STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **14**, November 2025, pp 2704–2723.
Published online in International Academic Press (www.IAPress.org)

Local Density-Aware Oversampling with Noise Resistance for Imbalanced Data Classification

Yuqing Zhang, Shuhao Fan, Wei Xue,*

School of Computer Science and Technology, Anhui University of Technology, Maanshan 243032, China

Abstract Imbalanced data classification has become a critical challenge in the field of machine learning. Traditional oversampling approaches often suffer from synthetic sample inaccuracy or aggravated class overlap problems. To better address these challenges, this paper proposes an oversampling method that integrates noise detection with adaptive sample generation. Specifically, we first introduce a noise detection strategy based on the average k-nearest neighbor distance, which identifies and removes high-interference noisy samples through local density analysis. Next, we design a weight allocation mechanism that jointly evaluates each instance's boundary risk and generation potential, prioritizing the synthesis of higher-weighted samples. Finally, to better preserve the classification boundary, we incorporate a neighbor class-sensitive coefficient into the sample generation process. Extensive experiments on 17 benchmark datasets demonstrate that the proposed method significantly outperforms well-known oversampling-based approaches, achieving superior classification performance.

Keywords Imbalanced data classification, Oversampling, Sample local density, Sample generation.

AMS 2010 subject classifications 62G09, 62H30

DOI: 10.19139/soic-2310-5070-2951

1. Introduction

Imbalanced data usually refers to a significant disparity in the number of samples across different classes [1]. In classification tasks, models tend to be biased towards the majority class, leading to misclassification [2]. This limitation will have a significant impact in practical applications, especially in fields such as medical diagnosis [3], credit card fraud detection [4], and fault detection [5]. Many methods have been proposed to solve the problem of imbalanced classification, which can be categorized into data-level approaches [6], and algorithm-level approaches [7]. Among them, the data-level approaches is more intuitive and easy to implement, which mainly include two strategies: oversampling and undersampling. The oversampling method aims to balance the dataset by increasing the number of minority class samples, thereby improving the model's ability to recognize the minority class. Oversampling methods have significant advantages in dealing with imbalanced data [8], so this paper will focus on exploring the oversampling method in imbalanced classification problems.

The most classic oversampling technique is SMOTE (Synthetic Minority Oversampling Technique) [9]. This method addresses the class imbalance problem by generating new synthetic samples. While SMOTE demonstrates an advantage over ROS (Random Oversampling) in mitigating the issue of model overfitting, it also has some drawbacks. First, SMOTE assumes that the sample density distribution is uniform [10], but in reality, the data distribution is often more complex. This results in SMOTE generating an excessive number of samples in data-dense areas while generating too few samples in regions where the data is sparse but crucial for distinguishing between different classes [11]. Second, SMOTE is particularly sensitive to noise [12]. Once there are noisy samples,

*Correspondence to: Wei Xue (Email: xuewei@ahut.edu.cn).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

it may use them to synthesize new samples, creating even more noisy samples. And if the noisy samples are located among the majority class samples, it may lead to class overlap [13]. These overlapping samples make it difficult for the classifier to make accurate judgments, thereby degrading the classifier's performance [2].

To address these issues of SMOTE, many improvements to SMOTE have been made. Density-based methods like the Adaptive Synthetic Sampling Approach (ADASYN) [14] and Density-Based SMOTE (DB-SMOTE) [15] decide where to generate new samples based on the local density distribution of the data. There are also clustering-based methods, such as KMeans-SMOTE [11], which first partition the minority class data into different regions before oversampling. Others such as Borderline-SMOTE [10] focuses on generating samples in the boundary region. Distance-SMOTE [16] determines synthesis locations and sample numbers based on distances to nearest neighbors of minority class samples. Although these methods have played a certain role in improving the uniformity of sample distribution in SMOTE, they are still easily disturbed by noise and lack precision in depicting the local density structure of complex data. None of these methods have achieved comprehensive optimization in terms of noise detection, boundary protection, and generated sample quality, limiting their application in complex imbalanced scenarios [17]. To this end, we propose a novel oversampling method, which can effectively mitigate the impact of noisy samples, enhance the quality of synthetic samples, clarify classification boundaries, and, ultimately, improve model performance. The main contributions of this paper are summarized as follows.

1) We employ nearest neighbor average distance and local density analysis, incorporating a threshold to effectively distinguish noisy samples.
2) We take into account both the boundary information and generation capability of samples, enabling the prioritization of high-confidence samples for generating new instances.
3) We introduce a dynamic coefficient during the generation phase to progressively clarify the classification boundary.

The rest of this paper is organized as follows. We briefly review some related work in Section 2. In Section 3, we present the proposed method in detail. The experimental results and analysis are provided in Section 4. Finally, we conclude this paper in Section 5.

2. Related work

Due to the factors such as lack of minority class samples, noise interference, and unclear classification boundaries, imbalanced data classification faces significant challenges [5]. Oversampling is an important strategy for solving imbalanced data classification. In [9], Chawla et al. proposed SMOTE, which generates synthetic data by performing linear interpolation between minority class samples. However, a disadvantage of SMOTE is that it ignores the potential negative effects of noisy samples, and thus may result in overlap between classes [12].

To reduce the impact of noise on the classification results, researchers have proposed filtering noise before resampling. Classic methods such as Tomek Links [18] and ENN (Edited Nearest Neighbor) [19] optimize data distribution by removing overlapping samples on the boundary. However, in high-dimensional data, these methods are susceptible to the "curse of dimensionality" and may excessively delete samples that are crucial for the classification boundary [20]. PDR-SMOTE [21] partitions samples into different regions and removes those identified as noise. However, due to the often complex distribution of boundary samples, this approach may fail to fully capture their characteristics. HSNF [22] performs denoising in two stages based on distinct noise identification mechanisms and classifies samples into safe and boundary regions. Nevertheless, it may mistakenly identify some important boundary samples as noise, leading to the loss of critical information.

During the generation phase, some efforts have integrated density-aware approaches into oversampling techniques. Cluster-based methods such as Cluster-SMOTE [23] and DB-SMOTE [15] partition minority classes into sub-clusters and generate samples within them. These methods improve distribution but ignore boundary regions, as clustering tends to focus on intra-group cohesion rather than inter-class separation [24]. SPMSC [25] employs the mean shift algorithm to cluster minority class samples in order to reduce the generation of duplicate samples. However, the mean shift algorithm has a weak ability to recognize clusters with complex shapes, which

may affect the clustering effect of the samples. AWNNAC [26] employs relative density to perform adaptive weighting on the denoised minority class samples. However, the nearest neighbor area control strategy it uses may cause the generated samples to be overly concentrated at the edges of the minority class region.

3. Method

Our method consists of three key components: noise identification and removal, adaptive weighting, and sample synthesis. Before introducing our method, we formalize the distance measurement used in our method. Given two samples $x = x_i{}_{i=1}^{n}$ and $y = \{y_i\}_{i=1}^{n}$, we use Euclidean norm to measure the distance (DIS) between x and y, which is defined as Eq. (1).

$$DIS(x,y) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{1}$$

3.1. Noise identification and removal

Considering that majority class noisy samples may potentially affect the identification of minority class noisy samples, first we need to identify and remove them. Specifically, if the nearest neighbor of a majority class sample belongs to the minority class, then that majority class sample is considered to be noise and is removed. Next, we identify noise in minority class samples. We consider samples that are isolated from other minority class samples as likely noise. So, we use the average distance between the minority sample x_i and its k-nearest neighbors as an indicator to measure its degree of isolation from other samples of the same class, which is defined by Eq. (2).

$$D(x_i) = \frac{1}{k}\sum_{x_j \in \mathcal{N}_k(x_i)} DIS(x_i, x_j) \tag{2}$$

where $\mathcal{N}_k(x_i)$ represents the k-nearest neighbors of x_i within the set of minority class samples. The larger the average distance, the more isolated the sample is, and thus the more likely it is to be a noisy sample. So we sort the sample points in ascending order based on the average distance between each minority class sample and its k-nearest neighbors. This sorting approach helps us observe the trend of changes in the average distance, thereby enabling more accurate detection of abrupt change points. Figure 1 shows the average distance distributions for two datasets. In the graph, the x-axis represents the i-th sample point, and the y-axis represents the average distance between that sample and its k nearest neighbors. The sample points in the graph are arranged in ascending order of the average distance.

Noise threshold is a factor that needs to be considered, as different datasets exhibit varying distance distributions. Therefore, we need to find a way to determine the threshold where the average distance between minority class samples and their k-nearest neighbors changes sharply across different datasets. In the curve of average distances between a sample and its k-nearest neighbors, when abrupt changes occur in the average distances, the curve's slope at corresponding points undergoes sharp variations, indicating such samples are highly likely to be noise. To precisely identify these transition points, slope calculations must be performed at each curve point. By analyzing slope variations, we can locate positions where distance jumps occur. Based on this requirement, we introduce Eq. (3) from [28].

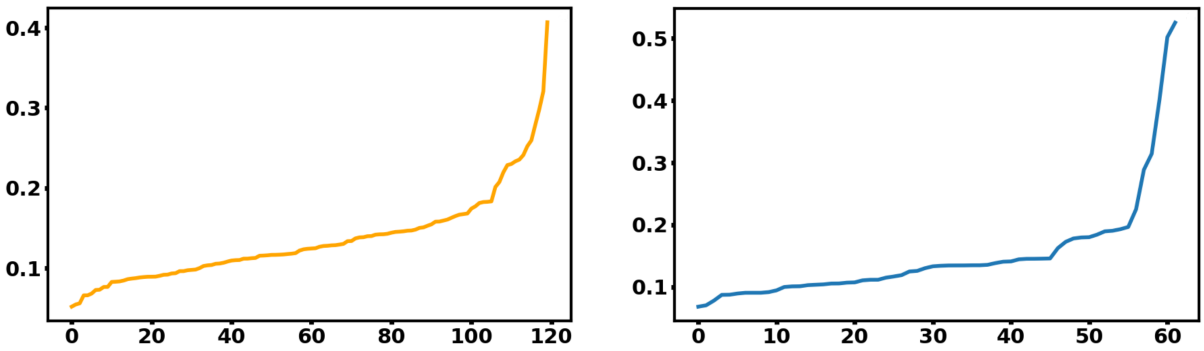
$$D_{\text{norm}} = \sin\left(\tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)\right) \tag{3}$$

Here, Δx and Δy represent the displacement increments of sample points along the x-axis and y-axis, respectively. Through this formula, the normalized distance metric D_{norm} is obtained, which maps the slope of the local displacement vector to the $[0,1]$ interval. This enables slope values to be comparable across different

datasets, facilitating more precise localization of abrupt changes in average distance and thereby determining the noise threshold.

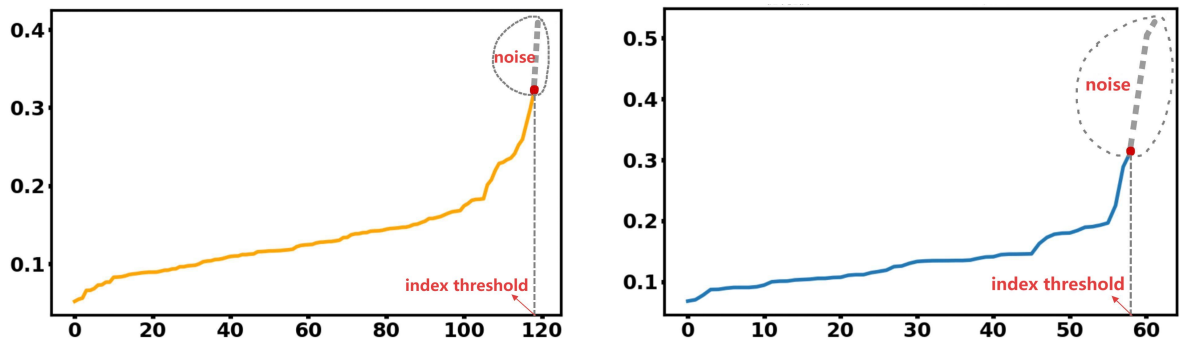
By repeating this process for all sample points, a candidate threshold set is obtained, from which the maximum value is selected as the final threshold. Then, the sample index corresponding to the maximal threshold value is defined as the index threshold. Precisely, for the Circles dataset and Ecoli1, the index threshold are 117 and 59 respectively, as indicated by the red marker in Figure 2, all subsequent sample points should be regarded as noise and removed.

Specifically, when the average distance from a given sample to its k-nearest neighboring samples is equivalent to the average distance from any single or multiple other samples to their corresponding k-nearest neighboring samples, these two or more samples will be regarded as noise threshold.



(a) Circles (b) Ecoli1

Figure 1. Illustration of average distance distributions.



(a) Circles (b) Ecoli1

Figure 2. Application of noise threshold strategy on two datasets.

3.2. Adaptive weighting

To fully use the information of minority class samples for generating new samples of higher quality, we design a weighting strategy for minority class samples. For each minority sample x_i, its weight w_i is determined by two key factors, namely association degree and classification difficulty. For the term association degree, given the complex associations among samples in real-world data, this term is introduced to quantify the closeness of associations between samples. It serves as a core indicator for measuring associations. For classification difficulty, as classification tasks are confronted with challenges such as data complexity, noise interference, and class imbalance, it is necessary to quantify the degree of classification difficulty in order to evaluate the performance

of classification algorithms. Classification difficulty serves as an important reference for assessing algorithm effectiveness.

Association degree. Generally speaking, the higher the degree of association between a minority sample and its neighboring samples, the more useful information they carry. Therefore, we introduce the concept of the two-way neighbor count $N_{connect}$ for samples. $\mathcal{N}_k(x_i)$ is the k-nearest neighbor set of a minority class sample x_i, for each k-nearest neighbor in $\mathcal{N}_k(x_i)$, if x_i also appears in the k-nearest neighbor sets of these k-nearest neighbors, then we refer to the k-nearest neighbor as the two-way neighbor of xi. As illustrated in Figure 3 (a), N_5 is a two-way neighbor of x_i. Then let $N_{connect}$ equal to the number of two-way neighbor, where $N_{connect}$ is less than or equal to k.

Classification difficulty. If the k-nearest neighbors of a minority class sample x_i contain more majority class samples, indicating that it is located in a class boundary or overlapping region, then it is possible to be misclassified by classifiers. Given the greater influence of such samples on the classification boundary, they should be assigned higher weights to emphasize their importance. We use N_{maj} represent the number of majority class samples in its k-nearest neighbors. As described in Figure 3 (b), for x_i, it has two majority k-nearest neighbors, so its N_{maj} is equal to 2.

In summary, the w_i is calculated by Eq. (4)

$$w_i = \frac{N_{\text{connect}}}{k} \cdot (N_{maj} + 1) \tag{4}$$

where the parameter 1 is to prevent the result from being 0 when the number of majority class samples in the k-nearest neighbors is 0. Meanwhile, the division by k is to standardize the numerical value. Minority class samples in proximity to other minority class samples exhibit higher N_{connect} values, indicating that they carry more discriminative information and are thus assigned greater weights. The $\left(N_{\text{maj}} + 1\right)$ term adjusts weights based on local majority class density, preventing over-sampling in densely populated regions while prioritizing boundary sample selection. In the process of weighting, this formula takes into full consideration the relationship between the sample and both the majority class samples and other minority class samples, thereby enabling a more accurate weighting process for the minority class.

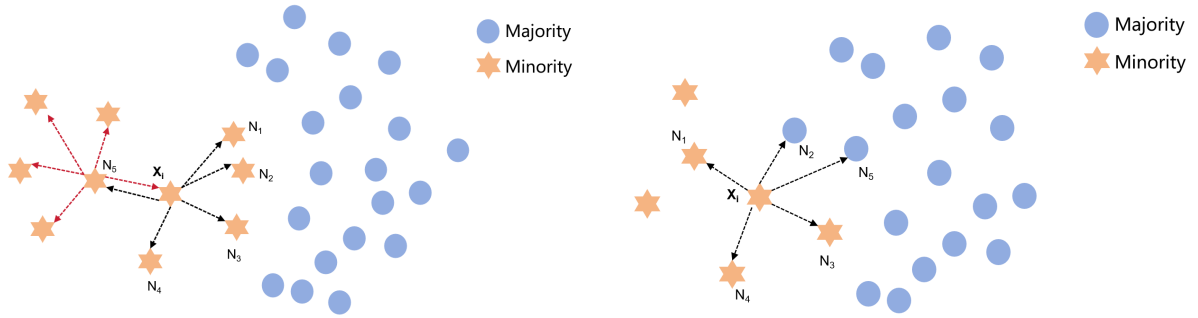


Figure 3. (a) Illustration of two-way neighbor of x_i; (b) Example of majority k-nearest neighbors of boundary sample x_i.

3.3. Sample synthesis

Based on the weights w_i of the minority class samples obtained in the previous section, we first determine the number s_i of synthetic samples that need to be generated for each minority class sample x_i. The total number of synthetic samples required S_{total}, is set as the difference between the number of majority class samples and the number of minority class samples, that is, $S_{\text{total}} = n_{\text{maj}} - n_{\text{min}}$. Subsequently, S_{total} is allocated to each minority class sample according to their weight proportions. Thus, s_i is defined as Eq. (5).

$$s_i = round(\frac{w_i}{\sum_{j=1}^{n_{\min}} w_j} \cdot S_{\text{total}}) \tag{5}$$

where $round(\cdot)$ denotes the rounding operation to minimize cumulative errors due to integer rounding. When generating synthetic samples, this work improves upon the classic SMOTE algorithm by introducing a neighbor class-sensitive interpolation coefficient α. The specific steps are as follows.

Neighbor selection. For the minority class sample x_i, randomly select a neighbor x_j from its k-nearest neighbor set $\mathcal{N}_k(x_i)$.

Adjustment of interpolation coefficient. If x_j belongs to the majority class, which is located in the class boundary region, set α=0.5 to limit the interpolation magnitude towards the majority class direction, avoiding the generation of potentially overlapping samples. If x_j belongs to the minority class, which is located in the dense region within the class, set α=1, making full use of the intra-class information to generate new samples. This design is inspired by Borderline-SMOTE [10], but it further enhances security.

Sample Generation. Generate the synthetic sample x_{new} using Eq. (6).

$$x_{\text{new}} = x_i + \alpha \cdot (x_j - x_i) \tag{6}$$

.

Table 1. Datasets description.

Dataset Name	Attributes	Samples	Imbalance ratio
Glass0	9	214	2.06
Glass6	9	214	6.38
Haberman	3	306	2.78
Segment0	19	2308	6.02
Vehicle2	18	846	2.88
Bupa	6	344	1.39
Circles	2	649	3.33
Heart	13	269	1.26
Paw02a-600-5-70-BI	2	600	5.00
Pima	8	768	1.87
Wine	13	178	1.50
Glass2	9	214	11.59
Wisconsin	9	683	1.86
Vehicle3	18	846	2.99
Page-blocks-1-3_ vs _ 4	10	472	15.86
Vehicle1	18	846	2.90
Moon	2	1099	9.99

4. Experiment

To evaluate the proposed method, we conduct experiments on imbalanced datasets and compare it with well-known oversampling-based methods. The implementation details and evaluation metrics are described below.

4.1. Experimental Setup

Datasets. The 17 datasets utilized in our experiments are sourced from the KEEL and UCI machine learning repositories, which are widely recognized in the field of imbalanced learning. As can be seen in TABLE 1.

These datasets vary in size, imbalance ratios, and cover diverse domains, thereby ensuring the diversity of our experimental setup. For datasets containing multiple classes, we make the class with the smallest number of samples as the minority class, while merging the remaining classes into a majority class, thus transforming the problem into a binary classification task.

Table 2. Some notations.

Notation	Meaning / Definition
TP	Number of True Positives
FP	Number of False Positives
TN	Number of True Negatives
FN	Number of False Negatives
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Recall	$\frac{\text{TP}}{\text{TP+FN}}$
Specificity	$\frac{\text{TN}}{\text{TN+FP}}$
TPR	$\frac{\text{TP}}{\text{Number of Minority}}$
FPR	$\frac{\text{TP}}{\text{Number of Majority}}$

Table 3. Evaluation metrics.

Metric	Definition
F-measure	$\frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$
G-mean	$\sqrt{Recall \cdot Specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

Baselines. We compares eight oversampling methods: ROS (Random Oversampling), SMOTE [9], B1-SMOTE [10], B2-SMOTE [10], D-SMOTE [16], SVM-SMOTE [27], ASN-SMOTE [29] and HSNF [22]. Four classifiers, namely K-nearest neighbors (KNN), random forest (RF), support vector machine (SVM) and naive bayes (NB) are used to evaluate the performance of these methods. Besides, we implement 5-fold cross-validation to prevent overfitting. Each cross-validation repeats three times, with the mean value representing the final result.

Evaluation metrics. We employ F-measure, G-mean, and AUC as evaluation metrics. F-measure represents the harmonic mean of precision and recall, effectively balancing these two metrics. G-mean is defined as the geometric mean of sensitivity (true positive rate) and specificity (true negative rate). AUC evaluates method's ability to discriminate between classes across different threshold settings. The closer the results number is to 1, the better the method's discrimination ability. The notations and metrics are shown in TABLE 2 and TABLE 3.

4.2. Results and analysis

Overall performance analysis. In order to better observe the data distribution after oversampling using our method, we select four two-dimensional datasets: 04clover5z-600-5-0-BI-full, Moon, paw02a-600-5-70-BI and 03subcl5-800-7-30-BI for visual analysis. As shown in Figure 4, red points represent majority class samples, orange denote minority, and green indicate synthetic samples. It can be observed that the red sample points and the generated green sample points do not occupy the same regions. This demonstrates that our method maintains distinct boundaries between different classes, thus avoiding class overlap. Meanwhile, the generated

green samples are consistent with the original distribution of the minority class, enabling them to better represent the true characteristics of the minority class. In summary, our observations show synthetic samples are uniformly distributed, avoid class overlap, and align with the minority class's original distribution, demonstrating their reliability.

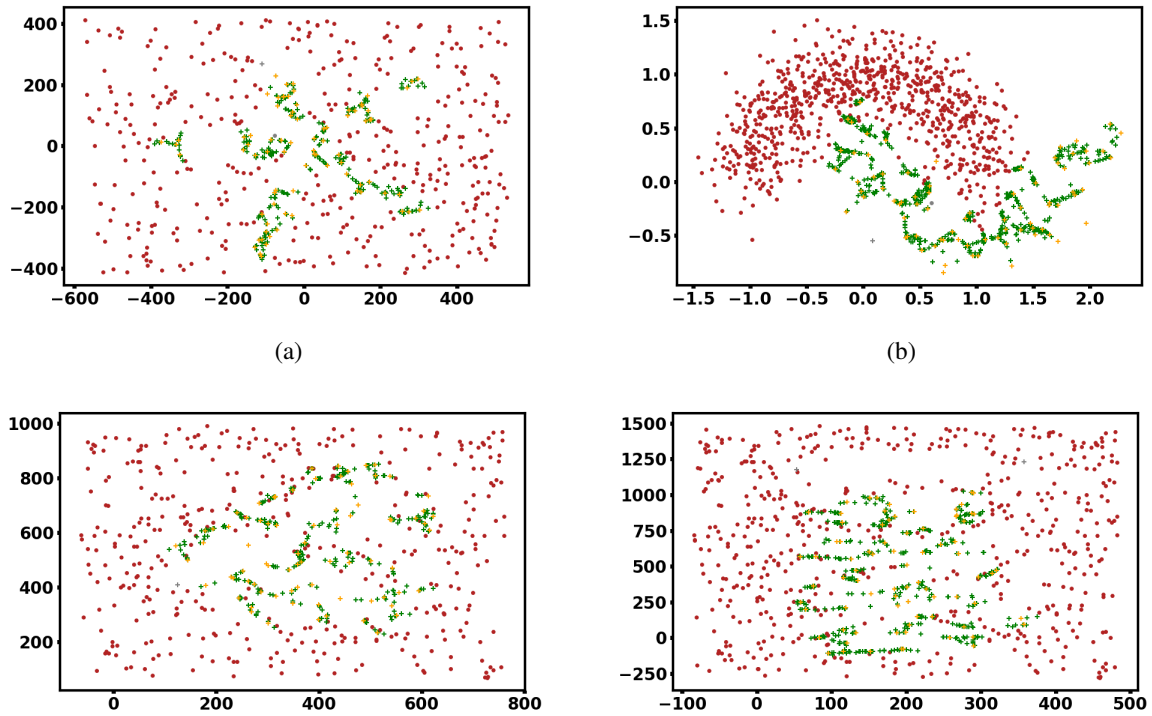


Figure 4. Data distributions after applying the proposed method. (a) 04clover5z-600-5-0-BI-full dataset; (b) Moon dataset; (c) paw02a-600-5-70-BI dataset; (d) 03subcl5-800-7-30-BI dataset.

We highlight the top-ranked method in red and the second-best in blue in Table 4 to Table 7. It can be observed that on the three metrics of F-measure, G-mean, and AUC, the proposed method achieves the optimal value in at least one-third of the metrics across all experimental scenarios on KNN, RF, SVM and NB classifiers. This result demonstrates that our method performs exceptionally well when dealing with class imbalance issues, which proves the effectiveness of our method. Specifically, in small-scale datasets with a relatively low imbalance ratio, sample features are relatively concentrated and class distributions are fairly balanced. Under such circumstances, our method demonstrates exceptional performance. For instance, on the Bupa dataset using a KNN classifier, our method achieved the top rankings in both F1-measure and G-mean values, with its AUC value being only approximately 0.0005 lower than the highest score. When confronted with large-scale datasets characterized by a high imbalance ratio, where the overall data volume is abundant but minority-class samples are scarce, our method still ensures outstanding performance. Taking the Page-Blocks-1-3 vs 4 dataset with an SVM classifier as an example, our method secured the highest scores across all three metrics. By comparing the experimental results across different dataset conditions, it can be observed that our method exhibits excellent adaptability and stability.

Table 4. The experimental results of 16 datasets obtained using the KNN classifier.

Dataset	Measure	ROS	SMOTE	B1-SMOTE	B2-SMOTE	D-SMOTE	SVM-SMOTE	ASN-SMOTE	HSNF	Ours
Glass0	F-measure	0.703972	0.692783	0.667022	0.689406	0.715614	0.703120	0.686585	0.635688	0.716522
	G-mean	0.774513	0.756412	0.727779	0.752361	0.784103	0.767012	0.755618	0.716941	0.778630
	AUC	0.858665	0.852797	0.871420	0.867655	0.869933	0.867312	0.845091	0.834527	0.882134
Glass6	F-measure	0.786434	0.831697	0.766434	0.742369	0.847081	0.750644	0.758152	0.764918	0.777922
	G-mean	0.890199	0.917136	0.856400	0.836267	0.919857	0.848135	0.876871	0.861534	0.846979
	AUC	0.910360	0.948468	0.908829	0.907838	0.930180	0.919459	0.924865	0.899099	0.870811
Haberman	F-measure	0.444940	0.460499	0.379789	0.367921	0.410489	0.418218	0.409808	0.433894	0.467107
	G-mean	0.605135	0.615657	0.545618	0.530388	0.571657	0.579464	0.570946	0.584967	0.618308
	AUC	0.657565	0.656250	0.635245	0.631642	0.621381	0.669567	0.639861	0.665155	0.656609
Segment0	F-measure	0.934230	0.945153	0.954457	0.941447	0.943751	0.933140	0.925976	0.930350	0.945172
	G-mean	0.984539	0.987847	0.985800	0.979735	0.986338	0.985553	0.980503	0.985047	0.987842
	AUC	0.994690	0.995798	0.994651	0.995782	0.994917	0.995272	0.996297	0.995976	0.997741
Vehicle2	F-measure	0.803256	0.779867	0.794278	0.763255	0.805897	0.795392	0.776790	0.772303	0.780012
	G-mean	0.890169	0.877010	0.892965	0.875400	0.892829	0.884905	0.873687	0.867017	0.890064
	AUC	0.953701	0.956238	0.952279	0.943599	0.957698	0.954543	0.953009	0.938842	0.958096
Bupa	F-measure	0.553270	0.570689	0.573963	0.523584	0.543989	0.568063	0.506472	0.438151	0.590990
	G-mean	0.610051	0.617750	0.619410	0.582150	0.603267	0.622414	0.586396	0.533441	0.638158
	AUC	0.663405	0.671663	0.649449	0.649720	0.665628	0.662389	0.665671	0.640135	0.666912
Circles	F-measure	0.796899	0.803293	0.825552	0.813206	0.799114	0.811024	0.87975	0.494676	0.882604
	G-mean	0.699434	0.710517	0.708507	0.705374	0.701327	0.690926	0.616204	0.660693	0.603310
	AUC	0.740314	0.758760	0.760864	0.748559	0.755103	0.760610	0.766590	0.759305	0.770047
Heart	F-measure	0.708525	0.706905	0.682544	0.686315	0.666527	0.679559	0.727976	0.560050	0.718036
	G-mean	0.665900	0.665495	0.649091	0.659938	0.634528	0.642064	0.661011	0.624534	0.649852
	AUC	0.695121	0.685755	0.686576	0.701975	0.696002	0.691987	0.702874	0.708671	0.707929
Paw02A-600-5-70-Bi	F-measure	0.536300	0.497389	0.523388	0.521320	0.544073	0.540624	0.558047	0.566499	0.569345
	G-mean	0.754099	0.708479	0.737838	0.736063	0.751599	0.752502	0.758586	0.753542	0.789335
	AUC	0.836850	0.850050	0.854500	0.847750	0.836950	0.843950	0.864900	0.857250	0.858250
Pima	F-measure	0.733369	0.739068	0.736398	0.724206	0.736599	0.734748	0.796233	0.575811	0.800799
	G-mean	0.670525	0.672707	0.680734	0.669015	0.697755	0.679318	0.689288	0.663911	0.684484
	AUC	0.737618	0.751765	0.738294	0.734644	0.750021	0.742790	0.748633	0.745434	0.750036
Wine	F-measure	0.671909	0.699885	0.712111	0.699413	0.729183	0.688484	0.680717	0.641425	0.730709
	G-mean	0.680291	0.698405	0.686183	0.693968	0.736117	0.692827	0.699982	0.682737	0.717172
	AUC	0.878937	0.891327	0.909985	0.901218	0.913868	0.889165	0.887486	0.823995	0.893558
Glass2	F-measure	nan	0.310942	0.269899	0.298182	0.300000	0.326185	0.321919	nan	0.298368
	G-mean	0.547585	0.671432	0.588681	0.631940	0.652259	0.673666	0.567834	0.215750	0.618904
	AUC	0.719979	0.696496	0.728162	0.710171	0.736282	0.719936	0.717500	0.522308	0.621261
Wisconsin	F-measure	0.958722	0.952482	0.959469	0.955384	0.956630	0.953649	0.949448	0.944082	0.959338
	G-mean	0.971478	0.967077	0.974491	0.971264	0.969469	0.971051	0.961000	0.954523	0.972378
	AUC	0.985974	0.985881	0.983854	0.982359	0.984593	0.982457	0.988288	0.984167	0.982883
Vehicle3	F-measure	0.518405	0.506302	0.501684	0.506286	0.501035	0.491550	0.527328	0.427320	0.507150
	G-mean	0.679602	0.669828	0.665542	0.670429	0.665619	0.654985	0.686623	0.583258	0.671887
	AUC	0.751646	0.753660	0.736209	0.730738	0.741995	0.746785	0.762056	0.705123	0.730941
Page-Blocks-1-3_vs_4	F-measure	0.627578	0.618024	0.601080	0.625929	0.636044	0.598156	0.624693	0.636190	0.596525
	G-mean	0.885546	0.895617	0.889198	0.888455	0.903123	0.886505	0.872042	0.877368	0.849339
	AUC	0.959217	0.966978	0.962371	0.964723	0.962484	0.963832	0.964930	0.931935	0.968598
Vehicle1	F-measure	0.525914	0.526651	0.557777	0.555883	0.549978	0.526625	0.549408	0.489685	0.542731
	G-mean	0.682936	0.682601	0.711070	0.709652	0.703637	0.683434	0.703259	0.644084	0.695642
	AUC	0.740843	0.750168	0.765051	0.750915	0.754127	0.746194	0.766459	0.728201	0.749286
Moon	F-measure	0.854661	0.835621	0.854603	0.808015	0.852207	0.831264	0.787971	0.799672	0.839421
	G-mean	0.951701	0.940375	0.938335	0.918226	0.946969	0.948764	0.953520	0.952822	0.954771
	AUC	0.969516	0.968092	0.969094	0.969411	0.968591	0.972088	0.980743	0.971958	0.981536

Table 5. The experimental results of 16 datasets obtained using the RF classifier.

Dataset	Measure	ROS	SMOTE	B1-SMOTE	B2-SMOTE	D-SMOTE	SVM-SMOTE	ASN-SMOTE	HSNF	Ours
Glass0	F-measure	0.898796	0.863061	0.855876	0.884658	0.883232	0.890854	0.875548	0.872727	0.874598
	G-mean	0.864812	0.823908	0.831784	0.852741	0.845772	0.857341	0.834534	0.810439	0.852645
	AUC	0.934782	0.925721	0.935785	0.936770	0.928413	0.940218	0.923505	0.895866	0.938723
Glass6	F-measure	0.978213	0.978351	0.975960	0.960168	0.972706	0.975373	0.981246	0.978729	0.984208
	G-mean	0.917447	0.912995	0.872339	0.857682	0.892729	0.910154	0.915794	0.880300	0.885455
	AUC	0.993694	0.974144	0.973514	0.984144	0.973694	0.988108	0.970991	0.974595	0.995315
Haberman	F-measure	0.784029	0.769171	0.785391	0.796295	0.797276	0.778773	0.793840	0.832409	0.778441
	G-mean	0.596652	0.570955	0.584318	0.572496	0.611308	0.596083	0.595760	0.539780	0.608129
	AUC	0.717925	0.710474	0.723611	0.724575	0.724330	0.726291	0.721217	0.710417	0.732451
Segment0	F-measure	0.997979	0.998232	0.998484	0.998734	0.997980	0.998736	0.996960	0.996454	0.998736
	G-mean	0.992874	0.993127	0.994657	0.996194	0.991582	0.996178	0.994432	0.992642	0.996200
	AUC	0.999908	0.999923	0.999939	0.999916	0.999908	0.999885	0.999885	0.999854	0.999916
Vehicle2	F-measure	0.991228	0.993631	0.989647	0.989641	0.992037	0.993593	0.987141	0.987187	0.992006
	G-mean	0.983661	0.987553	0.979058	0.980599	0.984358	0.990584	0.982726	0.981205	0.987428
	AUC	0.998725	0.998906	0.998246	0.998066	0.998688	0.999090	0.998104	0.997329	0.998760
Bupa	F-measure	0.760845	0.754437	0.764052	0.758009	0.771906	0.765034	0.788690	0.771156	0.772599
	G-mean	0.672116	0.676938	0.717794	0.694382	0.699615	0.717732	0.687015	0.550188	0.705075
	AUC	0.760800	0.766595	0.764132	0.762315	0.756435	0.778017	0.778190	0.724957	0.763461
Circles	F-measure	0.850481	0.833073	0.831861	0.833831	0.818846	0.835862	0.863596	0.843558	0.829632
	G-mean	0.683371	0.723306	0.709944	0.706550	0.684578	0.699638	0.698297	0.676599	0.728417
	AUC	0.777914	0.788828	0.772326	0.779419	0.773497	0.788235	0.793783	0.773205	0.788844
Heart	F-measure	0.850031	0.858740	0.870212	0.859097	0.847138	0.849602	0.854853	0.835324	0.851585
	G-mean	0.829727	0.834786	0.844440	0.834864	0.817231	0.822601	0.823383	0.738749	0.828425
	AUC	0.901685	0.903104	0.908080	0.901171	0.898913	0.900000	0.903949	0.888466	0.900845
Paw02A-600-5-70-Bi	F-measure	0.895340	0.880539	0.887529	0.888235	0.897721	0.886833	0.892669	0.905676	0.891583
	G-mean	0.745300	0.747377	0.758999	0.764908	0.771219	0.743199	0.777661	0.787482	0.795198
	AUC	0.878950	0.876100	0.878000	0.872850	0.882750	0.881650	0.891650	0.877300	0.883300
Pima	F-measure	0.815155	0.806209	0.798916	0.794294	0.806938	0.803998	0.815311	0.826130	0.808139
	G-mean	0.735328	0.735793	0.729246	0.736255	0.729188	0.737893	0.720514	0.686749	0.759286
	AUC	0.828491	0.831060	0.821106	0.819147	0.821467	0.829184	0.819481	0.826119	0.825459
Wine	F-measure	0.958683	0.940122	0.963561	0.948218	0.949455	0.946711	0.953340	0.952381	0.964089
	G-mean	0.945553	0.919514	0.950373	0.935834	0.932805	0.935626	0.940776	0.925162	0.949163
	AUC	0.995455	0.995455	0.995455	0.996753	0.995455	0.995455	0.994156	0.995455	0.999351
Glass2	F-measure	0.947460	0.917147	0.947472	0.934469	0.947940	0.945180	0.950228	nan	0.952652
	G-mean	0.328160	0.267943	0.223406	0.209948	0.112470	0.229488	0.215470	nan	0.369084
	AUC	0.751838	0.808205	0.762350	0.778333	0.724231	0.810897	0.785000	nan	0.710812
Wisconsin	F-measure	0.973025	0.975107	0.975058	0.976266	0.975070	0.973514	0.973112	0.968712	0.977205
	G-mean	0.961104	0.966341	0.966411	0.967548	0.966400	0.970096	0.960031	0.952403	0.971638
	AUC	0.993273	0.992736	0.993088	0.990962	0.992454	0.992197	0.993273	0.991983	0.990502
Vehicle3	F-measure	0.849394	0.846162	0.850507	0.852480	0.852510	0.852459	0.838735	0.863172	0.848727
	G-mean	0.696216	0.716425	0.740053	0.738928	0.695006	0.706524	0.699281	0.626864	0.729050
	AUC	0.854236	0.857183	0.855173	0.854854	0.857031	0.854563	0.860235	0.815318	0.853365
Page-Blocks-1-3_vs_4	F-measure	0.988707	0.990789	0.990970	0.988232	0.985231	0.983320	0.980159	0.984604	0.983763
	G-mean	0.892345	0.958289	0.919449	0.972192	0.856198	0.949328	0.900913	0.942152	0.892499
	AUC	0.996404	0.996404	0.999551	0.999101	0.996404	0.992809	0.985993	0.989213	0.988315
Vehicle1	F-measure	0.854718	0.851647	0.862746	0.858050	0.863943	0.856246	0.851599	0.854905	0.858029
	G-mean	0.748900	0.730778	0.759310	0.763612	0.747931	0.753180	0.750491	0.669933	0.778923
	AUC	0.865933	0.864656	0.866428	0.863440	0.863914	0.862269	0.859732	0.831207	0.869571
Moon	F-measure	0.983950	0.982316	0.984454	0.979917	0.982899	0.983393	0.970861	0.971823	0.983823
	G-mean	0.918507	0.945352	0.918693	0.899520	0.930923	0.932013	0.938596	0.944749	0.951517
	AUC	0.979245	0.987944	0.982768	0.979316	0.987944	0.982914	0.983368	0.979365	0.984994

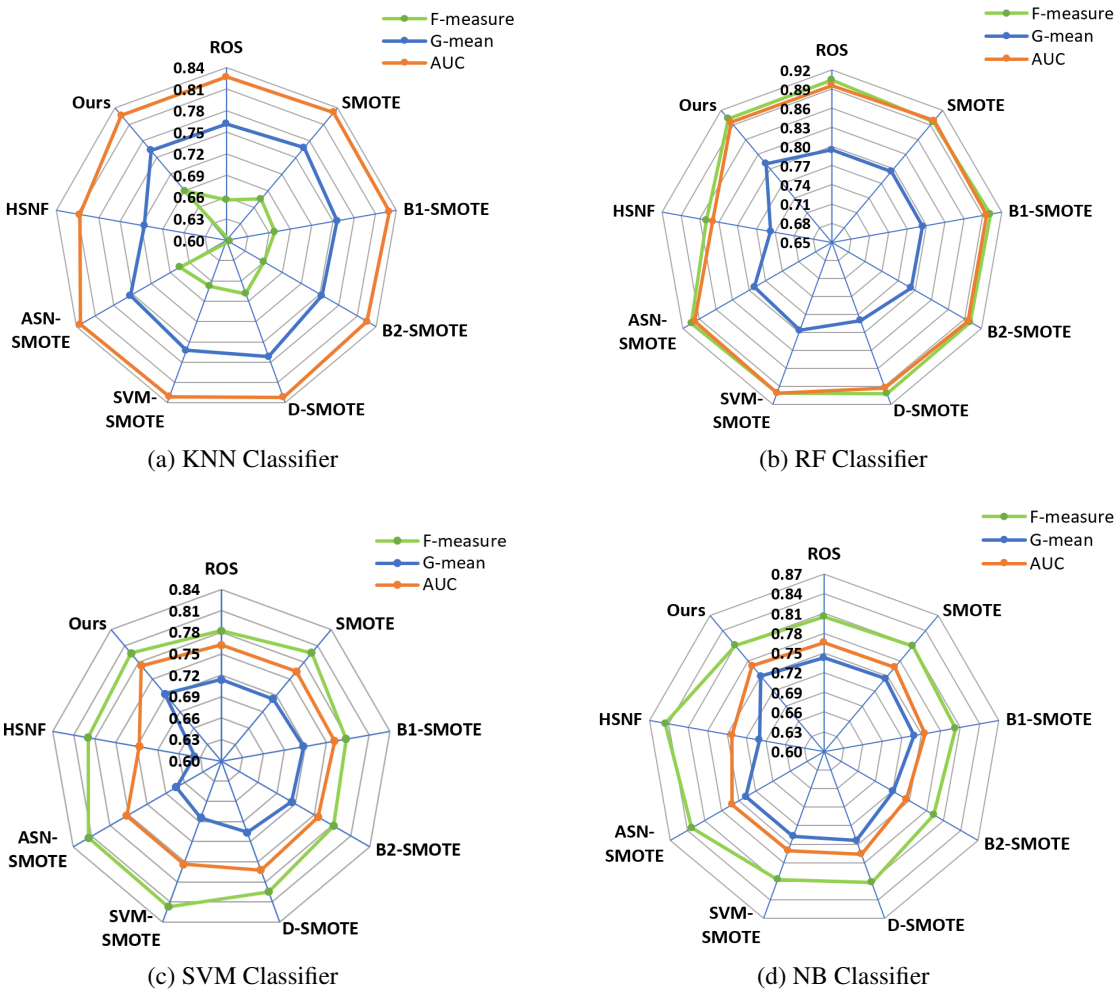
Table 6. The experimental results of 16 datasets obtained using the SVM classifier.

Dataset	Measure	ROS	SMOTE	B1-SMOTE	B2-SMOTE	D-SMOTE	SVM-SMOTE	ASN-SMOTE	HSNF	Ours
Glass0	F-measure	0.530301	0.541734	0.526123	0.516375	0.532953	0.534411	0.534675	0.602030	0.534411
	G-mean	0.601794	0.615816	0.601524	0.593598	0.607877	0.609182	0.605709	0.660209	0.609182
	AUC	0.713424	0.722742	0.713424	0.709975	0.719294	0.720443	0.716872	0.743309	0.720443
Glass6	F-measure	0.958080	0.965088	0.953829	0.930674	0.958080	0.945276	0.946707	0.958080	0.969918
	G-mean	0.914001	0.920215	0.866547	0.846784	0.914001	0.882914	0.904230	0.914001	0.866615
	AUC	0.920270	0.925676	0.886937	0.870721	0.920270	0.895495	0.912162	0.920270	0.886486
Haberman	F-measure	0.828160	0.829380	0.824473	0.821434	0.826142	0.829343	0.817217	0.837676	0.827415
	G-mean	0.562644	0.568385	0.578258	0.589139	0.576225	0.578190	0.566170	0.582197	0.569012
	AUC	0.619363	0.623529	0.626912	0.631776	0.625335	0.629641	0.615844	0.637048	0.623391
Segment0	F-measure	0.996793	0.997469	0.997550	0.980845	0.996962	0.997804	0.997297	0.997215	0.998399
	G-mean	0.992132	0.992385	0.996694	0.980201	0.991883	0.997807	0.992214	0.992130	0.994570
	AUC	0.992147	0.992400	0.996700	0.980378	0.991895	0.997810	0.992230	0.992146	0.994588
Vehicle2	F-measure	0.971002	0.967272	0.967094	0.933172	0.953985	0.953629	0.898527	0.915611	0.967481
	G-mean	0.944349	0.909759	0.942534	0.897575	0.931006	0.917161	0.869593	0.910954	0.951065
	AUC	0.946044	0.919651	0.944087	0.904801	0.934057	0.921386	0.882239	0.921059	0.951930
Bupa	F-measure	0.704325	0.694980	0.711783	0.699346	0.709184	0.715624	0.744594	0.770733	0.717398
	G-mean	0.668782	0.654735	0.673038	0.658637	0.669805	0.673154	0.610514	0.570250	0.674085
	AUC	0.673243	0.658773	0.676461	0.661199	0.672414	0.676071	0.639823	0.642578	0.677890
Circles	F-measure	0.559042	0.565391	0.543286	0.540000	0.555683	0.530181	0.869336	0.869336	0.506253
	G-mean	0.498356	0.518600	0.493089	0.494078	0.526612	0.487759	0.000000	0.000000	0.376832
	AUC	0.502747	0.527061	0.504091	0.505131	0.539091	0.496687	0.500000	0.500000	0.470768
Heart	F-measure	0.832220	0.825175	0.809413	0.740777	0.842484	0.844728	0.859444	0.821899	0.851159
	G-mean	0.816057	0.805249	0.787567	0.744339	0.820836	0.806010	0.824550	0.762947	0.827000
	AUC	0.819155	0.811993	0.799565	0.768442	0.826159	0.814070	0.830145	0.776473	0.829094
Paw02A-600-5-70-Bi	F-measure	0.536388	0.701398	0.492904	0.635265	0.683761	0.590666	0.655646	0.540223	0.704004
	G-mean	0.311871	0.190179	0.367970	0.239633	0.166730	0.172690	0.237087	0.251343	0.145059
	AUC	0.526333	0.534333	0.569667	0.495667	0.516000	0.519000	0.511667	0.542000	0.522000
Pima	F-measure	0.786832	0.787154	0.767122	0.753985	0.785153	0.793379	0.797480	0.811117	0.784504
	G-mean	0.698854	0.717443	0.722432	0.713669	0.716152	0.716132	0.702802	0.677064	0.720487
	AUC	0.708899	0.721959	0.725445	0.720274	0.719298	0.722409	0.713139	0.696519	0.724703
Wine	F-measure	0.797127	0.901243	0.795039	0.854599	0.861591	0.833828	0.864115	0.912065	0.914572
	G-mean	0.786843	0.880586	0.806206	0.842016	0.853934	0.825508	0.868566	0.854781	0.870528
	AUC	0.828874	0.889033	0.845296	0.861703	0.870346	0.851905	0.892713	0.868167	0.884964
Glass2	F-measure	0.482895	0.453931	0.574922	0.566939	0.467665	0.930733	0.663178	nan	0.449262
	G-mean	0.539378	0.516796	0.587323	0.581115	0.525512	0.093370	0.623782	nan	0.536406
	AUC	0.635000	0.621538	0.647222	0.642714	0.629060	0.496795	0.637500	nan	0.649957
Wisconsin	F-measure	0.972950	0.971880	0.968785	0.960257	0.971655	0.969909	0.970531	0.971013	0.971435
	G-mean	0.962578	0.963726	0.964103	0.964983	0.961609	0.963267	0.961609	0.955654	0.963861
	AUC	0.962781	0.963904	0.964524	0.965483	0.961821	0.963564	0.961821	0.956257	0.964347
Vehicle3	F-measure	0.722772	0.700832	0.731076	0.794052	0.728182	0.778698	0.654003	0.763965	0.724712
	G-mean	0.582622	0.571808	0.593452	0.624057	0.600526	0.603956	0.561765	0.516826	0.668837
	AUC	0.699251	0.677631	0.691158	0.698831	0.686633	0.690879	0.677421	0.639788	0.713647
Page-Blocks-1-3_vs_4	F-measure	0.934276	0.924218	0.941064	0.937782	0.925268	0.932811	0.901476	0.941910	0.957554
	G-mean	0.861855	0.709792	0.731908	0.887492	0.664296	0.849158	0.750104	0.740617	0.923964
	AUC	0.865356	0.766429	0.801735	0.899063	0.767178	0.861273	0.792209	0.810211	0.925846
Vehicle1	F-measure	0.755676	0.796241	0.764927	0.784252	0.779823	0.794132	0.749214	0.774919	0.754285
	G-mean	0.510452	0.704631	0.640336	0.658224	0.599178	0.585874	0.491552	0.589687	0.708149
	AUC	0.659012	0.740867	0.707311	0.725659	0.698767	0.683853	0.651824	0.670815	0.737543
Moon	F-measure	0.926943	0.928721	0.852078	0.837540	0.928868	0.907767	0.933536	0.933565	0.913002
	G-mean	0.880321	0.881935	0.831905	0.819574	0.882073	0.871881	0.871895	0.867083	0.876478
	AUC	0.881927	0.883432	0.839399	0.828384	0.883596	0.874415	0.873937	0.869437	0.878910

Table 7. The experimental results of 16 datasets obtained using the NB classifier.

Dataset	Measure	ROS	SMOTE	B1-SMOTE	B2-SMOTE	D-SMOTE	SVM-SMOTE	ASN-SMOTE	HSNF	Ours
Glass0	F-measure	0.584039	0.611811	0.610951	0.579302	0.615926	0.587480	0.605467	0.592174	0.633575
	G-mean	0.622423	0.639171	0.621033	0.602568	0.642306	0.630411	0.623013	0.604409	0.653670
	AUC	0.698153	0.704803	0.686823	0.680172	0.704803	0.705296	0.690517	0.672783	0.708005
Glass6	F-measure	0.947480	0.960036	0.953849	0.938547	0.960036	0.947551	0.951182	0.956721	0.973951
	G-mean	0.853668	0.864358	0.859057	0.813774	0.864358	0.853200	0.841631	0.789408	0.800925
	AUC	0.867568	0.878378	0.872973	0.831532	0.878378	0.867568	0.856306	0.827748	0.830000
Haberman	F-measure	0.828857	0.832079	0.828261	0.827805	0.828909	0.820771	0.814407	0.826726	0.822139
	G-mean	0.555257	0.547264	0.560282	0.564920	0.550633	0.545058	0.555951	0.541718	0.585909
	AUC	0.615335	0.613529	0.619363	0.619363	0.615703	0.609036	0.612647	0.609453	0.633252
Segment0	F-measure	0.877323	0.881756	0.882582	0.852820	0.884788	0.856628	0.899689	0.896536	0.861288
	G-mean	0.878463	0.882417	0.713327	0.706553	0.882792	0.853318	0.895044	0.892213	0.861994
	AUC	0.884297	0.887834	0.727557	0.714419	0.887582	0.860042	0.898443	0.895919	0.869142
Vehicle2	F-measure	0.853901	0.849748	0.863651	0.843131	0.831975	0.825401	0.826757	0.821205	0.876390
	G-mean	0.766475	0.776402	0.744937	0.698694	0.755702	0.746948	0.728569	0.725569	0.848280
	AUC	0.770573	0.778647	0.754008	0.710870	0.757620	0.747727	0.732417	0.729863	0.849999
Bupa	F-measure	0.423293	0.404152	0.454374	0.473192	0.423053	0.416517	0.528334	0.592342	0.397604
	G-mean	0.492791	0.479458	0.514228	0.527847	0.490240	0.488813	0.562951	0.543212	0.472350
	AUC	0.560209	0.550086	0.564618	0.571293	0.553190	0.555086	0.593325	0.563658	0.544138
Circles	F-measure	0.793573	0.811721	0.792105	0.762907	0.802923	0.820122	0.860163	0.875740	0.803551
	G-mean	0.742557	0.755825	0.727804	0.711449	0.744333	0.749813	0.724781	0.643375	0.753886
	AUC	0.743990	0.757020	0.729333	0.713939	0.745343	0.750414	0.733172	0.682909	0.756657
Heart	F-measure	0.841582	0.854410	0.852284	0.844633	0.849584	0.853333	0.855191	0.860838	0.856158
	G-mean	0.824198	0.832864	0.834608	0.828383	0.830650	0.833857	0.831692	0.823009	0.838283
	AUC	0.824275	0.833442	0.834928	0.828442	0.830942	0.834094	0.832428	0.827428	0.838442
Paw02A-600-5-70-Bi	F-measure	0.873050	0.868485	0.855711	0.865946	0.877095	0.882075	0.903834	0.895339	0.866875
	G-mean	0.788998	0.780403	0.795294	0.820498	0.785864	0.786892	0.766071	0.749844	0.796989
	AUC	0.791000	0.783000	0.798000	0.823000	0.790000	0.790000	0.775000	0.759000	0.798000
Pima	F-measure	0.803807	0.808776	0.778757	0.780935	0.804844	0.795182	0.811220	0.809916	0.790895
	G-mean	0.732191	0.743950	0.724137	0.730668	0.733402	0.725824	0.732729	0.707942	0.735431
	AUC	0.734579	0.745760	0.724464	0.730941	0.735509	0.727614	0.736067	0.716158	0.736342
Wine	F-measure	0.939489	0.941434	0.953247	0.920524	0.936655	0.939489	0.946818	0.925713	0.931648
	G-mean	0.916704	0.913541	0.919517	0.894662	0.889413	0.916704	0.918615	0.858259	0.912896
	AUC	0.921645	0.919740	0.927359	0.901558	0.901861	0.921645	0.924502	0.878571	0.919264
Glass2	F-measure	0.575818	0.599571	0.686661	0.593995	0.626925	0.614040	0.790891	0.925648	0.717979
	G-mean	0.563522	0.562954	0.589568	0.537975	0.583211	0.537531	0.490266	0.215936	0.612575
	AUC	0.612564	0.602692	0.615385	0.591795	0.615449	0.577051	0.589231	0.531410	0.616346
Wisconsin	F-measure	0.967787	0.968983	0.966350	0.966256	0.967866	0.966459	0.967866	0.966577	0.967467
	G-mean	0.962578	0.963726	0.964103	0.964983	0.961609	0.963267	0.961609	0.961423	0.966162
	AUC	0.962781	0.963904	0.964524	0.965483	0.961821	0.963564	0.961821	0.961657	0.966607
Vehicle3	F-measure	0.752862	0.740004	0.720141	0.727925	0.750528	0.773588	0.758777	0.778749	0.730772
	G-mean	0.670801	0.665997	0.671001	0.675118	0.669536	0.670078	0.669758	0.663530	0.678816
	AUC	0.672596	0.666875	0.674832	0.677963	0.671015	0.672563	0.670897	0.667185	0.681919
Page-Blocks-1-3_vs_4	F-measure	0.951820	0.953325	0.848274	0.924279	0.952142	0.947185	0.953500	0.949460	0.937896
	G-mean	0.690125	0.691147	0.819832	0.671390	0.689994	0.687009	0.670208	0.645591	0.680500
	AUC	0.794082	0.795206	0.835655	0.774981	0.794082	0.790712	0.775193	0.752959	0.783957
Vehicle1	F-measure	0.744411	0.748857	0.697982	0.677231	0.761808	0.760919	0.744231	0.765523	0.696990
	G-mean	0.680516	0.679827	0.667733	0.660802	0.694179	0.665877	0.677618	0.658792	0.666514
	AUC	0.683323	0.682812	0.676767	0.675582	0.697750	0.669480	0.681975	0.661783	0.676773
Moon	F-measure	0.929262	0.928177	0.900547	0.892943	0.928700	0.915368	0.935604	0.935736	0.914704
	G-mean	0.882415	0.881454	0.838404	0.846343	0.881937	0.878626	0.883392	0.874005	0.877984
	AUC	0.883932	0.882932	0.840894	0.848389	0.883430	0.880915	0.884940	0.875942	0.880412

We also compute the mean values of F-measure, G-mean, and AUC metrics across 17 datasets for KNN, RF, SVM and NB classifiers. As illustrated in Figure 5, when using the KNN classifier for evaluation, our method performs the best on the F-measure metric, with a score far exceeding other comparison methods. Meanwhile, in terms of the G-mean metric, our method's performance is second only to SMOTE. When using the RF classifier, our method achieves the third-highest F-measure score, behind only the ROS and ASN-SMOTE methods. However, our method achieves the highest value on the G-mean metric. When using the SVM classifier for evaluation, our method performs the best on the G-mean and AUC metrics, the F-measure metric is second only to SMOTE, SVM-SMOTE and ASN-SMOTE. when using the NB classifier for evaluation, our method performs also the best on the G-mean and AUC metrics, with a score exceeding other comparison methods. Meanwhile, in terms of the F-measure metric, Our method surpasses ROS, SMOTE, B1-SMOTE, B2-SMOTE, and SVM-SMOTE. Overall, these results show that our method has excellent performance in comprehensive classification and sensitivity to the identification of minority class samples.



(a) KNN Classifier

(b) RF Classifier

(c) SVM Classifier

(d) NB Classifier

Figure 5. The average value of indicators of nine methods.

Furthermore, we compute the average rankings of F-measure, G-mean, and AUC across 17 datasets for our method and eight others on these four classifiers. As can be seen in Figure 6 to Figure 9, a lower vertical axis value indicates a higher ranking. In Figure 6, we observe that for the KNN classifier, our method achieves the top rankings

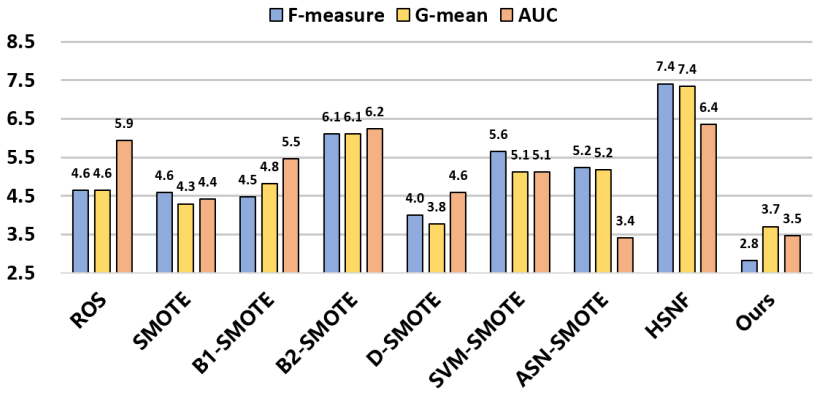


Figure 6. The average ranking of nine methods on KNN classifier.

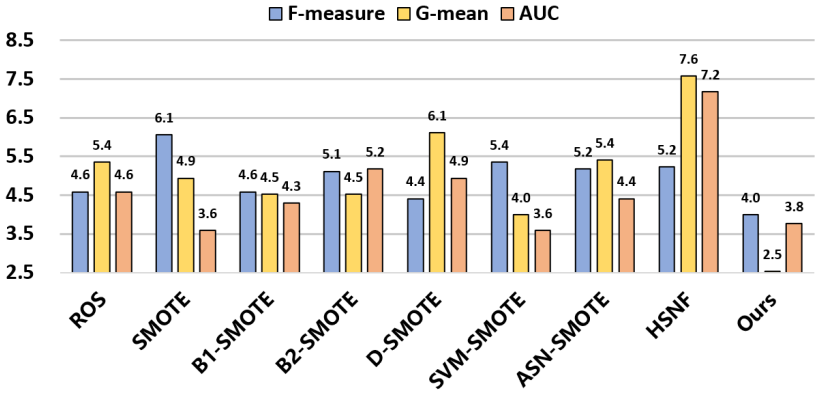


Figure 7. The average ranking of nine methods on RF classifier.

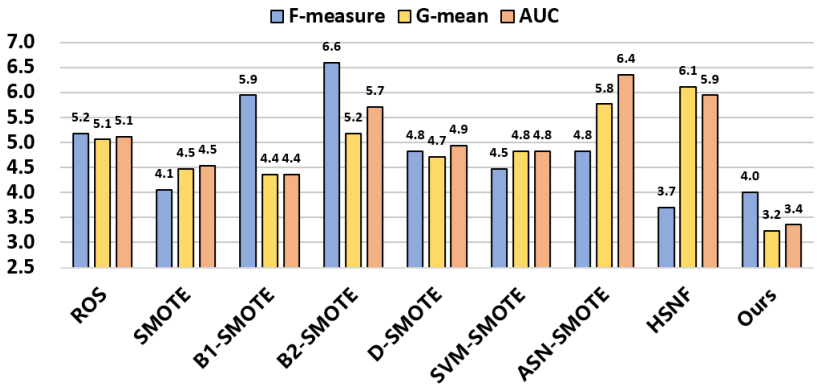


Figure 8. The average ranking of nine methods on SVM classifier.

in both the F-measure and G-mean metrics. Particularly for the F-measure, it outperforms the second-placed D-SMOTE by 1.2 rankings, while for the AUC metric, it is only 0.1 rankings behind the first-ranked ASN-SMOTE. In Figure 7, when using the RF classifier, our method still achieves the highest rankings in the F-measure and G-mean metrics. Among them, the ranking of the G-mean metric is significantly higher than that of other methods, and for the AUC metric, it is only 0.2 rankings lower compared to the method with the highest ranking. On the

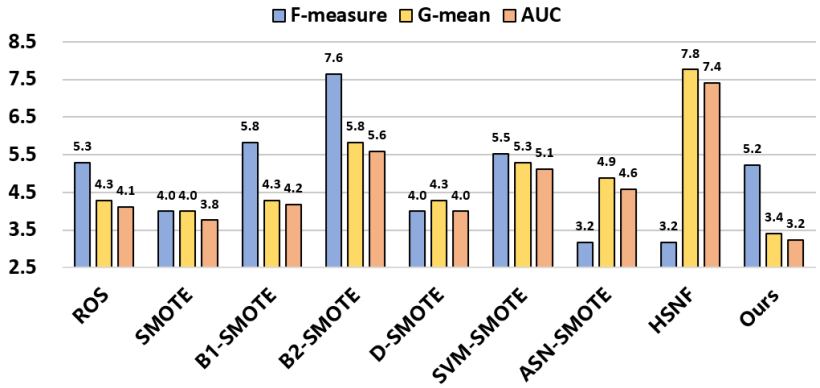


Figure 9. The average ranking of nine methods on NB classifier.

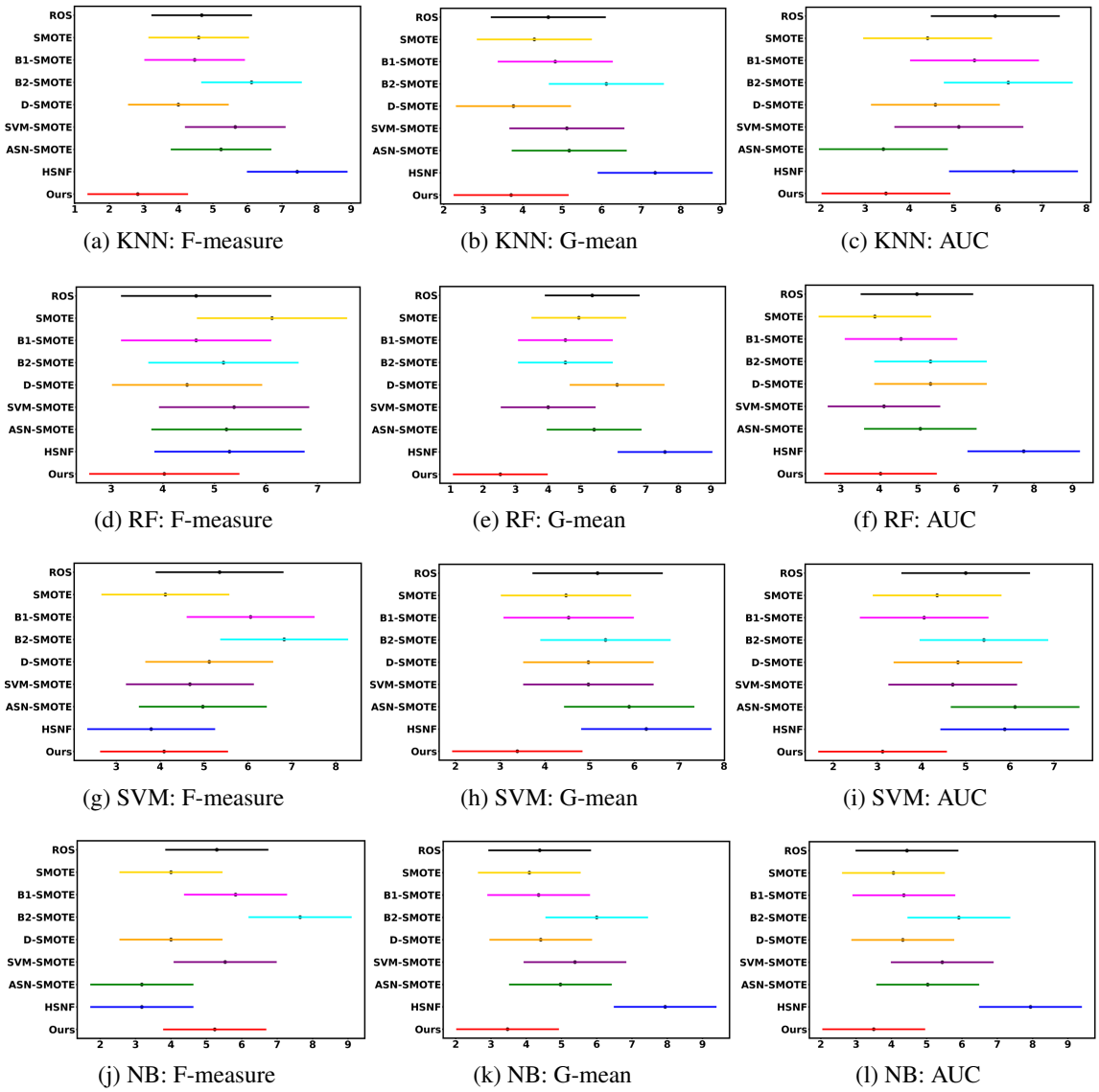
SVM classifier as depicted in Figure 8, the ranking for the G-mean metric is notably higher than that of the other methods, regarding the F-measure metric, our method ranks only 0.3 positions lower compared to the method with the top ranking. Finally, as shown in Figure 9, our method achieves the highest rankings in the G-mean and AUC metrics on the NB classifier. These show our method outperforms others.

Statistical test. *Friedman test* and *Nemenyi test* are commonly used nonparametric statistical methods for comparing the performance differences between multiple methods on multiple datasets in order to validate the statistical significance between methods. The Friedman test compares the overall performance of k algorithms on N datasets, while the Nemenyi test compares pairwise algorithm performance. Assuming no significant differences among nine methods, when the significance level is set at 0.05, the critical value is $F = 2.011$. Table 8 shows Friedman test results for three metrics on four classifiers. Only the RF classifier's F-measure and the SVM classifier's G-mean is below 2.011; others exceed it. Rejecting the null hypothesis implies significant differences among methods.

Table 8. Friedman test.

	F-measure	G-mean	AUC
KNN Classifier	4.858 (Reject)	3.467 (Reject)	3.144 (Reject)
RF Classifier	0.854 (Accept)	5.716 (Reject)	3.536 (Reject)
SVM Classifier	2.375 (Reject)	1.665 (Accept)	2.289 (Reject)
NB Classifier	6.367 (Reject)	4.921 (Reject)	4.870 (Reject)

Next, we perform the Nemenyi post-hoc test to analyze pairwise algorithm differences after rejecting the null hypothesis. The critical difference (CD) is calculated by $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$, where k is the number of methods and N the number of datasets. With $k = 9$ and $N = 17$, we find $CD = 2.914$ (using q_α = 3.102). Methods with average ranking differences exceeding CD are significantly different. Figure 10 visualizes the results: horizontal lines represent method rankings, with segment lengths equal to CD. Centers mark average rankings. Non-overlapping lines indicate significant differences; overlapping lines suggest no significant difference, with the leftmost method performing better. It can be observed that our method demonstrates the best overall performance: with the KNN classifier, it significantly outperforms B2-SMOTE in F-measure and HSNF in G-mean; with the RF classifier, it exceeds D-SMOTE, ASN-SMOTE, and HSNF in G-mean while also outperforming HSNF in AUC; with the SVM classifier, it surpasses ASN-SMOTE in AUC; and with the NB classifier, it outperforms HSNF in both G-mean and AUC.



(a) KNN: F-measure (b) KNN: G-mean (c) KNN: AUC

(d) RF: F-measure (e) RF: G-mean (f) RF: AUC

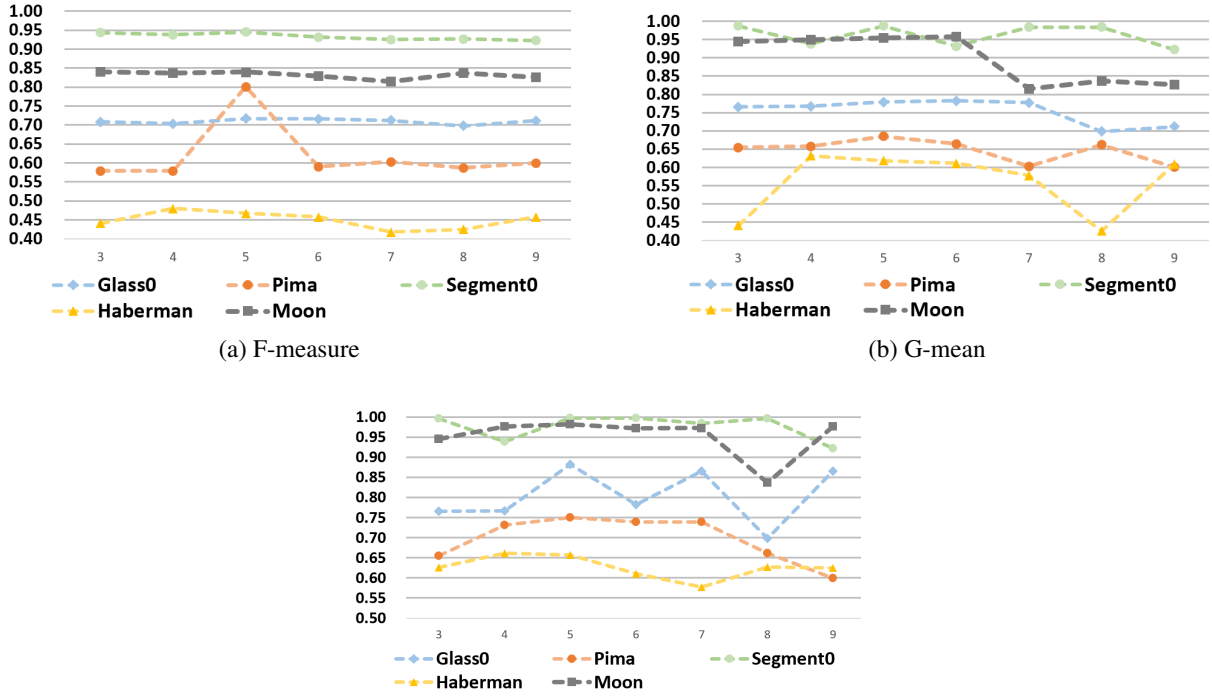
(g) SVM: F-measure (h) SVM: G-mean (i) SVM: AUC

(j) NB: F-measure (k) NB: G-mean (l) NB: AUC

Figure 10. Results of Nemenyi test.

4.3. Parameter analysis

To explore the impact of parameter k on the method's performance, taking the KNN classifier as an example, we randomly selected five datasets: Glass0, Pima, Segment0, Haberman, and Moon for experimentation. In our implementation, we set the default parameter $k = 5$. Figure 11 illustrates the trends in F-measure, G-mean, and AUC metrics as the parameter k varies. The figure reveals that the classifier's performance fluctuates with changes in the k value. For instance, the Pima dataset achieves optimal performance at $k = 5$, followed by an overall declining performance trend. In contrast, for the Haberman dataset, optimal performance is attained at $k = 4$, and subsequently, as k increases, the performance initially declines and then rises. Therefore, during the classification process, the optimal k values vary across different datasets, highlighting the significance of parameter selection for model performance.



(a) F-measure

(b) G-mean

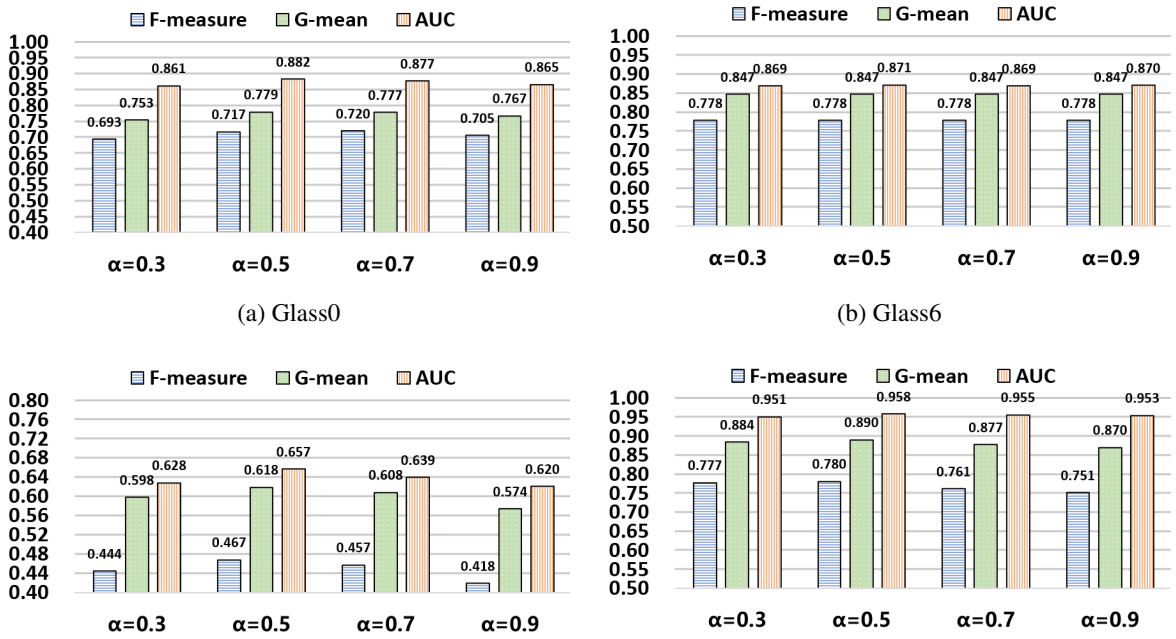
(c) AUC

Figure 11. The trends of datasets' metrics with the change of parameter k.

As illustrated in Figure 12, we examine the impact of the parameter α during the sample synthesis phase on the values of F-measure, G-mean, and AUC. Taking the KNN classifier as an example, we randomly select four datasets (Glass0, Glass6, Haberman, and Vehicle2) for validation. The results show that adjustments to α generally have an insignificant effect on these three metrics; however, when α is set to 5, the values exhibit relatively higher levels.

4.4. Comparison of runtime

Under the same experimental conditions, we compared the runtime of our proposed method with that of the comparative methods. The experimental results are shown in Table 9. The ROS method had the shortest runtime, while our proposed method exhibited slightly longer runtime than the other methods due to the need for iterative optimization during the sample weight calculation phase to ensure the rationality of generated sample distribution and the clarity of class boundaries. Nevertheless, this increase in computational complexity significantly enhanced classification performance, indicating that our proposed method achieves superior classification results through a reasonable trade-off between runtime and performance.



(a) Glass0 (b) Glass6

(c) Haberman (d) Vehicle2

Figure 12. The trends of datasets' metrics with the change of parameter α.

Table 9. Runtime comparison between the proposed method and comparable methods.

Dataset	Runtime (s)								
	ROS	SMOTE	B1-SMOTE	B2-SMOTE	D-SMOTE	SVM-SMOTE	ASN-SMOTE	HSNF	Ours
Glass0	0.0000	0.0000	0.0058	0.0054	0.0021	0.0064	0.1146	0.3002	0.3249
Glass6	0.0010	0.0020	0.0054	0.0053	0.0022	0.0060	0.0474	0.2801	0.1602
Haberman	0.0000	0.0158	0.0059	0.0050	0.0020	0.0000	0.1191	0.4004	0.3786
Segment0	0.0020	0.0173	0.0319	0.0290	0.0113	0.0847	7.3146	61.1420	31.7824
Vehicle2	0.0010	0.0040	0.0163	0.0118	0.0035	0.0319	2.0537	7.9226	6.5159
Bupa	0.0000	0.0000	0.0109	0.0109	0.0030	0.0056	0.3260	0.6918	0.9334
Circles	0.0004	0.0000	0.0067	0.0100	0.0069	0.0134	0.3069	2.1130	1.2359
Heart	0.0000	0.0000	0.0069	0.0063	0.0020	0.0074	0.2168	0.5501	0.8462
Paw02A-600-5-70-BI	0.0000	0.0032	0.0064	0.0067	0.0025	0.0078	0.8882	2.3900	3.6021
Pima	0.0000	0.0000	0.0120	0.0132	0.0000	0.0196	0.9647	3.2460	3.8889
Wine	0.0000	0.0000	0.0066	0.0086	0.0021	0.0030	0.0950	0.2686	0.3305
Glass2	0.0000	0.0000	0.0066	0.0086	0.0021	0.0030	0.0950	0.2686	0.3305
Wisconsin	0.0000	0.0160	0.0125	0.0094	0.0030	0.0045	0.8179	3.2920	3.2228
Vehicle3	0.0000	0.0040	0.0099	0.0107	0.0036	0.0192	1.3312	6.2503	5.3545
Page-Blocks-1-3_vs_4	0.0000	0.0000	0.0055	0.0043	0.0027	0.0056	0.1043	1.4527	0.3858
Vehicle1	0.0000	0.0000	0.0124	0.0114	0.0048	0.0306	1.5475	7.2477	5.9995
Moon	0.0010	0.0041	0.0075	0.0076	0.0030	0.0100	0.3481	4.3213	1.4824

5. Conclusion

This paper proposes an innovative SMOTE-based method to address the challenges posed by imbalanced data. The core of this method is to identify and remove noise data that may interfere with classification to ensure data quality; Subsequently, the number and location of synthetic samples are determined based on the density and distribution characteristics of the surrounding data of the sample. The experimental results show that our method not only successfully avoids the negative impact of noise, but also generates more reliable samples, resulting in a clearer classification boundary. Looking ahead, we plan to further improve the method, enhance its performance, and explore its potential application in more complex multi-classification tasks, in order to better meet the needs of practical applications.

Acknowledgements

This work was supported in part by the Natural Science Foundation of the Anhui Higher Education Institutions of China (No. 2023AH040149), the Anhui Provincial Natural Science Foundation (No. 2208085MF168), and the Undergraduate Training Programs for Innovation and Entrepreneurship (No. S202510360348).

REFERENCES

1. Ashkan Zakaryazad and Ekrem Duman, *A profit-driven Artificial Neural Network with applications to fraud detection and direct marketing*, Neurocomputing, vol. 175, pp. 121–131, 2016.
2. Haibo He and Edwardo A Garcia, *Learning from imbalanced data*, IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.
3. Maciej A. Mazurowski, Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker and Georgia D. Tourassi, *Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance*, Neural Networks, vol. 21, no. 2, pp. 427–436, 2008.
4. Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi, *Calibrating Probability with Undersampling for Unbalanced Classification*, 2015 IEEE Symposium Series on Computational Intelligence, pp. 159–166, 2015.
5. Bartosz Krawczyk, *Learning from imbalanced data: open challenges and future directions*, Progress in Artificial Intelligence, vol. 5, no. 4, pp. 221–232, 2016.
6. Yanmin Sun, Andrew K. C. Wong and Mohamed S. Kamel, *Classification of Imbalanced Data: a Review*, International Journal of Pattern Recognition and Artificial Intelligence, vol. 23, pp. 687–719, 2009.
7. Yitian Xu, Qian Wang, Xinying Pang and Ying Tian, *Maximum margin of twin spheres machine with pinball loss for imbalanced data classification*, Applied Intelligence, vol. 48, no. 1, pp. 23–34, 2018.
8. Alberto Fernández, Salvador García, Francisco Herrera and Nitesh V. Chawla, *SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary*, Journal of Artificial Intelligence Research, vol. 61, no. 1, pp. 863–905, 2018.
9. Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall and W Philip Kegelmeyer, *SMOTE: synthetic minority over-sampling technique*, Journal of Artificial Intelligence Research, vol. 16, no. 1, pp. 321–357, 2002.
10. Hui Han, Wen-Yuan Wang and Bing-Huan Mao, *Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning*, Advances in Intelligent Computing, Berlin, Heidelberg, pp. 878–887, 2005.
11. Georgios Douzas, Fernando Bacao and Felix Last, *Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE*, Information Sciences, vol. 465, pp. 1–20, 2018.
12. Rok Blagus and Lara Lusa, *SMOTE for high-dimensional class-imbalanced data*, BMC Bioinformatics, vol. 14, no. 1, p. 106, 2013.
13. Sebastián Maldonado, Julio López and Carla Vairetti, *An alternative SMOTE oversampling strategy for high-dimensional datasets*, Applied Soft Computing, vol. 76, pp. 380–389, 2019.
14. Haibo He, Yang Bai, Edwardo A Garcia and Shutao Li, *ADASYN: Adaptive synthetic sampling approach for imbalanced learning*, Proceedings of IEEE International Joint Conference on Neural Networks, pp. 1322–1328, 2008.
15. Chumphol Bunkhumpornpat, Krung Sinapiromsaran and Chidchanok Lursinsap, *DBSMOTE: Density-Based Synthetic Minority Over-sampling Technique*, Applied Intelligence, vol. 36, no. 3, pp. 664–684, 2012.
16. Jorge De La Calleja and Olac Fuentes, *A distance-based over-Sampling method for learning from imbalanced data sets*, The Florida AI Research Society, pp. 634–635, 2007.
17. György Kovács, *An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets*, Applied Soft Computing, vol. 83, p. 105662, 2019.
18. *Two Modifications of CNN*, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-6, no. 11, pp. 769–772, 1976.
19. Dennis L. Wilson, *Asymptotic Properties of Nearest Neighbor Rules Using Edited Data*, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-2, no. 3, pp. 408–421, 1972.
20. Charu C. Aggarwal, Alexander Hinneburg and Daniel A. Keim, *On the Surprising Behavior of Distance Metrics in High Dimensional Space*, International Conference on Database Theory, Springer Berlin Heidelberg, pp. 420–434, 2001.

21. Hongfang Zhou, Zongling Wu, Ningning Xu and Hao Xiao, *PDR-SMOTE: an imbalanced data processing method based on data region partition and K nearest neighbors*, International Journal of Machine Learning and Cybernetics, vol. 14, no. 12, pp. 4135–4150, 2023.
22. Lilong Duan, Wei Xue, Xiaolei Gu, Xiao Luo and Yongsheng He, *HSNF: Hybrid sampling with two-step noise filtering for imbalanced data classification*, Intelligent Data Analysis, vol. 27, no. 6, pp. 1573–1593, 2023.
23. Chen Si, Gong-De Guo and Li-Fei Chen, *Clustering Ensembles Based Classification Method for Imbalanced Data Sets*, Pattern Recognition and Artificial Intelligence, vol. 23, no. 6, pp. 772–780, 2010.
24. Sukarna Barua, Md. Monirul Islam, Xin Yao and Kazuyuki Murase, *MWMOTE–Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning*, IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 2, pp. 405–425, 2014.
25. Lilong Duan, Wei Xue, Jun Huang and Xiao Zheng, *Joint Sample Position Based Noise Filtering and Mean Shift Clustering for Imbalanced Classification Learning*, Tsinghua Science and Technology, vol. 29, no. 1, pp. 216–231, 2024.
26. Wei Xue, Lilong Duan, Xudong Hong and Xiao Zheng, *Adaptive weighting and nearest neighbor-based area control for imbalanced data classification*, Applied Soft Computing, vol. 177, p. 113171, 2025.
27. Nianyin Zeng, Hong Qiu, Zidong Wang, Weibo Liu, Hong Zhang and Yurong Li, *A new switching-delayed-PSO-based optimized SVM algorithm for diagnosis of Alzheimer's disease*, Neurocomputing, vol. 320, pp. 195–202, 2018.
28. Ashhadul Islam, Samir Brahim Belhaouari, Atiq Ur Rehman and Halima Bensmail, *KNNOR: An oversampling technique for imbalanced datasets*, Applied Soft Computing, vol. 115, p. 108288, 2022.
29. Xinkai Yi, Yingying Xu, Qian Hu, Sujatha Krishnamoorthy, Wei Li and Zhenzhou Tang, *ASN-SMOTE: a synthetic minority oversampling method with adaptive qualified synthesizer selection*, Complex & Intelligent Systems, vol. 8, pp. 2247–2272, 2022.