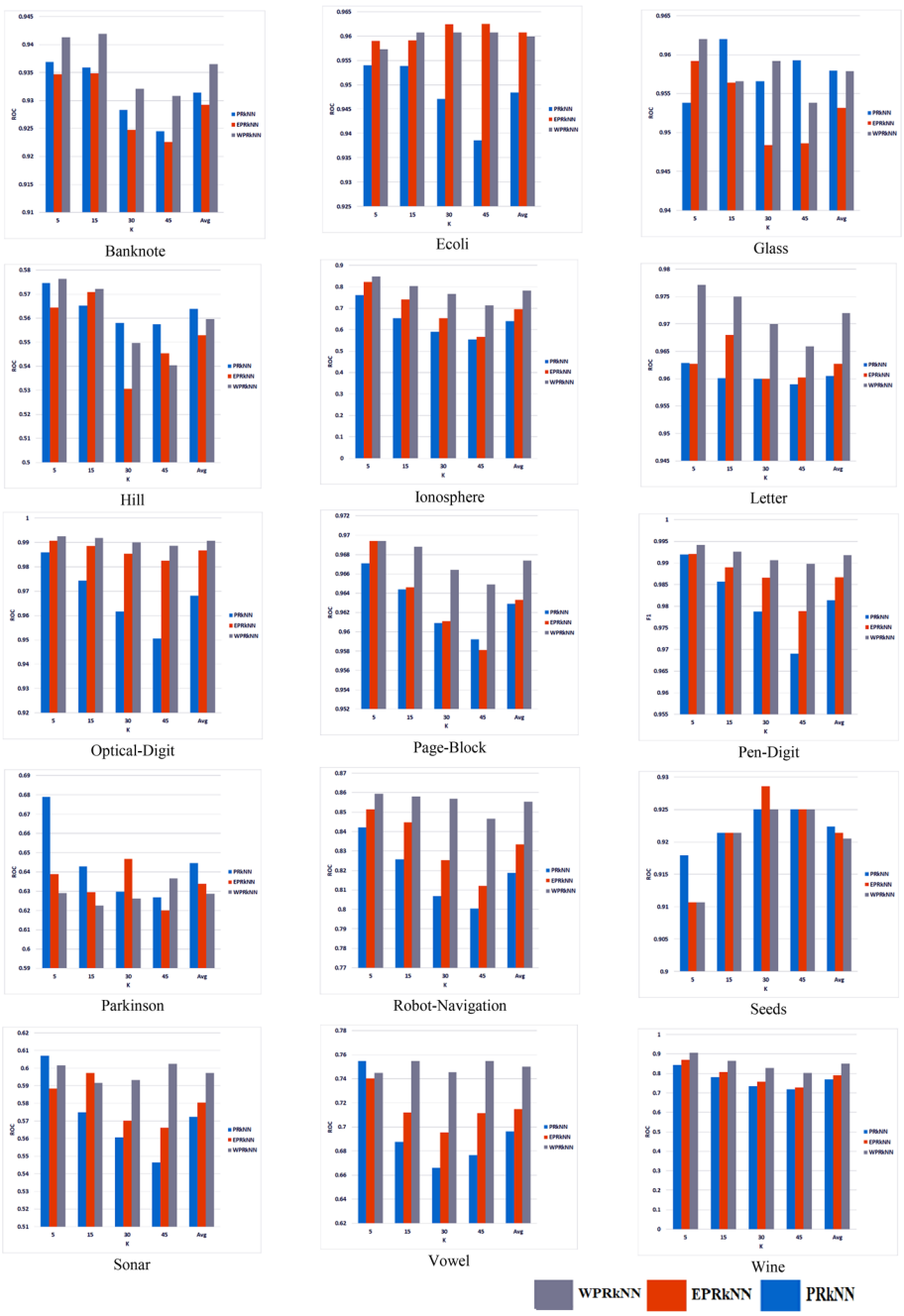

Fig. 8 ROC results in details for all models over all k values

additional experiments over nine benchmarked extremely-imbalanced datasets (Table 8) as given in Tables 9, 10, 11. Finally, in the fifth and sixth experiments, we evaluate our best model (WPRkNN) against the best kNN rivals over 12 noisy and time series datasets (Tables 12, 13), respectively.

Table 2 WPRkNN Model vs. Rival kNNs (Accuracy, Mean ± Standard Deviation)

Dataset	SkNN	DkNN	MVMCNN	WLMkNN	LMkNN	MkNN	CkNN	CDkNN	ExNRule	WPRkNN
Banknote	0.907 ± 0.015	0.895 ± 0.022	0.904 ± 0.015	**0.934 ± 0.008**	0.897 ± 0.016	0.890 ± 0.024	0.922 ± 0.016	0.918 ± 0.016	0.595 ± 0.000	**0.934 ± 0.006**
Glass	0.883 ± 0.039	0.919 ± 0.003	0.892 ± 0.025	0.902 ± 0.021	0.892 ± 0.033	0.874 ± 0.048	0.901 ± 0.051	**0.928 ± 0.008**	0.900 ± 0.007	**0.928 ± 0.005**
Letter	0.920 ± 0.019	0.932 ± 0.022	0.929 ± 0.031	0.933 ± 0.019	0.932 ± 0.023	**0.946 ± 0.007**	0.929 ± 0.023	0.941 ± 0.011	0.785 ± 0.001	**0.946 ± 0.008**
Transfusion	0.734 ± 0.039	0.472 ± 0.023	0.703 ± 0.010	**0.770 ± 0.002**	0.727 ± 0.009	0.745 ± 0.026	0.743 ± 0.028	0.651 ± 0.019	0.590 ± 0.000	0.736 ± 0.036
WPbc	0.732 ± 0.045	0.693 ± 0.029	0.672 ± 0.072	0.653 ± 0.059	0.689 ± 0.084	0.750 ± 0.019	0.712 ± 0.075	0.659 ± 0.062	**0.757 ± 0.004**	0.682 ± 0.086
Landsat	0.873 ± 0.007	0.831 ± 0.021	0.880 ± 0.011	0.880 ± 0.006	0.879 ± 0.006	0.860 ± 0.009	0.879 ± 0.007	**0.893 ± 0.004**	0.885 ± 0.001	0.885 ± 0.003
Wine	0.724 ± 0.077	0.644 ± 0.056	0.766 ± 0.097	0.775 ± 0.067	0.755 ± 0.106	0.687 ± 0.077	0.736 ± 0.091	0.816 ± 0.043	**0.965 ± 0.005**	0.801 ± 0.052
Page Blocks	0.939 ± 0.007	0.918 ± 0.012	0.945 ± 0.010	0.943 ± 0.003	0.945 ± 0.004	0.923 ± 0.011	0.931 ± 0.027	0.945 ± 0.005	0.945 ± 0.000	**0.948 ± 0.003**
PLRX	0.692 ± 0.033	**0.718 ± 0.000**	0.697 ± 0.048	0.693 ± 0.049	0.664 ± 0.047	0.700 ± 0.007	0.692 ± 0.036	0.690 ± 0.023	0.700 ± 0.006	0.705 ± 0.030
Ecoli	0.924 ± 0.008	0.921 ± 0.002	0.914 ± 0.032	0.907 ± 0.021	0.885 ± 0.035	0.921 ± 0.016	0.891 ± 0.258	**0.933 ± 0.004**	0.892 ± 0.001	0.930 ± 0.002
Optical digits	0.977 ± 0.004	0.909 ± 0.047	0.981 ± 0.008	0.981 ± 0.004	**0.984 ± 0.003**	0.969 ± 0.007	0.980 ± 0.004	0.980 ± 0.004	0.958 ± 0.001	**0.984 ± 0.002**
Robot-navigation	0.748 ± 0.024	0.692 ± 0.038	0.771 ± 0.029	0.783 ± 0.004	0.784 ± 0.008	0.734 ± 0.023	0.778 ± 0.009	0.777 ± 0.010	**0.873 ± 0.000**	0.783 ± 0.007
Pen_digits	0.976 ± 0.008	0.917 ± 0.051	0.980 ± 0.009	0.979 ± 0.006	0.985 ± 0.006	0.957 ± 0.022	0.966 ± 0.001	0.983 ± 0.004	0.971 ± 0.001	**0.986 ± 0.003**
Seeds	0.894 ± 0.007	**0.894 ± 0.004**	0.873 ± 0.008	0.892 ± 0.007	0.888 ± 0.002	0.894 ± 0.005	**0.894 ± 0.007**	0.884 ± 0.009	0.866 ± 0.005	**0.894 ± 0.007**
Wdbc	0.505 ± 0.023	0.546 ± 0.003	0.506 ± 0.015	0.490 ± 0.009	0.478 ± 0.020	0.551 ± 0.021	0.508 ± 0.025	0.471 ± 0.004	**0.938 ± 0.003**	0.496 ± 0.015
Win#	0	2	0	1	1	1	1	3	4	7

The bold values are reflect the higher performance

Table 3 WPRkNN Model vs. Rival kNNs (F1, Mean ± Standard Deviation)

Dataset	SkNN	DkNN	MVMCNN	WLMkNN	LMkNN	MkNN	CkNN	CDkNN	ExNRule	WPRkNN
Banknote	0.906 ± 0.015	0.892 ± 0.023	0.893 ± 0.016	**0.934 ± 0.008**	0.896 ± 0.016	0.890 ± 0.024	0.922 ± 0.016	0.917 ± 0.016	0.585 ± 0.000	**0.934 ± 0.006**
Glass	0.743 ± 0.144	0.877 ± 0.007	0.821 ± 0.121	0.860 ± 0.031	0.873 ± 0.030	0.717 ± 0.159	0.855 ± 0.036	**0.907 ± 0.012**	0.865 ± 0.016	0.899 ± 0.020
Letter	0.920 ± 0.019	0.929 ± 0.022	0.930 ± 0.029	0.934 ± 0.019	0.932 ± 0.023	**0.952 ± 0.003**	0.929 ± 0.022	0.942 ± 0.011	0.787 ± 0.001	0.946 ± 0.008
Transfusion	0.479 ± 0.044	0.437 ± 0.017	0.461 ± 0.018	0.520 ± 0.001	0.501 ± 0.017	0.455 ± 0.026	0.489 ± 0.040	0.527 ± 0.012	0.518 ± 0.000	**0.554 ± 0.011**
WPbc	0.424 ± 0.011	0.417 ± 0.011	0.433 ± 0.011	0.402 ± 0.019	0.435 ± 0.022	0.427 ± 0.008	0.418 ± 0.020	0.449 ± 0.033	**0.507 ± 0.014**	0.410 ± 0.021
Landsat	0.843 ± 0.010	0.773 ± 0.035	0.851 ± 0.014	0.854 ± 0.009	0.853 ± 0.012	0.824 ± 0.012	0.852 ± 0.010	**0.860 ± 0.007**	0.858 ± 0.001	**0.860 ± 0.005**
Wine	0.696 ± 0.097	0.584 ± 0.080	0.728 ± 0.102	0.764 ± 0.075	0.740 ± 0.122	0.648 ± 0.102	0.713 ± 0.108	0.811 ± 0.048	**0.967 ± 0.005**	0.792 ± 0.057
Page Blocks	0.550 ± 0.095	0.428 ± 0.074	0.621 ± 0.066	0.610 ± 0.045	0.625 ± 0.056	0.380 ± 0.100	0.564 ± 0.125	0.624 ± 0.062	**0.659 ± 0.006**	0.649 ± 0.052
PLRX	0.415 ± 0.001	0.434 ± 0.000	0.443 ± 0.007	0.446 ± 0.008	0.432 ± 0.032	0.414 ± 0.002	0.414 ± 0.008	0.421 ± 0.018	**0.450 ± 0.011**	**0.450 ± 0.002**
Ecoli	0.808 ± 0.029	0.853 ± 0.012	0.798 ± 0.041	**0.866 ± 0.041**	0.822 ± 0.066	0.789 ± 0.058	0.840 ± 0.186	**0.866 ± 0.003**	0.811 ± 0.006	0.833 ± 0.016
Optical digits	0.977 ± 0.004	0.905 ± 0.049	**0.984 ± 0.012**	0.981 ± 0.004	**0.984 ± 0.003**	0.969 ± 0.008	0.980 ± 0.004	0.980 ± 0.004	0.957 ± 0.001	**0.984 ± 0.003**
Robot-navigation	0.733 ± 0.023	0.631 ± 0.059	0.776 ± 0.008	0.770 ± 0.004	0.762 ± 0.012	0.715 ± 0.024	0.763 ± 0.009	0.763 ± 0.010	**0.848 ± 0.002**	0.767 ± 0.007
pen_digits	0.967 ± 0.008	0.917 ± 0.051	0.970 ± 0.018	0.923 ± 0.030	0.970 ± 0.016	0.949 ± 0.021	0.912 ± 0.001	0.974 ± 0.004	0.963 ± 0.001	**0.976 ± 0.003**
seeds	0.890 ± 0.008	**0.892 ± 0.004**	0.879 ± 0.011	0.890 ± 0.007	0.884 ± 0.002	0.890 ± 0.005	**0.892 ± 0.006**	0.882 ± 0.008	0.864 ± 0.006	**0.892 ± 0.008**
Wdbc	0.388 ± 0.010	0.421 ± 0.007	0.382 ± 0.010	0.406 ± 0.005	0.391 ± 0.016	0.401 ± 0.007	0.392 ± 0.009	0.394 ± 0.009	**0.932 ± 0.004**	0.393 ± 0.002
Win#	0	1	1	2	1	1	1	3	6	7

Table 4 WPRkNN Model vs. Rival kNNs (ROC, Mean ± Standard Deviation)

Dataset	SkNN	DkNN	MVMCNN	WLMkNN	LMkNN	MkNN	CkNN	CDkNN	ExNRule	WPRkNN
Banknote	0.910 ± 0.014	0.889 ± 0.025	0.900 ± 0.013	**0.937 ± 0.007**	0.896 ± 0.017	0.895 ± 0.025	0.926 ± 0.014	0.920 ± 0.015	0.585 ± 0.000	**0.937 ± 0.005**
Glass	0.932 ± 0.022	0.953 ± 0.002	0.938 ± 0.016	0.943 ± 0.012	0.937 ± 0.019	0.926 ± 0.028	0.934 ± 0.029	**0.958 ± 0.005**	0.942 ± 0.004	**0.958 ± 0.003**
Letter	0.958 ± 0.010	0.958 ± 0.016	0.960 ± 0.021	0.965 ± 0.010	0.964 ± 0.012	0.970 ± 0.002	0.963 ± 0.012	0.969 ± 0.006	0.888 ± 0.001	**0.972 ± 0.004**
Transfusion	0.523 ± 0.022	0.538 ± 0.010	0.510 ± 0.101	0.536 ± 0.001	0.524 ± 0.014	0.508 ± 0.009	0.529 ± 0.022	0.555 ± 0.011	0.547 ± 0.000	**0.573 ± 0.010**
WPbc	0.490 ± 0.013	0.465 ± 0.019	0.492 ± 0.011	0.447 ± 0.025	0.486 ± 0.023	0.492 ± 0.013	0.478 ± 0.029	0.487 ± 0.029	**0.532 ± 0.008**	0.464 ± 0.033
Landsat	0.923 ± 0.004	0.899 ± 0.013	0.927 ± 0.009	0.928 ± 0.004	0.928 ± 0.003	0.916 ± 0.005	0.927 ± 0.004	**0.935 ± 0.003**	0.933 ± 0.001	0.931 ± 0.002
Wine	0.793 ± 0.058	0.733 ± 0.042	0.821 ± 0.088	0.831 ± 0.050	0.816 ± 0.079	0.765 ± 0.058	0.802 ± 0.068	0.862 ± 0.033	**0.974 ± 0.004**	0.851 ± 0.040
Page Blocks	0.962 ± 0.004	0.949 ± 0.008	0.962 ± 0.009	**0.967 ± 0.002**	0.965 ± 0.002	0.952 ± 0.007	0.957 ± 0.017	0.962 ± 0.003	0.965 ± 0.000	**0.967 ± 0.002**
PLRX	0.488 ± 0.018	**0.718 ± 0.000**	0.501 ± 0.031	0.503 ± 0.027	0.487 ± 0.017	0.497 ± 0.005	0.430 ± 0.016	0.490 ± 0.010	0.509 ± 0.008	0.509 ± 0.016
Ecoli	0.957 ± 0.004	0.510 ± 0.000	0.944 ± 0.020	0.947 ± 0.012	0.934 ± 0.020	0.954 ± 0.009	0.923 ± 0.147	0.956 ± 0.002	0.938 ± 0.001	**0.960 ± 0.015**
Optical digits	0.987 ± 0.002	0.955 ± 0.001	0.989 ± 0.010	0.989 ± 0.002	**0.991 ± 0.002**	0.983 ± 0.004	0.989 ± 0.002	0.989 ± 0.002	0.976 ± 0.000	**0.991 ± 0.001**
Robot-navigation	0.832 ± 0.016	0.795 ± 0.026	0.852 ± 0.019	0.859 ± 0.003	0.856 ± 0.005	0.823 ± 0.016	0.852 ± 0.006	0.852 ± 0.007	**0.915 ± 0.000**	0.855 ± 0.005
pen_digits	0.987 ± 0.004	0.955 ± 0.028	0.988 ± 0.011	0.988 ± 0.003	0.990 ± 0.003	0.976 ± 0.012	0.971 ± 0.000	**0.992 ± 0.002**	0.984 ± 0.001	**0.992 ± 0.001**
seeds	0.917 ± 0.005	**0.921 ± 0.003**	0.903 ± 0.029	0.919 ± 0.005	0.916 ± 0.001	0.917 ± 0.003	**0.921 ± 0.005**	0.913 ± 0.006	0.899 ± 0.004	**0.921 ± 0.005**
Wdbc	0.416 ± 0.008	0.448 ± 0.004	0.409 ± 0.010	0.420 ± 0.005	0.402 ± 0.016	0.420 ± 0.010	0.419 ± 0.035	0.402 ± 0.007	**0.933 ± 0.005**	0.415 ± 0.007
Win#	0	2	0	2	1	0	1	3	4	9

The bold values are reflect the higher performance

Table 5 WPRkNN Model vs. ML models (Results in Average – Accuracy, Mean ± Standard Deviation)

Dataset	SVM	MNB	NB	DT	ANN	WPRkNN
Banknote	0.776 ± 0.039	0.810 ± 0.023	0.642 ± 0.039	0.908 ± 0.032	0.873 ± 0.026	**0.938 ± 0.006**
Glass	**0.990 ± 0.018**	0.827 ± 0.096	0.827 ± 0.069	0.958 ± 0.038	0.742 ± 0.081	0.928 ± 0.005
Letter	0.831 ± 0.012	0.537 ± 0.008	0.634 ± 0.009	0.875 ± 0.007	0.748 ± 0.019	**0.946 ± 0.008**
Transfusion	0.759 ± 0.015	0.690 ± 0.077	0.745 ± 0.045	**0.771 ± 0.049**	0.672 ± 0.134	0.736 ± 0.036
WPbc	0.680 ± 0.123	0.356 ± 0.075	**0.731 ± 0.074**	0.680 ± 0.123	0.543 ± 0.253	0.682 ± 0.086
Landsat	0.865 ± 0.015	0.772 ± 0.015	0.796 ± 0.016	0.858 ± 0.018	0.725 ± 0.113	**0.885 ± 0.003**
Wine	0.910 ± 0.055	0.871 ± 0.096	**0.960 ± 0.044**	0.909 ± 0.059	0.571 ± 0.155	0.801 ± 0.052
Page Blocks	**0.966 ± 0.005**	0.915 ± 0.010	0.909 ± 0.031	0.960 ± 0.008	0.952 ± 0.008	0.948 ± 0.003
PLRX	**0.713 ± 0.019**	**0.713 ± 0.019**	0.580 ± 0.127	0.552 ± 0.072	0.712 ± 0.019	0.705 ± 0.030
Ecoli	**0.991 ± 0.014**	0.764 ± 0.024	0.961 ± 0.026	0.970 ± 0.032	0.570 ± 0.094	0.930 ± 0.002
Optical digits	0.979 ± 0.006	0.907 ± 0.012	0.798 ± 0.022	0.906 ± 0.014	0.970 ± 0.006	**0.983 ± 0.002**
Robot-navigation	0.736 ± 0.012	0.587 ± 0.018	0.540 ± 0.023	**0.994 ± 0.004**	0.869 ± 0.013	0.783 ± 0.007
pen_digits	0.976 ± 0.007	0.708 ± 0.018	0.815 ± 0.008	0.949 ± 0.012	0.894 ± 0.056	**0.986 ± 0.003**
seeds	0.885 ± 0.068	0.781 ± 0.057	0.894 ± 0.070	0.890 ± 0.073	0.742 ± 0.102	**0.896 ± 0.007**
Wdbc	0.859 ± 0.066	0.925 ± 0.042	0.855 ± 0.060	**0.951 ± 0.025**	0.543 ± 0.253	0.496 ± 0.015
Win#	4	1	2	3	0	6

The bold values are reflect the higher performance

Table 6 WPRkNN Model vs. ML models (Results in Average—F1, Mean ± Standard Deviation)

Dataset	SVM	MNB	NB	DT	ANN	WPRkNN
Banknote	0.775 ± 0.040	0.806 ± 0.031	0.601 ± 0.052	0.906 ± 0.033	0.873 ± 0.026	**0.937 ± 0.006**
Glass	**0.971 ± 0.057**	0.683 ± 0.178	0.809 ± 0.098	0.899 ± 0.114	0.409 ± 0.105	0.899 ± 0.020
Letter	0.831 ± 0.011	0.525 ± 0.007	0.631 ± 0.009	0.875 ± 0.007	0.747 ± 0.019	**0.946 ± 0.008**
Transfusion	0.481 ± 0.053	0.637 ± 0.076	0.532 ± 0.105	**0.601 ± 0.066**	0.486 ± 0.106	0.554 ± 0.011
WPbc	**0.565 ± 0.144**	0.349 ± 0.078	0.532 ± 0.105	**0.565 ± 0.144**	0.341 ± 0.109	0.410 ± 0.021
Landsat	0.832 ± 0.019	0.691 ± 0.016	0.778 ± 0.016	0.836 ± 0.019	0.64 ± 0.106	**0.860 ± 0.005**
Wine	0.910 ± 0.067	0.876 ± 0.092	**0.961 ± 0.044**	0.908 ± 0.064	0.523 ± 0.176	0.792 ± 0.057
Page Blocks	0.788 ± 0.046	0.612 ± 0.051	0.613 ± 0.057	**0.802 ± 0.058**	0.635 ± 0.116	0.649 ± 0.052
PLRX	0.416 ± 0.006	0.416 ± 0.006	0.418 ± 0.091	**0.461 ± 0.079**	0.416 ± 0.006	0.450 ± 0.002
Ecoli	**0.958 ± 0.068**	0.446 ± 0.067	0.895 ± 0.087	0.846 ± 0.164	0.240 ± 0.074	0.833 ± 0.016
Optical digits	0.979 ± 0.005	0.907 ± 0.012	0.798 ± 0.022	0.906 ± 0.014	0.970 ± 0.006	**0.984 ± 0.003**
Robot-navigation	0.708 ± 0.022	0.520 ± 0.015	0.548 ± 0.022	**0.992 ± 0.006**	0.852 ± 0.015	0.767 ± 0.007
pen_digits	0.950 ± 0.042	0.621 ± 0.015	0.801 ± 0.026	0.923 ± 0.042	0.883 ± 0.063	**0.976 ± 0.003**
seeds	0.885 ± 0.065	0.779 ± 0.059	0.892 ± 0.067	0.888 ± 0.075	0.709 ± 0.131	**0.894 ± 0.008**
Wdbc	0.824 ± 0.091	0.918 ± 0.046	0.820 ± 0.085	**0.945 ± 0.028**	0.367 ± 0.056	0.393 ± 0.002
Win#	3	0	1	6	0	6

The bold values are reflect the higher performance

Experiment 1

The results obtained for F1 and ROC for proposed PRkNNs are shown in Figs. 7, 8, which demonstrate that WPRkNN is the best performer. It is seen that WPRkNN is the most effective over banknote, with PRkNN coming in second in all k values and the averaged performance. Both EPRkNN and WPRkNN are proven to be the most successful over Ecoli, nevertheless.

Table 7 WPRkNN Model vs. ML models (Results in Average—ROC, Mean ± Standard Deviation)

Dataset	SVM	MNB	NB	DT	ANN	WPRkNN
Banknote	0.779 ± 0.040	0.810 ± 0.031	0.615 ± 0.043	0.906 ± 0.033	0.876 ± 0.029	**0.934 ± 0.005**
Glass	**0.994 ± 0.010**	0.898 ± 0.056	0.899 ± 0.040	0.975 ± 0.022	0.850 ± 0.047	0.958 ± 0.003
Letter	0.912 ± 0.006	0.759 ± 0.004	0.810 ± 0.005	0.935 ± 0.003	0.869 ± 0.010	**0.972 ± 0.004**
Transfusion	0.521 ± 0.029	**0.671 ± 0.072**	0.559 ± 0.045	0.592 ± 0.057	0.564 ± 0.077	0.572 ± 0.010
WPbc	**0.567 ± 0.156**	0.534 ± 0.084	0.543 ± 0.083	**0.567 ± 0.156**	0.483 ± 0.061	0.464 ± 0.033
Landsat	0.919 ± 0.009	0.863 ± 0.008	0.877 ± 0.009	0.915 ± 0.011	0.835 ± 0.068	**0.931 ± 0.002**
Wine	0.934 ± 0.049	0.903 ± 0.072	**0.970 ± 0.033**	0.932 ± 0.044	0.678 ± 0.116	0.851 ± 0.040
Page Blocks	**0.978 ± 0.003**	0.947 ± 0.006	0.943 ± 0.019	0.975 ± 0.005	0.970 ± 0.005	0.967 ± 0.002
PLRX	0.500 ± 0.000	0.500 ± 0.000	0.439 ± 0.102	0.468 ± 0.082	0.500 ± 0.000	**0.509 ± 0.016**
Ecoli	**0.994 ± 0.007**	0.865 ± 0.014	0.978 ± 0.015	0.983 ± 0.018	0.754 ± 0.053	0.960 ± 0.015
Optical digits	0.988 ± 0.003	0.948 ± 0.006	0.888 ± 0.012	0.947 ± 0.008	0.983 ± 0.003	**0.991 ± 0.001**
Robot-navigation	0.824 ± 0.008	0.725 ± 0.012	0.693 ± 0.015	**0.996 ± 0.003**	0.913 ± 0.009	0.855 ± 0.005
pen_digits	0.987 ± 0.003	0.839 ± 0.010	0.898 ± 0.004	0.972 ± 0.007	0.942 ± 0.030	**0.992 ± 0.001**
seeds	0.914 ± 0.051	0.835 ± 0.042	0.921 ± 0.052	0.917 ± 0.055	0.807 ± 0.076	**0.922 ± 0.005**
Wdbc	0.808 ± 0.088	0.534 ± 0.084	0.803 ± 0.083	**0.943 ± 0.031**	0.500 ± 0.000	0.415 ± 0.007
Win#	4	1	1	3	0	7

The bold values are reflect the higher performance

On the other hand, taking the F1 results in detail, in Fig. 7, it is seen that over Ionosphere, Letter, Sonar, and Vowel, WPRkNN, followed by EPRkNN, is the most effective with the highest F1. Conversely, PRkNN is the worst. Over glass and page blocks, WPRkNN, followed by PRkNN, continues to be the best performer, while EPRkNN performs the poorest due to its lowest F1. In contrast to Parkinson, Seeds, and Haber, PRkNN, followed by EPRkNN, has the highest F1, and WPRkNN has the lowest. Except in the rare situation of k 30 over seeds, EPRkNN is better than both WPRkNN and PRkNN. It is worth noting that both PRkNN and EPRkNN exhibit a nearly identical F1 performance trend over letter and page blocks, with EPRkNN being slightly better over the former and PRkNN being better over the latter. Finally, over Hill-Valley, PRkNN and WPRkNN are competing fiercely, with PRkNN being better.

Regarding the ROC results in Fig. 8, (over Ionosphere, Letter, Sonar, and Vowel), WPRkNN, followed by EPRkNN, is the most effective with the highest ROC. Conversely, PRkNN is the worst. Over glass and page blocks, WPRkNN, followed by PRkNN, continues to be the best performer, with PRkNN being the best over glass. The EPRkNN performs the poorest due to its lowest ROC over Glass in particular.

It is worth noting that both PRkNN and EPRkNN exhibit a nearly identical ROC performance trend over the page blocks dataset, with EPRkNN being slightly better in the averaged performance. Incidentally, both WPRkNN and PRkNN have almost identical averaged performance over the letter dataset. Meanwhile, similarly to F1 over Parkinson, Seeds, and Haber, PRkNN, followed by EPRkNN, has the highest ROC, and WPRkNN has the lowest. Except in the rare situation of k 30 over seeds, EPRkNN is better than both WPRkNN and PRkNN. Finally, over Hill-Valley, PRkNN and WPRkNN are competing fiercely, with PRkNN being significantly better. In summary, regarding both F1 and ROC metrics, WPRkNN outperforms all models over six datasets, showing

stable performance over all k values, and PRkNN is the best over four datasets. Therefore, WPRkNN is generally our best-performing model due to this behavior tendency.

To sum up, WPRkNN is dominating the competition; stable, consistent, less sensitive to the k value, and the top performer over all k values and the averaged performance across both metrics. It is seen that in 12 datasets out of 16, WPRkNN has been the best performer, showing stable and reliable performance concerning both evaluation metrics over all datasets.

Experiment 2

In this experiment, we compare the relevant kNNs with our efficient WPRkNN (see Tables 2, 3, 4). The last row displays the number of times each kNN model has generally achieved its best performance. The best values secured are indicated in bold font in table's cells. Overall, the results show that our proposed WPRkNN is highly competitive, drawing outstanding performance across the vast majority of datasets used regarding all evaluation metrics.

As can be seen from Table 2, WPRkNN has been the top performer, securing 7 wins, followed by ExNRule and CDkNN. We also observed that in certain high-dimensional datasets, such as banknotes, letters, Ecoli (which is also extremely imbalanced), and transfusion, the ExNRule does, in fact, have low performance. The reason is that, in high-dimensional datasets, ExNRule may actually select all zero values when selecting samples and subsets of features.

Tables 3 and 4 continue to demonstrate how much better WPRkNN is than its competitors, with 7 and 9 wins, respectively, to take the top spot. ExNRule and CDkNN with F1, and ExRule and LMkNN with ROC are next. The WPRkNN has proven to be more successful with ROC in particular, demonstrating its capacity to mitigate the effects of bot class imbalance and overlapping impacts. Experiment 4 provides more assessment in this respect.

We can see from Tables 2, 3, 4 and Figs. 7, 8 that we have some top-performing kNNs (WPRkNN, EXNRule, CDkNN, and LMkNN) and others with degrading performance (SkNN, MVMCNN, DkNN, WLMkNN, and MkNN). For two reasons, the MVMCNN has surprisingly performed poorly across all metrics. Firstly, despite being highly efficient, MVMCNN cannot defeat its competitors in terms of accuracy, F1, and ROC due to their clustering-based abilities. The MVMCNN's poor behavior is a result of its design, which relies on clustering. Typically, clustering adoption can improve the kNN's efficiency but decrease its performance in return, even against SkNN. The MVMCNN limits the kNN's performance to small, confined regions that are recognized as clusters, which reduces the kNN's effectiveness while increasing its efficiency. Nevertheless, the MVMCNN voting mechanism contributes to the model's sustained performance over SkNN and, in certain instances, against other competing kNNs.

On the other hand, CkNN has not shown competitive performance vs. many models, including WPRkNN, LMkNN, and WLMkNN, because the competitors drawn in this paper are strong enough to be defeated, while the CkNN model was evaluated over weak kNNs and for small k values.

Moreover, our experimental study proved that the CkNN performance is even less than standard kNN for larger values of k, which goes in parallel with what the authors

clarified in their work. Surprisingly, both LMkNN and CkNN have close performance trends over the majority of datasets. We also provide Figs. 9, 10 to illustrate the detailed results of both accuracy and ROC metrics for the best seven kNN models across all datasets. Overall, both figures indicate that our WPRkNN is indeed competitive over the great majority and highly effective with RC in particular. The statistical tests are provided in Appendix B.

As seen in Figs. 9, 10, the performance of the proposed PRkNN models has been attributed to the combination of the PR scores, which were used to identify the neighbors' relevance and reduce the influence of irrelevant points, and the weighting schemes, which were used to lessen the unbalanced negative impact. Experimentally, such a combination has led to reliable kNN models that attempt to lessen the negative impact of class imbalance and overlapping problems and decrease their influence on the classification task.

Last but not least, Figs. 11, 12 illustrate the run time in seconds (training + testing) for all considered kNN models over all datasets. As can be seen from these figures, CDkNN, followed by SkNN, WLMkNN, and MVMCNN, is the most efficient variation. We noticed the efficiency of CDkNN due to the fact that it is based on simple class-wise computation represented in simple matrices. For MVMCNN, the

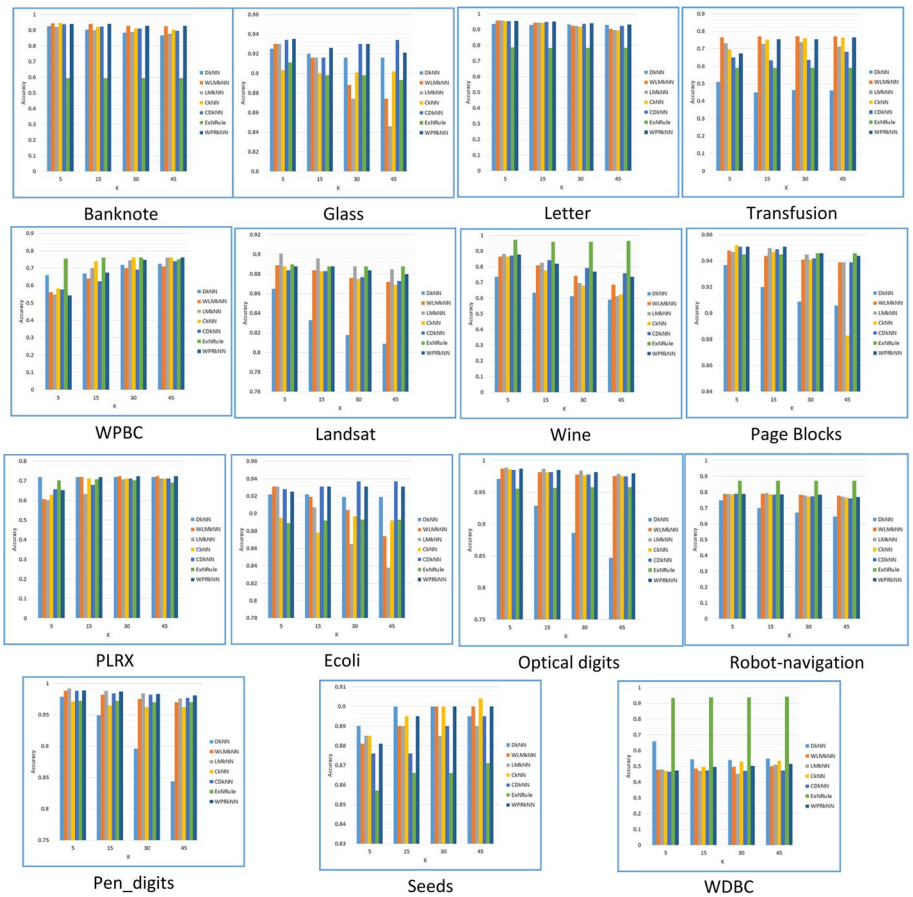

Fig. 9 Accuracy results of all kNN models over each k value across each data set

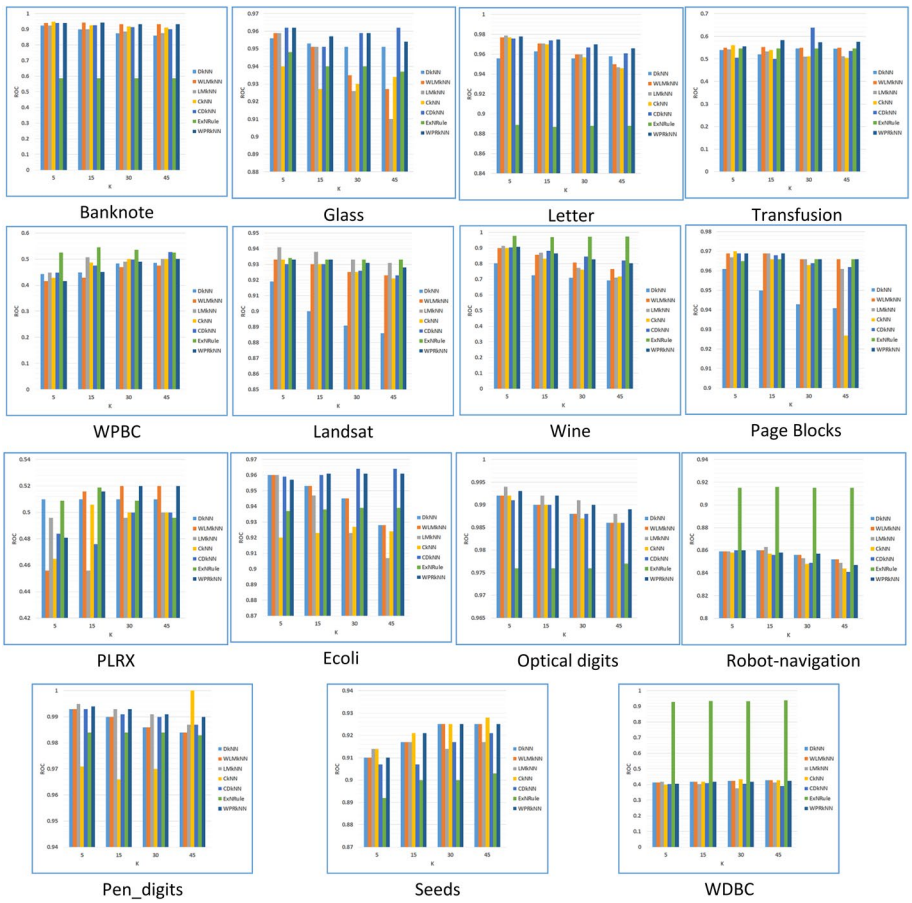

Fig. 10 ROC results of all kNN models over each k value across each data set

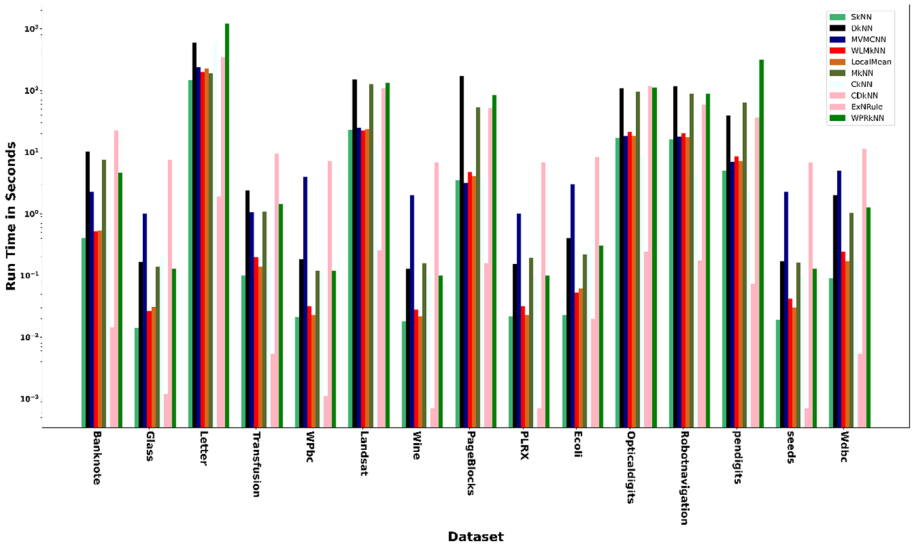

Fig. 11 Averaged runtime (in Seconds) for kNNs over datasets

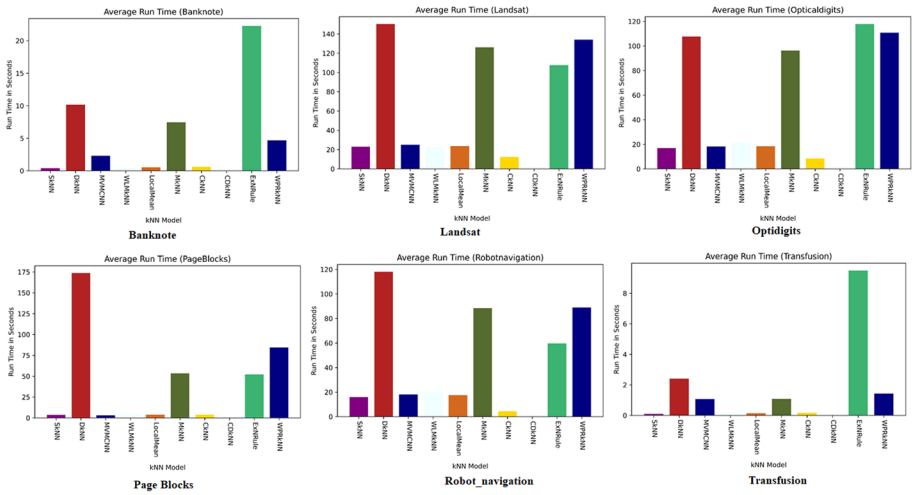

Fig. 12 Averaged runtime (in Seconds) for kNN models over some high-dimensional datasets

key reason behind its efficiency is its clustering-based design, which also, as just explained, has a negative impact on its efficacy. The LMkNN and WPRkNN come next in terms of efficiency to be compromised solutions between the most efficient group (CDkNN, WLMkNN, MVMCNN, CkNN, and SkNN) and the non-efficient group (DkNN, MkNN, and ExNRulekNN). CkNN can also be designated as non-efficient in many cases as it is a conditional version by nature.

In other words, before performing the classification, CkNN needs to check the class probabilities and check the class distributions overlap. Both operations take undeniable time from the point of checking until classification for each testing point. MkNN takes a longer time than CkNN because it needs to assign validity values to each training sample to get better results. These values have to be computed before performing the classification, which has a negative impact on the MkNN's run time. On the flip side, the DkNN is an inefficient kNN model as it uses the discernibility index, which needs a lengthy time to be computed for each training set. Further, for each test point, the DkNN considers all data points inside the radius, which also contributes to its inefficiency problem.

Finally, ExNRulekNN has been designated to be the most inefficient kNN model due to two reasons: however, since ExNRulekNN applies two layers of majority voting and runs the kNN classifier on a subset of data almost 500 times (B = 500 for the base classifiers, for example), illustrating its severe inefficiency. It is worth indicating that while our proposed models come in the middle of the efficient models list, this happens due to the nature of the models' design, which necessitates PR score computation and weighting scheme combinations. Both techniques have unescapable negative effects on their run time. However, considering the advantages of the models we propose (addressing the noise "irrelevant points," overlapping and imbalance concerns, improving accuracy, and enhancing generalizability), the computational cost disadvantage might be logically rational. Furthermore, rather than running time, the methodology's reliability is the primary focus of our research.

Table 8 Datasets description

Dataset	Dataset	# Instances	#Attributes	IR	FI
winequalityred3vs5	D1	691	11	68.10	0.7509
yeast2vs8	D2	482	8	23.10	1.1424
yeast4	D3	1484	8	28.09	1.2516
ecoli0146vs5	D4	280	6	13	1.3345
ecoli067vs5	D5	220	6	10.00	1.6921
Car-good	D6	1728	6	24.04	0.6837
dermatology-6	D7	358	34	16.9	12.041
cleveland-0_vs_4	D8	173	13	12.62	0.0632

Table 9 WPRkNN vs. Best kNNs (F1, Mean ± SD)

Dataset	CDkNN	ExNRule	WPRkNN
D1	**0.499 ± 0.010**	0.496 ± 0.000	**0.499 ± 0.010**
D2	**0.816 ± 0.020**	0.489 ± 0.000	0.812 ± 0.020
D3	**0.591 ± 0.037**	0.491 ± 0.000	0.582 ± 0.051
D4	0.896 ± 0.019	0.798 ± 0.025	**0.904 ± 0.007**
D5	**0.875 ± 0.023**	0.555 ± 0.020	0.873 ± 0.014
D6	0.613 ± 0.100	0.489 ± 0.000	**0.630 ± 0.089**
D7	0.909 ± 0.020	0.870 ± 0.025	**0.914 ± 0.024**
D8	0.490 ± 0.032	0.480 ± 0.000	**0.501 ± 0.044**
Win#	4	0	**5**

The bold values are reflect the higher performance

Experiment 3

We further compare our effective WPRkNN with popular ML models in this subsection. The last row (in Tables 5, 6, 7) shows how many times each kNN model has generally secured its best performance, while the bolded values refer to the top performance secured for each model on average for the corresponding dataset. The results explain themselves and clearly show that our proposed models have the power to compete with these strong ML models, achieving competitive performance and even much better accuracy and ROC. The parameter values to implement these ML models [44, 45] are provided in Appendix (C).

Experiment 4: highly-imbalanced datasets

In this experiment, we further validate the effectiveness of WPRkNN model by comparing it to its strong rivals over eight highly imbalanced datasets. This evaluation is done in order to further validate its superiority over the top-performing kNNs from experimental phase 2, namely, ExNRule and CDkNN. The datasets are often used in the literature and are accessible to the public in the KEEL repository [43]. Table 8 lists the datasets along with their IR rate and the overlapping degree as determined by Fisher's discriminant ratio (FI) [21]. To conduct these experiments, we tested all considered kNNs using fivefold cross-validation over nine extremely-imbalanced datasets with a varied IR ratio and k ranging from 1 to 25 with a step of 2

Table 10 WPRkNN vs. Rival kNNs (ROC, Mean ± SD)

Dataset	CDkNN	ExNRule	WPRkNN
D1	0.503 ± 0.012	0.500 ± 0.000	**0.504 ± 0.012**
D2	**0.760 ± 0.021**	0.500 ± 0.000	0.754 ± 0.022
D3	**0.565 ± 0.037**	0.500 ± 0.000	0.560 ± 0.043
D4	0.862 ± 0.026	0.747 ± 0.020	**0.872 ± 0.006**
D5	**0.832 ± 0.029**	0.548 ± 0.012	**0.832 ± 0.019**
D6	0.585 ± 0.075	0.500 ± 0.000	**0.597 ± 0.080**
D7	0.946 ± 0.012	0.815 ± 0.031	**0.949 ± 0.014**
D8	0.505 ± 0.026	0.500 ± 0.000	**0.515 ± 0.032**
Win#	3	0	**6**

The bold values are reflect the higher performance

Table 11 WPRkNN vs. Rival kNNs (GM, Mean ± SD)

Dataset	CDkNN	ExNRule	WPRkNN
D1	**0.010 ± 0.037**	0.000 ± 0.000	**0.010 ± 0.037**
D2	**0.711 ± 0.026**	0.000 ± 0.000	0.703 ± 0.028
D3	**0.328 ± 0.110**	0.000 ± 0.000	0.293 ± 0.152
D4	0.835 ± 0.035	0.669 ± 0.050	**0.852 ± 0.011**
D5	**0.807 ± 0.039**	0.192 ± 0.047	**0.807 ± 0.027**
D6	0.320 ± 0.233	0.000 ± 0.000	**0.365 ± 0.203**
D7	0.944 ± 0.013	0.780 ± 0.041	**0.947 ± 0.014**
D8	0.036 ± 0.099	0.000 ± 0.000	**0.065 ± 0.131**
Win#	4	0	**6**

The bold values are reflect the higher performance

(k = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, and 25). Then, the averaged performance is reported in Tables 9, 10, 11. We also add one more popular evaluation metric, which is the geometric mean (GM), as one of the most widely used to evaluate the models with imbalanced datasets (see Eqs. 19–21). As previously indicated, the last row in each table displays the average number of times each kNN model has achieved its best performance, and the bolded font values denote the average top performance achieved by each kNN model for the associated dataset.

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{19}$$

$$Sensitivity = \frac{True\ Positive}{True\ Positive + False\ negative} \tag{20}$$

$$GM = \sqrt{Specificity \times Sensitivity} \tag{21}$$

As shown in Tables 9, 10, 11 and Fig. 13, our WPRkN behaves better than its rivals (supported by the significance tests in Tables 4–6 in the appendix in our Github repository) due to its simple design combined with the proposed PR technique and weighting schemes. Both PR and the weighting method of the WPRkNN classifier greatly boost its competitiveness against powerful kNN extensions (CDkNN and

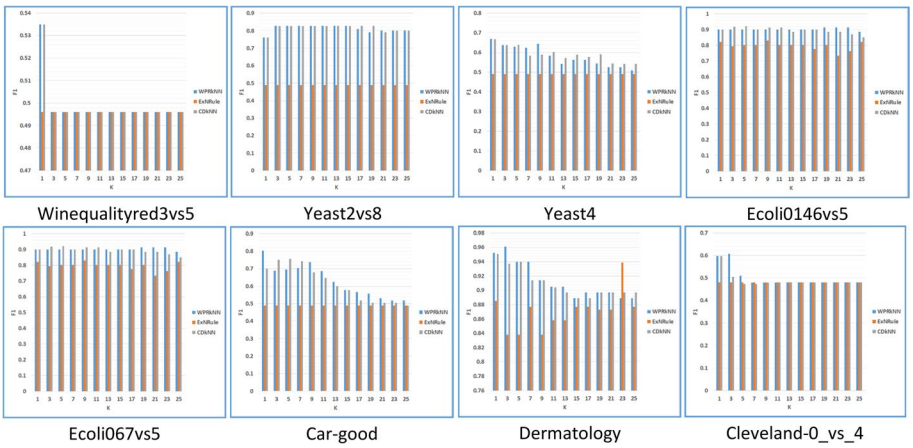

Fig. 13 F1 Results of all kNN models over each k value across each data set

Table 12 The used KEEL noisy datasets

Dataset	# Total Instances	#Testing Instances	#Attributes	Classes
Wine-an-nn	178	58	14	3
wdbc-an-nn	569	187	30	2
Segment-an-nn	2310	770	19	7
Page-blocks-an-nn	5473	1823	10	5
Hear h-an-nn	270	70	13	2
iris-an-nn	150	60	4	3
sonar-an-nn	208	66	60	2

ExNRule). The discussion below provides additional insights on the behavior of all studied kNN models.

Experiment 5: noisy datasets

In this experiment, we also further validate the potential competency of the WPRkNN model by comparing it to CDkNN, ExNRulekNN, ENN, and RCKNCN over six noisy datasets (see Table 12) taken from the KEEL repository [43]. Overall, a noisy dataset refers to a dataset where a portion of the data contains random variations, errors, or inconsistencies that do not reflect the true underlying patterns or relationships in the data. These distortions, or "noise," can arise from a variety of sources, such as imperfections in data collection, measurement errors, misreporting, or external factors that influence the data in unpredictable ways. In the context of this particular process, noise is intentionally introduced into the dataset by randomly modifying a percentage of the data points. Specifically, random alterations are made by replacing the values in a subset of data points with new values that are sampled uniformly from the minimum and maximum values of the respective columns. This process distorts the true values of the dataset, adding uncertainty and making it less predictable; (2) the proportion of data affected by the noise is controlled by a parameter called the noise_level, which determines how much of the dataset will be altered (in this case,

Table 13 WPRkNN vs. Rival kNNs (Accuracy, Mean ± SD)

Dataset/kNN	ExNRulekNN	CDkNN	ENN	WPRkNN
Wine-an-nn	**0.959 ± 0.030**	0.705 ± 0.091	0.702 ± 0.021	0.762 ± 0.068
wdbc-an-nn	0.624 ± 0.040	0.633 ± 0.049	0.592 ± 0.013	**0.672 ± 0.032**
Segment-an-nn	**0.818 ± 0.081**	0.714 ± 0.119	0.718 ± 0.015	0.729 ± 0.054
Page-blocks-an-nn	0.955 ± 0.001	0.952 ± 0.007	0.932 ± 0.009	**0.958 ± 0.007**
Hear h-an-nn	0.803 ± 0.087	**0.825 ± 0.052**	0.760 ± 0.024	0.817 ± 0.095
Win#	2	1	**0**	**2**

The bold values are reflect the higher performance

Table 14 WPRkNN vs. Rival kNNs (Best k's Averaged Accuracy ± SD)

Dataset/kNN	ExNRulekNN	CDkNN	RCKNCN	ENN	WPRkNN
Segment-an-nn	**0.825 ± 0.072 (3)**	0.748 ± 0.091 (14)	0.821 ± 0.117 (15)	0.752 ± 0.011 (1)	0.747 ± 0.076 (1)
Page-blocks-an-nn	0.956 ± 0.006 (1)	0.955 ± 0.004 (1)	0.948 ± 0.290 (4)	0.945 ± 0.005 (1)	**0.960 ± 0.007 (5)**
Hear h-an-nn	0.822 ± 0.062 (2)	**0.842 ± 0.056 (6)**	0.769 ± 0.293 (12)	0.813 ± 0.060 (1)	0.840 ± 0.092 (13)
Iris-an-nn	0.962 ± 0.050 (5)	**0.973 ± 0.034 (3)**	0.937 ± 0.298 (7)	**0.973 ± 0.034 (4)**	0.948 ± 0.048 (1)
sonar-an-nn	**0.862 ± 0.079 (2)**	0.802 ± 0.098 (1)	0.818 ± 0.240 (10)	0.805 ± 0.047 (2)	0.811 ± 0.066 (55)
wdbc-an-nn	0.643 ± 0.045 (4)	0.678 ± 0.027 (9)	**0.919 ± 0.196 (12)**	0.608 ± 0.065 (12)	0.702 ± 0.028 (10)
Win#	2	2	**1**	**1**	**1**

The bold values are reflect the higher performance

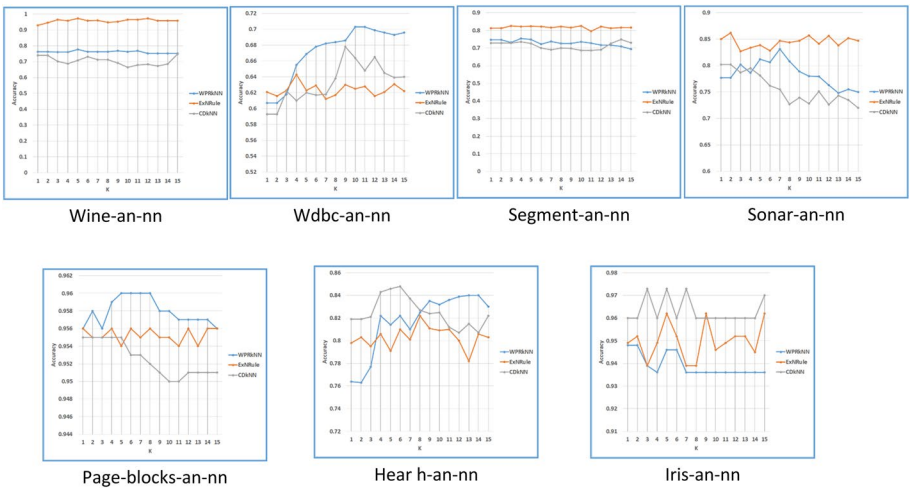

Fig. 14 F1 Results of all kNN models over each k value across each data set

5% by default); and (3) numeric columns are targeted for noise addition, as they are more susceptible to such random variations compared to categorical data.

It is worth indicating that, in experiments 5–6, for the representation coefficient-based kNN (RCKNCN) [42], we just reported its results. To conduct this experiment,

we tested all considered kNNs with k ranging from 1 to 15 with a step of 1 (k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15). Then, the averaged accuracy is reported in Table 13. Further, the best k value's averaged accuracy is reported in Table 14 (see Fig. 14 for details). When results are averaged, both WPRkNN and ExNrulekNN are the top models. The results are supported by the significance tests in our Github repository.

On the other hand, according to Table 14, both ExNrulekNN and CDkNN are the top performers.

Figure 14 shows that our PWRkNN is stable and robust, outperforming its competitors over the individual k values, notably over Wdbc-an-nn, Page-block-an-nn, and Hearh-an-nn, and achieving results that are mostly equivalent over some datasets (Segment-an-nn and Sonar-an-nn). On the other hand, while ExNrulekNN has been the best performer over Segment-an-nn and Sonar-an-nn, CDkNN beats both WPRkNN and ExNrulekNN over Iris-an-nn. In conclusion, there is no clear winner among these kNN models when looking at the final averaged values; all of them produce outcomes that are comparable. However, when looking at the individual k values, it is evident that our WPRkNN has dominated the majority of these values.

Experiment 6: time series datasets

In this final experiment, we further validate the potential competency of the WPRkNN model by comparing it to its strong rivals over six time series datasets taken from the UCR repository [45]. Table 15 draws the dataset's description. To conduct these experiments, we tested all considered kNNs with k ranging from 1 to 25 with a step of 1. Then, the best k value's averaged accuracy is reported in Table 14.

Furthermore, we expanded the comparative analysis across time series to include the local mean representation-based *k*-nearest neighbor approach (LMRKNN) [46]. We just report its results as drawn in [42]. The class-specific nearest neighbors' multi-local mean vectors (MLMKHNN) were employed by [46], which may have resulted in even better performance. The class-specific multi-local mean vectors of the *k* nearest neighbors per class were used to calculate the harmonic mean distances as the classification rule by the multi-local means-based *k*-harmonic nearest neighbor rule (MLMKHNN), which was predicated on the idea that each testing sample should have a uniform value of *k*. We also add ENN into the time series comparison study.

Table 15 The used UCR time series datasets

Dataset	Training Instances	Testing Instances	Classes	Time series lengths
Haptics	155	308	5	1092
Strawberry	370	613	2	235
GunPoint	50	150	2	150
Fish	175	175	7	463
Herring	64	64	2	512
Meat	60	60	3	448

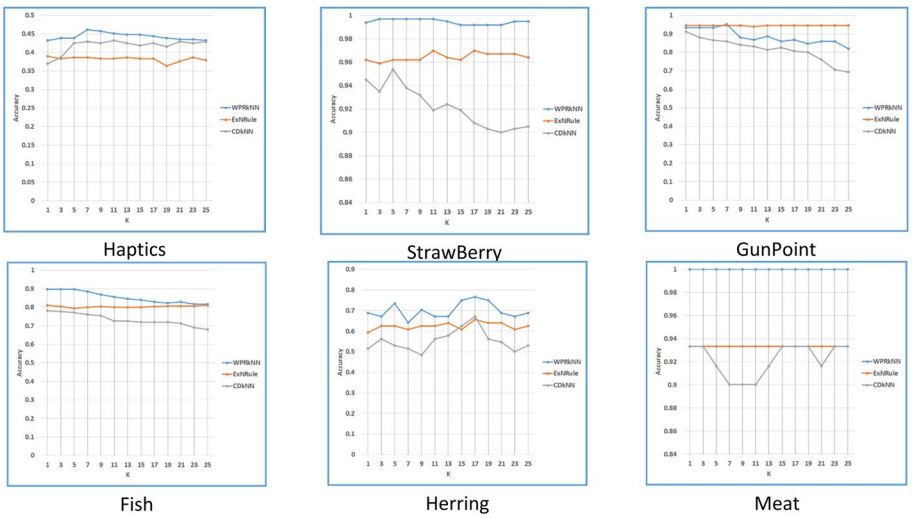

Fig. 15 Accuracy results of all kNN models over each k value across each data set

Table 16 WPRkNN vs. kNNs (Best k's Averaged Accuracy)

Dataset/kNN	LMRKNN	ExNRulekNN	CDkNN	RCKNCN	ENN	WPRkNN
Haptics	0.445 (13)	0.389 (3)	0.438 (10)	**0.477 (15)**	0.467 (6)	0.468 (6)
Strawberry	0.961 (21)	0.970 (2)	0.951 (5)	0.976 (28)	0.945 (1)	**0.997 (3)**
GunPoint	0.933 (3)	0.946 (1)	0.913 (1)	0.960 (9)	0.893 (1)	**0.963 (5)**
Fish	0.863 (5)	0.805 (1)	0.782 (1)	0.874 (9)	0.788 (5)	**0.897 (1)**
Herring	0.641 (2)	0.640 (5)	0.671 (17)	0.688 (6)	0.609 (16)	**0.766 (17)**
Meat	0.933 (1)	0.933 (1)	0.933 (1)	**100.00(11)**	0.950 (1)	**100 (1)**
Win #	0	0	0	2	0	**5**

The bold values are reflect the higher performance

Figure 15 shows that our WPRkNN is stable and robust, dominating its competitors over the individual k values and the averaged accuracy, notably over Haptics, StrawBerry, Fish, Herring, and Meat, and achieving results that are mostly equivalent over some datasets (Segment-an-nn and Sonar-an-nn). On the other hand, ExNrulekNN has been the best performer over GunPoint. In conclusion, our WPRkNN has been clearly superior over time series, either with the final averaged accuracy or at the individual k values. These results are supported by the significance tests in Table 8 in the appendix in our Github repository. To further leverage the information of neighbors, the local mean representation-based *k*-nearest neighbor method (LMRKNN) used the multi-local mean vectors of the class-specific *k* nearest neighbors to linearly represent the testing sample in order to fully explore the discriminative properties of representation learning [47]. Next, the classification rule is derived from the representation residuals between the categorical multi-local mean vectors of neighbors and the testing sample. Both Table 16 and Fig. 15 confirm the superiority of WPRkNN across all k values over each dataset.

Discussion

From the results across all six experiments, we can draw the following remarks: **Firstly**, over the datasets in Table 1 (experiments 1–3), we found that some models, such as MVMCNN and CkNN, have not performed as well as other ones like LMkNN and WLMkNN on UCI datasets in particular. This is due to the fact that the latter ones make use of the local mean concept, and these concept-based classifiers (LMkNN and WLMkNN) outperform CKNN and MVMCNN. In other words, this concept works on the class-based neighborhoods, and in general, the class-based neighborhood (e.g., CDkNN, ExNRulekNN, and PRkNN models) performs better than the general neighborhoods that were used in CkNN and the clustering-based MVMCNN. The sensitivity to irrelevant points (or what can be referred to as noisy points or outliers) has been mitigated in the local mean and class-based models. We also noticed that when we used big k values and took the averaged performance, DkNN, SkNN, CkNN, and MVMCNN were not able to draw good performance as they are only able to compete fiercely with small k values (e.g., 5) as given in Figs. 7, 8. Further, in contrast to CkNN, MkNN, MVMCNN, and SkNN, WPRkNN and local mean-based models are less sensitive to the size of the training sample. On the negative side, MkNN, LMkNN, WLMkNN, WPRkNN, and ExNRulekNN are more costly to compute than CDkNN, CkNN, MVMCNN, and SkNN; therefore, their effectiveness comes with a price.

For the MVMCNN model in particular, the degradation of the MVMCNN performance over the majority of the datasets is due to the clustering mechanism. In technical terms, MVMCNN is more sensitive to some datasets and more computationally complex since it uses the c-means clustering approach, which typically produces unstable clusters. In the testing phase, nevertheless, it is seen as highly efficient. Overall, the clustering-based kNN is efficient, yet at the expense of the algorithm's effectiveness. On the positive side, however, its performance still stands stronger than SkNN and MkNN and is also competitive with CkNN and WLMkNN in a few cases because of the commission procedure used for the voting in this algorithm. Similarly, our experimental analysis using the averaged accuracy demonstrates that CkNN performance has typically been either equivalent to or occasionally worse than SkNN in many cases when run over UCI datasets. Our conclusions are in line with the assertions made by the authors themselves that the CkNN performs well for low values of k.

Secondly, CKNN is computationally expensive due to the repeated distance calculations and sorting for each test point, especially when k is large. It is also sensitive to class imbalance, as the method may favor the majority class, and its performance can be negatively impacted by outliers, which can distort the distance calculations and centroid-based decision-making. Additionally, the complexity increases with the number of features (p), making it less efficient for high-dimensional data. On the other hand, outliers can significantly distort the sphere-based discernibility, making D-kNN sensitive to noise and potentially affecting the accuracy of classification. Finally, D-kNN can face performance issues with large datasets, as the sphere calculations and discerning boundaries require additional computational resources compared to standard k-NN. The modified KNN approach can also be sensitive to noise and outliers, as the changes introduced may amplify the impact of these data points on classification

results. Finally, like standard k-NN, it may face challenges with class imbalance, where the classifier tends to be biased toward the majority class.

Thirdly, the LMKNN and WLMKNN also struggle with class imbalance, as classes with fewer samples may not have enough neighbors to compute reliable local means, leading to less accurate predictions. Additionally, they are sensitive to noise, as outliers within the local neighborhood can significantly affect the computed mean, leading to biased or inaccurate classification results.

Fourthly, it is noted that CDkNN and ExNRulekNN have proven to be competitively successful and effective. The fact that CDkNN computes centroids both before and after adding test data is ultimately what makes it useful. Its lack of utility, especially in situations where the class separation is not statistically significant, is what makes it inferior, in many cases, to WPRkNN in terms of accuracy and ROC. CDkNN still faces challenges related to its sensitivity to outliers and class imbalance, as proven by the results of corresponding experiments. On the other hand, since ExNRulekNN applies two layers of majority voting, making it more resilient, it has been observed to perform well for small datasets and to be competitive with medium- and high-dimensional ones. But this version has the following numerous limitations: ExNRule KNN can be computationally expensive due to bootstrapping and multiple distance calculations for each test point, especially with large datasets. It is also sensitive to noisy data and outliers, which can distort the bootstrapping process and lead to inaccurate predictions. Additionally, the method's performance may degrade with high-dimensional data due to the curse of dimensionality. Finally, it runs the kNN classifier on a subset of data almost 500 times (B = 500 for the base classifiers, for example), illustrating its severe inefficiency, which is supported by the run time displayed in Figs. 11, 12.

Fifthly, experiments 5 and 6 further affirm the competency of WPRkNN model to its rivals on noisy and time-series datasets. As a matter of fact, the competitiveness of WPRkNN, which has been clearly illustrated through the results of all experiments, is mostly attributed to both PR and weighting techniques. To handle overlapping classes, PR is designed to handle cases where classes overlap in the feature space. It flags points as outliers when they are surrounded by neighbors from multiple classes, making it better suited for datasets with class overlap or complex structures. For local consistency evaluation, PR evaluates the local consistency of a point with its neighbors, considering the distribution of classes within the local neighborhood. Most importantly, compared to its rivals, WPRkNN has no parameter to tune, which increases its performance generalization.

Finally, two weighted kNNs that outperform the SMOTE-based weighting technique [48] on ten imbalanced datasets are compared to our WPRkNN. The outcomes for both kNNs are compared to ours in Table 17, demonstrating our model's competence in the vast majority. The neighbors are weighted by the multiplicative-inverse (MI) or additive-inverse (AI) of their distances when using CCW-based kNNs with another weighting approach. It should be noted that in [48], both CCW kNN approaches were regarded as the best models. As a result, we simply state them in Table 17 for comparison purposes, demonstrating our WPRkNN's competency with these useful weighting techniques. The ROC results for both CCW-based kNNs are just reported from [48] and contrasted with our proposed WPRkNN.

Table 17 Performance of WPRkNN vs. weighting strategies [48]—Average ROC Results

kNN/method	#Example	#Feature	Min_Class	Cov_Var	kNN-CCW^{MI}		kNN-CCW^{AI}		Weighted SMOTE-kNN		WPRkNN	
					K = 1	K = 11	K = 1	K = 11	K = 1	K = 11	K = 1	K = 11
Arrhythmia	452	263	2.88%	401.5	0.145	0.214	0.214	0.229	–	0.083	0.749	**0.805**
Balance	625	5	7.84%	444.3	0.063	0.130	0.130	0.149	–	0.135	0.784	**0.859**
Cleveland	303	14	45.54%	2.4	0.831	0.897	0.897	0.897	–	0. 889	0.576	**0.658**
Cmc	1473	10	22.61%	442.1	0.318	0.383	0.383	0.384	–	0.358	0.596	**0.643**
Credit	690	16	44.49%	8.3	0.846	**0.895**	0.895	0.894	–	0.891	0.661	0.684
Ecoli	336	8	5.95%	260.7	0.743	0.948	0.948	0.941	–	0..926	**0.962**	**0.956**
Heart	270	14	44.44%	3.3	0.818	**0.876**	**0.876**	**0.876**	–	0.878	0.685	0.788
Hepatitis	155	20	20.65%	53.4	0.555	0.646	0.569	0.645	–	0. 625	**0.690**	0.476
Pima	768	9	34.9%	70.1	0.587	0.667	0.618	0.665	–	0.657	0.643	**0.685**
Primary	339	18	4.13%	285.3	0.265	0.314	0.224	0.279	–	0.310	0.541	**0.636**
Win#					0	2	1	1	0	0	2	**7**

The bold values are reflect the higher performance

Work's applications

The results demonstrate how highly effective our models have been over the imbalanced and time series datasets. Therefore, there is a lot of promise for using the proposed models in areas where the class distribution of samples is inherently and severely skewed, such as fraud detection, spam and outlier identification, claim prediction, intrusion detection, etc.

Work's strength

Practically all relevant efforts focused on improving data classification from a single perspective (primarily class imbalance), ignoring factors such as class overlaps and/or the k selection value. Alternatively, consider class overlap while ignoring the class imbalance and/or k value selection sensitivity. However, given the complexity of real-world datasets, a comparison study based on a single factor is simplistic. For instance, any SMOTE variation, independent of the degree of IR, might be able to handle an imbalanced dataset that is linearly separable with ease [45, 47]. Therefore, our models' ability, using PR and weighting techniques combined, to reduce the combined detrimental impact of these problems is one of their novelties, as demonstrated by all six experiments. In addition, our models are simpler in design than the leading models under comparison, which have more intricate designs. Further, PR with weighting schemes is carefully designed with the aim of giving the irrelevant points "outliers" the maximum distance from the test point (see example in Fig. 1). Such a procedure prevents these points from making any significant contributions during classification. Not to mention that our PRkNN models have been seen as less sensitive to the k value selection (see Figs. 7, 8, 9, 10). The models maintain stable and growing performance as the k gets increased.

Work's limitations

Despite our models' proven high performance, they have inefficiency issues, even though they are not appreciably slower. However, considering the advantages of the models we offer (addressing imbalance and overlapping problems, enhancing accuracy, and improving generalizability), the computational cost disadvantage might not make sense. Furthermore, the main focus of our research is the reliability of the methods rather than the time frame. Not to mention that recent developments in high-performance computing have made it possible for the proposed models to run in parallel, alleviating this restriction. Another limitation is that despite having comparable results for WPRkNN with its rivals, WPRkNN performance over noisy and extremely balanced datasets is still lower than our expectation. Therefore, we intend to propose a more effective kNN model in the future.

Conclusions and future work

In this work, we proposed three kNN models with simplistic designs yet competitive performance to lessen the class imbalance and overlap problems' negative impact. Firstly, we presented a technique (called proximity ratio, PR) to simply recognize the overlapping points. The PR sought to ascertain the degree to which the classes

in the target dataset can be discriminated from each other and to efficiently identify the relevant neighbors from the non-relevant ones. Secondly, we proposed weighting schemes that were then combined with PR techniques. Both weighting schemes and PR techniques are later integrated with the kNN models. Thirdly, using six evaluation metrics (accuracy, F1, ROC, sensitivity, specificity, and GM) to accentuate the competitiveness of our proposed models versus their competitors (twelve kNNs and five ML models), we have performed in-depth experimental research across 42 datasets in six phases.

The results across all six experiments over fifty-two datasets showed the vivid competency of the proposed models versus their rivals. Furthermore, based on the findings obtained, our PRkNN models exhibit improved classification accuracy as the k grows, and their performance both stayed stable and improved as the k value rose. The WPRkNNs' performance has been promising not just over the imbalanced datasets but also over low- and high-dimensional datasets, regardless of whether they are single or multi-class datasets. Nevertheless, the proposed models have inefficiency concerns, even though they are not appreciably slow. So, it is appealing to achieve high accuracy rates at reasonable computational costs. As such, the future work seeks to find extremely efficient kNNs, including clustering-based model [2, 49], that would effectively address the trade-off between accuracy and run time and be highly superior over noisy and extremely imbalanced datasets.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s40537-025-01137-2.

Additional file 1.

Additional file 2.

Author contributions
Ali A. Amer: Conceptualization, Design, Methodology, Implementation, Results analysis, Writing—original draft, Writing. Sri Devi Ravana: Writing—original draft, Investigation, Experiments Administration. Riyaz Ahamed Ariyaluran Habeeb: Writing—review & editing, Validation.

Funding
No funding was received for conducting this work.

Data availability
The datasets used are publicly available.

Code availability
The code will be uploaded to our Github repository (https://github.com/aliamer/PRkNN-Models) when the paper is accepted.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 14 May 2024 Accepted: 25 March 2025
Published online: 11 April 2025

References

1. Halder RK, Uddin MN, Uddin MA, Aryal S, Khraisat A. Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. J Big Data. 2024;11(1):113.
2. Amer AA, Al-Razgan M, Abdalla HI, Al-Asaly M, Alfakih T, Al-Hammadi M. Neighboring-aware hierarchical clustering: a new algorithm and extensive evaluation. Int J Semant Web Inf Syst (IJSWIS). 2024;20(1):1–24.
3. Abdalla HI, Amer AA. Boolean logic algebra driven similarity measure for text based applications. PeerJ Comput Sci. 2021;7: e641.
4. Ma Y, Huang R, Yan M, Li G, Wang T. Attention-based local mean k-nearest centroid neighbor classifier. Expert Syst Appl. 2022;201: 117159.
5. Liu D, Jiang C, Cao Y. Probabilistic Local Mean K-Nearest Neighbors Classification. In 2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA) (pp. 1118–1123). IEEE; 2024.
6. Mailagaha Kumbure M, Luukka P. A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance. Granular Comput. 2022;7(3):657–71.
7. Mei J, Chen J. Application of KNN algorithm in diabetes prediction. Front Interdiscip Appl Sci. 2024;1(1):8–15.
8. Zeidan A, Vo HT. Efficient spatial data partitioning for distributed k NN joins. J Big Data. 2022;9(1):77.
9. Rezaei F, Abbasitabar M, Mirzaei S, Kamari Direh Z, Ahmadi S, Azizi Z, Danialy D. Improve data classification performance in diagnosing diabetes using the Binary Exchange Market Algorithm. J Big Data. 2022;9(1):43.
10. Uddin S, Haque I, Lu H, Moni MA, Gide E. Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. Sci Rep. 2022;12(1):6256.
11. Wang Y, Pan Z, Dong J. A new two-layer nearest neighbor selection method for kNN classifier. Knowl-Based Syst. 2022;235: 107604.
12. Khandelwal M, Rout RK, Umer S, Sahoo KS, Jhanjhi NZ, Shorfuzzaman M, Masud M. A pattern classification model for vowel data using fuzzy nearest neighbor. Intell Autom Soft Comput. 2023;35(3).
13. Ali A, Hamraz M, Gul N, Khan DM, Aldahmani S, Khan Z. A k nearest neighbour ensemble via extended neighbourhood rule and feature subsets. Pattern Recogn. 2023;142: 109641.
14. Abdalla HI, Amer AA. Towards highly-efficient k-nearest neighbor algorithm for big data classification. In 2022 5th International Conference on Networking, Information Systems and Security: Envisage Intelligent Systems in 5g//6G-based Interconnected Digital Worlds (NISS) (pp. 1–5). IEEE; 2022.
15. Wang J, Zhou Z, Li Z, Du S. A novel fault detection scheme based on mutual k-nearest neighbor method: application on the industrial processes with outliers. Processes. 2022;10(3):497.
16. Nguyen LV, Vo QT, Nguyen TH. Adaptive KNN-based extended collaborative filtering recommendation services. Big Data Cogn Comput. 2023;7(2):106.
17. Ren J, Wang Y, Mao M, Cheung YM. Equalization ensemble for large scale highly imbalanced data classification. Knowl-Based Syst. 2022;242: 108295.
18. Kim K. Normalized class coherence change-based kNN for classification of imbalanced data. Pattern Recogn. 2021;120: 108126.
19. Gweon H, Schonlau M, Steiner SH. The k conditional nearest neighbor algorithm for classification and class probability estimation. PeerJ Comput Sci. 2019;5: e194.
20. Saxena V, Bhardwaj S, Saxena AK. Enhancement of K nearest neighbour approach to solve the issue of pattern classification. In AIP Conference Proceedings (Vol. 2427, No. 1). AIP Publishing; 2023.
21. Yuan BW, Luo XG, Zhang ZL, Yu Y, Huo HW, Johannes T, Zou XD. A novel density-based adaptive k nearest neighbor method for dealing with overlapping problem in imbalanced datasets. Neural Comput Appl. 2021;33:4457–81.
22. Zhang H, Dong Y, Xu D. Accelerating exact nearest neighbor search in high dimensional Euclidean space via block vectors. Int J Intell Syst. 2022;37(2):1697–722.
23. Yang J, Tan X, Rahardja S. Outlier detection: how to select k for k-nearest-neighbors-based outlier detectors. Pattern Recogn Lett. 2023;174:112–7.
24. Shah AA, Ravana SD, Hamid S, Ismail MA. Web credibility assessment: affecting factors and assessment techniques. Information Research. 2015;20(1), paper 655. Retrieved from https://informationr.net/ir/20-1/paper663.html.
25. Moghadasi SI, Ravana SD, Raman SN. Low-cost evaluation techniques for information retrieval systems: a review. J Informet. 2013;7(2):301–12.
26. Voulgaris Z, Magoulas GD. Extensions of the k nearest neighbour methods for classification problems. In Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications, AIA (Vol. 8, pp. 23–28); 2008.
27. Adli NABZ, Ahmad M, Ghani NA, Ravana SD, Norman AA. An ensemble classification of mental health in Malaysia related to the Covid-19 pandemic using social media sentiment analysis. KSII Trans Internet Inf Syst (TIIS). 2024;18(2):370–96.
28. Pourseyyedi M, Forghani Y. Weighted version of extended nearest neighbors. Neural Process Lett. 2019;49:227–37.
29. Mehta S, Shen X, Gou J, Niu D. A new nearest centroid neighbor classifier based on k local means using harmonic mean distance. Information. 2018;9(9):234.
30. Zhao Y, Liu X. A classifier combining local distance mean and centroid for imbalanced datasets. In: International Conference on Communications and Networking in China. Cham: Springer International Publishing; 2020. p. 126–39.
31. Parvin H, Alizadeh H, Minati B. A modification on k-nearest neighbor classifier. Global J Comp Sci Technol. 2010;10(14):37–41.
32. Syaliman KU, Nababan EB, Sitompul OS. Improving the accuracy of k-nearest neighbor using local mean based and distance weight. J Phys Conf Ser. 2018;978:012047.
33. Wang AX, Chukova SS, Nguyen BP. Ensemble k-nearest neighbors based on centroid displacement. Inf Sci. 2023;629:313–23.
34. Zhou F, Gao S, Ni L, Pavlovski M, Dong Q, Obradovic Z, Qian W. Dynamic self-paced sampling ensemble for highly imbalanced and class-overlapped data classification. Data Min Knowl Disc. 2022;36(5):1601–22.

35. Gou J, Qiu W, Yi Z, Shen X, Zhan Y, Ou W. Locality constrained representation-based K-nearest neighbor classification. Knowl-Based Syst. 2019;167:38–52.
36. Gou J, Wang L, Yi Z, Yuan Y, Ou W, Mao Q. Weighted discriminative collaborative competitive representation for robust image classification. Neural Netw. 2020;125:104–20.
37. Mitani Y, Hamamoto Y. A local mean-based nonparametric classifier. Pattern Recogn Lett. 2006;27(10):1151–9.
38. Gou J, Yi Z, Du L, Xiong T. A local mean-based k-nearest centroid neighbor classifier. Comput J. 2012;55:1058–71.
39. Tang B, He H. ENN: extended nearest neighbor method for pattern recognition [research frontier]. IEEE Comput Intell Mag. 2015;10(3):52–60.
40. Abdalla HI, Altaf A, Hamzah AA. A threefold-ensemble k-nearest neighbor algorithm. Int J Comput Appl. 2025;1–14.
41. Suyanto S, Yunanto PE, Wahyuningrum T, Khomsah S. A multi-voter multi-commission nearest neighbor classifier. J King Saud Univ-Comput Inf Sci. 2022;34(8):6292–302.
42. Gou J, Sun L, Du L, Ma H, Xiong T, Ou W, Zhan Y. A representation coefficient-based k-nearest centroid neighbor classifier. Expert Syst Appl. 2022;194: 116529.
43. Alcalá-Fdez J, Sanchez L, Garcia S, del Jesus MJ, Ventura S, Garrell JM, et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput. 2009;13:307–18.
44. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
45. Azhar NA, Pozi MSM, Din AM, Jatowt A. An investigation of smote based methods for imbalanced datasets with data complexity analysis. IEEE Trans Knowl Data Eng. 2022;35(7):6651–72.
46. Gou J, Qiu W, Yi Z, Xu Y, Mao Q, Zhan Y. A local mean representation-based K-nearest neighbor classifier. ACM Trans Intell Syst Technol (TIST). 2019;10(3):1–25.
47. Chen B, Xia S, Chen Z, Wang B, Wang G. RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise. Inf Sci. 2021;553:397–428.
48. Liu W, Chawla S. Class confidence weighted k NN algorithms for imbalanced data sets. In Advances in Knowledge Discovery and Data Mining: 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24–27, 2011, Proceedings, Part II 15 (pp. 345–356). Springer Berlin Heidelberg. 2011.
49. Abdalla HI, Amer AA, Ravana SD. BoW-based neural networks vs. cutting-edge models for single-label text classification. Neural Comput Appl. 2023;35(27):20103–16.

Publisher's Note