scientific reports

Check for updates

OPEN

An improved SMOTE algorithm for enhanced imbalanced data classification by expanding sample generation space

Ying Li[1,2], Yali Yang[1], Peihua Song[1,2]✉, Lian Duan[3,4] & Rui Ren[1]

Class imbalance in datasets often degrades the performance of classification models. Although the Synthetic Minority Over-sampling Technique (SMOTE) and its variants alleviate this issue by generating synthetic samples, they frequently overlook local density and distribution characteristics. Consequently, developing methods that incorporate local spatial information to synthesize samples that better preserve the original data distribution is critical for improving model robustness in class-imbalanced scenarios. To address this gap, we propose an enhanced SMOTE algorithm (ISMOTE), which modifies the spatial constraints for synthetic sample generation. Unlike SMOTE, the proposed method first generates a base sample between two original samples. Then the Euclidean distance between the two samples is multiplied by a random number to generate a random quantity. This random quantity is added or subtracted based on the distance between the base sample and the original samples, ensuring that new samples are generated around the two original samples. By adaptively expanding the synthetic sample generation space, ISMOTE effectively alleviates distortions in local data distribution and density. This study compared the ISMOTE algorithm with seven mainstream oversampling algorithms, using three classifiers on thirteen public datasets from the KEEL, UCI, and Kaggle databases. Comparative analysis of 2D and 3D scatter plots revealed that ISMOTE yields more realistic data distributions. Experimental results demonstrated relative improvements in classifier performance, with F1-score, G-mean, and AUC increasing by 13.07%, 16.55%, and 7.94%, respectively. Furthermore, ISMOTE's parameter adaptability enables its application to multi-class imbalanced datasets.

Keywords Classification, Imbalanced datasets, Oversampling, SMOTE

Class imbalance is a prevalent issue in real-world datasets, characterized by a significant disparity in the number of samples across different categories. This phenomenon is ubiquitous across diverse domains, including medical disease diagnosis[1,2], financial fraud detection[3,4], software defect prediction[5,6], and hardware fault detection[7,8]. In binary classification tasks, the imbalance ratio (IR)—defined as the ratio of minority-class to majority-class samples—can range from tens to hundreds in practical applications, presenting substantial challenges for conventional machine learning algorithms[9]. Traditional machine learning models trained on imbalanced datasets often exhibit a bias toward the majority class[10], resulting in the misclassification of minority-class samples. Such errors can have critical real-world implications. For instance, in medical diagnostics, where cancer patients typically represent the minority class, a false negative (i.e., misclassifying a cancer patient as healthy) may delay life-saving interventions, with irreversible consequences. Thus, improving minority-class recognition accuracy is essential, underscoring the importance of research on class-imbalance mitigation.

Various techniques are currently available to address class imbalance[11,12]. These techniques are categorized into data-level and algorithm-level[13]. At the data level, methods include undersampling the majority class[14,15], oversampling the minority class[16,17], and hybrid sampling, which combines undersampling of the majority class with oversampling of the minority class[18–20] to achieve class balance. At the algorithm level, machine learning

[1]School of Logistics Management and Engineering, Nanning Normal University, Nanning 530001, Guangxi, China. [2]Guangxi Colleges and Universities Key laboratory of Intelligent Logistics Technology, Nanning Normal University, Nanning 530001, Guangxi, China. [3]School of Natural Resources and Surveying, Nanning Normal University, Nanning 530001, Guangxi, China. [4]Guangxi Natural Resources Intelligent Monitoring and Big Data Analysis Engineering Experimental Teaching Center, Nanning 530001, Guangxi, China. ✉email: 455756282@qq.com

algorithms are optimized to adapt to imbalanced data. The main methods include cost-sensitive learning[21,22], threshold moving strategies[23,24], ensemble learning[25,26] and neural networks[27,28]. Compared with algorithm-level modifications, data-level techniques offer greater implementation flexibility and model independence, contributing to their wider practical adoption.

The Synthetic Minority Oversampling Technique (SMOTE) is one of the most common oversampling methods. Compared to other oversampling methods, SMOTE generates more diverse synthetic samples, which helps improve the generalization ability of models. Additionally, the generation logic of SMOTE is relatively simple and easy to implement, and it performs well in addressing class imbalance across various fields. Especially in small and medium-sized datasets, SMOTE is often more effective than other methods. Relevant studies[29,30] indicate that SMOTE is one of the most widely used data imbalance handling techniques among oversampling methods. However, SMOTE's linear interpolation mechanism for synthetic sample generation presents two inherent limitations: (1) in high-density regions of minority class samples, excessive synthetic instances may be produced, potentially inducing model overfitting; and (2) the linearly interpolated samples may deviate from the underlying data distribution. To address these issues, this study examines SMOTE's sample generation mechanism and proposes an Improved SMOTE (ISMOTE) algorithm. ISMOTE extends the feasible solution space for synthetic sample generation, effectively mitigating both the density over-amplification problem and distributional distortion inherent in conventional SMOTE.

The remainder of this paper is structured as follows. Section 2 presents a comprehensive review of existing oversampling techniques. Section 3 details the fundamental principles and implementation procedures of the standard SMOTE algorithm, followed by a thorough presentation of our proposed Improved SMOTE (ISMOTE) algorithm, including its conceptual framework, algorithmic workflow, implementation steps, and formal pseudocode. Section 4 describes the experimental setup, including: (1) the thirteen benchmark datasets employed, (2) the evaluation metrics for classification performance assessment, (3) the experimental design methodology, and (4) a comprehensive analysis of the empirical results. Finally, Sect. 5 concludes the study with key findings and contributions.

The main contributions of this paper are summarized as follows:

Expansion of Sample Generation Space : ISMOTE modifies the sample generation conditions of the SMOTE algorithm, significantly expanding the space for generating new samples. Unlike SMOTE, which generates samples solely through linear interpolation, ISMOTE allows new samples to be generated not only between existing samples but also around them. This approach effectively mitigates the issue of generating excessive samples in high-density regions, thereby reducing the risk of overfitting and improving the quality of the generated data.

Improved Sample Distribution : ISMOTE introduces random quantities to dynamically adjust the positions of new samples. This ensures that the generated samples better align with the underlying distribution patterns of the original data. By enhancing the diversity and representativeness of the synthetic samples, ISMOTE improves the generalization capability of classifiers, particularly in imbalanced data scenarios.

Experimental Validation : Extensive experiments were conducted on multiple public datasets to validate the effectiveness of ISMOTE. The results demonstrate that ISMOTE consistently outperforms existing mainstream oversampling algorithms.

Related work

Oversampling techniques are widely used to address class imbalance by increasing the number of minority class samples, thereby improving classifier performance. These techniques can be categorized based on the data types they handle, including numerical data, categorical data, and image data[31]. This study focuses on categorical data, as it is prevalent in many real-world applications such as medical diagnosis, fraud detection, and software defect prediction. Compared to undersampling, oversampling is often preferred because it preserves the original data distribution and avoids information loss[32,33].

Batista et al.[34] proposed Random Oversampling (ROS), which duplicates minority class samples randomly to balance the dataset. While ROS is simple to implement, it often leads to overfitting, as the repeated samples do not introduce new information and may amplify noise in the data. Despite its limitations, ROS has been shown to outperform undersampling in certain scenarios when evaluated using metrics such as AUC.

To surmount the limitations of ROS, Chawla et al.[35] proposed SMOTE, which generates synthetic minority samples via interpolation between existing minority samples and their k - nearest neighbors. By generating diverse synthetic samples rather than merely replicating the existent ones, SMOTE mitigates overfitting to a certain degree. However, SMOTE is not without its drawbacks. Firstly, SMOTE has the potential to generate synthetic samples within the overlapping regions between classes. This phenomenon gives rise to noisy data, thereby degrading the performance of classifiers.Secondly, in areas where minority samples are densely concentrated, SMOTE tends to generate an excessive number of synthetic samples. This situation exacerbates the problem of overfitting. Thirdly, SMOTE confines new samples to the linear paths between existing samples. As a consequence, the generated samples may deviate from the actual data distribution.

He et al.[36] proposed the adaptive synthetic sampling algorithm (ADASYN), which uses a weighted distribution based on the difficulty of learning different minority class samples. More synthetic data is generated for harder-to-learn minority samples. This method reduces the bias caused by class imbalance and adaptively shifts the classification decision boundary towards the difficult samples. However, it faces challenges in sampling boundary samples and does not handle noisy data effectively.

In addition to the classic oversampling algorithms mentioned above, several improved algorithms based on SMOTE have been developed. Han et al.[37] proposed the Borderline-SMOTE method, which effectively avoids oversampling noisy samples and reduces the generation of noisy samples by selectively oversampling boundary minority class samples. However, the voting selection strategy of this method involves a large number of high-

density minority class samples in oversampling, without considering the problem of excessive generation of minority class samples and the distribution pattern of new samples. Douzas et al.[38] proposed a simple and effective oversampling method based on K-means clustering and SMOTE, which avoids noise generation and effectively overcomes both inter-class and intra-class imbalances. Intra-class imbalance refers to the uneven distribution of samples within the same class. However, K-Means SMOTE may increase classification errors for minority samples due to its reliance on clustering, which can be sensitive to data sparsity and noise. Douzas et al.[39] proposed Geometric SMOTE (G-SMOTE) as an enhancement to the SMOTE data generation mechanism. G-SMOTE generates synthetic samples within a geometric region in the input space, centered around each selected minority instance. While the default configuration defines this region as a hypersphere, G-SMOTE allows it to deform into a hyperellipsoid. G-SMOTE effectively parameterizes the data generation process and adapts to the specific characteristics of each imbalanced dataset. However, if the minority class samples contain noise or outliers, G-SMOTE may generate unrealistic synthetic samples around them, which can degrade classification performance.

Kunakorntum et al.[40] proposed a novel oversampling technique, Synthetic Minority based on Probability Distribution (SyMProD), to handle skewed datasets. This technique standardizes the data using Z-scores and removes noisy data. The proposed method then selects minority samples based on the probability distributions of the two classes. Synthetic instances are generated from the selected points and several nearest neighbors of the minority class. Wang et al.[41] propose a new deep learning (DL) based data balancing technique using an Auxiliary-guided Conditional Variational Autoencoder (ACVAE) trained with contrastive learning. Additionally, Wang et al. investigate an ensemble method where ACVAE generates synthetic positive samples, followed by a data undersampling technique.

In the context of ensuring data privacy, Du et al.[42] propose a secure privacy-preserving SMOTE (SP2-SMOTE) sampling method. It extends traditional SMOTE by allowing parties to independently generate synthetic samples without exposing the data, while effectively preventing unauthorized label inference through minority-class nearest neighbor interpolation. The SMOTE algorithm and its variants can be combined with ensemble learning[43] or machine learning algorithms[44] to solve data imbalance problems in specific fields. Imani et al.[45] also conducted a comprehensive analysis of the performance of Random Forest and XGBoost using SMOTE, ADASYN, and GNUS at different levels of imbalance.

Although the aforesaid methods have enhanced the classification performance in imbalanced datasets, they are burdened with several limitations. Existing methods typically struggle to generate synthetic samples that accord with the authentic distribution of minority classes, especially in complex or irregular data manifolds, which leads to issues in spatial distribution. Moreover, numerous methods generate an overabundance of synthetic samples in high - density regions, resulting in overfitting and diminished classifier generalization in terms of density control. Additionally, the generation of synthetic samples in overlapping or noisy regions remains a tough nut to crack, as it can undermine classifier performance, presenting a challenge in noise handling. To tackle these limitations, we propose ISMOTE, an enhanced SMOTE algorithm that extends the spatial scope for synthetic sample generation. Differing from existing methods, ISMOTE adaptively adjusts the positions of new samples in line with local density and distribution characteristics. This ensures that synthetic samples are distributed around original samples instead of being restricted to linear paths, maintain the natural density gradients of the original minority class, and avoid over-saturation in high-density regions. By surmounting these challenges, ISMOTE bolsters the robustness of classifiers in imbalanced data scenarios, particularly in applications such as early disease detection or fraud prevention where high precision for minority classes is imperative.

An improved SMOTE algorithm

This section first outlines the concept, implementation steps, and limitations of the SMOTE algorithm. Subsequently, the ISMOTE algorithm proposed in this study is introduced, detailing its concept, flowchart, implementation steps, and pseudocode.

The SMOTE algorithm

SMOTE is one of the most well-established oversampling algorithms, effectively mitigating the overfitting issue caused by random oversampling and improving the generalization capability of models. The basic principle of SMOTE is to balance the dataset by inserting synthetic samples between minority class samples. It synthesizes new minority class samples through linear interpolation with the k-nearest neighbor algorithm. Specifically, it randomly selects a minority class sample and one of its k-nearest neighbors, then generates a new minority class sample by performing linear interpolation between the two. The steps of the algorithm are as follows.

Determine the number of minority class samples to be synthesized, denoted as $n_{samples}$.

1) For each selected minority class sample X_q, compute its Euclidean distance to all other minority class samples. Select the k nearest neighbors of X_q, where $k=5$ by default.

2) Randomly select a sample X_j from the k nearest neighbor samples and synthesize a new minority class sample X_{new} with X_q using (1).

$$X_{new} = X_q + \alpha \times (X_q + X_j) \tag{1}$$

Where $\alpha(0,1)$ is randomly generated during the operation process.

3) If the number of synthesized samples reaches the required amount, the algorithm terminates. Otherwise, repeat from step 2.

As shown in Fig. 1, assuming that SMOTE randomly selects a sample X_q from minority class samples (where X_q is defined as the seed sample), then uses the k-nearest neighbor algorithm to find five nearest minority class samples. It then randomly selects one neighbor sample, X_j, from these five. The new synthetic sample, X_{new}, is

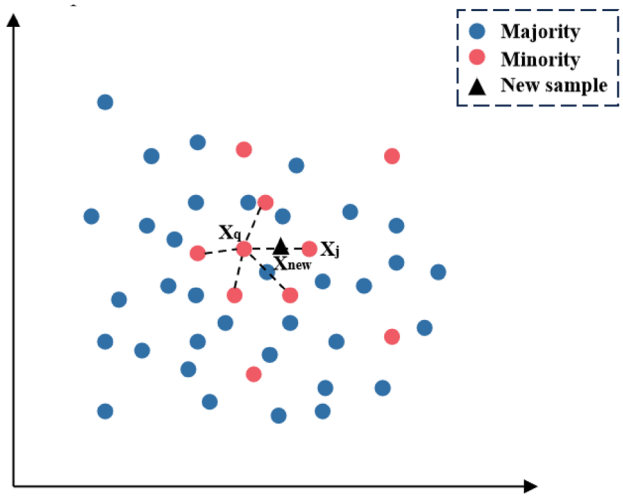

Fig. 1. Schematic diagram of the SMOTE algorithm's synthetic sample generation process.

generated through linear interpolation, as described in (1). The SMOTE algorithm generates new samples via linear interpolation along the line connecting two minority class samples.

In the SMOTE algorithm, the position of the newly generated sample is constrained to the space between the two original minority class samples and is influenced by their positions. As more new samples are generated, the density of minority class samples increases. Therefore, the SMOTE algorithm may lead to the distribution of oversampled minority class samples not conforming to the original distribution pattern of minority class samples.

The ISMOTE algorithm

The ISMOTE algorithm Idea

The SMOTE algorithm generates new minority class samples through linear interpolation between existing minority class samples. While this approach mitigates overfitting to some extent, it confines new samples to linear paths between existing samples, which may not accurately reflect the true distribution of the data. To address this limitation, ISMOTE extends the spatial scope for generating new samples by adaptively adjusting their positions based on local density and distribution characteristics. To expand the space for generating new samples, we change the conditions formula for generating new samples in the SMOTE algorithm. First, the Euclidean distance between the selected minority class sample and its k-nearest neighbor is calculated and multiplied by a random number between 0 and 1 to generate a random quantity. The base sample is generated by linear interpolation between the two samples. Then, when the base sample generation position is biased towards the k-nearest neighbor sample, the random quantity is subtracted to generate a new sample near the original sample. Similarly, when the base sample is closer to the original minority sample, the random quantity is added to generate a sample closer to the neighbor.

As shown in Fig. 2, (a) and (c) illustrate the positions of new samples generated by the SMOTE algorithm, and (b) and (d) show the positions of new samples generated by the ISMOTE algorithm. According to the SMOTE algorithm, a base sample X_{new1} is generated, which is located between the two selected minority class samples. Then, based on the distance of the base sample X_{new1} from the two original samples, the new sample X_{new} generation position is adjusted. If the generated base sample X_{new1} is far from the seed sample X_q, a random quantity is subtracted to position the new sample X_{new} around the seed sample X_q, as shown in Fig. 2(b). If the base sample X_{new1} is close to the seed sample X_q, a random quantity is added to position the new sample X_{new} around the selected neighboring sample X_j, as shown in Fig. 2(d). The new samples X_{new} generated by the algorithm are randomly distributed between two samples and around single samples, which increases the space for generating new samples and makes the distribution of new samples more consistent with the distribution pattern of the original samples.

The formula for generating new sample positions is revised. First, the Euclidean distance between the seed sample X_q and its randomly selected neighboring sample X_j is calculated, and this distance is multiplied by a random number between 0 and 1 to generate a random quantity. Secondly, a base sample X_{new1} is generated based on (2). Finally, this study proposes (3) to generate a new sample. According to the distance between X_{new1} and the seed sample X_q, a random quantity is added or subtracted from the position of X_{new1} to generate a new sample X_{new}. If X_{new1} is far away from X_q, the position of X_{new1} is changed by subtracting a random quantity to get the X_{new}, so that it is located around X_q. If X_{new1} is close to X_q, the position of X_{new1} is adjusted by adding a random quantity to place X_{new}, so that it is located around the neighboring sample X_j. The revised formula ensures that the generated samples are located around the two selected samples. Here, $\alpha\ (0,1)$ and $\beta \in (0,1)$ are randomly generated during each operation.

$$X_{new1} = X_q + \alpha \times (X_q - X_j) \tag{2}$$

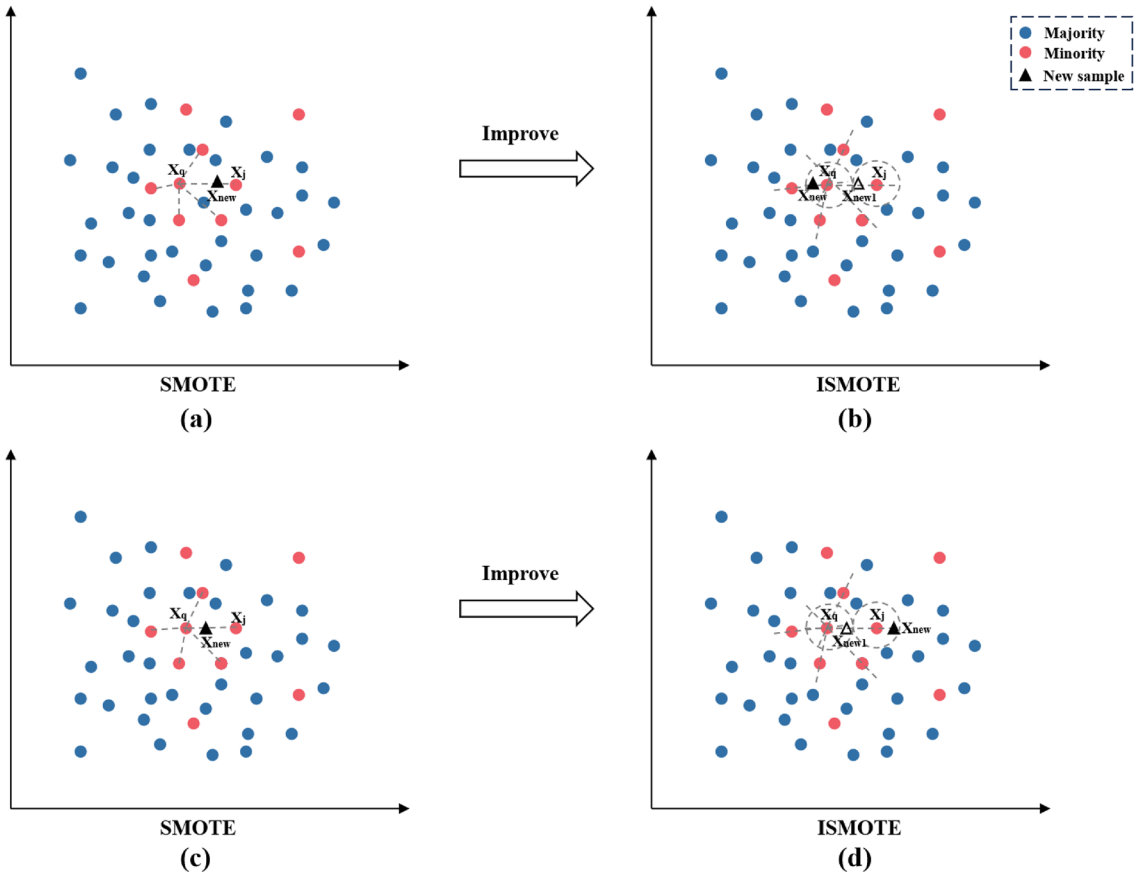

Fig. 2. Diagram of improvement idea for the ISMOTE algorithm.

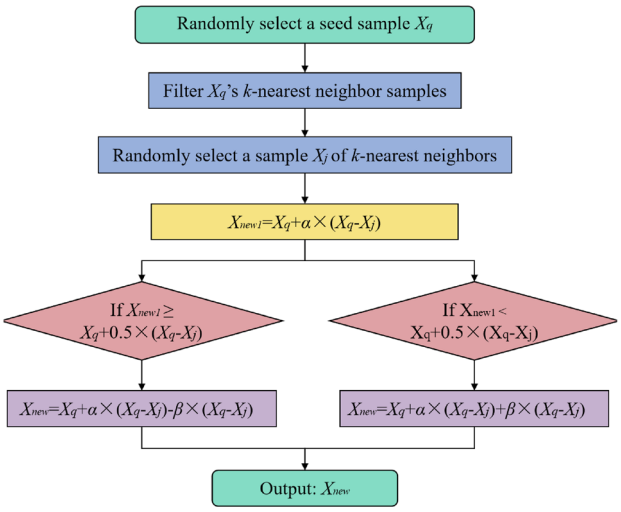

Fig. 3. The flowchart of the ISMOTE algorithm.

$$X_{new}=\begin{cases}X_q+\alpha\times(X_q-X_j)-\beta\times(X_q-X_j),if\ distance(X_{new1},X_q)\geqslant 0.5\times distance(X_q,X_j)\\ X_q+\alpha\times(X_q-X_j)+\beta\times(X_q-X_j),if\ distance(X_{new1},X_q)<0.5\times distance(X_q,X_j)\end{cases}\tag{3}$$

The flowchart, steps and pseudo-code of the ISMOTE algorithm
Based on the ISMOTE algorithm, the flowchart of the ISMOTE algorithm is shown in Fig. 3. According to the flowchart, the steps of the ISMOTE algorithm are as follows.

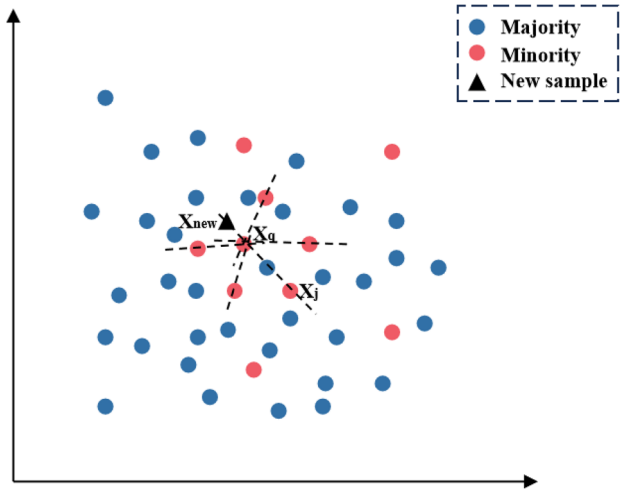

Fig. 4. Schematic Diagram of the ISMOTE Algorithm's Synthetic Sample Generation Process.

Input: A dataset with class imbalance W
Number of nearest neighbors k
Number of samples to be generated $n_{samples}$
Ouput: A new dataset with class balance W'
Steps:
1. Filter out the samples of minority class W_+
2. $\Psi=\emptyset$
3. for i ->1 to $n_{samples}$:
4. Randomly select a sample in W_+, denoted as X_q
5. Find the k nearest neighbors of X_q in W_+
6. Randomly select a nearest neighbor X_j
7. Generate a base sample X_{new1} of a minority class, according to $X_{new1} = X_q + \alpha \times (X_q - X_j)$ (α is a random number generated between (0,1))
8. if distance(X_{new1},X_q)≥0.5×distance(X_q,X_j):
9. $X_{new}=X_q+\alpha\times(X_q-X_j)-\beta\times(X_q-X_j)$ (β is a random number generated between (0, 1))
10. else:
11. $X_{new}=X_q+\alpha\times(X_q-X_j)+\beta\times(X_q-X_j)$
12. $\Psi=\Psi\cup X_{new}$
13. End
14. $W'=W_+\cup\Psi$

Algorithm 1. The ISMOTE algorithm.

1) Determine the number of samples of minority class to be synthesized, denoted as $n_{samples}$.
2) For each selected sample of minority class X_q (*1*N*- dimensional vectors, where N is the number of features), calculate its distance to all other minority class samples using the Euclidean distance formula. Select the k nearest neighbors of X_q, with k set to 5 by default.
3) From the k nearest neighbor samples, randomly select a sample X_j (a *1*N*-dimension vector, where N is the number of features of a sample) is selected. Generate a base sample X_{new1} randomly between X_q and X_j according to (2). Then, generate the new sample X_{new} by adjusting the position based on (3). If the number of new samples reaches the required amount, the algorithm terminates. Otherwise, repeat from step 2.

As shown in Fig. 4, in the ISMOTE algorithm, it is first assumed that a sample X_q (chosen as the seed sample) is randomly selected from the minority class samples. Using the k-nearest neighbors algorithm, the five nearest minority class samples nearest to X_q are identified. Then, randomly select one of these neighboring minority class samples, denoted as X_j. Generate a new synthetic minority class sample X_{new} using (2) and (3) proposed in this study. Compared to the SMOTE algorithm, the ISMOTE algorithm not only retains the position between the original two samples, but also expands the external surrounding space of single samples. This makes the overall distribution of the new samples more consistent with the original sample distribution pattern, improves data quality, and helps improve model performance.

The expansion of the generation space for new samples helps alleviate the problem of high sample density to some extent. The balanced dataset can be used for model training, enabling the model to better learn the

characteristics of minority class samples and improve classification performance. The ISMOTE algorithm can also be applied to multi-class classification tasks by adjusting the proportion of synthetic samples. The pseudocode is as follows.

Experiments and analysis

This section introduces the datasets and classifiers used in the research, as well as the metrics used to evaluate the experimental results. It details the experimental design, demonstrates and analyzes the visualization effects of datasets processed by seven oversampling algorithms, and compares and analyzes the results of three classifiers. The code for the ISMOTE algorithm and the dataset used in the experiment are available at https://github.com/Sunshine6828. 6/Improved-SMOTE-algorithm

Experimental datasets

Based on studies[20,48,52] and the requirements of this research, we selected thirteen classification datasets from three widely-used public databases for our experiments: the KEEL database, the UCI database[46], and the Kaggle competition platform. These databases serve as fundamental resources in data science and machine learning, providing extensive datasets for algorithm testing and validation across various domains and applications. The UCI database holds significant influence in machine learning research, while the KEEL database specializes in data mining and machine learning applications. Datasets from Kaggle competitions are particularly valued for their exceptional data quality and strong relevance to real-world business scenarios. The IR values of the thirteen selected datasets range from 1.25 to 29.17. Detailed information about the datasets is provided in Table 1. The features in the datasets are numerical. For non-numeric data, preprocessing is required to convert it into a numeric form before applying the ISMOTE algorithm. Techniques such as label encoding and one-hot encoding can convert non-numeric data into numeric form. Some multi-class datasets can be converted into binary classification datasets, resulting in multiple binary datasets with the same sample size but different minority class sizes and IR values. All datasets are processed as binary classification datasets. The method for calculating IR is given in (4).

$$IR = \frac{N_{maj}}{N_{\min}} \tag{4}$$

Here, N_{maj} represents the number of majority class samples, and N_{min} represents the number of minority class samples.

Evaluation metrics

A confusion matrix provides an intuitive evaluation of a classification model's performance by summarizing the classification results across different sample categories. In a confusion matrix, rows represent the true classes of samples, and columns represent the predicted classes. This paper focuses on binary classification, where the majority class is labeled as the negative class and the minority class as the positive class. In this matrix, a positive sample correctly predicted as positive is denoted as True Positive (TP); a negative sample incorrectly predicted as positive is False Positive (FP); a positive sample incorrectly predicted as negative is False Negative (FN); and a negative sample correctly predicted as negative is True Negative (TN). Table 2 shows the statistical results derived from the confusion matrix.

Common performance metrics such as precision, recall, F1 score, G-mean, and AUC can be derived from the confusion matrix. Precision measures the accuracy of positive predictions, while recall measures the model's ability to identify all actual positive samples. The F1 score, the harmonic mean of precision and recall (F1 is the F-measure when $\beta = 1$), offers a comprehensive evaluation of the classification performance for minority class samples. Specific calculation methods are provided in (5) to (9).

Dataset	ID	Samples	Features	Minority class	Minority samples	Majority class	Majority samples	IR
China_Telecom_Customer	D1	3463	19	class1	1534	class0	1929	1.25
glass0	D2	214	9	class1	70	class0	144	2.06
Iranian_Telecom_Customer	D3	3150	13	class1	495	class0	2655	5.36
ecoli2	D4	336	7	class1	52	class0	284	5.46
ecoli	D5	336	7	class1	35	class0	301	8.60
winequality-white-4_8	D6	4898	11	class4、8	238	reminder classes	4660	13.49
wilt	D7	4839	5	class1	261	class0	4578	17.54
Stroke_Prediction	D8	5110	11	class1	249	class0	4861	19.52
winequality-red-4_8	D9	1599	11	class4、8	71	reminder classes	1528	21.52
winequality-white-3_8	D10	4898	11	class3、8	195	reminder classes	4703	24.12
winequality-red-3_4	D11	1599	11	cass3、4	63	reminder classes	1436	24.38
winequality-white-3_4	D12	4898	11	class3、4	183	reminder classes	4715	25.77
winequality-red-4	D13	1599	11	class4	53	reminder classes	1546	29.17

Table 1. The datasets information.

Projected Values Actual Values	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Table 2. Binary confusion matrix.

$$\Pr ecision = \frac{TP}{TP + FP} \tag{5}$$

$$\mathrm{Re}\, call = Sensitivity = \frac{TP}{TP + FN} \tag{6}$$

$$Specificity = \frac{TN}{TN + FP} \tag{7}$$

$$F - measure = \frac{\left(1 + \beta^2\right) \times \mathrm{Re}\, call \times \Pr ecision}{\beta^2 \times \mathrm{Re}\, call \times \Pr ecision} \tag{8}$$

$$F1 = \frac{2 \times \mathrm{Re}\, call \times \Pr ecision}{\mathrm{Re}\, call + \Pr ecision} \tag{9}$$

Although the F1 score assesses individual class performance, it may not reflect the overall performance across both majority and minority classes and is sensitive to IR values. To provide a comprehensive assessment of the classifier's performance, G-mean is used as an additional evaluation metric. G-mean provides a balanced evaluation of the classification accuracy for both classes. The specific calculation method is shown in (10).

$$G - mean = \sqrt{Sensitivity \times Specificity} \tag{10}$$

AUC (Area Under the Curve) quantifies the area under the ROC (Receiver Operating Characteristic) curve, assessing the classifier's performance across both minority and majority classes. In cases of imbalanced sample distributions, the AUC provides a reliable measure of classifier performance. Thus, AUC is frequently employed to evaluate classifiers in imbalanced data scenarios. The specific calculation method is shown in (11).

$$AUC = \frac{1}{2}\left(1 + \frac{TP}{TP + FN} - \frac{FP}{FP + TN}\right) \tag{11}$$

Class imbalance can impact a classifier's predictive ability, yet high prediction accuracy can still be achieved. This happens because the trained classifier is often biased toward the majority class, which greatly outnumbers the minority class samples. Consequently, when the dataset is imbalanced, using accuracy as the performance evaluation metric is not comprehensive enough. The F1 score provides a balanced measure of the model's accuracy, with a higher F1 value indicating better performance. The G-mean evaluates classification performance across both majority and minority classes. AUC provides a comprehensive evaluation of the classifier's overall performance. Considering the factors discussed and based on previous research[47–49], F1, G-mean, and AUC are chosen as evaluation metrics.

Experimental design

Seven representative oversampling algorithms are selected as baselines to validate the effectiveness of the proposed ISMOTE algorithm. Due to the fact that the DeepSMOTE algorithm is a variant of SMOTE related to deep learning, this study mainly focuses on machine learning algorithms for experimentation, and the dataset used is relatively small. Therefore, the DeepSMOTE algorithm was not used as a comparative algorithm. In the G-SMOTE algorithm, synthetic samples are generated within a geometric region of the input space surrounding each selected minority class instance. In its default configuration (with parameter n_dim = 2), this region is defined as a hypersphere, while G-SMOTE also allows its deformation into a hyper-spheroid (with parameter n_dim = 3). Consequently, two variants of the method were experimentally evaluated: G-SMOTE (n_dim = 2) and G-SMOTE (n_dim = 3). These algorithms include ROS, SMOTE, ADASYN, Borderline-SMOTE (BSMOTE), K-means SMOTE (KSMOTE), GSMOTE($n = 2$), GSMOTE($n = 3$), and SYMPROD.

The classifiers are trained using training sets processed by these seven benchmark sampling algorithms and the ISMOTE algorithm. The performance of the classifiers indirectly reflects the performance of sampling algorithms. ROS is the simplest oversampling algorithm, balancing categories by randomly duplicating minority class samples. SMOTE is the most influential oversampling algorithm, serving as the foundation for various improvements. ADASYN generates more synthetic samples around hard-to-classify samples of the minority class, directing the classifier's focus toward these challenging samples. Borderline-SMOTE introduces a method to distinguish noisy samples, safe samples, and boundary samples which are only strengthened. K-means SMOTE interpolates in sparse areas of the minority class to avoid generating noisy data. G-SMOTE generalizes SMOTE's linear interpolation by constructing a geometric region (hyper-sphere or hyper-spheroid) around each minority instance. All these algorithms are typical extensions of the SMOTE algorithm, designed to

Classifier	Parameter
CART	criterion, max_depth, min_samples_leaf, min_samples_split, splitter
KNN	algorithm, metric, n_neighbors, p, weights
RF	max_depth, min_samples_leaf, min_samples_split, n_estimators

Table 3. Parameters used by the three classifiers.

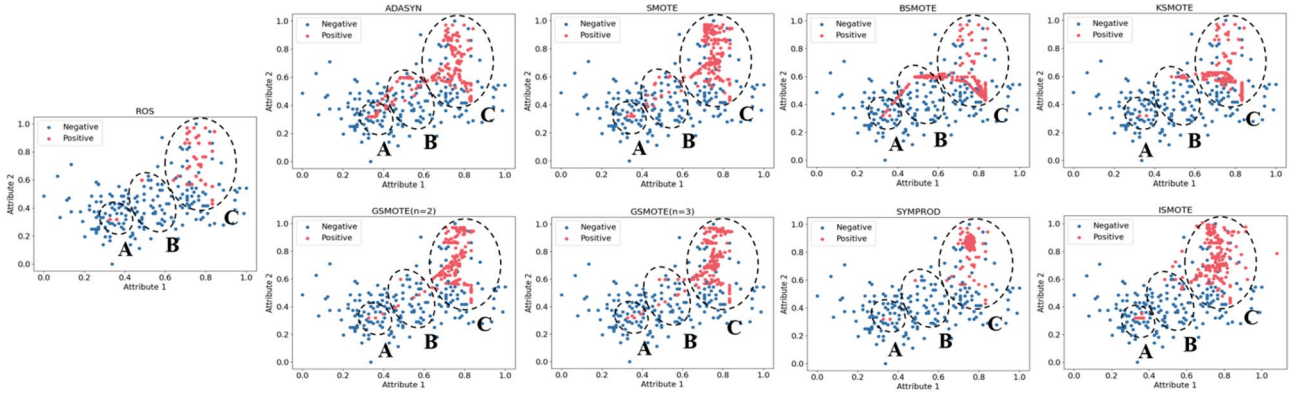

Fig. 5. Comparison of 2D sampling effects using different sampling methods.

select minority samples from different regions to generate new samples. The SYMPROD algorithm is the latest algorithm used separately in the smote-variants library, which uses probability distribution based minority class sample synthesis.

Previous research on sampling algorithms[16,20,49,51] has employed three classifiers in experiments, and this study also utilizes three classifiers. The algorithms with good classification performance include Classification and Regression Tree (CART)[50], K-Nearest Neighbors (KNN)[51,52], and Random Forest (RF)[53]. Therefore, these three algorithms are selected for the classification experiments. This study first visualize the data sampling results of the seven oversampling algorithms and analyze and compare the distribution of the generated data. Next, performing classification experiments on thirteen datasets processed by the seven oversampling algorithms using the three selected classifiers. Then comparing and analyzing the classification results using three evaluation metrics: F1, G-mean, and AUC.

In the experiments, the majority and minority class samples are first divided into training and test sets at a certain ratio, which are saved separately. The IR values of the training set and the test set are kept approximately the same level during the division. The ratio of the training set to the test set is 7:3. The divided training set is oversampled so that the ratio of positive to negative samples in the training set reaches approximately 1:1. The training set is used to train the classifier models, with synthetic samples participating only during the training phase. All the data in the test phase are real, with no synthetic samples involved, to obtain the most realistic experimental results. The parameters of the classifiers are optimized using grid search on the untreated training set, and the same parameters are applied to classifiers on the same dataset. The classifier parameters used are shown in Table 3. In order to avoid the impact of randomness on the experiment as much as possible and objectively compare the generalization performance improvement ability of different algorithms on the model, the training set experiment and the test set experiment both use ten-fold cross-validation to repeat the experiment ten times and take the average as the final result. The final result is output in the form of "mean ± standard deviation.

The environment for all experiments is as follows:

Hardware: 11th Gen Intel(R) Core(TM) i5-1155G7 (4 cores and 8 threads, clocked at 2.5 GHz), 16GB memory.

Software: The experimental platform uses a 64-bit Windows 11 operating system with PyCharm Community Edition 2023.3.4 (https://www.jetbrains.com/pycharm/down-load/) as the IDE. Python is adopted as the development language in this experiment.

Results and analysis

Comparative analysis of visualization results of oversampling algorithms

The data distribution can be effectively compared by visualizing the datasets processed by each oversampling algorithm. The ecoli2 dataset is selected as a demonstration dataset for distributing positive and negative samples. The imbalance rate is 5.46, and there is an overlap between positive and negative samples. To provide a clear and intuitive representation of the sample distribution, features with low overlap rates were selected for visualization. A 2D data distribution plot with two features (as shown in Fig. 5) and a 3D data distribution plot with three features (as shown in Fig. 6) are adopted for display and comparison. In these two figures, blue points

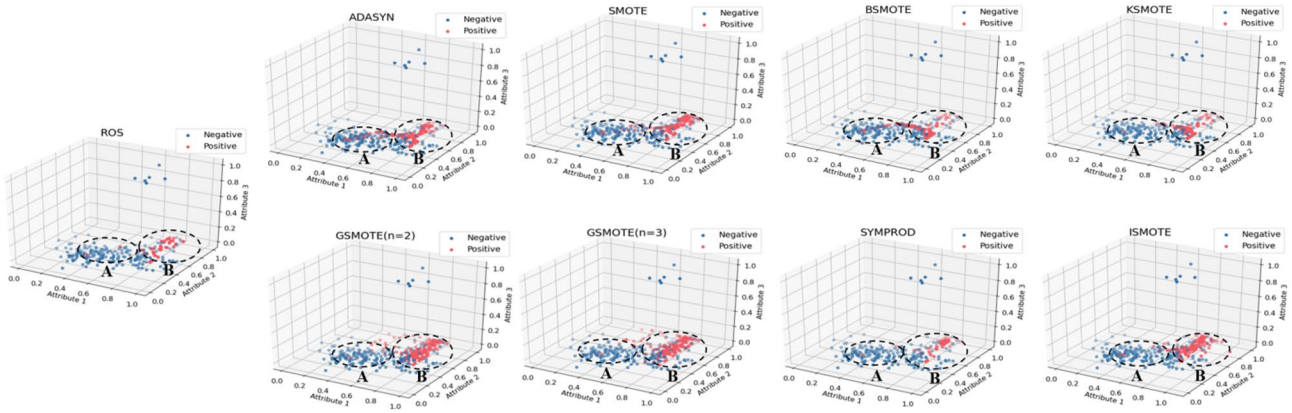

Fig. 6. Comparison of 3D sampling effects using different sampling methods.

represent majority class samples and red points represent minority class samples. Both Figs. 5 and 6 contain eight subplots, corresponding to ROS, SMOTE, ADASYN, BSMOTE, KSMOTE, GSMOTE($n=2$), GSMOTE($n=3$), STMPROD, and ISMOTE respectively.

Since ROS achieves data balance by randomly replicating minority class samples, the distribution of minority class samples in the ROS subgraphs of Figs. 5 and 6 is consistent with the original minority class sample distribution. As shown by the distribution of red sample points in the figures. The SMOTE subplot shows that the newly generated red sample points are distributed within the original distribution space of red sample points, with most of them clustered together. This indicates that the new samples generated by SMOTE through linear interpolation are limited to the original sample distribution space and relatively concentrated. Additionally, the algorithm generates a large number of new minority class samples in the blank area between the two clusters of red sample points, which may lead to the creation of excessively noisy. The ADASYN subgraph shows that the newly generated red sample points are more biased towards the boundary area. This algorithm generates a large number of new samples in areas where there are fewer minority class samples at the boundaries, which may result in the production of more noisy samples. The BSMOTE, GSMOTE($n=2$), GSMOTE($n=3$), and KSMOTE subgraphs show that these two algorithms generate a large number of new red sample points in areas with fewer minority class samples. These points mix with the blue sample points, increasing the complexity of the overlapping regions. Notably, in the 3D data distribution scatter plots, the G-SMOTE algorithm generated a significant number of minority class samples in sparse regions. This approach, while effective in expanding minority class representation, may potentially introduce new noisy samples. Additionally, these algorithms may result in inconsistencies between the sparsity distributions of the newly generated and original minority class samples. The SYMPROD subgraph shows that all generated samples are densely concentrated in one area, greatly increasing the density of the original samples.

The ISMOTE subgraphs in Figs. 5 and 6 show that the new red sample points generated by ISMOTE are similar to the distribution of the original red sample points, indicating that the data distribution after oversampling is largely consistent with the original data distribution. Comparing regions A, B, and C in Figs. 5 and 6, it is evident that ISMOTE generates more samples in areas with dense original minority class samples and fewer samples in areas with sparser minority class samples. This indicates that, compared to other algorithms, ISMOTE's red sample point distribution more closely resembles that of the original minority class samples. In comparison to the SMOTE algorithm, ISMOTE does not generate new samples excessively in areas devoid of minority class samples, such as region B in Fig. 5. Additionally, due to the expansion of the new sample generation space, ISMOTE ensures a more reasonable distribution density of newly generated minority class samples, alleviating the problem of excessive sample density to some extent. Since the ISMOTE algorithm generates new minority class samples around the original minority class samples, it may produce minority class noise samples outside the boundary. However, the number of these noise samples is small and has little impact on the overall quality of the dataset.

Comparison and analysis of classifier classification results

In some experimental datasets, KSMOTE fails to effectively execute its core K-Means clustering step due to the excessive dispersion of minority class samples. This is because KSMOTE relies on clustering the minority class samples and generating new samples near the cluster centers. However, when the minority class samples are highly scattered and lack a clear clustering structure, K-Means may fail to form valid clusters, preventing KSMOTE from performing the sample generation process. As a result, KSMOTE is unable to obtain valid evaluation metrics in these datasets, leading to empty results. All performance metrics are derived by training classifiers on training sets processed by the respective oversampling algorithms and then testing the resulting models on the test sets. The best values for each metric are highlighted in bold.

Tables 4 and 5, and 6 present the F1, G-mean, and AUC performance results for the CART classifier, respectively. In Table 4, for D5, the model trained on the ISMOTE-processed training set shows a 13.07% improvement in the F1 test performance indicator compared to models trained on sets processed by the seven

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.7770 ± 0.0007	0.7566 ± 0.0007	0.7795 ± 0.0006	0.7749 ± 0.0008	0.7825 ± 0.0007	0.7663 ± 0.0003	0.7747 ± 0.0006	0.7763 ± 0.0004	**0.7910 ± 0.0004**
D2	0.4097 ± 0.0390	0.5530 ± 0.0474	0.5883 ± 0.0423	0.6183 ± 0.0311	0.5953 ± 0.0427	0.5780 ± 0.0417	0.5653 ± 0.0545	0.6197 ± 0.0385	**0.6617 ± 0.0483**
D3	0.8061 ± 0.0014	0.7422 ± 0.0022	0.7060 ± 0.0019	0.7453 ± 0.0015	0.7886 ± 0.0037	0.7900 ± 0.0008	0.7815 ± 0.0015	**0.8199 ± 0.0023**	0.7940 ± 0.0021
D4	0.6823 ± 0.0342	0.6340 ± 0.0364	0.7160 ± 0.0478	0.3763 ± 0.0487	**0.8107 ± 0.0347**	0.6427 ± 0.0255	0.5583 ± 0.0343	0.7640 ± 0.0638	0.7783 ± 0.0325
D5	0.1473 ± 0.0296	0.3603 ± 0.0302	0.3780 ± 0.0385	0.4130 ± 0.0351	0.2913 ± 0.0331	0.3534 ± 0.0395	0.2827 ± 0.0434	0.2407 ± 0.0334	**0.5437 ± 0.0330**
D6	**0.3652 ± 0.0035**	0.1974 ± 0.0014	0.1978 ± 0.0017	0.2108 ± 0.0019	--	0.3606 ± 0.0040	0.2012 ± 0.0015	0.0999 ± 0.0047	0.2314 ± 0.0021
D7	0.9836 ± 0.0000	0.9850 ± 0.0000	0.9857 ± 0.0000	0.9883 ± 0.0000	--	0.9845 ± 0.0000	0.9846 ± 0.0000	**0.9887 ± 0.0000**	0.9781 ± 0.0000
D8	0.1645 ± 0.0044	0.1857 ± 0.0032	0.1849 ± 0.0027	0.1021 ± 0.0050	--	0.2130 ± 0.0070	0.1802 ± 0.0030	**0.2459 ± 0.0063**	0.1770 ± 0.0042
D9	0.1151 ± 0.0168	0.1223 ± 0.0039	0.1300 ± 0.0068	**0.1815 ± 0.0097**	--	0.1151 ± 0.0176	0.1339 ± 0.0077	0.0000 ± 0.0000	0.1578 ± 0.0056
D10	**0.3280 ± 0.0053**	0.1723 ± 0.0016	0.1715 ± 0.0026	0.2176 ± 0.0027	--	0.3169 ± 0.0054	0.1607 ± 0.0024	0.2089 ± 0.0139	0.1768 ± 0.0028
D11	0.1513 ± 0.0185	0.1546 ± 0.0101	0.1521 ± 0.0111	0.1133 ± 0.0086	--	0.0864 ± 0.0129	0.0889 ± 0.0074	0.1744 ± 0.0150	**0.1809 ± 0.0101**
D12	0.2019 ± 0.0108	0.1578 ± 0.0028	0.1949 ± 0.0016	0.1802 ± 0.0043	--	**0.2185 ± 0.0051**	0.2160 ± 0.0025	0.2450 ± 0.0098	0.1699 ± 0.0022
D13	0.0414 ± 0.0120	0.0970 ± 0.0092	0.0556 ± 0.0112	0.1345 ± 0.0181	--	0.0737 ± 0.0201	0.1522 ± 0.0094	0.0957 ± 0.0173	**0.1743 ± 0.0056**

Table 4. Comparison of F1 values for testing performance of CART models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.8020 ± 0.0004	0.7850 ± 0.0004	0.8049 ± 0.0004	0.7977 ± 0.0004	0.8074 ± 0.0006	0.7934 ± 0.0002	0.7989 ± 0.0005	0.7989 ± 0.0005	**0.8142 ± 0.0003**
D2	0.4367 ± 0.0515	0.5661 ± 0.0803	0.6176 ± 0.0517	0.6634 ± 0.0403	0.6280 ± 0.0467	0.6092 ± 0.0434	0.5983 ± 0.0565	0.6598 ± 0.0380	**0.7025 ± 0.0470**
D3	0.8856 ± 0.0006	0.8374 ± 0.0010	0.8176 ± 0.0008	0.8519 ± 0.0009	0.8673 ± 0.0022	0.8794 ± 0.0003	0.8715 ± 0.0009	0.8831 ± 0.0010	**0.8909 ± 0.0004**
D4	0.8048 ± 0.0301	0.7584 ± 0.0530	0.8015 ± 0.0689	0.4426 ± 0.0374	0.8325 ± 0.0391	0.7360 ± 0.0335	0.6653 ± 0.0514	0.8221 ± 0.0720	**0.8653 ± 0.0427**
D5	0.1850 ± 0.0131	0.4403 ± 0.0301	0.4546 ± 0.0270	0.4638 ± 0.0278	0.3559 ± 0.0260	0.4297 ± 0.0402	0.3481 ± 0.0334	0.2836 ± 0.0149	**0.6293 ± 0.0322**
D6	0.6356 ± 0.0042	0.5957 ± 0.0044	0.5919 ± 0.0045	0.5813 ± 0.0042	--	**0.6759 ± 0.0023**	0.6025 ± 0.0027	0.1999 ± 0.0227	0.6474 ± 0.0027
D7	0.8574 ± 0.0024	0.9073 ± 0.0014	0.9142 ± 0.0016	0.9171 ± 0.0008	--	0.9465 ± 0.0011	0.8856 ± 0.0031	0.9104 ± 0.0021	**0.9539 ± 0.0004**
D8	0.3196 ± 0.0171	0.4817 ± 0.0136	0.4763 ± 0.0170	0.3000 ± 0.0229	--	0.4853 ± 0.0102	0.4338 ± 0.0204	0.4789 ± 0.0090	**0.4994 ± 0.0134**
D9	0.1905 ± 0.0167	0.4962 ± 0.0182	0.4899 ± 0.0273	0.5283 ± 0.0388	--	0.2656 ± 0.0060	0.4002 ± 0.0299	0.0000 ± 0.0000	**0.5580 ± 0.0250**
D10	0.6367 ± 0.0135	0.6816 ± 0.0044	0.6698 ± 0.0076	0.6635 ± 0.0078	--	**0.6961 ± 0.0048**	0.6082 ± 0.0145	0.3661 ± 0.0282	0.6915 ± 0.0044
D11	0.2740 ± 0.0022	0.4142 ± 0.0341	0.4556 ± 0.0305	0.3177 ± 0.0167	--	0.2019 ± 0.0029	0.2929 ± 0.0243	0.2691 ± 0.0165	**0.5614 ± 0.0256**
D12	0.4011 ± 0.0195	0.6354 ± 0.0125	0.6787 ± 0.0050	0.5914 ± 0.0102	--	0.5015 ± 0.0238	**0.7026 ± 0.0103**	0.3447 ± 0.0204	0.6948 ± 0.0097
D13	0.0745 ± 0.0118	0.3274 ± 0.0418	0.2116 ± 0.0149	0.2926 ± 0.0298	--	0.1449 ± 0.0141	0.4043 ± 0.0226	0.1426 ± 0.0258	**0.6931 ± 0.0320**

Table 5. Comparison of G-mean values for testing performance of CART models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.8029 ± 0.0001	0.7864 ± 0.0002	0.8060 ± 0.0000	0.7986 ± 0.0001	0.8086 ± 0.0001	0.7948 ± 0.0001	0.7997 ± 0.0000	0.7997 ± 0.0001	**0.8149 ± 0.0001**
D2	0.6167 ± 0.0239	0.6750 ± 0.0293	0.7508 ± 0.0275	0.7408 ± 0.0225	0.7558 ± 0.0175	0.7192 ± 0.0167	0.7167 ± 0.0194	0.7583 ± 0.0158	**0.7800 ± 0.0187**
D3	0.8899 ± 0.0000	0.8461 ± 0.0000	0.8277 ± 0.0000	0.8584 ± 0.0000	0.8738 ± 0.0000	0.8834 ± 0.0000	0.8767 ± 0.0000	0.8882 ± 0.0000	**0.8936 ± 0.0000**
D4	0.8594 ± 0.0109	0.8258 ± 0.0209	0.8685 ± 0.0264	0.6702 ± 0.0187	0.8941 ± 0.0158	0.8277 ± 0.0127	0.7807 ± 0.0177	0.8826 ± 0.0251	**0.9032 ± 0.0159**
D5	0.5728 ± 0.0100	0.6861 ± 0.0158	0.6967 ± 0.0150	0.7103 ± 0.0160	0.6356 ± 0.0150	0.6922 ± 0.0187	0.6331 ± 0.0175	0.6117 ± 0.0100	**0.7897 ± 0.0115**
D6	0.6868 ± 0.0007	0.6141 ± 0.0013	0.6136 ± 0.0015	0.6171 ± 0.0010	--	**0.7081 ± 0.0012**	0.6182 ± 0.0006	0.5239 ± 0.0003	0.6575 ± 0.0010
D7	0.8690 ± 0.0008	0.9123 ± 0.0008	0.9191 ± 0.0010	0.9217 ± 0.0011	--	0.9482 ± 0.0009	0.8937 ± 0.0010	0.9161 ± 0.0013	**0.9548 ± 0.0004**
D8	0.5581 ± 0.0000	0.5942 ± 0.0000	0.5907 ± 0.0000	0.5282 ± 0.0000	--	0.6059 ± 0.0000	0.5802 ± 0.0000	**0.6113 ± 0.0000**	0.5957 ± 0.0000
D9	0.5419 ± 0.0053	0.5937 ± 0.0042	0.5952 ± 0.0062	**0.6558 ± 0.0077**	--	0.5506 ± 0.0038	0.5832 ± 0.0065	0.5000 ± 0.0000	0.6443 ± 0.0084
D10	0.7010 ± 0.0019	0.6953 ± 0.0019	0.6894 ± 0.0028	0.6962 ± 0.0026	--	**0.7339 ± 0.0030**	0.6519 ± 0.0027	0.5840 ± 0.0019	0.7026 ± 0.0022
D11	0.5674 ± 0.0000	0.6076 ± 0.0000	0.6152 ± 0.0000	0.5620 ± 0.0000	--	0.5283 ± 0.0000	0.5326 ± 0.0000	0.5826 ± 0.0000	**0.6598 ± 0.0000**
D12	0.5964 ± 0.0027	0.6638 ± 0.0022	0.7004 ± 0.0023	0.6465 ± 0.0023	--	0.6352 ± 0.0041	**0.7258 ± 0.0026**	0.5863 ± 0.0025	0.7083 ± 0.0027
D13	0.5031 ± 0.0100	0.5602 ± 0.0236	0.4980 ± 0.0150	0.5948 ± 0.0195	--	0.5198 ± 0.0115	0.6275 ± 0.0123	0.5503 ± 0.0123	**0.7497 ± 0.0114**

Table 6. Comparison of AUC values for testing performance of CART models trained on training sets processed by various sampling algorithms.

benchmark algorithms. Similarly, for D2, the F1 test performance indicator of the model trained on the training set processed by the ISMOTE algorithm is enhanced by 4.20% compared to the models trained on the training sets processed by the five baseline oversampling algorithms. For D13, the model trained on the training set processed by the ISMOTE algorithm showed an 2.21% improvement in the F1 test performance indicator. For D1 and D11, models trained on the training sets processed by the ISMOTE algorithm showed varying degrees of performance improvement in testing compared to the models trained on the training sets processed by the seven benchmark oversampling algorithms, although the extent of improvement is smaller. For datasets D3, D7, and, D8, the SYMPROD algorithm demonstrates notably improved performance. For other datasets, the difference in the F1 of the test performance indicator for the model trained with the ISMOTE-processed training set is within an acceptable range compared to models trained on other algorithms.

Table 5 demonstrates that models trained on ISMOTE-processed datasets achieve significantly higher G-mean test values compared to those using the seven baseline algorithms, with improvements of 28.88% for D13, 16.55% for D5, 11.58% for D11, and 3.91% for D2. For D1, D3, D4, D7, D8, and D9, models trained on the training sets processed by the ISMOTE algorithm showed varying degrees of performance improvement in testing compared to the models trained on the training sets processed by the seven benchmark oversampling algorithms. For datasets D6, D10, and D12, the G-SMOTE algorithm demonstrates notably improved performance.

In Table 6, across eight datasets, the AUC test performance indicators for models trained on ISMOTE-processed training sets show varying degrees of improvement compared to models trained on sets processed by the seven baseline algorithms. For D13, the AUC test performance indicator of the model trained on the ISMOTE-processed training set increases by 12.22%. For D5, the AUC test performance indicator of the model trained on the ISMOTE-processed training set increases by 7.94%. For D11, the model trained on the training set processed by the ISMOTE algorithm showed an 4.46% improvement in the AUC test performance indicator. For D1, D2, D4, and D7, models trained on the training sets processed by the ISMOTE algorithm showed varying degrees of performance improvement in testing compared to the models trained on the training sets processed by the seven benchmark oversampling algorithms. The models trained on ISMOTE-processed training sets do not achieve the best performance on D3, D6, D8, D9, D10, and D12.

The results of the F1, G-mean, and AUC performance indicators on the KNN classifier are shown in Tables 7 and 8, and 9, respectively. In Table 7, compared to the models trained on training sets processed by the seven baseline algorithms, the F1 value of the test performance indicator for the model trained with the training set processed by the ISMOTE algorithm improves by 0.43% on dataset D1. For dataset D9, the F1 test performance indicator improves by 0.29%. For D2, D6, D10, and D12, the F1 test performance indicator for models trained on ROS-processed training sets shows improvements compared to models trained on sets processed by the seven baseline algorithms. Although the model trained with the training set processed by the ISMOTE algorithm does not achieve the best test performance on datasets D5 and D8, the difference is within an acceptable range. For datasets D3, D4, D7, D9, D11, and D13, other algorithms demonstrate notably improved performance.

In Table 8, compared to models trained on training sets processed by the seven benchmark algorithms, the G-mean of the test performance indicator for the model trained with the ISMOTE-processed training set improves by 2.71% on D10. For D9, the G-mean of the test performance indicator for the model trained with the ISMOTE-processed training set improves by 2.25%. For datasets D1, D4, D5, D6, D7, and D8, the ISMOTE algorithm demonstrates notably improved performance. However, for D12 and D13, the G-mean of the test performance indicator for the model trained using the ISMOTE-processed training set shows a larger gap compared to models trained using other algorithms. For D2, D3, and D11, the difference in the G-mean of the test performance indicator for the model trained with the ISMOTE-processed training set is within an acceptable range compared to models trained on other algorithms.

Table 9 demonstrates that models trained on ISMOTE-processed datasets achieve significantly higher AUC test values compared to those using the seven baseline algorithms, with improvements of 1.79% for D9, 1.72%

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.7050 ± 0.0008	0.7074 ± 0.0003	0.7044 ± 0.0007	0.7089 ± 0.0006	0.7027 ± 0.0006	0.7070 ± 0.0007	0.7022 ± 0.0006	0.7047 ± 0.0009	**0.7132 ± 0.0005**
D2	**0.6937 ± 0.0277**	0.6790 ± 0.0334	0.6340 ± 0.0227	0.6433 ± 0.0379	0.6203 ± 0.0393	0.6457 ± 0.0391	0.6918 ± 0.0398	0.6580 ± 0.0454	0.6750 ± 0.0451
D3	0.8142 ± 0.0013	0.8148 ± 0.0020	0.8095 ± 0.0007	0.8098 ± 0.0006	0.8143 ± 0.0017	0.8126 ± 0.0014	**0.8236 ± 0.0016**	0.8197 ± 0.0019	0.8122 ± 0.0015
D4	0.7573 ± 0.0400	0.7339 ± 0.0263	0.7233 ± 0.0215	0.8033 ± 0.0480	0.8287 ± 0.0319	0.7312 ± 0.0298	0.6960 ± 0.0242	**0.8577 ± 0.0490**	0.7644 ± 0.0312
D5	0.5810 ± 0.0287	0.6234 ± 0.0402	0.6077 ± 0.0367	**0.6770 ± 0.0280**	0.6070 ± 0.0303	0.5790 ± 0.0375	0.5760 ± 0.0312	0.5784 ± 0.0241	0.6733 ± 0.0397
D6	**0.3874 ± 0.0031**	0.3352 ± 0.0025	0.3375 ± 0.0026	0.3406 ± 0.0038	--	0.3130 ± 0.0025	0.3219 ± 0.0022	0.3703 ± 0.0049	0.3436 ± 0.0035
D7	0.9652 ± 0.0000	0.9524 ± 0.0001	0.9469 ± 0.0001	0.9561 ± 0.0000	--	0.9603 ± 0.0000	0.9607 ± 0.0000	**0.9786 ± 0.0000**	0.9428 ± 0.0001
D8	0.1173 ± 0.0043	0.1845 ± 0.0034	0.1811 ± 0.0032	**0.1921 ± 0.0036**	--	0.1786 ± 0.0044	0.1733 ± 0.0036	0.0604 ± 0.0022	0.1908 ± 0.0030
D9	0.1489 ± 0.0182	0.1424 ± 0.0077	0.1392 ± 0.0099	0.1251 ± 0.0058	--	0.1449 ± 0.0119	0.1737 ± 0.0200	0.0513 ± 0.0086	**0.1766 ± 0.0134**
D10	**0.3708 ± 0.0070**	0.2805 ± 0.0050	0.2672 ± 0.0045	0.3318 ± 0.0049	--	0.3078 ± 0.0053	0.3139 ± 0.0071	0.3584 ± 0.0088	0.2987 ± 0.0052
D11	0.2153 ± 0.0209	0.1631 ± 0.0124	0.1634 ± 0.0105	0.1857 ± 0.0137	--	**0.2363 ± 0.0211**	0.2158 ± 0.0181	0.2225 ± 0.0229	0.1908 ± 0.0126
D12	**0.3247 ± 0.0194**	0.2656 ± 0.0051	0.2556 ± 0.0045	0.2794 ± 0.0050	--	0.2806 ± 0.0054	0.2926 ± 0.0078	0.2388 ± 0.0110	0.2564 ± 0.0068
D13	0.1402 ± 0.0180	0.1129 ± 0.0143	0.0872 ± 0.0051	**0.1652 ± 0.0179**	--	0.1356 ± 0.0129	0.1187 ± 0.0191	0.0683 ± 0.0117	0.1192 ± 0.0130

Table 7. Comparison of F1 values for testing performance of KNN models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.7299 ± 0.0005	0.7310 ± 0.0005	0.7265 ± 0.0004	0.7256 ± 0.0004	0.7319 ± 0.0005	0.7298 ± 0.0007	0.7274 ± 0.0005	0.7372 ± 0.0009	**0.7381 ± 0.0005**
D2	0.7071 ± 0.0319	0.7002 ± 0.0649	0.6753 ± 0.0523	0.6790 ± 0.0487	0.6502 ± 0.0433	0.6897 ± 0.0482	**0.7162 ± 0.0590**	0.6930 ± 0.0587	0.7084 ± 0.0578
D3	0.8871 ± 0.0008	0.9038 ± 0.0004	**0.9077 ± 0.0005**	0.9055 ± 0.0004	0.8872 ± 0.0007	0.8953 ± 0.0007	0.9033 ± 0.0004	0.8884 ± 0.0009	0.8980 ± 0.0005
D4	0.8624 ± 0.0412	0.8848 ± 0.0300	0.8746 ± 0.0318	0.8764 ± 0.0474	0.8627 ± 0.0329	0.8763 ± 0.0338	0.8338 ± 0.0256	0.8863 ± 0.0552	**0.9024 ± 0.0315**
D5	0.7533 ± 0.0061	0.7930 ± 0.0324	0.7863 ± 0.0324	0.8599 ± 0.0264	0.7664 ± 0.0259	0.7654 ± 0.0323	0.7696 ± 0.0296	0.6851 ± 0.0250	**0.8646 ± 0.0322**
D6	0.6589 ± 0.0033	0.6866 ± 0.0029	0.6984 ± 0.0025	0.6652 ± 0.0031	--	0.6588 ± 0.0018	0.6604 ± 0.0027	0.5454 ± 0.0102	**0.6988 ± 0.0038**
D7	0.6794 ± 0.0111	0.7653 ± 0.0029	0.7577 ± 0.0056	0.7411 ± 0.0058	--	0.7729 ± 0.0013	0.7620 ± 0.0031	0.5287 ± 0.0148	**0.7867 ± 0.0025**
D8	0.3055 ± 0.0146	0.5058 ± 0.0108	0.4979 ± 0.0118	0.5019 ± 0.0108	--	0.4812 ± 0.0180	0.4749 ± 0.0117	0.1016 ± 0.0082	**0.5106 ± 0.0120**
D9	0.2423 ± 0.0278	0.4124 ± 0.0341	0.3892 ± 0.0266	0.2961 ± 0.0309	--	0.3705 ± 0.0246	0.4005 ± 0.0291	0.0663 ± 0.0058	**0.4349 ± 0.0246**
D10	0.6784 ± 0.0105	0.7037 ± 0.0091	0.6909 ± 0.0108	0.7042 ± 0.0111	--	0.7132 ± 0.0079	0.7123 ± 0.0116	0.5864 ± 0.0163	**0.7394 ± 0.0146**
D11	0.3302 ± 0.0198	0.4355 ± 0.0186	0.4206 ± 0.0270	0.3791 ± 0.0246	--	0.4894 ± 0.0255	**0.4895 ± 0.0353**	0.3309 ± 0.0196	0.4839 ± 0.0241
D12	0.5830 ± 0.0331	**0.6629 ± 0.0045**	0.6557 ± 0.0214	0.6311 ± 0.0135	--	0.6460 ± 0.0146	0.6469 ± 0.0129	0.3764 ± 0.0315	0.6284 ± 0.0179
D13	0.2324 ± 0.0175	0.3133 ± 0.0443	0.2844 ± 0.0199	**0.3400 ± 0.0411**	--	0.2929 ± 0.0268	0.2877 ± 0.0377	0.0736 ± 0.0088	0.3176 ± 0.0260

Table 8. Comparison of G-mean values for testing performance of KNN models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.7316 ± 0.0001	0.7326 ± 0.0001	0.7286 ± 0.0001	0.7290 ± 0.0001	0.7332 ± 0.0001	0.7319 ± 0.0001	0.7287 ± 0.0001	0.7390 ± 0.0001	**0.7393 ± 0.0001**
D2	0.7858 ± 0.0179	0.7717 ± 0.0215	0.7433 ± 0.0235	0.7533 ± 0.0239	0.7483 ± 0.0220	0.7567 ± 0.0210	0.7783 ± 0.0176	0.7750 ± 0.0204	**0.7892 ± 0.0187**
D3	0.8918 ± 0.0000	0.9061 ± 0.0000	**0.9096 ± 0.0000**	0.9075 ± 0.0000	0.8918 ± 0.0000	0.8986 ± 0.0000	0.9059 ± 0.0000	0.8930 ± 0.0000	0.9007 ± 0.0000
D4	0.9004 ± 0.0165	0.9062 ± 0.0121	0.8983 ± 0.0125	0.9115 ± 0.0162	0.9157 ± 0.0136	0.9080 ± 0.0119	0.8744 ± 0.0118	**0.9269 ± 0.0196**	0.9207 ± 0.0115
D5	0.8444 ± 0.0000	0.8594 ± 0.0122	0.8569 ± 0.0125	0.8994 ± 0.0100	0.8494 ± 0.0100	0.8464 ± 0.0115	0.8433 ± 0.0122	0.8161 ± 0.0100	**0.9019 ± 0.0115**
D6	0.7013 ± 0.0007	0.7107 ± 0.0012	0.7177 ± 0.0014	0.6994 ± 0.0010	--	0.6884 ± 0.0011	0.6922 ± 0.0010	0.6477 ± 0.0011	**0.7200 ± 0.0007**
D7	0.7295 ± 0.0016	0.7843 ± 0.0013	0.7792 ± 0.0015	0.7691 ± 0.0016	--	0.7920 ± 0.0012	0.7859 ± 0.0013	0.6520 ± 0.0013	**0.7999 ± 0.0011**
D8	0.5372 ± 0.0000	0.5984 ± 0.0000	0.5971 ± 0.0000	0.6005 ± 0.0000	--	0.5908 ± 0.0000	0.5862 ± 0.0000	0.5157 ± 0.0000	**0.6061 ± 0.0000**
D9	0.5685 ± 0.0069	0.5910 ± 0.0050	0.5868 ± 0.0053	0.5619 ± 0.0045	--	0.5886 ± 0.0039	0.6087 ± 0.0045	0.5171 ± 0.0038	**0.6266 ± 0.0069**
D10	0.7287 ± 0.0021	0.7379 ± 0.0013	0.7260 ± 0.0027	0.7437 ± 0.0033	--	0.7451 ± 0.0023	0.7450 ± 0.0022	0.6768 ± 0.0017	**0.7623 ± 0.0029**
D11	0.6054 ± 0.0000	0.6098 ± 0.0000	0.6109 ± 0.0000	0.6076 ± 0.0000	--	**0.6565 ± 0.0000**	0.6500 ± 0.0000	0.6087 ± 0.0000	0.6424 ± 0.0000
D12	0.6769 ± 0.0031	0.7054 ± 0.0032	0.7019 ± 0.0011	0.6940 ± 0.0027	--	0.7035 ± 0.0034	**0.7065 ± 0.0031**	0.5912 ± 0.0023	0.6902 ± 0.0042
D13	0.5806 ± 0.0112	0.5746 ± 0.0195	0.5593 ± 0.0158	**0.6112 ± 0.0158**	--	0.5794 ± 0.0175	0.5801 ± 0.0208	0.5264 ± 0.0075	0.5814 ± 0.0187

Table 9. Comparison of AUC values for testing performance of KNN models trained on training sets processed by various sampling algorithms.

for D10. For datasets D1, D2, D5, D6, D7, and D8, the ISMOTE algorithm demonstrates notably improved performance. For D3, D4, D11, and D12, the difference in the AUC of the test performance indicator for the model trained with the ISMOTE-processed training set is within an acceptable range compared to models trained on other algorithms. However, for D13, the AUC of the test performance indicator for the model trained using the ISMOTE-processed training set shows a larger gap compared to models trained using other algorithms.

The F1, G-mean, and AUC performance indicators of the RF classifier are shown in Tables 10 and 11, and 12, respectively. In Table 10, compared to models trained on training sets processed by the seven baseline algorithms, the F1 value of the test performance indicator for models trained with the ISMOTE-processed training set shows varying degrees of improvement across five datasets. For D3, D5, D6, D7, D9 and D13 the difference in the AUC of the test performance indicator for the model trained with the ISMOTE-processed training set is within an acceptable range compared to models trained on other algorithms. However, for D2 and D10, the F1 of the test performance indicator for the model trained using the ISMOTE-processed training set shows a larger gap compared to models trained using other algorithms.

Table 11 demonstrates that models trained on ISMOTE-processed datasets achieve significantly higher G-mean test values compared to those using the seven baseline algorithms, with improvements of 7.48% for D8, 16.55% for D11, and 11.10% for D12. For D1, D3, D4, D5, D6, D7, D9, D10, and D13, the G-mean value of the model test performance indicator trained with the ISMOTE processed training set shows significant performance improvements. However, for D2, the G-mean of the test performance indicator for the model trained using the ISMOTE-processed training set shows a larger gap compared to models trained using other algorithms.

In Table 12, compared to models trained on training sets processed by the seven baseline algorithms, the AUC value of the test performance indicator for models trained with the ISMOTE-processed training set shows varying degrees of improvement across twelve datasets. For D11, the AUC value of the test performance indicator for the model trained with the ISMOTE-processed training set increases by 11.10%. For D1, D3, D4,

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.8376 ± 0.0003	0.8287 ± 0.0004	0.8280 ± 0.0005	0.8276 ± 0.0004	0.8367 ± 0.0002	0.8293 ± 0.0003	0.8297 ± 0.0006	0.8321 ± 0.0004	**0.8381 ± 0.0003**
D2	0.6670 ± 0.0481	0.6847 ± 0.0319	0.6400 ± 0.0333	0.7333 ± 0.0000	0.7177 ± 0.0326	0.7000 ± 0.0241	0.6727 ± 0.0411	**0.7837 ± 0.0351**	0.6924 ± 0.0410
D3	0.8524 ± 0.0012	**0.8576 ± 0.0012**	0.8134 ± 0.0018	0.8222 ± 0.0016	0.8125 ± 0.0011	0.8295 ± 0.0015	0.8546 ± 0.0006	0.8474 ± 0.0012	0.8402 ± 0.0014
D4	0.8627 ± 0.0417	0.7943 ± 0.0427	0.6970 ± 0.0517	0.7053 ± 0.0453	0.7270 ± 0.0372	0.7407 ± 0.0358	0.7963 ± 0.0479	0.8017 ± 0.0647	**0.9220 ± 0.0275**
D5	0.3600 ± 0.0264	0.3870 ± 0.0230	0.3613 ± 0.0489	0.4113 ± 0.0401	**0.5137 ± 0.0341**	0.3417 ± 0.0455	0.4327 ± 0.0358	0.3467 ± 0.0364	0.5053 ± 0.0490
D6	0.4088 ± 0.0046	0.4349 ± 0.0040	**0.4409 ± 0.0053**	0.4325 ± 0.0052	--	0.4400 ± 0.0065	0.4350 ± 0.0040	0.3359 ± 0.0062	0.4359 ± 0.0057
D7	**0.9917 ± 0.0000**	0.9898 ± 0.0000	0.9887 ± 0.0000	0.9880 ± 0.0000	--	0.9894 ± 0.0000	0.9902 ± 0.0000	0.9906 ± 0.0000	0.9882 ± 0.0000
D8	0.2696 ± 0.0044	0.2692 ± 0.0034	0.2338 ± 0.0042	0.2737 ± 0.0038	--	0.2432 ± 0.0037	0.2168 ± 0.0040	0.0436 ± 0.0018	**0.3026 ± 0.0037**
D9	0.0513 ± 0.0086	0.1301 ± 0.0201	0.1281 ± 0.0153	0.1460 ± 0.0136	--	**0.1465 ± 0.0171**	0.1403 ± 0.0090	0.1130 ± 0.0133	0.1434 ± 0.0143
D10	0.4739 ± 0.0068	0.3540 ± 0.0088	0.3892 ± 0.0093	0.3946 ± 0.0078	--	0.3857 ± 0.0144	0.4094 ± 0.0123	**0.5273 ± 0.0107**	0.3643 ± 0.0098
D11	0.0000 ± 0.0000	0.1470 ± 0.0123	0.1300 ± 0.0191	0.0607 ± 0.0095	--	0.1673 ± 0.0174	0.1563 ± 0.0180	0.1183 ± 0.0090	**0.1702 ± 0.0075**
D12	0.1729 ± 0.0095	0.3735 ± 0.0068	0.3930 ± 0.0077	0.2865 ± 0.0075	--	0.2712 ± 0.0126	0.2543 ± 0.0108	0.2096 ± 0.0118	**0.4018 ± 0.0063**
D13	0.0000 ± 0.0000	0.1047 ± 0.0173	0.1019 ± 0.0162	0.0607 ± 0.0095	--	0.1070 ± 0.0180	**0.1203 ± 0.0283**	0.0000 ± 0.0000	0.1120 ± 0.0108

Table 10. Comparison of F1 values for testing performance of RF models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.8556 ± 0.0002	0.8478 ± 0.0002	0.8465 ± 0.0004	0.8463 ± 0.0003	0.8551 ± 0.0005	0.8488 ± 0.0002	0.8486 ± 0.0003	0.8513 ± 0.0002	**0.8565 ± 0.0001**
D2	0.6902 ± 0.0597	0.7129 ± 0.0428	0.6760 ± 0.0526	0.7695 ± 0.0000	0.7541 ± 0.0325	0.7439 ± 0.0391	0.7100 ± 0.0420	**0.8093 ± 0.0276**	0.7289 ± 0.0548
D3	0.9203 ± 0.0004	0.9218 ± 0.0003	0.9061 ± 0.0006	0.8973 ± 0.0008	0.8670 ± 0.0006	0.9126 ± 0.0005	0.9208 ± 0.0005	0.8886 ± 0.0007	**0.9231 ± 0.0005**
D4	0.8892 ± 0.0446	0.8400 ± 0.0500	0.7960 ± 0.0681	0.7703 ± 0.0565	0.7653 ± 0.0346	0.7969 ± 0.0427	0.8216 ± 0.0509	0.8320 ± 0.0715	**0.9379 ± 0.0337**
D5	0.4429 ± 0.0314	0.4463 ± 0.0307	0.4511 ± 0.0271	0.4628 ± 0.0303	0.6212 ± 0.0299	0.4267 ± 0.0405	0.5284 ± 0.0330	0.3680 ± 0.0277	**0.6239 ± 0.0364**
D6	0.5196 ± 0.0049	0.6359 ± 0.0031	0.6226 ± 0.0040	0.6076 ± 0.0046	--	0.6016 ± 0.0048	0.5928 ± 0.0091	0.4402 ± 0.0157	**0.6776 ± 0.0039**
D7	0.8787 ± 0.0023	0.9260 ± 0.0012	0.9177 ± 0.0013	0.8887 ± 0.0029	--	0.9042 ± 0.0018	0.9114 ± 0.0025	0.8385 ± 0.0057	**0.9511 ± 0.0007**
D8	0.5903 ± 0.0075	0.6307 ± 0.0072	0.5789 ± 0.0131	0.5972 ± 0.0075	--	0.5674 ± 0.0103	0.5240 ± 0.0115	0.0666 ± 0.0083	**0.7055 ± 0.0118**
D9	0.0638 ± 0.0063	0.1988 ± 0.0124	0.1934 ± 0.0185	0.1967 ± 0.0107	--	0.2015 ± 0.0118	0.1994 ± 0.0168	0.1323 ± 0.0176	**0.2469 ± 0.0209**
D10	0.5681 ± 0.0112	0.5990 ± 0.0137	0.6406 ± 0.0160	0.6025 ± 0.0204	--	0.5673 ± 0.0262	0.5877 ± 0.0199	0.6046 ± 0.0158	**0.6769 ± 0.0153**
D11	0.0000 ± 0.0000	0.2050 ± 0.0117	0.2039 ± 0.0117	0.0704 ± 0.0005	--	0.2104 ± 0.0010	0.2058 ± 0.0118	0.1407 ± 0.0004	**0.3214 ± 0.0194**
D12	0.2496 ± 0.0145	0.5833 ± 0.0146	0.6045 ± 0.0213	0.4793 ± 0.0163	--	0.4186 ± 0.0202	0.3907 ± 0.0215	0.2906 ± 0.0296	**0.6676 ± 0.0072**
D13	0.0000 ± 0.0000	0.1387 ± 0.0159	0.1451 ± 0.0112	0.0760 ± 0.0115	--	0.1511 ± 0.0194	0.1547 ± 0.0198	0.0000 ± 0.0000	**0.2063 ± 0.0228**

Table 11. Comparison of G-mean values for testing performance of RF models trained on training sets processed by various sampling algorithms.

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE($n=2$)	GSMOTE($n=3$)	SYMPROD	ISMOTE
D1	0.8563 ± 0.0001	0.8485 ± 0.0001	0.8473 ± 0.0001	0.8471 ± 0.0001	0.8558 ± 0.0001	0.8494 ± 0.0001	0.8493 ± 0.0001	0.8519 ± 0.0001	**0.8569 ± 0.0000**
D2	0.7800 ± 0.0155	0.7817 ± 0.0226	0.7583 ± 0.0194	0.8158 ± 0.0000	0.8283 ± 0.0150	0.7950 ± 0.0183	0.7733 ± 0.0178	**0.8633 ± 0.0119**	0.7975 ± 0.0175
D3	0.9224 ± 0.0000	0.9237 ± 0.0000	0.9082 ± 0.0000	0.9011 ± 0.0000	0.8747 ± 0.0000	0.9146 ± 0.0000	0.9230 ± 0.0000	0.8945 ± 0.0000	**0.9247 ± 0.0000**
D4	0.9292 ± 0.0166	0.8933 ± 0.0189	0.8628 ± 0.0258	0.8528 ± 0.0220	0.8558 ± 0.0160	0.8752 ± 0.0158	0.8918 ± 0.0176	0.8944 ± 0.0250	**0.9592 ± 0.0123**
D5	0.6917 ± 0.0158	0.6972 ± 0.0158	0.6856 ± 0.0150	0.7047 ± 0.0160	0.7786 ± 0.0115	0.6867 ± 0.0187	0.7311 ± 0.0150	0.6689 ± 0.0150	**0.7836 ± 0.0175**
D6	0.6402 ± 0.0004	0.6949 ± 0.0008	0.6897 ± 0.0010	0.6826 ± 0.0009	--	0.6791 ± 0.0007	0.6753 ± 0.0010	0.6090 ± 0.0010	**0.7204 ± 0.0008**
D7	0.8888 ± 0.0009	0.9295 ± 0.0008	0.9224 ± 0.0009	0.8970 ± 0.0011	--	0.9103 ± 0.0010	0.9171 ± 0.0013	0.8571 ± 0.0018	**0.9526 ± 0.0004**
D8	0.6566 ± 0.0000	0.6769 ± 0.0000	0.6446 ± 0.0000	0.6622 ± 0.0000	--	0.6410 ± 0.0000	0.6160 ± 0.0000	0.5125 ± 0.0000	**0.7297 ± 0.0000**
D9	0.5165 ± 0.0042	0.5510 ± 0.0041	0.5526 ± 0.0055	0.5604 ± 0.0038	--	0.5605 ± 0.0038	0.5599 ± 0.0056	0.5445 ± 0.0055	**0.5647 ± 0.0065**
D10	0.6738 ± 0.0016	0.6855 ± 0.0021	0.7099 ± 0.0023	0.6964 ± 0.0035	--	0.6741 ± 0.0016	0.6821 ± 0.0020	0.6988 ± 0.0016	**0.7292 ± 0.0029**
D11	0.4978 ± 0.0000	0.5598 ± 0.0000	0.5565 ± 0.0000	0.5196 ± 0.0000	--	0.5652 ± 0.0000	0.5620 ± 0.0000	0.5457 ± 0.0000	**0.5902 ± 0.0000**
D12	0.5593 ± 0.0021	0.6854 ± 0.0033	0.6926 ± 0.0015	0.6220 ± 0.0019	--	0.6074 ± 0.0024	0.6003 ± 0.0030	0.5700 ± 0.0020	**0.7230 ± 0.0037**
D13	0.4968 ± 0.0000	0.5385 ± 0.0075	0.5421 ± 0.0100	0.5225 ± 0.0100	--	0.5482 ± 0.0166	0.5496 ± 0.0168	0.4989 ± 0.0000	**0.5556 ± 0.0114**

Table 12. Comparison of AUC values for testing performance of RF models trained on training sets processed by various sampling algorithms.

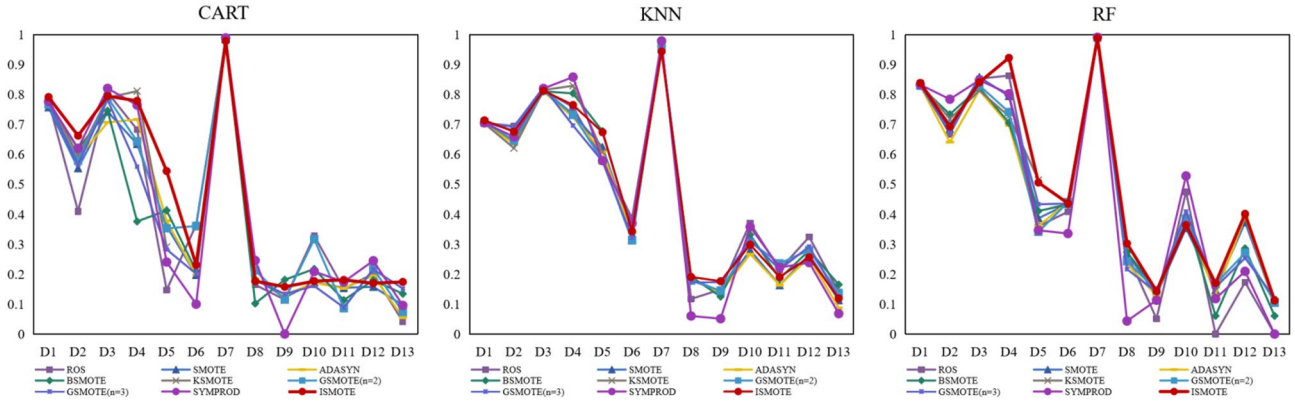

Fig. 7. Comparison of F1 values for test performance of three classifiers trained on training sets processed by various sampling algorithms.

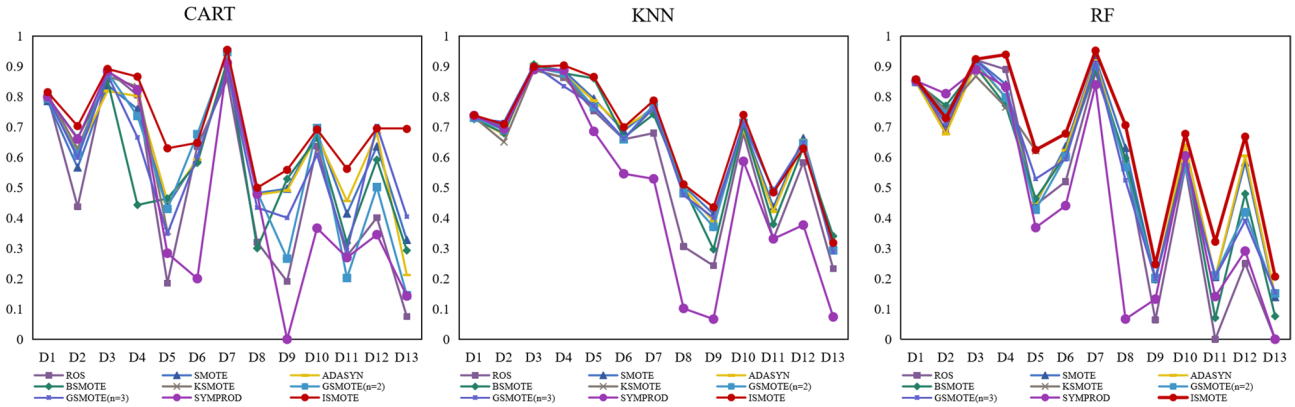

Fig. 8. Comparison of G-mean values for test performance of three classifiers trained on training sets processed by various sampling algorithms.

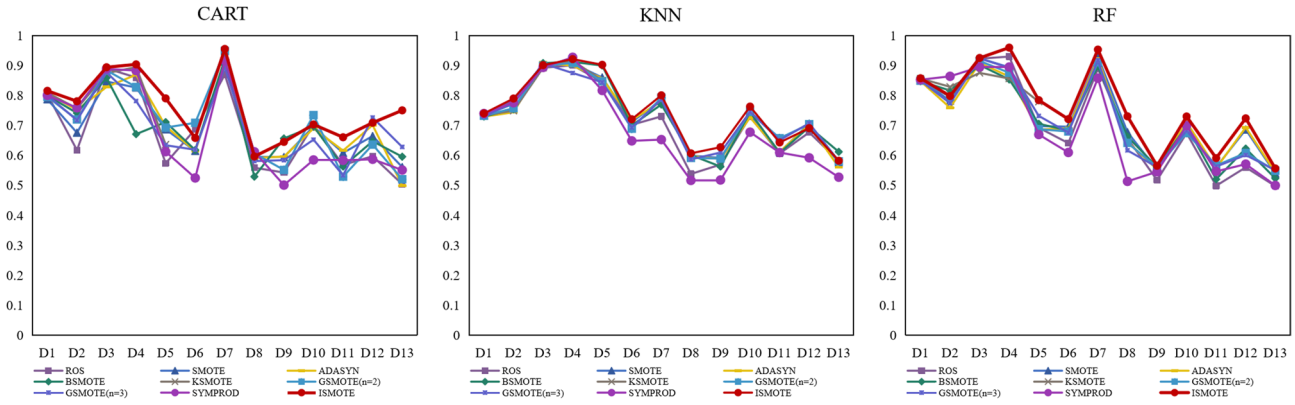

Fig. 9. Comparison of AUC values for test performance of three classifiers trained on training sets processed by various sampling algorithms.

D5, D6, D7, D8, D9, D10, D12, and D13, the AUC value of the model test performance indicator trained with the ISMOTE processed training set shows significant performance improvements. However, for D2, the AUC of the test performance indicator for the model trained using the ISMOTE-processed training set shows a larger gap compared to models trained using other algorithms.

Figures 7 and 8, and 9 are line graphs depicting the evaluation metrics of the CART, KNN, and RF classifiers, providing a more intuitive comparison of each classifier's performance. Compared to the other seven sampling

Dataset	ROS	SMOTE	ADASYN	BSMOTE	KSMOTE	GSMOTE(n=2)	GSMOTE(n=3)	SYMPROD	ISMOTE
D1	0.0020 s	0.0036 s	0.0044 s	0.0032 s	0.0431 s	0.0014 s	0.0011 s	0.0060 s	0.0036 s
D2	0.0024 s	0.0033 s	0.0041 s	0.0040 s	0.0295 s	0.0011 s	0.0010 s	0.0035 s	0.0031 s
D3	0.0031 s	0.0049 s	0.0129 s	0.0107 s	0.0381 s	0.0030 s	0.0029 s	0.2345 s	0.0056 s
D4	0.0020 s	0.0033 s	0.0041 s	0.0031 s	0.0294 s	0.0011 s	0.0011 s	0.0060 s	0.0030 s
D5	0.0026 s	0.0031 s	0.0037 s	0.0031 s	0.0408 s	0.0010 s	0.0010 s	0.0060 s	0.0029 s
D6	0.0027 s	0.0041 s	0.0145 s	0.0199 s	--	0.0029 s	0.0026 s	0.7711 s	0.0054 s
D7	0.0029 s	0.0032 s	0.0053 s	0.0057 s	--	0.0031 s	0.0032 s	0.7948 s	0.0031 s
D8	0.0034 s	0.0048 s	0.0116 s	0.0112 s	--	0.0026 s	0.0026 s	0.7255 s	0.0043 s
D9	0.0031 s	0.0028 s	0.0044 s	0.0049 s	--	0.0013 s	0.0012 s	0.0825 s	0.0028 s
D10	0.0041 s	0.0037 s	0.0130 s	0.0111 s	--	0.0021 s	0.0022 s	0.7686 s	0.0040 s
D11	0.0045 s	0.0035 s	0.0059 s	0.0057 s	--	0.0020 s	0.0021 s	0.0863 s	0.0039 s
D12	0.0031 s	0.0046 s	0.0103 s	0.0103 s	--	0.0023 s	0.0023 s	0.7863 s	0.0041 s
D13	0.0030 s	0.0027 s	0.0042 s	0.0047 s	--	0.0012 s	0.0012 s	0.0847 s	0.0029 s

Table 13. Comparison of the runtime of seven oversampling algorithms on ten datasets.

algorithms, most datasets processed by the ISMOTE algorithm proposed in this study can achieve higher F1 scores, G-mean values, and AUC values for the classifiers. Therefore, the ISMOTE algorithm, compared to the other seven mainstream oversampling algorithms, improves classifier performance to some extent by mitigating overfitting. The overlap and sparsity of samples in the dataset can significantly affect the performance of oversampling algorithms, leading to varying results across different datasets when using the ISMOTE algorithm. However, ISMOTE expands the sample generation space compared to other algorithms, the newly generated samples exhibit more diverse feature representations. As a result, ISMOTE shows improved performance compared to other algorithms.

The time complexity of the SMOTE algorithm needs to be calculated separately for the nearest neighbor search and sample generation. For N minority class samples, computing the Euclidean distances between all sample pairs using the k-nearest neighbors algorithm has a time complexity of $O(d \bullet N \log N)$, where d is the feature dimension. When generating M new samples, each interpolation requires $O(d)$ time, resulting in a total complexity of $O(d \bullet M)$. Therefore, the overall time complexity of the SMOTE algorithm is $O(d \bullet N \log N) + O(d \bullet M) \approx O(d \bullet N \log N)$.

The time complexity of the ISMOTE algorithm also needs to be calculated separately for the nearest neighbor search and sample generation. For N minority class samples, computing the Euclidean distances between all sample pairs using the k-nearest neighbors algorithm has a time complexity of $O(d \bullet N \log N)$, where d is the feature dimension. When generating M new samples, each interpolation requires $O(d)$ time, resulting in a total complexity of $O(d \bullet M)$. Additionally, the distance assessment and perturbation adjustments in ISMOTE require $O(d)$ time per modification, leading to a total complexity of $O(d \bullet M)$. However, this additional perturbation calculation does not increase the overall complexity order. Therefore, the time complexity of the ISMOTE algorithm remains $O(d \bullet N \log N) + O(d \bullet M) \approx O(d \bullet N \log N)$, the same as that of SMOTE.

Under identical experimental conditions, Table 13 presents a comparison of the runtime for seven algorithms across thirteen datasets. The results represent the average of ten experiments. For datasets where the KSMOTE algorithm could not be applied, the runtime is left blank. For dataset D7, the runtime cost of ISMOTE algorithm is second only to ROS. For dataset D9, the running time cost of ISMOTE algorithm is second only to GSMOTE algorithm. For datasets D10, D11, and D13, the running time cost of ISMOTE algorithm is second only to SMOTE algorithm and GSMOTE algorithm. For datasets D1, D2, D4, D5, D8, and D12, the ISMOTE algorithm ranks third in terms of runtime, second only to ROS and GSMOTE algorithms. For datasets D3 and D6, the ISMOTE algorithm ranks fourth in runtime, only behind ROS, GSMOTE, and SMOTE algorithms. Therefore, the ISMOTE algorithm improves classification performance without significantly affecting runtime.

Non-parametric statistical tests

This study utilizes Friedman's test and Holm's test to evaluate the statistical significance of the average rankings between ISMOTE and the baseline algorithms. The average ranking of algorithms is considered to measure how good an algorithm is relative to its competitors.Rankings are determined by assigning positions to each algorithm based on how much it contributes to the classifier's performance improvement. The sampling algorithm that enables the classifier to achieve the best performance on a specific dataset is assigned rank 1, the algorithm with the second-best performance receives rank 2, and so on. This process is repeated for each performance metric across all datasets, after which the average ranking is computed. Finally, rankings are computed for all performance metrics across all datasets, and the statistical significance of the ISMOTE algorithm is validated using Holm's test.

Figure 10 presents the ranking statistics of different classifiers based on Friedman's test. Specifically, for the CART classifier, ISMOTE ranks first in F1, G-mean, and AUC. For the KNN classifier, ISMOTE continues to rank first in G-mean, and AUC. For the RF classifier, ISMOTE ranks first in G-mean, and AUC, indicating its overall best performance on the RF classifier. These ranking results demonstrate that ISMOTE shows the most

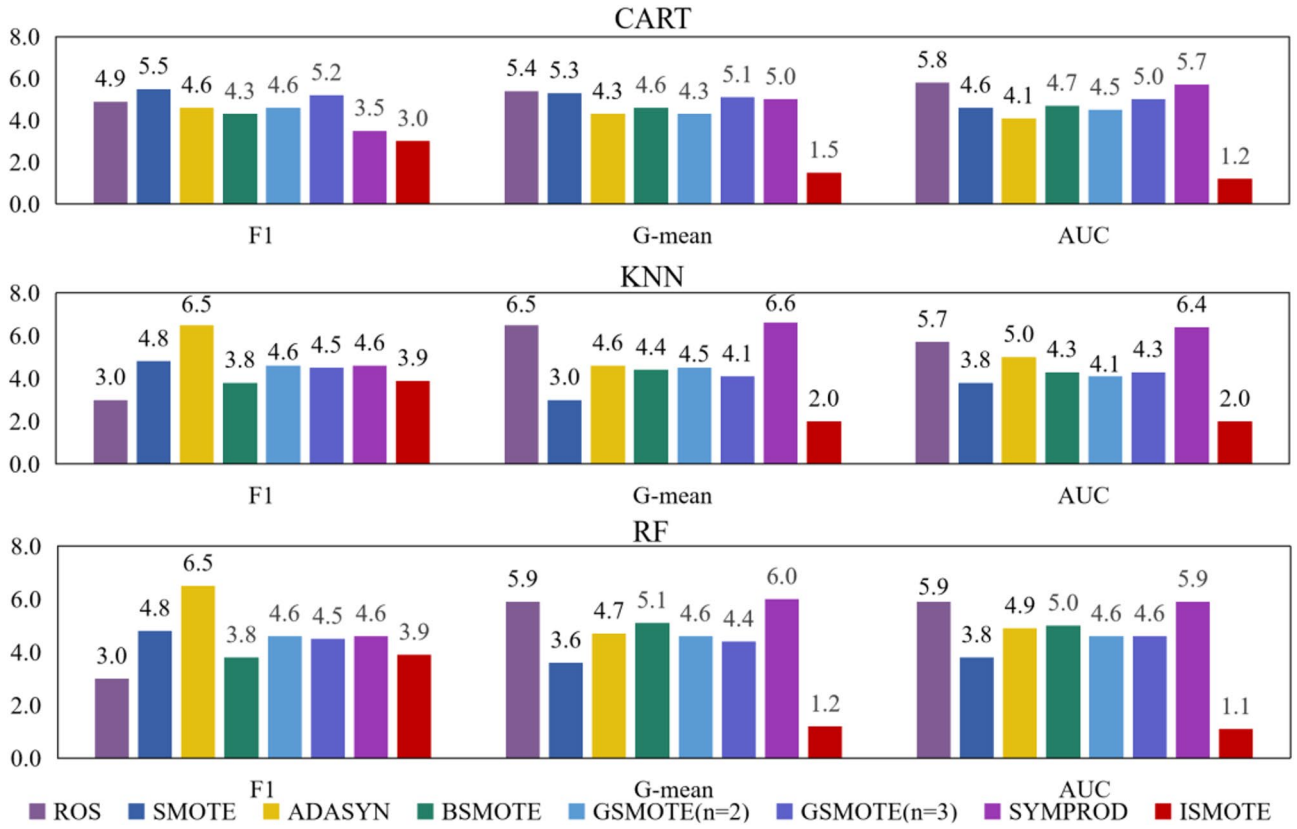

Fig. 10. Rankings obtained through Friedman's test.

Algorithm	Average ranking	*p*	Holm	Hypothesis
ROS	5.4	0.00002504	0.00005008	Rejected
SMOTE	5.1	0.00000009	0.00000068	Rejected
ADASYN	4.3	0.00000025	0.00000151	Rejected
BSMOTE	4.6	0.00000090	0.00000363	Rejected
GSMOTE($n=2$)	4.5	0.00075121	0.00075121	Rejected
GSMOTE($n=3$)	5.1	0.00000072	0.00000363	Rejected
SYMPROD	4.7	0.00001617	0.00004851	Rejected
ISMOTE	**1.9**	---	---	---

Table 14. Statistical analysis results of all hybrid sampling comparative algorithms based on CART classifier.

significant improvement in G-mean and AUC, while remaining competitive in F1. Although it does not achieve the highest rank in F1 for some classifiers, its performance remains within a reasonable range.

All experimental results were analyzed using Friedman's test for ranking, followed by Holm's test to assess whether ISMOTE exhibits significant advantages over other methods. Due to the missing results of the KSMOTE algorithm on certain datasets, it will not be subjected to non parametric statistical testing. The statistical analysis results are summarized in Tables 14, 15 and 16. At a significance level of $\alpha = 0.05$, all pairwise comparisons reject the null hypothesis (which assumes no significant differences among the compared methods), indicating that ISMOTE is statistically superior to other oversampling methods in most datasets. Holm's test results show that the p-values for all comparisons are less than 0.05, confirming that ISMOTE's improvements are statistically significant.

Friedman's test first analyzes the average rankings of different oversampling methods across multiple datasets, revealing that ISMOTE ranks higher. Holm's post-hoc test further computes the pairwise comparison p-values between ISMOTE and other methods, finding that $p < 0.05$, which confirms that ISMOTE's improvements are statistically significant. Since Holm's test p-values across all datasets are below 0.05, it can be concluded that ISMOTE's improvements in multiple datasets are statistically significant.

Notably, in the RF classifier, ISMOTE achieved the highest ranking in F1, G-mean, and AUC, significantly outperforming SMOTE, ADASYN, and other baseline methods in statistical analysis. In conclusion, ISMOTE

Algorithm	Average ranking	*p*	Holm	Hypothesis
ROS	5.1	0.000645	0.002579	Rejected
SMOTE	3.8	0.000062	0.000310	Rejected
ADASYN	5.4	0.000003	0.000022	Rejected
BSMOTE	4.2	0.021303	0.021303	Rejected
GSMOTE($n=2$)	4.4	0.001148	0.003444	Rejected
GSMOTE($n=3$)	4.3	0.005604	0.011208	Rejected
SYMPROD	5.9	0.000034	0.000204	Rejected
ISMOTE	**2.6**	---	---	---

Table 15. Statistical analysis results of all hybrid sampling comparative algorithms based on KNN classifier.

Algorithm	Average ranking	*p*	Holm	Hypothesis
ROS	5.6	0.00000103	0.000005	Rejected
SMOTE	4.0	0.00000024	0.000001	Rejected
ADASYN	5.0	0.00000019	0.000001	Rejected
BSMOTE	5.0	0.00000385	0.000008	Rejected
GSMOTE($n=2$)	4.5	0.00000256	0.000008	Rejected
GSMOTE($n=3$)	4.2	0.00000158	0.000006	Rejected
SYMPROD	5.6	0.00002212	0.000022	Rejected
ISMOTE	**1.7**	---	---	---

Table 16. Statistical analysis results of all hybrid sampling comparative algorithms based on RF classifier.

demonstrates statistical significance in various experimental settings, with substantial improvements in F1 and AUC compared to baseline methods, while maintaining strong competitiveness in G-mean. These statistical analyses further enhance the reliability of the experimental results, confirming the effectiveness of ISMOTE in imbalanced data classification tasks.

Conclusion

This research proposes an improved SMOTE algorithm to address the limitations of the original SMOTE algorithm. The new algorithm, ISMOTE, enhances SMOTE by modifying the generation space position of new samples. When the generation position of a new sample biases towards the k-nearest neighbors or the original sample, a random quantity is subtracted or added to ensure that the new sample is generated around the k-nearest neighbors or the original sample. ISMOTE generates new samples randomly between two samples or around single samples, thereby increasing the space for new sample generation and aligning the sample distribution more closely with the original sample distribution pattern. By expanding the generation space of new samples, ISMOTE effectively mitigates the problem of excessively high density in newly generated samples. Experimental results and visualizations demonstrate that ISMOTE effectively overcomes the problems of new sample distributions not conforming to the original sample distribution pattern and excessively high density of newly generated samples. It improves the classification accuracy of imbalanced datasets and enhances the performance of classification models. By calculating the proportion of oversampled data required for each minority class in multi-class problems and adjusting the parameters of the oversampling algorithm, the goal of balancing multiple classes can be achieved. Therefore, ISMOTE has good universality.

Certainly, the ISMOTE algorithm still has some limitations in certain aspects. First, regarding computational efficiency, although it maintains the same time complexity as SMOTE, the perturbation mechanism increases runtime, requiring a trade-off between efficiency and performance when processing large-scale datasets. Second, in terms of parameter sensitivity, the random perturbation coefficients (α, β) require empirical tuning, and improper settings may affect the quality of generated samples.Finally, for high-dimensional data, it is recommended to use feature selection methods in conjunction with ISMOTE to enhance the effectiveness of Euclidean distance measurement.

The ISMOTE algorithm can be applied to the problem of imbalanced data sets across various fields, including medical diagnosis, customer churn in telecommunications, financial fraud detection, software defect prediction, and hardware fault detection. However, the ISMOTE algorithm has higher computational complexity than the SMOTE algorithm due to the intricacy of the new sample positioning conditions. Additionally, the imbalance rate of the experimental datasets in this study is at a medium-low level. The next step will focus on addressing datasets with a high class imbalance rate. In the future, ISMOTE could be integrated with undersampling techniques to remove noise after oversampling, potentially enhancing dataset quality and further improving algorithm performance.

Data availability

Basic data supporting the results can be obtained through the website. The thirteen classification datasets are selected from the public KEEL database (https://sci2s.ugr.es/keel/datasets.php), the public UCI database (https://archive.ics.uci.edu/datasets), and the Kaggle dadasets (https://www.kaggle.com/datasets). The code of the ISM OTE algorithm and the dataset used in the experiment can be found at https://github.com/Sunshine68286/Improved-SMOTE-algorithm.

Received: 10 February 2025; Accepted: 27 June 2025
Published online: 02 July 2025

References

1. Karamti, H. et al. Improving prediction of cervical cancer using Knn imputed Smote features and multi-model ensemble learning approach. *Cancers* **15** (17), 4412 (2023).
2. Fotouhi, S., Asadi, S. & KattanMW A comprehensive data level analysis for cancer diagnosis on imbalanced data. *Biomed. Inf.* **90**, 103089 (2019).
3. Ileberi, E., Sun, Y. & Wang, Z. Performance evaluation of machine learning methods for credit card fraud detection using SMOTE and adaboost. *IEEE Access.* **9**, 165286–165294 (2021).
4. Abd El-Naby, A., Hemdan, E. E. D. & El-Sayed, A. An efficient fraud detection framework with credit card imbalanced data in financial services. *Multimed Tools Appl.* **82** (3), 4139–4160 (2023).
5. Ranjan, R. P. & Kumar, N. N. Software bug severity and priority prediction using SMOTE and intuitionistic fuzzy similarity measure. *Appl. Soft Comput.* **150**, 111048 (2024).
6. Feng, S. et al. Improving the undersampling technique by optimizing the termination condition for software defect prediction. *Expert Syst. Appl.* **235**, 121084 (2024).
7. Liu, D. et al. Feature-level SMOTE: augmenting fault samples in learnable feature space for imbalanced fault diagnosis of gas turbines. *Expert Syst. Appl.* **238**, 122023 (2024).
8. Wei, J. et al. Novel extended NI-MWMOTE-based fault diagnosis method for data-limited and noise-imbalanced scenarios. *Expert Syst. Appl.* **238**, 121799 (2024).
9. Thabtah, F., Hammoud, S., Kamalov, F. & Gonsalves, A. Data imbalance in classification: experimental evaluation, Informa. *Sciences* **513**, 429–441 (2020).
10. Rekha, G., Tyagi, A. K., Sreenath, N. & Mishra, S. Class Imbalanced Data: Open Issues and Future Research Directions. In 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 1–6. (2021).
11. Guo, H. et al. Learning from class-imbalanced data: review of methods and applications. *Expert Syst. Appl.* **73**, 220–239 (2017).
12. Yang, K. et al. Incremental weighted ensemble broad learning system for imbalanced data. *IEEE Trans. Knowl. Data Eng.* **34** (12), 5809–5824 (2022).
13. Wang, X., Li, L. & Lin, H. A review of SMOTE class algorithm research. *Comput. Sci.* **18** (5), 1135–1159 (2024).
14. de Morais, R. F. & Vasconcelos, G. C. Boosting the performance of over-sampling algorithms through under-sampling the minority class. *Neurocomputing* **343**, 3–18 (2019).
15. Dai, Q., Liu, J. & Liu, Y. Multi-granularity relabeled under-sampling algorithm for imbalanced data. *Appl. Soft Comput. J.* **124**, 109083 (2022).
16. Tao, X. et al. SVDD boundary and DPC clustering technique-based oversampling approach for handling imbalanced and overlapped data. *Knowl. Based Syst.* **234**, 107588 (2021).
17. Sonoda, R. Fair oversampling technique using heterogeneous clusters. *Inf. Sci.* **640**, 119059 (2023).
18. Sáez, J. A., Luengo, J., Stefanowski, J. & Herrera, F. SMOTE–IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Inf. Sci.* **291**, 184–203 (2015).
19. Li, J. et al. SMOTE-NaN-DE: addressing the noisy and borderline examples problem in imbalanced classification by natural neighbors and differential evolution. *Knowl. Based Syst.* **223**, 107056 (2021).
20. Zhang, A. et al. SMOTE-RkNN: A hybrid re-sampling method based on SMOTE and reverse k-nearest neighbors. *Inf. Sci.* **595**, 70–88 (2022).
21. Yu, H., Sun, C., Yang, X., Zheng, S. & Zou, H. Fuzzy support vector machine with relative density information for classifying imbalanced data. *IEEE Trans. Fuzzy Syst.* **27** (12), 2353–2367 (2019).
22. Khan, S. H., Hayat, M., Bennamoun, M., Sohel, F. A. & Togneri, R. Cost-Sensitive learning of deep feature representations from imbalanced data. *IEEE Trans. Neural Networks Learn. Syst.* **29** (8), 3573–3587 (2018).
23. Esposito, C., Landrum, G. A., Schneider, N., Stiefl, N. & Riniker, S. GHOST: adjusting the decision threshold to handle imbalanced data in machine learning. *J. Chem. Inf. Model.* **61** (6), 2623–2640 (2021).
24. Fernando, K. R. M. & Tsokos, C. P. Dynamically weighted balanced loss: class imbalanced learning and confidence calibration of deep neural networks. *IEEE Trans. Neural Networks Learn. Syst.* **33** (7), 2940–2951 (2022).
25. Dongdong, L. et al. Entropy-based hybrid sampling ensemble learning for imbalanced data. *Int. J. Intell. Syst.* **36** (7), 3039–3067 (2021).
26. Wang, Y. An ensemble learning imbalanced data classification method based on sample combination optimization. Journal of Physics: Conference Series 1284(1): 012135. (2019).
27. Zhong, X. & Wang, N. Ensemble learning method based on CNN for class imbalanced data. *J. Supercomputing.* **80** (7), 10090–10121 (2024).
28. Buda, M., Maki, A. & Mazurowski, M. A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* **106**, 249–259 (2018).
29. Elreedy, D. & Atiya, A. F. A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance. *Inf. Sci.* **505**, 32–64 (2019).
30. Pradipta, G. A., Wardoyo, R., Musdholifah, A., Sanjaya, I. N. H. & Ismail, M. SMOTE for handling imbalanced data problem: a review. In 2021 Sixth International Conference on Informatics and Computing (ICIC), Jakarta, Indonesia, 1–8. (2021).
31. Dablain, D., Krawczyk, B. & Chawla, N. V. DeepSMOTE: fusing deep learning and SMOTE for imbalanced data. *IEEE Trans. Neural Networks Learn. Syst.* **34** (9), 6390–6404 (2023).
32. Kaur, P. & Gosain, A. Comparing the Behavior of Oversampling and Undersampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise. ICT Based Innovations: Proceedings of CSI 2015. Springer Singapore, 23–30. (2018).
33. Mohammed, R., Rawashdeh, J. & Abdullah, M. Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. In the 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 243–248. (2020).
34. Batista, G. E., Prati, R. C. & Monard, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsl.* **6** (1), 20–29 (2004).

35. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. SMOTE: synthetic minority Over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002).
36. He, H., Bai, Y., Garcia, E. A. & Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), 322–1328. (2008).
37. Han, H., Wang, W. Y. & Mao, B. H. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. InInternational conference on intelligent computing, Berlin, Heidelberg: Springer Berlin Heidelberg, 3644(5):878–887. (2005).
38. Douzas, G., Bacao, F. & Last, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Inf. Sci.* **46** (5), 1–20 (2018).
39. Douzas, G. & Bacao, F. Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE[J]. *Inf. Sci.* **501**, 118–135 (2019).
40. Kunakorntum, I., Hinthong, W. & Phunchongharn, P. A synthetic minority based on probabilistic distribution (SyMProD) oversampling for imbalanced datasets[J]. *IEEE Access.* **8**, 114692–114704 (2020).
41. Wang, A. X. et al. Addressing imbalance in health data: synthetic minority oversampling using deep learning[J]. *Comput. Biol. Med.* **188**, 109830 (2025).
42. Du, W. et al. Secure Privacy-Preserving SMOTE for Vertical Federated Learning[C]//International Conference on Advanced Data Mining and Applications. Singapore: Springer Nature Singapore, : 301–315. (2024).
43. Suguna, R. et al. Mitigating class imbalance in churn prediction with ensemble methods and SMOTE[J]. *Sci. Rep.* **15** (1), 1–20 (2025).
44. Huang, J. et al. Deciphering decision-making mechanisms for the susceptibility of different slope geohazards: A case study on a SMOTE-RF-SHAP hybrid model[J]. *J. Rock Mech. Geotech. Eng.* **17** (3), 1612–1630 (2025).
45. Imani, M., Beikmohammadi, A. & Arabnia, H. R. Comprehensive analysis of random forest and XGBoost performance with SMOTE, ADASYN, and GNUS under varying imbalance levels[J]. *Technologies* **13** (3), 88 (2025).
46. Alcalá-Fdez, J. et al. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *J. Mult Valued Log. Soft Comput.* **17**, 255–287 (2015).
47. Swana, E. F., Doorsamy, W. & Bokoro, P. Tomek link and SMOTE approaches for machine fault classification with an imbalanced dataset. *Sensors* **22** (9), 3246–3246 (2022).
48. Cheng, K. et al. Grouped SMOTE with noise filtering mechanism for classifying imbalanced data. *IEEE Access.* **7**, 170668–170681 (2019).
49. Shaik, A. B. & Srinivasan, S. A Brief Survey on Random Forest Ensembles in Classification Model. In International Conference on Innovative Computing and Communications: Proceedings of ICICC, 2:253–260, Springer Singapore. (2018).
50. Chen, B., Xia, S., Chen, Z., Wang, B. & Wang, G. RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise. *Inf. Sci.* **553**, 397–428 (2020).
51. Maulidevi, N. U. & Surendro, K. SMOTE-LOF for noise identification in imbalanced data classification. *J. King Saud University-Computer Inform. Sci.* **34** (6), 3413–3423 (2022).
52. Ramentol, E., Caballero, Y., Bello, R. & Herrera, F. SMOTE-RSB *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowl. Inf. Syst.* **33**, 245–265 (2012).
53. Guan, H., Zhang, Y., Xian, M., Cheng, H. D. & Tang, X. SMOTE-WENN: solving class imbalance and small sample problems by oversampling and distance scaling. *Appl. Intell.* **51** (3), 1–16 (2020).

Acknowledgements

This research was supported in part by the National Natural Science Foundation of China (Grant No. 62062051), Major Talent Project in Guangxi Province, Emergency Management Joint Innovation Technology Research Project of Guangxi (2024GXYJ052), Research Projects in Philosophy and Social Sciences of Guangxi (24GLF005) and Research Project on School Safety, Stability, and Emergency Response of Guangxi (GXAW2024A008).

Author contributions

Y.Y.L. conceptualized and designed the study; Y.Y.L. was responsible for preprocessing the data, conducting code experiments, and checking the results; Y.Y.L. and L.Y. analyzed and interpreted the data, and are major contributors to writing the manuscript; L.Y., S.P.H., D.L., and R.R. gave constructive suggestions for the manuscript. All authors revised the manuscript for important intellectual content and approved the final version.

Funding

This research was supported in part by the National Natural Science Foundation of China (Grant No. 62062051), Major Talent Project in Guangxi Province, Emergency Management Joint Innovation Technology Research Project of Guangxi (2024GXYJ052), Research Projects in Philosophy and Social Sciences of Guangxi (24GLF005) and Research Project on School Safety, Stability, and Emergency Response of Guangxi (GXAW2024A008).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to P.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.