Information Sciences 629 (2023) 313–323

Contents lists available at ScienceDirect

Information Sciences

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

Ensemble k-nearest neighbors based on centroid displacement

Alex X. Wang, Stefanka S. Chukova, Binh P. Nguyen [*]

School of Mathematics and Statistics, Victoria University of Wellington, Wellington 6140, New Zealand

A R T I C L E   I N F O

Keywords:
kNN
Classification
Similarity
Nearest neighbors
Centroid displacement

A B S T R A C T

k-nearest neighbors (k-NN) is a well-known classification algorithm that is widely used in different domains. Despite its simplicity, effectiveness and robustness, k-NN is limited by the use of the Euclidean distance as the similarity metric, the arbitrarily selected neighborhood size k, the computational challenge of high-dimensional data, and the use of the simple majority voting rule in class determination. We sought to address the last issue and proposed the Centroid Displacement-based k-NN algorithm, where centroid displacement is used for class determination. This paper presents a simple yet efficient variant of our previous work, named Ensemble Centroid Displacement-based k-NN, which leverages the homogeneity of the nearest neighbors of test instances. Extensive experiments on various real and synthetic datasets were conducted to show the effectiveness and robustness of the proposed algorithm. Our experimental results demonstrate that the proposed algorithm is able to enhance the classification performance of the standard k-NN algorithm and its variants and also improve the computational efficiency. The performance of our algorithm was consistent and robust for both balanced and imbalanced datasets.

1. Introduction

The k-nearest neighbors (k-NN) algorithm is a supervised machine learning algorithm (ML) widely used for pattern recognition in many different disciplines, such as face recognition, text classification, graph analysis, disease prediction, anomaly detection, time series forecast, among others [1–4]. During model training, instead of estimating the hypothetical function once for the entire training data space, k-NN just stores all the training samples. Therefore, it is sometimes called a “lazy learning method” because it postpones learning until the new instance is encountered for prediction [5]. In classification, the concept and mechanism of k-NN are simple. Given an unlabeled test instance, k-NN first finds the k number of nearest neighbors in the training dataset based on a predefined distance metric and then assigns the most common class label among the selected neighbors.

k-NN has several advantages, including conceptual simplicity, a substantial theoretical foundation, strong generalization performance and no assumptions on data distribution [6–8]. It is also adaptive and flexible for integration, where numerous algorithms built on k-NN have been proposed for different ML problems, such as minority data oversampling, multi-label prediction and clustering problems [9–11]. However, despite its simplicity, effectiveness, robustness and flexibility, k-NN is reported to have several limitations, ranging from the use of the Euclidean distance as the similarity metric, the arbitrarily selected neighborhood size k, the computational challenge of high-dimensional data, and the use of a simple majority voting rule (i.e., selecting the mode of the classes from the nearest neighbors) in the class probability estimation [12]. To address these limitations, a large number of k-NN variants have been proposed. A comprehensive review can be found in [13]. Among them, we focused on the issue of using the simple

* Corresponding author.
E-mail address: binh.p.nguyen@vuw.ac.nz (B.P. Nguyen).

https://doi.org/10.1016/j.ins.2023.02.004
Received 25 July 2022; Received in revised form 24 January 2023; Accepted 1 February 2023

Available online 6 February 2023

majority voting rule in class probability estimation and proposed the Centroid Displacement-based k-NN algorithm (CDNN) [14], where centroid displacement is used for class determination. It is demonstrated that CDNN is adaptive to noise and complex class distributions, allowing the correct class label to be assigned to the test instance without extensive tuning parameter k for the dataset.

This paper introduces Ensemble Centroid Displacement-based k-NN (ECDNN), a simple and efficient variant of the CDNN algorithm that exploits the homogeneity of the nearest neighbors for each test instance. This is, to the best of our knowledge, the first ensemble k-NN algorithm that incorporates the data-centric Artificial Intelligence (AI) philosophy [15], where both the algorithm and data are integral to the success of applications. We use the standard k-NN algorithm to predict the class of a test instance when we have high confidence in that prediction (i.e., when its nearest neighbors are homogeneous and likely belong to the same class); otherwise, we use CDNN, a more sophisticated algorithm, to make prediction (when the intra-class distribution of the nearest neighbors is overlapping and the in-between distances are small). In this way, the proposed algorithm seeks to identify a conditional feature space based on both the distance metric and the homogeneity of the test instance's nearest neighbors. By combining the advantages of both underlying algorithms, the proposed ensemble method is anticipated to improve its computational efficiency and discriminative capability for classification.

We evaluate the performance of ECDNN by using both artificial and real-life datasets for both binary and multi-class classification problems. The experimental results demonstrate the computational efficiency and prediction effectiveness of the proposed ECDNN, compared with other k-NN based algorithms. To promote future applications of our work, we have implemented CDNN and ECDNN in the `scikit-learn` framework [16], a well-known ML library for the Python programming language, and made the code of our algorithms publicly available. `SciPy` and `NumPy`, two well-established Python libraries, are utilized for all internal operations for compatibility and improved computational efficiency. Overall, this paper has the following contributions:

- The proposed ECDNN algorithm is intuitive and simple to execute and explain the prediction.
- A `scikit-learn` compatible Python library [17] containing both CDNN and ECDNN has been developed and is accessible at https://github.com/coksvictoria/ECDNN.
- Extensive experimental results on various datasets demonstrate the computational efficiency and predictive accuracy of ECDNN.
- This study provides direction for the development of similar algorithms where the homogeneity of nearest neighbors can be used as an information channel for ensemble learning and where different algorithms can be applied based on the data characteristics.

2. Related work

2.1. k-NN

The classic k-NN algorithm, proposed by Fix and Hodges [18], is one of the most fundamental and simple algorithms. It is a non-parametric method with local approximation that can be used for both classification and regression. In classification, the problem can be defined as follows: we are given a set of points $P = \{(x_i, c_i) : x_i \in \mathbb{R}^d, c_i \in C\}$, where C is a set of class labels (for instance, a binary class labeling would be: $C = \{0, 1\}$) and x_i is a d-dimensional vector. The task at hand is, given a point $(q, c) \notin P$, and where c is unknown, attempt to assign a class label from C to the point based on the majority class of the k closest neighbors to q in P. The nearest neighbors are selected based on a distance metric that measures the difference or similarity between two instances. The default Euclidean distance of k-NN for samples x_i (i =1,2,. . . ,n) and x is defined as:

$$d(\mathrm{x}, \mathrm{x}_i) = \sqrt{(\mathrm{x} - \mathrm{x}_i)^T(\mathrm{x} - \mathrm{x}_i)}. \tag{1}$$

Given an instance x, k-NN assigns the most common class of x's k nearest neighbors to it, as shown in Equation (2) [19].

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)), \tag{2}$$

where $y_1, y_2, ..., y_i$ are the k nearest neighbors of x and $\delta(c, c(y_i)) = 1$ if $c = c(y_i)$ and $\delta(c, c(y_i)) = 0$ otherwise. In this way, k-NN can also be used for regression. In this case, the new instance is given a label that is the average of the labels of its k nearest neighbors.

Despite its simplicity, efficacy, robustness and applicability, k-NN is reported to have a number of drawbacks [12]. First, as a distance-based algorithm, the core of k-NN is dependent on the distance or similarity metrics used to quantify the distances between the test data instance and instances in the training dataset [8]. Previous research has analyzed the performance of k-NN with various distance metrics, given that the default Euclidean distance has been reported to produce suboptimal results [20]. Second, a successful application of k-NN requires, in addition to a well-selected distance metric, a careful selection of the number of nearest neighbors k [21]. Meta-heuristic algorithms, such as Particle Swarm Optimization, have been proposed as an effective method for determining k [22]. Third, as a member of the lazy learning family, k-NN faces the computational challenges associated with large datasets [4]. Instead of learning a discriminative function from the training data, the k-NN algorithm memorizes the entire training dataset. Whenever a new data point arrives during prediction, k-NN searches for the nearest neighbors in the entire training set, which is computationally expensive for moderate datasets and impractical for large datasets. Lastly, the majority vote system disregards data proximity, which is problematic, especially when the distance between the test sample and its nearest neighbors varies substantially [15].

Algorithm 1 The CDNN algorithm.

```
Require: Training set $T = \{(x_1, c_1), ..., (x_i, c_i)\}$; Test instance x; Number of nearest neighbors $k$.
 1: function CDNN(x)
 2:     for all $(x_i, c_i) \in T$ do
 3:         $d(x, x_i) = \sqrt{(x - x_i)^T (x - x_i)}$
 4:     end for
 5:     $D_x^k \leftarrow$ List of $k$ nearest neighbors to x based on $d(x, x_i)$
 6:     $S_x^j \leftarrow \{(x_t, c_t) \in D_x^k | c_t = c^j, c^j \in C\}$      ▷ Split $D_x^k$ into disjointed sets of instances having the same class label
 7:     $p_x^j \leftarrow \frac{\sum_{\forall x_t \in S_x^j} x_t}{|S_x^j|}$      ▷ Centroid of $S_x^j$
 8:     $q_x^j \leftarrow \frac{\left(\sum_{\forall x_t \in S_x^j} x_t\right) + x}{|S_x^j| + 1}$      ▷ New centroid if x is inserted into $S_x^j$
 9:     $\delta_x^j \leftarrow \sqrt{(p_x^j - q_x^j)^T (p_x^j - q_x^j)}$      ▷ Centroid displacement
10:     $c(x) \leftarrow c^{\arg\min_j \delta_x^j}$      ▷ Predicted class label of x
        return $c(x)$
11: end function
```

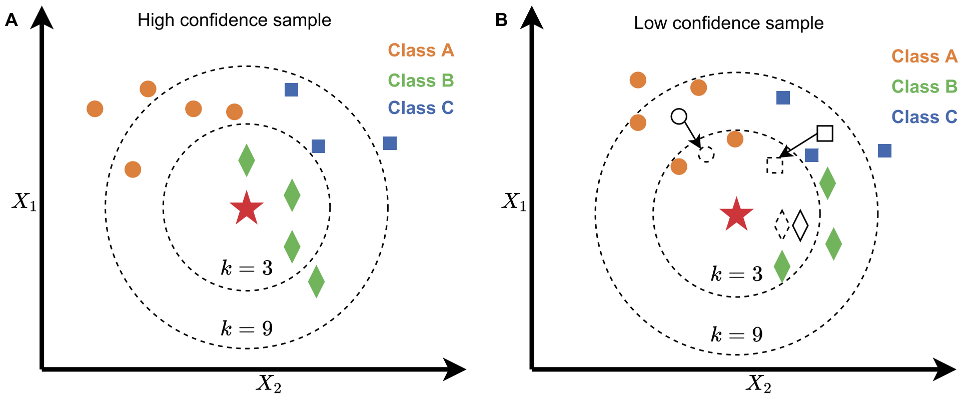

Fig. 1. Visual illustration of the proposed algorithm.

2.2. CDNN

Inspired by the k-means algorithm, which aims to divide M points in N dimensions into k clusters such that the within-cluster distance is minimized [23], we proposed the Centroid Displacement-based k-NN (CDNN) algorithm (Algorithm 1), which uses centroid displacement for class determination [14]. The traditional k-NN algorithm only considers feature similarity based on distance metrics and disregards instance labeling information. In instances with a high degree of similarity in irrelevant characteristics, the decision will be biased [24]. Our proposed CDNN algorithm mitigates this issue by taking into account the average distance group by different classes.

In the example depicted in Fig. 1B, the test instance will be labeled as class A with $k = 3$ and $k = 9$ according to the classic k-NN algorithm. However, under CDNN, the test instance is assigned to class B, as class B has the smallest centroid displacement. This conclusion is likely accurate given the proximity of the test instance to its "diamond" neighbors. Given that the CDNN algorithm computes the centroid displacement iteratively for all classes, it appears as though the unlabeled instance is inserted into the class during the prediction process. It is computationally intensive, particularly for high-dimensional datasets [25].

3. The proposed ECDNN algorithm

In the traditional k-NN algorithm, the value of k is unaffected by the positions of the test instances. The bias and variance of the prediction vary across locations; consequently, the optimal k that achieves the best bias-variance trade-off is also dependent on the feature vector. If the same k is used for all test samples, the selected k may not be optimal for all situations [21,26]. In k-NN, the determination of an optimal k at the global level for improved prediction is a significant challenge. Several ensemble algorithms, such as Ensemble approach k-NN (EA-KNN) [27], which employs the classic k-NN algorithm with multiple sets of neighbors, have attempted to resolve this issue exhaustively. From $k = 1$ to $k = \lfloor \sqrt{n} \rfloor$, where n is the size of the training data set, the ensemble algorithm uses majority rule to identify the class, i.e., the class with the most votes (by 1-NN, 3-NN, 5-NN, … $\lfloor \sqrt{n} \rfloor$-NN) is selected. Other k-NN variants, such as adaptive k-NN algorithms, attempted to determine a locally optimal k based on the data distribution of each test instance. Despite the fact that the value of k is chosen dynamically based on the location of test instances, the algorithm is the same for each test instance, and the calculation for determining the optimal value of k at the local level is computationally intensive.

We intend to leverage local information from a data-centric perspective, using various algorithms based on the homogeneity of test instances' nearest neighbors. Most ML technologies provide a relative qualification of the prediction's confidence, such as a percentage or probability, i.e. a number between 0 and 1. In this sense, confidence can be thought of as an estimate of the distance between the test and training instances. The confidence level is high when the labels of the nearest neighbors are highly homogeneous. A simple model, such as k-NN, can be used for high confidence predictions, whereas a more complex model, such as CDNN, is required for low confidence predictions.

The intuition behind ECDNN, alongside with the definition of sample confidence, is presented in Fig. 1. The high confidence sample is shown in Fig. 1A, where class B is well clustered around the new instance. The low confidence sample is shown in Fig. 1B, where the separations between classes A, B and C are not significant. As k is 3 for the high confidence sample query, it searches for the 3 nearest neighbors and finds that all 3 nearest neighbors are of class B. Then if k is 9 for the high confidence sample, by using the majority voting rule, it is still to be classified as class B. On the other hand, for the low confidence sample, it is classified as class A by k-NN when k is 3 or 9. However, we can clearly see that the sample is closer to class B, which we can use CDNN as explained above. Therefore, we hypothesize that if we can apply k-NN or CDNN accordingly to the test instance, the accuracy will be increased. Moreover, it will be more computationally efficient given that the high confidence samples are predicted based on majority voting (using k-NN), instead of centroid displacement, which requires extra two steps of calculation.

Algorithm 2 The ECDNN algorithm.

```
Require: Training set $\mathrm{T} = \{(\mathrm{x}_1, c_1), ..., (\mathrm{x}_i, c_i)\}$; Test instance x; Number of nearest neighbors $k$; Number of nearest neighbors for homogeneity $n$; function card($S$) to return the number of elements in set $S$.
 1: function ECDNN(x)
 2:     for all $(\mathrm{x}_i, c_i) \in \mathrm{T}$ do
 3:         $d(\mathrm{x}, \mathrm{x}_i) = \sqrt{(\mathrm{x} - \mathrm{x}_i)^T (\mathrm{x} - \mathrm{x}_i)}$
 4:     end for
 5:     $\mathrm{D}^{\mathrm{n}}_{\mathrm{x}} \leftarrow$ List of $n$ nearest neighbors to x based on $d(\mathrm{x}, \mathrm{x}_i)$
 6:     $\mathrm{S}^i_{\mathrm{x}} \leftarrow \{(\mathrm{x}_t, c_t) \in \mathrm{D}^n_{\mathrm{x}} | c_t = c^i, c^i \in \mathrm{C}\}$
 7:     if card($S^i_x$) = 1 then
 8:         $c(x) \leftarrow c^i$
 9:     else
10:         $\mathrm{D}^{\mathrm{k}}_{\mathrm{x}} \leftarrow$ List of $k$ nearest neighbors to x based on $d(\mathrm{x}, \mathrm{x}_i)$
11:         $\mathrm{S}^j_{\mathrm{x}} \leftarrow \{(\mathrm{x}_t, c_t) \in \mathrm{D}^k_{\mathrm{x}} | c_t = c^j, c^j \in \mathrm{C}\}$
12:         $\mathrm{p}^j_{\mathrm{x}} \leftarrow \frac{\sum_{\forall \mathrm{x}_t \in \mathrm{S}^j_{\mathrm{x}}} \mathrm{x}_t}{|\mathrm{S}^j_{\mathrm{x}}|}$                    ▷ Centroid of $\mathrm{S}^j_{\mathrm{x}}$
13:         $\mathrm{q}^j_{\mathrm{x}} \leftarrow \frac{\left(\sum_{\forall \mathrm{x}_t \in \mathrm{S}^j_{\mathrm{x}}} \mathrm{x}_t\right) + \mathrm{x}}{|\mathrm{S}^j_{\mathrm{x}}| + 1}$                    ▷ New centroid if x is inserted into $\mathrm{S}^j_{\mathrm{x}}$
14:         $\delta^j_{\mathrm{x}} \leftarrow \sqrt{(\mathrm{p}^j_{\mathrm{x}} - \mathrm{q}^j_{\mathrm{x}})^T (\mathrm{p}^j_{\mathrm{x}} - \mathrm{q}^j_{\mathrm{x}})}$                    ▷ Centroid displacement
15:         $c(x) \leftarrow c^{\arg\min_j \delta^j_{\mathrm{x}}}$
16:     end if
        return $c(x)$                    ▷ Predicted class label of x
17: end function
```

To demonstrate our hypothesis of using the data-centric approach for building ensemble models in a controlled experiment, we created a synthetic binary dataset that consists of three groups of data, each with different degrees of complexity. We split each group of data into train and test datasets. We train multiple k-NN based models, including CDNN, Radius NN, k-NN and Weighted k-NN (WKNN), on each training partition and report the accuracy of prediction on the test dataset. To visualize the result, we first project our data into two dimensions using the principal component analysis (PCA) due to its simplicity and popularity [28]. The contour plots in Fig. 2 show the transformed PCA components on the x and y axes and probability z (of the data points being predicted as class blue as the contours). The darker the shade of blue indicates a higher probability of being in class blue while the lighter the shade of blue indicates a lower probability of being in class blue (so a higher probability of being in class red given this is a binary classification problem). The rest of the interpretation will follow the same color schema. Every intersection line between colors in the contour plot represents one cut along the vertical axis. That means the probability is the same along the line, where the left-side area has darker blue (a higher probability of being in class blue) than the right-side area (a lower probability of being in class blue). If one looks at the contour present at the extreme left (and right) of the plot, there are no other boundaries or contour surfaces, which means dots from this area have a high probability of being predicted as class blue (and class red, respectively). The distance between the extreme left side and the next area is very large, which means the gradient of the probability in this area is very flat, so one can predict samples in the area with high confidence.

As shown in the first column of Fig. 2, the complexity of the data gradually increases from row 1 to row 3. Classes in data group 1 are well-separated, so k-NN and WKNN perform better than other algorithms. Data group 2 shows moderate overlap between the two classes, with CDNN and k-NN outperforming others. Data in data group 3 exhibit substantial overlap, where CDNN outperforms all others. Given the foregoing, the best solution would be to use k-NN and WKNN for data group 1, CDNN and k-NN for data group

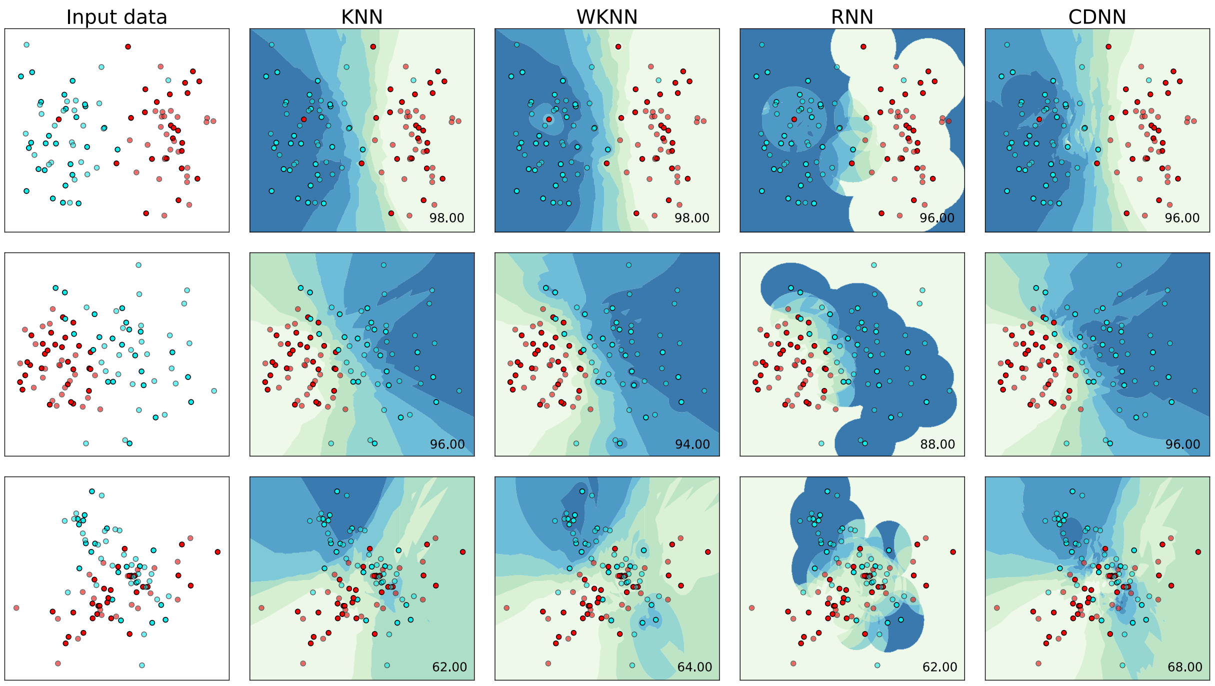

Fig. 2. Visual illustration of the proposed algorithm compared with other k-NN variants based on a simulated binary dataset with two classes: class blue and class red. Training and testing points are depicted in solid and translucent colors, correspondingly. The lower right of each subplot shows the classification accuracy on the test set of the algorithm.

2, and CDNN for data group 3. Therefore, compared to each model and the voting ensembles of them, ECDNN would be the best choice.

4. Experiments

4.1. Dataset

To verify the effectiveness of the processed ECDNN, we conducted extensive experiments on 22 datasets to compare the algorithm with classic k-NN, CDNN and all k-NN variants available in the `scikit-learn` library [29]. Our experiments involve 16 real-world datasets and 6 synthetic datasets. The 16 real-world datasets were collected from the UCI Machine Learning Repository, where 6 have minor imbalance issues and 10 have severe imbalance issues [30], while the synthetic datasets were generated using the `make_classification` function of the `scikit-learn` package. All datasets were selected or generated carefully to cover a variety of data complexity in terms of sample size, dimensionality, imbalance ratio (IR), noise and sparsity. The detailed information about these real and synthetic datasets is listed in Table 1. We also present the data complexity for each dataset by following the previous study [31]. Specifically, we report the average number of features per sample, the average number of PCA components per sample, the ratio of the original feature size to the PCA component size and data sparsity, where a higher value indicates greater dataset complexity. For each dataset, $\#C$ is the number of classes, $\#S$ is the number of samples, $\#F$ is the number of features, $\#F_{PCA}$ is the minimum number of principal components necessary to retain 95% of the variance, and Sparsity refers to the proportion of zero-valued data points. We also report IR for all binary datasets.

4.2. Competing algorithms

In order to validate the performance of the new algorithm, we compared the classification results of ECDNN and CDNN to those of four k-NN based classification algorithms available in the `scikit-learn` library. WKNN is a simple and robust extension of the k-NN method that leverages the distance information between neighbors [32]. For classification, as described previously, a majority voting scheme is applied to the k nearest data points, whereas in regression, the mean of the k nearest data points is calculated as the output. Radius NN is similar to the classic k-NN, but instead of finding the k nearest neighbors purely based on the ranking, it finds all the neighbors within a given radius r [33]. As such, the radius-based approach to selecting neighbors is more appropriate for sparse data, preventing examples that are far away in the feature space from contributing to a prediction [34]. However, setting the radius requires some domain knowledge. The Nearest Centroid (NC) neighbors algorithm was proposed to remedy the issue of arbitrary k, which considers the distribution of training instances in the neighborhood of the query [12]. It works on a simple principle: given a data point (observation), NC simply assigns it to the class of the training samples whose local mean or centroid is closest to it [35].

Table 1
Overview of the datasets used in this study.

Name	Source	$\#C$	IR	$\#S$	$\#F$	$\#F/\#S$	$\#F_{PCA}/\#S$	$\#F/\#F_{PCA}$	Sparsity
iris	UCI	3	-	150	4	0.0267	0.0133	2.00	0.0000
wine	UCI	3	-	178	13	0.0356	0.0223	1.60	0.4971
breastcancer	UCI	2	1.7:1	569	30	0.0527	0.0176	3.00	0.0047
digits	UCI	10	-	1797	64	0.0730	0.0562	1.30	0.0000
olivetti	UCI	40	-	4096	400	10.2400	0.3075	33.30	0.0000
paris	UCI	2	1:1	5828	2200	2.6491	0.1314	20.17	0.0001
S1	Synthetic	2	1:1	100	2	0.0200	0.0200	1.00	0.0000
S2	Synthetic	2	1:1	1000	2	0.0020	0.002	1.00	0.0000
S3	Synthetic	3	-	1000	10	0.0100	0.0090	1.11	0.0000
S4	Synthetic	3	-	1000	50	0.0500	0.0420	1.19	0.0000
S5	Synthetic	10	-	5000	50	0.0100	0.0092	1.09	0.0000
S6	Synthetic	10	-	5000	100	0.0200	0.0182	1.10	0.0000
ecoli	UCI	2	8.6:1	336	7	0.0208	0.0179	1.17	0.0020
optical_digits	UCI	2	9.1:1	5620	64	0.0114	0.0075	1.52	0.4961
satimage	UCI	2	9.3:1	6435	36	0.0056	0.0009	6.00	0.0000
pen_digits	UCI	2	9.4:1	10992	16	0.0015	0.0009	1.60	0.1369
abalone	UCI	2	9.7:1	4177	10	0.0024	0.0010	2.50	0.2223
sick_euthyroid	UCI	2	9.8:1	3163	42	0.0133	0.0057	2.33	0.4467
spectrometer	UCI	2	11:1	531	93	0.1751	0.0075	23.25	0.0000
car_eval_34	UCI	2	12:1	1728	21	0.0122	0.0087	1.40	0.7500
isolet	UCI	2	12:1	7797	617	0.0791	0.0260	3.04	0.0036
us_crime	UCI	2	12:1	1994	100	0.0502	0.0176	2.86	0.0560

4.3. Experimental settings and evaluation

To obtain a comprehensive and concrete result, we conducted the experiments in an exhaustive approach, where we used all k from 5 to 25, incremental by 2 for all k-NN-based queries [36]. For Radius NN and NC, we just used the default hyper-parameters for simplicity. For ECDNN, in addition to the hyper-parameter k, we also need to define the homogeneity vote hyper-parameter n. According to Cover and Hart, the 1-NN classifier risk is at most twice the Bayes risk [37]. As n increases, the risk can decrease, while the risk converges to the optimal Bayesian risk as the number of training samples approaches infinity. In reality, one always has access to a finite number of training instances, so n should be set larger than 1. Moreover, the classical 1NN method only provides a binary output for each prediction without a probability, which is essential for measuring prediction uncertainty [38]. It is also suggested by the previous study that 1-NN is highly likely to lead to over-fitting issues especially for the binary classification problem [8]. With all of the above, we selected $n = 2$ as the default hyper-parameter for our ECDNN algorithm. Although the main focus of this paper is k-NN based algorithms, we also conducted a secondary experiment to compare ECDNN with some widely-used classification algorithms, including Decision Tree (DT), Gaussian Naive Bayes (GNB), Logistic Regression (LR), Random Forest (RF) and Support Vector Machine (SVM). We expect that the outcome of this experiment will give us a better idea of how well ECDNN works.

The primary performance metric used in this study is the F1 score, given its popularity and robustness in classification performance evaluation. F1 is one of the most popular evaluation metrics, especially for imbalanced datasets [39,40]. It is a harmonic mean of precision and recall. All of these metrics are derived from the confusion matrix, which is a table often used to describe the performance of a classifier [41]. For multi-class classification, the F1 score is computed first for each class, then the results are averaged to get the final performance measure [42]. A comparative study of performance metrics in classification can be found in [43].

All our experiments were repeated ten times, with each using 5-fold cross validation, so a total of 50 results were collected and then averaged. We include both the prediction performance in terms of the F1 score and computational efficiency in terms of seconds when presenting the average results. To determine whether the proposed algorithm's performance improvements are statistically significant, we use the 5x2cv paired t-test [44] to compare the proposed ECDNN's performance with that of k-NN over different k from 5 to 25, incrementally by 2.

5. Results and discussion

In this section, we first present the experimental results, where ECDNN is compared with k-NN, WKNN, Radius NN, NC and CDNN across 22 datasets. We also report computational complexity in terms of seconds, mainly to show the improvement of ECDNN over CDNN. The average F1 score of each algorithm on 22 datasets is included in Table 2. The rankings of each algorithm are also listed in the table. As shown in Table 2, based on the average results on the 22 datasets, the proposed ECDNN algorithm achieves the highest average F1 score and ranks at the top. This result confirms that the proposed ECDNN algorithm is able to improve the classification performance for its building blocks. Moreover, ECDNN outperforms all other competing algorithms on 11 datasets where they are all high-dimensional and the majority of them have severe imbalance problems. This result shows that the proposed ECDNN algorithm is robust and stable for real-world datasets where high-dimensionality and class imbalance are common.

Table 2
Algorithm ranking based on average F1 score of the proposed ECDNN classifier compared with k-NN, WKNN, Radius NN, NC and CDNN on real-world and synthetic datasets. For each dataset, we highlight the algorithm with the best performance.

dataset	k-NN		WKNN		Radius NN		NC		CDNN		ECDNN	
	F1	rank	F1	rank	F1	rank	F1	rank	F1	rank	F1	rank
iris	0.9513	4	0.9586	2	0.9406	5	0.8574	6	**0.9599**	1	0.9518	3
wine	0.9681	3	0.9665	4	0.4705	6	0.9744	1	0.9721	2	0.9664	5
breastcancer	0.9571	4	0.9609	3	0.6193	6	0.9246	5	0.9626	2	**0.9659**	1
digits	0.9637	4	0.9669	3	0.1815	6	0.8889	5	0.9704	2	**0.9763**	1
olivetti	0.6914	5	0.7089	4	0.0527	6	0.8775	1	0.8222	3	0.8771	2
paris	0.5946	4	0.5881	5	0.5977	3	0.5181	6	0.5993	2	**0.6049**	1
S1	0.8844	5	0.9019	3	**0.9095**	1	0.8243	6	0.9009	4	0.9074	2
S2	**0.8992**	1	0.8943	2	0.8781	5	0.8408	6	0.8936	3	0.8899	4
S3	0.979	4	0.9798	3	0.5319	6	0.8736	5	0.9814	2	**0.9837**	1
S4	0.9642	4	0.9659	2	0.4655	6	0.7729	5	**0.9686**	1	0.9655	3
S5	0.9026	4	0.9095	3	0.1802	6	0.5837	5	**0.9181**	1	0.9140	2
S6	0.9075	4	0.9154	3	0.1802	6	0.5644	5	**0.9252**	1	0.9177	2
Average	0.8886	4	0.8931	3	0.5006	6	0.7917	5	0.9062	2	**0.9101**	1
ecoli	0.7892	2	**0.7974**	1	0.7754	5	0.6926	6	0.7881	3	0.7816	4
optical_digits	0.9769	4	0.9773	3	0.4741	6	0.7652	5	0.9791	2	**0.9823**	1
satimage	0.8127	4	0.8149	3	0.6901	5	0.4727	6	0.8222	2	**0.8340**	1
pen_digits	0.9933	4	0.9939	3	0.9726	5	0.5860	6	0.9949	2	**0.9964**	1
abalone	0.5304	4	0.5337	3	0.4754	6	**0.5842**	1	0.5271	5	0.5497	2
sick_euthyroid	0.7134	4	0.7451	3	0.6123	5	0.5332	6	0.7583	2	**0.7718**	1
spectrometer	0.8524	4	0.859	3	0.4804	6	0.6304	5	0.8692	2	**0.8727**	1
car_eval_34	0.7099	5	0.7274	3	0.4798	6	0.7251	4	0.7547	2	**0.7640**	1
isolet	0.9140	3	0.9140	3	0.4800	6	0.6018	5	0.9156	2	**0.9173**	1
us_crime	0.6241	5	0.6302	4	0.4804	6	**0.6862**	1	0.6428	3	0.6548	2
Average	0.7916	4	0.7993	3	0.5920	6	0.6277	5	0.8052	2	**0.8125**	1

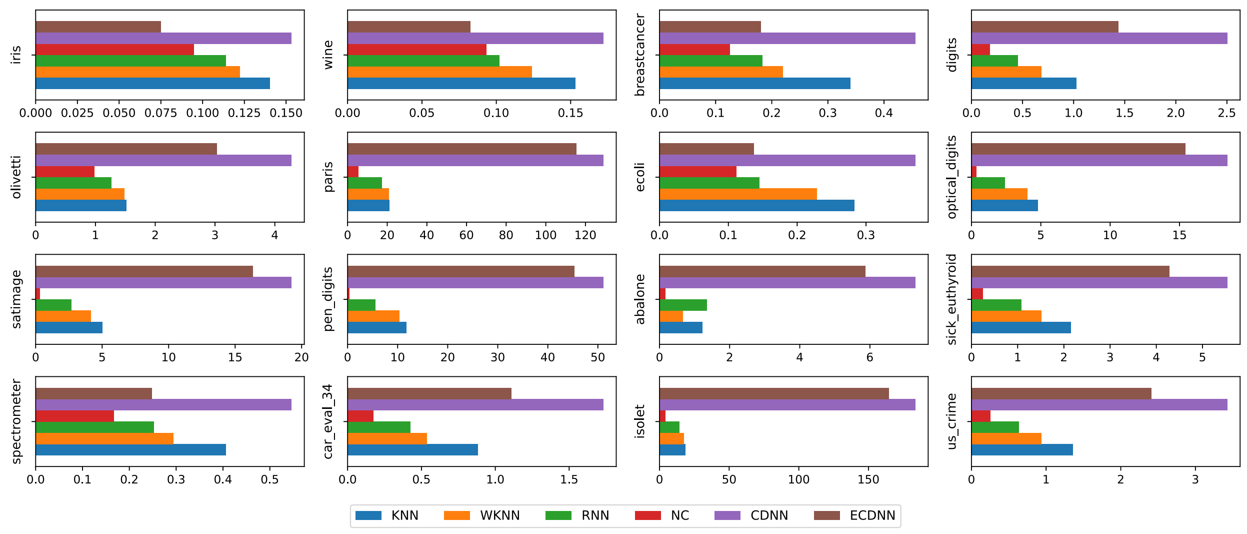

Fig. 3. Average execution time of the proposed ECDNN classifier compared with k-NN, WKNN, Radius NN, NC and CDNN on all datasets. All results are shown in seconds.

In addition to the classification performance, we also evaluate all algorithms on the running times. The average time of each algorithm on 16 real-world datasets is presented in Fig. 3. Based on the results, we can rank the running time of these algorithms in ascending order as follows: $NC < RNN < WKNN < KNN < ECDNN < CDNN$. A complex algorithm such as CDNN requires an iterative calculation of the distances between the test instance and each training sample, while a simple algorithm such as NC is solely based on the distances between the test instance and the centroid of each class. This conclusion depends on the specific dataset. One can see that ECDNN, despite being an ensemble algorithm, is significantly faster than one of its building blocks, CDNN, for all datasets. For simple datasets (such as `iris` and `wine`), where the numbers of data samples and features are small, ECDNN is even faster than all other algorithms. This demonstrates the computational superiority of the data-centric ensemble algorithm, where instead of relying on aggregations of the individual prediction models, it applies different algorithms for different test instances.

A similar pattern has been observed in the second experiment, where ECDNN is compared with some widely-used ML algorithms that are not k-NN based. The result is presented in Table 3, where ECDNN significantly outperforms other algorithms regarding the

Table 3
ECDNN compared with DT, GNB, LR, RF and SVM on real-world and synthetic datasets. For each dataset, we highlight the algorithm with the best performance.

	F1						Time (seconds)	
	DT	GNB	LR	RF	SVM	ECDNN	SVM	ECDNN
iris	0.9442	0.9525	0.9580	0.9527	**0.9617**	0.9518	0.18	0.07
wine	0.9048	0.9740	0.9811	**0.9812**	0.9809	0.9664	0.25	0.08
breastcancer	0.9195	0.9271	**0.9749**	0.9561	0.9739	0.9659	0.81	0.18
digits	0.8507	0.7806	0.9680	0.9750	**0.9810**	0.9763	12.18	1.44
olivetti	0.5135	0.8150	**0.9605**	0.9033	0.9094	0.8771	40.00	3.03
paris	0.5363	0.5260	0.5486	0.6364	**0.6476**	0.6049	1,832.72	115.37
S1	0.8708	0.8221	0.8363	0.8898	0.8531	**0.9074**	0.15	0.09
S2	0.8461	0.8366	0.8436	0.8799	0.8873	**0.8899**	1.60	0.37
S3	0.8595	0.8999	0.9514	0.9663	0.9832	**0.9837**	2.02	0.35
S4	0.5602	0.7744	0.8628	0.8751	**0.9785**	0.9655	7.56	0.53
S5	0.2734	0.6025	0.5981	0.7447	**0.9442**	0.9140	185.37	11.67
S6	0.1982	0.5809	0.5710	0.5870	**0.9306**	0.9177	326.89	19.37
ecoli	0.7554	0.6299	0.7419	0.7692	0.7813	**0.7816**	0.31	0.14
optical_digits	0.8942	0.2398	0.9116	0.9358	0.9779	**0.9823**	27.08	15.42
satimage	0.7446	0.6894	0.4913	0.8020	0.7596	**0.8340**	61.95	16.34
pen_digits	0.9696	0.7982	0.8862	0.9890	0.9955	**0.9964**	26.98	45.25
abalone	**0.5892**	0.5887	0.4803	0.5404	0.4754	0.5497	34.23	5.87
sick_euthyroid	0.9041	0.3370	0.8258	**0.9256**	0.5730	0.7718	16.19	4.28
spectrometer	0.8526	0.6749	**0.9190**	0.8991	0.9135	0.8727	0.65	0.25
car_eval_34	**0.9506**	0.7906	0.9488	0.9419	0.9256	0.7640	4.03	1.11
isolet	0.7869	0.7290	0.9080	0.8465	**0.9520**	0.9173	333.27	164.62
us_crime	0.6688	0.6872	**0.7421**	0.7161	0.6704	0.6548	9.95	2.41
Average	0.7451	0.7116	0.8140	0.8506	**0.8662**	0.8657	132.93	18.56

average F1 score, except for SVM. However, while the difference between the average F1 score of ECDNN and that of SVM is minimal (0.8657 vs. 0.8662), ECDNN works significantly faster (18.56 vs. 132.93). At the individual dataset level, both ECDNN and SVM rank top for 7 datasets each, but ECDNN is more robust on datasets with severe class imbalance problems. Therefore, from the results above, one can see that the superiority of the proposed ECDNN over those competitive algorithms is reflected from two perspectives.

- The proposed ECDNN outperforms all other algorithms in classification performance measured by the average F1 score.
- Alongside this improvement in classification performance, the computational efficiency has also been improved.

To further explore the sensitivity of the nearest neighbor based algorithms to the neighborhood size k, we also conducted a comprehensive experiment to show the average F1 score of k-NN, WKNN, CDNN and ECDNN in terms of k on 16 real-world datasets. The result is presented in Fig. 4, where there are 16 subplots, and each presents the experimental result for one dataset. In each subplot, the average F1 score is displayed on the y-axis, and the x-axis represents the number of nearest neighbors. A higher F1 score indicates better classification performance, while a flatter line means a higher degree of stability of the algorithm over different neighborhood sizes k. k-NN, WKNN, CDNN and ECDNN are depicted in different colors, so we can compare them and analyze their performance patterns over different k. It can be shown that the classification performance varies with the neighborhood size, and the degree of variation is significantly dependent on the dataset. Overall, the proposed ECDNN achieves significant performance improvement and a high degree of stability over k-NN, WKNN and CDNN for the majority of the datasets, except for `iris` and `wine`, where the data is simple. The robustness of ECDNN is also demonstrated by its small variation over large k, where the performance of the other three algorithms degraded significantly.

The results of the 5x2cv paired t-test are depicted in Fig. 5, where a darker shade of blue indicates a lower p value and a higher level of statistical significance. One can clearly see that the proposed algorithm can significantly improve the classification performance (i.e., $p < 0.05$) of k-NN for large k and datasets with high dimensionality, such as `digits`, `olivetti`, `satimage` and others. Therefore, with all the results presented above, we show that the proposed algorithm is more efficient and robust, especially with high-dimensional datasets. We also believe that the current study can shed some light on future ensemble model construction, in which local data can be integrated as part of the algorithm.

6. Conclusions

A large number of previous studies have proposed numerous improvements to k-NN from various perspectives. To the best of our knowledge, researchers have not yet focused on the homogeneity of the nearest neighbors of test instances. This could be a new research direction based on the recently proposed data-centric AI methodology, according to which both algorithms and data are essential components of an effective AI solution. Therefore, rather than optimizing a single algorithm for the entire test dataset, we can choose and apply different algorithms for each partition of the test dataset based on the data characteristics of each partition.

In this study, we proposed an ensemble learning method that consists of k-NN and CDNN. k-NN is the weak learner for queries with high confidence, while CDNN is the strong learner for queries with low confidence. In future work, one can replace k-NN and

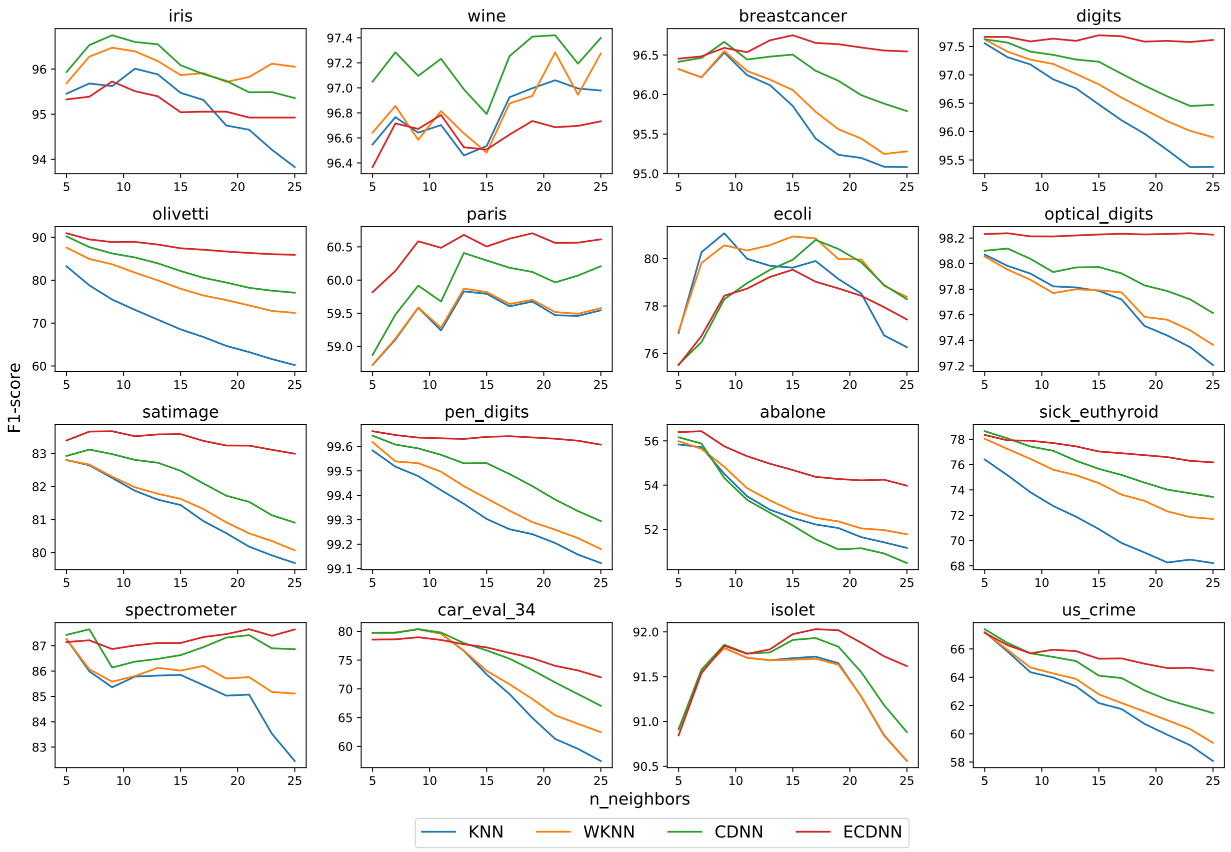

Fig. 4. Average F1 score of nearest neighbor based methods in terms of the neighborhood size k on 16 real-world datasets.

CDNN with other alternatives. For k-NN and most of its variants, one algorithm is trained and applied to the whole test dataset for prediction. There are also some k-NN ensemble algorithms where a set of classifiers is trained and combined using some form of voting for prediction. In our case, we categorize the test datasets into two partitions based on the homogeneity of the nearest neighbors of the query. If the query is located at the center of the clustering of a class, then there is likely a high degree of homogeneity, so the prediction is relatively easy to make with high confidence. On the other hand, if the query is located in the overlapping areas of different classes, then there is a low degree of homogeneity; therefore, we need a more sophisticated decision function for prediction in this case. Therefore, our proposed ECDNN is constructed based on a data-centric approach, where the homogeneity of the nearest neighbors controls which algorithm to apply based on the characteristics of the data.

The experimental results demonstrate the advantages of the proposed ECDNN from two perspectives: (1) with the weak learner is used for high confidence queries, it helps to reduce the computational complexity and improve the overall computational efficiency; and (2) the strong learner is used for low confidence queries, so the prediction efficiency is not compromised. We have also developed and open-sourced CDNN and ECDNN based on the `scikit-learn` API. Given that the `scikit-learn` library is one of the most widely used ML libraries, which enables an experimental pipeline that many different ML algorithms can be run, evaluated and compared in a rigorous and consistent manner [45], implementing the algorithm in the same API schema will accelerate future projects utilizing our algorithms.

Our future work will aim at further enhancements of the proposed ECDNN from two perspectives. Firstly, to overcome the limitation of the global k value, we will develop a new adaptive k value selection method based on local information. Secondly, we will examine the proposed ECDNN and distance metrics learning algorithms for heterogeneous data with a combination of numerical and categorical features. From the point of view of distance metric learning, we will look at how to handle and calculate distance for both categorical features and missing values.

CRediT authorship contribution statement

Alex X. Wang: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Stefanka S. Chukova:** Supervision, Validation, Writing – review & editing. **Binh P. Nguyen:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing.

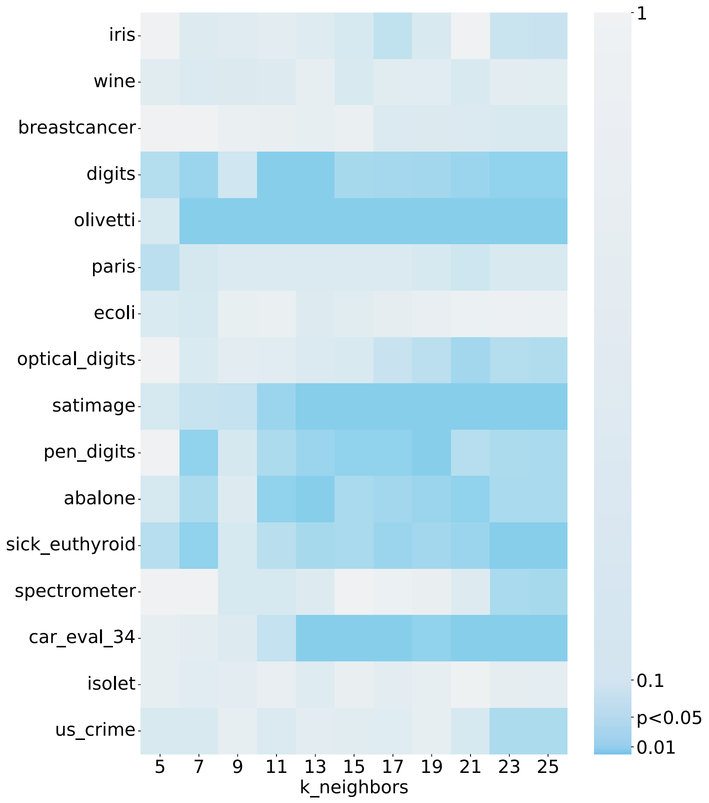

Fig. 5. Results of 5x2cv paired *t*-test by datasets to show whether the performance improvement is statistical significantly between ECDNN vs *k*-NN ($p < 0.05$).

Declaration of competing interest

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

Data availability

The data are public datasets which can be downloaded on the Internet. Code are provided with our Github link.

Acknowledgements

The work of BPN was partly supported by a research programme funded by the New Zealand Ministry of Business, Innovation and Employment (MBIE) under contract VUW RTVU1905 and the MBIE Strategic Science Investment Fund for Data Science under contract VUW RTVU1914.

References

[1] P. Soucy, G.W. Mineau, A simple KNN algorithm for text categorization, in: Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001), IEEE, 2001, pp. 647–648.
[2] S. Zhang, Cost-sensitive KNN classification, Neurocomputing 391 (2020) 234–242, https://doi.org/10.1016/j.neucom.2018.11.101.
[3] B. Wang, Z. Mao, A dynamic ensemble outlier detection model based on an adaptive k-nearest neighbor rule, Inf. Fusion 63 (2020) 30–40, https://doi.org/10.1016/j.inffus.2020.05.001.
[4] Y. Song, X. Kong, C. Zhang, A large-scale-nearest neighbor classification algorithm based on neighbor relationship preservation, Wirel. Commun. Mob. Comput. (2022), https://doi.org/10.1155/2022/7409171.
[5] M.-L. Zhang, Z.-H. Zhou ML-KNN, A lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048, https://doi.org/10.1016/j.patcog.2006.12.019.
[6] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, KNN model-based approach in classification, in: R. Meersman, Z. Tari, D.C. Schmidt (Eds.), OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”, in: LNCS, vol. 2888, Springer, 2003, pp. 986–996.
[7] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, et al., Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37, https://doi.org/10.1007/s10115-007-0114-2.
[8] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, Efficient k-nearest neighbors search in graph space, Pattern Recognit. Lett. 134 (2020) 77–86, https://doi.org/10.1016/j.patrec.2018.05.001.

[9] N. Rastin, M. Taheri, M.Z. Jahromi, A stacking weighted k-nearest neighbour with thresholding, Inf. Sci. 571 (2021) 605–622, https://doi.org/10.1016/j.ins.2021.05.030.
[10] J. Zhai, J. Qi, C. Shen, Binary imbalanced data classification based on diversity oversampling by generative models, Inf. Sci. 585 (2022) 313–343, https://doi.org/10.1016/j.ins.2021.11.058.
[11] S. Sengupta, S. Das, Selective nearest neighbors clustering, Pattern Recognit. Lett. 155 (2022) 178–185, https://doi.org/10.1016/j.patrec.2021.10.005.
[12] Z. Pan, Y. Wang, Y. Pan, A new locally adaptive k-nearest neighbor algorithm based on discrimination class, Knowl.-Based Syst. 204 (2020) 106185, https://doi.org/10.1016/j.knosys.2020.106185.
[13] S. Uddin, I. Haque, H. Lu, M.A. Moni, E. Gide, Comparative performance analysis of k-nearest neighbour (KNN) algorithm and its different variants for disease prediction, Sci. Rep. 12 (1) (2022) 1–11, https://doi.org/10.1038/s41598-022-10358-x.
[14] B.P. Nguyen, W.-L. Tay, C.-K. Chui, Robust biometric recognition from palm depth images for gloved hands, IEEE Trans. Human-Mach. Syst. 45 (6) (2015) 799–804, https://doi.org/10.1109/THMS.2015.2453203.
[15] E. Strickland, Andrew Ng, AI Minimalist: the machine-learning pioneer says small is the new big, IEEE Spectr. 59 (4) (2022) 22–50, https://doi.org/10.1109/MSPEC.2022.9754503.
[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830, http://jmlr.org/papers/v12/pedregosa11a.html.
[17] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop, Languages for Data Mining and Machine Learning, 2013, pp. 108–122.
[18] E. Fix, J. Hodges, Discriminatory analysis, nonparametric discrimination: Consistency properties, Technical Report 4, USAF School of Aviation Medicine, Randolph Field, 1951.
[19] L. Jiang, Z. Cai, D. Wang, S. Jiang, Survey of improving k-nearest-neighbor for classification, in: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), vol. 1, IEEE, 2007, pp. 679–683.
[20] R. Todeschini, D. Ballabio, V. Consonni, F. Grisoni, A new concept of higher-order similarity and the role of distance/similarity measures in local classification methods, Chemom. Intell. Lab. Syst. 157 (2016) 50–57.
[21] Ö.F. Ertuğrul, M.E. Tağluk, A novel version of k nearest neighbor: dependent nearest neighbor, Appl. Soft Comput. 55 (2017) 480–490, https://doi.org/10.1016/j.asoc.2017.02.020.
[22] N. Yamuna, J.A. Vijay, B. Gomathi, PSO-based hybrid weighted k-nearest neighbor algorithm for workload prediction in cloud infrastructures, in: The New Advanced Society: Artificial Intelligence and Industrial Internet of Things Paradigm, 2022, pp. 373–393.
[23] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137, https://doi.org/10.1109/TIT.1982.1056489.
[24] Y. Ruan, Y. Xiao, Z. Hao, B. Liu, A nearest-neighbor search model for distance metric learning, Inf. Sci. 552 (2021) 261–277, https://doi.org/10.1016/j.ins.2020.11.054.
[25] R. Bellman, Dynamic programming, Science 153 (3731) (1966) 34–37, https://doi.org/10.1126/science.153.3731.34.
[26] P. Zhao, L. Lai, Efficient classification with adaptive KNN, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11007–11014, https://ojs.aaai.org/index.php/AAAI/article/view/17314.
[27] A.B. Hassanat, M.A. Abbadi, G.A. Altarawneh, A.A. Alhasanat, Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach, https://doi.org/10.48550/ARXIV.1409.0919, 2014.
[28] X. Yuan, D. Ren, Z. Wang, C. Guo, Dimension projection matrix/tree: interactive subspace visual exploration and analysis of high dimensional data, IEEE Trans. Vis. Comput. Graph. 19 (12) (2013) 2625–2633, https://doi.org/10.1109/TVCG.2013.150.
[29] O. Kramer, Scikit-learn, in: Machine Learning for Evolution Strategies, Springer, 2016, pp. 45–53.
[30] D. Dua, C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.
[31] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 289–300, https://doi.org/10.1109/34.990132.
[32] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.
[33] J.L. Bentley, Survey of techniques for fixed radius near neighbor searching, Tech. Rep., Stanford Linear Accelerator Center, Calif. (USA), 1975.
[34] E. Elhamifar, R. Vidal, Sparse manifold clustering and embedding, Adv. Neural Inf. Process. Syst. (2011) 24.
[35] R. Tibshirani, T. Hastie, B. Narasimhan, G. Chu, Diagnosis of multiple cancer types by shrunken centroids of gene expression, Proc. Natl. Acad. Sci. 99 (10) (2002) 6567–6572, https://doi.org/10.1073/pnas.082099299.
[36] Z. Xie, W. Hsu, Z. Liu, M.L. Lee, SNNB: a selective neighborhood based Naive Bayes for lazy learning, in: 2002 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Springer, 2002, pp. 104–114.
[37] N. Biswas, S. Chakraborty, S.S. Mullick, S. Das, A parameter independent fuzzy weighted k-nearest neighbor classifier, Pattern Recognit. Lett. 101 (2018) 80–87, https://doi.org/10.1016/j.patrec.2017.11.003.
[38] H. Gweon, H. Yu, A nearest neighbor-based active learning method and its application to time series classification, Pattern Recognit. Lett. 146 (2021) 230–236, https://doi.org/10.1016/j.patrec.2021.03.016.
[39] B.L. Sturm, Classification accuracy is not enough, J. Intell. Inf. Syst. 41 (3) (2013) 371–406, https://doi.org/10.1007/s10844-013-0250-y.
[40] J. Xiao, Y. Wang, J. Chen, L. Xie, J. Huang, Impact of resampling methods and classification models on the imbalanced credit scoring problems, Inf. Sci. 569 (2021) 508–526, https://doi.org/10.1016/j.ins.2021.05.029.
[41] J.T. Townsend, Theoretical analysis of an alphabetic confusion matrix, Percept. Psychophys. 9 (1) (1971) 40–50, https://doi.org/10.3758/BF03213026.
[42] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognit. Lett. 30 (1) (2009) 27–38, https://doi.org/10.1016/j.patrec.2008.08.010.
[43] N. Seliya, T.M. Khoshgoftaar, J. Van Hulse, A study on the relationships of classifier performance metrics, in: Proceedings of the 21st IEEE International Conference on Tools with Artificial Intelligence, IEEE, 2009, pp. 59–66.
[44] S. Raschka, Model evaluation, model selection, and algorithm selection in machine learning, https://doi.org/10.48550/ARXIV.1811.12808, 2018.
[45] R.F. Zhang, R.J. Urbanowicz, A scikit-learn compatible learning classifier system, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (GECCO 2020), 2020, pp. 1816–1823.