Sensors and Materials, Vol. 36, No. 11 (2024) 4835–4847
MYU Tokyo

S & M 3834

Tackling Class-imbalanced Learning Issues Based on Local Neighborhood Information and Generative Adversarial Networks

Chien-Chih Chen,[1] Yao-San Lin,[2] and Hung-Yu Chen[1*]

[1]Department of Information Management, National Chin-Yi University of Technology, Taiwan, ROC.
[2]Department of Industrial Engineering and Management, National Chin-Yi University of Technology, Taiwan, ROC.

(Received June 29, 2024; accepted October 23, 2024)

Keywords: class-imbalanced data, data augmentation, local neighborhood information, generative adversarial networks

Sensors are extensively used to collect data from systems. For example, in intelligent manufacturing, accelerometers are employed to gather process inputs and outputs in real time. However, abnormal events represent a small portion of the data, posing challenges for machine learning algorithms. Most algorithms lack the ability to account for equivalent sample representations. When addressing class imbalance, the widely used synthetic minority oversampling technique (SMOTE) has limitations. SMOTE does not consider the relative distributions between minority and majority class samples, potentially creating minority samples within the majority distribution. Additionally, its linear approach may miss nonlinear relationships among sample attributes. To overcome these issues, we propose a novel data augmentation method based on local neighborhood information and generative adversarial networks (GANs). Our approach first leverages density-based spatial clustering of applications with noise to identify minority class noises and then computes neighborhood types for minority samples using the *k*-nearest neighbors algorithm. On the basis of these neighborhood types (safe or dangerous), we create synthetic samples using GANs and bootstrapping. Evaluation on ten publicly available imbalanced datasets shows that our proposed method surpasses all other approaches for the majority of the datasets.

1. Introduction

Sensors, such as temperature sensors and scintillation detectors, are extensively used to collect data from various systems.[1,2] With the help of machine learning algorithms, intelligent systems are developed to monitor abnormal events. In most management tasks, decision-makers are particularly concerned with abnormal events because they have the potential to significantly impact future operations. However, abnormal events typically constitute a small minority within a dataset. This class imbalance can lead to biased inferences from machine learning algorithms and deep learning networks, despite their effectiveness in extracting valuable information from data.

*Corresponding author: e-mail: chenhy@ncut.edu.tw
https://doi.org/10.18494/SAM5224

ISSN 0914-4935
https://myukk.org/

1.1 Research background

Class-imbalanced data, such as skewed data, present a common challenge in machine learning algorithms for classification tasks. In class-imbalanced data, the distribution of classes is not even. Some classes have significantly more examples (named majority classes or negative classes) than in other classes (named minority classes or positive classes). Class-imbalanced data arise in many real-world scenarios, such as medical data analysis,[3] heart transplant outcome prediction,[4] and customer credit risk assessment.[5] In these cases, the minority classes are often critical and misclassifying them can be costly. When using traditional machine learning algorithms to train classifiers with imbalanced data, the models tend to be biased towards the majority class. This happens because the minority class examples are often treated as acceptable errors during the training process.

However, not all class-imbalanced data are difficult to learn. As illustrated in Fig. 1, there are three typical scenarios. In Fig. 1(a), where there is no overlap between the majority and minority class samples (referred to as minority samples), most classifiers can accurately discriminate between the two classes. Figure 1(b) depicts a possible decision-tree-like representation of this

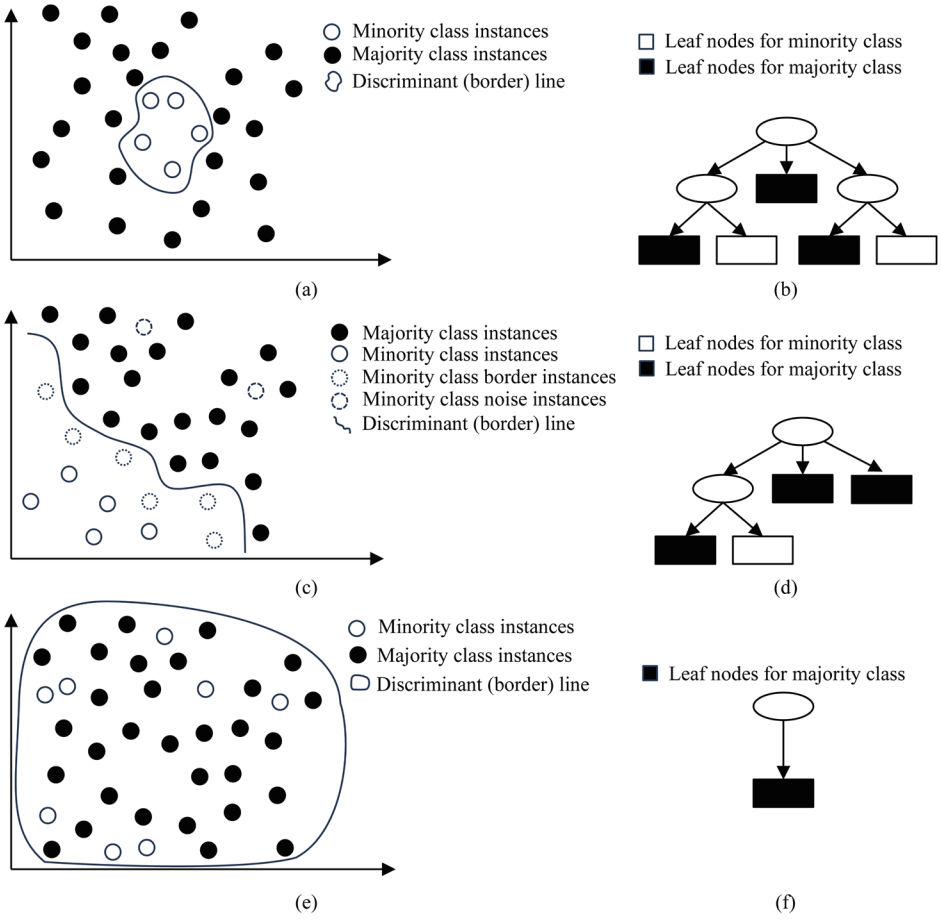

Fig. 1. Three scenarios for class-imbalanced data: (a) non-overlapping and (b) decision tree for non-overlapping; (c) mild overlapping and (d) decision tree for mild overlapping; (e) extreme overlapping and (f) decision tree for extreme overlapping.

scenario. In Fig. 1(c), a situation called mild overlap occurs, where some minority samples are surrounded by the majority class. Here, most classifiers can still identify minority samples, but with a potentially lower accuracy rate. This is because the surrounded minority samples might be treated as negligible errors during training, leading the classifier to misclassify a similar minority sample as belonging to the majority class. Figure 1(d) shows this effect. In Fig. 1(e), we see extreme overlap, a situation particularly common in big data. Here, minority samples, often consisting of only a few data points (disjuncts), are scattered and surrounded by the majority class. In this case, as shown in Fig. 1(f), classifiers may incorrectly infer that all samples belong to the majority class.

Previous research on handling class imbalance can be broadly categorized into three approaches.

1. Cost-sensitive methods modify the training phase of traditional machine learning algorithms. These methods assign higher penalties for misclassifying minority samples or adjust the weight of each data point based on whether it is classified correctly.[6]
2. Kernel-based methods focus on modifying the kernel function used by machine learning algorithms, for example, support vector machines (SVMs), to create a more robust model during training with imbalanced datasets.[7]
3. Data-oriented methods[8] directly address the class imbalance in the training data itself. This category includes two main approaches: undersampling and oversampling. Undersampling attempts to create a more balanced training set by removing majority class instances. Oversampling methods, conversely, generate additional minority class instances. A popular example is the synthetic minority oversampling technique (SMOTE).[8] SMOTE relies on the *k*-nearest neighbors (*k*NNs) algorithm to generate synthetic data. SMOTE identifies *k*NNs belonging to the minority class for a given minority instance. Then, SMOTE creates new synthetic instances (SMOTE instances) by interpolating between a minority instance (called the seed example) and one of its *k*NNs (called the selected example), as illustrated in Fig. 2 and Eq. (1).

$$\begin{cases} x_1''' = x_1'' + \left(x_1' - x_1''\right) \times s_1, \\ x_2''' = x_2'' + \left(x_2' - x_2''\right) \times s_2, \end{cases} \tag{1}$$

where $s_1 \in [0,1]$ and $s_1 \in [0,1]$ are random numbers.

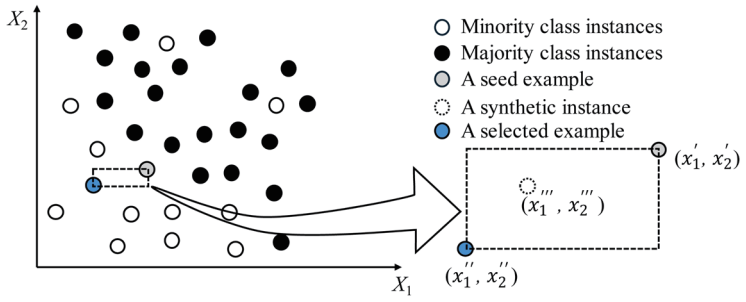

Fig. 2. (Color online) SMOTE generates synthetic instances between a seed minority sample and one of the seed's *k*NNs.

While SMOTE effectively addresses class imbalance, it neglects the distribution of the majority class when generating synthetic instances. This can lead to noise data, as some SMOTE instances become isolated by the majority. To address these limitations, several SMOTE extensions have been developed, which can be broadly categorized into two groups. 1. Data cleaning approaches[9,10] remove noise instances through an iteration process after sample creation. 2. Seed example selection approaches identify suitable minority examples as seed examples for SMOTE to prevent the generation of noisy synthetic data. In the second group, some typical methods are borderline-SMOTE (B1-SMOTE and B2-SMOTE),[11] ADASYN,[12] safe-level SMOTE (SL-SMOTE),[13] and local neighborhood SMOTE (LN-SMOTE).[14]

Focusing on seed example selection approaches, there are three main strategies employed to assess whether a minority sample can be considered a seed example. As illustrated in Fig. 3, these approaches categorize minority samples into three regions: safe, boundary, and dangerous. The classification of these regions is based on the class ratio within a minority sample's *k*NNs. When $k = 1$, meaning all k neighbors belong to the minority class, the region is considered safe. Conversely, when $k = 0$, indicating all k neighbors belong to the majority class, the region is considered dangerous. When k falls between 0 and 1, i.e., some k neighbors belong to the majority class, the region is considered a boundary area, where classifiers would potentially draw a discriminant line. Table 1 summarizes SMOTE extensions and their advantages and disadvantages in accordance with the seed example selection strategy they adopt.

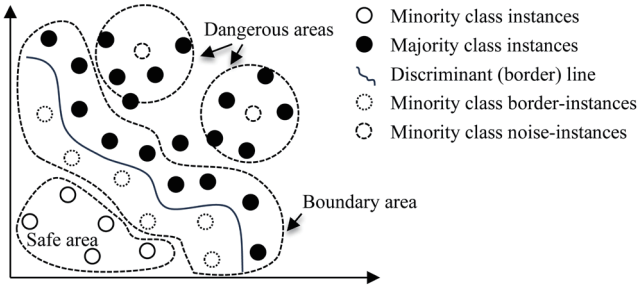

Fig. 3. SMOTE generates synthetic instances between a seed minority sample and one of the seed's *k*NNs.

Table 1
Categories of SMOTE extensions based on their seed example selection strategies.

Categories	Dangerous area	Boundary area	Safe area
Extensions	ADASYN,[12] and MWMOTE[15]	B1-SMOTE,[11] B2-SMOTE,[11] and BIBO[16]	SL-SMOTE[13] and LN-SMOTE[14]
Advantages	Improve the identification of minority instances, especially those surrounded by majority instances.	Improve the identification of minority instances, especially those in boundary areas.	Improve the identification of minority instances.
Disadvantages	Increase misclassification of majority instances.	Increase misclassification of majority instances.	Information on the minority instances surrounded by majority instances is still missing.

1.2 Research motivation

While SMOTE and its extensions have been effective in handling class-imbalanced data, their reliance on Eq. (1) for linear synthetic sample generation can lead to noisy samples, especially when there are nonlinear relationships between attributes. Recognizing this limitation, researchers[17,18] have explored generative adversarial networks (GANs)[19] as an alternative. However, current GAN-based approaches lack a mechanism to identify suitable seed examples from the minority class. Additionally, generating synthetic samples in areas where the majority class is dominant can lead to misclassifications.

Accordingly, we propose a novel mechanism to identify suitable seed examples from the minority class for GAN-based synthetic sample generation. The proposed mechanism first utilizes density-based spatial clustering of applications with noise (DBSCAN)[20] to eliminate minority class noise. Subsequently, *k*NN is employed to classify the neighborhood types (local neighborhood information) of the remaining minority samples. Minority samples identified as having "safe" neighborhoods are used to train the GAN, while those with "dangerous" neighborhoods are included in a bootstrapping procedure.[21]

To assess our method's efficacy, we employed ten datasets from the UC Irvine Machine Learning Repository (UCI) encompassing various class imbalance ratios. We compared our method's effectiveness against established benchmarks, including SMOTE, B1-SMOTE, ADASYN, SL-SMOTE, and LN-SMOTE. The results convincingly demonstrate that our proposed method surpasses all other approaches on the majority of datasets.

The remainder of this study is organized as follows. In Sect. 2, we explain the proposed method. Section 3 concerns the details the experimental datasets and setup and the obtained results are presented. Finally, Sect. 4 is the conclusion.

2. Proposed Method

The proposed method is illustrated in Fig. 4. It consists of three main steps: clustering the minority class with DBSCAN, identifying the neighborhood types of minority samples using *k*NN, and generating synthetic minority samples. Details are provided below.

2.1 Data normalization

Clustering algorithms often rely on distance metrics to group data points. When attributes have significantly different scales, distances between points can be misleading. To address this issue, data normalization is typically performed before clustering. In this study, the min-max normalization is employed. This technique transforms each numerical attribute's values to a range between 0 and 1. The transformed value $x_{i,j}{}'$ is computed as

$$x_{i,j}{}' = \frac{x_{i,j} - min_j}{max_j - min_j} \in [0,1], \tag{2}$$

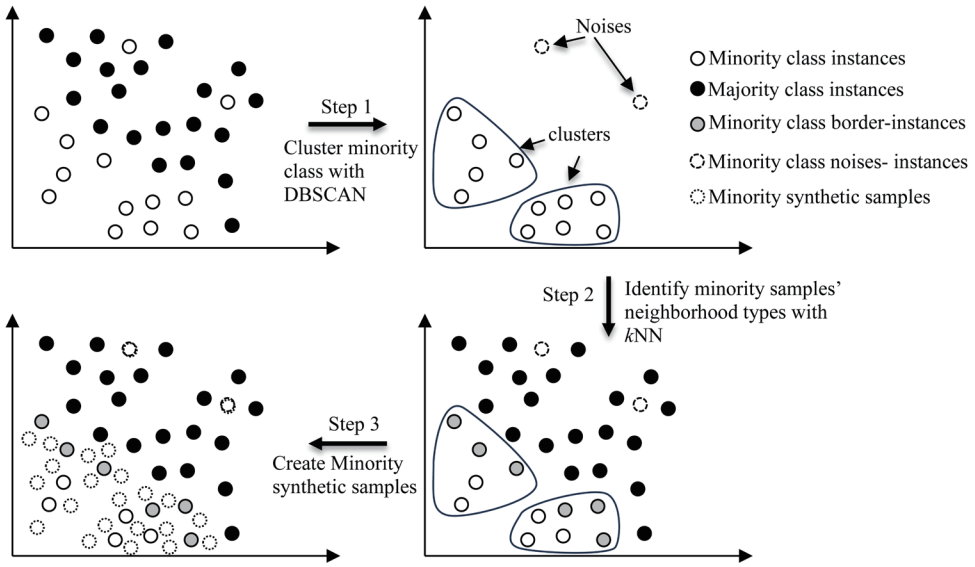

Fig. 4. Processes of the proposed method.

where $x_{i,j}$ is the ith original value in the jth attribute, min_j is the minimum value in the jth attribute, and max_j is the maximum value in the jth attribute.

2.2 Data clustering

The purpose of DBSCAN is to assist in identifying the distributions of minority samples. Samples from the minority class that are far from other data points are considered noise. These noisy points may be surrounded by majority class samples, indicating they are located in dangerous neighborhoods. Creating samples in such dangerous neighborhoods can lead models to misclassify majority class instances. Conversely, if the noisy points are surrounded by other minority samples, they are in safe neighborhoods. However, creating samples in safe neighborhoods may lead models to output classification rules that do not exist in the real data. DBSCAN relies on two parameters: *Eps* (the search radius) and *MinPts* (the minimum number of data points needed to form a cluster). Figure 5 illustrates the process of finding clusters in DBSCAN involving three types of points.

1. Core points: These have at least *MinPts* (e.g., 2) other points within a distance of *Eps* from them (e.g., point *p* in Fig. 5).
2. Border points: These are located within *Eps* of one or more core points but do not have enough neighboring points within *Eps* to be considered core points (e.g., point *B*).
3. Noise points: These are neither core nor border points (e.g., point *C*).

Suppose we have an untested point q (the qth instance). The closeness function of a core point p determines whether q is part of the same cluster by

$$N_p(q) = \begin{cases} 1, & \text{if } dist(p,q) \le Eps \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

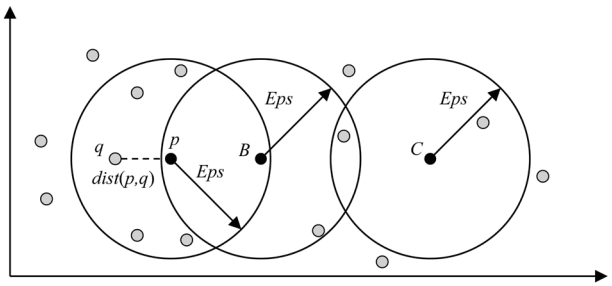

Fig. 5. Process of finding clusters in DBSCAN.

where $p \neq q$ and $dist(p, q)$ is defined as

$$dist(p,q) = \left[\sum_{j=1}^{m} \left| x_{p,j} - x_{q,j} \right|^2 \right]^{\frac{1}{2}}. \tag{4}$$

When $p \neq q$, *dist* represents the Euclidean distance and m is the number of attributes. When $q \in Eps(p)$ and $|Eps(p)| \geq MinPts$ are satisfied, points p and q belong to the same cluster. In this study, *MinPts* is set to 2 and the default *Eps* value is calculated as

$$Eps = \left[\sum_{j=1}^{m} \left[\frac{\sum_{i=1}^{n-1} \sum_{k=i+1}^{n} \left| x_{i,j} - x_{k,j} \right|}{\sum_{i=1}^{n-1} i} \right]^2 / m \right]^{\frac{1}{2}}, \tag{5}$$

where n is the number of minority samples.

2.3 Neighborhood type identification

Following the implementation of DBSCAN, we will utilize *k*NN to calculate the class ratio for each minority sample within its *k*NNs. The class ratio is computed as *nn*/*k*, where *nn* represents the number of minority class neighbors and k is typically set to 5 in SMOTE extensions.[12,14] On the basis of the class ratio, the neighborhood type for each minority sample is defined, as detailed in Table 2.

2.4 Sample generation

In this study, we employ two sample generation strategies. Referring to Table 2, minority samples are first categorized into two groups: safe (encompassing safe, relatively safe, and slightly safe) and dangerous (including dangerous, relatively dangerous, and slightly dangerous).

Table 2
Class ratios and their definition of neighborhood types.

Class ratio	Neighborhood type
1.0	Safe
0.8	Relatively safe
0.6	Slightly safe
0.4	Slightly dangerous
0.2	Relatively dangerous
0.0	Dangerous

Samples belonging to the safe group are then used to train a GAN for sample generation, while those in the dangerous group and those identified as noises by DBSCAN are treated as a population for bootstrapping, as shown in Fig. 6. The strategy is to avoid creating synthetic samples that would lower the models' discriminant power of the majority class.

3. Empirical Evaluation

In this section, we will introduce the experimental datasets, experimental environment, and experimental results.

3.1 Experimental datasets

There are ten public datasets taken from the UCI dataset repository for our experiments, as summarized in Table 3. In Table 3, "A" and "B" denote the number of dataset samples and the number of minority class samples, respectively. Imbalanced rates are computed as B/(A − B), i.e., the number of minority samples divided by the number of majority class samples. The datasets Abalone, Poker, Satimage, and Vowel have been modified to simulate binary class-imbalanced datasets. "Abalone_9_v_18" means the dataset contains samples with class labels 9 and 18; "Poker_5_v_6", "Poker_5_v_7", and "Poker_6_v_7" denote datasets containing samples with class labels 5 and 6, 5 and 7, and 6 and 7, respectively. "Satimage" contains class label 4 as the minority class and the other class labels as the majority class. "Vowel0" contains class label 0 as the minority class and the other class labels as the majority class.

3.2 Experimental environment

Table 4 summarizes the key details of our experimental settings. To ensure robust evaluation, we employed ten times stratified tenfold cross-validation for all experiments. Performance was measured using both accuracy and F1-score. Accuracy provides a baseline indicator for classification tasks, whereas F1-score is particularly well-suited for evaluating datasets with imbalanced classes. We utilized Python as the programming language and leveraged two powerful libraries: scikit-learn (sklearn) and PyTorch. Scikit-learn facilitated various tasks including min-max normalization, DBSCAN, *k*NN, bootstrapping, C4.5 decision tree construction, and metric calculation. PyTorch, on the other hand, was instrumental in building

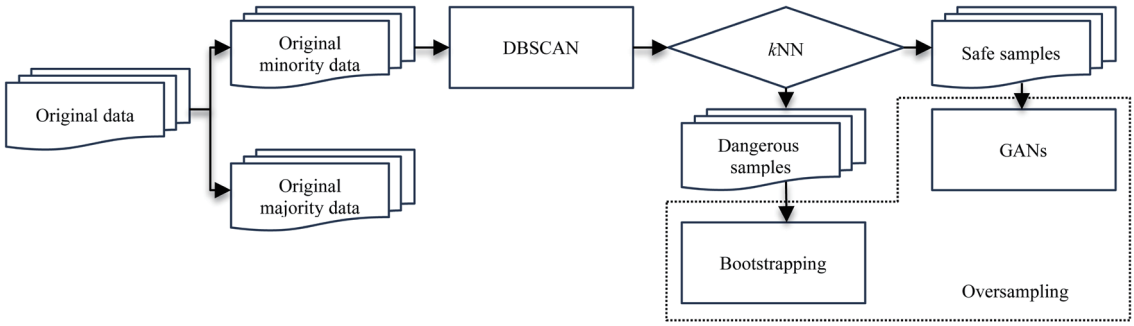

Fig. 6. Flow of the proposed sample generation.

Table 3
Details of the datasets adopted in our experiments.

Dataset	# of samples (A)	# of attributes	Minority class	# of minority samples (B)	Imbalanced rate
Abalone_9_v_18	731	8	18	42	0.061
Bupa	345	6	Positive	142	0.725
Cleveland	303	13	Recurrent	35	0.130
Haberman	306	3	Die	81	0.360
Pima	768	8	Positive	268	0.536
Poker_5_v_6	3510	10	6	1460	0.712
Poker_5_v_7	2286	10	7	236	0.115
Poker_6_v_7	1696	10	7	236	0.162
Satimage	6435	36	Label_4	626	0.108
Vowel0	990	13	0	90	0.100

Table 4
Details of experimental settings.

Parameter	Value
Evaluation approach	Ten times stratified tenfold cross-validation
Performance metrics	Accuracy rates (%) and *F*1-scores
Programming language	Python
Imported packages	scikit-learn (sklearn) and PyTorch
Evaluation approach	sklearn.model_selection.StratifiedKFold
Performance metrics	sklearn.metrics. accuracy_score sklearn.metrics.f1_score
Data normalization (min-max)	sklearn.preprocessing.MinMaxScaler
Data clustering (DBSCAN)	sklearn.cluster.DBSCAN
Data neighbor searching (*k*NN)	sklearn.neighbors.NearestNeighbors
Data oversampling (bootstrapping)	sklearn.utils.resample
Data oversampling (GAN)	PyTorch
Data modeling (C4.5 decision trees)	sklearn.tree.DecisionTreeClassifier

GANs. The effectiveness of the proposed method was compared against those of established oversampling techniques: SMOTE, B1-SMOTE, ADASYN, SL-SMOTE, and LN-SMOTE. All these comparison methods were also implemented in Python.

For a binary classification dataset, the model's forecasting performance can be summarized using a confusion matrix, as shown in Table 5. Referring to the four key categories, true positive (*TP*), true negative (*TN*), false positive (*FP*), and false negative (*FN*), we can compute performance metrics such as *accuracy*, *recall*, *precision*, and *F*1-*score* using Eqs. (6)–(9), respectively.

3.3 Experimental results

Tables 6 and 7 summarize the experimental results. "Original" refers to results obtained with a C4.5 decision tree trained on a dataset without any oversampling technique. Boldface values in the tables indicate the best-performing method among the seven tested on each dataset.

$$Accuracy\ rate = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$F1\text{-}score = \frac{TP}{TP + FP} \tag{9}$$

Table 5
Confusion matrix.

		Predicted values	
		Positive	Negative
Actual values	Positive	True positive (*TP*)	False negative (*FN*)
	Negative	False positive (*FP*)	True negative (*TN*)

Table 6
Summary of the experimental results: accuracy rates (%).

Datasets	Original	SMOTE	B1-SMOTE	ADASYN	SL-SMOTE	LN-SMOTE	Proposed method
Abalone_9_v_18	91.792	87.686	91.096	91.671	91.123	91.478	91.525
Bupa	63.781	63.347	62.000	63.389	63.800	63.071	63.542
Cleveland	72.760	72.715	72.588	73.615	73.062	72.671	74.051
Haberman	65.276	63.794	65.125	63.690	63.619	63.206	66.362
Pima	69.767	68.791	67.879	69.039	70.209	69.832	71.661
Poker_5_v_6	93.832	98.672	97.561	98.066	98.604	98.553	98.878
Poker_5_v_7	96.461	100.000	96.234	99.676	100.000	99.948	100.000
Poker_6_v_7	81.525	74.240	74.789	74.765	74.903	74.273	75.620
Satimage	90.890	90.082	90.315	90.539	90.466	90.566	91.125
Vowel0	98.348	98.061	98.864	98.465	98.202	98.187	98.595

Table 7
Summary of the experimental results: F1-scores.

Datasets	Original	SMOTE	B1-SMOTE	ADASYN	SL-SMOTE	LN-SMOTE	Proposed method
Abalone_9_v_18	0.316	0.288	0.311	0.332	0.284	0.302	0.395
Bupa	0.560	0.573	0.564	0.568	0.578	0.570	0.583
Cleveland	0.698	0.698	0.697	0.706	0.702	0.697	0.713
Haberman	0.329	0.367	0.381	0.350	0.342	0.351	0.782
Pima	0.565	0.573	0.559	0.570	0.586	0.577	0.594
Poker_5_v_6	0.926	0.984	0.971	0.977	0.983	0.982	0.986
Poker_5_v_7	0.822	1.000	0.807	0.984	1.000	0.998	1.000
Poker_6_v_7	0.374	0.197	0.202	0.211	0.214	0.195	0.232
Satimage	0.543	0.551	0.542	0.559	0.551	0.558	0.565
Vowel0	0.912	0.900	0.939	0.918	0.907	0.907	0.920

An analysis of the data in Tables 6 and 7 reveals that the proposed method generally outperforms the others across the ten datasets. Consider the Haberman dataset as an example. While the accuracy rates (%) of SMOTE, B1-SMOTE, ADASYN, SL-SMOTE, and LN-SMOTE fall below that of the "Original" method (65.276%), their F1-scores show the opposite trend. This suggests a trade-off: these methods improved the C4.5 decision tree's ability to discriminate the minority class but weakened its discrimination of the majority class. This likely occurs because these five methods generate unsuitable minority samples that confuse the model with respect to the majority class. In contrast, the proposed method creates samples using bootstrapping only when the local neighborhood is deemed "safe," mitigating this issue.

An interesting discovery is the result for the "Poker_6_v_7" dataset. Using oversampling techniques, including the proposed method, worsened both accuracy rates and F1-scores. The created synthetic instances became noisy samples, leading to unfavorable decision tree outputs.

4. Conclusions

Nowadays, various sensors are extensively used to collect data from systems, particularly in intelligent manufacturing. Sensors, such as cameras and accelerometers, are installed in equipment to monitor abnormal events in real time. Decision-makers are concerned about abnormal events, which can significantly impact future operations. However, these abnormal events (minority class samples) are much rarer than normal events (majority class samples) in the data. This type of data is called class-imbalanced data. Traditional machine learning algorithms tend to be biased towards the majority class as they consider minority class examples to be acceptable errors during training. In the past few decades, SMOTE has shown effectiveness in handling class-imbalanced data. However, it neglects the majority class distribution, leading to the generation of noisy synthetic instances. SMOTE extensions, including data cleaning and seed example selection approaches, are aimed at improving synthetic sample generation. Seed example selection approaches categorize minority samples into safe, boundary, and dangerous regions in accordance with the class ratio of their *k*NNs. However, these approaches still rely on

linear methods to create samples, which can lead to noisy samples when the relationships between attributes are nonlinear. In recent years, researchers have explored GANs as an alternative to SMOTE for synthetic sample generation. However, these studies still lack a mechanism for identifying suitable seed examples.

In this paper, we proposed a novel method that addresses this gap. We first employed DBSCAN to identify and remove noise from the minority samples. Then, we used *k*NN to classify the remaining minority samples into safe and dangerous neighborhoods on the basis of their local characteristics. Samples residing in safe neighborhoods were used to train the GAN, while those in dangerous neighborhoods were used in a bootstrapping procedure. The evaluation results on ten publicly available imbalanced datasets demonstrated that our proposed method outperforms all other approaches on the majority of the datasets. While we used Eq. (5) to determine a default value for the most critical parameter (*Eps*) in DBSCAN, the optimal value often required manual fine-tuning. Exploring heuristic algorithms, such as genetic algorithms, to automate *Eps* selection is a promising future direction.

References

1 I. Ramirez-Zavala Sergio, E. Vargas-Rodriguez, D. Guzman-Chavez Ana, and M. Salazar-Martinez Oscar: Sens. Mater. **36** (2024) 2943. https://doi.org/10.18494/SAM5026
2 H. Kimura, T. Fujiwara, M. Tanaka, T. Kato, D. Nakauchi, N. Kawaguchi, T. J. S. Yanagida: Sens. Mater. **35** (2023) 513. https://doi.org/10.18494/SAM4146
3 D.-C. Li, C.-W. Liu, and S. C. Hu: Comput. Biol. Med. **40** (2010) 509. https://doi.org/10.1016/j.compbiomed.2010.03.005
4 A. Dag, A. Oztekin, A. Yucel, S. Bulur, and F. M. Megahed: Decis. Support Syst. **94** (2017) 42. https://doi.org/10.1016/j.dss.2016.10.005
5 X. Zhang and L. Yu: Expert Syst. Appl. **237** (2024) 121484. https://doi.org/10.1016/j.eswa.2023.121484
6 I. D. Mienye and Y. Sun: Inf. Med. Unlocked **25** (2021) 100690. https://doi.org/10.1016/j.imu.2021.100690
7 J. Ren, Y. Wang, Y.-m. Cheung, X.-Z. Gao, and X. Guo: Pattern Recognit. **133** (2023) 108992. https://doi.org/10.1016/j.patcog.2022.108992
8 N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer: J. Artif. Intell. Res. **16** (2002) 321. https://doi.org/10.1613/jair.953
9 J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera: Inf. Sci. **291** (2015) 184. https://doi.org/10.1016/j.ins.2014.08.051
10 G. E. Batista, R. C. Prati, and M. C. Monard: ACM SIGKDD Explorations Newsletter **6** (2004) 20. https://doi.org/10.1145/1007730.1007735
11 H. Han, W.-Y. Wang, and B.-H. Mao: Advances in Intelligent Computing (ICIC, 2005) 878. https://doi.org/10.1007/11538059_91
12 H. Haibo, B. Yang, E. A. Garcia, and L. Shutao: 2008 IEEE Int. Joint Conf. Neural Networks (IEEE, 2008) 1322. https://doi.org/10.1109/IJCNN.2008.4633969
13 C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap: Advances in Knowledge Discovery and Data Mining (PAKDD, 2009) 475. https://doi.org/10.1007/978-3-642-01307-2_43
14 T. Maciejewski and J. Stefanowski: 2011 IEEE Symp. Computational Intelligence and Data Mining (CIDM, 2011) 104. https://doi.org/10.1109/CIDM.2011.5949434
15 H. He and E. A. Garcia: IEEE Trans. Knowl. Data Eng. **21** (2009) 1263. https://doi.org/10.1109/TKDE.2008.239
16 D. C. Li, Q. S. Shi, Y. S. Lin, and L. S. Lin: Entropy **24** (2022) 322. https://doi.org/10.3390/e24030322
17 R. Sauber-Cole and T. M. Khoshgoftaar: J. Big Data **9** (2022) 98. https://doi.org/10.1186/s40537-022-00648-6
18 B. Zhu, X. Pan, S. vanden Broucke, and J. Xiao: Inf. Sci. **609** (2022) 1397. https://doi.org/10.1016/j.ins.2022.07.145
19 G. Ian, P.-A. Jean, M. Mehdi, X. Bing, W.-F. David, O. Sherjil, C. Aaron, and B. Yoshua: arXiv:1406.2661 (2014). https://doi.org/10.48550/arXiv.1406.2661

20 J. Sander, M. Ester, H.-P. Kriegel, and X. Xu: Data Min. Knowl. Discovery **2** (1998) 169. https://doi.org/10.1023/A:1009745219419
21 B. Efron and R. J. Tibshirani: An Introduction to the Bootstrap (New York, Chapman and Hall, 1994) 1st ed. https://doi.org/10.1201/9780429246593

About the Authors

Chien-Chih Chen is an assistant professor in the Department of Information Management of National Chin-Yi University of Technology, Taiwan. His current interests are focused on machine learning with small data sets. His articles have appeared in Decision Support Systems, Omega, Automation in Construction, Computers and Industrial Engineering, International Journal of Production Research, Neurocomputing, and other publications. (frick@ncut.edu.tw)

Yao-San Lin is an associate professor in the Department of Industrial Engineering and Management of National Chin-Yi University of Technology, Taiwan, and has taught statistical analysis, information management and machine learning for many years. In the past ten years of research, he has twice won the IEEE Best Paper Award (2019 and 2013), and his research results have been published in international academic journals: Applied Soft Computing, Decision Support Systems, Neurocomputing, European Journal of Operational Research, International Journal of Production Research, Computers & Operations Research, and Expert Systems with Applications. His current research interests are mainly in the field of manufacturing and natural language processing. (yslin@ncut.edu.tw)

Hung-Yu Chen is an assistant professor in the Department of Information Management of National Chin-Yi University of Technology, Taiwan. His current interests are concentrated on human behavior with misinformation. His articles have been published in Decision Support Systems, Neurocomputing, International Journal of Advanced Manufacturing Technology, and other publications. (chenhy@ncut.edu.tw)