Hindawi
Scientific Programming
Volume 2021, Article ID 9947621, 18 pages
https://doi.org/10.1155/2021/9947621

Research Article

AWSMOTE: An SVM-Based Adaptive Weighted SMOTE for Class-Imbalance Learning

Jia-Bao Wang, Chun-An Zou, and Guang-Hui Fu

School of Science, Kunming University of Science and Technology, Kunming, 650500, China

Correspondence should be addressed to Guang-Hui Fu; guanghuifu@kust.edu.cn

Received 15 March 2021; Revised 21 April 2021; Accepted 26 April 2021; Published 13 May 2021

Academic Editor: Michele Risi

In class-imbalance learning, Synthetic Minority Oversampling Technique (SMOTE) is a widely used technique to tackle class-imbalance problems from the data level, whereas SMOTE blindly selects neighboring minority class points when performing an interpolation among them and inevitably brings collinearity between the generated new points and the original ones. To combat these problems, we propose in this study an adaptive-weighting SMOTE method, termed as AWSMOTE. AWSMOTE applies two types of SVM-based weights into SMOTE. A kind of weight is used in variable space to combat the drawbacks of collinearity, while another weight is utilized in sample space to purposefully choose those support vectors from the minority class as the neighboring points in the interpolation. AWSMOTE is compared with SMOTE and its improved versions with six simulated datasets and 22 real-world datasets. The results demonstrate the effectiveness and advantages of the proposed approach.

1. Introduction

The imbalanced classification problems are now attracting more attention in many fields of science, such as medical diagnosis [1], target making [2], text categorization [3–5], bioinformatics [6], and fraudulent credit card transactions [7], among others. The scenario of uneven data distribution occurs when one class far outnumbers the others. The data that we face are often imbalanced in real-world problems. In dealing with the imbalanced data, canonical classifiers usually tend to the majority samples to ensure overall accuracy, and most of minority samples are misclassified as majority samples. However, in many cases, the minority class is far more important and may carry more useful information; misclassification of minority samples can result in high costs. For example, misclassifying a patient as a healthy person may lead to deterioration and even death due to missing the optimal treatment period [8].

Class imbalance has been deemed as an open challenge in machine learning and data mining. Many traditional classification methods pay more attention to the majority class and ignore the minority class; hence, they are probably not suitable for imbalanced data [9]. To resolve the class imbalance problem, researchers have proposed several methods from the point of data level and algorithm level [10]. At the data level, the imbalance question is addressed by resampling, i.e., to modify a set of imbalanced data so as to obtain balanced data distribution to the subsequent learning tasks, while at the algorithm level, it is done by proposing new classification algorithms or modifying existing classifiers. These two approaches can be used independently or in combination to enhance classification performance.

Data resampling has been proved to be a highly effective method to deal with class imbalance, resampling includes oversampling [11], undersampling [12], and hybrid sampling [10]. Data resampling methods are the most frequently used in the imbalanced learning. From the algorithm level, cost-sensitive learning, one-class classification methods, and ensemble learning algorithms are some of the proposed methods that can be used for solving the problem of imbalanced learning [13]. Cost-sensitive learning considers higher misclassification costs for rare examples and measures the loss of unequal misclassification caused by the misclassification of machine learning algorithms. One-class classification (OCC) method is a classification method used

for datasets with only positive samples. It attempts to find patterns from the dataset and thus effectively separates the positive samples from the potentially negative samples from the larger hypothesis space. OCC only focuses on the similarity or matching degree of the sample. Ensemble learning is a general method for classifying data streams and has shown the potential to increase the classification accuracy as compared to an individual classifier [14]. Bagging [15], Boosting [16], SMOTEBoost [17], and RareBoost [18] are popular in imbalanced learning.

In this paper, inspired by SMOTE and SVM, a new effective oversampling method, called AWSMOTE, is proposed to deal with imbalanced learning. In the proposed algorithm, it divides minority samples into support vectors and nonsupport vectors according to SVM and assigns different weights to samples. Then, AWSMOTE makes each minority support vector generate the same number of new samples and uses SVM to predict these samples. The additional weight of each support vector is determined according to the accuracy of new sample prediction, namely, the more correctly predicted the new samples are, the greater the weight of the minority sample will be. Finally, we add the additional weights to the initial weights of the support vectors to highlight the importance of support vectors in generating new samples, and the weights of nonsupport vectors keep remain. Thus, different number of new samples can be generated by the weight of each minority class adaptively. Adaptive weighting has been widely used in the filed of imbalanced learning [19–21], such as Bobbili et al. [19] proposed a combination of informed sampling based on SMOTE and a postclassification adaptive weighting that takes into account a priori knowledge about a dataset. Different from the general adaptive weighted methods, AWSMOTE can judge which minority sample is more suitable to generate more new samples to improve the classification effect, and it combines with SVM to highlight the role of support vector. At the same time, we introduce the weight of variable through the estimate vector in SVM, which can weaken the collinearity brought by SMOTE and make the generated samples more consistent with the characteristics of the classifier.

The main contributions of this paper are summarized below:

(1) AWSMOTE considers and combines the weight of sample and variable in the process of oversampling, which focuses on the distributional characteristics and improves the classification accuracy of the data.

(2) According to the characteristics of SVM, AWSMOTE introduces a kind of weight in variable space to weaken the drawbacks of collinearity of SMOTE.

(3) AWSMOTE generates different numbers of new samples adaptively by using the weight of each minority sample.

(4) This paper performs plentiful experiments to compare AWSMOTE with other SMOTE methods. The statistical significance test is also performed to verify whether AWSMOTE significantly outperforms the other methods in terms of Precision, F_1, ACC, G-mean, and AUC.

This paper is organized as follows. Section 2 presents SVM, SMOTE, and other variations that are relevant for this study. The concrete demonstration of the proposed oversampling algorithm is presented in Section 3. The experimental results of the simulated and real datasets are discussed in Section 4. Finally, the summary and the prospect of future work are presented in Section 5.

2. SMOTE and SVM in Imbalanced Learning

2.1. Resampling Methods and SMOTE. The most common approaches of data resampling methods are random oversampling (ROS) and random undersampling (RUS) [22]. RUS randomly eliminates some samples of the majority class to achieve the purpose of balancing class distribution. Based on RUS, some new undersampling methods have been proposed, such as Tomek links [23], Neighborhood Cleaning Rule (NCL) [24], Edited Nearest Neighbor (ENN) [25], and One-sided Selection (OSS) [26]. Contrary to undersampling, ROS selects the existing minority samples randomly and replicates them to the original data. To avoid overfitting caused by ROS, Synthetic Minority Oversampling Technique (SMOTE) [27] generates the new samples by using an artificial interpolation method between the original minority samples and their nearest neighbors. SMOTE has been modified to propose improved versions such as ADASYN [28], Borderline-SMOTE [29], and DBSMOTE [30]. Hybrid sampling is a combination of any of these two methods, which eliminates some of the examples before or after resampling. The most common hybrid sampling methods are ROS + RUS [31], SMOTE + Tomek Links [32], SMOTE + ENN [32], SMOTE + NCL [33], etc. Furthermore, there are many other resampling methods that are associated with data types to make the algorithms more generalizable, such as Yang et al. [34] proposed an Adaptive Mahalanobis Distance-based Oversampling (AMDO) to overcome the problem of overlapping between classes and lacking of representative data and mixed-type data.

SMOTE is a traditional oversampling method, which adds new, factitious minority samples by interpolating between original minority examples rather than simply duplicating the former original samples. The minority class is oversampled by selecting each minority sample and generating synthetic samples along the line segments linking portion or all of the K minority nearest neighbors. According to the count of oversampling required, the K nearest neighbors are randomly chosen. The generated new synthetic sample $\mathbf{x}'$ formula is

$$\mathbf{x}' = \mathbf{x} + r \times (\mathbf{x} - \mathbf{x}^p), \tag{1}$$

where $r \in [0, 1]$ is a random number and $\mathbf{x}^p$ is randomly chosen among K nearest neighbors of the minority sample $\mathbf{x}$. The above operation can be repeated to obtain the required synthetic minority instances. SMOTE effectively improves the defect that ROS generates a large number of invalid

information samples, but it still has its own deficiencies. When linear interpolation is used on the minority samples, it cannot provide a scalar control on the number of the new samples and cannot select minority samples and synthesize the new samples with guidance, thus resulting in the poor quality of the new samples [35].

In order to eliminate its shortcomings, numerous variations of SMOTE have been proposed. Borderline-SMOTE first divide the minority instances into three sets, safe, noise, and danger, and then oversample the new instances in the appropriate region. To improve the bias of the original data distribution and optimize the efficiency of the classification of minority class, He et al. presented an ADASYN method which generates different numbers of the new samples for different minority samples based on the data distribution, which is equivalent to putting a weight for each minority instance. Both BLSMOTE and ADASYN are based on K nearest neighbor, which requires a large amount of calculation, and can be easily affected by noise problems. To address this problem, Density-based SMOTE (DBSMOTE) has been proposed, and it generates synthetic instances along the shortest path from each minority instance to a pseudocentroid of a minority-class cluster. DBSMOTE just takes $O(m^2)$, which is no worse than others in the SMOTE family, where m is the minority class size used in the computation [30].

2.2. Support Vector Machines for Classification. SVM [36] is a kind of generalized linear classifier that performs classification according to supervised learning. As the skew associated with imbalanced datasets pushes the hyperplane closer to the minority class [37], several proposals [38–41] attempt to bias the SVM algorithm so that the hyperplane is far away from the minority class.

For a binary classification problem, the basic thought of SVM is to find the separating hyperplane that can correctly divide the training dataset and have the maximum margin. Consider the training set $\{(\mathbf{x}_i, y_i)\}, i = 1, 2, \ldots, n$, where $\mathbf{x}_i \in R^p$ is a p-dimensional column vector and p is the number of features of the datasets and $y_i \in \{+1, -1\}$ denotes the class label. In order to find the optimal hyperplane with the maximum margin, the following quadratic optimization problem must be solved [42]:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & y_i\left(\mathbf{w}\cdot\Phi\left(\mathbf{x}_i\right) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & i = 1, 2, \ldots, n, \end{aligned} \tag{2}$$

where $\mathbf{w}$ and b are the estimate vector and the intercept of the separating hyperplane, respectively, $\xi_i (i = 1, 2, \ldots, n)$ are the slack variables used to normalize overfitting, and C is the soft boundary constraints for penalizing misclassified points.

Support vectors are the vital elements of the training set. Each class always has at least one support vector, and often more. As shown in Figure 1, the blue and red points are support vectors (SVs), which determine the separating hyperplane [43]. Given the two classes of support vectors, we can easily construct the maximum margin hyperplane. The non-SVs can be removed from the training set without changing the position and orientation of the hyperplane. Hence, support vectors play a key role in constructing the classification hyperplane in SVM [44].

Finally, the separating hyperplane is defined as

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b, \tag{3}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_p)^T$ are the estimate vectors. $f(\mathbf{x}) > 0$, the sample $\mathbf{x}$ is classified as minority class, while $f(\mathbf{x}) < 0$, the sample $\mathbf{x}$ is classified as majority class. The plus or minus value of each component indicates whether the influence of variable is positive or negative, and the magnitude signifies the degree of deviation. According to the properties of the support vector and the estimate vector from SVM, variable and sample are weighted, respectively, which is the main contribution of this study.

3. AWSMOTE: Adaptive-Weighting SMOTE

3.1. Weighting in Variable Space. Based on the analysis above, the collinearity problem may occur between the new samples generated by SMOTE and the original samples, that is, original data and the synthetic data are highly correlated, and it probably causes SMOTE to generate some invalid information samples in this case, which thus leads to the poor quality of the generated samples.

As compared to SMOTE and the derived methods, the advantage of introducing the variable weight assists in improving the distribution and weakening the collinearity of minority samples. We obtain the estimate vector $\mathbf{w}$ from the hyperplane of SVM in equation (3) as the weight of variable. Figure 2 shows the influence of variable weight on sample generation in two-dimensions, the horizontal axis represents variable V_1, and the vertical axis represents variable V_2, while w_1 and w_2 are the weights of variable 1 and variable 2, respectively. SMOTE generates new synthetic sample $\mathbf{x}'$ along the line segment between the minority sample $\mathbf{x}$ and its selected nearest neighbor $\mathbf{x}^p$. As the value and the direction of the weights are different, the generated samples are no longer linear combinations of $\mathbf{x}$ and $\mathbf{x}^p$. When $w_2 > w_1 > 0$, the generated sample $\mathbf{x}_1'$ is close to the nearest neighbor $\mathbf{x}^p$. In this case, $w_2 > w_1$, it is considered that the value and direction of w_2 can be considered to have a far bigger impact on the sample generation; hence, the direction of the generated minority sample $\mathbf{x}_1'$ is more inclined to w_2. Similarly, when $w_1 > w_2 > 0$, the weight and direction of w_1 have a greater influence on the sample generation; thus, the direction of the generated minority sample $\mathbf{x}_2'$ is more inclined to w_1. Conversely, when $w_1 < 0, w_2 < 0$, the direction of generated minority sample $\mathbf{x}_3'$ is opposite $\mathbf{x}_1'$.

In our algorithm, the element of j^{th} variable x_{ij}' of a new generated sample $\mathbf{x}_i'$ is defined as

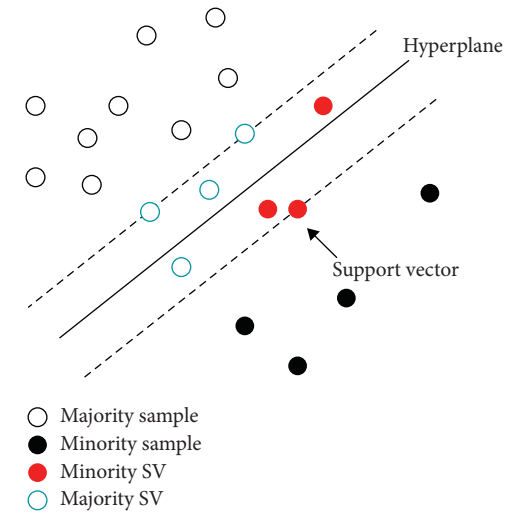


Figure 1: Support vector machines for classification.

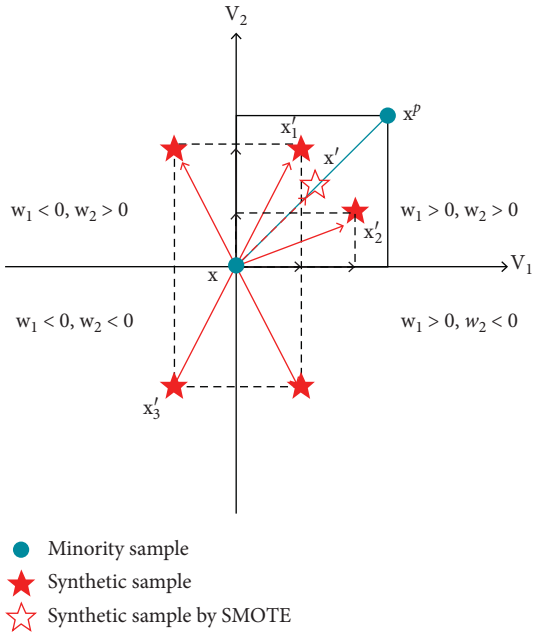


Figure 2: The distribution of the generated samples with different variable weights.

$$x_{ij}' = x_{ij} + r \times \left(\mathbf{x}_i - \mathbf{x}_i^k\right) \times w_j, \tag{4}$$

where r is a random number ranging from $[0, 1]$, $\mathbf{x}_i^k$ is one of K neighbors of minority sample $\mathbf{x}_i$, and w_j is the weight of variable where $j \in [1, p]$. When $w_1 = \cdots = w_p = w$, each variable weight has the same influence on the sample generation. In this case, the new sample is generated on the line segments joining the minority instance $\mathbf{x}_i$ with its neighbor $\mathbf{x}_i^k$, which is the way SMOTE synthesizes the new instance. In our approach, considering that each variable weight is not identical, hence, the influence of variable on sample generation is different. For example, w_1 is greater than other variable weights $w_2, \ldots, w_p$; thus, it is reasonable to assume that x_{i1}' plays a more important role in x_{ij}'; hence, x_{i1}' has more effect on the generation of $\mathbf{x}_i'$. Considering that, it is effective to introduce the variable weight to optimize the distribution of minority class to some extent.

When the variables are correlated, Figure 3(a) shows a negative correlation between the variables; Figure 3(b) shows a positive correlation between the variables. Under the influence of variable weights, a new sample is not created along the line of a minority sample and one of its nearest neighbors. When oversampling the boundary samples, SMOTE may intensify overlapping between classes as the neighboring point may be on the classification edge. However, after introducing the variable weights, the generated sample is not necessarily toward the neighboring samples. Therefore, the class-overlapping is avoided to some extent.

3.2. Weighting in Sample Space. SMOTE chooses the original minority samples and randomly selects the nearest neighbors for linear interpolation. However, the impact of each sample is not considered. ADASYN applies different weights on each minority sample; thus, different numbers of samples are generated. However, in high-dimensional space, the distance between the samples is approximately equal, which adversely affects the definition of the nearest neighbor, thus affecting the distribution of the generated samples.

In our algorithm, we propose a new adaptive oversampling method that weighs minority instances in sample space. Based on the SVM classifier, the minority instances are divided into nonsupport vectors and support vectors, while the number of support vectors is s and the number of nonsupport vectors is $m - s$ (m denotes the number of minority samples). We assign the initial weight of the support vector to (s/m) and the initial weight of the nonsupport vector to $(1 - (s/m))$. Since the separating hyperplane is mainly affected by support vectors, we further add different weights to the support vectors according to the accuracy of predicting the sample generation of each support vector; after that, the weight of each support vector is the initial weight plus additional weight, while the weight of nonsupport vectors remains unchanged. According to the different weights of each minority sample, the new samples are generated by selecting the nearest neighbor adaptively. As shown in Figure 4, different minority instances are due to the difference in weight, so the generated instances with the nearest neighbors are different.

3.3. AWSMOTE. Based on the above, we improve the method of SMOTE according to the weights of variables: for each minority class instance $\mathbf{x}_i$ with p variables, randomly select one of K neighbor instances $\mathbf{x}_i^k$, usually $K = 5$. The distance between two instances is calculated by Euclidean distance dis_i^k:

$$\text{dis}_i^k = \left(\mathbf{x}_i - \mathbf{x}_i^k\right) = \sqrt{\sum_{j=1}^{p} \left(x_{ij} - x_{ij}^k\right)^2}. \tag{5}$$

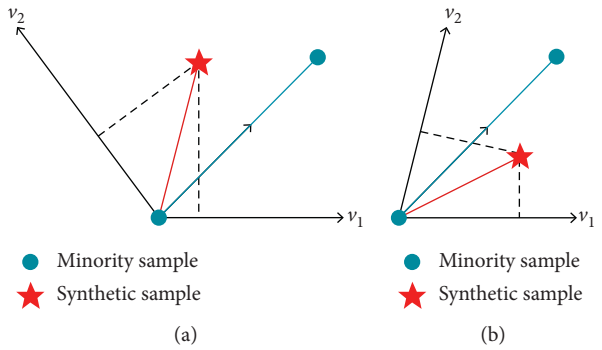


Figure 3: The distribution of the generated samples with correlative variables.

Each of the element of j^{th} variable that generates a new instance is defined as

$$x_{ij}^{'k} = x_{ij} + w_j \times r_i^k \times \text{dis}_i^k, \tag{6}$$

where $r_i^k \in [0, 1]$ is a random number, $j = 1, 2, \ldots, p$ and $k = 1, 2, \ldots, K$, and w_j is the weight of each variable so that the generated sample variables have similar properties to the separating plane. The proposed method is called as Adaptive-Weighting SMOTE (AWSMOTE):

Algorithm 1 is the whole process of AWSMOTE. Steps 1 and 2 calculate the weight of each variable and minority sample, respectively. The number of the new samples by each minority instance is computed based on the minority sample weights, and then, these samples should be generated according to the proposed method (steps 3–11). Finally, these samples should be added to the original minority class.

Algorithm 2 is the algorithm for calculating CaseWeight defined in Section 3.2. Steps 1 and 2 distinguish the minority samples with support vectors and nonsupport vectors and then assign different initial weights to them. Each sample generated by the support vector is predicted to obtain additional weights according to steps 3–14. Standardize all sample weights and output based on steps 15 and 16.

The time complexities of AWSMOTE include three key steps: step 1, step 2, and step 8 in Algorithm 1. In step 1, where most SVs are not at the upper bound and $s/n \ll 1$, the time complexity is $O(s^3 + s^2 n + spn)$. If $s/n \approx 1$, then the time complexity is $O(s^3 + sn + spn)$. For the case where most SVs are at the upper bound and $s/n \ll 1$, then the time complexity is $O(s^2 + spn)$. Finally, the time complexity is $O(pn^2)$, where most SVs are at the upper bound and $s/n \approx 1$ [45]. The time complexities are $O(ps)$ in step 2 and $O(m^2)$ in step 8, respectively [30]. In short, the overall time complexity of AWSMOTE algorithm is the sum of the time complexities of above three steps.

4. Experimental Setup and Result Analysis

In this section, we first introduce five measures in our experiments. In the following, AWSMOTE is compared to the classical SMOTE and other improved SMOTE methods, respectively, with simulation data and the instances. The classifier we adopt is SVM, and the kernel function of the SVM is radial.

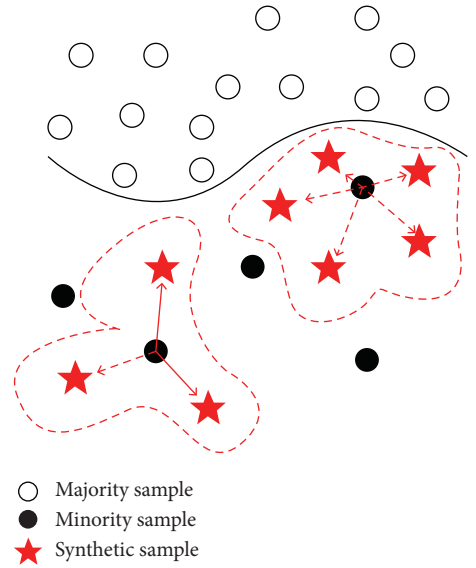


Figure 4: The influence of case weight on the distribution of the generated samples.

4.1. Performance Evaluation. In a binary classification problem, the accuracy of minority samples is far more important than the accuracy of majority samples, so it is essential to pay more attention to the minority class performance for the evaluation criteria. Table 1 presents the confusion matrix for a binary classification problem: the minority class presents the positive class, and the majority class presents the negative class. The main evaluation measures from the confusion matrix are as follows. In our experiments, we use Precision, F_1, G-mean, Accuracy, and AUC as performance measures:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}. \end{aligned} \tag{7}$$

Input: majority class $\mathscr{M}$, minority class m, nearest neighbor K, $M = |\mathscr{M}|$, and $m = |m|$
Output: a new minority class m_*
(1)A set of resamples from $\mathscr{M}$ and m are used to train SVM classifier and obtain the vector of variable weights $\mathbf{w}^V = (w_1^V, w_2^V, \ldots, w_p^V)^T$
(2)Calculate the vector of each minority case weight $\mathbf{w}^C = (w_1^C, w_2^C, \ldots, w_m^C)^T$:$\mathbf{w}^C = CaseWeight(\mathscr{M}, \mathscr{M}, K, \mathbf{w}^V)$
(3)**for** $i = 1$ to m **do**
(4) Count the number of samples generated for each minority case S_i: $S_i = [(M - m) \times w_i^C]$
(5) **for** $k = 1$ to K **do**
(6) Compute the k nearest neighbors of $\mathbf{x}_i$, obtain dis_i^k;
(7) **for** $j = 1$ to p **do**
(8) The element of j^{th} variable $x_{ij}^{'k}$ of the new sample $\mathbf{x}_i'$: $x_{ij}^{'k} = x_{ij} + w_j^V \times \text{dis}_i^k \times r_i^k$
Where $r_i^k \in [0, 1]$ is a random number
(9) **end for**
(10) **end for**
(11) Add all the samples $\mathbf{x}_i'$ generated by $\mathbf{x}_i$ to the generated minority class $\{m_+\}$
(12)**end for**
(13) $m_* = m + \{m_+\}$
(14)**return** A new minority class m_*

Algorithm 1: AWSMOTE.

Input: majority class $\mathscr{M}$, minority class $\mathscr{M}$, the vector of the variable weights $\mathbf{w}^V$, nearest neighbor K, $M = |\mathscr{M}|$, and $m = |\mathscr{M}|$
Output: the vector of each minority sample weight $\mathbf{w}^C$
(1)All $\mathscr{M}$ and a set of resamples from $\mathscr{M}$ are used to train SVM classifier
(2)Obtain the set of minority support vector $\mathcal{S}$, $s = |\mathcal{S}|$
(3)**for** $i = 1$ to s **do**
(4) **for** $k = 1$ to K **do**
(5) Compute the k nearest neighbors of $\mathbf{x}_i$ ($\mathbf{x}_i \in \mathcal{S}$) and obtain dis_i^k;
(6) **for** $j = 1$ to p **do**
(7) The element of j^{th} variable $x_{ij}^{'k}$ of the new sample $\mathbf{x}_i'$: $x_{ij}^{'k} = x_{ij} + w_j^V \times \text{dis}_i^k \times r_i^k$
Where $r_i^k \in [0, 1]$ is a random number
(8) **end for**
(9) Add the generated samples $\mathbf{x}_i^{'k}$ to set $\mathbf{x}_i'$ which is generated by $\mathbf{x}_i$ and obtain new minority samples $\mathbf{x}_i'$ as testing set; the original m and M as training set are used to train SVM classifier, and the number of correct predictions is s_i
(10) **end for**
(11) The additional weight of the support vector $\mathbf{x}_i$ is (s_i/K)
(12) The new weight of each support vector is $w_i^C = (s/m) + (s_i/K)$
(13) Standardized w_i^C, $w_i^C \leftarrow \mathbf{w}^C$
(14)**end for**
(15)The initial weight of the nonsupport vector $(1 - (s/m)) \leftarrow \mathbf{w}^C$, standardized $\mathbf{w}^C$
(16)**return** The vector of each minority sample weight $\mathbf{w}^C$

Algorithm 2: CaseWeight.

F_1 considers both Precision and Recall to compute the score, that is, the weighted harmonic mean of two criteria [46]:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{8}$$

G-mean is the geometric mean of accuracy of each class, which is commonly utilized when the data sets are highly imbalanced. G-mean is defined as [47]

$$G - \text{mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TN}}{\text{TN} + \text{FP}}}. \tag{9}$$

Table 1: Confusion matrix for a binary classification problem.

	Actual negative	Actual positive
Predict negative	True negative (TN)	False negative (FN)
Predict positive	False positive (FP)	True positive (TP)

Accuracy (ACC) is the ratio of the number of correctly classified samples to the total samples, and ACC is defined as [48]

$$\text{ACC} = \frac{\text{TP} + \text{FN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{10}$$

AUC is defined as the area under by the receiver operating characteristic (ROC) curve [49]; the closer the AUC gets to 1, the better the model built for the imbalanced data.

4.2. Simulation Study. In this section, the experimental results of the simulated datasets are discussed. First, the results of different methods are shown and analyzed, and then, the distribution of simulated datasets generated by various methods is shown to illustrate the advantages and disadvantages of different oversampling methods.

Six simulated datasets (Sim 1–6) are used to visually demonstrate the effects of different evaluations after several oversampling methods. The size of the total samples is fixed as 900, whereas the numbers of features are set as 50 and 500. The class imbalance ratios (IRs) are set as 3, 9, and 19, respectively. These datasets are presented in Table 2, which shows different dimensions and imbalance ratios in order to ensure the universality of the methods.

The class imbalance ratio (IR) is defined in equation (4), where M and m are the numbers of majority and minority instances in the dataset [50]. Therefore, the bigger the IR, the higher the imbalance of the problem. The IR is computed as follows:

$$\mathrm{IR} = \frac{M}{m}. \tag{11}$$

Figure 5 shows the results of the evaluations corresponding to each method. Each subgraph is ranked in the descending order according to the algorithm's performance in different measures. NO-RS stands for data which are classified directly by SVM without oversampling, and the methods we use for comparison are SMOTE, DBSMOTE, BLSMOTE, and ADASYN.

For the classification measures of each dataset, it can be seen that AWSMOTE outperforms SMOTE and the derived methods in Figure 5, and the proposed method shows great improvements especially in F_1 and Precision across all datasets. Among the simulated datasets tested, only in Sim 3 the results are worse than ADASYN by using AWSMOTE for ACC, AUC, and G-mean, but AWSMOTE is found still better than others. The poor performance of NO-RS in some measures such as G-mean is probably caused by the excessive-class overlapping or the high IR of the original datasets, so all minority samples are misclassified as majority samples. Generally, it can be observed that ADASYN and SMOTE representative implement the best results except AWSMOTE. This is consistent with the consensus. So, to summarize, from the aspect of simulated datasets, the classification ability of AWSMOTE is found to be competitive.

We simulate the distribution of data by using all methods in two-dimensional space. As shown in Figure 6, the title of each subgraph represents the oversample method. Figure 6(a) shows the distribution of the original data; the red point presents the majority class, the blue triangle presents the minority class, and the yellow square presents synthetic minority class. From Figures 6(b)–6(e), we show that SMOTE and the derived methods reach the class balance by synthesizing minority samples. However, these methods have their own disadvantages. In Figure 6(b), SMOTE generates randomly new instances all over the minority samples, which leads to an increase synthetic minority instances within the safe minority region. Considering the blindness of the selection of samples, an increase in synthetic minority instances can be on the edge of classification, or even on the majority class area as noise. Therefore, it does not efficiently increase the classification accuracy for classifying the minority instances by using SMOTE [51]. In Figure 6(c), it can be observed that BLSMOTE is concerned about the borderline minority samples to generate new instances, but the new minority instances mainly focus on the borderline region and show strong collinearity; thus, there are many invalid information samples on the classification boundary which confuses the classifier [24]. In Figure 6(d), the new instances are generated on the basis of the center of minority class, and it is also hard to avoid the generation of the new instances on the edge of classification by using DBSMOTE. It is highly possible that a part of the minority samples is all misclassified as the majority class, which indicates that the classifier learning information is not generalized enough. It is also observed that both BLSMOTE and DBSMOTE may cause a serious migration of the class center. For all these reasons, SMOTE, BLSMOTE, and DBSMOTE cannot generate minority instances adaptively according to sample characteristics [52]. ADASYN effectively alleviates the defects of other methods, which generate different numbers of new instances for different minority samples based on the data distribution in Figure 6(e). The principle of ADASYN shows that the more the majority samples of the K nearest neighbors of each minority sample, the higher the weight is assigned; thus, the more minority instances are generated. However, in this way, more new minority instances occur on the classification boundary, and thus, class-overlapping leads to the classification difficulty.

Our method is shown in Figure 6(f). As mentioned above, we divide the minority samples into support vectors and nonsupport vectors and then assign different weights to these vectors so that different minority samples generate different new instances purposively. Meanwhile, we introduce the variable weight so that the value and direction of variables affect the distribution of new instances. It can be seen that our method can effectively alleviate the overlap between classes, and weaken the collinearity of instances and, hence, consequently resolves the problem of classification difficulty caused by the excessive generation of new boundary instances.

Table 2: Simulated datasets.

Id	Dataset	Total	Minority	Majority	Features	IR
1	Sim1	900	225	675	50	3
2	Sim2	900	225	675	500	3
3	Sim3	900	90	810	50	9
4	Sim4	900	90	810	500	9
5	Sim5	900	45	855	50	19
6	Sim6	900	45	855	500	19

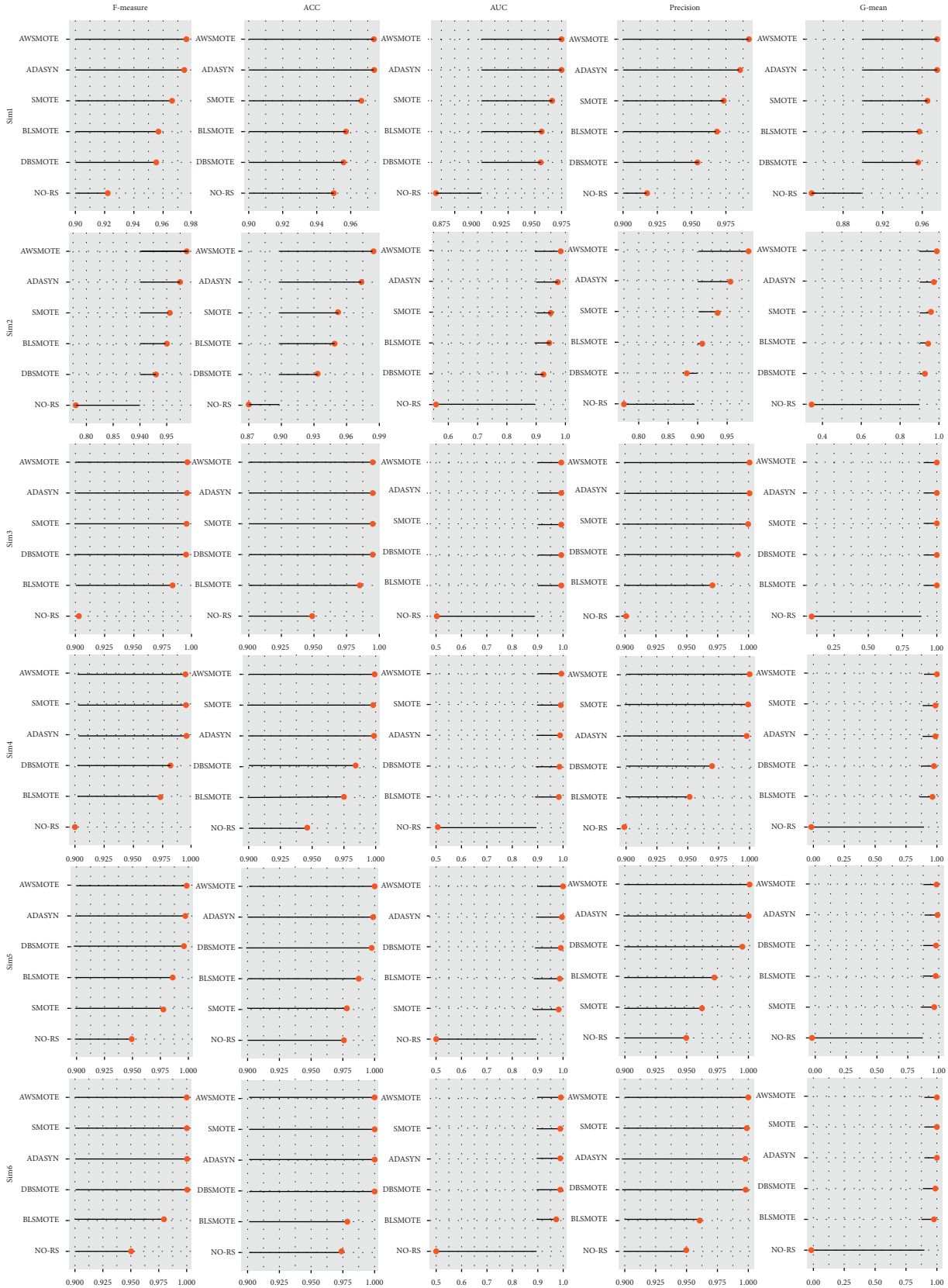


Figure 5: The performance of each method on metrics of different simulated datasets.

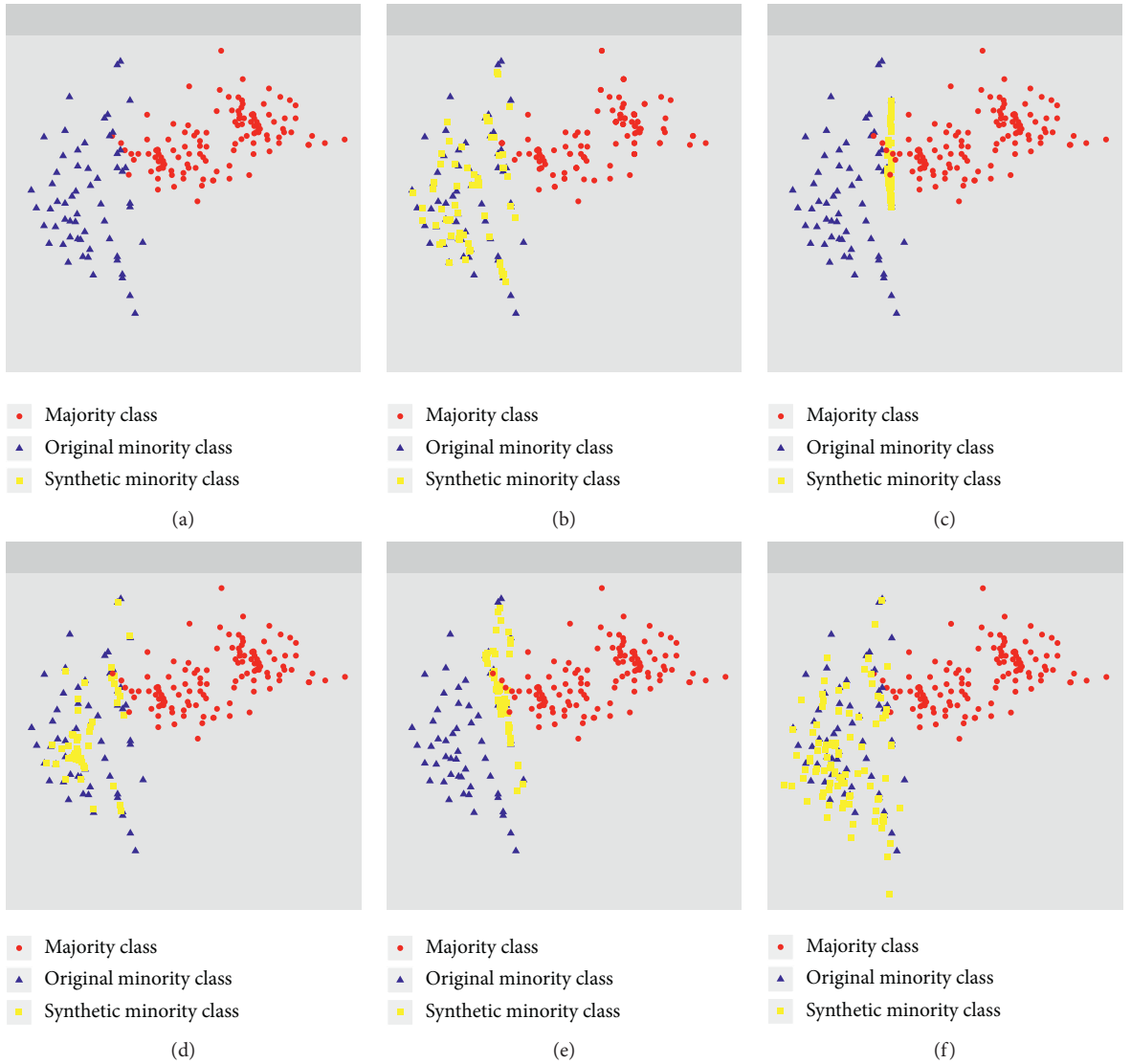


Figure 6: The distribution of simulated data generation by comparison of the oversampling methods. (a) Origin. (b) SMOTE. (c) BLSMOTE. (d) DBSMOTE. (e) ADASYN. (f) AWSMOTE.

4.3. Experimental Results and Discussion on Real Datasets. This section presents a comparison of methods in UCI Machine Learning Repository, KEEL Repository, and metabonomics datasets. Data description of the twenty-two real datasets with different degrees of imbalance studied in this work, 1–7 from UCI and 8–9 are metabonomics datasets, while the rest from the KEEL. The smallest dataset is CHD with 72 instances and the largest one is Wine-white3vs7 with 900 samples. The IR of these datasets varies between 1.29 and 44. Additionally, the majority class is constructed by grouping all the labels except the minority class, and the use of multiclass datasets is compelling that only a small part of the binary dataset available. The relevant information is presented in Table 3 for each dataset.

Libras movement: a hand movement type in LIBRAS, and the first 10 classes were merged into a majority class and the other into a minority class

Accent: dataset featuring single English words read by speakers from six different countries for accent detection and recognition, and class ES was studied as the minority class

Seeds: measurements of geometrical properties of kernels belonging to three different varieties of wheat, and type3 was studied as the minority class

CHD: a metabonomics dataset for healthy and coronary heart disease patients of the binary problem

HNP: a metabonomics dataset for healthy and brain damage patients of the binary problem

The best performance of the oversampling methods with a fixed classifier is in bold. AWSMOTE shows stronger competitiveness in F_1, AUC, ACC, and G-mean. Table 4 shows the F_1 comparisons of the six algorithms. Our algorithm is slightly inferior to other algorithms only in three

Table 3: Summary description of real datasets.

Id	Dataset	Total	Minority	Majority	Features	IR
1	WDBC	569	212	357	30	1.68
2	Libras movement	360	120	240	90	2
3	Accent	329	29	300	12	10.34
4	Seeds	210	70	140	7	2
5	Speech features	756	192	564	752	2.93
6	Musk1	476	207	269	166	1.29
7	Musk2	321	52	269	167	5.17
8	CHD	72	51	21	77	2.42
9	HNP	171	104	67	42	1.55
10	Vehicle0	846	199	647	18	3.52
11	Glass0123vs456	214	52	162	8	3.11
12	Newthyroid1	215	35	180	5	5.14
13	Glass4	214	13	201	9	15.46
14	Glass5	214	9	205	9	22.77
15	Shuttlec2vsc4	129	6	123	9	20.5
16	Cleveland0vs4	177	13	164	13	12.61
17	Glass06vs5	108	9	99	9	11
18	Ecoli0146vs5	260	20	240	6	13
19	Ecoli046vs5	203	20	183	7	9.15
20	Zoo3	101	5	96	16	19.2
21	Winewhite3vs7	900	20	880	11	44
22	Winewhite9vs4	168	6	162	11	27

Table 4: F_1 with different oversampling methods from real datasets.

Id	NO-RS	SMOTE	BLSMOTE	ADASYN	DBSMOTE	AWSMOTE
1	0.9691	0.9742	0.9617	0.9622	0.9740	0.9748
2	0.8820	0.9056	0.8861	0.9039	0.9130	0.9156
3	0.9559	0.9903	——	0.9952	0.9950	0.9972
4	0.9676	0.9641	——	0.9599	0.9700	0.9711
5	0.9065	0.9146	0.9393	0.9446	0.9250	0.9503
6	0.9117	0.8789	0.9181	0.9230	0.9137	0.9194
7	0.9503	0.9927	0.9859	0.9930	0.9881	0.9933
8	0.8339	0.8516	0.8962	0.9010	0.8823	0.9082
9	0.9143	0.9333	0.9391	0.9287	0.9110	0.9434
10	0.9765	0.9573	0.9724	0.9724	0.9745	0.9769
11	0.9478	0.9545	0.9543	0.9525	0.9531	0.9559
12	0.9769	0.9929	——	0.9832	0.9953	0.9944
13	0.9655	0.9870	——	0.9857	0.9871	0.9873
14	0.9761	0.9962	——	0.9951	0.9903	0.9977
15	0.9622	0.9769	——	0.9995	0.9987	1
16	0.9566	0.9939	——	0.9936	0.9939	0.9942
17	0.9537	0.9987	——	0.9985	0.9902	1
18	0.9898	0.9878	——	0.9860	0.9561	0.9935
19	0.9870	0.9797	——	0.9903	0.9886	0.9941
20	0.9756	——	——	——	0.9925	0.9960
21	0.9863	0.9800	——	0.9777	0.9908	0.9874
22	0.9705	0.9930	——	0.9941	0.9941	0.9946

datasets, namely, Musk1, Newthyroid1, and winewhite3vs7, whereas AWSMOTE is found the best in other datasets, especially in terms of Speech Features, HNP, Ecoli0146vs5, and Zoo3. AWSMOTE shows the best performance with F_1 of 0.9503, 0.9434, 0.9935, and 0.9960, which are significantly increased. For ACC in Table 5, AWSMOTE is ranked second as compared to ADASYN in terms of Speech Features, Musk1, and Musk2 and ranked after DBSMOTE in terms of winewhite3vs7. It also shows remarkable performance in terms of musk1, NHP, ecoli046vs5, and Zoo3. Based on Table 6, AWSMOTE can achieve competitive results on these real datasets in AUC, which shows that our method is closer to 1. AUC is the most commonly used evaluation metrics to measure the quality of the binary classification model, which reflects the integrity of the model. Hence, AWSMOTE is a good solution to imbalanced learning. However, AWSMOTE shows a little less impressive for Precision in Table 7, but it is still the second-best method for

Table 5: Accuracy with different oversampling methods from real datasets.

Id	NO-RS	SMOTE	BLSMOTE	ADASYN	DBSMOTE	AWSMOTE
1	0.9691	0.9747	0.9626	0.9628	0.9668	0.9767
2	0.8962	0.9243	0.8911	0.9129	0.9135	0.9261
3	0.9162	0.9904	——	0.9953	0.9949	0.9973
4	0.9571	0.9648	——	0.9626	0.9709	0.9748
5	0.8489	0.9154	0.9371	0.9536	0.9164	0.9515
6	0.8985	0.9247	0.9215	0.9229	0.9016	0.9125
7	0.9133	0.9929	0.9855	0.9940	0.9877	0.9938
8	0.7212	0.8434	0.8814	0.9027	0.8494	0.9030
9	0.8942	0.9357	0.9385	0.9318	0.8900	0.9549
10	0.9640	0.9542	0.9731	0.9729	0.9768	0.9774
11	0.9220	0.9553	0.9543	0.9525	0.9490	0.9559
12	0.9602	0.9933	——	0.9837	0.9933	0.9943
13	0.9334	0.9862	——	0.9859	0.9867	0.9874
14	0.9534	0.9960	——	0.9952	0.9897	0.9978
15	0.9274	0.9765	——	0.9993	0.9988	1
16	0.9169	0.9939	——	0.9937	0.9939	0.9942
17	0.9118	0.9987	——	0.9985	0.9891	1
18	0.9808	0.9878	——	0.9926	0.9822	0.9935
19	0.9760	0.9798	——	0.9905	0.9878	0.9944
20	0.9523	——	——	——	0.9919	0.9960
21	0.9777	0.9795	——	0.9777	0.9905	0.9876
22	0.9428	0.9927	——	0.9943	0.9932	0.9946

Table 6: AUC with different oversampling methods from real datasets.

Id	NO-RS	SMOTE	BLSMOTE	ADASYN	DBSMOTE	AWSMOTE
1	0.9616	0.9742	0.9628	0.9627	0.9586	0.9771
2	0.8325	0.9152	0.8916	0.9121	0.9139	0.9185
3	0.5391	0.9904	——	0.9953	0.9948	0.9972
4	0.9533	0.9648	——	0.9619	0.9710	0.9715
5	0.7205	0.9154	0.9372	0.9483	0.9131	0.9525
6	0.8938	0.9008	0.9215	0.9235	0.8981	0.9288
7	0.7535	0.9928	0.9853	0.9930	0.9876	0.9934
8	0.5758	0.8466	0.8660	0.9022	0.8284	0.9032
9	0.8834	0.9357	0.9387	0.9315	0.8786	0.9493
10	0.9482	0.9533	0.9731	0.9732	0.9752	0.9763
11	0.8910	0.9553	0.9544	0.9525	0.9493	0.9559
12	0.8778	0.9930	——	0.9837	0.9942	0.9944
13	0.5116	0.9872	——	0.9860	0.9870	0.9874
14	0.5000	0.9963	——	0.9952	0.9896	0.9978
15	0.5100	0.9769	——	0.9995	0.9988	1
16	0.5016	0.9939	——	0.9937	0.9939	0.9942
17	0.5150	0.9987	——	0.9985	0.9882	1
18	0.8662	0.9878	——	0.9866	0.9776	0.9934
19	0.8775	0.9798	——	0.9905	0.9875	0.9941
20	0.5000	——	——	——	0.9927	0.9960
21	0.5000	0.9799	——	0.9769	0.9906	0.9875
22	0.5000	0.9931	——	0.9942	0.9923	0.9946

Glass0123vs456 and Cleveland0vs4, whereas it has greatly improved in Speech Features, Musk1, CHD, and HNP, which is significantly better than NO-RS, SMOTE, Borderline-SMOTE, and DBSMOTE. AWSMOTE shows a good effect on the datasets of high dimension and IR, such as Speech Features, Glass 5, and Shuttlec2vsc4 in G-mean (Table 8). The higher the G-mean, the higher the accuracy of the method in predicting all classes. This means our algorithm exhibits better accuracy for both minority and majority classes, and it does not sacrifice one class over the other. This is one of the advantages of AWSMOTE in imbalanced learning.

Figure 7 shows the comparison of improvement rates of NO-RS data by using various methods under different measures, and the default value is set as 1 if the result is not calculated. It can be seen that AWSMOTE is basically better than other methods; especially F_1, Precision, and ACC show greater hoisting, and some methods, such as SMOTE on

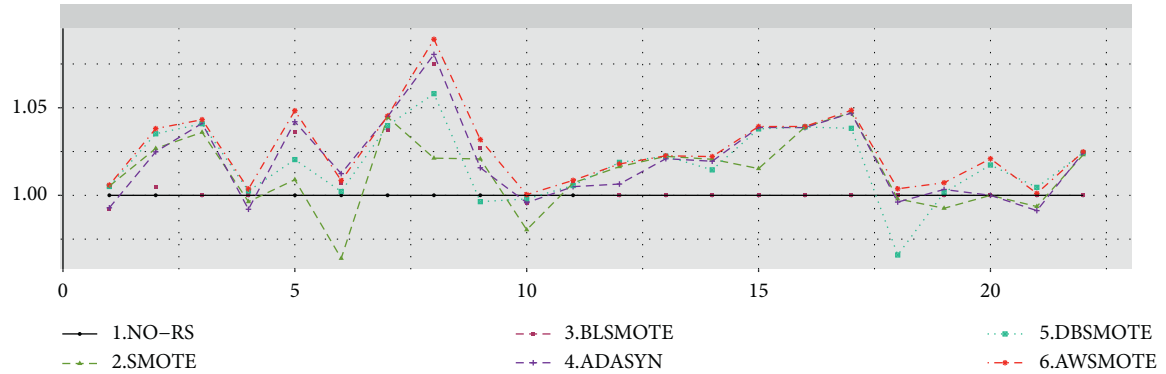
Table 7: Precision with different oversampling methods from real datasets.

Id	NO-RS	SMOTE	BLSMOTE	ADASYN	DBSMOTE	AWSMOTE
1	0.9611	0.9740	0.9876	0.9701	0.9578	0.9676
2	0.8832	0.9684	0.9353	0.9800	0.9380	0.9813
3	0.9156	0.9974	——	0.9998	0.9942	1
4	0.9717	0.9724	——	1	0.9794	1
5	0.8402	0.9240	0.9440	0.9780	0.8860	0.9825
6	0.8941	0.9397	0.9553	0.9470	0.9029	0.9674
7	0.9097	0.9998	0.9779	1	0.9860	0.9989
8	0.7177	0.8393	0.8289	0.8769	0.8196	0.9015
9	0.8953	0.9618	0.9502	0.9434	0.8916	0.9735
10	0.9752	0.9459	0.9948	1	0.9900	1
11	0.9449	0.9561	0.9786	0.9556	0.9624	0.9648
12	0.9553	1	——	0.9954	0.9948	1
13	0.9333	0.9990	——	1	0.9932	1
14	0.9534	1	——	1	0.9899	1
15	0.9273	0.9554	——	1	0.9991	1
16	0.9169	1	——	1	0.9978	0.9980
17	0.9116	0.9987	——	0.9989	0.9810	1
18	0.9800	0.9943	——	0.9973	0.9448	0.9975
19	0.9745	0.9853	——	0.9997	0.9842	1
20	0.9523	——	——	——	1	1
21	0.9777	0.9914	——	1	0.9945	1
22	0.9428	1	——	1	0.9894	1

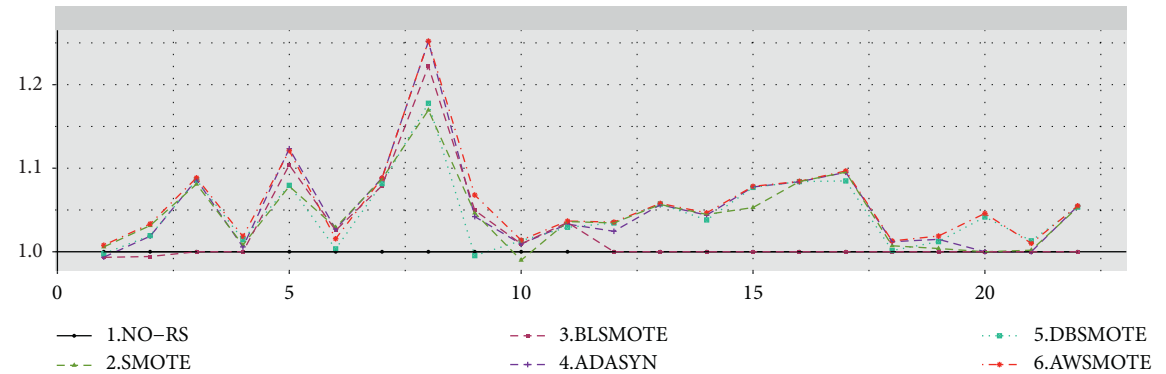
Table 8: G-mean with different oversampling methods from real datasets.

Id	NO-RS	SMOTE	BLSMOTE	ADASYN	DBSMOTE	AWSMOTE
1	0.9609	0.9700	0.9623	0.9626	0.9579	0.9717
2	0.8259	0.9117	0.8896	0.9083	0.9131	0.9173
3	0.1870	0.9904	——	0.9953	0.9948	0.9972
4	0.9526	0.9640	——	0.9609	0.9700	0.9709
5	0.6686	0.9150	0.9370	0.9475	0.9111	0.9519
6	0.8923	0.8970	0.9204	0.9228	0.8971	0.9270
7	0.7069	0.9927	0.9852	0.9930	0.9875	0.9934
8	0.2635	0.8393	0.8587	0.8985	0.8118	0.9001
9	0.8795	0.9344	0.9380	0.9305	0.8742	0.9482
10	0.9474	0.9531	0.9728	0.9728	0.9749	0.9763
11	0.8865	0.9550	0.9545	0.9522	0.9489	0.9556
12	0.8647	0.9930	——	0.9835	0.9943	0.9959
13	0.0404	0.9871	——	0.9859	0.9870	0.9874
14	0	0.9962	——	0.9952	0.9896	0.9978
15	0.0282	0.9765	——	0.9996	0.9988	1
16	0.0057	0.9939	——	0.9937	0.9939	0.9972
17	0.0424	0.9987	——	0.9985	0.9880	1
18	0.8459	0.9878	——	0.9863	0.9776	0.9935
19	0.8629	0.9796	——	0.9904	0.9874	0.9941
20	0.9523	——	——	——	0.9926	0.9959
21	0	0.9798	——	0.9774	0.9906	0.9875
22	0	0.9931	——	0.9941	0.9922	0.9946

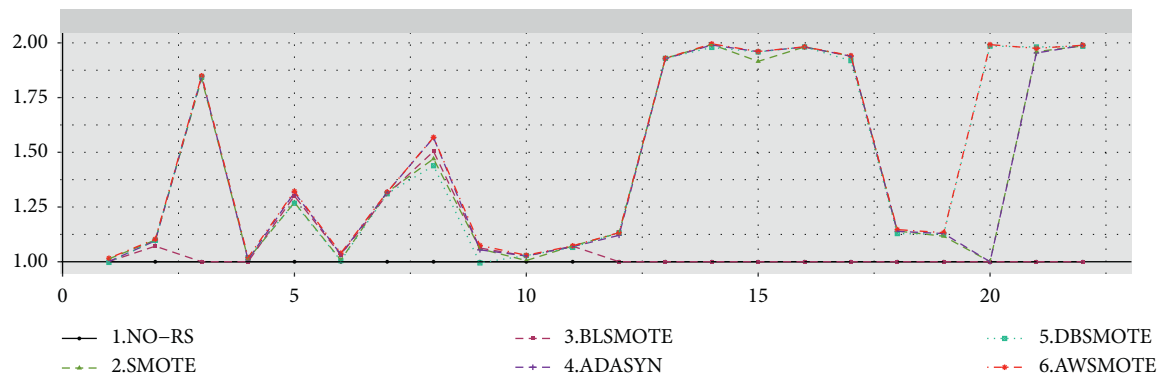
Musk1 and DBSMOTE on WDBC, show poor performance even before oversampling. For F_1 and ACC, AWSMOTE displays a prominent expression, as mentioned previously. As compared to the original datasets and other methods, AWSMOTE shows a significant improvement, which further confirms that AWSMOTE indeed exhibits excellent performance in these measures, especially in terms of Speech Features, CHD, and Ecoli0146vs5. Although AWSMOTE does not improve greatly on AUC, as shown in Figure 7(c), the greatest improvement by using AWSMOTE is nearly two times as compared to the original datasets, such as Glass 5 and Cleveland0vs4. In Figure 7(d) the result shows that AWSMOTE improves Precision, though it does not show great influence for Precision in Table 7, which still shows the strong influence on Speech Features, CHD and so on, meanwhile, the results suggest that the excellent demonstration of other methods in part datasets may be the reason for overfitting. Based on Figure 7(e), G-mean is not obvious because most original dataset scores are close to 0, and each method shows a significant increase to more than 0.9.



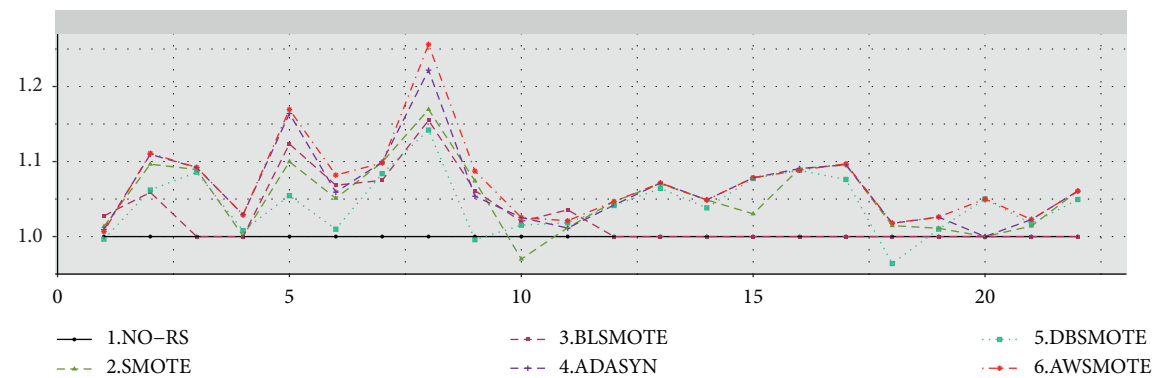
(a)



(b)

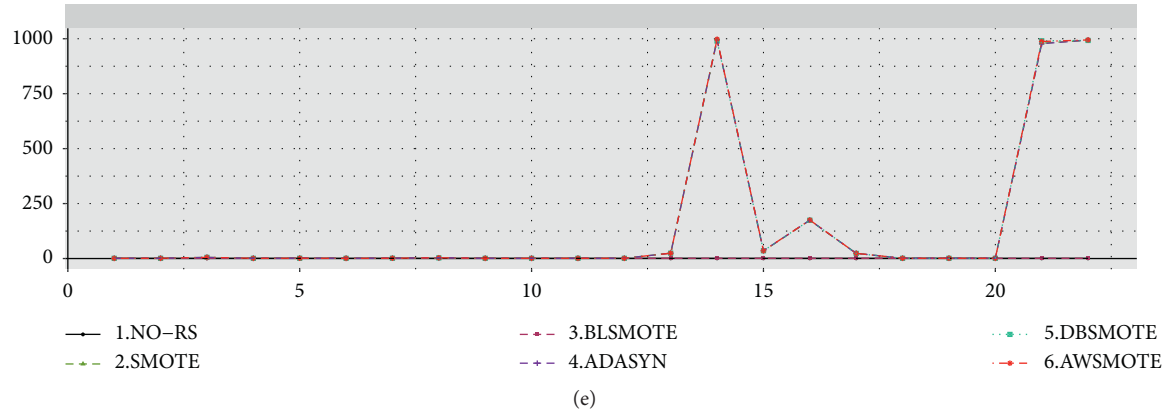


(c)



(d)

Figure 7: Continued.



(e)

Figure 7: The lifting rate diagram of each method is compared with NO-RS. (a)F_1. (b) ACC. (c) AUC. (d) Precision. (e) G-mean.

For a direct comparison to the methods, the average measures attained from the measures of NO-RS for each dataset are subtracted by the measures of other oversampling algorithms. The resulting measure improvements achieved are summarized in Figure 8. It can further be observed that all measure improvements benefit from every competitive oversampling method. The AUC avails most from the application of all methods, especially for AWSMOTE, where the maximum measure improvements of more than 0.26 are shown by all metrics. The greatest mean measure improvements are also achieved by using AWSMOTE, with the average gains ranging from 0.02 to 0.27.

Ranking the comparative methods on multiple datasets is an appropriate way to evaluate [53]. Friedman test [54] is commonly used to compare the mean ranks of different methods. The best performing algorithm defining the rank is 1, followed by 2, and so on. Averaging the ranks for these methods in the same measure, the smaller value signifies the higher rank of performance. If the null hypothesis is rejected, i.e., all methods are diverse, then we conduct a post-hoc test to find which algorithms are different from AWSMOTE. The Bonferroni-Dunn test [55] is used as the post-hoc test method. The critical difference (CD) value, as the key difference, is determined by the number of algorithms and datasets, and CD value is calculated as

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{12}$$

where k and N are the number of algorithms and datasets we adopted, respectively. In our experiments, $k = 6$ and $N = 22$, so the critical value $q_{\alpha=0.05}$ is equal to 2.5760, and the CD value is equal to 1.5240. If the difference in the mean ranks of two comparative methods is greater than the CD value, then the performance of two methods is statistically significant.

Table 9 presents the results of Friedman test on F_1, ACC, AUC, Precision, and G-mean for several methods on twenty-two datasets. In this case, the best algorithm for the performance of all metrics is AWSMOTE, especially the average rank of AUC and G-mean are both 1.0454. In terms of F_1, ACC, AUC, and G-mean, the best performing algorithm with average ranks of 1.1363, 1.2727, and 1.0454 is AWSMOTE, which is significantly better than Borderline-SMOTE and NO-RS. Table 10 presents the results of Bonferroni-Dunn test for six comparative methods, a value greater than CD (CD = 1.524) indicates statistically significant differences between the methods, which is highlighted in boldface. It shows that AWSMOTE is significantly superior to other methods.

Considering all the results, AWSMOTE significantly outperforms all other oversampling techniques in terms of measures. Every SMOTE derivative does not show a good performance on all real data because each dataset has its own characteristics. The reason BLSMOTE cannot calculate is that K exceeds the size of danger in numerous datasets. It is possible that the original data is highly imbalanced or class-overlapping may cause BLSMOTE to divide most of the minority samples as noise. It shows that BLSMOTE is seriously affected by the distribution of the original data, which causes many limitations for the algorithm. When the sample size is too small, especially when the minority class is extremely scarce, such as Zoo3, SMOTE creates more null and reduces the performance of the algorithm. In the same way, ADASYN cannot be run when the sample is very small, and it is more related to K. In the poor performance of DBSMOTE such as WDBC, Speech Features, Musk1, and HNP, it is likely that the abundance of the original data near the boundary affects the distribution of the generated samples, and it consequently leads to the severe deviation of the minority class center. So, to summarize, SMOTE and the derived methods are more or less affected by the distribution of raw data and the K value, especially when the data are highly imbalanced. These results make it difficult for the classifiers to distinguish the minority samples from the majority samples even after oversampling.

AWSMOTE is still outstanding in Speech Features, CHD, Zoo3, and so on which illustrate that it has higher generalization ability for small sample, high imbalance ratio, and dimensional space datasets because AWSMOTE is less susceptible from these effects. First, AWSMOTE does not roughly divide the samples according to the distribution of datasets, such as BLSMOTE and DBSMOTE. It uses the SVM classifier to directly differentiate support vectors and nonsupport vectors, emphasizes the importance of support

Figure 8: Mean measure improvement of the competitive methods across datasets.

Table 9: Mean ranks of six comparative methods over all datasets.

	NO-RS	SMOTE	BLSMOTE	Adas	DBSMOTE	AWSMOTE
F_1	4.5909	3.659	5.0681	3.409	3.1363	1.1363
ACC	5.0227	3.2272	5.0454	3.1136	3.3181	1.2727
AUC	5.2272	3.1136	4.909	3.1363	3.5681	1.0454
Precision	5.1363	3.1363	4.7272	2.3181	4.0681	1.6136
G-mean	5.2954	3.2045	4.8636	3.1136	3.4772	1.0454

Table 10: Bonferroni-Dunn test for six comparative methods (CD = 1.5240).

	Friedman	NO-RS	SMOTE	BLSMOTE	Adas	DBSMOTE
F_1	Reject	3.4546	2.5227	3.9318	2.2727	2
ACC	Reject	3.75	1.9545	3.7727	1.8409	2.0454
AUC	Reject	4.1818	2.0682	3.8636	2.0909	2.5227
Precision	Reject	3.5227	1.5227	3.1136	0.7045	2.4545
G-mean	Reject	4.25	2.1591	3.8182	2.0682	2.4318

vectors to the classification, which maintains the status of important minority samples, and then measures the weights of the samples accordingly. AWSMOTE considers that a minority sample is fit for oversampling if the accuracy of prediction of its generated samples is high. In this way, AWSMOTE enhances the classification performance of the classifier. Briefly, AWSMOTE decides whether to generate a new sample or not depends on the predicted result of the sample. Meanwhile, AWSMOTE retains the characteristics of the classifier by using the variable weight from SVM to make the distribution of the generated samples, which can reduce the oversampling class overlap to some extent. So, AWSMOTE is distinct from SMOTE and the derived methods generate samples between different classes and worsen the classification. In addition, these algorithms have the problem of producing more duplication and overlap of data points; thus, it cannot provide useful information for models. The way that AWSMOTE generates samples by using variable weight provides the samples more information for the classifier than other algorithms. The above

factors assist AWSMOTE to improve the classification effect and generalization ability, and it also obtains significant results by using the experiments.

5. Conclusion and Discussion

This paper illustrates the validity of considering variable weight and sample weight in the process of oversampling and proposes a new adaptive SMOTE algorithm according to different weights, AWSMOTE. The synthetic samples are generated toward the weight of variable and the distribution is determined by the weight of case. In order to reveal the performance of AWSMOTE, we compare it to other oversampling methods with six simulated datasets and twenty-two UCI and KEEL imbalanced datasets. The results show that the algorithm is effective as compared to other algorithms in the datasets, especially the instances. Based on the analyses, we conclude that AWSMOTE significantly improves the effect of classification from imbalanced datasets.

AWSMOTE is suitable for datasets with different dimensions and imbalance ratios; especially, when the sample size is small, it provides several advantages. First, the collinearity between the generated samples and the original samples can be weakened by using the variable weight. Furthermore, according to the sample weight, each sample can be adaptive in generating different numbers of samples, which reflects the different importance of samples. Finally, a method that combines these two aspects may be unstable, but AWSMOTE still shows a relatively good performance irrespective of the imbalance ratio and dimensions of the dataset.

For future work, AWSMOTE is extremely general as it can be easily extended from at least four aspects. (1) According to the principle that the larger the weight of variable is, the greater influence the variable is, when new samples are generated, the range of random number r in AWSMOTE corresponding to each generated variable is adjusted so as to weaken the collinearity. For example, the absolute value of weight is greater while $r \in (0.5, 1]$, otherwise $r \in (0, 0.5]$. (2) According to the kernel function of SVM, the importance of each support vector in classification will be further studied. (3) Different types of classifiers other than SVM can be considered, such as Partial Least Squares Regression. (4) AWSMOTE can also be extended from binary classification to multiclass problems. Therefore, many generalization algorithms can be derived from AWSMOTE. In addition, the weights of sample and variable present strengths for dealing class-imbalanced data, so finding a weighting method that depends on the characteristics of the datasets is surely worth a try.

Data Availability

The data used to support the findings of this study are available at UCI and KEEL Repository.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

JW and GF conceived the study. JW designed the algorithm and wrote the manuscript. JW, CZ, and GF performed the computation and analysis. All authors read and approved the final manuscript.

Acknowledgments

This work is financially supported by the National Natural Science Foundation of P.R. China (Grant Nos. 11761041 and 21775058). The funder played no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

References

[1] K. Zahirnia, M. Teimouri, R. Rahmani, and A. Salaq, "Diagnosis of type 2 diabetes using cost-sensitive learning," in *Proceedings of the 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, October 2015.

[2] O. E. Drummond, "Target classification with data from multiple sensors," *AeroSense*, vol. 2002, 2002.

[3] A. Onan and S. Korukoglu, "A feature selection model based on genetic rank aggregation for text sentiment classification," *Journal of Information Science*, vol. 43, no. 1, pp. 25–38, 2015.

[4] A. Onan, S. Korukoğlu, and H. Bulut, "Ensemble of keyword extraction methods and classifiers in text classification," *Expert Systems with Applications*, vol. 57, pp. 232–247, 2016.

[5] A. Onan, S. Korukoğlu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Information Processing & Management*, vol. 53, no. 4, pp. 814–833, 2017.

[6] G. H. Fu, Y. J. Wu, M. J. Zong, and J. Pan, "Hellinger distance-based stable sparse feature selection for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 21, no. 1, 2010.

[7] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[8] F. Provost and G. M. Weiss, "Learning when training data are costly: the effect of class distribution on tree induction," *Journal Of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[9] R. Blagus and L. Lusa, "Smote for high-dimensional class-imbalanced data," *Bmc Bioinformatics*, vol. 14, no. 1, pp. 1–16, 2013.

[10] W. Qiang, "A hybrid sampling svm approach to imbalanced data classification," *Abstract and Applied Analysis*, vol. 2014, no. 5, 7 pages, 2014.

[11] L. Jiang, C. Qiu, and C. Li, "A novel minority cloning technique for cost-sensitive learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 04, pp. 1–18, 2015.

[12] A. Onan and V. G. Díaz, "Consensus clustering-based undersampling approach to imbalanced learning," *Scientific Programming*, vol. 2019, no. 3, 14 pages, 2019.

[13] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[14] Z.-H. Zhou, "Ensemble learning," *Encyclopedia of Biometrics*, pp. 270–273, 2009.

[15] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[16] R. S. Y. Freund, "Experiments with a new boosting algorithm," *Technical Report*, in *Proceedings of the Thirteenth International Conference*, Bari, Italy, July 1996.

[17] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: improving prediction of the minority class in boosting," in *Proceedings of the European Conference on Knowledge Discovery in Databases: Pkdd*, Cavtat-Dubrovnik, Croatia, September 2003.

[18] M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: comparison and improvements," in *Proceedings of the IEEE International Conference on Data Mining*, San Jose, CA, USA, November 2001.

[19] N. P. Bobbili and A. M. Cretu, "Adaptive weighting with smote for learning from imbalanced datasets: a case study for traffic offence prediction," in *Proceedings of the 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, Ottawa, ON, Canada, June 2018.

[20] W. Huang, G. Song, L. Man, W. Hu, and K. Xie, *Adaptive weight optimization for classification of imbalanced data*-Springer Berlin Heidelberg, Berlin, Germany, 2013.

[21] Ş. Ertekin, "Adaptive oversampling for imbalanced data classification," *Information Sciences and Systems*, vol. 264, 2013.

[22] H. He and E. A. Garcia, "Learning from imbalanced data," *Institute of Electrical and Electronics Engineers Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[23] I. Tomek, "Two modifications of cnn," *Institute of Electrical and Electronics Engineers Transactions on Systems Man & Cybernetics SMC-6*, vol. 11, pp. 769–772, 1976.

[24] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," *Artificial Intelligence in Medicine*, vol. 2101, pp. 63–66, 2001.

[25] I. Tomek, "An experiment with the edited nearest-neighbor rule," *Institute of Electrical and Electronics Engineers Transactions on Systems Man & Cybernetics SMC-6*, vol. 6, pp. 448–452, 2007.

[26] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186, Nashville, TN, USA, December 1997.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[28] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, Hong Kong, China, June 2008.

[29] H. Han, W.-Y. Wang, B.-H. Mao, and Borderline-smote, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," *Lecture Notes in Computer Science*, in *Proceedings of the 2005 International Conference on Advances in Intelligent Computing-Volume Part I*, pp. 878–887, Hefei, China, August 2005.

[30] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Dbsmote: density-based synthetic minority over-sampling technique," *Applied Intelligence*, vol. 36, no. 3, pp. 664–684, 2012.

[31] C. Seiffert, T. M. Khoshgoftaar, and J. V. Hulse, "Hybrid sampling for imbalanced data," in *Proceedings of the IEEE International Conference on Information Reuse & Integration*, pp. 202–207, Las Vegas, NV, USA, July 2008.

[32] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *Acm Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[33] Y. Sun, F. Liu, and N. C. L. Smote, "A re-sampling method with filter for network intrusion detection," in *Proceedings of the IEEE International Conference on Computer & Communications*, pp. 1157–1161, San Francisco, CA, USA, April 2016.

[34] X. Yang, Q. Kuang, W. Zhang, and G. Zhang, "AMDO: an over-sampling technique for multi-class imbalanced problems," *Institute of Electrical and Electronics Engineers Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1672–1685, 2017.

[35] K. Li, W. Zhang, Q. Lu, and X. Fang, "An improved smote imbalanced data classification method based on support degree," in *Proceedings of the International Conference on Identification*, pp. 34–38, Beijing, China, October 2014.

[36] V. Vapnik, "An overview of statistical Learning Theory," *Institute of Electrical and Electronics Engineers Transactions on Neural Networks and Learning Systems*, vol. 10, no. 5, pp. 988–999, 1998.

[37] P. Branco, L. Torgo, and R. P. Ribeiro, "A Survey of Predictive Modeling on Imbalanced Domains," *ACM Computing Surveys*, vol. 49, no. 2, pp. 1–50, 2016.

[38] G. Wu and E. Y. Chang, "Kba: kernel boundary alignment considering imbalanced data distribution," *Institute of Electrical and Electronics Engineers Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.

[39] G. M. Fung and O. L. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, no. 1-2, pp. 77–97, 2005.

[40] S. Datta and S. Das, "Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs," *Neural Networks*, vol. 70, pp. 39–52, 2015.

[41] Y.-H. Shao, W.-J. Chen, J.-J. Zhang, Z. Wang, and N.-Y. Deng, "An efficient weighted Lagrangian twin support vector machine for imbalanced data classification," *Pattern Recognition*, vol. 47, no. 9, pp. 3158–3167, 2014.

[42] L. Gonzalez-Abril, H. Nuñez, C. Angulo, and F. Velasco, "Gsvm: an svm for handling imbalanced accuracy between classes inbi-classification problems," *Applied Soft Computing*, vol. 17, pp. 23–31, 2014.

[43] Q. Cao and S. Z. Wang, "Applying over-sampling technique based on data density and cost-sensitive svm to imbalanced learning," in *Proceedings of the International Conference on Information Management*, Shanghai, China, December 2011.

[44] B. G. R. Lloyd, "Support vector machines for classification and regression," *Analyst*, vol. 135, no. 2, pp. 230–267, 2010.

[45] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[46] R. Baezayates and B. Ribeironeto, *Modern Information Retrieval*, China Machine Press, Beijing, China, 2004.

[47] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[48] T. F. Foster Provost and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in

Proceedings of the Fifteenth International Conference on Machine Learning, Long Beach, CA, USA, June 1997.

[49] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2005.

[50] L. Li, H. He, and J. Li, "Entropy-based sampling approaches for multi-class imbalanced problems," *Institute of Electrical and Electronics Engineers Transactions on Knowledge and Data Engineering*, vol. 32, no. 11, pp. 2159–2170, 2020.

[51] H. A. Majzoub, I. Elgedawy, Y. Akaydin, and M. K. Ulukk, "Hcabsmote: a hybrid clustered affinitive borderline smote approach for imbalanced data binary classification," *Arabian Journal for Science and Engineering. Section A, Sciences*, vol. 45, no. 4, pp. 3205–3222, 2020.

[52] A. Fernández, V. López, M. Galar, M. J. Del Jesus, and F. Herrera, "Analysing the classification of imbalanced datasets with multiple classes: binarization techniques and ad-hoc approaches," *Knowledge-Based Systems*, vol. 42, pp. 97–110, 2013.

[53] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.

[54] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Publications of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1939.

[55] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.