Knowledge-Based Systems 239 (2022) 108002

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Robust gravitation based adaptive *k*-NN graph under class-imbalanced scenarios

Yuanting Yan [a], Tianxiao Zhou [a], Zhong Zheng [a], Hao Ge [b,*], Yiwen Zhang [a], Yanping Zhang [a]

[a] Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Anhui, 230601, China

[b] School of Electronic and Electrical Engineering, Chuzhou University, Chuzhou, 239000, China

ARTICLE INFO

Article history:
Received 16 July 2021
Received in revised form 14 December 2021
Accepted 17 December 2021
Available online 28 December 2021

Keywords:
Imbalanced learning
Graph-based learning
K-associated graph
Gravitational force
Classification

ABSTRACT

As a typical single parametric graph model, the *k*-Nearest Neighbor Graph (*k*-NNG) is characterized by its high efficiency in detecting topology information and remarkable capability for combining global and local features. However, graph-based supervised learning methods, including *k*-NNG, do not explicitly consider imbalanced class distribution; as a result, they often tend to perform vulnerable robustness in most imbalanced scenarios. This paper presents an adaptive *k*-NNG-based imbalanced classification method that can automatically determine the *k* value during graph construction. Our work presents two novel methodologies: (i) two types of adaptive graph construction methods to address the inherent complex characteristic of imbalanced class distribution; (ii) two graph-based gravitational classification rules to overcome the adverse bias towards the majority class in traditional KNN-based methods. The latter can effectively combine local nearest neighbors and subgraph information when calculating the gravitational force between pairs of vertices. Extensive experiments demonstrate the superiority of our method over other classification algorithms in imbalanced scenarios.

1. Introduction

As the most natural choice for data representation, the graph is already widely used in many tasks involving clustering, dimensionality reduction, manifold learning, etc. Typical representative applications include protein interactions in biology, friend recommendations in social networks, etc. The reasons graph-based research has received such widespread attention can be attributed to: (1) its ability to capture data topological information as a form of representation for relationships between arbitrary entities, which is probably the most important feature [1–3]; (2) the properties of the graph make it possible to effectively combine local features with global information [4]; (3) graphs are capable of representing higher-order structures from dyadic relationships between entities, including neighborhoods, communities, modules, connected components, etc. [5]; (4) it enables the detection of clusters of arbitrary shapes, owing to its ability to capture local topological features of the data distribution and express the heterogeneity of local topological information [6,7]; (5) graphs are superior interpretable models that validate the hypothesis under study by exploring these generated network models in conjunction with a priori knowledge of the relevant domain. Existing network measures [8], such as density, degree distribution, clustering coefficients, centrality, average path length, etc., as well as measures proposed for research-based directions, can be used to explain the similarities and differences of these networks.

Graph-based models facilitate important methods in machine learning, data mining, etc. For example, unsupervised learning includes graph-based clustering [6,7], unsupervised multiple graph learning based feature selection [9], construction of local structures based on nearest neighbor graphs in manifold learning [10,11], etc. In semi-supervised learning, the graph-based method also exceeds expectations in learning local and global consistency [4], but its value is more apparent in supervised learning owing to its efficiency in learning topology information. The tasks in this area can be summarized into two groups: one involves classifying graph data directly by measuring similarities between graph pairs, such as the classification of chemical compound [12,13]; the other involves identifying vector data as vertices in a graph and then classifying them within the graph structure—a typical representative of this methodology is the nearest neighbor graph (NNG) model [14,15]. The construction of the NNG is one of the most important aspects of these types of methods; in recent years, a myriad of methods for constructing *k*-NNG were proposed, which can be roughly divided into two categories according to the strategies in selecting neighborhoods: ε-ball based

* Corresponding author.
E-mail address: togehao@126.com (H. Ge).

https://doi.org/10.1016/j.knosys.2021.108002
0950-7051/

methods and the k-nearest neighbor (k-NN) methods [16,17]. The former makes a bounded threshold ε for any pair of vertices v_i and v_j; the two vertices will be connected only if $dist\left(v_i, v_j\right) \leq \varepsilon$; however, it tends to produce a large number of non-connected subgraphs or even fully connected graphs since the threshold is also sensitive to data distribution. On the contrary, the k-NN methods generate edges for each vertex with its top k nearest vertices to retain the local nature of the distribution, which makes it more robust than the ε-ball based methods.

The methods of constructing a k-NNG can also be categorized into two parts: unsupervised solutions and supervised solutions. The main difference between the two solutions is based on whether label information is adopted to construct the graph. For the supervised graph construction solution, Rohban, et al. [18] concluded that using label information can optimize the graph structure, and they proved that the optimal NNG is a subgraph of a graph with larger k-valued nearest neighbors. A high-level classification was proposed by Thiago [3] et al. which constructed the class topologies and pattern information from training data, specifically, to construct a graph network that accurately fits the topological information of the data, the method combines the characteristics of both the k-NN-based graphs and ε-radius graphs, using the ε-radius method in densely sampled regions and the k-NN-based method in sparse regions. A crucial drawback of the traditional k-NN-based undirected NNGs is their tendency to generate vertices with an unbalanced number of edges (i.e., the degree of the vertex is much greater than the others') [8]. To alleviate this phenomenon, Jebara et al. [19] proposed a b-matching graph that ensures that each vertex maintains the same number of edges and produces a balanced (regular) graph. Zhang et al. [11] proposed adaptive manifold learning, which realized an adaptive neighbor selection criterion based on neighborhood contraction and expansion. Cheng et al. [17] proposed a sparse coding-based ℓ^1-norm nearest neighbor graph reconstruction method called ℓ^1-graph, which not only adaptively selects neighborhood for each vertex, but also exhibits greater resistance to data noise.

Bertini et al. [14] proposed a labeled k-NNG called a K-associated graph (KAG), whereby only vertices with the same class may be connected, and each class consists of a series of components. Furthermore, the concept of purity is also proposed to describe the degree of connectivity of vertices within different connected components, and it finally generates the optimal NNG by iteratively fusing subgraphs, which is called the K-associated optimal graph (KAOG). The KAOG does not need to preset the value of k; moreover, it does not use a graph kernel or Laplacian. Mahdi et al. [15] proposed a modified K-association optimal graph (MKAOG) to address the KAOG construction process' sensitivity to noise samples by modifying the fusion rules of KAOG so that smaller connected components could be merged together. However, all the above methods assume a balanced dataset not commonly found in real-life scenarios; in binary scenarios, one class (the majority class) has a much greater number of samples than the other (the minority class). He et al. [20] concluded that the decision boundaries of general classifiers tend to favor the majority class. There is also a general problem of class overlap in imbalanced datasets. The above problems affect the construction of NNGs, where the composition process is highly likely to ignore a small number of class vertices, preventing the effective formation of their nearest-neighbor structure. Minority class samples tend to generate a large number of isolated vertices, and because the minority class may decompose into several smaller sub-concepts–unlike noisy samples–the minority class sub-graphs consisting of a few samples (outlier concepts or disjoint) may be far from the dominant minority area. Moreover, because these sub-graphs lack a sufficient sample size, the minority class NNG formed by the nearest neighbor rule is susceptible to becoming an isolated sub-graph, causing a lack of minority-connected components of appropriate size, ultimately deteriorating the classification performance of the minority class. Therefore, the construction of a robust k-NNG under imbalanced scenarios remains a challenge.

To address the aforementioned issues, this paper proposes a heuristic method for minority class NNG construction called IMbalanced K-associated Optimal Graph (IMKOG), which heuristically removes majority samples that affect the purity of those components during the iterative graph construction process. The underlying mechanism of IMKOG expands the potential minority topographical region without decreasing the purity of minority subgraphs, thus, reducing incorrect skewing towards majority classes. Meanwhile, this study designs an adaptive k-value selection method for prediction according to the nearest-neighbor principle. The process can not only determine the candidate set used to calculate the test sample gravity with adaptive k-values but can also take the neighbor information, structural information of the associated subgraph, and imbalance rate into account in the gravity value calculation.

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

- We propose a new k-NNG called IMKAOG which extends the k-NNG model to imbalanced scenarios.
- We investigate two structural strength-based methods for the construction of k-NNG in class-imbalance distribution.
- Two gravity-based classification criteria are proposed that not only decide the number of neighbors to adaptively test vertices, but also utilize the topological information for the calculation of gravitation.
- Experiments conducted on imbalanced data demonstrate the effectiveness of our model when compared with other methods.

The remainder of this paper is organized as follows: Section 2 presents the related work; Section 3 describes the proposed IMKOG algorithm; Extensive experiments are described in Section 4; And finally, Section 5 presents the Conclusion of this work.

2. Related work

2.1. KAG

Definition 1 (*k-Nearest Neighbor Graph (k-NNG)*)**.** Given a training dataset $X \in \mathbb{R}^{n\times d}$, the corresponding k-NNG can be defined as $G = (V, E)$. V $= \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, $\forall v_i \in V$ corresponds to a sample $x_i \in X$, and the edge set $E = \left\{e_{ij}\right\}$ represents the union of edges corresponding to the k nearest neighbors of all vertices. For an arbitrary vertex $v_i \in$ V, there exists an edge $e_{ij} = \left\langle v_i, v_j\right\rangle, v_j \in N_i^k$, specifically, $\left\langle v_i, v_j\right\rangle = \left\langle v_j, v_i\right\rangle$ denotes the existence of an undirected edge between v_i and v_j, i.e., v_i is one of the k-nearest neighbors of v_j or v_j is one of the k-nearest neighbors of v_i. Moreover, if a number is endowed to the edge in the graph, it is called a weighted graph; otherwise, the graph is unweighted.

Definition 1 does not consider the direction of the edges in the k-NNG, which is an undirected graph. It should be noted that a vertex v_j as the nearest neighbor to v_i does not imply that v_i is also the nearest neighbor of v_j, and the K-associated graph (KAG) proposed in [14] is a directed unweighted NNG more consistent with the asymmetric representation of most real-world datasets. Furthermore, KAG is a label-dependent NNG generated with a fixed value of k, and it can be broadly divided into three parts:

Step1: For each vertex v_i, the label-dependent k nearest neighbors are obtained and denoted as $\Delta_{v_i,k}$; subsequently, the corresponding edge set E is built based on $\Delta_{v_i,k}$.

$$\Delta_{v_i,k} = \left\{ v_j \,\middle|\, v_j \in N_{v_i,k} \wedge y_{v_j} = y_{v_i} \right\} \tag{1}$$

Step2: For the vertices set and edge set obtained by **Step1**, the connected component $\{C_\alpha\}$ corresponding to the graph G=(V, E) can be obtained by depth-first-search (DFS) or disjoint-set forests.

Step3: Calculate the purity of each component with the following equation:

$$D_\alpha = \frac{1}{N_\alpha} \sum_{v_i \in C_\alpha} \left(d_i^{in} + d_i^{out} \right) \tag{2}$$

Here C_α represents a component, v_i is a vertex in C_α, d_i^{in} is the number of edges starting from v_i, and d_i^{out} shows the number of edges ending with v_i. N_α is the number of vertices with C_α; hence, D_α denotes the average degree of C_α. A larger value of D_α indicates a higher probability of connectivity between the vertices in this component. Then, the purity of C_α is calculated as follows:

$$\Phi_\alpha = \frac{D_\alpha}{2k} \tag{3}$$

To analyze this formula, assuming that there exists a component and the number of nearest neighbors is k, the purity is at maximum when the k nearest neighbors of all vertices in the component have the same label, which means that the total out-degree and in-degree of this component are both kN_α; so the D_α of this component is $2k$. Using Eq. (2), the corresponding purity in this case is $max\,(\Phi_\alpha) = 1$. Conversely, if all the nearest neighbors of a vertex in the component are from different classes, then this component is composed of a single vertex with $D_\alpha = 0$ and its purity is 0. In summary, the value of purity Φ_α ranges from 0 to 1.

2.2. KAOG and MKAOG

For a given dataset, KAG provides all components the same value of k. However, this operation implicitly assumes that the data is uniformly distributed in the feature space. This assumption is too strict to apply on real dataset, and thus, the K-associated optimal graph (KAOG) is constructed based on KAG in [14].

To find the best value of k for each component, the KAOG proposes a heuristic iterative method that starts from a KAG with $k = 1$ and gradually increases the k-value to obtain a sequence of different KAGs. In this way, the KAOG records the best component by comparing the components with different values of k, where the key of the process is the corresponding purity value, as it measures the degree of connectivity of vertices within the current component; the higher the purity of the connected component, the higher is the probability that a new vertex will require classification into that component in the testing phase. The criterion of comparison is stated as follows:

$$\Phi_\beta^{(K+z)} \geq \Phi_\alpha^{(K)} \quad forall \quad C_\alpha^{(K)} \subseteq C_\beta^{(K+z)} \tag{4}$$

Eq. (4) represents the criterion for determining whether a series of components (i.e. $C_\beta^{(K+z)}$) will be accepted for fusion during the iterative process. α, β are the subscripts of the corresponding components. Fig. 1 gives a schematic diagram of the fusion process of KAOG with z=1 (k increases by 1 at each iteration), and the iterative process starts at k = 1; thus, the current corresponding optimal graph is set as G^{opt} : $G^{opt} \leftarrow G^{k=1}$, shown at the top left of Fig. 1, which exhibits a part of G^{opt} with two components $C_1^{k=1}$ and $C_2^{k=1}$ represented as red circles, and then as $k = k+1$, a new graph $G^{k=2}$ is obtained by KAG with $k = 2$, which is depicted at the top right of Fig. 1. By comparing the two subplots in Fig. 1, we

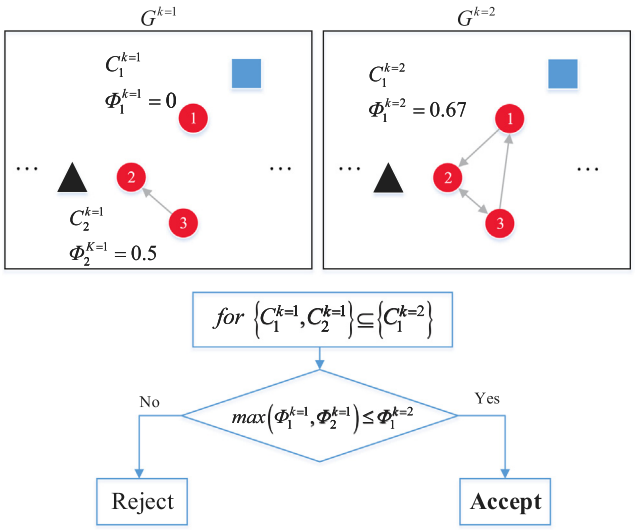


Fig. 1. The merging process in the KAOG algorithm.

demonstrate that $C_1^{k=1}$ and $C_2^{k=1}$ tend to merge together in $G^{k=2}$, and according to Eq. (4), this merging process is accepted only if the purity of the corresponding new component ($C_1^{k=2}$) is higher than the purity of $C_1^{k=1}$ and $C_2^{k=1}$. Then, the new component ($C_1^{k=2}$) will replace the $C_1^{k=1}$ and $C_2^{k=1}$ in G^{opt}; otherwise, G^{opt} remains unchanged. This process of iteration ensures all components of $G^{k=2}$ are traversed.

Bertini et al. [14] proposed to abort the entire iterative update process by employing a heuristic criterion. However, Eq. (4) is a strict criterion; the small components are barely merged together when in the presence of excessive noise. To solve this problem, in the graph construction phase, Mohammadi et al. [15] modified the merging rule in KAOG to make it more suitable for subgraph merging in scenarios with a high level of noise, called MKAOG, and the modified rules are as follows:

$$If \left(\Phi_\beta^{(k)} \geq \frac{1}{N} \sum \Phi_\alpha^{(opt)} \right) forall \quad C_\alpha^{(opt)} \subseteq C_\beta^{(k)}, \tag{5}$$

where α, β are the subscripts of the corresponding components. Under this rule, the subgraphs are allowed to be fused if the purity of the new component obtained after fusion is not smaller than the average of the purity of all components before fusion. This approach makes the merging process of small connected components less sensitive to the presence of noise.

2.3. Graph-model for class-imbalance problem

Class imbalance poses a significant challenge to traditional machine learning methods, which lean to focusing more on the majority class but ignoring the minority class. In addition to the degree of class imbalance, the learning of imbalanced data is related to many factors such as disjoint samples and class overlap, etc. To illustrate this, Fig. 2 shows a schematic representation of class imbalance and class overlap under binary classification: the minority class is distributed among several small regions (A, B, C, D) in this example, and displays a significant class overlap in region B.

The imbalance problem is a primary concern in the field of machine learning [20]. A considerable number of algorithms have emerged to deal with the classification of imbalanced data which investigate the reliability of imbalance learning from a variety of perspectives. Recent studies have shown that class overlap

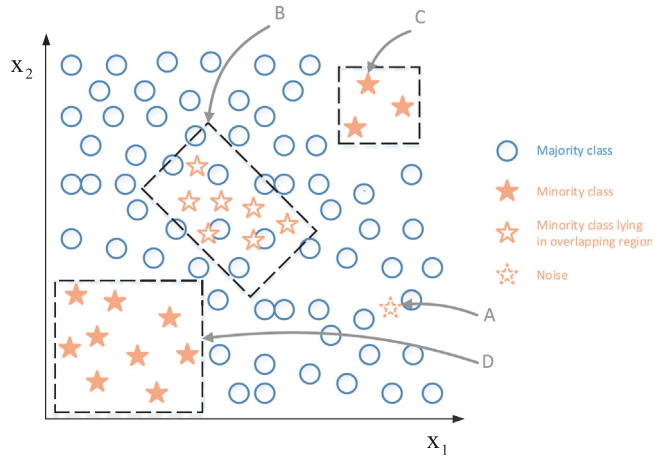


Fig. 2. Example of imbalanced data distribution. The circles in the figure indicate the majority class samples, and the pentagrams represent the minority class samples.

is a primary factor in the deterioration of classification performance [21–23].

Relevant imbalance learning involving the graph-based model has also been studied [24–26], and their algorithms address the class imbalance problem primarily using graph ensemble or structural information to suppress the adverse effects of data imbalance on classification. For example, Pan et al. [24] found that the selection of subgraph features is also affected by class imbalance, and a boosting framework was proposed that combined the subgraph feature selection and margin optimization process to progressively select more valuable subgraph features from imbalanced data; this makes it more suitable for imbalanced and noise-laden graph data classification tasks.

Liu et al. [25] proposed a graph-based boosting algorithm for semi-supervised learning by assigning higher weights to minority classes, to accommodate the imbalanced classification of data. Manukyan et al. [26] used a graph model called "class cover catch digraphs" (CCCD) to deal with class imbalance and overlap problems in classification, which undersamples the majority classes while retaining their missing class information. However, these methods mainly focus on either utilizing the advantage of the graph-based model in learning data structure information or conducting a resampling process in a graph to improve imbalanced data learning performance. Research on designing specific graph construction methods for imbalanced data is scarce.

3. The proposed method

As with general graph-based classification methods, the proposed IMKOG has two main phases: constructing the graph with training data and assigning the "testing data" label to the new vertices. In the training phase, a new version of KAG is proposed that embeds the majority cleaning in overlapping regions into the graph learning process, which is supervised by the graph construction iteration. In the testing phase, two gravity-based classification rules are introduced to the k-NNG-based classification model.

3.1. IMKG

In this section, we describe a k-NNG called "**IM**balanced **K**-associated **G**raph" (IMKG) that has the same configuration as KAG, in that it generates the corresponding directed NNGs by the rule of label dependency for a given value of k. It should be noted, however, that the IMKG introduces two extra methods: (1) The generation of components represented in this study by a disjoint tree of $\{C_{min}^i\}$ for minority classes of vertices. (2) For each component C_{min}^i, IMKG defines a concept set $\varphi_{v_j,k}$ as Eq. (6), and the shortest distance l_{v_i} from $\varphi_{v_j,k}$ to the associated v_j in C_{min}^i is defined in Eq. (8).

$$\varphi_{v_j,k} = \left\{ v_i | v_i \in N_{v_j}^k,\, v_j \in C_{min}^\alpha \wedge y_{v_i} \neq y_{v_j} \right\}, \tag{6}$$

where v_i denotes the vertex that belongs to the majority class, and it is one of the k nearest neighbors of minority vertex v_j; in this paper, we will refer to these as "negative neighbors" for vertex v_j. $N_{v_j}^k$ is the k nearest neighbors of v_j, and y_{v_i} is the label of v_i.

To better understand the concept $\varphi_{v_j,k}$ described in Eq. (6), Fig. 3 gives the NNGs on one synthetic dataset with various k values. Subplot (b) shows the k-NNG with $k = 1$; subplot (c) gives part of subplot(b), where vertexes 98 and 118 are the negative neighbors (majority class) of minority vertexes 20, and 29, respectively according to Eq. (6); and shows that these two vertices are also the negative nearest neighbors of minority components $C_1 = \{20, 28, 43\}$, $C_2 = \{23, 29\}$, respectively. Subplot (d) shows part of the k-NNG with $k = 2$, the vertices 98, 118, and 143 are the negative neighbors of minority vertices 20, 29, and 23, respectively, and similarly, vertices 98, 118, and vertices 98, 118, 148 are the negative neighbors of minority component $C_1 = \{20, 28, 43\}$ and $C_2 = \{23, 29\}$, respectively. Accordingly, we extend the concept of negative neighbors from "vertex-vertex" to "vertex-component". Thus, the set of all negative neighbors for a given component can be obtained as follows:

$$\varphi_{C,k} = \bigcup_{v_j \in C_{min}^\alpha} \varphi_{v_j,k}, \tag{7}$$

where C_{min}^α denotes a minority component and $\varphi_{C,k}$ denotes the union of all negative neighbors of the vertices within the minority component. Thus, the shortest distance l_{v_i} from each negative neighbor to the associated component in Eq. (7) can be defined as follows:

$$l_{v_i} = \min \left\{ d\left(v_i, v_j\right) \middle| v_j \in N_{v_i}^{-1} \right\}, \tag{8}$$

where $d(\cdot)$ denotes the Euclidean distance, v_i is the negative neighbor, and $N_{v_i}^{-1}$ represents all the minority vertices that consider v_i as a negative neighbor in the current component. It should be noted that one majority vertex may belong to the neighbor of several minority vertices in one component. Correspondingly, there exist several distance values; the smallest is adopted to represent the distance between the negative neighbor and the minority component. Then, for a given minority component, the set of the shortest distances corresponding to all negative neighbors is defined as follows:

$$\ell_C = \bigcup_{v_j \in \varphi_{C,k}} \{l_{v_j}\}. \tag{9}$$

Similarly, when considering Fig. 3 as an example, one can observe that the number of directed edges which point from the minority class to the corresponding negative nearest neighbors increases proportionally with the value k, represented by the yellow directed edges in the figure. This is especially apparent in the class overlap region, and the presence of these negative vertices can seriously hinder the fusion of minority subgraphs. Processing these negative samples efficiently is crucial for the subsequent construction of the NNG.

Considering the complexity of the data distribution, this study further proposes a method for detecting the key negative neighbors in $\varphi_{C,k}$ by combining the structural information of the corresponding component. Specifically, it uses structural information

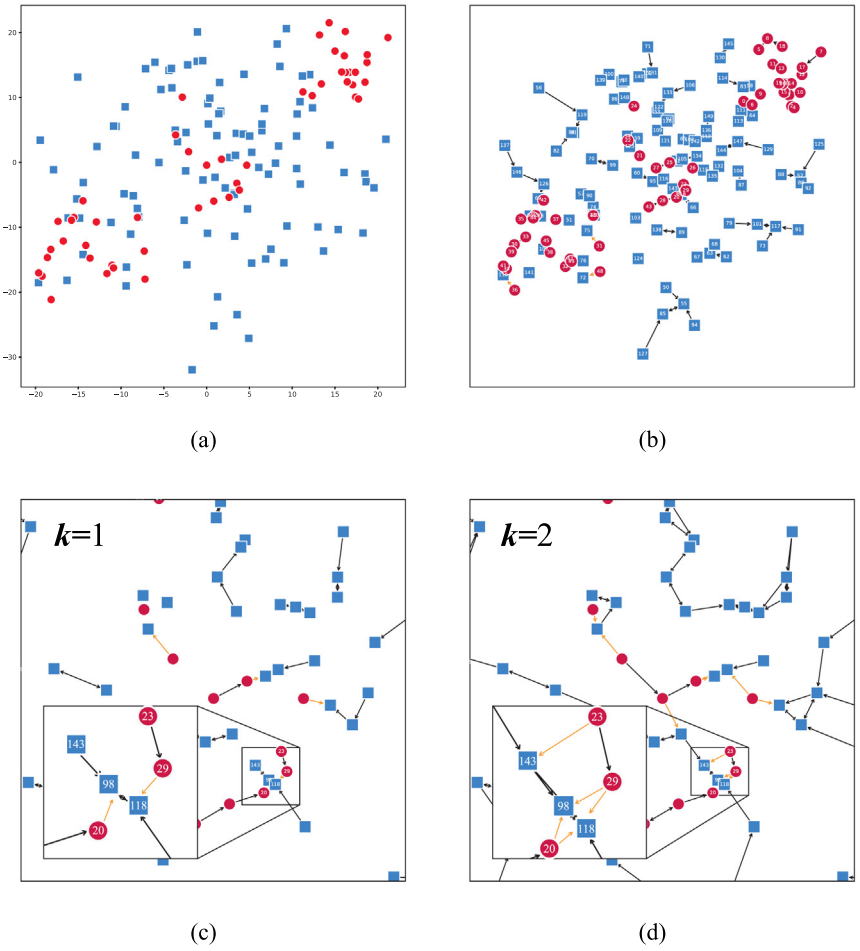


Fig. 3. Detecting the negative neighbors of vertices in k-NNG.

to constrain the deletion of samples in the overlapping region to avoid possible deletion of important majority vertices. A minority component C^i_{min} contains a set of edges and weights w_{ij}, which are measured using the Euclidean distance $d(v_i, v_j)$ between each pair of vertices. In addition, we employ the weight for measuring the structural strength $s_{e_{ij}}=1/d(v_i, v_j)$ of each edge within the current subgraph. Thus, if two vertices have a shorter distance between them, they have a greater structural strength value.

Let $S_C = \bigcup_{e_{ij}\in E^{\alpha}_{min}} \{s_{e_{ij}}\}$ be the union of the edge strength in a minority component. This study investigates two criteria: the maximum strength criterion and the average strength criterion, to remove part of the negative neighbors.

$$(I)\ CDS_{C,k} = \left\{v_i | 1/l_{v_i} \geq max\left(S_C\right),\ v_i \in \varphi_{C,k},\ l_{v_i} \in \ell_C\right\} \tag{10}$$

$$(II)\ CDS_{C,k} = \left\{v_i | 1/l_{v_i} \geq mean\left(S_C\right),\ v_i \in \varphi_{C,k},\ l_{v_i} \in \ell_C\right\} \tag{11}$$

From Eqs. (10) and (11), it is demonstrated that, in general, the number of negative neighbors that satisfy Eq. (10) is less than that in Eq. (11). Therefore, Eq. (10) is a stricter constraint compared to Eq. (11).

In Fig. 4, the yellow edges represent the directed edges from the minority vertices to its negative neighbors, and the black edges denote the directed edges between two minority vertices. There are three subgraphs **a**, **b**, **c**; for simplicity, we use subgraph **a** as an example to depict the process of detecting candidate sets for key negative neighbors. There are three edges in component **a** with strength values s_1, s_2, s_3 ($s_2 > s_1 > s_3$); hence, $S_{C^a_{min}} = \{s_1, s_2, s_3\}$. The associated negative neighbors of **a** are vertices 1 and 2, i.e., $\varphi_{C_a,k} = \{v_1, v_2\}$, and the distance between vertex 1 and **a** is $l_1 = \min\left\{d\left(v_1, v_i\right) | v_i \in N^{-1}_{v_1}\right\} = d_1$. Similarly, $l_2 = d_2$. According to Eq. (10), $1/l_1 > max\left(S_{C^a_{min}}\right) = s_2$, $1/l_2 < max\left(S_{C^a_{min}}\right) = s_2$; therefore only v_1 will be added to the candidate set. If Eq. (11) is selected as the criterion, $1/l_1 > mean\left(S_{C^a_{min}}\right) =$

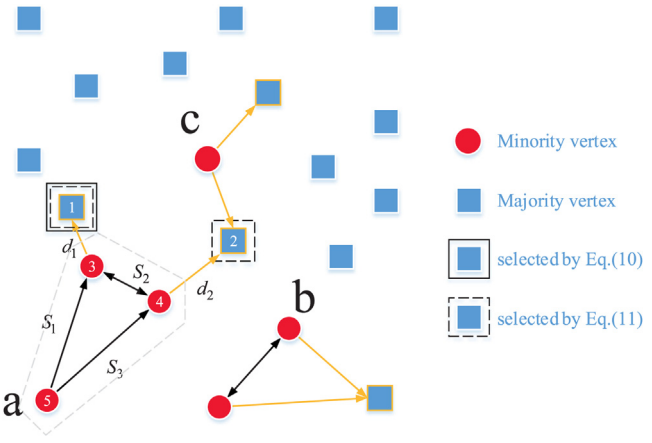


Fig. 4. The sketch map to illustrate negative neighbors according to Eqs. (10) and (11).

$\frac{1}{3}(s_1+s_2+s_3)$, $1/l_2 > mean\left(S_{C^a_{min}}\right)=\frac{1}{3}(s_1+s_2+s_3)$, then v_1 and v_2 are included in the candidate sets. Accordingly, by using the above two-candidate-set detection criteria, we can obtain an IMKG (Algorithm 1) which consists of a series of components, corresponding candidate sets, and the purity value calculated by the KAG algorithm, which is stated as $G^k=\bigcup\left\{C_i, \Phi_{C_i}, CDS_{C_i}\right\}$.

Algorithm 1: Constructing Imbalanced K-associated Graph **IMKG**

Input: imbalanced data(D), k;
Output: an imbalanced k-associated graph: $G^k=\bigcup\left\{C_i, \Phi_{C_i}, CDS_{C_i}\right\}$;
Obtaining $G^{(k)}=\{C_1,...,C_i,...,C_m\}$ according to the KAG algorithm where component $C_i=\left(G'\left(V',E'\right);\Phi_i\right)$;
for $C_i \in G^{(k)}$ **do**
  Calculate the negative neighbors $\varphi_{C_i,k}$ of the current component C_i // Eq. (6) , (7)
  Calculate the shortest distance ℓ_C from all $\varphi_{C_i,k}$ to the component C_i // Eq. (8), (9)
  Obtain the corresponding set of candidates $CDS_{C_i,k}$; // Eq. (10) or (11)
  Update $G^{(k)} \leftarrow G^{(k)} \cup \left\{CDS_{C_i,k}\right\}$
end
Return $G^k=\bigcup\left\{C_i, \Phi_{C_i}, CDS_{C_i}\right\}$.

3.2. IMKOG

Our method first constructs an IMKG for the minority class with a specific k, then employs an iterative subgraph fusion rule by gradually increasing the value of k to obtain an optimal minority k-NNG. It should be noted that a purity-based posterior method is embedded in the iteration to determine which vertex should be removed from the candidate set.

Fig. 5 shows a schematic diagram of iterative subgraph fusion process starting with $k=1$. Fig. 5(a) shows the minority subgraph with $G^{opt} \Leftarrow G^{k=1}$, then $G^{k=2}$ is obtained by IMKG algorithm with $k=2$, as shown in Fig. 5(b); using $C_1^{k=2}, C_2^{k=2}$ as examples, their negative neighbors $\varphi_{C_i,k}$ can be obtained, i.e. $\varphi_{C_1,k}=(1,3,4)$, $\varphi_{C_2,k}=(2,4)$. According to the two detection rules represented by the proposed Eqs. (10) and (11), we can obtain two different candidate sets. Fig. 5(c) shows the result by applying Eq. (10) as the detection rule, whereby there are two minority components $C_1^{k=2}$ and $C_2^{k=2}$ in $G^{k=2}$, and their corresponding candidate sets are 3, 4 and 4 respectively. Therefore, the candidate set of the minority graph $G^{k=2}$ is (3, 4). In this case, we obtain a new training set by temporarily removing the candidate set (i.e., (3, 4)) from the initial training set, and obtain the corresponding new graph $\hat{G}^{k=2}$ by IMKG algorithm, as shown in Fig. 5(e). Next, we make a judgement to whether accept the fusion or not. We compare G^{opt} (subplot (a)) and $\hat{G}^{k=2}$ (subplot (e)) to decide whether to fuse the components in $G^{k=2}$ according to Eq. (5). The components that accept the fusion will replace the corresponding components in G^{opt}. Here, $\hat{C}_1^{k=2}$ replaces C_1^{opt} and C_2^{opt} in G^{opt}, and the candidate set in $G^{k=2}$ (i.e., 3, 4) will be removed from the training set, which is denoted as $Discard_{\hat{C}_i,k}$:

$$Discard_{\hat{C}_i^k,k}=\left\{v_j | v_j \in CDS_{C_\alpha^k} \wedge C_\alpha^k \subseteq \hat{C}_i^k\right\} \tag{12}$$

The above iterative update process is terminated when the purity of all components is nearly unchanged; specifically, the stopping condition is in accordance with the KAOG algorithm. Because the number of minority vertices remains constant during the construction process, some of the majority vertices are removed during construction of the minority optimal graph, after which the majority optimal graph $G_{maj}^{(opt)}$ is constructed. It should be noted that the majority optimal graph is constructed using the MKAOG method. Algorithm 2 presents the proposed IMKOG algorithm.

Algorithm 2: IMbalanced K-Optimal Graph **IMKOG**

Input: Imbalanced data (D);
Output: Imbalanced K-optimal graph $G^{(opt)}$;
$G^{(opt)}=\emptyset, D^{(opt)}=D, k=1$;
step1: Construct minority class optimal nearest neighbor graphs
$G_{min}^{opt} \leftarrow IMKG\left(D^{(opt)},k\right)$
while $De^{(k)} - lastAvgDegree > De^{(k)}/k$ **do**
  $lastAvgDegree \leftarrow De^{(k)}$
  $D^{(temp)}=D^{(opt)}$
  $k=k+1$
  $G_{min}^{(k)} \leftarrow IMKG\left(D^{(opt)},k\right)$
  for ***all*** $CDS_{C_i,k}$ ***in*** $G_{min}^{(k)}$ **do**
    $D^{(temp)}=D^{(temp)}-CDS_{C_i,k}$
  end
  $\hat{G}_{min}^{(k)} \leftarrow IMKG\left(D^{(temp)},k\right)$
  for $\hat{C}_\beta^{(k)} \subset \hat{G}_{min}^{(k)}$ **do**
    if $\hat{\Phi}_\beta^{(k)} \geq \frac{1}{N}\sum \Phi_\alpha^{(opt)}$ forall$C_\alpha^{(opt)} \subseteq \hat{C}_\beta^{(k)}$ **then**
      $G_{min}^{opt}=G_{min}^{opt}-\cup_{C_\alpha^{(opt)} \subseteq \hat{C}_\beta^{(k)}} C_\alpha^{(opt)}$
      $G_{min}^{opt}=G_{min}^{opt} \cup \left\{\hat{C}_\beta^{(k)}\right\}$
      $Discard_{\hat{C}_i^k,k}=\left\{v_j | v_j \in CDS_{C_\alpha^k} \wedge C_\alpha^k \subseteq \hat{C}_i^k\right\}$
      $D^{(opt)}=D^{(opt)}-Discard_{\hat{C}_\beta,k}$
    end
  end
end
step2: Construct majority class optimal nearest neighbor graphs
The optimal nearest-neighbor graph for majority classes G_{maj}^{opt} is constructed using the method proposed in MKAOG [15].
step3: Obtaining IMKOG
Return $G^{opt}=G_{min}^{opt} \cup G_{maj}^{opt}$

3.3. Gravity-based classification methods

In the testing phase, traditional methods equate the class label of a test vertex to the probability of the vertex belonging to different components consisting of vertices with the same label, such as the vertex classification method of Bayes theory in [14]. In the data imbalance scenario, such methods are highly susceptible to dominance by the majority class, which means that Bayesian prior probabilities tend to favor the majority class and ultimately affect the performance of the test sample. In addition, the value of k in the testing phase has a significant impact on the prediction performance [11,12]. In recent years, methods based on modified gravitational rules have emerged [27–29], and we propose using gravitational rules as a substitute for the NN-based method to overcome the aforementioned drawback. In addition, these modified methods have been extended to imbalance learning to alleviate the negative effect of data imbalance on classification by assigning higher weights to minority samples, thus, improving their classification performance.

In this study, gravity rules are introduced into the classification based on the NNG. Furthermore, this study proposes two adaptive gravity classification rules by considering the choice of appropriate nearest neighbors: specifically, one combines local information and the other combines sub-graph structure information in classification.

3.3.1. Candidate sets for gravitational calculations

In this study, the k values of the test vertex are determined by the local distribution; specifically, by means of the k values of the corresponding components. Thus, the candidate sets for gravitation force calculations are determined by the k values of the test vertex adaptively.

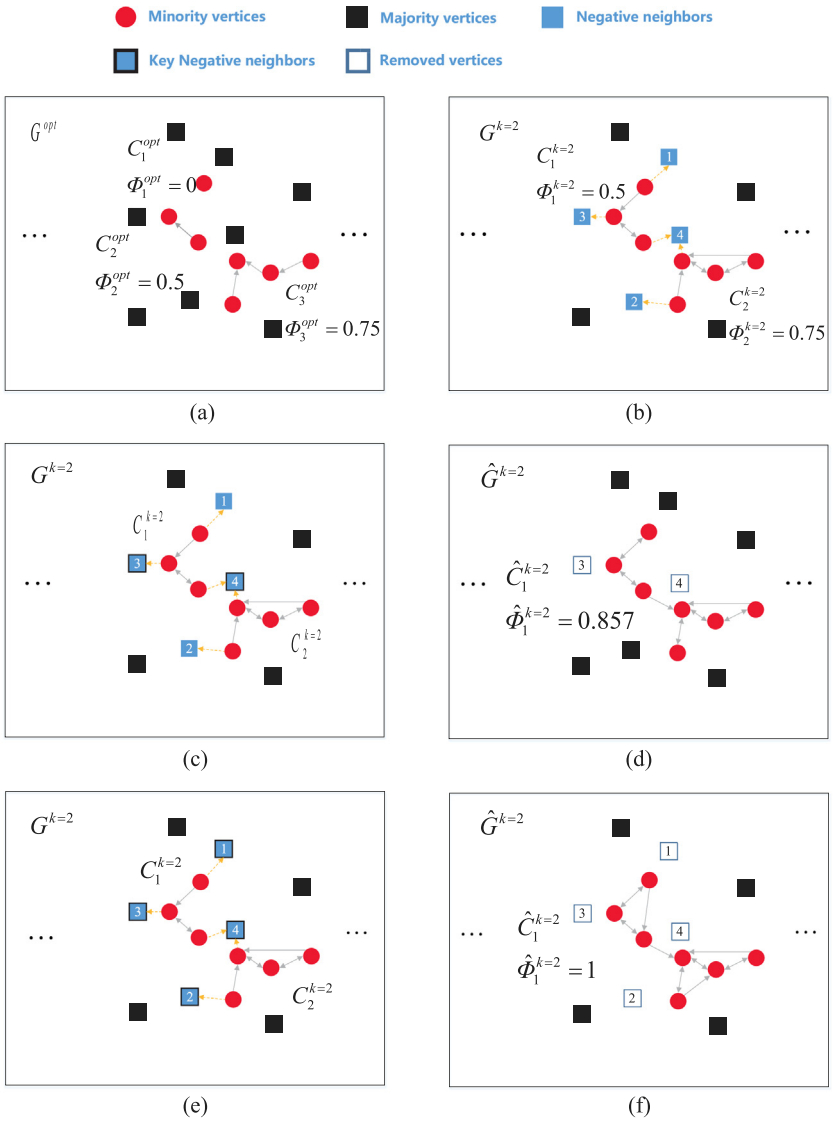


Fig. 5. Flowchart of the iterative fusion of a minority class subgraph with combined cleaning operation: (a) The 1-associated graph. (b) The 2-associated graph with negative neighbors drawn as blue squares. (c) The key negative neighbors selected according to Eq. (10). (d) A new imbalanced 2-associated graph constructed with input data, excluding the key negative neighbors in subplot (c). (e) The key negative neighbors selected according to Eq. (11). (f) A new imbalanced 2-associated graph constructed with input data, excluding the key negative neighbors in subplot (e).

The minority components $C_{min} = \left\{C_{min}^1, C_{min}^2, \ldots, C_{min}^{\mathcal{N}_{min}}\right\}$ and the majority components $C_{maj} = \left\{C_{maj}^1, C_{maj}^2, \ldots, C_{maj}^{\mathcal{N}_{max}}\right\}$ are obtained in the training phase with corresponding values of k and purity Φ. Given a test vertex v_{test}, the nearest neighbor v_{test}^{nn} of v_{test} can be found first, and the value of the nearest neighbor k_{test} for prediction is determined as follows:

$$k_{test} = \begin{cases} \lceil k \cdot \Phi \rceil, v_{test}^{nn} \in positive \\ k, \quad v_{test}^{nn} \in negative \end{cases} \tag{13}$$

In Eq. (13), if the nearest neighbor v_{test}^{nn} of v_{test} belongs to the minority class, then the purity of the component to which the minority vertex belongs is inversely proportional to the overlap degree of the region where the component is located. For the region with greater overlap degree, the test sample near it tends to favor the majority class when we apply the k-nearest neighbor rule. To overcome this undesirable propensity, we use the formula $k_{test} = \lceil k \cdot \Phi \rceil$, such that the test vertex near the minority component in regions with a high degree of overlap uses a smaller k value to mitigate the effects of data imbalance; otherwise, $k_{test} = k$. Fig. 6(a) gives an example of the adaptive selection of k values, where the pentagram represents the test sample; the nearest neighbor of the test sample is v_{min}^1; and the k values of the corresponding component is 4. Finally, the k value of the test vertex is $k_{test} = 2$, according to Eq. (13).

After fixing the value of k, it is vital to determine a more suitable set of vertices as the candidate set for calculating the gravitational force, and this study investigates two methods for selecting this candidate:

$$(I)\, X_{candi} = \bigcup_{\alpha=1}^{\mathcal{N}} \left\{C^{\alpha}(\mathrm{V}) \,\middle|\, \upsilon_i \in N_k^{test}, \upsilon_i \cap C^{\alpha}(\mathrm{V}) \neq \emptyset\right\} \tag{14}$$

$$(II)\, X_{candi} = \left\{\upsilon_i \,\middle|\, \upsilon_i \in N_k^{test}\right\} \tag{15}$$

In Eq. (14), where $\mathcal{N}$ is the total number of components, N_k^{test} denotes the k-nearest neighbors of the test vertex, and $C^{\alpha}(\mathrm{V})$ denotes all the vertices in a component. Thus, it selects all vertices of the associated subgraphs corresponding to the k-nearest neighbors of the test vertex as candidate sets; this calculation of gravitational force is called "sub-graph level" prediction. In other words, this method first finds the k-nearest neighbors of

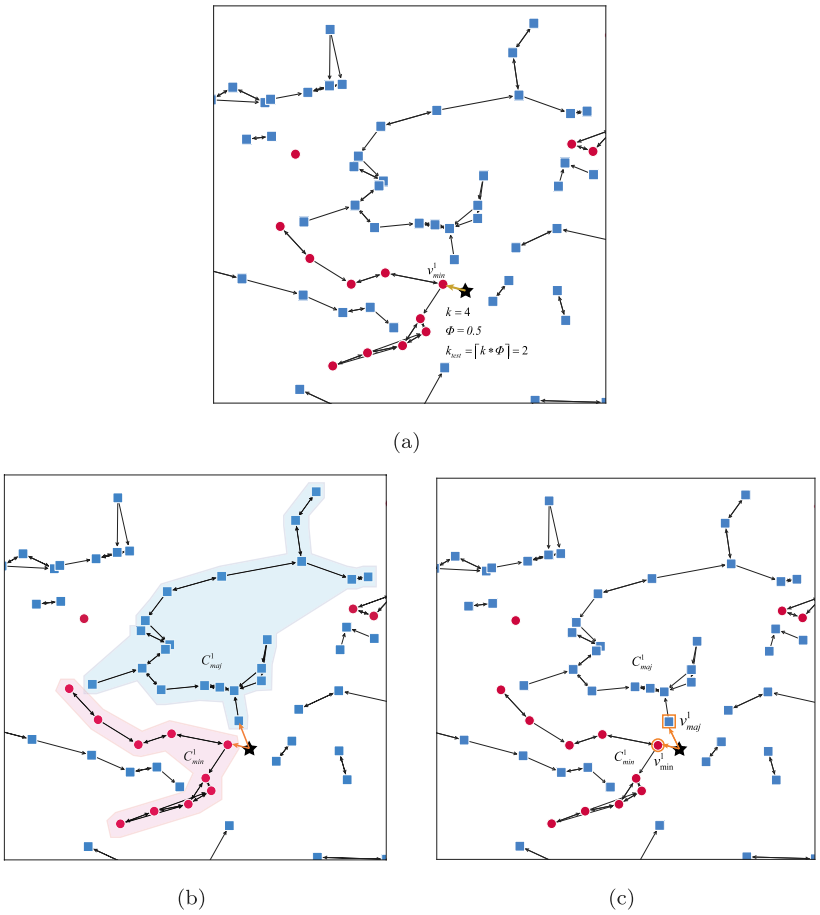


Fig. 6. (a) Calculating the nearest neighbor k value of the test vertex, (b) Rules for gravitational classification at the sub-graph level, (c) Rules for gravitational classification at the neighbor vertices level.

the test vertex; if any of its neighbors belong to a subgraph, then all the vertices in the subgraph are included in the candidate set for the calculation of gravitational force. Eq. (15) only selects the k-nearest neighbors of the test vertex as candidate sets, and its calculation of the gravitational force is called "neighbor vertex level" prediction.

To describe the process clearly, the two strategies for selecting the gravitational candidate sets proposed in this study are illustrated in Figs. 6(b) and 6(c). The gravitational force calculation at the sub-graph level is shown in Fig. 6(b); since $k_{test} = 2$, the two nearest neighbors of the test vertex are within two different components, C^1_{min}, C^1_{maj}, respectively. The candidate sets for the gravitational force calculation include all vertices in these two subgraphs (i.e., all the vertices in C^1_{min}, C^1_{maj}); similarly, the two nearest neighbors v^1_{min} and v^1_{maj} are adopted to calculate the gravitational force at the neighbor vertex level as shown in Fig. 6(c).

3.3.2. Gravity-based classification principles

The gravitational forces between the testing vertex and the corresponding candidate sets are calculated as follows:

$$F(x_{test}) = \sum_{x_i \in X_{candi}, y_i \in \{1,-1\}} y_i D(x_{test}, x_i), \tag{16}$$

where X_{candi} is the candidate set determined by x_{test}, y_i is the label of x_i, and $D(\cdot)$ denotes the gravity of the training samples x_{test} and x_i in the candidate set.

$$D(x_{test}, x_i) = G\frac{m_{x_{test}} m_{x_i}}{d(x_{test}, x_i)^2}, \tag{17}$$

where G is the gravitational constant and $m_{x_{test}}$ and m_{x_i} are the masses of samples x_{test} and x_i, respectively; in this case, the mass of x_{test} is set as 1 because it does not play a role in the classification and $d(x_{test}, x_i)$ denotes the Euclidean distance. For class-imbalanced scenarios, the samples in the candidate set must satisfy the following equation: $m_P n_P = m_N n_N$ [28], where n_P and n_N are the numbers of positive and negative samples, respectively; for simplicity, let $m_N = 1$, and then $m_P = \frac{n_N}{n_P} = IR$, thus the mass of x_i is defined as follows:

$$m_{x_i} = \begin{cases} IR, x_i \in X_{pos} \cap X_{candi} \\ 1, \quad x_i \in X_{\text{neg}} \cap X_{candi} \end{cases}, \tag{18}$$

With the two categories of candidates defined in Eqs. (14) and (15). There exist two gravity-based classification rules: sub-graph level prediction and vertex level prediction. At the sub-graph level, the entire component comprises the data particle for gravity calculation, and differs from the general gravity-based rule that uses a geometrical neighborhood relationship in a group of data elements; this method considers the topology relationship, instead. For the vertex level, it considers that either the size or the distribution characteristic of subgraph can be extreme in some scenarios; this study proposes a vertex-level gravity-based classification rule based on Eq. (15), where the set used for gravity calculation is the k_{test} nearest neighbors of the test sample, which only considers the effect of the local pattern on the classification results. Ultimately, the prediction of the test vertex is determined by comparing the summation of the gravitational force with different classes. As this study includes binary classification, the

Table 1
Specification of the data sets from *AUC*.

Datasets	Instances	Features	Minority name	IR	Abbreviation
Breast-tissue	106	10	carfad	1.9	Brt
Breast-w	699	10	Malignant	1.9	Brw
Seed	210	8	1	2	See
VC	310	7	Normal	2.1	Vc
Glass	214	10	1	2.1	Gla
German	1000	25	2	2.3	Ger
ILPD	583	11	2	2.5	Ilp
Haberman	306	4	2	2.8	Hab
Parkinsons	195	24	0	3.1	Par
Transfusion	748	5	1	3.2	Tra
Wpbc	198	34	recur	3.2	Wpb
Vehicle	846	19	van	3.3	Veh
CMC	1473	10	2	3.4	Cmc
Hepatitis	155	20	1	3.8	Hep
Libra	360	91	1211	4	Lib
Yeast	1484	9	MIT	5.1	Yea
Ecoli	336	8	pp	5.5	Eco
Fertility	100	10	O	7.3	Fer
Page-block	5473	11	other:1	8.8	Pag
Optdigits	5620	65	8	9.1	Opt
Satimage	6435	37	4	9.3	Sat
Abalone	4177	11	7	9.7	Aba
Climate	540	19	0	10.7	Cli
PhishingData	1353	9	0	12.1	Phi
Movement_libras	360	91	1	14	Mov
Seisimic-bumps	2584	19	1	14.2	Sei
Wilt	4839	6	w	17.5	Wil
Winequality	6497	12	≤ 4	25.4	Win
Ozone-Level	2536	73	1	33.7	Ozo
Parkinsons_updrs	5875	22	4	41.6	Par

Table 2
Confusion matrix for binary classification tasks.

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

prediction of the test sample is as follows:

$$label\,(x_{test}) = \begin{cases} 1, & F(x_{test}) > 0 \\ -1, & otherwise \end{cases}. \tag{19}$$

4. Experiment

4.1. Experiment setting

In this section, thirty real-world datasets from the UCI machine-learning repository were selected to validate the performance of our method. Table 1 gives the detailed information on the dataset, including the number of samples (**Instances**), the number of features (**Features**), the imbalance rate (**IR**), and the abbreviation of the dataset name (**Abbreviation**). Also among these datasets, some are converted from the multi-class to binary class by means of the One-vs-Rest (OvR) strategy, and the information in those classes set as minority classes (**Minority name**) are also included in Table 1 (e.g. in "**winequalit**", the classes with car and fad are considered as minority and the rest of the class fall into the majority class).

As traditional accuracy measurements are not suitable for the comparison of classifiers in imbalanced scenarios, this study employs three commonly used metrics for the comparison of imbalanced data [30]: *G-mean*, $F1$, and *AUC*. These are all calculated based on the confusion matrix, as shown in Table 2. Specific definitions of these evaluation metrics can be found in [20]; for the sake of brevity, we do not enumerate them here.

To verify the reliability of the proposed model in comparison to the traditional graph model, three groups of comparative experiments are conducted in this study. To obtain the average performance on each dataset, five runs of 5-fold cross-validation are employed.

Table 3
Abbreviations for the four variations of our proposed framework.

	$X_{candi}\,(V_C)$	$X_{candi}\left(N_k^{test}\right)$
$min\,(S)$	IMKOG $(Gk-I)$	IMKOG $(Gkv-I)$
$max\,(S)$	IMKOG $(Gk-II)$	IMKOG $(Gkv-II)$

(1) Experiment A: Comparison of the four methods proposed in this study.

(2) Experiment B: Validation of the graph construction rules and gravitational classification method based on the NNG in this study.

(3) Experiment C: Comparison between our proposal and other classifiers.

4.2. Experiment A: Comparison of the four methods in this study

Two graph construction strategies and two gravity-based classification methods are proposed in this study. Specifically, two types of thresholds are employed in the former iterative process according to the sub-graph strength for detecting the candidate set: $(min\,(S))$ and $(max\,(S))$. Two gravity calculation methods: the subgraph level gravity calculation $X_{candi}\,(V_C)$ and the neighbor vertex level gravity calculation $X_{candi}\left(N_k^{test}\right)$. Therefore, four methods are proposed in total, as shown in Table 3:

For simplicity, we refer to the four methods in Table 3 as *Gk-I*, *Gkv-I*, *Gk-II*, and *Gkv-II*, and it is worth noting that the methods mentioned in this study are non-parameters.

Table 4 reports the result of *AUC* and the average rank of the four methods proposed in this paper on thirty datasets, where the best performance is represented in boldface text. As demonstrated, either the mean *AUC* or the average rank shows that *Gkv-II* is better. A possible reason for this may be that *Gkv-II* cleans more difficult negative samples–it can detect more negative neighbors despite the increased learning difficulty in overlapping regions–and when the proportion of minority samples is relatively small, this structural topology-based strategy works better to connect more minority vertices and improve the final accuracy of the minority class.

Next, we further verify whether there is a significant difference between the four methods in terms of their performance on those datasets and analyze which of them performs the best. Because there are fewer than five methods for comparison, the number of algorithms for comparison is small. In such cases, as with the test adopted in [31], the Friedman aligned-rank test [32] is used in this section to compare the four methods, and the results are shown in Fig. 7. It is demonstrated that the average aligned ranks of *Gkv-II* are lower than others, indicating that it achieves the best performance, and the corresponding p-value is 0.0010, which shows a significant difference among them. Then, the Holm post hoc test is conducted to compare the control method with the other three methods, as shown in Table 5. These tests were conducted in Python-3.7 using the Python library in [33]. We can see that the smaller adjusted p-values provide strong evidence against the null hypothesis, which indicates that *Gkv-II* is significantly better than the other methods.

4.3. Experiment B: Verifying the effectiveness of the proposed methodology

To investigate the effectiveness of the proposed strategies involved in the graph construction and testing phases, a sequence

Table 4
Performance results of the four methods on *AUC*.

Datasets	Gk-I	Gkv-I	Gk-II	Gkv-II	Datasets	Gk-I	Gkv-I	Gk-II	Gkv-II
Brt	0.7852	**0.8051**	0.7521	0.7976	Yea	0.7050	0.6836	0.7587	**0.7680**
Brw	0.9437	0.9488	0.9510	**0.9529**	Eco	0.9107	0.8985	**0.9268**	0.9203
See	0.9057	0.9057	**0.9150**	0.9107	Fer	**0.6007**	0.5980	0.5567	0.5686
Vc	0.7645	0.7402	0.7662	**0.7822**	Pag	0.8088	0.8066	0.8582	**0.8586**
Gla	0.7902	0.7862	0.7802	**0.8182**	Opt	**0.9799**	0.9785	0.9652	0.9572
Ger	**0.6268**	0.6159	0.5690	0.6160	Sat	0.7505	0.7316	0.7685	**0.7708**
Ilp	0.5745	0.5707	**0.6480**	0.6273	Aba	0.5655	0.5442	0.7476	**0.7497**
Hab	**0.5650**	0.5066	0.5410	0.5280	Cli	**0.6064**	0.6053	0.5637	0.5997
Par	0.9346	**0.9374**	0.9233	0.9259	Phi	0.5642	**0.5825**	0.5724	0.5773
Tra	0.6150	0.6028	0.6183	**0.6219**	Mov	0.8391	0.8554	0.8325	**0.8562**
Wpb	0.6106	0.5743	0.5519	**0.6210**	Sei	0.5381	0.5365	0.5767	**0.5845**
Veh	0.8804	0.8784	0.8717	**0.8908**	Wil	0.6307	0.6039	0.6231	**0.6395**
Cmc	0.5718	0.5644	0.6199	**0.6260**	Win	0.5803	0.5750	0.5728	**0.5865**
Hep	0.6744	0.6817	0.6981	**0.7085**	Ozo	0.5767	0.5874	0.6000	**0.6210**
Lib	**0.8983**	0.8947	0.8865	0.8879	Par	0.9772	**0.9837**	0.9770	0.9814
Average	0.7258	0.7195	0.7331	**0.7451**	AveRank	2.55	2.98	2.80	**1.67**

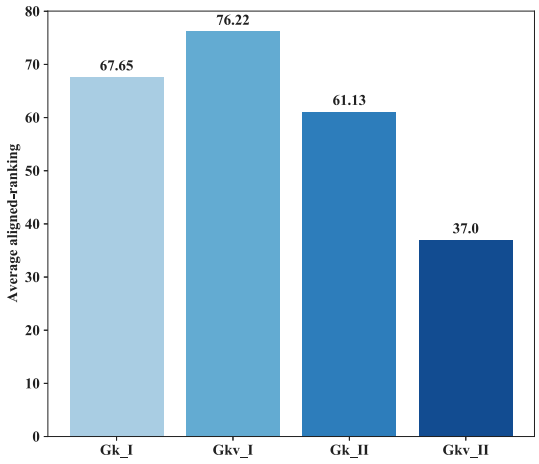


Fig. 7. Friedman aligned-ranks of the four methods on thirty datasets.

Table 5
Holm test result for the comparison among the four methods.

Control method: Gkv-II			
Comparison	Z	Adjusted *p*-value	Hypothesis(= 0.01)
Gk-I	3.412	0.00129	Rejected
Gk-II	2.687	0.00721	Rejected
Gkv-I	4.366	0.00004	Rejected

of comparison experiments is organized in this section, which consists of two parts:

(*a*) *Gk-I*, *Gkv-I*, *Gk-II*, *Gkv-II* vs. $KAOG_C$, $KAOG_V$, $MKAOG_C$, $MKAOG_V$

In the first part, we compare our methods with KAOG and MKAOG to validate the effectiveness of the graph construction methods proposed in this paper. For fairness, the KAOG and MKAOG in the comparative group are modified in the testing phase in accordance with our own methods. $KAOG_C$, $KAOG_V$, $MKAOG_C$, and $MKAOG_V$ denote that the testing rules of KAOG and MKAOG are modified into the two gravity-based classification rules, respectively. The corresponding subscripts *C* and *V* represent the subgraph-based and neighbor-vertex-based gravity classification rules, respectively. It should be noted that the *k* values of the four methods in the testing stage are still consistent with KAOG and MKAOG, i.e., in KAOG, the *k* value is determined by the maximum *k* among the components in the optimal graph, whereas in MKAOG, it is based on the number of classes because it is binary classification in this work with $k = 2$.

(*b*) *Gk-I*, *Gkv-I*, *Gk-II*, *Gkv-II* vs. *I* (*Bayes*) , *II* (*Bayes*)

In this section, the effectiveness of the classification methods is verified by comparing it with the Bayes-based rule in [14]. Therefore, we modify the testing phase of our methods into a Bayes-based method, where *I* (*Bayes*) and *II* (*Bayes*) represent the replacement of the classification rules of the two *k*-NNG models proposed in this paper with the Bayesian rule-based classification method in KAOG. Meanwhile, the setting of value *k* is the same as that of the KAOG.

As shown in Table 6, to ensure fairness, experiment B(a) was limited to comparisons between methods with the same classification rules; i.e., *Gk-I*, *Gk-II* is compared with $KAOG_C$ and $MKAOG_C$, using the same rule of classification. The results show the average ranking of the former (3.57, 4.07) is higher than that of the latter (5.90, 4.83). Similar results occur for the nearest-neighbor vertex-based gravity classification method (i.e., *Gkv-I*, *Gkv-II* vs. $KAOG_V$, $MKAOG_V$). In particular, when we encountered datasets with relatively high imbalance rates, such as "winequality" and "Ozone-Level", the proposed methods improved the *AUC* significantly compared to KAOG and MKAOG. The study demonstrates that the two methods are more reliable than the traditional nearest-neighbor models such as KAOG and MKAOG in imbalanced scenarios.

Table 7 shows that the average rank of *Gkv-II* in experiment B(b) on *AUC* is highest when compared with the other methods. More specifically, for fairness, we compare *Gk-I*, *Gkv-I* with *I* (*Bayes*), and *Gk-II*, *Gkv-II* with *II* (*Bayes*). The results of our methods are consistently better than Bayes-based methods in terms of average rank on *AUC*, which indicates that the gravity classification rules based on *k*-NNG are superior than the Bayes-based methods in imbalanced scenarios. We hypothesize that this result may be because KAOG struggles with learning from imbalanced data, even though KAOG displays a remarkable performance which benefits from the local structure information, as well as the global statistical properties of the constructed graph. Specifically, the KAOG uses the number of classes as the value of *k*, which determines the number of connections with test vertices during the testing phase. When confronted with a majority class dominated scenario, the Bayes rule is highly likely to favor the majority class, which is also a challenge for the selection of *k* values for the *k*-NN-based method in imbalanced scenarios. Conversely, the classification rules proposed in this paper, especially *Gkv-II*, exhibits a significant improvement in this case.

4.4. Experiment C: Comparison with other methods

To verify the effectiveness of the algorithm, the comparison experiments also employ five classifiers written in Python using

Table 6
Comparison of results Experiment B(a) on *AUC* metric. Bold represents the highest scores among the algorithms for each corresponding dataset.

Datasets	Gk-I	Gkv-I	Gk-II	Gkv-II	$KAOG_C$	$KAOG_V$	$MKAOG_C$	$MKAOG_V$
Brt	0.7852	**0.8051**	0.7521	0.7976	0.7682	0.7916	0.7605	0.7782
Brw	0.9437	0.9488	0.9510	**0.9529**	0.9481	0.9472	0.9519	0.9517
See	0.9057	0.9057	**0.9150**	0.9107	0.8979	0.8871	0.9029	0.9093
Vc	0.7645	0.7402	0.7662	**0.7822**	0.7337	0.7388	0.7413	0.7352
Gla	0.7902	0.7862	0.7802	**0.8182**	0.7852	0.7782	0.7952	0.7910
Ger	**0.6268**	0.6159	0.5690	0.6160	0.6189	0.6115	0.6144	0.6091
Ilp	0.5745	0.5707	**0.6480**	0.6273	0.5233	0.5216	0.5344	0.5473
Hab	**0.5650**	0.5066	0.5410	0.5280	0.5312	0.5364	0.5119	0.5069
Par	0.9346	**0.9374**	0.9233	0.9259	0.9165	0.9201	0.9137	0.9153
Tra	0.6150	0.6028	0.6183	**0.6219**	0.5958	0.5981	0.5473	0.5428
Wpb	0.6106	0.5743	0.5519	**0.6210**	0.5849	0.5781	0.5808	0.5847
Veh	0.8804	0.8784	0.8717	**0.8908**	0.8757	0.8759	0.8798	0.8830
Cmc	0.5718	0.5644	0.6199	**0.626**	0.5635	0.5560	0.5318	0.5364
Hep	0.6744	0.6817	0.6981	0.7085	0.7048	0.6677	0.6893	**0.7151**
Lib	**0.8983**	0.8947	0.8865	0.8879	0.8776	0.8670	0.8770	0.8848
Yea	0.7050	0.6836	0.7587	**0.7680**	0.7114	0.7105	0.7166	0.7178
Eco	0.9107	0.8985	**0.9268**	0.9203	0.9165	0.9082	0.9139	0.9010
Fer	0.6007	0.5980	0.5567	0.5686	0.5476	0.5585	**0.6165**	0.5554
Pag	0.8088	0.8066	0.8582	**0.8586**	0.8273	0.8217	0.8436	0.8404
Opt	0.9799	0.9785	0.9652	0.9572	0.9723	0.9725	0.9799	**0.9803**
Sat	0.7505	0.7316	0.7685	0.7708	0.8007	0.7960	0.8033	**0.8048**
Aba	0.5655	0.5442	0.7476	**0.7497**	0.5427	0.5507	0.5485	0.5437
Cli	**0.6064**	0.6053	0.5637	0.5997	0.5491	0.5429	0.5647	0.5699
Phi	0.5642	**0.5825**	0.5724	0.5773	0.5064	0.5048	0.5089	0.5041
Mov	0.8391	**0.8554**	0.8325	0.8562	0.8120	0.8264	0.8154	0.8417
Sei	0.5381	0.5365	0.5767	**0.5845**	0.5236	0.5248	0.5240	0.5234
Wil	0.6307	0.6039	0.6231	**0.6395**	0.5479	0.5500	0.5655	0.5623
Win	0.5803	0.5750	0.5728	**0.5865**	0.5339	0.5328	0.5328	0.5371
Ozo	0.5767	0.5874	0.6000	**0.6210**	0.5172	0.5121	0.5315	0.5314
Par	0.9772	**0.9837**	0.9770	0.9814	0.9693	0.9735	0.9833	0.9804
Average	0.7258	0.7195	0.7331	**0.7451**	0.7068	0.7054	0.7094	0.7095
AveRank	3.57	4.33	4.07	**2.13**	5.90	6.13	4.83	4.93

Table 7
Comparison of the results of experiment B(b) on *AUC* metric.

Datasets	Gk-I	Gkv-I	Gk-II	Gkv-II	II-Bayes	I-Bayes
Brt	0.7852	**0.8051**	0.7521	0.7976	0.7333	0.7909
Brw	0.9437	0.9488	0.9510	**0.9529**	0.9493	0.9462
See	0.9057	0.9057	**0.9150**	0.9107	0.8936	0.8886
Vc	0.7645	0.7402	0.7662	**0.7822**	0.7791	0.7126
Gla	0.7902	0.7862	0.7802	**0.8182**	0.7776	0.7757
Ger	**0.6268**	0.6159	0.5690	0.6160	0.6141	0.5543
Ilp	0.5745	0.5707	**0.6480**	0.6273	0.6141	0.5277
Hab	**0.5650**	0.5066	0.5410	0.5280	0.5644	0.5108
Par	0.9346	**0.9374**	0.9233	0.9259	0.9235	0.9365
Tra	0.6150	0.6028	0.6183	0.6219	**0.6289**	0.5667
Wpb	0.6106	0.5743	0.5519	**0.6210**	0.5722	0.5430
Veh	0.8804	0.8784	0.8717	0.8908	0.8641	**0.8958**
Cmc	0.5718	0.5644	0.6199	**0.6260**	0.6070	0.5255
Hep	0.6744	0.6817	0.6981	**0.7085**	0.6962	0.6636
Lib	**0.8983**	0.8947	0.8865	0.8879	0.8890	0.8761
Yea	0.7050	0.6836	0.7587	**0.7680**	0.7653	0.6786
Eco	0.9107	0.8985	**0.9268**	0.9203	0.9155	0.9254
Fer	**0.6007**	0.5980	0.5567	0.5686	0.5439	0.5618
Pag	0.8088	0.8066	0.8582	**0.8586**	0.8152	0.7803
Opt	**0.9799**	0.9785	0.9652	0.9572	0.9546	0.9737
Sat	0.7505	0.7316	0.7685	**0.7708**	0.7589	0.7591
Aba	0.5655	0.5442	0.7476	**0.7497**	0.7240	0.5026
Cli	**0.6064**	0.6053	0.5637	0.5997	0.5044	0.5212
Phi	0.5642	**0.5825**	0.5724	0.5773	0.5377	0.5188
Mov	0.8391	0.8554	0.8325	**0.8562**	0.8099	0.7987
Sei	0.5381	0.5365	0.5767	**0.5845**	0.5847	0.5035
Wil	0.6307	0.6039	0.6231	**0.6395**	0.5552	0.5180
Win	0.5803	0.5750	0.5728	**0.5865**	0.5551	0.5008
Ozo	0.5767	0.5874	0.6000	**0.6210**	0.5266	0.5174
Par	0.9772	**0.9837**	0.9770	0.9814	0.9819	0.9777
Average	0.7258	0.7195	0.7331	**0.7451**	0.7213	0.6917
AveRank	3.20	3.53	3.33	**1.97**	3.90	5.03

the classical machine learning library scikit-learn. Similar to that in the literature [14,15], the comparison algorithm in this section uses the classical classifier SVM, the nearest neighbor classifier KNN, and the classic tree model-based classifier DT. Among these, the kernel function of the SVM uses the radial basis function; the KNN k value is set to the commonly used value of 5. In addition, LDA, as a non-parametric classifier, was chosen for this experiment. The gravitation-based methods, GFRNN, as another contender approach, is also included. In addition, five large real-datasets are used to further support our experiment results; among them, the dataset "**Readmission-Analysis**" is collected by a researcher from Columbia University [34] and the rest of four data sets can be found in UCI repository. Details of the five datasets are shown in Table 8.

Table 8
Details of the large datasets.

Name	Instances	Features	Minority	IR	Abbreviation
Readmission_Analysis	7000	30	Positive	5.1	Ra
Musk	6598	168	1(musk)	5.5	Mus
ticdata2000	9000	86	1	15.7	Tic
Dry_Bean_Dataset	13610	16	Bombay	25.1	Dbd
Crowdsourced_Mapping	10544	28	water	51.7	Cro

Table 9 shows that GFRNN achieves the best overall performance on *AUC*, while our *Gkv-II* obtained the second-best average *AUC*. The study shows that GFRNN performs better than our proposed methods by more than 10% on six datasets: "Ger", "Hep", "Fer", "Cli", "Mov" and "Ozo". Meanwhile, our proposal performed better than GFRNN on five datasets: "Gla", "Veh", "Opt", "Par" and "Mus". As GFRNN calculates the gravitational forces for the test sample using the samples within a fixed radius, the difference illustrates that both GFRNN and our proposal have distinct advantages in some specific scenarios. To provide a clearer analysis of the performance results in Table 9, we report the ranking of the compared methods in different datasets in Table 10. One can observe that the *Gkv-II* ranks highly (ranking smaller than 6) on most data sets, which implies that it is more

Table 9
Comparison of the results of all algorithms on *AUC*.

Datasets	Gk_I	Gkv_I	Gk_II	Gkv_II	KAOG	MKAOG	GFRNN	LDA	SVM	DT	KNN
Brt	0.7852	**0.8051**	0.7521	0.7976	0.7713	0.7933	0.7971	0.7468	0.7125	0.7872	0.7743
Brw	0.9437	0.9488	0.9510	**0.9529**	0.9464	0.9453	0.9150	0.9396	0.9498	0.9116	0.9489
See	0.9057	0.9057	0.9150	0.9107	0.8793	0.9086	0.9279	**0.9693**	0.8957	0.8664	0.9129
Vc	0.7645	0.7402	0.7662	0.7822	0.7273	0.7598	0.7518	**0.7990**	0.7774	0.7597	0.7508
Gla	0.7902	0.7862	0.7802	**0.8182**	0.7683	0.7869	0.7395	0.7106	0.6551	0.7877	0.7726
Ger	0.6268	0.6159	0.5690	0.6160	0.6187	0.6179	**0.7116**	0.6906	0.6904	0.6340	0.6101
Ilp	0.5745	0.5707	**0.6480**	0.6273	0.5331	0.5348	0.6274	0.5200	0.5149	0.5849	0.5412
Hab	0.5650	0.5066	0.5410	0.5280	0.5189	0.5280	**0.5890**	0.5563	0.5460	0.5597	0.5097
Par	0.9346	**0.9374**	0.9233	0.9259	0.9158	0.9280	0.9005	0.8273	0.7496	0.9223	0.9167
Tra	0.6150	0.6028	0.6183	**0.6219**	0.6031	0.5589	0.5925	0.5526	0.5501	0.5278	0.5999
Wpb	0.6106	0.5743	0.5519	0.6210	0.5837	0.5987	0.6247	**0.6929**	0.6633	0.5819	0.6091
Veh	0.8804	0.8784	0.8717	0.8908	0.8711	0.8812	0.8093	0.9349	**0.9490**	0.9162	0.9077
Cmc	0.5718	0.5644	0.6199	**0.6260**	0.5545	0.5367	0.6017	0.5362	0.5159	0.5525	0.5727
Hep	0.6744	0.6817	0.6981	0.7085	0.6869	0.6914	**0.7934**	0.7510	0.7421	0.6417	0.7083
Lib	**0.8983**	0.8947	0.8865	0.8879	0.8712	0.8820	0.8741	0.7068	0.7307	0.7800	0.8567
Yea	0.7050	0.6836	0.7587	0.7680	0.7156	0.7115	**0.7845**	0.6949	0.6571	0.6963	0.7364
Eco	0.9107	0.8985	0.9268	0.9203	0.8988	0.9062	0.8895	0.8009	0.7927	0.8447	**0.9323**
Fer	0.6007	0.5980	0.5567	0.5686	0.5621	0.6008	**0.6557**	0.4886	0.4978	0.5508	0.5086
Pag	0.8088	0.8066	0.8582	0.8586	0.8251	0.8453	0.8742	0.7728	0.7299	**0.9017**	0.8356
Opt	0.9799	0.9785	0.9652	0.9572	0.9730	0.9816	0.6585	0.8977	0.8906	0.8907	**0.9825**
Sat	0.7505	0.7316	0.7685	0.7708	0.7986	0.8022	0.7924	0.5033	0.5006	0.7484	**0.8207**
Aba	0.5655	0.5442	0.7476	0.7497	0.5438	0.5466	**0.7912**	0.4998	0.5000	0.5976	0.5700
Cli	0.6064	0.6053	0.5637	0.5997	0.5444	0.5583	**0.8159**	0.7308	0.7959	0.6799	0.6109
Phi	0.5642	0.5825	0.5724	0.5773	0.5103	0.5047	0.5789	0.5000	0.5000	**0.6868**	0.5303
Mov	0.8391	0.8554	0.8325	0.8562	0.7987	0.8557	**0.9581**	0.8199	0.8397	0.7242	0.7657
Sei	0.5381	0.5365	0.5767	0.5845	0.5216	0.5242	**0.6778**	0.5575	0.4996	0.5585	0.5330
Wil	0.6307	0.6039	0.6231	0.6395	0.5538	0.5623	0.7054	0.5614	0.5000	**0.8948**	0.5786
Win	0.5803	0.5750	0.5728	0.5865	0.5325	0.5353	**0.6611**	0.5260	0.5000	0.5832	0.5257
Ozo	0.5767	0.5874	0.6000	0.6210	0.5179	0.5221	**0.7441**	0.5347	0.5095	0.6076	0.5457
Par	0.9772	**0.9837**	0.9770	0.9814	0.9701	0.9799	0.7443	0.5063	0.5714	0.9741	0.9835
Ra	0.8357	0.8270	0.8074	**0.9019**	0.8519	0.8820	0.8621	0.803	0.7955	0.7776	0.8956
Mus	**0.9151**	0.9115	0.9124	0.9085	0.9069	0.9092	0.8028	0.856	0.8666	0.8911	0.9035
Tic	0.5212	0.5141	0.5873	0.5914	0.5007	0.5034	**0.6132**	0.5174	0.5011	0.5408	0.5073
Dbd	**1.0000**	**1.0000**	0.9997	0.9997	**1.0000**	**1.0000**	0.9995	0.9971	**1.0000**	0.9984	**1.0000**
Cro	0.8948	0.9005	0.8915	0.8760	0.8860	0.8965	0.8656	0.8361	0.8022	**0.9020**	0.8982
Average	0.7412	0.7353	0.7483	0.7609	0.7218	0.7308	**0.7637**	0.6954	0.6826	0.7389	0.7330

stable; conversely, GFRNN performs worse on several datasets such as "Brw", "Gla", "Veh", "Opt" and "Mus".

4.5. Statistical tests

To assess whether there is a significant difference between the compared algorithms, this section uses a non-parametric Friedman test, and the corresponding post-hoc test is performed to further detect the difference in performance among the algorithms.

Table 11 shows the Friedman statistic χ_F^2 with $(k-1)$ degrees of freedom and the F-distribution with k-1 and $(k-1)(N-1)$ degrees of freedom for the three comparison experiments, where k denotes the number of comparison algorithms and N denotes the number of datasets. Experiment B(a): $\chi_F^2(7)$: 14.0671, $F_F(7,203)$: 2.0549; Experiment B(b): $\chi_F^2(5)$: 11.0705, $F_F(5,145)$: 2.2766; Experiment C: $\chi_F^2(10)$: 18.3070, $F_F(10, 340)$: 1.86, when the statistical value is greater than the corresponding critical value, then the null hypothesis is rejected and it becomes apparent that all comparison algorithms are significantly different.

Fig. 8 shows the Friedman test at a significant level of $\alpha = 0.05$. The top line is the critical range CD of the corresponding average ranking and the numbers on the axes represent the average ranking of the algorithms, with the rightmost end of the axes being the lowest (best) ordinal value, while the algorithms connected by the same line means that there is no significant difference between them. Fig. 8(a) shows that in experiment B(a), *Gkv-II* obtains the best ranking, followed by *Gk-I*, *Gk-II* and *Gkv-I*, and it is significantly different from the other methods. Fig. 8(b) shows that our methods are significantly different from the Bayes-based methods which, to some extent, validates the effectiveness of this paper's graph gravity-based classification method in imbalanced scenarios.

The comparison of significance in Fig. 8(c) shows that the first ranked method is *Gkv-II*, which is not significantly different from the *GRFNN*, *Gk-I*, *Gk-II*, KNN, *Gkv-I* and MKAOG methods but is significantly different from the other four methods. It should be indicated that comparisons between our four proposals reveal that *Gkv-I* performed the worst overall. This may be because the composition method based on the maximum edge strength threshold is too conservative in dealing with complex distribution scenarios, and it fails to identify the majority of vertices that affect the graph construction of the minority class. Moreover, it further hinders the formation of minority class subgraphs at a relatively large scale.

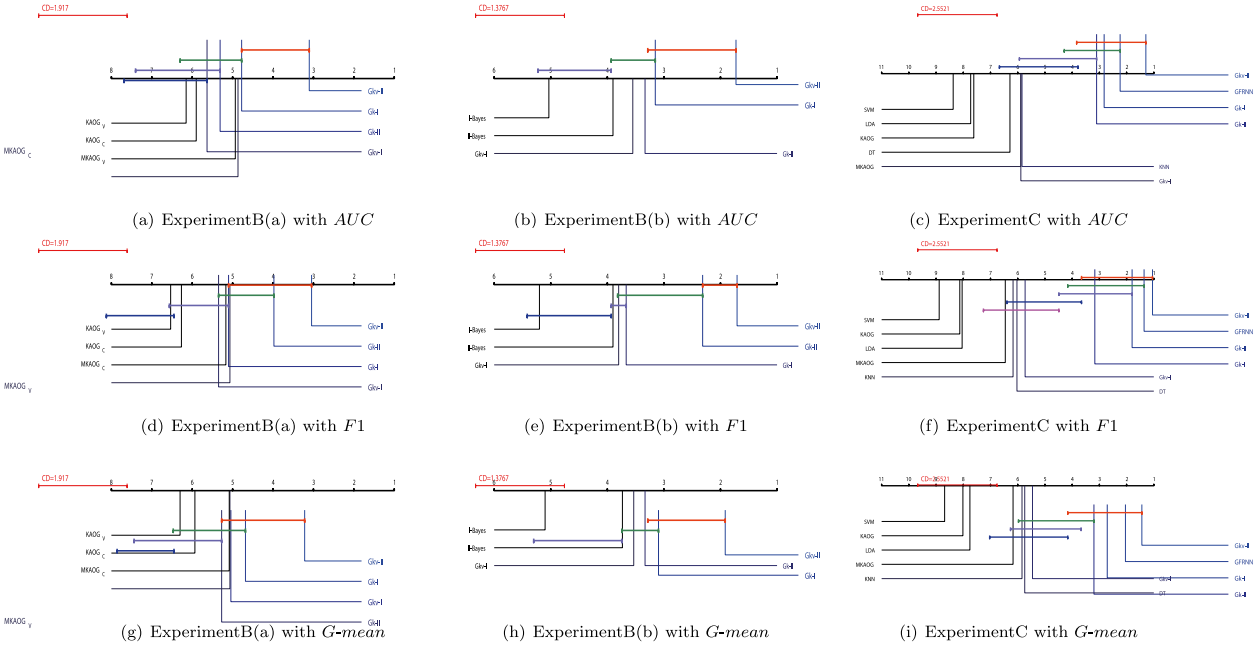
For clarity, the Friedman test results of the experiments on $F1$ and *G-mean* are shown in Fig. 8(d)–8(i); similarly, the study shows that the results on $F1$ and *G-mean* are substantially consistent with those on *AUC*, but they are not enumerated here.

4.6. Visualization comparison of nearest neighbor graphs

To better understand our IMKOG compared with the other two k-NNG-based classifiers, KAOG and MKOAG, in an intuitive manner, visualization results of the classification decision domains on two-dimensional simulated data were processed and compared. As is shown in Fig. 9, the data consists of four Gaussian models with different scales generated by the Python library scikit-learn, where the red solid circles represent the minority class samples and the blue squares represent the majority class samples. The figure shows that the minority class consists of three clusters, located at the top right, middle, and bottom left of the figure. It also

Table 10
The ranks of all algorithms in experiment C on *AUC*.

Datasets	Gk_I	Gkv_I	Gk_II	Gkv_II	KAOG	MKAOG	GFRNN	LDA	SVM	DT	KNN
Brt	6	1	9	2	8	4	3	10	11	5	7
Brw	8	5	2	1	6	7	10	9	3	11	4
See	7	7	3	5	10	6	2	1	9	11	4
Vc	5	10	4	2	11	6	8	1	3	7	9
Gla	2	5	6	1	8	4	9	10	11	3	7
Ger	5	9	11	8	6	7	1	2	3	4	10
Ilp	5	6	1	3	9	8	2	10	11	4	7
Hab	2	11	6	7	9	7	1	4	5	3	10
Par	2	1	5	4	8	3	9	10	11	6	7
Tra	3	5	2	1	4	8	7	9	10	11	6
Wpb	5	10	11	4	8	7	3	1	2	9	6
Veh	7	8	9	5	10	6	11	2	1	3	4
Cmc	5	6	2	1	7	9	3	10	11	8	4
Hep	10	9	6	4	8	7	1	2	3	11	5
Lib	1	2	4	3	7	5	6	11	10	9	8
Yea	7	10	3	2	5	6	1	9	11	8	4
Eco	4	7	2	3	6	5	8	10	11	9	1
Fer	3	4	7	5	6	2	1	11	10	8	9
Pag	8	9	4	3	7	5	2	10	11	1	6
Opt	3	4	6	7	5	2	11	8	10	9	1
Sat	7	9	6	5	3	2	4	10	11	8	1
Aba	6	8	3	2	9	7	1	11	10	4	5
Cli	6	7	9	8	11	10	1	3	2	4	5
Phi	6	2	5	4	8	9	3	10	10	1	7
Mov	6	4	7	2	9	3	1	8	5	11	10
Sei	6	7	3	2	10	9	1	5	11	4	8
Wil	4	6	5	3	10	8	2	9	11	1	7
Win	4	5	6	2	8	7	1	9	11	3	10
Ozo	6	5	4	2	10	9	1	8	11	3	7
Par	5	1	6	3	8	4	9	11	10	7	2
Ra	6	7	8	1	5	3	4	9	10	11	2
Mus	1	3	2	5	6	4	11	10	9	8	7
Tic	5	7	3	2	11	9	1	6	10	4	8
Dbd	3.5	3.5	7.5	7.5	3.5	3.5	9	11	3.5	10	3.5
Cro	5	2	6	8	7	4	9	10	11	1	3
AverRank	4.99	5.87	5.24	3.64	7.61	5.87	4.49	7.71	8.36	6.29	5.84



(a) ExperimentB(a) with AUC (b) ExperimentB(b) with AUC (c) ExperimentC with AUC

(d) ExperimentB(a) with $F1$ (e) ExperimentB(b) with $F1$ (f) ExperimentC with $F1$

(g) ExperimentB(a) with $G\text{-}mean$ (h) ExperimentB(b) with $G\text{-}mean$ (i) ExperimentC with $G\text{-}mean$

Fig. 8. The Friedman test results of experiments B and C: (a) Comparison of *AUC* for validating the effectiveness of graph construction in experiment B(a) (corresponding to Table 6), (b) Comparison of *AUC* for validating the effectiveness of classification methods in experiment B(b) (corresponding to Table 7), (c) *AUC* comparison of our methods versus the other seven methods in experiment C (corresponding to Table 8), (d) Comparison of $F1$ for validating the effectiveness of graph construction in experiment B(a), (e) Comparison of $F1$ for validating the effectiveness of classification methods in experiment B(b), (f) Comparison of $F1$ for our methods versus the other seven methods in experiment C, (g) Comparison of *G-mean* for validating the effectiveness of graph construction in experiment B(a), (h) Comparison of *G-mean* for validating the effectiveness of classification methods in experiment B(b), (i) Comparison of *G-mean* for our methods versus the other seven methods in experiment C.

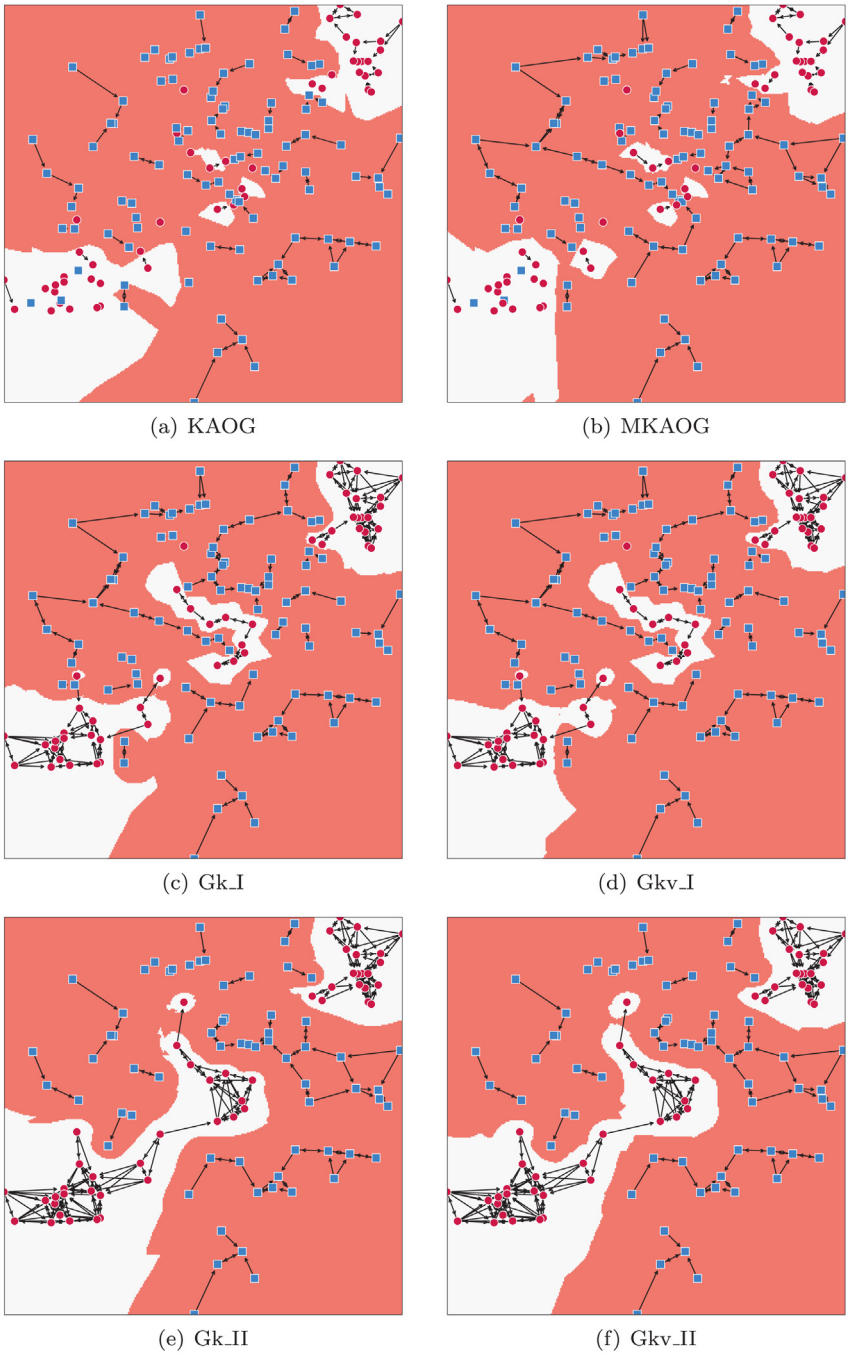


Fig. 9. Classification visualization results of six methods on artificial data, where (a) KAOG, (b) MKOAG, (c) The methods of graph construction with the maximum edge strength as threshold + gravity-based classification rule on subgraph level (*Gk-I*), (d) The methods of graph construction with maximum edge strength as threshold + gravity based classification rule on neighbor vertex level (*Gkv-I*), (e) The method of graph construction with average edge strength as threshold + gravity-based classification rule on subgraph level (*Gk-II*), (f) the method of graph construction with average edge strength as threshold + gravity-based classification rule on neighbor vertex level (*Gkv-II*).

shows that, when compared to KAOG and MKAOG, our four methods obtain a clearer class decision domain in the boundary region, which benefits from the removal of negative neighbors during the iterative graph construction process. Moreover, the minority vertices have a higher probability of being connected, whereas before, they were isolated. Therefore, our proposals strengthen the generation and broadening of the minority class neighbor structure in that local region.

As shown by the comparison in Fig. 9(c), 9(d) and Fig. 9(e), 9(f), two edge strength thresholds correspond to two NNG models. The threshold determines whether a majority vertex intruding into the nearest-neighbor structure of the minority class is removed. Thus, it determines the tolerance degree of the algorithms on highly overlapped majority vertices. The method of graph construction that applies the maximum edge strength as the threshold corresponds to a stricter constraint, and this strategy determines fewer samples to be removed. As shown in the figure, the minority class has a smaller range of influence compared to the models illustrated by Figs. 9(e) and 9(f), which apply the average strength as the threshold. A comparison between Fig. 9(d), 9(f) and Fig. 9(c), 9(e) shows the difference between the neighbor vertex-based gravity classification rule and the subgraph-based

Table 11
Friedman statistical test for the three groups of comparison experiments.

		Experiment B(a)	Experiment B(b)	Experiment C
AUC	χ_F^2	58.1306	42.6238	65.8805
	F_F	11.1002	11.5118	7.8838
F1	χ_F^2	84.8083	56.6048	105.6610
	F_F	19.6454	17.5763	14.7028
G-mean	χ_F^2	63.6139	39.619	76.9312
	F_F	12.6023	10.409	9.5788

gravity classification rule: the former only considers the k-nearest neighbors of test vertices for gravity calculation, indicating that it focuses on the local information of the vertices, while the latter calculates the vertices' gravity in the gravity formula from the whole associated subgraph of the test vertex (i.e., the set of particles for gravity calculation is all vertices of the associated subgraph). We find that the decision boundary is somewhat smoother for the gravity classification rule based on the subgraph structure.

5. Conclusion

In this study, we addressed the problem of constructing nearest-neighbor graphs in scenarios with imbalanced class distributions, which were gradually constructed by iterative posterior-based methods with simultaneous detection of negative vertices, and two different gravity-based classification rules were investigated for prediction with a NNG. Extensive experiments showed that the methods proposed in this paper achieved superior performance when compared to KAOG and MKAOG with respect to imbalanced data. As the graph-based model is characterized by its capability to capture data topology information, studying the sampling methods from the graph perspective is interesting and meaningful, which merits future efforts.

CRediT authorship contribution statement

Yuanting Yan: Conceptualization, Funding acquisition, Writing – review & editing. **Tianxiao Zhou:** Investigation, Methodology, Simulation experiment, Writing – original draft. **Zhong Zheng:** Data curation, Writing – review & editing. **Hao Ge:** Validation, Writing – review & editing. **Yiwen Zhang:** Funding acquisition, Review & editing. **Yanping Zhang:** Resources, Funding acquisition, Review & editing.

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.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 61806002 and 61872002, the National Key R&D Program of China (2019YFB1704101), the Higher Education Natural Science Foundation of Anhui Province (KJ2020ZD63), the Natural Science Foundation of Anhui Province (No. 2108085MF215), and the Doctoral Scientific Research Start-Up Foundation of Anhui University.

References

[1] M.G. Carneiro, L. Zhao, Organizational data classification based on the importance concept of complex networks, IEEE Trans. Neural Netw. Learn. Syst. 29 (8) (2017) 3361–3373.
[2] T.C. Silva, L. Zhao, Network-based high level data classification, IEEE Trans. Neural Netw. Learn. Syst. 23 (6) (2012) 954–970.
[3] T.C. Silva, L. Zhao, High-level pattern-based classification via tourist walks in networks, Inform. Sci. 294 (2015) 109–126.
[4] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Advances in Neural Information Processing Systems, 2004, pp. 321–328.
[5] I. Brugere, B. Gallagher, T.Y. Berger-Wolf, Network structure inference, a survey: Motivations, methods, and applications, ACM Comput. Surv. 51 (2) (2018) 1–39.
[6] P. Franti, O. Virmajoki, V. Hautamaki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (2006) 1875–1881.
[7] H. Li, X. Liu, T. Li, R. Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognit. 102 (2020) 107206.
[8] L.d.F. Costa, F.A. Rodrigues, G. Travieso, P.R. Villas Boas, Characterization of complex networks: A survey of measurements, Adv. Phys. 56 (1) (2007) 167–242.
[9] P. Zhou, L. Du, X. Li, Y.-D. Shen, Y. Qian, Unsupervised feature selection with adaptive multiple graph learning, Pattern Recognit. 105 (2020) 107375.
[10] B. Wang, F. Pan, K.-M. Hu, J.-C. Paul, Manifold-ranking based retrieval using k-regular nearest neighbor graph, Pattern Recognit. 45 (4) (2012) 1569–1577.
[11] Z. Zhang, J. Wang, H. Zha, Adaptive manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2011) 253–265.
[12] M. Deshpande, M. Kuramochi, N. Wale, G. Karypis, Frequent substructure-based approaches for classifying chemical compounds, IEEE Trans. Knowl. Data Eng. 17 (8) (2005) 1036–1050.
[13] B. Li, X. Zhu, L. Chi, C. Zhang, Nested subtree hash kernels for large-scale graph classification over streams, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 399–408.
[14] J.a.R. Bertini Jr., L. Zhao, R. Motta, A. de Andrade Lopes, A nonparametric classification method based on k-associated graphs, Inform. Sci. 181 (24) (2011) 5435–5456.
[15] M. Mohammadi, B. Raahemi, S.A. Mehraban, E. Bigdeli, A. Akbari, An enhanced noise resilient K-associated graph classifier, Expert Syst. Appl. 42 (21) (2015) 8283–8293.
[16] L. Qiao, L. Zhang, S. Chen, D. Shen, Data-driven graph construction and graph learning: A review, Neurocomputing 312 (2018) 336–351.
[17] B. Cheng, J. Yang, S. Yan, Y. Fu, T.S. Huang, Learning with ℓ^1-graph for image analysis, IEEE Trans. Image Process. 19 (4) (2009) 858–866.
[18] M.H. Rohban, H.R. Rabiee, Supervised neighborhood graph construction for semi-supervised classification, Pattern Recognit. 45 (4) (2012) 1363–1372.
[19] T. Jebara, J. Wang, S.-F. Chang, Graph construction and b-matching for semi-supervised learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 441–448.
[20] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[21] Y. Zhu, Y. Yan, Y. Zhang, Y. Zhang, EHSO: Evolutionary hybrid sampling in overlapping scenarios for imbalanced learning, Neurocomputing 417 (2020) 333–346.
[22] E.R. Fernandes, A.C. de Carvalho, Evolutionary inversion of class distribution in overlapping areas for multi-class imbalanced learning, Inform. Sci. 494 (2019) 141–154.
[23] H.K. Lee, S.B. Kim, An overlap-sensitive margin classifier for imbalanced and overlapping data, Expert Syst. Appl. 98 (2018) 72–83.
[24] S. Pan, X. Zhu, Graph classification with imbalanced class distributions and noise, in: Twenty-Third International Joint Conference on Artificial Intelligence, Citeseer, 2013.
[25] Z. Liu, W. Jin, Y. Mu, Graph-based boosting algorithm to learn labeled and unlabeled data, Pattern Recognit. 106 (2020) 107417.
[26] A. Manukyan, E. Ceyhan, Classification of imbalanced data with a geometric digraph family, J. Mach. Learn. Res. 17 (1) (2016) 6504–6543.
[27] L. Peng, H. Zhang, B. Yang, Y. Chen, A new approach for imbalanced data classification based on data gravitation, Inform. Sci. 288 (2014) 347–373.
[28] Y. Zhu, Z. Wang, D. Gao, Gravitational fixed radius nearest neighbor for imbalanced problem, Knowl.-Based Syst. 90 (2015) 224–238.
[29] Z. Wang, Y. Li, D. Li, Z. Zhu, W. Du, Entropy and gravitation based dynamic radius nearest neighbor classification for imbalanced problem, Knowl.-Based Syst. 193 (2020) 105474.
[30] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.

[31] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, EUSboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling, Pattern Recognit. 46 (12) (2013) 3460–3471.
[32] J. Hodges, E.L. Lehmann, Rank methods for combination of independent experiments in analysis of variance, in: Selected Works of EL Lehmann, Springer, 2012, pp. 403–418.
[33] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, A. Bugarín, Stac: a web platform for the comparison of algorithms using statistical tests, in: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2015, pp. 1–8.
[34] G. Du, J. Zhang, F. Ma, M. Zhao, Y. Lin, S. Li, Towards graph-based class-imbalance learning for hospital readmission, Expert Syst. Appl. 176 (2021) 114791.