Neural Computing and Applications (2024) 36:12397–12425
https://doi.org/10.1007/s00521-024-09676-0

ORIGINAL ARTICLE

Imbalanced instance selection based on Laplacian matrix decomposition with weighted k-nearest-neighbor graph

Qi Dai[1] · Jian-wei Liu[1] · Long-hui Wang[2]

Received: 27 November 2023 / Accepted: 25 March 2024 / Published online: 21 April 2024

Abstract
Data are an essential component for building machine learning models. Linearly separable high-quality data are conducive to building efficient classification models. However, the collected dataset is not of high quality, and the number of instances for difference class is not absolutely consistent. Therefore, models built on these datasets are vulnerable to problems such as class-imbalance, class-overlap, and other problems. Traditional instance selection algorithms mainly determine whether there is redundancy or overlap in instances based on the degree of similarity between instances. Therefore, these methods only focus on the local information of the dataset and ignore the global approximate relationship of the instances in the dataset. In this paper, an instance selection method based on the global relationship of instances in the dataset is proposed, called instance selection based on Laplacian matrix decomposition with weighted k-nearest-neighbor graph (LMD-WNG). First, this method tries to construct a new distance-weighted Laplacian matrix using the weighted k-nearest-neighbor graph. Then, the distance-weighted Laplacian matrix is decomposed using a Schur decomposition method. Finally, according to the eigenvalues of the decomposed real matrix, a training dataset suitable for model learning is selected, and a classifier is constructed on the new training data. The experimental results show that as the imbalance ratio increases, LMD-WNG becomes more sensitive to parameter k. When the significance level is $p = 0.05$, the analysis results using Friedman ranking and the Holm's post hoc test show that LMD-WNG is significantly better than or similar to other state-of-the-art algorithms on 30 datasets.

Keywords Weighted k-nearest-neighbor graph · Laplacian matrix · Matrix decomposition · Instance selection · Class-imbalance problems

1 Introduction

In the real world, even though different industries require purer raw materials, high-quality raw materials are more conducive to product manufacturing and save on production costs. For example, higher-quality iron ore is in demand in the metallurgical field, which has given birth to mineral processing technology, and higher-quality crude oil is in demand in the petroleum field, which has also promoted the development of the petrochemical field. For machine learning models, high-quality data are as important as high-quality crude oil or iron ore, because they are more conducive to building efficient machine learning models. However, in the real world, we have to admit that it is hard to collect high-quality dataset that is conducive to building models. In data collection, various data problems

✉ Jian-wei Liu
liujw@cup.edu.cn

Qi Dai
dai18232576157@163.com

Long-hui Wang
2522514454@qq.com

[1] Department of Automation, College of Information Science and Engineering / College of Artificial Intelligence, China University of Petroleum, Beijing, 260 Mailbox China Changping District, Beijing 102249, China

[2] College of Science, North China University of Science and Technology, 21 Bohai Road, Caofeidian Xincheng, Tangshan 063210, Hebei, China

often appear, such as class distribution imbalance [1], class overlap [2, 3], missing values [4, 5], and noise [6, 7].

The class-imbalance problem is an important challenge in the real world [8–10]. Taking the binary classification problem as an example, in the collected dataset, if the number of instances of one class is significantly higher than that of the other class, there will be a class-imbalance phenomenon, which is also called a class-imbalance problem. If datasets with a class-imbalance problem, we call them class-imbalance datasets. However, in the industrial domain, binary classification problems occur more frequently and are more prone to highly imbalanced datasets [11]. In the real world, common class-imbalance problems include but are not limited to: spam classification [12, 13], medical aided diagnosis [14], software defect detection [15], anomaly detection [16, 17], and intrusion detection [18, 19].

Traditional classifiers are trained on balanced or nearly balanced datasets, and they need to do their best to improve the overall accuracy of the model. However, it is easy to cause the minority instances to be misclassified or even directly regarded as noise and deleted [20]. Let's take a simple example. Now we have a dataset with a class-imbalance problem in our hands. There are 2000 instances in this dataset, of which the number of majority instances is 1800. Assuming that the classification model we built directly labels all minority instances as the majority class, if we only observe the overall accuracy, our model can achieve an accuracy of 90%, which is already excellent classification performance. In fact, our model did not accurately identify any of the minority instances in the dataset. However, in the real world, minority instances are very important. For example, for datasets in fault diagnosis, there are far more normal instances than faulty instances. If we briefly use the traditional classifier, it will be difficult to identify these truly faulty instances, resulting in significant economic losses and even casualties [21].

Methods to cope with the class-imbalance problem can be simply divided into preprocessing (also known as data resampling) [22, 23], algorithm modification [24, 25], cost-sensitive learning [26, 27], and ensemble learning [28, 29]. The preprocessing technique is a very common one, and it is independent of classifiers. They are widely used and can be combined with any traditional classifier to build an efficient classifier [30]. In these proposed data preprocessing techniques, they all follow a common assumption, namely, that in the preprocessing process, each input instance is independent of each other and that there is no connection between them. However, in the real world, these instances may have subtle differences in feature values, but there must be a certain relationship with other instances in the dataset. In some data preprocessing techniques, they use nearest neighbors to represent the neighbor relationship between instances in a local area [31]. However, they cannot efficiently capture the global structure in the dataset. The use of graphs to represent data structures is rare among preprocessing techniques for handling class-imbalance problems.

However, we also realize that the global graph structure information can provide researchers with more new knowledge. In the class-imbalance problem, the global structural information of the dataset may be ignored based on the local information between instances in the dataset. However, in real-world datasets, the local neighbor relationships between instances are not always simple neighborhood relationships. Each instance may have many global relationships with the global instances of the dataset that cannot be accurately represented. For example, if there are small disjuncts in the dataset, if only the local information between instances is considered, these instances may be considered to be noise, and they will be eliminated. However, based on the global information, it is not difficult to find that there may be more valuable information in local areas formed by small disjuncts. Therefore, considering the global structural information of the dataset is a far-sighted strategy.

The global structure information of the dataset has a variety of representation methods, such as adjacency matrix and incidence matrix. However, the adjacency matrix and the incidence matrix, respectively, represent the adjacent relationship or the correlation between instances. Adjacency matrices and incidence matrices cannot show the number of correlations between instances and have certain limitations. Laplacian matrix is a common graph structure information representation method based on adjacency matrix. The Laplacian matrix not only represents the adjacent relationship in combination with the adjacency matrix, but the degree matrix can also accurately show the neighborhood relationship between instances. The more complex the neighborhood relationship, the larger the value corresponding to the instance in the degree matrix. Therefore, in the LMD-WNG algorithm, we use the Laplacian matrix to represent the global structural information.

The Laplacian matrix, also known as the Kirchhoff matrix, is a commonly used matrix representation in graph theory [32]. Some researchers have used the Laplacian matrix for the class-imbalance problem [33–35]. Ye et al. [36] combined Laplacian eigenmaps to generate instances in the mapped feature subspace, thereby avoiding SMOTE from generating noisy minority pseudo-instances. In this method, the best feature subspace in the dataset is found by means of Laplacian feature mapping, and the instances in the dataset are separated in this subspace. They are the first to use the Laplacian eigenmaps approach for class-imbalanced data preprocessing. However, no researchers have applied Laplacian matrices to instance selection or undersampling techniques.

In summary, this is the first attempt to construct a new instance selection method using Laplacian matrices and matrix decomposition techniques. The method is called instance selection based on Laplacian matrix decomposition with weighted k-nearest-neighbor graph (LMD-WNG). In this method, the data space is first searched using the k-NN, and a weighted k-nearest-neighbor graph (Wk-NNG) is constructed using the distance metric method. Then, the distance-weighted Laplacian matrix in the original dataset is obtained according to the weighted k-nearest-neighbor graph. Finally, using the matrix decomposition technique (Schur decomposition), the Laplacian matrix is decomposed, and the relationship between instances in the dataset is found according to the eigenvalues of the decomposed diagonal matrix.

The contributions of LMD-WNG are as follows:

1. To address the class-imbalance problem, we try to use the k-nearest-neighbor graph to extract the global structural information;
2. We use a weighted neighbor graph to construct a Laplacian matrix that fully represents the global structural information of each instance. According to the weighted Laplacian matrix, the nearest-neighbor relationship and the distance value between instances can be clearly presented, providing new insights for dealing with class-imbalance problems.
3. We combine the graph structure information and the matrix decomposition method to select instances that are beneficial to classifier learning based on the eigenvalues of the global weighted Laplacian matrix, which alleviates the impact of the class-imbalance problem on classifier performance.

The remainder of this paper describes: In Sect. 2, we review recent research progress on resampling techniques in the field of class-imbalance learning, especially instance selection and undersampling methods. In Sect. 3, the motivation for LMD-WNG and the calculation process of the algorithm are introduced. The introduction of the experimental setup, datasets, base classifiers, and evaluation metrics will be given in Sect. 4. Next, the parameter sensitivity analysis of LMD-WNG is presented in Sect. 5. The experimental results and nonparametric statistical test results with other state-of-the-art comparative algorithms are given in Sect. 6. Finally, in Sect. 7, we conclude this paper with our main conclusions and future work.

2 Related works

With the advent of the big data era, various datasets have emerged in different fields, and these are often imbalanced. As the size of the data increases, it will also face more serious class-imbalance problems, especially when class-imbalance problems and other data problems such as class overlap coexist [37]. However, in the real world, these small-scale class-imbalance problems are not as simple as imagined.

In general, most researchers tend to divide methods into four categories: data-level method, algorithm-level method, cost-sensitive learning, and ensemble learning. In addition, some researchers believe that cost-sensitive learning is a reweighting method, and both cost-sensitive learning and ensemble learning are algorithm-level methods. Among these four types of methods, the data-level method is an easy-to-implement method. Among all the proposed data preprocessing methods, they can be divided into oversampling [38], which adds minority instances, undersampling [39], which removes majority instances, and hybrid sampling [40], which uses both methods simultaneously. Since the LMD-WNG instance selection method is a preprocessing method similar to undersampling, in the subsequent introduction, we will pay attention to the data preprocessing method.

2.1 Oversampling

Random oversampling (ROS) balances the training dataset by randomly replicating minority instances. When there are few minority instances, the randomly copied minority instances will cover a large area of the original data space, which will cause the model to fall into the problem of overfitting. Synthetic minority oversampling technique (SMOTE) [41] is designed to overcome the overfitting of random replication instances, and it expands the number of minority instances by performing linear interpolation between two minority instances. However, they only synthetic minority instances in the data space of minority instances and do not expand the overall feature space of minority instances. When the class-overlap problem exists simultaneously in the dataset, the pseudo-minority instances generated by SMOTE may move to the majority class area, causing a more serious class-overlap problem.

In order to avoid the overfitting problem caused by SMOTE, a lot of work has been carried out. In the latest research, some researchers are committed to transforming the original data space into a different data space to create synthetic minority instances. Dai et al. [42] mapped all instances to one-dimensional space according to distance mapping and searched for neighbor instances of minority instances in one-dimensional space. Yuan et al. [43] synthesized minority instances on the convex hull of minority instances, increased the feature space of minority instances, and avoided generating noisy minority instances in the majority class region. Li et al. [44] proposed a new oversampling method called subspace minority oversampling (SMO). This method divides the features in the dataset into common features and unique features and uses different interpolation methods to linearly interpolate the minority instances in the dataset. Islam et al. [45] proposed a technique called k-nearest-neighbor overresampling (KNNOR) by identifying the compactness and position of minority instances relative to majority instances.

In recent years, partial oversampling methods still use different local search methods to search for local neighborhood relationships and synthesize new instances in a linear or nonlinear manner. However, when there are class-overlap or small disjuncts, the local neighborhood relationship cannot accurately represent the global relationship between the instances. Therefore, the global graph structure information can be used to discover minority instances with insufficient information from the global relationships.

2.2 Undersampling

Random undersampling (RUS) balances the training dataset by randomly eliminating instances from the majority area. If the RUS is used directly, there is a risk of losing important information in the majority instances, and the method lacks interpretability. The class-overlap problem is another difficult factor affecting classifier performance. It helps to alleviate the impact of class-overlap problems on classifier performance by selecting overlapping instances that are not conducive to classifier learning. In current research on the class-imbalance problem, some researchers still use local information or underlying structural information to propose new undersampling methods. Recent undersampling methods built based on local information are as follows: Hoyos-Osorio et al. [46] proposed a related information undersampling method based on the principle of information preservation to reduce the underlying data structure for extracting majority instances. Yan et al. [47] proposed the spatial distribution undersampling (SDUS) method according to the distribution of the dataset. In this method, top-down and bottom-up strategies can be used to select subsets of majority instances from different perspectives. Farshidvard et al. [48] proposed a two-stage approach to address class imbalance using undersampled data-level and ensemble learning methods. This method keeps the convex hull of each majority cluster formed by clustering without minority instances, thereby controlling the size of each cluster.

The undersampling method can be considered an instance selection method. Undersampling alleviates the impact of class imbalance on the classifier by removing redundant or overlapping majority instances. At present, some researchers have noticed that using techniques such as clustering is an effective method to discover local structural information. However, they did not use techniques such as clustering to discover the global structural information and use the structural information to propose new instance selection techniques. To the best of our knowledge, the overall structural characteristics of the dataset are not fully considered in undersampling methods to deal with the class-imbalance problem. In [49], we attempted to discover global features of datasets using similarity matrices and matrix decomposition techniques. However, we only considered the similarity between instances and did not deeply explore the number of nearest neighbors between them. In this paper, we try to combine distance-weighted Laplacian matrices and matrix decomposition methods to explore the distribution structure around instances in the original dataset and propose a new instance selection algorithm.

3 The algorithm of LMD-WNG

We mainly introduce the relevant theory and algorithm details of the LMD-WNG. In Sect. 3.1, we briefly introduce the motivation of LMD-WNG. In Sect. 3.2, we detail the theoretical basis of the proposed LMD-WNG instance selection algorithm. In Sect. 3.3, we introduce the overall process of LMD-WNG in detail and give the pseudocode of the algorithm. Furthermore, the time complexity of the algorithm is presented in Sect. 3.4.

3.1 Motivation

The impact of the class-imbalance problem is particularly severe when it coexists with other difficulties. Therefore, studying the solution to the class-imbalance problem is a fundamental way to improve the learning performance of classifiers. For the class-imbalance problem, the data preprocessing technique is similar to a data refinement technique designed for the class-imbalance problem. These methods can convert sparse and complex datasets into high-quality training data that is conducive to classifier learning through these preprocessing techniques.

More preprocessing techniques to cope with the imbalanced dataset. Some researchers believe that oversampling techniques are more effective than undersampling techniques. However, we believe that such a description is not comprehensive, and not all oversampling techniques are optimal on all datasets. Let’s simply think about it. When the minority instances are too sparse and overlap with the majority instances, the traditional oversampling technique is used directly, and the synthetic minority instances still overlap with the majority instances, which is not conducive to the classification boundary of the traditional classifier learning dataset. Besides that, using oversampling in the laboratory may improve the results of evaluation metrics. In the field of practical application, the synthesized pseudo-minority instances are likely not to represent the actual instances, causing the classification model to fail to recognize new unknown instances [50]. Therefore, we believe that the oversampling technique and the undersampling technique in the resampling technique do not have an absolute advantage but should be developed together for different problems.

Laplacian matrix is a common method in graph theory. Ye et al. [36] introduced the Laplacian matrix into the oversampling technique for the first time. They construct a k-nearest-neighbor graph (k-NNG) using k-nearest neighbors and use Laplacian eigenmaps to find the optimal low-dimensional space for the dataset. Inspired by this, we try to introduce the idea of a Laplacian matrix into instance selection or undersampling techniques. In [49], we used the metric learning method, constructed a similarity matrix, and searched for the global similarity trend of the dataset using positive and negative inertial trends, thereby achieving undersampling of the dataset. We think that instances of different classes that are more similar in the dataset are more likely to become overlapping instances in the dataset.

In addition, the edges of the k-NNG have no weight. When generating the Laplacian matrix, we directly mark the corresponding position in the adjacency matrix as 1. There is an underlying assumption in using this approach, namely that the weights of instances connected to vertices are considered to be the same, which is not conducive to distinguishing the distance between adjacent instances in the k-nearest-neighbor graph. Therefore, we use the distance metric to calculate the distance between adjacent instances as the weight of the corresponding edge in the k-NNG and use the Wk-NNG to generate a distance-weighted Laplacian matrix.

Since the Laplacian matrix generated using the k-nearest-neighbor graph is a sparse matrix, it cannot be directly used for instance selection. Therefore, we use the matrix decomposition method to decompose the generated distance-weighted Laplacian matrix and use the eigenvalues of the matrix to select the appropriate majority class as the training set of the classifier. Furthermore, we need to clarify the reason for using eigenvalues to select instances. The mathematical meaning of eigenvalues is shown in Definition 1.

Definition 1 (Eigenvalues). Let there be an n-dimensional square matrix $\boldsymbol{M}$. If there is a constant λ and an n-dimensional nonzero column vector $\boldsymbol{x}$, so that $\boldsymbol{Mx} = \lambda\boldsymbol{x}$ is established, then λ is said to be the characteristic value or eigenvalue of the matrix $\boldsymbol{M}$.

According to the definition of eigenvalues, we can rewrite the equation $\boldsymbol{Mx} = \lambda\boldsymbol{x}$ as the relation $(\lambda\boldsymbol{E} - \boldsymbol{M})\boldsymbol{x} = 0$, where $\boldsymbol{E}$ is the identity matrix. During the solution process, the determinant $\det(\lambda\boldsymbol{E} - \boldsymbol{M}) = 0$ is required. Therefore, we can solve for the eigenvalues of a matrix by constructing a system of homogeneous linear equations for the relation. The Laplacian matrix obtained through the weighted k-nearest-neighbor graph is not a positive-definite matrix, so its corresponding eigenvalues must be positive and negative.

Furthermore, the weighted Laplacian matrix we obtain represents the adjacent position and compactness between instances in the dataset. Therefore, the eigenvalues in a system of homogeneous linear equations can be regarded as the global compactness of the corresponding instances. We rely on empirical methods to use the eigenvalues after matrix decomposition as instance selection criteria, and a new heuristic instance selection method is proposed. When the eigenvalue corresponding to the instance is negative, it means that the information about the instance is more independent and needs to be retained.

In conclusion, based on the above ideas, the matrix decomposition method is used to calculate the eigenvalues of the distance-weighted Laplacian matrix, and instances that are more suitable for classification can be selected according to the eigenvalues.

3.2 Preliminary knowledge

Before introducing the LMD-WNG instance selection algorithm, we need to introduce the preliminary knowledge used by LMD-WNG. In Sect. 3.2.1, we briefly introduce weighted k-nearest-neighbor graphs (k-NNG). The Laplacian matrix used is described in Sect. 3.2.2. Finally, the matrix decomposition method used (Schur decomposition [51]) is presented in Sect. 3.2.3.

3.2.1 Weighted k-nearest-neighbor graph (Wk-NNG)

Graph theory is an important branch of mathematics. It uses graphs as the research object. In graph theory, a graph is composed of several given points (or instances) and lines connecting two points. This type of graph is usually used to describe a certain relationship between points (or instances). The basic representation of a graph is shown in Definition 2.

Definition 2 (Graph). A graph can be represented as $\mathcal{G} = (\boldsymbol{v}, E)$, where $\boldsymbol{v} = \{v_1, v_2, \ldots, v_n\}$ represents the set of vertices in the graph and $E = \{e_{ij}\}, i,j = 1, 2, \ldots, n$ represents the set of edges connecting two vertices.

It should be noted that graphs can be divided into directed graphs and undirected graphs according to whether the edges have directions. If the generated graph is a directed graph, then e_{ij} means the edge connecting vertex i and vertex j; e_{ij} means that the direction of the edge is from vertex i to vertex j; otherwise, it means that the direction of the edge is from vertex j to vertex i. Although the graph generated in the k-NNG is a directed graph, in our algorithm we only consider the connectivity between instances and the distance metric on edges. Figure 1 shows a simple graph structure.

The k-NNG [52] is a special graph built using the k-NN method that can represent the relationship between instances in a dataset and thus discover the overall data structure of the dataset. However, k-NNG is a directed graph defined by a given set of instances in a metric space. In the dataset, the connectivity information between all instances can be discovered by using the k-NNG. We give the definition of the k-NNG in Definition 3.

Definition 3 (k-nearest neighbor graph [53]). Suppose the dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^d$ contains n instances, and d represents the dimension of each instance in the dataset. $\mathcal{G} = (\boldsymbol{v}, A_h)$ is a directed graph, the vertex set $\boldsymbol{v}$ contains all instances in the dataset $\mathcal{D}$, and the adjacency matrix $\boldsymbol{A} = A(i,j)_{n \times n}$ constructed according to the dataset $\mathcal{D}$ is defined as follows:

$$\boldsymbol{A}(i,j) = \begin{cases} 1 & \text{if } \boldsymbol{x}_j \in N(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $N(\boldsymbol{x}_i)$ represents the set of k-nearest neighbors of instance $\boldsymbol{x}_i$.

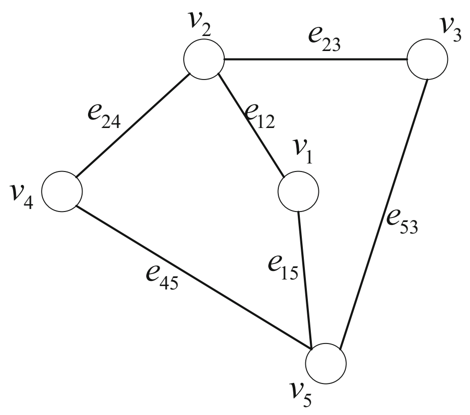


Fig. 1 A simple graph structure

When $A(i,j) = 1$, instance $\boldsymbol{x}_i$ and instance $\boldsymbol{x}_j$ are one-way connected, and there is a unilateral connection relationship between them. It should be noted that there are mutual nearest neighbors in the dataset. When $A(i,j) = A(j,i) = 1$, instance $\boldsymbol{x}_i$ and instance $\boldsymbol{x}_j$ are bidirectional connected, and there is a bidirectional connection relationship between them.

Remark 1 It should be noted that not all instances have bidirectional connectivity. The nearest-neighbor relationship formed by using the k-nearest-neighbor search is not a symmetrical relationship; that is, it is assumed that instance $\boldsymbol{x}_i$ is the nearest neighbor of instance $\boldsymbol{x}_j$ in the dataset $\mathcal{D}$. However, it does not mean that instance $\boldsymbol{x}_i$ is also instance $\boldsymbol{x}_j$'s nearest neighbor.

As the hyperparameter k increases, there must be bidirectional connectivity in the dataset. In the LMD-WNG algorithm, we do not focus on the samples of bidirectionally connected relations. In the k-NNG we constructed, when $A(i,j) = A(j,i) = 1$ appears, it is only calculated once in the adjacency matrix, and the value of the corresponding position is 1. (If a distance-weighted k-NNG is used, the value of the corresponding position of the adjacency matrix is $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$.)

Furthermore, according to the definition of the k-NNG given in Definition 1, LMD-WNG will search for the k-NN in the dataset, and we can easily get the k-nearest-neighbor graph. In LMD-WNG, k is a hyperparameter, and the parameter sensitivity analysis of LMD-WNG will be given in Sect. 5.

In many applications, the direction of edges in the formed k-NNG is ignored, and the graph becomes an undirected graph. If the k-NNG of definition 1 is used to construct the Laplacian matrix, it is not difficult to find that the k-NN searched for the instance $\boldsymbol{x}_i$ will all be marked as 1. We believe that in the resulting k-NNG, the weights corresponding to all instances are the same. However, such an approach is unreasonable, the main reason being that not all values obtained by distance metrics are equal. The smaller the value of the distance metric, the closer the distance or similarity between instances. It is not advisable for us to treat all instances equally. Therefore, in the k-NNG we constructed, we need to calculate the degree of similarity between instances.

In the field of machine learning, there are many methods that can measure the correlation or similarity, such as the Manhattan distance, Minkowski distance, Euclidean distance, and cosine similarity. Different measurement methods can discover different characteristics between instances. Euclidean distance is a classic distance measurement and has been recognized by most researchers

[52, 53]. In addition, Euclidean distance has the following advantages:

1. Euclidean distance is very intuitive to use and easy to implement.
2. Euclidean distance works very well when you encounter low-dimensional data and the size of the vector is important.

In addition, we also noticed that when the dimensionality is high or there are obvious dimensional differences in the features, the Euclidean distance will also fall into the problem of dimensionality disaster or inaccurate measurement. For such datasets, we can replace other measurement methods, such as standardized Euclidean distance and Mahalanobis distance. However, in order to be consistent with the k-nearest-neighbor graph search process, using k-nearest neighbors, Euclidean distance is used to calculate the degree of similarity. The definition of Euclidean distance is shown in Definition 4.

Definition 4 (Euclidean distance). Suppose the dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^d$ contains n instances, and d represents the dimension of each instance in the dataset. The Euclidean distance between instance $\boldsymbol{x}_i$ and instance $\boldsymbol{x}_j$ is defined as follows:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^k - x_j^k\right)^2} \tag{2}$$

Correspondingly, we will calculate the Euclidean distance as the weight of the edge with connected instances and construct a Wk-NNG. The adjacency matrix $\boldsymbol{A} = A(i,j)_{n \times n}$ formed at this time can be rewritten as:

$$\boldsymbol{A}(i,j) = \begin{cases} d(\boldsymbol{x}_i, \boldsymbol{x}_j) & \text{if } \boldsymbol{x}_j \in N(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3.2.2 Laplacian matrix

The Laplacian matrix is the core content of the whole graph theory. With the in-depth study of graph neural networks, using Laplacian matrices to solve graph structure problems is becoming more and more popular [33–35]. Computing the adjacency matrix and degree matrix according to the graph structure is an important step in computing the Laplacian matrix. Laplacian matrices are widely used in clustering and other applications. There are two common forms of Laplacian matrices: classical Laplacian matrices and normalized Laplacian matrices. In the proposed LMD-WNG algorithm, the classical Laplacian matrix is used. Therefore, we will introduce the definition of the classical Laplacian matrix, as shown in Definition 5. In addition, for a better understanding, we give the calculation process of the classical Laplacian matrix according to the simple graph structure shown in Fig. 1 in Example 1.

Definition 5 (Classical Laplacian matrix). Let $\mathcal{G} = (\boldsymbol{v}, A_h)$ be a graph, then the classical Laplacian matrix is defined as:

$$\boldsymbol{L} = \boldsymbol{D}(\mathcal{G}) - \boldsymbol{A}(\mathcal{G}) \tag{4}$$

where $\boldsymbol{A}(\mathcal{G})$ denotes the adjacency matrix of graph $\mathcal{G}$, and $\boldsymbol{D}(\mathcal{G})$ denotes the degree matrix of graph $\mathcal{G}$. In the degree matrix, except for the diagonal elements, the elements at other corresponding positions are all zero. And the corresponding element $D_i, i = 1, 2, \ldots, n$ on the diagonal represents the number of edges drawn from the vertex i.

Example 1. According to the graph structure given in Fig. 1, we can get the adjacency matrix as:

$$\boldsymbol{A}(\mathcal{G}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The degree matrix is:

$$\boldsymbol{D}(\mathcal{G}) = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Therefore, the classical Laplacian matrix is:

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{D}(\mathcal{G}) - \boldsymbol{A}(\mathcal{G}) \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{bmatrix} \end{aligned}$$

Since the values of the adjacency matrix in the classical Laplacian matrix are all expressed as 1, the classical Laplacian matrix ignores the distance difference between the k-nearest-neighbor instances. Therefore, calculate the distance between adjacent instances in the adjacency matrix and calculate the distance obtained as the weight of the adjacency matrix. In the LMD-WNG instance selection algorithm, the calculated distance-weighted Laplacian matrix is expressed as follows:

$$\boldsymbol{L}\prime = \boldsymbol{D}(\mathcal{G}) - \boldsymbol{A}\prime(\mathcal{G}) = \boldsymbol{D}(\mathcal{G}) - \boldsymbol{d} \otimes \boldsymbol{A}(\mathcal{G}) \tag{5}$$

where $\otimes$ represents Hadamard product, $\boldsymbol{L}\prime$ represents the distance-weighted Laplacian matrix, and $\boldsymbol{A}\prime(\mathcal{G})$ represents the adjacency matrix weighted by distance, $\boldsymbol{d}$ represents the distance matrix between instances. It should be noted that in the LMD-WNG instance selection algorithm, the Euclidean distance is used to calculate the distance between adjacent instances. Therefore, no matter which vertex the edge emanates from, the distance value is the same, i.e., $d_{ij} = d_{ji}$.

Continuing from Example 1, we can get the weighted Laplacian matrix as:

$$\begin{aligned}\boldsymbol{L}\prime &= \boldsymbol{D}(\mathcal{G}) - \boldsymbol{A}\prime(\mathcal{G})\\ &= \begin{bmatrix} 2 & 0 & 0 & 0 & 0\\ 0 & 3 & 0 & 0 & 0\\ 0 & 0 & 2 & 0 & 0\\ 0 & 0 & 0 & 2 & 0\\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}\\ &\quad - \begin{bmatrix} 0 & d_{21} & 0 & 0 & d_{51}\\ d_{12} & 0 & d_{32} & d_{42} & 0\\ 0 & d_{23} & 0 & 0 & d_{53}\\ 0 & d_{24} & 0 & 0 & d_{54}\\ d_{15} & 0 & d_{35} & d_{45} & 0 \end{bmatrix}\\ &= \begin{bmatrix} 2 & -d_{21} & 0 & 0 & -d_{51}\\ -d_{12} & 3 & -d_{32} & -d_{42} & 0\\ 0 & -d_{23} & 2 & 0 & -d_{53}\\ 0 & -d_{24} & 0 & 2 & -d_{54}\\ -d_{15} & 0 & -d_{35} & -d_{45} & 3 \end{bmatrix}\end{aligned}$$

3.2.3 Schur decomposition

Matrix decomposition is a commonly used mathematical method that has important research value and significance in the analysis of matrix theory. The Schur decomposition method is a common method of matrix decomposition. In the proposed LMD-WNG instance selection algorithm, we also decompose the distance-weighted classical Laplacian matrix using the Schur decomposition method. The relevant theorems and corollaries of Schur's decomposition are shown in [51]. Schur decomposition has three important properties, namely subspace invariance of Schur vectors, scaling transformation invariance of Schur vectors, and perturbation stability [51]. Therefore, we can use the Schur decomposition method to solve the eigenvalues of the weighted Laplacian matrix more stably and select appropriate instances based on the eigenvalues.

According to the conclusion drawn in [51], we can see that the time consumption of using matrix decomposition technique is relatively large, but we can use GPU or other acceleration algorithms for matrix calculation to achieve accelerated calculation of matrix decomposition. Based on the three important properties of Schur decomposition mentioned above, we can see that when the constructed Laplacian matrix is perturbed, the result of using matrix decomposition will not cause a great impact, and the formed eigenvalues are stable. During the instance selection process, by observing the decomposed eigenvalues, we can clearly know which instances are selected by the model. Therefore, methods using matrix decomposition are highly interpretable. In addition to using the Schur decomposition method to decompose the distance-weighted Laplacian matrix, most matrix decomposition methods such as QR decomposition and SVD decomposition can be used [51].

3.3 Algorithm description

LMD-WNG is an instance selection method combining Laplacian matrices and matrix decomposition techniques. LMD-WNG is the first novel approach combining Laplacian matrices and matrix decomposition techniques on class-imbalance problems. This method is divided into four phases: constructing the k-NNG; computing the classical distance-weighted Laplacian matrix; matrix decomposition (Schur decomposition); and instance selection. In the first phase, we use the k-NN to search the instance space and form a k-NNG. The second phase is to calculate the adjacency matrix and degree matrix according to the k-NNG and calculate the classical distance-weighted Laplacian matrix of the k-NNG. The Schur decomposition is carried out in the third phase, which mainly decomposes the classical distance-weighted Laplacian matrix to obtain the corresponding eigenvalues of the matrix. In the fourth phase, instance selection is to select instances in the majority class according to the size of the eigenvalues in the Laplacian matrix. Finally, the selected majority instances are combined with the minority instances to form a new training set. The flowchart of LMD-WNG instance selection algorithm is shown in Fig. 2.

Next, we introduce the details of the LMD-WNG instance selection algorithm in detail and give the pseudocode of LMD-WNG in Algorithm 1.

Phase 1: Search the instance space and generate the k-NNG.

Step 1: According to Definition 2, the k-NNG of the instance space is constructed using the k-NN method. If instance $\boldsymbol{x}_i$ is one of the k-nearest neighbors of instance $\boldsymbol{x}_j$

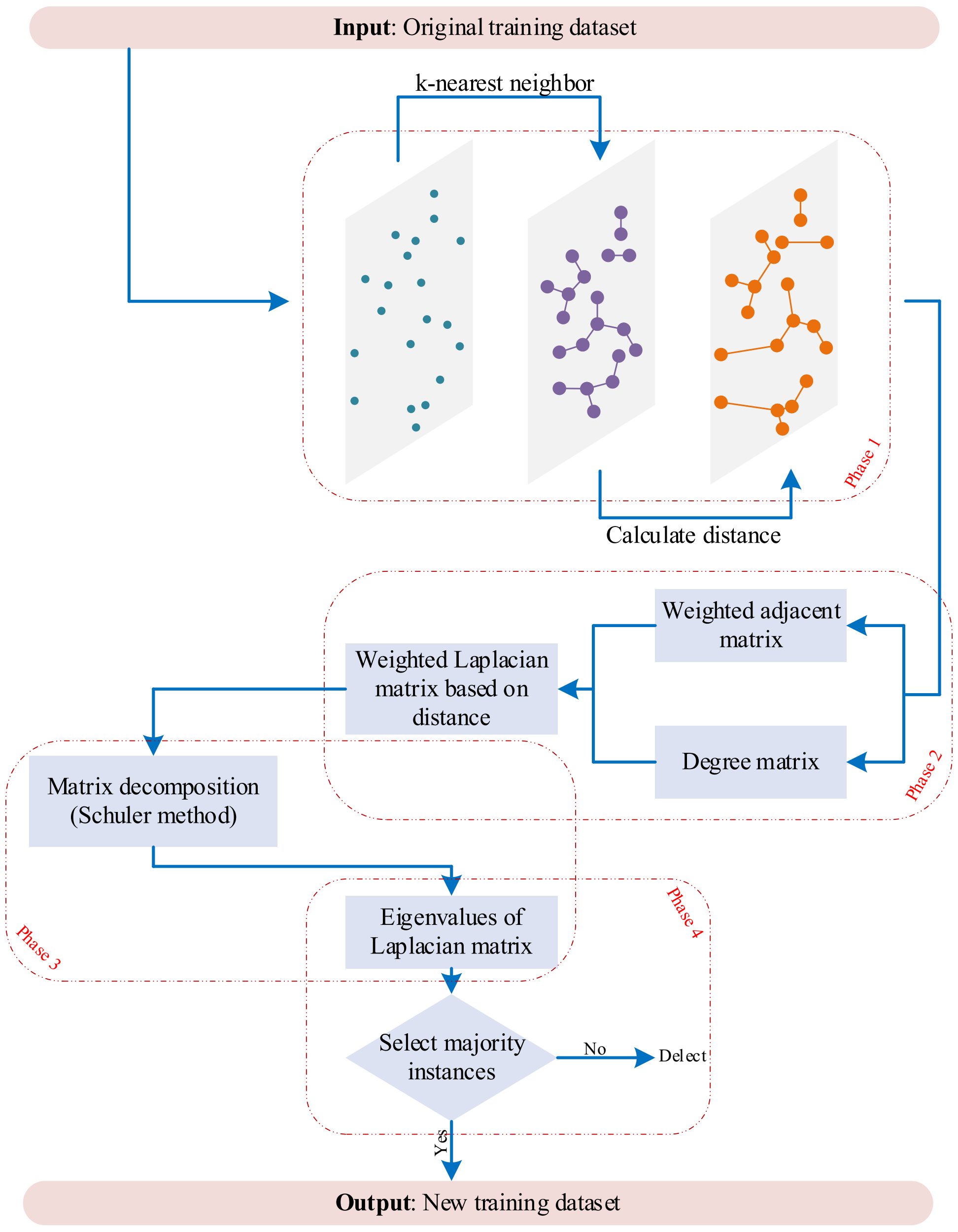

Input: Original training dataset
k-nearest neighbor
Calculate distance
Phase 1
Weighted adjacent matrix
Weighted Laplacian matrix based on distance
Degree matrix
Phase 2
Matrix decomposition (Schuler method)
Phase 3
Eigenvalues of Laplacian matrix
Phase 4
Select majority instances
No
Delect
Yes
Output: New training dataset

Fig. 2 The flowchart of LMD-WNG

or instance $\boldsymbol{x}_j$ is one of the k-nearest neighbors of instance $\boldsymbol{x}_i$, use a line connection between instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. In Sect. 3.2.1, we also mentioned that instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ may be k-nearest neighbors to each other, which is called a bidirectional connection. In this case, we only connect once. In addition, the generated k-NNG is a standard directed graph, but the direction of the connection does not affect our use. Therefore, we directly treat the resulting graph as an undirected graph.

Step 2: The Euclidean distance between neighboring instances is calculated according to Eq. (2) and used as the weight of the edge connecting two instances.

Phase 2: Compute the distance-weighted Laplacian matrix $\boldsymbol{L}\prime$.

Step 3: Calculate the weighted adjacency matrix $\boldsymbol{A}\prime(\mathcal{G})$ and the corresponding degree matrix $\boldsymbol{D}(\mathcal{G})$ according to the k-nearest-neighbor graph $\mathcal{G}$.

Step 4: According to Definition 4, the distance-weighted Laplacian matrix L of the instance space is calculated.

Phase 3: Decompose the distance-weighted Laplacian matrix $\boldsymbol{L}\prime$.

Step 5: Decompose the weighted Laplacian matrix $\boldsymbol{L}\prime$ using the Schur decomposition method described in Sect. 3.2.3. Calculate the eigenvalue vector corresponding to the matrix $\boldsymbol{L}\prime$ according to the Schur decomposition method.

Phase 4: Majority instance selection.

Step 6: Arrange in ascending order according to the eigenvalues of the distance-weighted Laplacian matrix $\boldsymbol{L}\prime$.

Step 7: Select the majority instances in ascending order of eigenvalues and form a new training dataset together with the minority instances.

It should be noted that, assuming that in the dataset $\mathcal{D}$, the number of instances in the majority class is $|N^{-}|$, and the number of instances in the minority class is $|N^{+}|$, then in the fourth phase, there are two situations:

Case 1. When the number of instances $\left|N^{-}_{\mathrm{neg}}\right|$ with negative eigenvalues in the corresponding position in the majority instances is more than the number of instances in the minority class $|N^{+}|$, all instances with negative eigenvalues will be selected to join the training set. We consider instances with negative eigenvalues to be more independent and have fewer instances around them. If they are deleted rashly, it may not be conducive to building an efficient classification model.

Case 2. When the number of instances $\left|N^{-}_{\mathrm{neg}}\right|$ with negative eigenvalues in the corresponding position in the majority class is less than the number of instances in the minority class $|N^{+}|$, we will select the majority instances in order from small to large. When the majority instances are balanced with the minority instances, we stop adding more instances to the new training set.

The pseudocode of the LMD-WNG is shown in Algorithm 1.

Algorithm 1 LMD-WNG instance selection algorithm

```
Input: training dataset $\mathsf{D}$ , Hyperparameter $k$
Output: new training dataset $\mathsf{D}_{new}$

1. $\boldsymbol{M} = \boldsymbol{A}^{'}(\mathsf{G}) = \boldsymbol{D}(\mathsf{G}) = \boldsymbol{L}^{'}(\mathsf{G}) = \varnothing$
2. For i=1 to n do
3.     For j=1 to n do
4.         $\boldsymbol{M}_{ij} \leftarrow \sqrt{\sum_{k=1}^{d}(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k)^2}$
5.     End for
6. End for
7. Searching sample space and construct k-nearest neighbor graph $\mathsf{G}$
8. $\boldsymbol{A}^{'}(\mathsf{G}) \leftarrow$ Distance weighted adjacency matrix of graph $\mathsf{G}$
9. $\boldsymbol{D}(\mathsf{G}) \leftarrow$ Degree matrix of graph $\mathsf{G}$
10. $\boldsymbol{L}^{'}(\mathsf{G}) \leftarrow \boldsymbol{D}(\mathsf{G}) - \boldsymbol{A}^{'}(\mathsf{G})$
11. Decompose the matrix $\boldsymbol{L}^{'}(\mathsf{G})$ using the Schur decomposition method
12. Calculate the eigenvalues of the matrix $\boldsymbol{L}^{'}(\mathsf{G})$
13. If $\left|N_{neg}^{-}\right| \geq \left|N^{+}\right|$ then:
14.     Add all samples in $N_{neg}^{-}$ to the new training set
15.     Else
16.     Select majority instances in descending order of eigenvalues
17. End if
18. Blending minority instances and selected majority instances to form a new training dataset $\mathsf{D}_{new}$
```

3.4 Complexity

We derive the computational complexity of the LMD-WNG instance selection algorithm. Before deriving the complexity, we assume that the number of instances in the dataset $\mathcal{D}$ is n, and the number of features is d.

In the LMD-WNG instance selection algorithm, its computational complexity mainly comes from two phases: k-nearest-neighbor graph construction and Schur decomposition. For the other two phases, a simple search is performed on the distance-weighted Laplacian matrix, so their computational complexity is negligible. In the first phase, we need to search the entire instance space using the k-nearest-neighbor algorithm. Since there may be differences in the methods to implement the k-nearest-neighbor search, we only consider directly searching the instance space using the k-nearest-neighbor method. The computational complexity of the whole process is $O(n^2d)$. In addition, in the third phase, its complexity mainly comes from decomposing the distance-weighted Laplacian matrix $\boldsymbol{L}\prime$ using the Schur decomposition method. Therefore, according to the results of [54], it can be known that its computational complexity is $O(n^3)$.

In summary, the overall complexity of LMD-WNG is $O(n^2d + n^3)$. Since the dataset we use is not high-dimensional, the dimension d has less impact on complexity. Therefore, the final complexity of LMD-WNG can be expressed as $O(n^3)$. For large-scale datasets, this complexity is high and consumes a lot of computing resources. However, traditional methods are not suitable for larger-scale imbalanced datasets, and artificial neural networks are good methods for processing them. During the training process of the network, we will not directly use the entire dataset for training. The mini-batch training method is an effective method for training the neural network. However, when encountering highly imbalanced datasets, we cannot guarantee that the data distribution in each mini-batch is balanced. Therefore, LMD-WNG can be used as a pre-processing method to select the best instances in mini-batches for training.

4 Experimental setting and evaluation metrics

We use Python for modeling, conduct experiments on 30 public datasets, and compare them with state-of-the-art algorithms to verify the effectiveness of the proposed instance selection algorithm (LMD-WNG). The experimental computer is as follows: CPU: i7-6700M; System: Win10 Pro; Interpreter: Python 3.10; imbalanced_learn: 0.10.1; Scikit_learn: 1.2.2; Numpy: 1.22.3; Pandas: 1.4.2. The base classifier used in the comparative experiment is introduced in Sect. 4.1. The basic information about the dataset is shown in Sect. 4.2. Furthermore, the evaluation metrics used in the experiments will be introduced in Sect. 4.3.

It should be noted that due to the large number of datasets used in this paper, we divided all datasets into three categories according to the imbalance ratio (IR) [55] in the parameter sensitivity analysis and comparative experiment. In [55], they consider such datasets as severely imbalanced when the IR is greater than 10. Throughout the experiments, we also followed their approach to analyzing the experimental results in terms of IRs. However, we classify imbalanced data into three types: mildly imbalanced problems, moderately imbalanced problems, and highly imbalanced problems. When $\mathrm{IR} \le 5$, we consider such a dataset to be only mildly imbalance problems. We consider such datasets to have moderately imbalanced problems when $5 < \mathrm{IR} \le 10$ applies. When $10 < \mathrm{IR}$, the dataset is a highly imbalanced dataset. We generally believe that the higher the imbalance ratio, the more difficult it is to classify directly using traditional classifiers.

4.1 Classifiers

We chiefly introduce the two classifiers used in the parameter sensitivity analysis and comparative experiments. They are gradient boosting tree (GBDT) based on boosting, random forest (RF) based on bagging, and support vector machine (SVM), respectively. GBDT and RF are classic methods based on decision trees. The tree structure method has better interpretability, and we can clearly see the division process of each node. Therefore, in our experiments, LMD-WNG is used to improve the performance of tree-based classification algorithms.

Gradient boosting decision tree (GBDT) [56] is an iterative decision tree algorithm. The algorithm uses classification and regression decision trees (CART) [57] as the base classifier of the model. Therefore, it can not only perform classification but also handle regression analysis problems. At the beginning of the proposal, the researchers believed that GBDT has a strong generalization ability and can adapt to most common application scenarios.

Random forest (RF) [58] is an efficient learning algorithm based on Bagging. This algorithm also uses CART as the base classifier. Therefore, it, like GBDT, is able to handle both classification and regression problems. In addition, it is also an ensemble learning algorithm that can achieve parallelism and has strong generalization abilities.

Support vector machine (SVM) [59] is a commonly used machine learning algorithm. SVM has been widely used in classification and regression problems. The SVM algorithm maps the original data into a high-dimensional space, making it easier for instances to separate from each other in this space. In addition, SVM can reduce the generalization error as much as possible while minimizing structural risks, thereby effectively avoiding the model from falling into overfitting problems.

It should be noted that the ensemble strategies used by GBDT and RF are not the same. GBDT uses the decision tree to repeatedly learn the training data in series, while RF learns on the training set in parallel. Therefore, RF is able to achieve parallel computing, while GBDT cannot. Throughout the experiments, the classifiers we use are directly invoked from the scikit-learn library [60]. In order to be fair, in subsequent experiments, we only adjusted the parameters of the LMD-WNG instance selection algorithm, while the hyperparameters of the base classifier used the default hyperparameters in the corresponding library without any adjustment.

4.2 Baseline datasets

Throughout the experiments, we collected 30 datasets from the KEEL [61] databases. The meta information of the datasets is shown in Table 1 (arranged in ascending order of IR). In the field of data mining, most researchers believe that the smaller the number of instances, the more difficult it will be to classify. The main reason is that the model obtained by training on such a dataset may be biased. Therefore, researchers use the imbalance ratio to measure the skewness of instances. The imbalance ratio (IR) is an important indicator for measuring the distributional imbalance of datasets. IR refers to the ratio of the number of instances in the majority class (negative class) to the number of instances in the minority class (positive class) [55]. In subsequent experiments, the dataset with the largest imbalance ratio we used was poker-8-9_vs_5, which reached 82. The dataset with the smallest imbalance ratio, ecoli-0_vs_1, has an imbalance ratio of 1.86.

4.3 Evaluation metrics

Evaluation metrics are an important way to assess the performance of models. Accuracy (Acc) is the most common evaluation metric for classification problems, but it is unwise to use it directly for class-imbalanced problems. If you directly use accuracy to evaluate the base classifier, you cannot query the results of the minority instances (as described in Sect. 1).

Therefore, we need to use evaluation metrics that pay more attention to classification performance in minority instances. When studying class-imbalance problems, commonly used evaluation metrics include F1, Recall, AUC, G-mean, MCC, etc. The area under the receiver operating characteristic (ROC) curve (AUC) and the geometric mean (G-mean) are two common evaluation metrics. G-mean can be regarded as a variant of accuracy. During the calculation of G-mean, it is necessary to calculate the correct prediction numbers for different types of instances. When the classification performance of minority instances is poor, the G-mean will be very low or even zero. Therefore, G-mean is an evaluation metric that pays more attention to minority instances. In addition, it should be noted that in the process of calculating AUC [62], it is necessary to first draw the ROC curve and calculate the sum of all trapezoidal areas according to the change in the threshold. For binary class-imbalance problems, most classifiers directly output the class labels of instances. Therefore, when solving the AUC, the AUC value is calculated directly using the labels (two labels, 0 and 1, respectively) as probabilities. It should be noted that some researchers use probability as the output of the model. However, there are certain differences in the AUC obtained by these two calculation methods (but this does not indicate that there is an evaluation error in the literature). The current use of these two methods is confusing and does not facilitate the assessment of results. Therefore, it is recommended to use a unified AUC calculation method in the field of class-imbalance problem research in the future. Throughout the experiments, we still use labels as output and directly calculate the results of the AUC.

5 Parameter sensitivity analysis

Parameter sensitivity analysis is helpful to understand the degree of influence of hyperparameters on the uncertainty of model output. In our proposed LMD-WNG instance selection algorithm, the hyperparameter k needs to be manually set during the k-NNG construction stage.

The size of the hyperparameter k affects the result of the subsequent construction of the distance-weighted Laplacian matrix. When the hyperparameter k is larger, more neighboring instances are selected in the instance space, the value of the corresponding position in the newly formed Laplacian matrix is larger, and the matrix is denser. Conversely, when the value of k is smaller, the matrix is sparser. When the value of k is small, the eigenvalues of the instances located of the sparse region in the instance space are smaller, and they are easier to be selected. Conversely, instances located in dense areas of the instance space are more likely to be ignored. As the value of k increases, the changes in the dense area are not obvious. Instead, in the degree matrix, the values corresponding to the positions of instances in sparse regions become larger, causing these instances to be removed during the selection process as well. Therefore, according to the above analysis, we conclude that the size of the k value will affect the

Table 1 Meta information of datasets

Datasets	#Feature	#Majority	#Minority	#Class	Imbalanced ratio
ecoli-0_vs_1	7	143	77	2	1.86
pima	8	500	268	2	1.87
glass0	9	144	70	2	2.06
vehicle2	18	628	218	2	2.88
vehicle1	18	629	217	2	2.9
vehicle3	18	634	212	2	2.99
vehicle0	18	647	199	2	3.25
segment0	19	1979	329	2	6.02
glass6	9	185	29	2	6.38
yeast3	8	1321	163	2	8.1
ecoli3	7	301	35	2	8.6
yeast-2_vs_4	8	463	51	2	9.08
ecoli-0-6-7_vs_3-5	7	200	22	2	9.09
ecoli-0-1_vs_5	6	220	22	2	11
glass2	9	197	17	2	11.59
yeast-1_vs_7	7	429	30	2	14.3
abalone9-18	8	689	42	2	16.4
yeast-1-4-5-8_vs_7	8	663	30	2	22.1
yeast4	8	1433	51	2	28.1
winequality-red-4	11	1546	53	2	29.17
yeast-1-2-8-9_vs_7	8	917	30	2	30.57
yeast5	8	1440	44	2	32.73
abalone-17_vs_7-8-9-10	8	2280	58	2	39.31
abalone-21_vs_8	8	567	14	2	40.5
yeast6	8	1449	35	2	41.4
winequality-white-3_vs_7	11	880	20	2	44
abalone-19_vs_10-11-12-13	8	1590	32	2	49.69
winequality-white-3-9_vs_5	11	1457	25	2	58.28
abalone-20_vs_8-9-10	8	1890	26	2	72.69
poker-8-9_vs_5	10	2050	25	2	82

classification performance of the model. When the value of k is small, more instances are selected; on the contrary, as the value of k increases, fewer instances are selected. However, the degree of influence on the model needs further verification.

In order to further verify the influence of k value on model performance, in the sensitivity analysis part, we choose $k = 3$, 5, 7 and 9 for experiments. In our experiment, k should not take a larger value because, when the value of k is too large, the instances in the sparse area will be included, which is not conducive to the selection of instances. Due to space reasons, we only use AUC as the evaluation metric to analyze the effect of the hyperparameter k on the performance of both GBDT and RF classifiers. When $k = n$, the adjacency matrix with similarity is used for calculation, and the LMD-WNG at this time is similar to our previous work [49]. However, in previous work, we did not consider the number of connections between neighbors (i.e., we did not compute the degree matrix). In Fig. 3, the classification performance on mildly, moderately, and highly imbalanced datasets is given using GBDT. In Fig. 4, the classification performance on mildly, moderately, and highly imbalanced datasets is given using RF. Figure 5 shows the impact of hyperparameter k on SVM.

We first observe the experimental results of the LMD-WNG instance selection algorithm when using GBDT as the base classifier. When LMD-WNG is used to process mildly imbalanced datasets, the value of k has no obvious impact on the experimental results, as shown in Fig. 3a. We continue to observe the experimental results in Fig. 3b. It is not difficult to find that as the value of the hyperparameter k increases, the AUC value of the classifier fluctuates more obviously. We can observe with the naked eye that the fluctuations in AUC values are irregular; that is, the optimal k values are different on different datasets.

We can simply conclude that LMD-WNG is more sensitive to the hyperparameter k as the dataset imbalance ratio increases. The setting of parameters is more likely to affect the classification performance of the model. When we observe the experimental results in Fig. 3c, the experimental results confirm our conclusion, and the influence of hyperparameter k does not have any rules, showing a highly disordered state. When using LMD-WNG to cope with highly imbalanced datasets, the choice of hyperparameter k is more important because the performance of the model is more dependent on k.

Next, we proceed to analyze the parameter sensitivity of the LMD-WNG instance selection algorithm when using RF. The choice of hyperparameters did not cause serious performance differences when using the LMD-WNG algorithm together with RF on mildly imbalanced datasets. Taking a closer look at the results in Fig. 4a, we find that there is no obvious advantage for smaller k values. The main reason is that in the instance space, the difference in the number of instances of the majority class and the minority class is not obvious, and when the k-NN is used to search the instance space, the degree matrix of the instance does not increase significantly. Therefore, the LMD-WNG is not sensitive to the parameters at this time. However, LMD-WNG is more sensitive to the choice of hyperparameter k as the dataset imbalance ratio increases. For moderately imbalanced datasets (as shown in Fig. 4a) and highly imbalanced datasets (as shown in Fig. 4a), the overall performance fluctuation of the model is obviously irregular.

Next, observe the experimental results using the SVM shown in Fig. 5. We noticed that when the imbalance ratio (IR) of the dataset is low, the impact of different hyperparameters on the model's performance is not obvious. As the imbalance ratio increases, the impact of hyperparameter settings on the performance of the model gradually increases. However, for highly imbalanced datasets, the overall performance of the model fluctuates significantly and appears irregular. Therefore, we believe that the setting of hyperparameters is crucial to the performance of the LMD-WNG. If the hyperparameter settings are unreasonable, it can easily make it difficult for SVM to identify unknown minority instances. In addition, the performance of SVM depends on the number of support vectors in the training dataset. When the imbalance ratio (IR) is large, the existence of boundary instances may cause the support vectors to also show an obvious imbalance. If you blindly expand the parameters, it may cause the loss of support vectors near the boundary, causing the overall performance of the model to decline.

To sum up, for the problem of class imbalance, we also consider that the imbalance of class distribution is not the decisive factor affecting the performance of the model. From the experimental results, it can be seen that there are still significant differences in performance for different degrees of class-imbalance datasets. Therefore, for such datasets, we still need to explore the most serious data problems existing in the dataset and mine other difficult factors in the dataset to effectively solve the class-imbalance problem. In addition, the parameter sensitivity of our proposed LMD-WNG instance selection algorithm increases with the increase in the IR of the dataset and exhibits an irregular phenomenon. Therefore, in our model, the choice of hyperparameters becomes more important when dealing with datasets with high imbalance.

But we don't need to worry; there is only one hyperparameter in the LMD-WNG instance selection algorithm, and the optimal hyperparameter can be searched in the sample space by various methods. In the subsequent comparative experiments (Sect. 6), we perform fivefold cross-validation on the original dataset. In order to select the optimal hyperparameter k, a hierarchical partitioning method is used on the training set corresponding to each fold, and 80% of the instances are selected as the real training set, and the remaining 20% are used as the verification set. By selecting the hyperparameter k corresponding to the optimal G-mean (Sect. 4.3) on the validation set as the real hyperparameter for the comparison experiment.

6 Comparative experiments

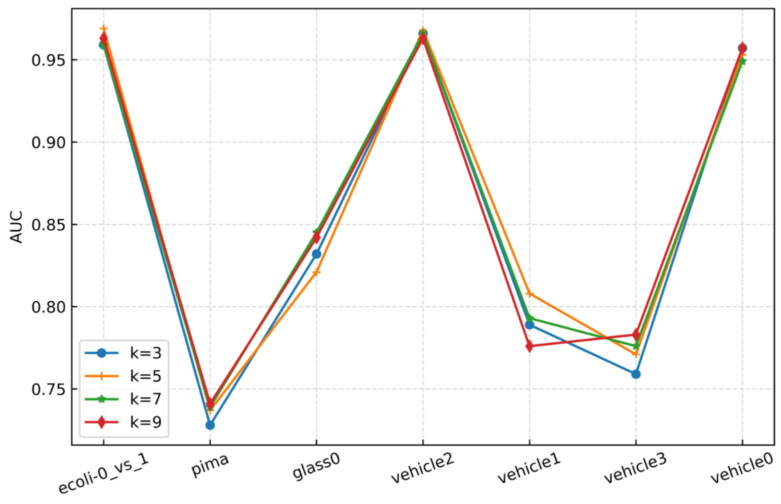
In this section, we will conduct a complete comparative experimental analysis to verify the effectiveness of the LMD-WNG algorithm by comparing it with state-of-the-art and classic undersampling and oversampling. It should be noted that throughout the experiment, the hyperparameter selection method in Sect. 5 is followed, the performance of the model was still verified using fivefold cross-validation, and the mean score was used as the final score of the model.

6.1 Baseline methods

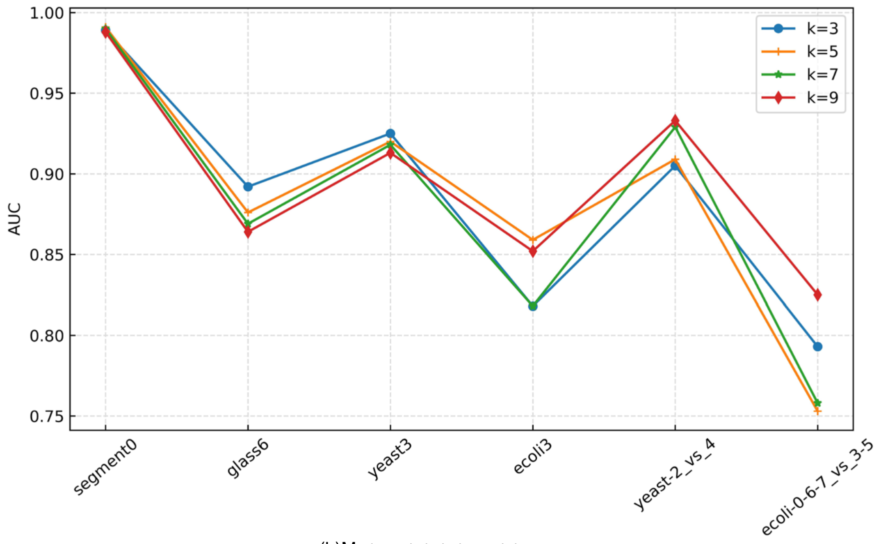
In this selection, we introduce some resampling and ensemble learning algorithms used in comparative experiments.

Random undersampling (RUS) is a classic undersampling method. This method only randomly removes majority instances until the minority and majority instances are balanced in the training set. On some datasets, the performance of the classifier has improved, but the random selection of instances in the algorithm increases the uncertainty of the algorithm.

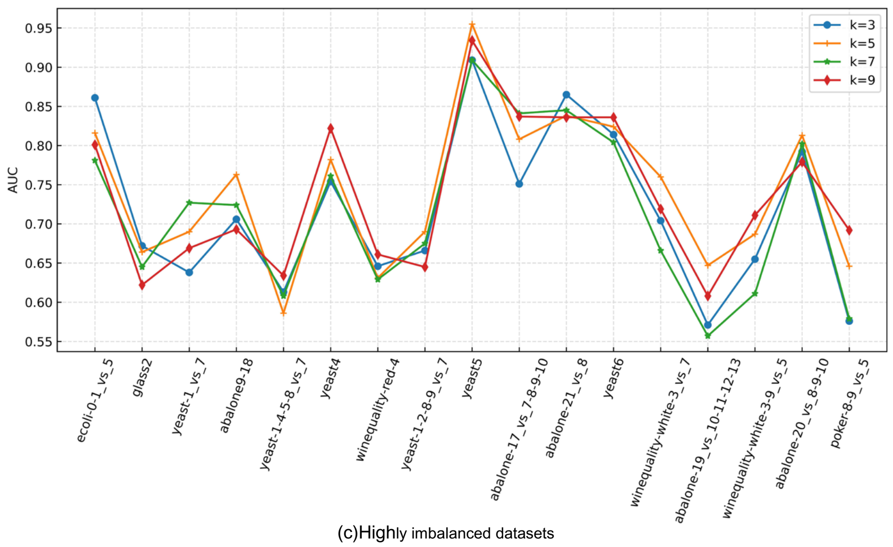
Fig. 3 The results of parameter sensitivity analysis (GBDT)



(a)Mildly imbalanced datasets



(b)Moderately imbalanced datasets



(c)Highly imbalanced datasets

Fig. 4 The results of parameter sensitivity analysis (RF)

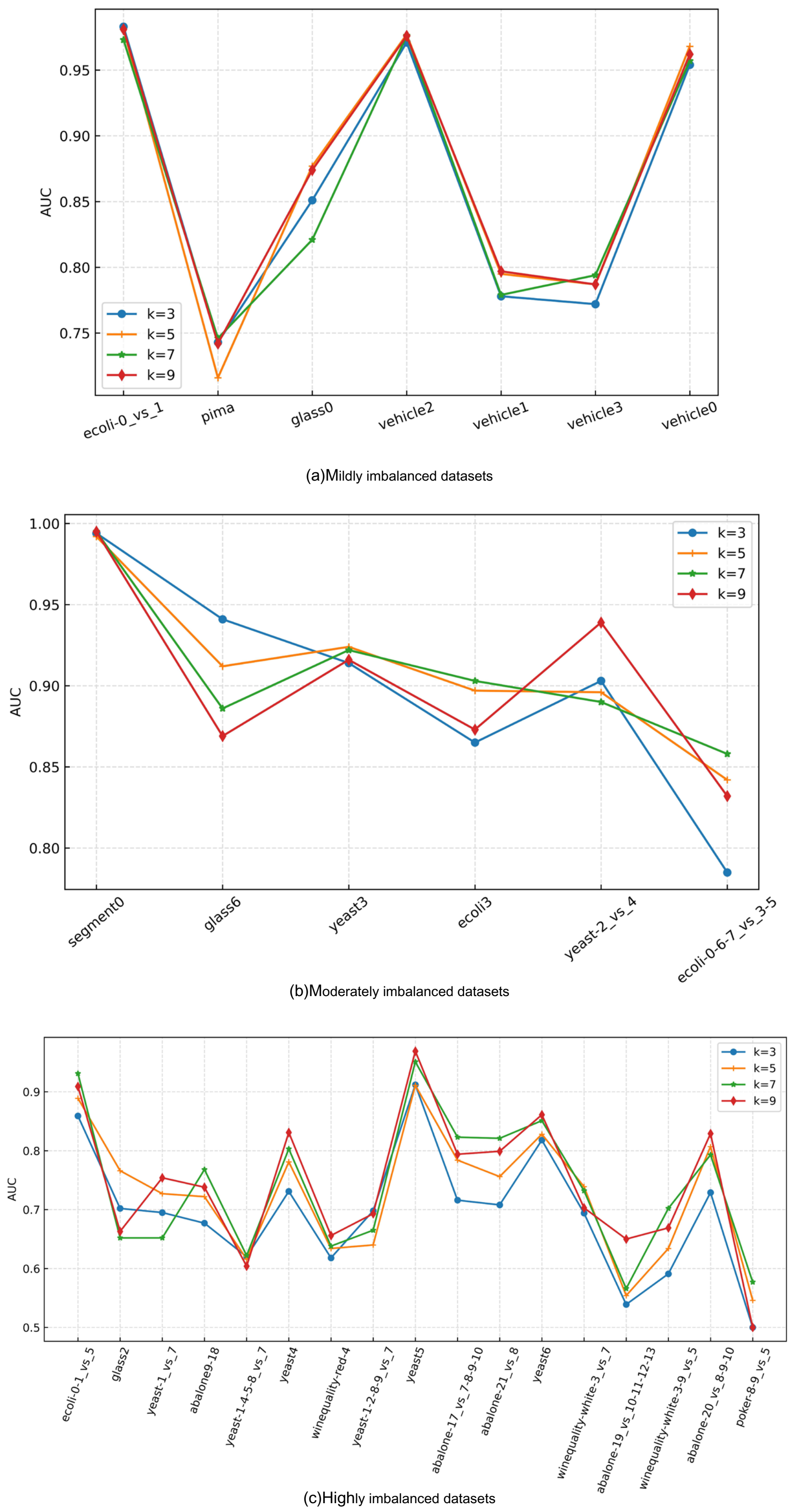
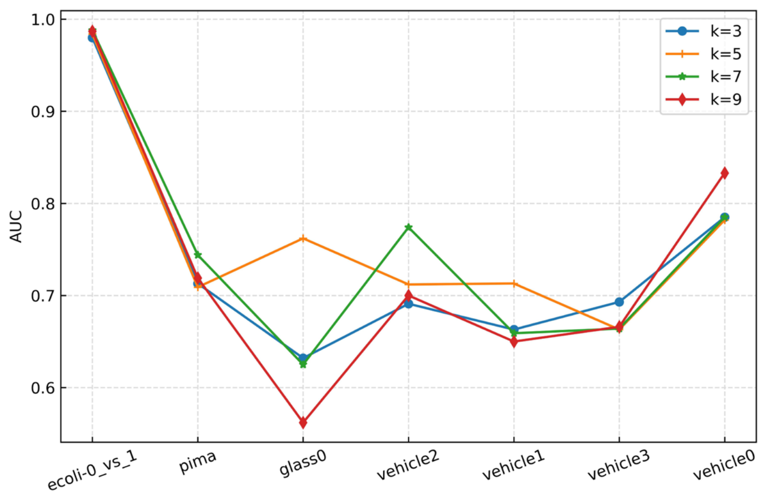
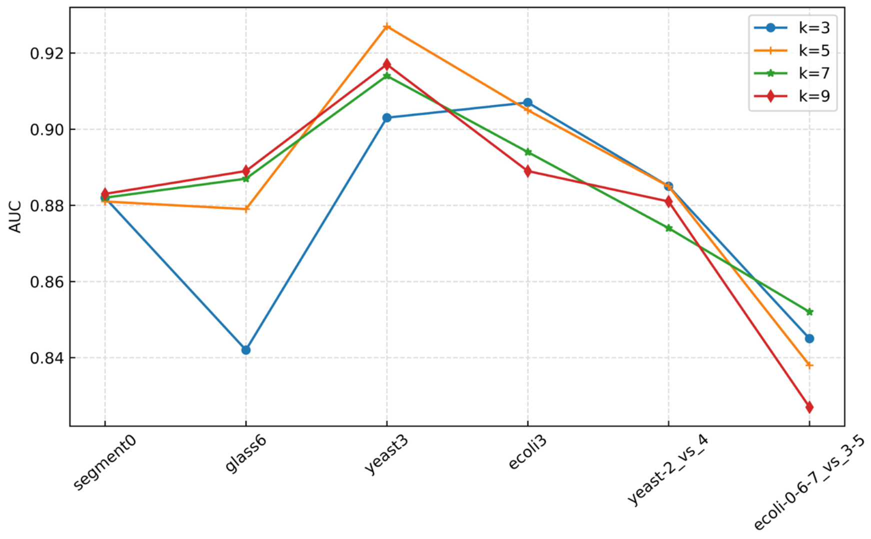


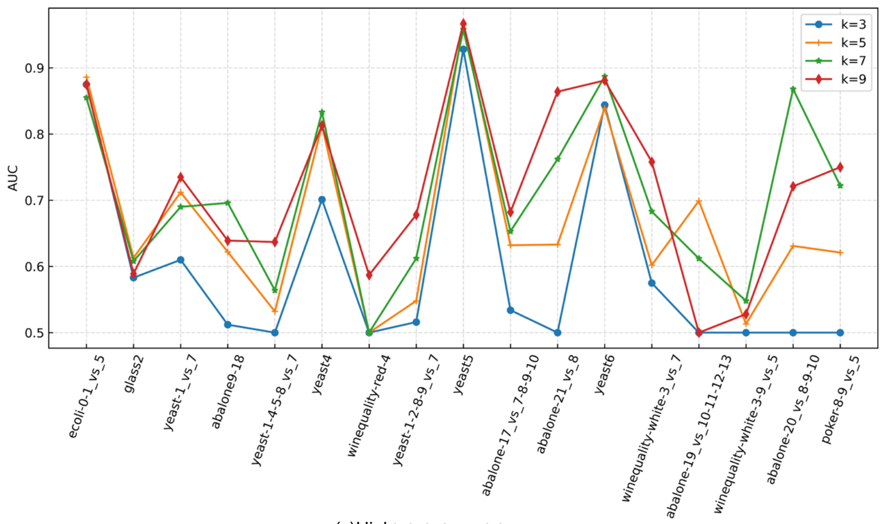
Fig. 5 The results of parameter sensitivity analysis (SVM)



(a)Mildly imbalanced datasets



(b)Moderately imbalanced datasets



(c)Highly imbalanced datasets

Synthetic minority oversampling technique (SMOTE) [41] is a classic method proposed to overcome the overfitting problem of random oversampling (ROS) methods. This method randomly selects an instance among the k-nearest-neighbor minority instances of the minority anchor instance and randomly synthetic minority instances in the space connecting the two instances.

Tomek links (TL) [63] is a classic class-overlap undersampling method considering the sample space. There are many ways to use this method, and you can choose different ways to process datasets according to actual needs. However, when faced with highly imbalanced datasets, the effect of directly using TL to process datasets is not obvious.

NB-TL [64] is an improved method that considers the problem of class-overlap. This method can expand the scope of classic TL to clean up overlapping regions in the instance space and improve the classification performance of TL.

CDSMOTE [65] uses a class decomposition to reduce the dominance of majority instances and uses oversampling techniques to enhance the feature representation of minority instances. The class decomposition strategy can prevent the loss of important information.

KNNOR [45] uses the k-nearest-neighbor technique to identify the position and compactness between the minority instance and the majority instance and determine the key areas that need enhancement. KNNOR is able to generate more reliable synthetic minority instances and maintain the model's robustness to noise.

MESA [66] is a novel ensemble learning framework trained using task-agnostic independent meta-samplers. This method involves resampling the dataset iteratively to obtain multiple base classifiers and form a cascade ensemble learning model.

It should be noted that CDSMOTE, KNNOR, and MESA are implemented based on the open source code provided by the author, and the code is improved according to the description in the literature, and its parameters adopt the best parameters in the literature. RUS, SMOTE, and TL are implemented according to the imbalanced_learn open source code library and use its default parameters. NB-TL is reproduced according to the details given in [64]. In [64], the author believes that the value of k has no obvious impact on the performance of the model. Therefore, the hyperparameters of NB-TL are set to $k = 5$.

6.2 Results and analysis

In this section, we use AUC [62] and G-mean as the evaluation metrics of the model, and the calculation process is shown in Sect. 4.3. Note that the AUC used in the experiments is calculated by directly outputting labels. In our experiments, we use the mean of the fivefold cross-validation as the final result of the model, and the cross-validation training set is nested and divided into 80% of the training set and 20% of the validation set. It should be noted that other state-of-the-art models are trained directly using cross-validated training sets. For state-of-the-art and classic comparison algorithms, we have briefly introduced them in Sect. 5.1. The results and open service code of the entire comparative experiment have been published on GitHub (https://github.com/cup-dq/LMD-WNG). In order to prove the effectiveness of the proposed LMD-WNG instance selection algorithm, we conducted comparative experiments on 30 publicly available datasets. Tables 2 and 3 show the classification performance of the two ensemble classifiers, GBDT, RF and SVM, respectively, when AUC and G-mean are used as evaluation metrics.

Observing the experimental results in Table 2 when using AUC as the evaluation metric, the mean value of the LMD-WNG instance selection algorithm on all datasets is better than other resampling algorithms, and the overall performance is comparable to other resampling techniques and has a competitive advantage. Judging from the number of best scores obtained, we observe that when GBDT is selected as the base classifier, LMD-WNG obtains the best scores on 15 datasets, accounting for 50% of the entire experimental dataset. MESA is the suboptimal model, achieving the best score on 7 datasets. And KNNOR and CDSMOTE achieved the best scores on four datasets, respectively. When RF is selected as the base classifier, LMD-WNG achieves the best score on 19 datasets, accounting for 63.3% of the entire experimental dataset. MESA is the suboptimal model, achieving the best scores on 7 datasets, accounting for 23.3% of the entire dataset. In addition, we observed that the mean score of LMD-WNG's AUC on all datasets is better than other comparison methods, and compared with RUS, using GBDT and RF improved by 3.08% and 2.73%, respectively.

When choosing to use SVM, the performance improvement of LMD-WNG is not obvious compared with other resampling methods. LMD-WNG achieves the best score on only 15 datasets, accounting for 50% of the entire dataset. Judging from the average values on all datasets, the overall performance of LMD-WNG still has certain competitive advantages. Moreover, according to the experimental results in Table 2, it can be seen that the oversampling technique does not achieve higher performance on datasets with a higher imbalance ratio. If you blindly increase the number of minority instances, it may lead to a decline in model performance or even overfitting problems.

G-mean is a variation evaluation metric based on accuracy. It can focus on the recognition effect of minority instances. When the accuracy of minority instances is higher, the overall G-mean of the model is higher. When G-mean is closer to 1, it means that the performance improvement of minority instances is more obvious. When the model has difficulty identifying minority instances, the G-mean of the model will drop significantly.

Next, we continue to observe the experimental results in Table 3 when using the G-mean as the evaluation metric. When using GBDT as the base classifier, LMD-WNG achieved the best scores on 12 datasets, accounting for 40% of the entire experimental dataset. Furthermore, MESA achieves the best scores on 7 datasets, and KNNOR achieves the best scores on 6 datasets. Simply looking at the number of datasets that achieved the best scores, LMD-WNG has comparable performance to MESA and KNNOR. However, for the mean score of the entire experimental dataset, the mean score of LMD-WNG is higher, with a performance improvement of 2.89% compared to RUS. For TL and NB-TL, the G-mean score of the model is significantly lower than other models. When the class distribution of the dataset is not very different, the performance gap between TL and NB-TL is not obvious. When the imbalance ratio of the dataset is very high and there are too few minority instances, using TL and NB-TL cannot effectively remove the redundant majority instances existing in the dataset, resulting in an insignificant performance improvement of the classifier. This is also a major drawback of only using TL to process the majority instances in overlapping areas.

When using RF as the base classifier, LMD-WNG achieved the best score on 16 datasets, accounting for 53.3% of the total number of experimental datasets. The three comparison methods of MESA, KNNOR, and CDSMOTE achieved the best performance on 6 datasets, 3 datasets, and 1 dataset, respectively. However, we note that the mean score of MESA is optimal over the entire experimental dataset, while the mean score of LMD-WNG is suboptimal. RF is an ensemble learning algorithm based on bagging. When the number of instances in the dataset is too small, a large number of repeated instances will appear in the training set of the base classifier. Further analyzing the principle of the LMD-WNG instance selection algorithm, we found that the reason for the abnormal performance of LMD-WNG is related to parameter settings. When the parameter selection is too large, the model may suffer from overfitting problems because it may be better on the training set but cannot identify the category of unknown instances. Therefore, for highly imbalanced datasets, a more appropriate parameter selection strategy is needed to determine the hyperparameters of LMD-WNG.

When SVM is selected for classification, LMD-WNG achieves the best score on 14 datasets, accounting for 46.67% of the entire dataset. Simply looking at the number of datasets with the best scores, the advantages of LMD-WNG are not obvious, and the overall performance of the model can be optimized through other strategies. In order to further discover the overall performance of each resampling technique on 30 datasets. We present in Figs. 6, 7, and 8 the mean scores of the two evaluation metrics, AUC and G-mean, when using GBDT, RF, and SVM, respectively.

First, observe the classification performance on all datasets when GBDT is used in Fig. 5. It is not difficult to see that when using AUC as the evaluation metric, the LMD-WNG instance selection algorithm has comparable performance to the three methods of MESA, KNNOR, and CDSMOTE, and has certain obvious advantages compared with other resampling techniques. However, when using G-mean, the performance improvement of the LMD-WNG instance selection algorithm is more obvious.

When we choose RF, the experimental results of all compared algorithms are shown in Fig. 7. We observe the experimental results when AUC is used in Fig. 7a, and the proposed LMD-WNG instance selection algorithm outperforms other comparative algorithms and has a strong competitive advantage compared with ensemble learning techniques. Furthermore, when observing the use of G-mean as the evaluation metric, we can draw the same conclusion as when using GBDT: that our proposed LMD-WNG is an excellent algorithm.

Next, we continue to observe the classification performance of different resampling methods when using the SVM shown in Fig. 8. Observing the overall performance evaluated using AUC in Fig. 8b, we notice that the performance differences of different resampling methods are not obvious. However, when G-mean is used to evaluate the performance, there are significant fluctuations in the performance of the TL method. Analyzing the principle of TL, we found that TL forcibly deletes the majority instances adjacent to the boundary minority instances, expanding the classification boundary of the classifier on the dataset. However, for SVM, its classification performance mainly comes from the support vectors on the boundary. Therefore, the TL-based resampling technique does not necessarily improve the overall performance of using SVM.

Although we analyzed the LMD-WNG instance selection algorithm from different angles, the performance on different datasets or the overall experimental dataset is better, but we do not know whether it is statistically significant. Therefore, in Sect. 6.3, we perform statistical tests on all compared algorithms using the Friedman $1*N$ test and Holm's post hoc test on all datasets. From a statistical point of view, verify whether there is statistical significance between the LMD-WNG sample selection strategy and other methods.

Table 2 The performance of three classifiers, GBDT, RF, and SVM (using AUC)

Datasets	RUS			SMOTE			TL			NB-TL		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0_vs_1	0.973	0.967	0.984	0.969	0.978	0.984	0.966	0.975	0.984	0.966	0.974	0.984
pima	0.733	0.739	0.714	0.733	0.729	0.729	0.734	0.728	0.698	0.735	0.743	0.706
glass0	0.824	0.856	0.673	0.813	0.862	0.684	0.837	0.859	0.5	0.813	0.842	0.5
vehicle2	0.967	0.976	0.699	0.968	**0.985**	0.766	0.972	0.983	0.546	0.971	0.980	0.547
vehicle1	0.782	0.773	0.656	0.755	0.737	0.662	0.728	0.711	0.583	0.798	0.777	0.654
vehicle3	0.743	0.763	0.662	0.751	0.733	0.659	0.701	0.693	0.5	0.771	0.766	0.668
vehicle0	0.943	0.960	0.784	0.959	0.959	0.812	0.951	0.958	0.601	0.958	0.966	0.713
segment0	0.991	0.994	0.872	0.99	0.995	0.847	0.986	0.989	0.846	0.987	0.994	0.853
glass6	0.909	0.932	0.638	0.908	0.911	**0.887**	0.883	0.878	0.561	0.881	0.911	0.5
yeast3	0.897	0.926	0.919	0.896	0.887	0.915	0.886	0.857	0.863	0.907	0.886	0.881
ecoli3	0.835	0.871	0.895	0.795	0.806	0.884	0.785	0.785	0.812	0.845	0.818	0.902
yeast-2_vs_4	0.923	0.929	0.875	0.872	0.898	0.896	0.862	0.853	0.801	0.863	0.871	0.817
ecoli-0-6-7_vs_3-5	0.815	0.802	0.861	0.812	0.805	0.842	0.831	0.835	0.835	0.811	0.835	0.828
ecoli-0-1_vs_5	0.881	0.909	0.907	0.908	0.895	0.889	0.818	0.868	0.898	0.843	0.845	0.898
glass2	0.617	0.678	0.583	0.664	0.637	0.601	0.502	0.548	0.5	0.531	0.548	0.512
yeast-1_vs_7	0.752	0.712	**0.752**	0.667	0.674	0.697	0.642	0.613	0.5	0.64	0.629	0.621
abalone9-18	0.721	0.715	0.718	0.716	0.742	0.747	0.612	0.581	0.5	0.633	0.581	0.5
yeast-1-4-5-8_vs_7	0.574	0.613	0.629	0.611	0.549	0.645	0.53	0.5	0.5	0.528	0.500	0.5
yeast4	0.804	0.797	0.812	0.774	0.709	0.83	0.611	0.586	0.698	0.656	0.604	0.758
winequality-red-4	0.613	0.651	0.579	0.608	0.58	0.567	0.515	0.518	0.5	0.524	0.500	0.5
yeast-1-2-8-9_vs_7	0.641	0.689	0.691	0.611	0.601	0.701	0.543	0.598	0.5	0.56	0.564	0.648
yeast5	0.950	0.943	0.96	0.903	0.917	0.942	0.837	0.802	0.625	0.871	0.862	0.792
abalone-17_vs_7-8-9-10	0.799	0.817	0.697	0.791	0.709	0.749	0.615	0.544	0.5	0.598	0.542	0.5
abalone-21_vs_8	0.838	0.757	0.759	0.823	0.814	**0.881**	0.732	0.664	0.5	0.728	0.731	0.85
yeast6	0.820	0.853	0.881	0.818	0.793	0.867	0.709	0.670	0.697	0.751	0.711	0.866
winequality-white-3_vs_7	0.718	0.731	0.634	0.640	0.543	**0.769**	0.646	0.650	0.619	0.598	0.625	0.593
abalone-19_vs_10-11-12-13	0.605	0.648	0.598	0.628	0.594	0.696	0.531	0.500	0.5	0.509	0.500	0.5
winequality-white-3-9_vs_5	0.664	0.684	**0.632**	0.595	0.535	0.508	0.574	0.539	0.5	0.615	0.539	0.516
abalone-20_vs_8-9-10	0.774	0.792	0.676	0.792	0.686	0.869	0.577	0.500	0.5	0.617	0.521	0.5
poker-8-9_vs_5	0.650	0.566	0.643	0.536	0.500	0.75	0.500	0.500	0.5	0.500	0.500	0.551
Average	0.7919	0.8014	0.7461	0.7769	0.7588	0.7758	0.7205	0.7095	0.6222	0.7336	0.7222	0.6719

Datasets	CDSMOTE			KNNOR			MESA			LMD-WNG		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0_vs_1	0.969	0.977	0.955	0.962	0.966	0.981	**0.973**	0.973	0.981	0.969	**0.983**	**0.988**
pima	0.74	0.751	0.715	**0.747**	**0.764**	0.721	0.731	0.722	0.719	0.741	0.746	**0.744**
glass0	0.817	0.824	0.5	0.792	0.789	0.691	0.827	0.831	0.754	**0.855**	**0.877**	**0.762**
vehicle2	0.961	0.978	0.563	0.929	0.927	0.748	0.954	0.966	0.766	**0.974**	0.977	**0.774**
vehicle1	0.773	**0.8**	0.619	0.698	0.696	0.681	0.792	0.768	0.701	**0.818**	0.797	**0.713**
vehicle3	0.772	0.77	0.5	0.684	0.699	0.673	0.773	0.778	**0.721**	**0.783**	**0.794**	0.693
vehicle0	0.956	0.965	0.741	0.882	0.923	0.813	0.951	0.956	0.808	**0.965**	**0.968**	**0.833**
segment0	0.991	0.994	0.847	0.981	0.979	0.853	0.988	0.988	**0.965**	**0.991**	**0.995**	0.882
glass6	**0.913**	0.903	0.867	0.887	0.879	0.877	0.906	0.907	0.871	0.892	**0.941**	0.885
yeast3	0.912	0.911	0.881	0.927	**0.931**	0.879	0.916	0.919	0.899	**0.935**	0.924	**0.927**
ecoli3	0.851	0.879	0.893	**0.885**	0.857	0.887	0.878	0.875	0.869	0.859	**0.903**	**0.907**
yeast-2_vs_4	0.908	0.911	0.835	0.938	0.92	**0.896**	**0.958**	0.923	0.866	0.933	**0.939**	0.885
ecoli-0-6-7_vs_3-5	0.842	0.842	**0.902**	0.772	0.781	0.842	**0.861**	0.847	0.823	0.825	**0.858**	0.852

Table 2 (continued)

Datasets	CDSMOTE			KNNOR			MESA			LMD-WNG		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0-1_vs_5	0.839	0.895	**0.916**	0.82	0.882	0.895	**0.913**	0.839	0.855	0.861	**0.931**	0.886
glass2	0.637	0.664	0.5	**0.679**	0.702	0.594	0.674	**0.799**	0.583	0.672	0.766	**0.613**
yeast-1_vs_7	**0.761**	0.622	0.592	0.697	0.748	0.711	0.694	0.731	0.692	0.727	**0.754**	0.735
abalone9-18	0.728	0.701	0.532	0.745	0.692	0.749	0.703	0.734	**0.758**	**0.763**	**0.768**	0.696
yeast-1-4-5-8_vs_7	**0.675**	0.580	0.593	0.643	0.604	0.638	0.650	0.590	**0.663**	0.634	**0.622**	0.637
yeast4	0.787	0.746	0.785	0.803	0.830	0.818	0.822	0.807	0.806	**0.822**	**0.831**	**0.833**
winequality-red-4	0.601	0.584	0.5	0.650	0.645	0.552	**0.664**	0.637	0.525	0.661	**0.656**	**0.587**
yeast-1-2-8-9_vs_7	0.63	0.671	0.662	0.678	0.693	0.696	**0.728**	**0.748**	**0.701**	0.690	0.698	0.678
yeast5	0.936	0.936	0.826	0.956	0.961	0.933	**0.962**	0.944	0.953	0.955	**0.969**	**0.967**
abalone-17_vs_7-8-9-10	0.823	0.763	0.541	0.801	0.744	0.725	0.808	0.818	**0.755**	**0.841**	**0.823**	0.682
abalone-21_vs_8	0.857	0.822	0.86	0.832	0.766	0.772	0.859	**0.823**	0.84	**0.865**	0.821	0.864
yeast6	0.839	0.816	0.856	**0.867**	0.842	0.861	0.842	0.842	0.833	0.836	**0.861**	**0.887**
winequality-white-3_vs_7	0.626	0.564	0.747	0.748	0.681	0.757	0.748	0.699	0.768	**0.760**	**0.739**	0.758
abalone-19_vs_10-11-12-13	0.626	0.618	0.589	0.616	0.617	0.698	0.604	**0.655**	0.678	**0.647**	0.650	**0.699**
winequality-white-3-9_vs_5	0.628	0.529	0.567	0.660	0.658	0.541	0.660	**0.753**	0.554	**0.711**	0.702	0.548
abalone-20_vs_8-9-10	**0.825**	0.759	0.835	0.778	0.795	0.752	0.802	**0.846**	0.827	0.813	0.829	**0.868**
poker-8-9_vs_5	0.533	0.562	**0.789**	0.500	0.587	0.768	0.655	**0.631**	0.676	**0.692**	0.577	0.75
Average	0.7919	0.7779	0.7169	0.7852	0.7853	0.7667	0.8099	0.8116	0.7739	**0.8163**	**0.8233**	**0.7844**

6.3 Nonparametric statistical tests

In order to further analyze the statistical difference between the proposed LMD-WNG instance selection method and other resampling or ensemble learning methods on all datasets, we use the Friedman 1**N* test and Holm's post hoc test to conduct statistical analysis on the model [67]. We first used the Friedman test to calculate the ranking of different resampling techniques on all datasets and then used Holm's post hoc test to calculate the unadjusted *p* values and adjusted *p* values (APVs). We consider statistical significance between resampling techniques when the *p* value is less than 0.05. Tables 4, 5, and 6 show the Friedman 1**N* test and Holm's post hoc test results using GBDT, RF, and SVM, respectively.

The results of nonparametric statistical tests using GBDT, AUC, and G-mean as evaluation metrics are given in Table 4. We observe the results given in Table 4. We believe that the performance of the proposed LMD-WNG instance selection algorithm and MSEA are approximate, and there is no statistical significance between them. We reject the hypothesis of a statistically significant difference in their performance. Furthermore, LMD-WNG outperforms other comparison algorithms with significant statistical differences.

Random forest is a very robust ensemble learning algorithm. RF has strong generalization ability for different problems. The statistical results using RF are shown in Table 5. We note that both the Friedman ranking of the resampling technique and the results using Holm's post hoc test show the advancement of LMD-WNG. Therefore, experimental results on all datasets show that LMD-WNG outperforms other resampling techniques with statistical differences.

Table 6 shows the statistical test results using SVM. According to the results of Friedman ranking and Holm's post hoc test, we believe that the performance of LMD-WNG and SMOTE is similar. SMOTE is an effective method to increase the number of minority instances. During the processing of KNNOR and CDSMOTE, it is necessary to avoid boundary instances from participating in the synthesis of new minority instances. Therefore, when faced with highly imbalanced problems, KNNOR and CDSMOTE may suffer from the problem of a small number of instances in the boundary region. However, more pseudo-minority instances may gather near minority instances far from the boundary, causing the performance of the classifier to decline.

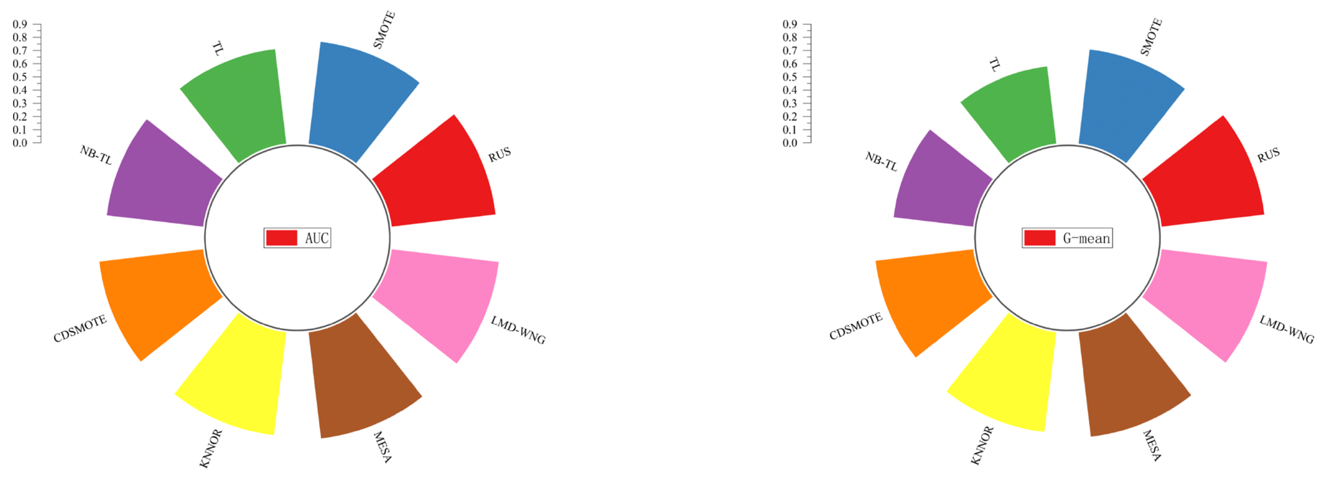
Table 3 The performance of three classifiers, GBDT, RF, and SVM (using G-mean)

Datasets	RUS			SMOTE			TL			NB-TL		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0_vs_1	0.942	0.958	0.983	0.969	0.971	0.983	0.966	0.964	0.983	0.965	0.963	0.983
pima	0.731	0.737	0.713	0.730	0.725	0.727	0.730	0.724	0.678	0.730	0.739	0.705
glass0	0.821	0.855	0.586	0.812	0.862	0.603	0.834	0.857	0	0.809	0.841	0
vehicle2	0.967	0.975	0.673	0.967	**0.985**	0.751	0.971	0.983	0.292	**0.971**	0.980	0.306
vehicle1	0.781	0.771	0.652	0.752	0.729	0.658	0.715	0.695	0.536	0.796	0.777	0.647
vehicle3	0.74	0.761	0.657	0.748	0.725	0.657	0.684	0.667	0	0.769	0.765	0.663
vehicle0	0.935	0.960	0.754	**0.959**	0.942	0.79	0.951	0.958	0.439	0.958	0.965	0.68
segment0	0.991	0.994	0.865	0.99	0.995	0.833	0.985	0.989	0.832	0.987	0.994	0.84
glass6	0.897	0.928	0.504	0.904	0.901	**0.8853**	0.871	0.856	0.159	0.868	0.901	0
yeast3	0.896	0.925	0.912	0.892	0.884	0.911	0.880	0.849	0.854	0.905	0.882	0.876
ecoli3	0.831	0.868	0.892	0.769	0.796	0.881	0.738	0.759	0.796	0.828	0.802	0.9
yeast-2_vs_4	0.923	0.927	0.872	0.866	0.892	0.892	0.850	0.829	0.768	0.851	0.859	0.788
ecoli-0-6-7_vs_3-5	0.808	0.787	0.851	0.779	0.777	0.831	0.808	0.812	0.814	0.784	0.816	0.791
ecoli-0-1_vs_5	0.871	0.906	0.902	**0.903**	0.878	0.877	0.793	0.836	0.886	0.819	0.797	0.886
glass2	0.533	0.663	0.402	0.525	0.432	0.467	0.095	0.211	0	0.203	0.213	0.285
yeast-1_vs_7	**0.746**	0.708	**0.745**	0.6	0.605	0.668	0.531	0.37	0	0.538	0.384	0.52
abalone9-18	0.715	0.711	0.705	0.614	0.698	0.734	0.423	0.304	0	0.459	0.353	0
yeast-1-4-5-8_vs_7	0.563	0.600	0.622	0.513	0.227	0.641	0.162	0	0	0.162	0	0
yeast4	0.803	0.794	0.804	0.753	0.650	0.825	0.465	0.41	0.673	0.565	0.445	0.727
winequality-red-4	0.609	**0.642**	0.548	0.512	0.408	0.546	0.123	0.085	0	0.152	0	0
yeast-1-2-8-9_vs_7	0.625	0.688	**0.681**	0.431	0.413	0.679	0.196	0.393	0	0.276	0.278	0.503
yeast5	0.949	0.942	0.959	0.898	0.910	0.941	0.821	0.762	0.499	0.857	0.849	0.761
abalone-17_vs_7-8-9-10	0.793	0.816	0.675	0.772	0.648	0.746	0.475	0.261	0	0.433	0.261	0.5
abalone-21_vs_8	0.831	0.661	0.744	0.797	0.706	**0.872**	0.604	0.429	0	0.602	0.513	0.831
yeast6	0.814	0.849	0.879	0.791	0.758	0.857	0.644	0.563	0.611	0.705	0.621	0.864
winequality-white-3_vs_7	0.699	**0.708**	0.464	0.409	0.198	**0.742**	0.539	0.415	0.437	0.341	0.373	0.265
abalone-19_vs_10-11-12-13	**0.601**	**0.643**	0.355	0.551	0.447	0.679	0.163	0	0	0.075	0	0
winequality-white-3-9_vs_5	0.66	0.675	**0.533**	0.419	0.178	0.243	0.356	0.179	0	0.429	0.179	0.263
abalone-20_vs_8-9-10	0.759	0.787	0.662	0.742	0.536	0.86	0.304	0	0	0.371	0.089	0
poker-8-9_vs_5	0.635	0.556	0.636	0.178	0	0.715	0	0	0	0	0	0.151
Average	0.7823	0.7932	0.7077	0.7182	0.6625	0.7498	0.5892	0.5387	0.3419	0.6069	0.5546	0.4912

Datasets	CDSMOTE			KNNOR			MESA			LMD-WNG		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0_vs_1	0.968	0.977	0.954	0.961	0.966	0.981	0.962	0.973	0.981	**0.969**	**0.983**	**0.987**
pima	0.738	0.749	0.696	0.736	**0.753**	0.71	0.731	0.721	0.707	**0.740**	0.744	**0.727**
glass0	0.814	0.821	0	0.788	0.786	0.613	0.824	0.828	0.746	**0.843**	**0.876**	**0.752**
vehicle2	0.961	0.978	0.298	0.929	0.926	0.727	0.954	0.965	0.751	0.963	0.976	**0.76**
vehicle1	0.770	**0.798**	0.512	0.695	0.695	0.68	0.791	0.767	**0.697**	**0.804**	0.793	0.692
vehicle3	0.768	0.770	0	0.683	0.693	0.656	0.772	0.777	**0.715**	**0.778**	**0.789**	0.672
vehicle0	0.956	0.964	0.723	0.880	0.920	0.791	0.951	0.956	0.784	0.956	**0.968**	**0.816**
segment0	0.991	0.994	0.833	0.981	0.979	0.826	0.988	0.988	**0.964**	**0.991**	**0.995**	0.877
glass6	0.896	0.893	0.862	0.884	0.873	0.875	**0.904**	0.901	0.866	0.890	**0.937**	0.862
yeast3	0.911	0.910	0.872	**0.926**	**0.931**	0.873	0.916	0.918	0.898	0.925	0.924	**0.916**
ecoli3	0.841	0.878	0.886	**0.873**	0.852	0.884	0.872	0.873	0.866	0.855	**0.902**	**0.905**
yeast-2_vs_4	0.903	0.907	0.808	0.936	0.918	**0.892**	**0.947**	0.921	0.851	0.931	**0.938**	0.881
ecoli-0-6-7_vs_3-5	0.824	0.825	**0.894**	0.763	0.764	0.829	**0.852**	0.838	0.811	0.814	**0.849**	0.843

Table 3 (continued)

Datasets	CDSMOTE			KNNOR			MESA			LMD-WNG		
	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM	GBDT	RF	SVM
ecoli-0-1_vs_5	0.825	0.878	0.883	0.801	0.872	0.886	0.901	0.807	0.846	0.859	**0.927**	**0.908**
glass2	0.514	0.553	0	**0.661**	0.599	0.488	0.648	**0.793**	0.426	0.645	0.763	**0.541**
yeast-1_vs_7	0.730	0.558	0.588	0.690	0.746	0.68	0.676	0.723	0.676	0.717	**0.747**	0.714
abalone9-18	0.631	0.662	0.164	0.734	0.655	**0.736**	0.615	0.733	0.735	**0.752**	**0.762**	0.671
yeast-1-4-5-8_vs_7	0.629	0.344	0.485	0.63	0.574	0.634	**0.648**	0.556	**0.648**	0.623	**0.611**	0.619
yeast4	0.771	0.715	0.777	0.801	**0.826**	0.813	**0.821**	0.806	0.795	0.820	0.822	0.826
winequality-red-4	0.528	0.432	0	0.645	0.640	0.532	**0.654**	0.625	0.183	0.650	0.641	**0.556**
yeast-1-2-8-9_vs_7	0.506	0.584	0.641	0.675	0.685	0.663	**0.717**	**0.741**	0.672	0.660	0.630	0.656
yeast5	0.934	0.933	0.807	**0.956**	0.961	0.932	0.952	0.943	0.952	0.954	**0.968**	**0.966**
abalone-17_vs_7-8-9-10	0.815	0.735	0.257	0.798	0.742	0.719	0.794	0.816	**0.752**	**0.837**	**0.822**	0.675
abalone-21_vs_8	0.845	0.704	0.835	0.822	0.738	0.765	0.851	**0.805**	0.818	**0.857**	0.704	0.858
yeast6	0.821	0.795	0.84	**0.861**	0.837	0.851	0.836	0.837	0.809	0.821	**0.852**	**0.883**
winequality-white-3_vs_7	0.401	0.297	0.708	**0.740**	0.670	0.73	0.660	0.672	0.741	0.730	0.690	0.741
abalone-19_vs_10-11-12-13	0.571	0.524	0.563	0.581	0.591	0.681	0.577	0.642	0.676	0.594	0.592	**0.684**
winequality-white-3-9_vs_5	0.480	0.177	0.334	0.627	0.648	0.1282	0.645	**0.748**	0.389	**0.694**	0.642	0.239
abalone-20_vs_8-9-10	0.789	0.721	0.821	0.761	0.793	0.716	0.792	**0.842**	0.82	**0.801**	0.824	**0.86**
poker-8-9_vs_5	0.178	0.533	0.707	0	0.563	**0.74**	0.648	**0.611**	0.652	**0.674**	0.404	0.715
Average	0.7436	0.7203	0.5916	0.7606	0.7732	0.7344	0.7966	**0.8042**	0.7409	**0.8049**	0.8025	**0.7601**



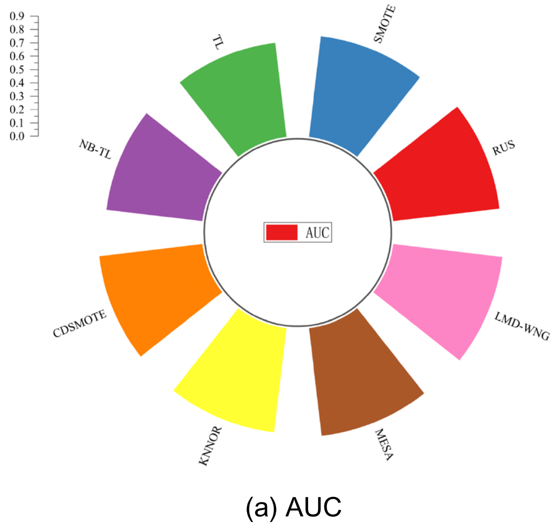
(a) AUC (b) G-mean

Fig. 6 Mean results on all datasets (using GBDT)

6.4 Global analysis

Combining the experimental results shown in Tables 2 and 3, Figs. 5 and 6, and the analysis results using the Friedman 1*N test and the Nemenyi post hoc test shown in Figs. 7 and 8, we can draw the following conclusions:

1. According to the parameter analysis results in Sect. 5, we can conclude that the LMD-WNG instance selection algorithm becomes more sensitive to the hyperparameter k as the IR of the dataset increases. Therefore, more appropriate parameter selection and optimization methods need to be considered.



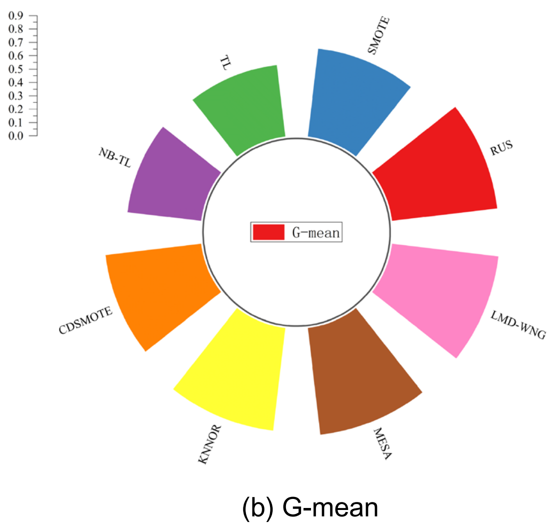
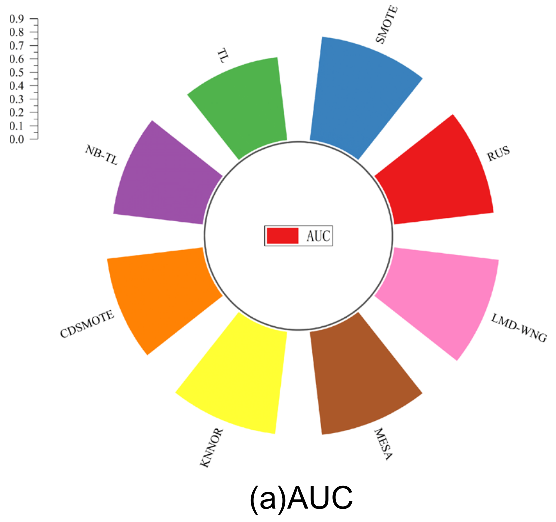


Fig. 7 Mean results on all datasets (using RF)



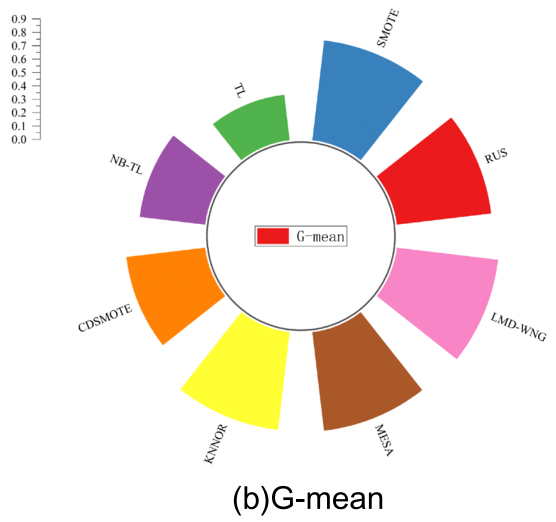


Fig. 8 Mean results on all datasets (using SVM)

2. According to the comparative experimental results in Sect. 6, we found that when the IR of the dataset increases, the performance of the oversampling method will fluctuate significantly and degrade continuously with the increase of the IR. Therefore, we believe that when the dataset is highly imbalanced, data augmentation techniques that can expand the minority instance area should be selected as much as possible or used together with undersampling techniques.
3. We use datasets with a wide range of IRs. Therefore, we believe that the LMD-WNG instance selection algorithm can adapt to most datasets with imbalanced ratios.
4. According to the analysis results of Friedman 1*N test and Nemenyi post hoc test, we believe that the LMD-WNG instance selection algorithm is superior or partially superior to other resampling or ensemble learning techniques. The LMD-WNG instance selection algorithm is an instance selection method with competitive advantages.

In addition, oversampling techniques fluctuated considerably across all datasets throughout the experiment. Therefore, we briefly analyze the reasons why the performance of the oversampling technique decreases as the IR of the dataset increases. When the imbalance in the dataset is high, there are fewer instances of the minority class in the dataset. If you blindly use the oversampling algorithm to increase the minority instances, more pseudo-instances are bound to be needed to balance the training dataset. When there are too many pseudo-instances in the training dataset, the density of the real minority instances in the

Table 4 Results and APVs of Friedman ranking and Holm's post hoc test (using GBDT)

Algorithms	AUC			Algorithms	G-mean		
	Friedman ranking	Unadjusted *p* value	APVs (Holm)		Friedman ranking	Unadjusted *p* value	APVs (Holm)
LMD-WNG	2.0500	–	–	LMD-WNG	2.2833	–	–
MESA	3.0167	0.126405	0.126405	MESA	2.9500	0.291841	0.291841
CDSMOTE	3.7333	0.007777	0.015555	RUS	4.0167	0.006132	0.01328
RUS	4.2000	0.000675	0.002026	CDSMOTE	4.0833	0.004427	0.01328
SMOTE	5.0000	0.000003	0.000012	KNNOR	4.3333	0.001190	0.004759
KNNOR	5.0833	0.000002	0.000008	SMOTE	5.2667	0.000002	0.000012
NB-TL	6.2167	0	0	NB-TL	6.2333	0	0
TL	6.7000	0	0	TL	6.8333	0	0

Table 5 Results and APVs of Friedman ranking and Holm's post hoc test (using RF)

Algorithms	AUC			Algorithms	G-mean		
	Friedman ranking	Unadjusted *p* value	APVs (Holm)		Friedman ranking	Unadjusted *p* value	APVs (Holm)
LMD-WNG	1.5833	–	–	LMD-WNG	2.0000	–	–
MESA	3.7167	0.000743	0.001112	RUS	3.4167	0.025094	0.046866
RUS	3.7667	0.000556	0.001112	MESA	3.4333	0.023433	0.046866
CDSMOTE	4.4167	0.000007	0.000022	KNNOR	4.4000	0.000148	0.000443
KNNOR	4.8333	0	0.000001	CDSMOTE	4.5000	0.000077	0.000309
SMOTE	5.0833	0	0	SMOTE	5.3667	0	0.000001
NB-TL	5.9167	0	0	NB-TL	6.0333	0	0
TL	6.6833	0	0	TL	6.8500	0	0

Table 6 Results and APVs of Friedman ranking and Holm's post hoc test (using SVM)

Algorithms	AUC			Algorithms	G-mean		
	Friedman ranking	Unadjusted *p* value	APVs (Holm)		Friedman ranking	Unadjusted *p* value	APVs (Holm)
LMD-WNG	2.2833	–	–	LMD-WNG	2.2333	–	–
SMOTE	3.2167	0.140017	0.140017	SMOTE	3.1667	0.140017	0.140017
KNNOR	3.7667	0.019009	0.038018	MESA	3.8167	0.012298	0.036895
MESA	3.8667	0.012298	0.036895	RUS	3.8167	0.012298	0.036895
RUS	4.0000	0.006642	0.026567	KNNOR	3.9333	0.007190	0.028758
CDSMOTE	5.3000	0.000002	0.000009	CDSMOTE	5.7000	0	0
NB-TL	6.1833	0	0	NB-TL	5.9833	0	0
TL	7.3833	0	0	TL	7.3500	0	0

training dataset will be reduced, so the classifiers will focus on the pseudo-minority instances.

However, it has to be said that GBDT is a boosting-based ensemble learning technique. When there are hard-to-learn instances in the training dataset, it iterates until it learns correctly. If we introduce too many pseudo-instances in the dataset, and these pseudo-instances are not real instances, it is likely to bias the GBDT model performance toward pseudo-instances. Therefore, when the dataset is highly imbalanced and GBDT needs to be used as the base classifier, we have two suggestions: On the one hand, we can consider increasing minority instances with oversampling techniques that can expand the minority class region. But such techniques are also likely to synthesize more pseudo-instances in the majority class region, leading to a decrease in the performance of the model. On the other hand, we can avoid generating more minority instances or introducing more noise by combining oversampling techniques with instance selection or undersampling techniques to reduce the number or density of majority instances while generating synthetic instances.

When we choose RF, we will also face the same problem. Although the pseudo-minority instances generated by using the oversampling technique will not directly affect the learning of the entire classifier like using GBDT. However, RF is a bagging-based ensemble learning technique that requires bootstrap sampling of the training dataset during the learning process. If the number of minority instances is too large, they are likely to be flooded into each base classifier. These generated pseudo-instances may not affect the results of individual classifiers. However, if the majority vote is directly used for ensemble in the final integration stage, the final model may be biased toward those base classifiers filled with many pseudo-minority instances, reducing the performance of the overall model.

SVM is a machine learning method that can effectively handle small-scale datasets. However, SVM needs to learn a better hyperplane with higher discrimination in the dataset. If there is a large amount of redundancy in the dataset or if instances in boundary areas are confused, the performance of the classifier may significantly decrease. LMD-WNG can effectively select instances that are beneficial to classifier learning through the matrix decomposition technique. However, LMD-WNG does not consider the number of instances selected. When facing a highly imbalanced dataset, the selected instances may be consistent with the number of minority instances, resulting in insufficient data subsets to obtain a hyperplane with high discrimination.

6.5 Discussion and limitations

The method proposed in this study searches the global structural information by using the k-nearest-neighbor graph and selects instances that are beneficial to classifier learning. However, we noticed that hyperparameters in the k-nearest-neighbor graph are the main cause of fluctuations in classifier performance. When a larger k is selected, the adjacency relationship between instances becomes more complex, and it is easy to include instances with weak adjacency relationships between instances in the selection process. However, when a smaller k is selected, the classifier may be limited to the surrounding instances, making it difficult to mine the global adjacency relationship between instances. To sum up, the hyperparameter k is an important factor that hinders the instance selection process. Of course, we can also provide optimal hyperparameters for different datasets through other optimization methods or hyperparameter selection strategies.

When faced with large-scale datasets, the complexity of the matrix decomposition technique used in LMD-WNG will increase significantly. This is also another limitation of LMD-WNG. However, the high-complexity problem of matrix decomposition can be alleviated by the following strategies: (1) Acceleration methods for matrix operations alleviate high-complexity problems by improving the matrix calculation process or using hardware devices such as GPUs. (2) Data chunking is another strategy to alleviate high-complexity problems. We can fuse instances selected from the chunked data using ensemble learning or other fusion strategies by randomly partitioning large-scale datasets into different regions. (3) Mini-batch training is currently an effective method for training machine learning models. Small batch training can reduce complexity issues when we do not use the entire dataset for training. Therefore, we do not need to be pessimistic. The high-complexity problem of LMD-WNG can be alleviated through various strategies in future research.

For the class-imbalance problem, in addition to the imbalance of class distribution and class-overlap, the presence of noise is also another important issue that affects the performance of the classifier. In LMD-WNG, the k-nearest-neighbor graph we use is a strongly constrained graph structure search method. We did not consider the degree of correlation or distance between instances. The noise may be far away from the real instances of other classes. However, under the strong constraint of the k-nearest-neighbor graph, at least k instances will be connected to it. Therefore, LMD-WNG may be affected by noise to a certain extent. In future research, we can alleviate the impact of noise by improving the structural information search strategy of LMD-WNG and using graph structure search methods with weaker constraints.

In addition, our previous work [2] also proved that more data information may exist in the subspace. Feature selection or subspace are effective methods to reduce the dimensionality of data [68]. It is obvious that when the number of features changes, the global correlation or data information of most datasets will also change. Therefore, how to select instances from the subspace that are beneficial to classifier learning is another challenge. Likewise, how to combine selected instances in the subspace also requires further research.

7 Conclusions

The LMD-WNG is a novel method used to solve the class-imbalance problem. According to the instance information in the dataset, it is necessary to select instances with a large amount of information to join the training dataset. When the number of minority instances in the dataset is large, we can use traditional resampling techniques to enhance the minority class or remove the majority class. However, when there are few minority instances in the dataset, we need to screen the majority instances in the training set to improve the overall performance of the model. In our proposed instance selection algorithm for LMD-WNG, we for the first time use graph structure methods and matrix decomposition methods for instance selection for the class-imbalance problem. Experiments on 50 baseline datasets show that the LMD-WNG instance selection algorithm outperforms or partially outperforms other comparative algorithms. For highly imbalanced datasets, the performance of LMD-WNG is more stable and will not be affected by the class-imbalance problem. However, the performance of the LMD-WNG instance selection algorithm will become more sensitive to the hyperparameter k as the dataset imbalance ratio increases. Therefore, more efficient parameter selection methods are needed to determine hyperparameters.

LMD-WNG is the first algorithm to transform data into a graph structure and select instances. Therefore, in future work, we can combine it with other methods and be able to fully explore the selection of instances according to the data structure. We also need to further explore the performance and selection strategy of the LMD-WNG instance selection algorithm on multi-class-imbalanced datasets. In addition, we can also try to explore the application of LMD-WNG ideas to graph-structured data or graph neural networks.

Data availability Data supporting the findings of this study are available from the corresponding author Qi Dai on request.

Declarations

Conflict of interest We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled “Imbalanced Instance Selection Based on Laplacian Matrix Decomposition with Weighted k-Nearest Neighbor Graph”.

References

1. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
2. Dai Q, Liu JW, Liu Y (2022) Multi-granularity relabeled under-sampling algorithm for imbalanced data. Appl Soft Comput 124:109083
3. Mayabadi S, Saadatfar H (2022) Two density-based sampling approaches for imbalanced and overlapping data. Knowl Based Syst 241:108217
4. Xiong R, Pelger M (2023) Large dimensional latent factor modeling with missing observations and applications to causal inference. J Econom 233(1):271–301
5. Lin WC, Tsai CF, Zhong JR (2022) Deep learning for missing value imputation of continuous data and the effect of data discretization. Knowl Based Syst 239:108079
6. Khoshgoftaar TM, Van Hulse J, Napolitano A (2010) Comparing boosting and bagging techniques with noisy and imbalanced data. IEEE Trans Syst Man Cybern Part A Syst Hum 41(3):552–568
7. Maulidevi NU, Surendro K (2022) SMOTE-LOF for noise identification in imbalanced data classification. J King Saud Univ Comput Inf Sci 34(6):3413–3423
8. Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Prog Artif Intell 5(4):221–232
9. Koziarski M, Woźniak M, Krawczyk B (2020) Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise. Knowl Based Syst 204:106223
10. Zhu J, Wang Z, Chen J, Chen YPP, Jiang YG (2022) Balanced contrastive learning for long-tailed visual recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6908–6917
11. Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239
12. Dhal P, Azad C (2024) A fine-tuning deep learning with multi-objective-based feature selection approach for the classification of text. Neural Comput Appl 36(7):3525–3553
13. Dhal P, Azad C (2023) A lightweight filter based feature selection approach for multi-label text classification. J Ambient Intell Humaniz Comput 14(9):12345–12357
14. Woźniak M, Wieczorek M, Siłka J (2023) BiLSTM deep neural network model for imbalanced medical data of IoT systems. Future Gener Comput Syst 141:489–499
15. Malhotra R, Kamal S (2019) An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. Neurocomputing 343:120–140
16. Yuan Z, Chen H, Li T, Sang B, Wang S (2021) Outlier detection based on fuzzy rough granules in mixed attribute data. IEEE Trans Cybern 52(8):8399–8412

17. Ibrahim MH (2021) ODBOT: outlier detection-based oversampling technique for imbalanced datasets learning. Neural Comput Appl 33(22):15781–15806
18. Ding H, Chen L, Dong L, Fu Z, Cui X (2022) Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener Comput Syst 131:240–254
19. Al S, Dener M (2021) STL-HDL: a new hybrid network intrusion detection system for imbalanced dataset on big data environment. Comput Secur 110:102435
20. Sun Y, Wong AK, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 23(04):687–719
21. Pirizadeh M, Alemohammad N, Manthouri M, Pirizadeh M (2021) A new machine learning ensemble model for class imbalance problem of screening enhanced oil recovery methods. J Pet Sci Eng 198:108214
22. Dai Q, Liu JW, Yang JP (2022) Class-imbalanced positive instances augmentation via three-line hybrid. Knowl Based Syst 257:109902
23. Fajardo VA, Findlay D, Jaiswal C, Yin X, Houmanfar R, Xie H, Liang J, She X, Emerson DB (2021) On oversampling imbalanced data with deep conditional generative models. Expert Syst Appl 169:114463
24. Wang G, Wong KW (2022) An accuracy-maximization learning framework for supervised and semi-supervised imbalanced data. Knowl Based Syst 255:109678
25. Liu J (2021) Fuzzy support vector machine for imbalanced data with borderline noise. Fuzzy Sets Syst 413:64–73
26. Zhang Y, Wang G, Huang X, Ding W (2023) TSK fuzzy system fusion at sensitivity-ensemble-level for imbalanced data classification. Inf Fusion 92:350–362
27. Liu W, Fan H, Xia M, Xia M (2022) A focal-aware cost-sensitive boosted tree for imbalanced credit scoring. Expert Syst Appl 208:118158
28. Tong H, Lu W, Xing W, Liu B, Wang S (2022) SHSE: a subspace hybrid sampling ensemble method for software defect number prediction. Inf Softw Technol 142:106747
29. Dai Q, Liu JW, Yang JP (2023) SWSEL: sliding window-based selective ensemble learning for class-imbalance problems. Eng Appl Artif Intell 121:105959
30. Ren J, Wang Y, Cheung YM, Gao XZ, Guo X (2023) Grouping-based oversampling in kernel space for imbalanced data classification. Pattern Recognit 133:108992
31. Douzas G, Bacao F, Last F (2018) Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. Inf Sci 465:1–20
32. Merris R (1994) Laplacian matrices of graphs: a survey. Linear Algebra Appl 197:143–176
33. Zhao X, Jia M, Lin M (2020) Deep Laplacian auto-encoder and its application into imbalanced fault diagnosis of rotating machinery. Measurement 152:107320
34. Zhou J, Jiang Z, Wang S (2020) Laplacian least learning machine with dynamic updating for imbalanced classification. Appl Soft Comput 88:106028
35. Ren L, Seklouli AS, Zhang H, Wang T, Bouras A (2023) An adaptive Laplacian weight random forest imputation for imbalance and mixed-type data. Inf Syst 111:102122
36. Ye X, Li H, Imakura A, Sakurai T (2020) An oversampling framework for imbalanced classification based on Laplacian eigenmaps. Neurocomputing 399:107–116
37. Santos MS, Abreu PH, Japkowicz N, Fernández A, Soares C, Wilk S, Santos J (2022) On the joint-effect of class imbalance and overlap: a critical review. Artif Intell Rev 55:1–69
38. Kovács G (2019) An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Appl Soft Comput 83:105662
39. Xia S, Zheng Y, Wang G, He P, Li H, Chen Z (2021) Random space division sampling for label-noisy classification or imbalanced classification. IEEE Trans Cybern 52(10):10444–10457
40. Zhang A, Yu H, Huan Z, Yang X, Zheng S, Gao S (2022) SMOTE-RkNN: a hybrid re-sampling method based on SMOTE and reverse k-nearest neighbors. Inf Sci 595:70–88
41. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
42. Dai Q, Liu JW, Zhao JL (2023) Distance-based arranging oversampling technique for imbalanced data. Neural Comput Appl 35(2):1323–1342
43. Yuan X, Chen S, Zhou H, Sun C, Yuwen L (2023) CHSMOTE: convex hull-based synthetic minority oversampling technique for alleviating the class imbalance problem. Inf Sci 623:324–341
44. Li T, Wang Y, Liu L, Chen L, Chen CP (2023) Subspace-based minority oversampling for imbalance classification. Inf Sci 621:371–388
45. Hoyos-Osorio J, Alvarez-Meza A, Daza-Santacoloma G, Orozco-Gutierrez A, Castellanos-Dominguez G (2021) Relevant information undersampling to support imbalanced data classification. Neurocomputing 436:136–146
46. Yan Y, Zhu Y, Liu R, Zhang Y, Zhang Y, Zhang L (2022) Spatial distribution-based imbalanced undersampling. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3161537
47. Farshidvard A, Hooshmand F, MirHassani SA (2023) A novel two-phase clustering-based under-sampling method for imbalanced classification problems. Expert Syst Appl 213:119003
48. Islam A, Belhaouari SB, Rehman AU, Bensmail H (2022) KNNOR: an oversampling technique for imbalanced datasets. Appl Soft Comput 115:108288
49. Dai Q, Liu JW, Shi YH (2023) Class-overlap undersampling based on Schur decomposition for class-imbalance problems. Expert Syst Appl 221:119735
50. Shelke MS, Deshmukh PR, Shandilya VK (2017) A review on imbalanced data handling using undersampling and oversampling technique. Int J Recent Trends Eng Res 3(4):444–449
51. Golub GH, Van Loan CF (2013) Matrix computations. JHU Press, Baltimore
52. Franti P, Virmajoki O, Hautamaki V (2006) Fast agglomerative clustering using a k-nearest neighbor graph. IEEE Trans Pattern Anal Mach Intell 28(11):1875–1881
53. Qin Y, Yu ZL, Wang CD, Gu Z, Li Y (2018) A novel clustering method based on hybrid k-nearest-neighbor graph. Pattern Recognit 74:1–14
54. Su Q, Niu Y, Liu X, Zhu Y (2012) Embedding color watermarks in color images based on Schur decomposition. Opt Commun 285(7):1792–1802
55. Barua S, Islam MM, Yao X, Murase K (2012) MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26(2):405–425
56. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232
57. Quinlan JR (1986) Induction of decision trees. Mach Learn 1:81–106
58. Breiman L (2001) Random forests. Mach Learn 45:5–32
59. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
60. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

61. Derrac J, Garcia S, Sanchez L, Herrera F (2015) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multi Valued Log Soft Comput 17:255–287
62. Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310
63. Tomek I (1976) Two modifications of CNN. IEEE Trans Syst Man Cybern Part A Syst Hum 6:769–772
64. Vuttipittayamongkol P, Elyan E (2020) Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. Inf Sci 509:47–70
65. Elyan E, Moreno-Garcia CF, Jayne C (2021) CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification. Neural Comput Appl 33:2839–2851
66. Liu Z, Wei P, Jiang J, Cao W, Bian J, Chang Y (2020) MESA: boost ensemble imbalanced learning with meta-sampler. Adv Neural Inf Process Syst 33:14463–14474
67. Garcı S, Triguero I, Carmona CJ, Herrera F (2012) Evolutionary-based selection of generalized instances for imbalanced classification. Knowl Based Syst 25(1):3–12
68. Dhal P, Azad C (2023) Hybrid momentum accelerated bat algorithm with GWO based optimization approach for spam classification. Multimed Tools Appl 83:1–41