Hindawi
Computational Intelligence and Neuroscience
Volume 2023, Article ID 4860536, 26 pages
https://doi.org/10.1155/2023/4860536

Research Article

Rebalance Weights AdaBoost-SVM Model for Imbalanced Data

Chunyu Piao, Nan Wang, and Chaofeng Yuan

School of Mathematical Sciences, Heilongjiang University, Harbin, Heilongjiang 150080, China

Correspondence should be addressed to Nan Wang; 2003137@hlju.edu.cn

Received 18 August 2022; Revised 16 November 2022; Accepted 14 December 2022; Published 19 January 2023

Academic Editor: Paolo Gastaldo

Classification of imbalanced data is a challenging task that has captured considerable interest in numerous scientific fields by virtue of the great practical value of minority accuracy. Some methods for improving generalization performance have been developed to address this classification situation. Here, we propose a cost-sensitive ensemble learning method using a support vector machine as a base learner of AdaBoost for classifying imbalanced data. Considering that the existing methods are not well studied in terms of how to precisely control the classification accuracy of the minority class, we developed a novel way to rebalance the weights of AdaBoost, and the weights influence the base learner training. This weighting strategy increases the sample weight of the misclassified minority while decreasing the sample weight of the misclassified majority until their distributions are even in each round. Furthermore, we included *P*-mean as one of the assessment markers and discussed why it is necessary. Experiments were conducted to compare the proposed and comparison 10 models on 18 datasets in terms of six different metrics. Through comprehensive experimental findings, the statistical study is performed to verify the efficacy and usability of the proposed model.

1. Introduction

Classification research is an essential field of study in data science. In balanced data classification, the support vector machine (SVM) [1] and other classification modeling approaches have been broadly discussed and used successfully in a variety of applications [2]. In specific situations where there are imbalanced datasets, the traditional methods always reach their limits in the practical application of classification [3]. They are intended to produce a model that matches the training data well; in an imbalanced dataset, this strategy ignores unusual cases. For this reason, standard classifiers generally perform poorly. With the growth of data mining and data analysis, imbalanced class learning has become a hot topic, and many academics have conducted comprehensive studies on the subject [3, 4]. They defined the nature of imbalance data categorization along three dimensions: concept complexity, training set size, and degree of imbalance between the two classes. They also show how imbalanced datasets can invalidate many conventional classifiers and offer some instances of how to tackle these difficulties. Ghosh et al. attempted to tackle the problem using deep learning systems in 2021, and they discovered that deeper architectures are useful on some data with specific structures in both artificial and real image datasets [5]. Thus far, many excellent research results and improved algorithms have been proposed to solve the problems, such as covering data streams and big data analytics, in this field. Imbalance classification problems include binary imbalance classifications and multiple imbalance classifications that can be transformed into binary problems for solution. Binary imbalance classification problems are not only frequently encountered in real life but are also interesting problems in machine learning (ML). This class of problems is characterized by the fact that the number of samples from one side of the dichotomous dataset involved, called the minority class, is smaller than that of the other side, called the majority class. This minority class is more interested in classification tasks such as medical diagnosis [6]. The identification target detects the people with diseases that belong to the minority, and the consequences of the minority being misclassified are more severe than in the reverse case. The same is valid for detecting images [7], fraud [8], managing risk [9], classifying text [10], and recognizing faces [11]. Binary classification problems with imbalanced data are prevalent compared to other issues in real life. As a basis for

classification problems, solutions to binary classification problems can be derived from other classification problems, such as multiple classification problems. Therefore, it is vital to study binary classifications in rare class classifications.

In the literature, operating on data or algorithms are the two leading solutions to the problem of imbalanced datasets [12]. The analysis of minority class structure involves determining the imbalance rate and whether it is the overlapping type of distribution; this is the critical reason why the study is complicated. The classification of extremely imbalanced datasets is even more complicated [4]. Resampling techniques for data processing have been adopted to renew the class distribution, such as sampling less of the prevalent class, sampling more of the minority, or more complex techniques [13, 14]. The most popular synthetic minority oversampling technique (SMOTE) is a simple and effective resampling method, which is also widely studied and used in combination with algorithms in the binary imbalance problem. However, resampling may undertake the risk of losing the essential information of the majority, overfitting the minority, and causing the preprocessed dataset to be unlike the raw data. Thus, in many cases, data-level methods are not studied alone but in combination with algorithm-level methods [15–17]. At the algorithmic level, they train the classifier through the data without making distribution changes. Weighting or thresholding support functions or class likelihood estimates can produce better results than resampling the data and can be applied to any conventional classifier [4]. In addition, cost-sensitive learning and ensemble schemes are popular algorithms. A series of cost-sensitive versions of SVM are proposed, for example, cost-sensitive SVM (CS-SVM), which uses SVM to extend its loss function to achieve the objective [18]. SVM based on density weight (DSVM) and improved 2-norm-based density-weighted least squares SVM (IDLSSVM) for binary class imbalanced learning problems [19]. Entropy-based fuzzy twin SVM (EFTWSVM), where fuzzy membership values are assigned based on the entropy values of samples [20]. Entropy-based fuzzy least squares SVM (EFLSSVM) and entropy-based fuzzy least squares twin SVM (EFLSTWSVM) for class imbalanced datasets [21], and other cost-sensitive algorithms [22–26]. However, an appropriate decision boundary cannot be found when the minority samples are sparse [27]. Boosting ensemble approaches overcome learning challenges from imbalanced data classes effectively [28, 29]. AdaBoost [30], the representative boosting strategy, enhances the classification performance of a model by minimizing the error probability. AdaBoost follows the output of the current classifier to modify the sample weight distribution for the next round, and it distinguishes between instances of correct classification when weights are reduced and those of misclassification when weights are increased. However, it does not make further distinctions between different classes of instances, which results in the weights of different classes of instances being increased or decreased in the same way, which is clearly an unsuitable strategy for the imbalanced classification problem. In an imbalanced problem, a good strategy of weighting is one that can distinguish between different instance categories, with which more weight can be given to those relevant instances that have high recognition importance. For this reason, researchers have developed a series of algorithms that try to adjust the weights of instances according to their category labels, which are AdaC [31], CSB1, CSB2 [32], and AdaCost [33].

SVM and other algorithms can be embedded into a boosting process and have verified exceptional capabilities [34–36]. Considering the binary classification of the data imbalance, we propose an SVM-based ensemble method that increases the focus on minority accuracy. Our approach is manipulated in two aspects:

(1) Ensure the weights of the misclassified minority and majority samples will not be changed in the same way, because we are concerned with whether the misclassified minority samples can be corrected in the next round. We propose a novel way to rebalance the instances' weights in each boosting processing, although several methods have been proposed for implementing weight updating in AdaBoost [37]. Thus, the sums of the weights of the misclassified minority and majority samples are balanced with each other.

(2) We use a different approach to vary the parameter C, a crucial parameter for the SVM algorithm, so that each sample receives different costs related to the probability of misclassification, rather than simply dividing the samples into minority and majority categories. The key to this operation is to determine cost items as a function of the weighting of the AdaBoost framework at each iteration, retraining the SVM algorithm for the current iteration by changing the cost for each sample to affect the next iteration positively. Combining these two aspects establishes a link between the AdaBoost frame's weight and the SVM-based classifier's cost items. The proposed method uses the AdaBoost weight-adjustment process to solve the data imbalance problem. The rebalanced weights assign the determination of the cost items in SVM learners at each iteration. Different SVM-based learners can be generated to improve the generalization performance based on the previous self-adjusting weights of the instances during the boosting process. Additionally, we have conducted experiments on various UCI ML repository [38] datasets. Apart from some routine evaluation indicators used for binary classification tasks, we use the P-mean to highlight the accuracy of minority classification to display the high efficiency of the presented algorithm.

The remainder of this paper is organized as follows. Section 2 summarizes the related literature on classifying imbalanced data in recent years. Section 3 introduces the background models, including the SVM model, the enhanced AdaBoost model, and AdaBoost with SVM-based and cost-sensitive SVM. Section 4 presents our method and

procedure. Section 5 presents the results and comparisons with other methods based on different datasets and metrics. Section 6 presents the conclusions.

2. Related Works

The topic of imbalanced data classification is a difficulty for all researchers, and in addition to a number of successful strategies that have been investigated, researchers are still attempting to address this obstacle using the most recent methodologies. The binary imbalanced job, as the cornerstone of the imbalanced classification problem, arises from numerous real life applications. Hazarika and Gupta presented a novel density-weighted twin SVM (DWTWSVM) for binary imbalance data classification and used density-weighted least squares twin SVM (DWLSTSVM) to boost the computational speed; then, the optimization problem is turned into solving the 2-norm of slack variables and equality constraints [39]. Additionally, using ensemble techniques to handle the imbalanced binary classifications [4]. We intend to address the imbalanced binary classification problem by altering the base classifier of the AdaBoost algorithm, for example, SVM. The SVM algorithm has been considered as one of the most effective classification methods since its introduction, and many excellent research results and improved algorithms have been proposed to solve the classification problem thus far [40]. Because of the vast potential, using SVM as component classifiers is not a new attempt. Many researchers have long focused on combining SVM and ensemble learning methods. Sun et al. [2] analyzed the AdaBoost algorithm and developed three forms of inputting cost terms into the AdaBoost algorithm framework to achieve cost-sensitive purposes. The costs marked the uneven importance of identification between classes and participated in the weight update of AdaBoost. Based on the research of Sun and Kamel, Tao et al. [41] employed cost-sensitive SVMs as base classifiers, while the normal boosting process was modified into a cost-sensitive classifier by presenting a self-adaptive method for determining the cost weight sequences of misclassification. This method allowed for adapting the various contributions of minority samples in the SVM classifier at each round according to the previous classifiers obtained. In this way, different classifiers were generated, thus improving the generalization performance. Lee et al. [28] introduced a weight adjustment factor mechanism of weighted SVM, which was used as a weak learner, and it was proposed for the imbalance data classification target. Instances were classified into four categories: bounded support vector (BSV), support vector (SV), positive noise, and others based on location. They gave different adjustment factors for BSV, SV, and positive noisy instances. In the process of learning a weighted SVM, the weights of instances in the AdaBoost algorithm were multiplied based on the adjustment factors. Wang and Sun [42] proposed an alternative method to improve AdaBoost based on the AD AdaBoost [43] algorithm. The imbalanced ratio of data was a factor and was defined as $b = N_p/N_n$, the majority, and the minority size ratio. In addition to the implementation of the parameters C and the weights, there was also the implementation of the SVM-based ensemble algorithm by changing σ. Li et al. [44] proposed an AdaBoost-SVM method in which the sequence of trained radial basis function SVM (RBFSVM) component classifiers was inserted into the AdaBoost framework. The large σ-values at the beginning were reduced as the boosting iterations proceeded. This allowed a range of RBFSVM component learners with adaptively different parameters, which would have better generalization than the AdaBoost method using SVM component classifiers with fixed σ-values.

Additionally, the SVM ensemble models' high performance has encouraged researchers to develop applications in different fields. For example, Sun et al. [16] proposed a dynamic approach that proved the efficiency of financial distress forecasting with two types of sample imbalances. This method was a forecasting method that combined the time-weighted strategy and AdaBoost with both the SVM-based integration algorithm and an oversampling technique. The results showed that the embedded integration model had significant advantages over the simple base classification model, although both the simple and embedded integration models improved the identification of rare financially distressed samples. In recent years, Liu et al. [45] presented an AdaBoost algorithm that shared SVM with a series of parameter methods to transfer the source task positive and unlabeled learning problem knowledge to the target task. The method combined the weak classifiers into a strong AdaBoost model for prediction. In addition, they considered the similarity of fuzzy examples in terms of minority and majority classes to refine the classifier's decision boundary. Yao et al. [46] solved the class imbalance problem in forecasting corporate credit risk in the supply chain context using the suggested hybrid model, which combined SVM and AdaBoost ensemble models with an artificial imbalance rate model and distinct feature selection approaches. They claimed that the proposed model mitigated the problem of class imbalance. This not only enhanced the sample distribution variety but also made the AdaBoost integration more stable and generalizable. Wei et al. [47] proposed a fault diagnosis algorithm to address the problem of poor accuracy of actuator failure identification under airplane closed-loop control. The algorithm extracted failure features using the aggregate experience model decomposition method and principal component analysis (PCA). Simultaneously, an adaptive SVM method was embedded in the AdaBoost framework to perform classification operations on them. Additionally, SVM-based ensemble methods have gained tremendous application in various areas in recent years [48–52].

3. Background Models

The following is the basic form of the binary classification model. Suppose a binary classification training dataset is given in which each sample consists of an instance and label as $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ instance $x_i \in X \subseteq R_n$, label $y_i \in Y = \{-1, +1\}$. X is the instance space, and Y is the label collection. According to the mathematical representation in

the algorithms' derivation and implementation, we used positive and negative classes to refer to minority and majority classes, respectively.

3.1. SVM Model. The SVM learning strategy maximizes the interval, which is called the margin. A wider margin corresponds to a more significant difference between the two types, making it easier for us to distinguish between them. Therefore, finding the optimal decision hyperplane corresponds to the maximum margin between the two types of samples. The SVM model designs a hyperplane with dimensions $m-1$. The hyperplane can divide the data of N samples in m dimensions into two categories. For nonlinearly separable data that can cause problems with the algorithm, two approaches can solve this issue. The first one is to enhance the low-dimensional data through a kernel function and use the SVM model in high dimensions to find the appropriate decision hyperplane. The second method introduces slack variables violating the interval constraints slightly. Soft margin SVM can be converted to optimize:

$$\begin{aligned} &\min_{w,\xi} \ \frac{1}{2}w^T w + C\sum_i \xi_i, \\ &\text{s.t. } y_i\left(w^T \cdot x_i + b\right) \geq 1 - \xi_i, \\ &\xi_i \geq 0, i = 1, 2, \ldots N. \end{aligned} \tag{1}$$

Equation (1) is generally transformed into its dual problem and then solved. After the learning problem of SVM is transformed into convex quadratic programming, it has a globally optimal solution. Several optimization algorithms exist for the fast implementation of this problem. Here, we use the sequential minimum optimization (SMO) algorithm. The essential principle behind this approach is that if all variable solutions fulfill the Karush-Kuhn-Tucker (KKT) condition of this optimization issue, the optimization issue's solution is attained because KKT is both a necessary and sufficient condition for it; otherwise, two variables are selected, additional variables are specified, and a quadratic programming problem is created for these two variables. The subproblem has two variables: one violates the KKT condition negatively, while the other is found automatically by the restrictions. In this approach, the SMO algorithm constantly decomposes and solves the original issue into subproblems. Through these, we can build a decision function using as follows:

$$f(x) = \text{sign}\left(\sum_i \lambda_i y_i K\left(x_i, x_j\right) + b^*\right). \tag{2}$$

3.2. Cost-Sensitive Support Vector Machine. Cost-sensitive classification learning approaches eschew the common classification strategy of supposing that the cost of all misclassifications is the same and then design classification algorithms that minimize the probability of error. However, in some of the aforementioned cases, this strategy is suboptimal; for example, one type of error is costlier than the others, or examples from different categories occur with different probabilities. Consequently, it is important to develop extensions of cost-sensitive techniques. Veropoulos et al. [53] presented a penalized regularized cost-sensitive SVM model to reduce the negative overwhelming impact. According to the class labels of the training data, the samples are classified into two exact classes: positive $S_+ = \{i|(x_i, y_i) \in S, y_i = 1, i = 1,..., N\}$ and negative $S_- = \{i|(x_i, y_i) \in S, y_i = -1, i = 1,..., N\}$. As the set S is divided into indices S_+ and S_-, this model also introduces C_+ and C_- penalty factors for positive and negative slack variables. In the optimization process, the positive samples retained higher penalty values than the negative samples. It implements the SVM problem as follows:

$$\begin{aligned} &\min_{w,\xi} \ \frac{1}{2}w^T w + C_+ \sum_{\{i|y_i=+1\}}^{N} \xi_i + C_- \sum_{\{i|y_i=-1\}}^{N} \xi_i, \\ &\text{s.t. } y_i\left(w^T \cdot x_i + b\right) \geq 1 - \xi_i, \\ &\xi_i \geq 0, i = 1, 2, \ldots N. \end{aligned} \tag{3}$$

3.3. Enhanced AdaBoost Model. The accuracy-oriented nature of the AdaBoost algorithm prevents it from achieving the desired results if it is applied directly to the classification of imbalanced data. Thus, several researchers have made various improvements to the AdaBoost framework. The enhanced AdaBoost model [42] can be improved by adding the weighted voting parameters α, which are determined by the overall fault rate and the accuracy of the minority primary interest. K_m is the sum of all positive samples' weights, and P_m is the sum of the sample weights labeled positive and predicted to be positive. The ratio of these two is denoted as γ_m.

$$\begin{aligned} K_m &= \sum_{i:y_i=1} \omega_{mi}, \\ P_m &= \sum_{i:y_i=1, G_m\left(x_i\right)=1} \omega_{mi}, \\ \gamma_m &= \frac{P_m}{K_m}. \end{aligned} \tag{4}$$

After initializing the sample weight D_1, we find the base classifier $G_m(x)$ to minimize the error:

$$e_m = \sum_{i:G_m\left(x_i\right)\neq y_i} \omega_{mi}. \tag{5}$$

We repeat the computation of the weak classifier weight α_m:

$$\alpha_m = \frac{1}{2}\ln\left\{\frac{(1-e_m)}{e_m}\right\} + k\exp\left\{\beta\left(2\gamma_m - 1\right)\right\}. \tag{6}$$

Then, we renormalize $\omega_{m+1,i}$ until m reaches M, the previous setting iterations. The parameters k and β in the

above expression are crucial to ensure that the enhanced AdaBoost boosts the classifying efficiency of S_+ and maintains a low global error rate. The final output $G(x)$ is a linear combination of a series of weak classifiers.

3.4. AdaBoost with SVM-Based. In imbalance classification problems, the generalization performed by the SVM-based AdaBoost method is superior to that of a single SVM [44]. In this section, we describe the SVM-based AdaBoost model. The AdaBoost model is a forward stepwise additive model consisting of a basic classifier whose loss function is exponential $L(y, f(x)) = \exp[-y f(x)]$. $f_{m-1}(x)$ is the first $m-1$ base classifier. The α_m^* and G_m^* that minimize (7) are the α_m and $G_m(x)$ obtained by the AdaBoost algorithm:

$$(\alpha_m, G_m(x)) = \min_{\alpha,G} \sum_i \exp\left[-y_i\left(f_{m-1}(x_i) + \alpha G(x_i)\right)\right]. \tag{7}$$

The algorithm learning model is equivalent to the final classifier of AdaBoost when the base classifier is $G_m(x)$:

$$G(x) = \sum_m \alpha_m G_m(x). \tag{8}$$

Because multiple parameters are involved in both SVM and AdaBoost algorithms, there are various combinations of SVM with ensemble learning. These include the cost-sensitive AdaBoost algorithm [2, 31] and AdaC2, which have shown good and relatively stable performance [2]. AdaBoost with a heterogeneous SVM could also work well [54]. The following is a brief and concise example of an SVM cost-sensitive ensemble based on adaptive cost weights [41], a method that adaptively considers the various contributions of positives to the SVM classifier during boosting based on the previously obtained classifiers. This study has done more work on positive class samples because of their greater importance, giving a higher cost value to positive instances that are misclassified than to all correctly classified positive instances. This approach has allowed for a great deal of work on instance placement to be completed, assigning larger cost values to borderline instances rather than instances far from the boundary in all cases where it does not matter whether the positives are classified correctly. By incorporating the costs into the updated weights, the weights of positives with higher costs further increase when they are misclassified; otherwise, they further decrease. Initialize $D_1 = C_+/Z_0$ for all positive instances, and $D_1 = C_-/Z_0$ for all negative instances, $Z_0 = p*C_+ + n*C_-$, where p and n are the number of the positive and negative classes. C_- defaults to one. We calculate the weight updating parameter as follows:

$$\alpha_m = \frac{1}{2}\ln\frac{\sum_{i:\ y_i = G_m(x_i)} C_{m,i}\omega_{m,i}}{\sum_{i:\ y_i \neq G_m(x_i)} C_{m,i}\omega_{m,i}}. \tag{9}$$

Further, we update and normalize sample weights:

$$\omega_{m+1,i} = \frac{C_{mi}\omega_{mi}\exp\left(-\alpha_m y_i G_m(x_i)\right)}{Z_m}, \tag{10}$$

$$C_{mi} = \begin{cases} C_+ \cdot \left(1 + \dfrac{1}{1+\exp\left(g_{m-1}(x_i)\right)}\right), & \text{if } y_i = 1, \\ C_-, & \text{if } y_i = -1. \end{cases} \tag{11}$$

In $G_m(x_i) = \text{sign}(g_m(x_i))$, where $g(x_i)$ is the value associated with x_i calculated by the decision function of SVM and Z_m represents normalization value, the final classifier is $G(x) = \text{sign}(\sum \alpha_m \bullet G_m(x_i))$.

4. Proposed SVM-Based AdaBoost Ensemble

We use SVM as a fundamental weak learner and extend the weight design of the AdaBoost framework to cost-sensitive classification problems. This extension means rebalancing the sum of positive and negative sample weights that are misclassified. A cost-sensitive ensemble classification algorithm is derived in which the weights of misclassified samples from the positive class are added, and the weights of misclassified samples classified correctly from the prevalent class are reduced in our approach. This guarantees that more weights are cumulated in the positive class to influence the training. Moreover, the update of the sample cost vector by the SVM learner is indirectly determined by the associated weight term. With this strategy, the SVM classifier can adaptively consider the different contributions of each instance in each iteration based on the previous boosting process. Instead of focusing too much on samples from the positive class as in other cost-sensitive class algorithms, our algorithm assigns a higher cost value to all misclassified cases during the SVM training process and further handles misclassified samples from the positive class in the rebalancing phase. This allows our algorithm to not only impact significantly on forming the classification but also to address the cost-sensitivity problem. In this section, we detail the two improvements we made to AdaBoost with SVM and the theoretical justification for each part, giving the algorithmic description at the end. Figure 1 shows the procedure of the proposed approach. The specific procedure and detailed computational equations for solving our dual problem are shown in Algorithm 1.

4.1. Cost Weights AdaBoost-SVM Model. One way to combat the problem of skewed datasets is to work on the penalty factor, which is to give a larger penalty factor to positive classes with small sample sizes, indicating that we value this part of the sample. The penalty factor C is not a variable, and the whole optimization problem is solved with a value of C that you must specify beforehand. After specifying this value, you can obtain a classifier and then evaluate it with the test

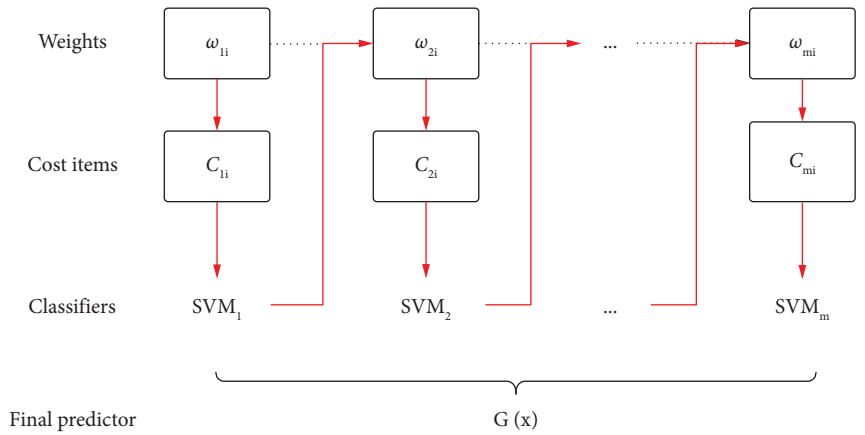

Figure 1: The procedure of the proposed approach.

(1) Input: Training samples $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$.
(2) Initialize the $D_1 = (\omega_{11}, \ldots, \omega_{1i}, \ldots, \omega_{1N})$ with $\omega_{1i} = 1/N$, and the $C_1 = (C_{11}, \ldots, C_{1i}, \ldots, C_{1N})$.
(3) For $m = 1, 2, \ldots, M$.

(a) Using C_m to obtain an SVM classifier $G_m(x)$ on the training dataset.
(b) Calculating the coefficient α_m according to equation (15).
(c) Updating rebalanced D_{m+1} using equations (16), (17).
(d) Updating the value of C_{m+1} using equation (14).

(4) Building linear combinations of basic classifiers.

$$f(x) = \sum_m \alpha_m G_m(x)$$

(5) Output: The final classifier.

$$G(x) = \operatorname{sign}(f(x)) = \operatorname{sign}\left(\sum_m \alpha_m G_m(x)\right)$$

Algorithm 1: Proposed SVM-based ensemble algorithm.

data. If the result is not satisfactory, change the value of C and repeat the process. This is a parameter search process; however, it is unlike the optimization problem itself. Here, we determine C automatically by the updated weight values for a reason, not by random guessing.

Compared to cost-sensitive SVM, we assign a different cost to every misclassified instance instead of one in distinct classes. To solve the problem efficiently and to apply the kernel technique more conveniently, we convert the primary problem into its dual problem and then solve it. The Veropoulos model of the soft margin SVM prototyping [1, 55] is developed as follows:

$$\begin{aligned} &\min_{w,\xi} \ \frac{1}{2} w^T w + C^T \xi, \\ &\text{s.t. } y_i\left(w^T \cdot x_i + b\right) \geq 1 - \xi_i, \\ &\xi_i \geq 0, i = 1, 2, \cdots N. \end{aligned} \tag{12}$$

C is a column vector, $C = [C_1, C_2 \ldots C_N]^T$. ξ denotes a vector of corresponding slack variables of the training data $\xi = [\xi_1, \xi_2 \ldots \xi_N]^T$. We take the Gaussian kernel function, and the dual and kernelized formulation can be derived as follows:

$$\begin{aligned} &\min_{\lambda} \ \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j K\left(x_i, x_j\right) - \sum_i \lambda_i, \\ &\text{s.t. } \sum_i \lambda_i y_i = 0, \\ &0 \leq \lambda_i \leq C_i, i = 1, 2, \cdots N. \end{aligned} \tag{13}$$

To find the optimal solution $\lambda^* = (\lambda_1^*, \lambda_2^*, \ldots, \lambda_N^*)$, we choose a positive component $0 \leq \lambda_j \leq C_j$ of λ^* and compute $b^* = y_j - \sum \lambda_i^* y_i K(x_i, x_j)$ to construct a decision function $f(x) = \operatorname{sign}(\sum \lambda_i^* y_i K(x_i, x_j) + b^*)$.

We divide all the classified samples into three types. In the collection of the first sample type $M1 = \{i | G_m(x_i)\ y_i = 1\}$, samples in this set type are all correctly classified and labeled as i_{M1}. In the collection of the second sample type $M2 = \{\, i \mid G_m(x_i) = -1,\ y_i = 1\}$, the positive samples are classified incorrectly as negative, and we label the weight of this type of sample as i_{M2}. These samples are concerning, as we consider positives classified correctly as a superior task, and we need to rebalance these instances. In the collection of the third sample type $M3 = \{i | G_m(x_i) = 1, y_i = -1\}$, the samples in the third set are classified incorrectly as negative samples into the positive. We

consider them less important than the second type of misclassification; therefore, they would not be weighed again. We denote these samples as i_{M3}. $M3$ and $M2$ are all misclassified samples; $M1$, $M2$, and $M3$ are combined into a complete set. Instead of using a parameter constant C in the standard SVM to control the maximum hyperplane interval in the objective function while ensuring the minimum deviation of instances, we would give a parameter vector C so that each slack variable ξ_i has a weight C_i. We use a different C for each outlier, which means we value each sample differently. We assign a smaller C to those samples that are inconsequential compared with those instances that are not to be misclassified. We update the cost terms continuously as the weights change during the boosting process. This cost weights the AdaBoostSVM model we call Ada-SVM. The cost item of the i-th sample at the $(m+1)$-th round is applied as follows:

$$C_{m+1,i} = \begin{cases} C_{mi}, & \text{if } i \in M1, \\ C_{mi} \cdot \left(1 + N \cdot \omega_{m+1,i}\right), & \text{if } i \in M2 \cup M3. \end{cases} \tag{14}$$

4.2. Rebalance Weights Model for Imbalanced Data. Unlike other cost-sensitive classification algorithms, we construct a new computational rule to assign weights to the instances based on the weight adjustment of the original AdaBoost framework. Here, $G_m(x_i)$ is the category of the i-th sample predicted by the base learner SVM at the m-th iteration. α_m is the coefficient of the m-th base classifier, and ω_{mi} is the weight of the i-th sample at the m-th iteration. (15) is used to calculate α_m.

$$\alpha_m = \frac{1}{2} \ln \frac{\sum_{\{i|M1\}} \omega_{mi}}{\sum_{\{i|M2+M3\}} \omega_{mi}}. \tag{15}$$

At the $(m+1)$-th iteration, the weight of misclassified positive and negative instances is formulated as follows:

$$\omega_{m+1,i_{M2}} = b \cdot \frac{\omega_{mi_{M2}}}{Z_m} \exp\left(-\alpha_m y_i G_m(x_i)\right) (m \geq 2). \tag{16}$$

The weight of other samples is formulated as follows:

$$\omega_{m+1,i_{M2+M3}} = \frac{\omega_{mi_{M2+M3}}}{Z_m} \exp\left(-\alpha_m y_i G_m(x_i)\right), \tag{17}$$

where

$$b = \frac{\sum_{\{i|M3\}} \omega_{mi}}{\sum_{\{i|M2\}} \omega_{mi}} \ (m \geq 2), \tag{18}$$

$$\begin{aligned} Z_m = &\sum_{\{i|M2\}} b \cdot \omega_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right) \\ &+ \sum_{\{i|M2+M3\}} \omega_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right), \end{aligned} \tag{19}$$

$$D_{m+1} = \left(\omega_{m+1,1}, \cdots, \omega_{m+1,i}, \cdots, \omega_{m+1,N}\right). \tag{20}$$

Theorem 1. *The training error bound for the final classifier of the rebalancing AdaBoost is as follows:*

$$\frac{1}{N} \sum_i I\left(G(x_i) \neq y_i\right) \leq \frac{1}{N} \sum_i \exp\left(-y_i f(x_i)\right) \leq \prod_m Z_m. \tag{21}$$

The proof is shown in Appendix A.

Theorem 2. *The training error bound for the binary classification problem rebalancing AdaBoost:*

$$\frac{1}{N} \sum_i I\left(G(x_i) \neq y_i\right) \leq \left(1 + \frac{b}{2}\right)^M \exp\left(-2M\gamma^2\right), \tag{22}$$

where $\gamma = 1/2 - e_m$.

The proof is shown in Appendix B.

5. Experimental

Our cost-sensitive ensemble learning method was driven by the goal of obtaining all positive samples classified correctly and dealing effectively with imbalanced datasets. We illustrate the effectiveness of our method for the imbalanced data classification problem by using experimental data. The proposed method was compared with ten other approaches in six metric dimensions on our selected dataset with different imbalance ratios. Because our method is based on changing the AdaBoost base classifier to SVM, we compare our method with the original AdaBoost and SVM algorithms without any tricks, respectively, to show that combining the two methods makes our method better than both of them. To compare the performance with other improved algorithm-level classification methods, SVM-based cost-sensitive methods are chosen, and these included the prorated cost method of using the inverse of the size of positives and negatives as the penalty constants for the different classes, CS-SVM, and SMOTE + SVM. We also compare our approach with other state-of-the-art ensemble and data-level resampling strategies, namely, Easy Ensemble [56], SMOTEBoost [57], and SMOTEBagging [57]. Ada-SVM is used to prove that the rebalancing trick is working. The decision tree algorithm was used as a comparison algorithm to illustrate that our approach improved the recall for the positive samples and was not traded off by sacrificing other metrics.

5.1. Description of Datasets. Fourteen datasets with different numbers of attributes and sample sizes were selected from the UCI for the test to assess the behavior of the suggested method in handling classification tasks of imbalanced datasets There were some missing attribute values in several datasets, and KNN handled missing attribute value processing. Table 1 lists a detailed description of the dataset used. All datasets had two output labels, denoting the positive and negative categories. The attributes indicate the size of features in datasets. Imbalanced Ratio (IR) is the ratio of the number of samples in the negative class to the number of samples in the positive class. Generally, the larger the IR value is, the more harmful it is to the performance of traditional classifiers. We selected some datasets with IR in the range of 1 to 50 for our experiments. We believe that the ratio of the number of samples

Table 1: Description of the datasets.

Datasets	Attributes	Instances	IR	Instances: attributes
Ecoli_cp	7	336	1	48
WDBC	30	569	2	19
Breast cancer	9	699	2	78
Ionosphere	34	351	2	10
Wine3	13	178	3	14
WPBC	33	194	3	6
Ecoli_im	7	336	3	48
Hepatitis	18	154	4	9
SPECT	22	267	4	12
Teaching	5	151	4	30
German credit	24	1000	5	42
Ecoli_pp	7	336	5	48
Segmentation	19	2100	6	111
Glass6	9	214	6	24
Ecoli_imU	7	336	9	48
Ecoli_om	7	336	16	48
Yeast5	8	1484	33	186
Pageblock	10	5473	9	547

to the number of features may also be an influential factor in the classification of imbalanced data, and we chose datasets with this ratio in the range of 6 to 547 for our experiments. The training set was normalized before training by rescaling each feature to homoskedasticity for SVM-based algorithms, and the rescaled features that did not distort the original distribution were used on the testing dataset. Preprocessing eliminated the numerical difference between each feature X; further, controlling the size of each column of feature X within a specific range made the model prediction performance more accurate.

5.2. Evaluation Metrics. Specific evaluation metrics to observe the model's performance in each category were introduced to evaluate the classifier's classification performance.

5.2.1. Confusion Matrix. The confusion matrix in Table 2 shows how the classification model made mistakes when making predictions. All cases were recorded in the following four categories: TP, FN, FP, and TN.

5.2.2. Accuracy, Precision, Recall, and Specificity. Classification accuracy was the index for evaluating the classifier's performance. This indicator was suitable for datasets with balanced categories. For lopsided datasets, the accuracy became unreliable. If positive examples: negative examples = 1 : 99, then the classifier would incorrectly predict all positive examples as negative. An accuracy rate of 0.99 was possible; however, this model could not identify positive examples. Precision, recall, and specificity are metrics generally used for binary classification problems.

Table 2: Confusion matrix.

	Predicted positive	Predicted negative
True positive	TP	FN
True negative	FP	TN

$$\begin{aligned} \text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}, \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}}. \end{aligned} \tag{23}$$

5.2.3. Receiver Operating Characteristic (ROC) and Area under the ROC Curve (AUC), G-Mean, F1-Score, and P -Mean. The ROC curve was a recurrent evaluation index for the two-class classification problem; it visually compared the performance of different models on the same dataset. When comparing the performance of the two models, if the ROC of one model completely wrapped the ROC of the other, the former was superior to the latter in terms of classification performance. The use of the area under the ROC curve (AUC) can afford a better model. AUC inherited the insensitivity as a scalar of the ROC curve and was often used in these classifications [58].

In the imbalanced classification problem, it was not comprehensive to consider any indicator alone; we needed to combine metrics to measure the model's efficiency. The G-mean [7] indicator considered the accuracy of both the positive and negative samples. It was different from the overall accuracy and avoided the dominant influence of negative samples on the classification performance. The $F1$-

score value allowed us to focus on small outliers as the harmonic mean of a set of numbers and was biased toward the smallest element in the list. We paid attention to positive samples and metrics biased toward positive samples, such as the recall of positive samples, when comparing the results and evaluating an algorithm. $F1$-scores were occasionally unreliable because they were influenced by the minimum value rather than the maximum value. Hence, we used the P -mean, and the geometric mean belonged to recall and precision, which was considered positive and were not so biased toward the positive as the $F1$-score [7]. The reason for using P-mean was to examine how well the algorithm performed in classifying positive samples; this aspect has not been widely studied. We considered the P-mean an essential index to discriminate whether the classification of positive samples was sufficient. Given the increasing emphasis on recall, P-mean reconciled the values of precision and recall, evaluating the model's performance more comprehensively.

Figure 2 shows that in some circumstances, the usage of P-mean is required. The Hepatitis dataset is stratified sampled and divided into two sections, with 80% of the data serving as the training set and 20% serving as the testing set. Multiple trials were run with our model and the SMOTEBoost model, and the results were displayed as box plots, with the performance of the two models under the $F1$-score and P-mean assessment criteria highlighted in magenta and blue, respectively. Evidently, the median of our proposed method is slightly lower than that of SMOTEBoost under the $F1$-score evaluation criterion but significantly higher than that of SMOTEBoost under the P-mean evaluation criterion, and in fact, our method outperforms SMOTEBoost in the goal of increasing the accuracy of small class samples. Under all assessment criteria, the data findings provided by our technique are clearly more focused than those obtained by the SMOTEBoost method, reflecting the improved stability of our suggested classification model.

$$\begin{aligned} \text{G} - \text{mean} &= \sqrt{\text{recall} \cdot \text{specificity}}, \\ \text{F1} - \text{score} &= \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}, \\ \text{P} - \text{mean} &= \sqrt{\text{recall} \cdot \text{precision}}. \end{aligned} \tag{24}$$

5.3. Experimental Settings. All methods were compared with the proposed approach to produce a comprehensive evaluation. The data findings depicted by the icons are all averages of the testing data following a five-fold cross-validation. To avoid the effect caused by the number of two-class instances for training and eliminate randomness before training, we adopted stratified sampling to ensure that the samples after the five-folds split had the same imbalance ratio as the entire sample set, and then four-folds are used as training data with the remaining one-fold as the testing data. Cross-validation may result in a skewed test set in severe circumstances of data imbalance, leading to an inaccurate evaluation [5]. As a result, we also carried out balanced testing, with all assessments based on five-fold stratified cross-validation. We consider the data with an IR higher than or equal to five to be highly imbalanced.

For a more comprehensive comparison of our method with other methods, we ran two independent groups of tests [3, 4]: (1) 17 datasets to compare our method with other methods in terms of inquiry structure of classes and sample-to-feature ratio, and (2) we derived 12 new datasets, each with 1000 examples, from a random undersampling in the Pageblock dataset with different IR , aiming to explore the effect of IR on our method compared with others while maintaining the remaining conditions. We examine the same IR levels with the datasets in the first group and higher IRs of 36, 40, and 50 at the same data size.

In the proposed SVM-based AdaBoost ensemble, three parameters had been prespecified: the parameter σ for the Gaussian kernel, the initial penalty vector C for each instance, and T for the number of iterations. We employed grid-search to predefine the Gaussian width parameters for the SVM classifier to avoid implications of parameters on the performance. We use one as the σ value for all datasets, and for others, including the Ecoli, Breast Cancer, Ionosphere, Teaching, and Pageblock datasets, we use 10.

The optimization problem was solved in such a way that C was always a fixed value. The cost factor C in cost-sensitive learning was determined for a specific reason, such as using the ratio of the number of samples between categories as the cost, similarly to the prorated cost method. Under some circumstances, the relevance of distinguishing distinct samples was described by cost items, and the cost of a specific sample relied on the properties of the unique situation. For example, regarding detecting fraud, the cost of missing a specific fraud case was determined by the amount of money involved [8]. Herein, we set the initial value to one, and the adjustment scale was updated by itself according to the changes in the iteration process.

The experiments of datasets presented in Table 1 were conducted to investigate the influence of the iteration parameter T on the performance of the proposed approach. This was a critical factor for improving classification performance. The G-mean, $F1$-score, P-mean, AUC, accuracy, and recall in the panels in Figure 3 demonstrate that the algorithm was convergent and typically had stability after T reached 25 under settled σ. Therefore, we set the parameter T for all ensemble learning to a constant value of 25.

To ensure consistency across experiments, all classification algorithms involving the SVM parts are written in uniform handwritten code. Decision tree, easy ensemble, SMOTEBoost, SMOTEBagging, SMOTE + SVM, and AdaBoost, which make direct calls to the packages in Python, are algorithms that use default settings. Because the base classifier SVM has uncertainties in the selection of support vectors, the results would be slightly different for each run of our model. The implementation of the proposed method is publicly available in a GitHub repository (https://github.com/PChunyu/SVM-Adaboost-C). The computer configuration used to run all methods in this paper is an Intel (R) Core (TM) i5-7500 CPU @ 3.40 Hz and 3.40 GHz.

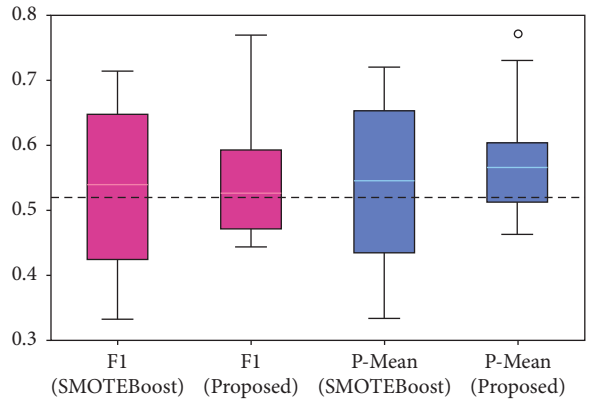

Figure 2: Comparison between SMOTEBoost and the proposed method for hepatitis classification results on $F1$-score and P-mean metrics.

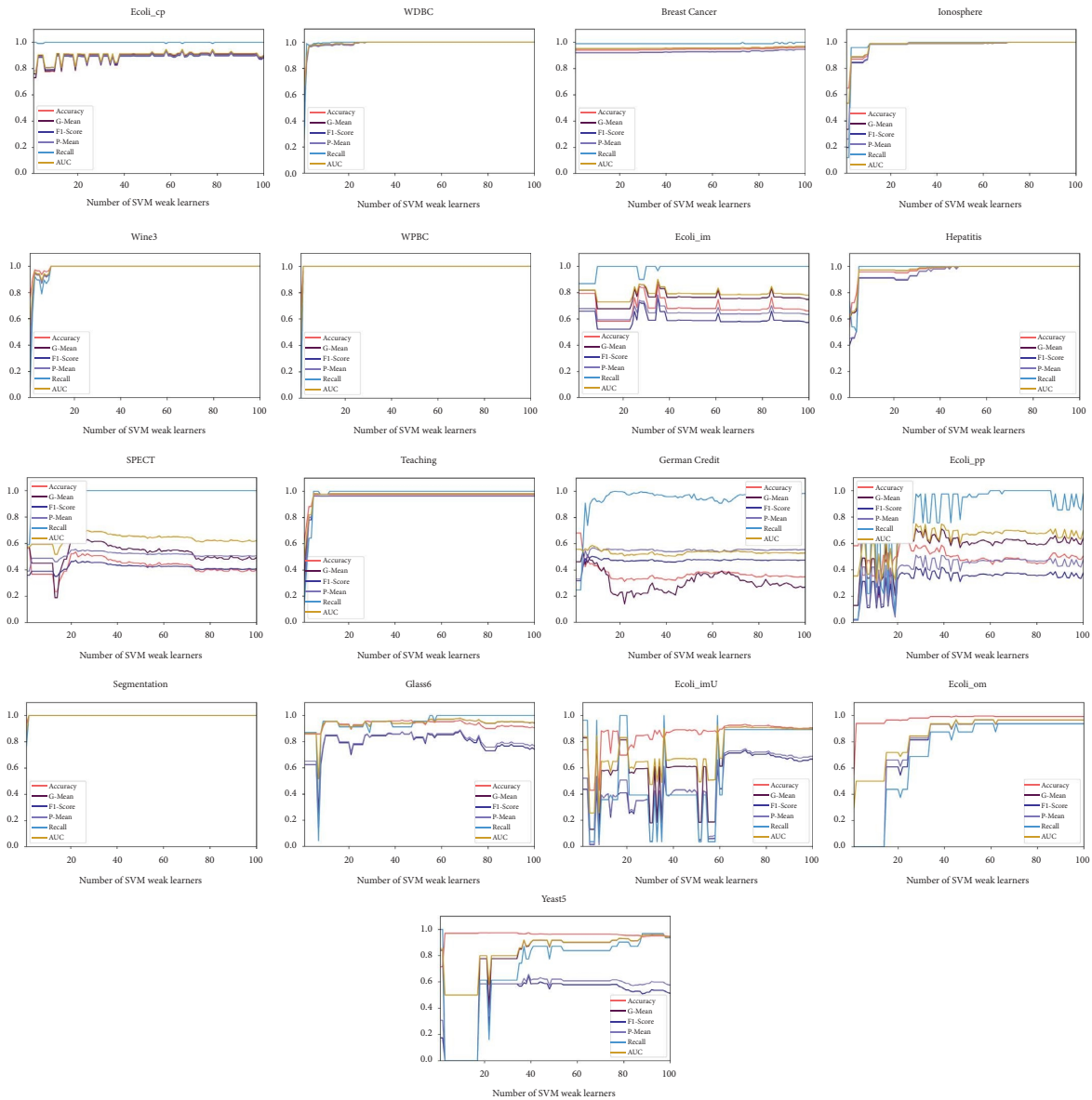

Figure 3: The trend of evaluation metrics of the training dataset with T.

5.4. Experimental Results and Statistical Tests. In this section, we first show the effectiveness of our proposed rebalancing strategy and the proposed model. We present the experimental results of two groups in three dimensions: structure of classes, sample-to-feature ratio, and IR. At last, we performed hypothesis tests on all metrics between the proposed method and others to illustrate the validity of our model.

To confirm the validity of our proposed rebalancing weights approach, we compared the classification performance of Ada-SVM and proposed, and their differences are before and after rebalancing weights. Figure 4 shows the trend for accuracy and recall. Our approach's performance was superior to that of Ada-SVM in terms of recall. After almost 20 iterations, the proposed method stabilized the recall rate at one, and the accuracy rate was the same or lower than Ada-SVM. Our approach improved the recall at the expense of accuracy in some datasets, an inevitable consequence of enhancing the recall rate. The rebalancing scheme contributed to the proposed method's better generalization performance than that of Ada-SVM in terms of recall.

The Yeast 5 dataset with multiple feature dimensions was projected into the two-dimensional space using PCA to observe the decision boundary and demonstrate the effectiveness of the proposed ensemble. Figure 5 shows a set of classifications using different methods. Among the classification boundaries produced by the presented techniques, the proposed model classifies the best and has the highest overall accuracy while ensuring a recall value of one. The decision boundary for simple noncost-sensitive classifiers, such as no cost SVM, AdaBoost, and SMOTE + SVM, is a plane with no convergence, and although SMOTE + SVM is used as a classification method for minority class accuracy, the cost-insensitive regular SVM classifier is still used after SMOTE generates new data. These classifiers either perform better in accuracy alone or in recall only while ignoring total accuracy. The majority of the ensemble algorithms have irregularly delineated boundaries, which can more accurately identify regions with minority samples. However, evidently, the more detailed delineation of the minority by SMOTEBoost and SMOTEBagging does not guarantee that they achieve the desired results on the test set data, whereas Easy Ensemble does. Under normal circumstances, without any adjustment, the decision boundary of the cost-sensitive classifier for lopsided data was curved toward positive [55]. CS-SVM and prorated cost panels demonstrated the decision boundary's warp toward the positive and were affected by imbalanced training data, particularly the positives. It was the reason CS-SVM and prorated cost led to poor generalization performance one testing data. Ada-SVM, No-Cost SVM, decision tree, SMOTEBoost, SMOTEBagging, and AdaBoost, regardless of accuracy, do not achieve a high recall value when compared to other techniques. The rest of the methods could correctly classify positive samples, whereas the accuracy was lower than that of the proposed method. Our method produced decision boundaries that did not favor the positive as in other cost-sensitive classification algorithms; it tended to the negative. This is because our proposed method aggravated the misclassified positive samples and improved the base classifier after readjusting the cost term for all misclassified samples. In general, the panel of the suggested approach showed the best fit among all the techniques. The maximum accuracy was achieved when the positive recall reached one.

5.4.1. First Group Datasets. Table 3 displays the experimental results of the first group; the bolded black values reflect the best values across all techniques. Except for the WDBC, Breast Cancer, Ionosphere, and Yeast5 datasets, where it does not achieve the greatest recall value, our technique has the highest recall values in all 13 datasets. On these four datasets, the difference between the recall value and the greatest recall value of our technique is not significant: 0.9297 (0.9576 for Easy Ensemble), 0.9713 (0.9834 for CS-SVM), 0.9603 (0.9923 for CS-SVM), and 0.8194 (0.9750 for Prorated Cost), respectively. It is also worth noting that our technique obtains a recall of one for both the Wine3, SPECT, and Ecoli_im datasets, as well as an average value of one after five-fold cross-validation, which indicates that the greatest recall value is obtained for each of the five cross-validation trials, demonstrating the excellent efficiency of our technique. Some algorithms also showed significant accuracy-orientedness in the experiments; the more lopsided the data, the worse the performance of the no cost, decision tree, and AdaBoost algorithms. As one of the most classic and simple methods, Easy Ensemble has the best performance in every metric on the WDBC dataset and the highest performance in the G-mean, $F1$-score, accuracy, and AUC metrics on the Ecoli_cp and Breast Cancer datasets. These algorithms in combination with SMOTE have excellent figures in other aspects except the recall metric. In some circumstances, CS-SVM obtained very impressive results as one of the representations of cost-sensitive classifiers at the algorithm level. CS-SVM, on the other hand, trained the classifier with fixed positive and negative category costs, causing it to perform better exclusively on a specific dataset, and prorated the cost as well. This restricts them to just marginally improved outcomes on specific datasets.

Although our method achieves the best level among all methods on the WPBC, Hepatitis, and German datasets, the recall levels of these datasets are overall low compared to the other datasets, where the recall values are all close to one. This has a lot to do with the structure of classes; a good classification can be obtained even using canonical classifiers from nonoverlapping distributions [4]. We selected the Ecoli_pp and German Credit datasets with consistent imbalance levels and a close sample number to feature number size ratios, as illustrated in Figure 6. After projecting the training and test set sample points of these two datasets into the two-dimensional plane separately, we can find that the two datasets differ significantly in the distribution of sample points. Both on the testing set and the training set, the distribution of positive and negative class samples on the Ecoli_pp dataset overlaps less, while the opposite is true for the German Credit dataset, which leads to the fact that all methods classify Ecoli_pp much better than German Credit on almost all metrics. Clearly, the lower the sample size to

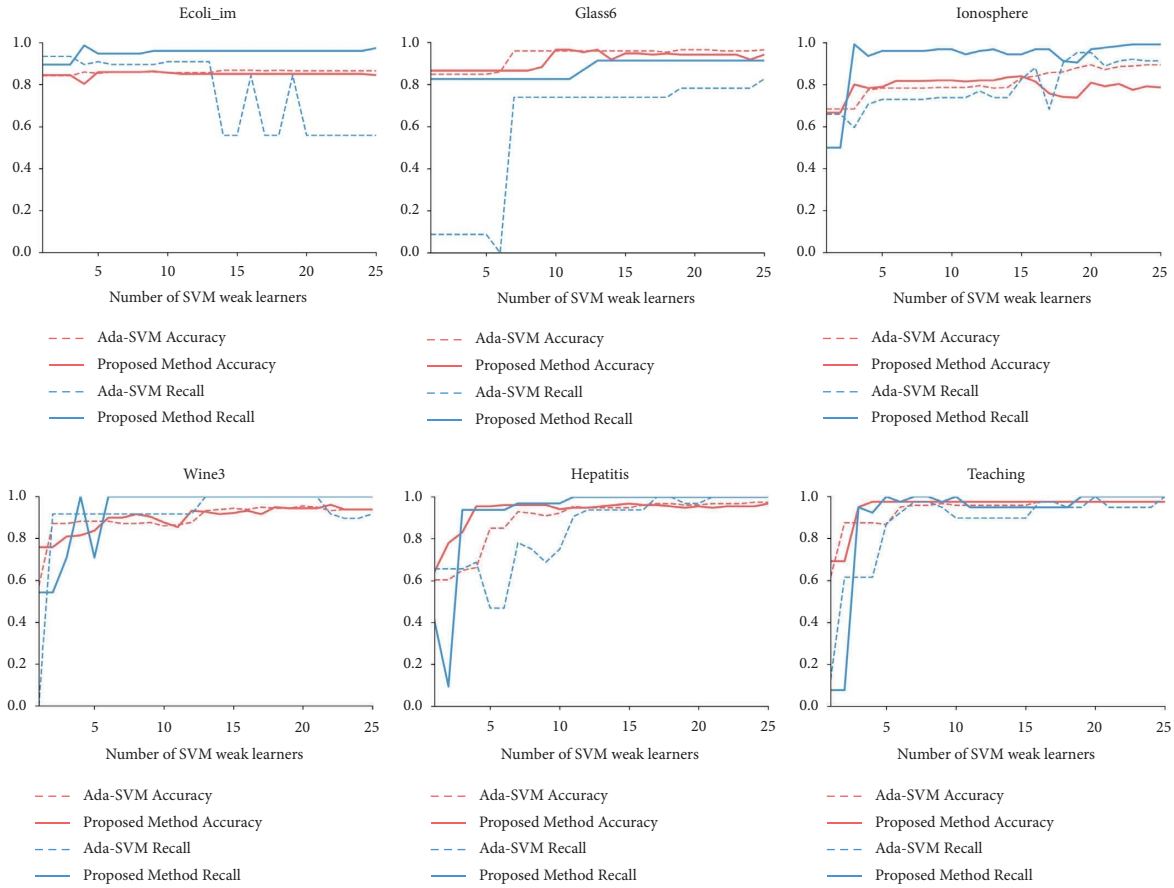

Figure 4: Trends in accuracy and recall before and after rebalancing.

feature ratio, the more features are relative to the sample size, and too many features can easily lead to overfitting; additionally, in some models, when the sample size to feature ratio is too low, the model's prediction becomes worse. However, this is not completely absolute, because it is difficult to control the other characteristics of the dataset at exactly the same level when discussing the sample number to feature number ratio. Hepatitis, for example, has a high ratio; however, because of structural overlaps, it is more impacted by the category structure, and even with a high sample-to-feature ratio, classification scores are still rather low. By comparing the data results, we found that a sample number to feature number ratio between 10 and 20 is a desirable range; for example, for the Wine3 and SPECT datasets with the highest recall, their sample number to feature number ratios are 14 and 12.

The most important is that our method obtained a considerable advantage in the positive recall at various IR. The performance of the recall results is depicted in Figure 7. The value on the bar is the sum of the all-dataset's recall of the corresponding algorithm. The proposed approach had the maximum value of recall (15.93), followed by Easy Ensemble (14.57).

Table 4 displays the Shapiro-Wilk test findings because not all data followed the normal distribution at 5%. For statistical analysis, we used the parametric T-test and the nonparametric Wilcoxon test independently to examine if there were any obvious differences between the proposed strategy and the other ways in the experiment.

The recall value of the proposed model does not only look higher than other methods by visualization; however, by statistical tests, we found that the recall value of our method is statistically significantly higher among other methods, as compared to their respective counterparts. The results are listed in Table 5. Bold indicated significant values at the 0.05 level. Statistical results demonstrated that the proposed SVM-based ensemble learning method performed better than a single no-cost standard SVM in all metrics except the AUC and accuracy. The T-test and the Wilcoxon test results showed that our model was significantly better than other models in recall performances at the 5% level, while it is not statistically significantly different from other models in terms of other metrics. However, it is significantly different from SMOTEBoost and SMOTEBagging only in terms of accuracy because our method discards accuracy to some extent in exchange for higher recall.

5.4.2. Second Group Dataset. To observe how our method performs under different IRs, Table 6 presents the second group of experimental findings. They come from a resampling of the same dataset, with only different IRs under the condition that other features are guaranteed to be the same. Our method maintains the highest level even at the IR of 36

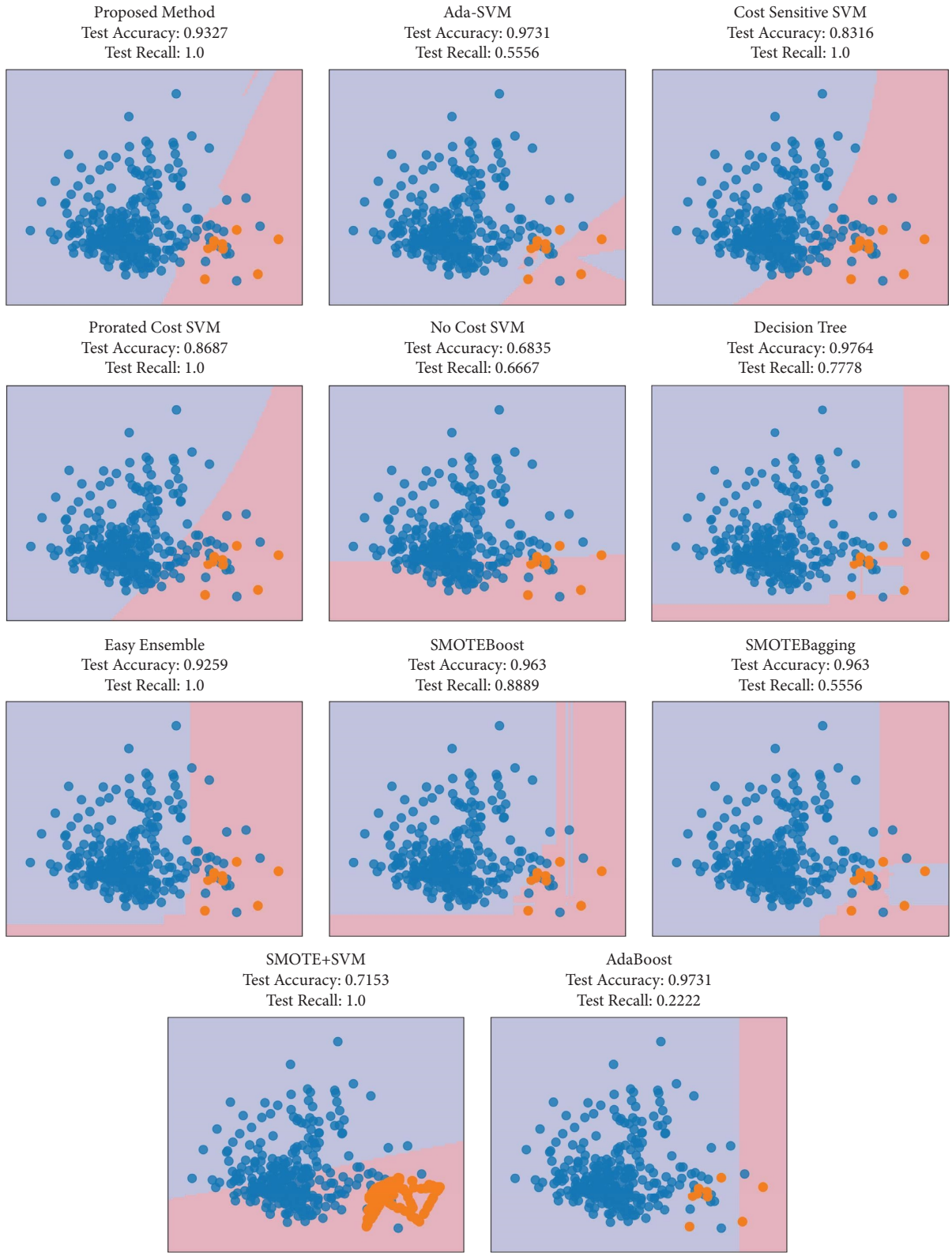

Figure 5: Demonstration of decision boundaries and test accuracy in all methods.

and 50; however, the recall value only stays at approximately 0.9.

The higher the IR of the dataset, the worse the classification performance tends to be for all methods. Our method does not perform as well on the second group datasets as first, although with the same IR as first. Overall, the classical Easy Ensemble and SMOTEBagging algorithms show a series of better performances on the Pageblock

Table 3: *G*-mean, *F*1-score, accuracy, AUC, and recall performance comparisons of different algorithms on seventeen UCI datasets.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.8195	0.8265	0.8183	0.8629	**0.9724**
		Ada-SVM	0.8162	0.8107	0.8215	0.8561	0.8956
		CS-SVM	0.6242	0.6245	0.7052	0.9505	0.8069
		Prorated cost	0.6558	0.7807	0.7409	0.9798	0.9722
		No cost	0.6833	0.7381	0.7203	0.9197	0.8828
Ecoli_cp	1	Decision tree	0.9071	0.8952	0.9137	0.9089	0.8744
		Easy ensemble	**0.9582**	**0.9517**	**0.9583**	**0.9890**	**0.9581**
		SMOTEBoost	0.9279	0.9208	0.9315	0.9738	0.9091
		SMOTEBagging	0.9503	0.9439	0.9525	0.9831	0.9372
		SMOTE + SVM	0.7854	0.8150	0.7952	0.9187	0.9073
		AdaBoost	0.9568	0.9511	**0.9583**	0.9782	0.9512
		Proposed	0.9274	0.9050	0.9279	0.9682	0.9297
		Ada-SVM	0.9132	0.8962	0.9279	0.9498	0.8771
		CS-SVM	0.7716	0.7784	0.7782	0.9757	0.9105
		Prorated cost	0.8715	0.8523	0.9069	0.9876	0.7969
		No cost	0.6744	0.6498	0.7437	0.8933	0.6082
WDBC	2	Decision tree	0.9102	0.8882	0.9175	0.9111	0.8866
		Easy ensemble	**0.9661**	**0.9576**	**0.9684**	**0.9901**	**0.9576**
		SMOTEBoost	0.9513	0.9406	0.9561	0.9885	0.9339
		SMOTEBagging	0.9551	0.9422	0.9561	0.9884	0.9527
		SMOTE + SVM	0.7531	0.7303	0.7843	0.9379	0.6750
		AdaBoost	0.5974	0.6352	0.6127	0.6666	0.6976
		Proposed	0.9523	0.9271	0.9471	0.9845	0.9713
		Ada-SVM	0.9422	0.9199	0.9428	0.9850	0.9418
		CS-SVM	0.9494	0.9191	0.9399	0.9903	**0.9834**
		Prorated cost	0.9520	0.9326	0.9528	0.9899	0.9502
		No cost	0.6300	0.5722	0.8110	0.7688	0.4639
Breast cancer	2	Decision tree	0.9157	0.8956	0.9299	0.9170	0.8754
		Easy ensemble	**0.9699**	**0.9537**	**0.9671**	**0.9937**	0.9793
		SMOTEBoost	0.9374	0.9207	0.9457	0.9886	0.9132
		SMOTEBagging	0.9533	0.9380	0.9570	0.9894	0.9419
		SMOTE + SVM	0.7181	0.7540	0.7478	0.8232	0.6992
		AdaBoost	0.9492	0.9354	0.9557	0.9903	0.9297
		Proposed	0.8813	0.8472	0.8659	0.9344	0.9603
		Ada-SVM	0.7841	0.7450	0.8803	0.9740	0.7372
		CS-SVM	0.8097	0.7635	0.7805	**0.9805**	**0.9923**
		Prorated cost	0.9216	0.8856	0.9089	0.9797	0.9760
		No cost	0.5917	0.5101	0.6266	0.7068	0.5382
Ionosphere	2	Decision tree	0.8814	0.8517	0.8946	0.8828	0.8412
		Easy ensemble	**0.9107**	0.8869	0.9174	0.9704	0.8889
		SMOTEBoost	0.9086	**0.8870**	**0.9202**	0.9515	0.8729
		SMOTEBagging	0.9024	0.8763	0.9118	0.9686	0.8735
		SMOTE + SVM	0.4384	0.4930	0.6000	0.7621	0.5111
		AdaBoost	0.8885	0.8840	0.8911	0.9596	0.8311

Table 3: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.9436	0.8801	0.9211	0.9847	**1.0000**
		Ada-SVM	0.9417	0.8790	0.9267	0.9701	0.9778
		CS-SVM	0.9741	0.9512	0.9721	0.9967	0.9800
		Prorated cost	0.9318	0.8952	0.9387	0.9885	0.9200
		No cost	0.7447	0.6279	0.8033	0.9679	0.7822
Wine3	3	Decision tree	0.9154	0.8842	0.9386	0.9185	0.8756
		Easy ensemble	0.9689	0.9421	0.9663	**1.0000**	0.9778
		SMOTEBoost	0.9705	0.9577	0.9775	0.9919	0.9578
		SMOTEBagging	**0.9808**	**0.9692**	**0.9832**	0.9992	0.9778
		SMOTE + SVM	0.8266	0.8648	0.8423	0.9657	0.9923
		AdaBoost	0.9643	0.9436	0.9667	0.9846	0.9600
		Proposed	0.5573	0.4280	0.5401	0.6737	**0.7178**
		Ada-SVM	0.4479	0.3223	0.7171	0.5989	0.3244
		CS-SVM	**0.6235**	0.4775	0.7368	**0.7439**	0.5356
		Prorated cost	0.6019	0.4656	0.6718	0.6819	0.5711
		No cost	0.4555	0.3921	0.5128	0.5500	0.7022
WPBC	3	Decision tree	0.5222	0.3623	0.6768	0.5702	0.3667
		Easy ensemble	0.5890	0.4114	0.6012	0.7046	0.5933
		SMOTEBoost	0.5905	0.4137	0.6821	0.6796	0.4733
		SMOTEBagging	0.6048	0.4909	**0.8036**	0.7113	0.4111
		SMOTE + SVM	0.3931	**0.5413**	0.5001	0.5886	0.6413
		AdaBoost	0.5130	0.3738	0.7674	0.7107	0.3111
		Proposed	0.8764	0.7388	0.8483	0.9422	**0.9342**
		Ada-SVM	0.7850	0.6903	0.8569	0.9361	0.6892
		CS-SVM	0.5561	0.4483	0.6963	0.7249	0.5267
		Prorated cost	0.3868	0.3282	0.5499	0.6945	0.5692
		No cost	0.7751	0.6431	0.7651	0.8555	0.8292
Ecoli_im	3	Decision tree	0.8125	0.7274	0.8780	0.8213	0.7158
		Easy ensemble	**0.8782**	0.7551	0.8629	0.9431	0.9108
		SMOTEBoost	0.8484	0.7442	0.8720	0.9073	0.8175
		SMOTEBagging	0.8219	0.7310	0.8748	**0.9483**	0.7392
		SMOTE + SVM	0.6711	0.6828	0.7103	0.7574	0.7458
		AdaBoost	0.8439	**0.7675**	**0.8928**	0.9361	0.7683
		Proposed	**0.7771**	0.5824	0.7338	0.8786	0.8810
		Ada-SVM	0.7444	0.6255	0.8443	0.8272	0.6238
		CS-SVM	0.7357	0.6245	0.8503	0.8412	0.6000
		Prorated cost	0.6245	0.4702	0.8052	0.7967	0.5095
		No cost	0.5990	0.4345	0.7338	0.8222	0.5524
Hepatitis	4	Decision tree	0.5660	0.4074	0.7600	0.6289	0.4048
		Easy ensemble	0.7625	0.5870	0.7723	0.8365	0.7524
		SMOTEBoost	0.7235	0.5969	0.8381	0.8452	0.5905
		SMOTEBagging	0.7218	0.6185	**0.8508**	0.8713	0.5714
		SMOTE + SVM	0.7311	**0.7628**	0.7584	**0.8914**	0.8217
		AdaBoost	0.7021	0.5773	0.8372	0.7634	0.5571

Table 3: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.4859	0.4208	0.4153	0.7810	**1.0000**
		Ada-SVM	0.4552	0.3536	0.7604	0.7541	0.4182
		CS-SVM	**0.7469**	0.5417	0.7155	0.8468	0.8182
		Prorated cost	0.6581	0.4917	0.6365	0.8044	0.8000
		No cost	0.5254	0.3695	0.4944	0.6757	0.7636
SPECT	4	Decision tree	0.5931	0.4149	0.7530	0.6512	0.4364
		Easy ensemble	0.7200	0.5189	0.7305	0.8104	0.7091
		SMOTEBoost	0.6829	0.5178	0.7980	0.8486	0.5818
		SMOTEBagging	0.6790	0.5346	0.8165	0.8212	0.5455
		SMOTE + SVM	0.6087	**0.7352**	0.6652	**0.8584**	0.9293
		AdaBoost	0.6995	0.5791	**0.8354**	0.8379	0.5455
		Proposed	0.4580	0.5405	0.4763	0.7392	**0.9400**
		Ada-SVM	0.6295	0.5381	0.6766	0.7102	0.5689
		CS-SVM	0.6206	0.5303	**0.6886**	0.7343	0.6044
		Prorated cost	0.5937	0.4955	0.6495	**0.7819**	0.6156
		No cost	0.4852	0.4913	0.5034	0.6684	0.7867
Teaching	4	Decision tree	**0.6807**	0.5828	0.7348	0.6981	0.5711
		Easy ensemble	0.6505	0.5568	0.6417	0.6855	0.6933
		SMOTEBoost	0.5962	0.4819	0.6428	0.7424	0.5289
		SMOTEBagging	0.6511	0.5499	0.6884	0.7184	0.5711
		SMOTE + SVM	0.4552	0.5842	0.5394	0.6162	0.7195
		AdaBoost	0.5974	**0.6352**	0.6127	0.6666	0.6976
		Proposed	0.2842	0.4636	0.3660	0.5800	**0.9133**
		Ada-SVM	0.5100	0.3767	0.7060	0.6735	0.3100
		CS-SVM	0.5044	0.4836	0.5560	0.7265	0.7433
		Prorated cost	0.6228	0.5170	0.6660	0.7275	0.6267
		No cost	0.5265	0.4234	0.5920	0.6216	0.5567
German credit	5	Decision tree	0.6265	0.4975	0.6860	0.6386	0.5200
		Easy ensemble	**0.7117**	0.5976	0.7020	0.7668	0.7400
		SMOTEBoost	0.6652	0.5499	**0.7340**	**0.7709**	0.5433
		SMOTEBagging	0.6398	0.5216	0.7206	0.7649	0.4967
		SMOTE + SVM	0.5271	**0.6827**	0.5993	0.6777	0.8543
		AdaBoost	0.5974	0.6352	0.6127	0.6666	0.6976
		Proposed	0.8481	0.6215	0.7860	**0.9719**	**0.9636**
		Ada-SVM	0.8941	0.8010	0.9344	0.9565	0.8436
		CS-SVM	0.7280	0.6688	0.7972	0.9345	0.7673
		Prorated cost	0.6471	0.5203	0.7227	0.9587	0.7091
		No cost	0.5629	0.4807	0.6257	0.6281	0.8127
Ecoli_pp	5	Decision tree	0.8013	0.6923	0.9076	0.8218	0.6964
		Easy ensemble	**0.9020**	0.7661	0.9137	0.9590	0.8873
		SMOTEBoost	0.8734	0.7746	0.9258	0.9493	0.8109
		SMOTEBagging	0.8508	0.7845	**0.9375**	0.9459	0.7545
		SMOTE + SVM	0.7494	**0.8141**	0.7765	0.9340	0.9613
		AdaBoost	0.8498	0.7737	0.9316	0.9227	0.7509

Table 3: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.8115	0.5124	0.7173	0.9280	**0.9853**
		Ada-SVM	0.8198	0.7228	0.9252	0.9525	0.7113
		CS-SVM	0.8031	0.5729	0.7680	0.9721	0.9027
		Prorated cost	0.7405	0.5394	0.8597	0.9914	0.7933
		No cost	0.5563	0.3262	0.4642	0.6836	0.8900
Segmentation	6	Decision tree	0.9490	0.9315	0.9810	0.9500	0.9067
		Easy ensemble	0.9614	0.8858	0.9648	0.9977	0.9567
		SMOTEBoost	0.9900	**0.9817**	**0.9948**	**0.9997**	0.9833
		SMOTEBagging	**0.9656**	0.9622	0.9895	0.9887	0.9333
		SMOTE + SVM	0.6417	0.3457	0.5441	0.6966	0.8507
		AdaBoost	0.9599	0.9287	0.9795	0.9934	0.9333
		Proposed	**0.9496**	0.8191	0.9393	0.9523	**0.9667**
		Ada-SVM	0.8572	0.7644	0.9389	0.9443	0.7800
		CS-SVM	0.6086	0.5525	0.6193	0.9346	0.9000
		Prorated cost	0.3100	0.2508	0.3545	0.9177	0.8333
		No cost	0.8053	0.6747	0.9162	0.8748	0.7333
Glass 6	6	Decision tree	0.8801	0.8212	0.9534	0.8892	0.8000
		Easy ensemble	0.9105	0.7716	0.9298	**0.9753**	0.8933
		SMOTEBoost	0.8950	0.8470	0.9627	0.9604	0.8267
		SMOTEBagging	0.9117	0.8433	0.9578	0.9502	0.8600
		SMOTE + SVM	0.8526	**0.8582**	0.8568	0.9252	0.8595
		AdaBoost	0.8679	0.8264	**0.9623**	0.9440	0.7800
		Proposed	0.7865	0.3996	0.6755	0.8365	**0.9714**
		Ada-SVM	0.8117	0.4849	0.7345	0.8477	0.9429
		CS-SVM	0.7323	0.5434	0.9018	0.8321	0.6000
		Prorated cost	0.8009	0.5379	0.8633	0.8882	0.7429
		No cost	0.7798	0.4184	0.6873	0.9123	0.9429
Ecoli_imU	9	Decision tree	0.7408	0.5834	0.9166	0.7767	0.6000
		Easy ensemble	**0.8648**	0.5619	0.8541	**0.9384**	0.8857
		SMOTEBoost	0.7876	0.5972	0.9048	0.9231	0.6857
		SMOTEBagging	0.7862	0.6270	**0.9167**	0.9266	0.6571
		SMOTE + SVM	0.6325	**0.6666**	0.7289	0.8883	0.8000
		AdaBoost	0.6629	0.4792	0.8959	0.8871	0.4857
		Proposed	0.8736	0.4531	0.7864	0.8881	**1.0000**
		Ada-SVM	0.7977	0.3020	0.6996	0.8169	0.9500
		CS-SVM	0.8482	0.6540	0.8747	0.9040	0.8500
		Prorated cost	0.8150	0.7329	0.9732	0.9053	0.7000
		No cost	0.7862	0.3271	0.6882	0.9243	0.9500
Ecoli_om	16	Decision tree	0.8218	0.6992	0.9644	0.8405	0.7000
		Easy ensemble	0.8652	0.5543	0.9196	0.9786	0.8500
		SMOTEBoost	0.8835	0.8111	0.9792	0.9780	0.8000
		SMOTEBagging	**0.9164**	**0.8743**	**0.9851**	0.9687	0.8500
		SMOTE + SVM	0.6443	0.6765	0.7391	0.8828	0.8031
		AdaBoost	0.8090	0.7043	0.9702	**0.9885**	0.7000

Table 3: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.8662	0.4837	0.9448	0.9057	0.8194
		Ada-SVM	0.7273	0.4710	0.9636	0.9341	0.5444
		CS-SVM	0.7229	0.2288	0.7279	0.9722	0.8278
		Prorated cost	0.9005	0.3774	0.8417	0.9842	**0.9750**
		No cost	0.8041	0.3688	0.9218	0.9593	0.7333
Yeast 5	33	Decision tree	0.7652	0.6094	0.9784	0.8003	0.6111
		Easy ensemble	0.9392	0.5181	0.9474	**0.9868**	0.9333
		SMOTEBoost	**0.9196**	0.6996	0.9778	0.9866	0.8639
		SMOTEBagging	0.8704	0.7481	**0.9852**	0.9690	0.7722
		SMOTE + SVM	0.7638	**0.8142**	0.7920	0.9171	0.8764
		AdaBoost	0.7629	0.6264	0.9798	0.9796	0.6139

The bold values represent the best values across all techniques.

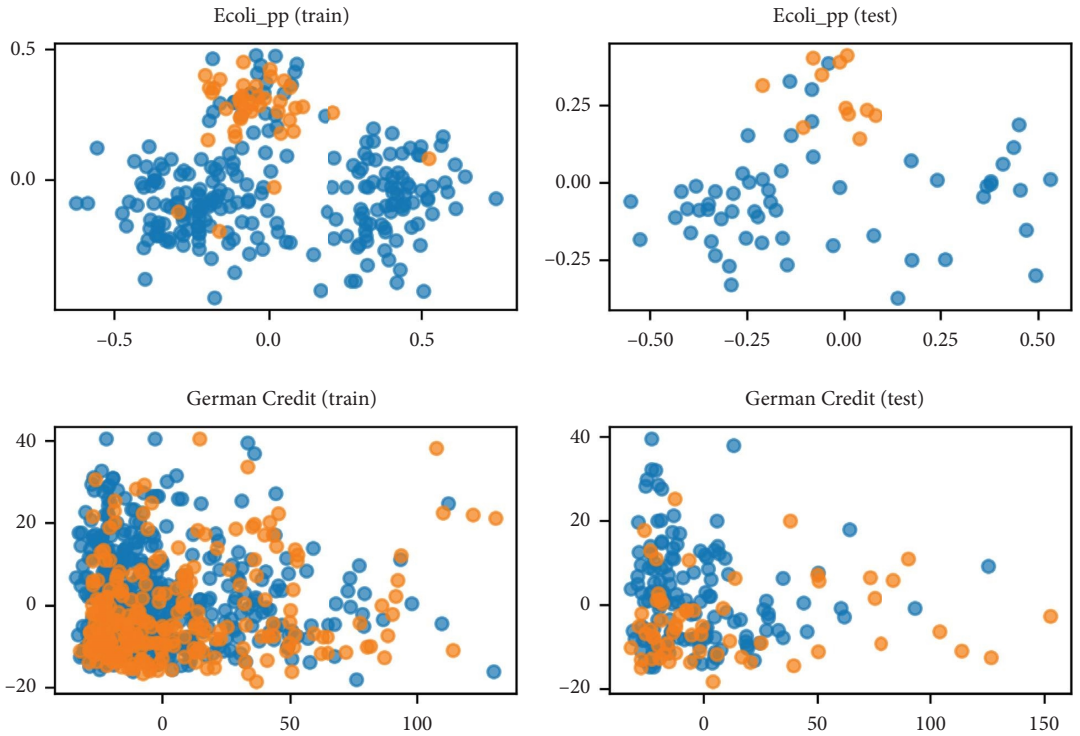

Figure 6: Comparison structure of classes between Ecoli_pp and German credit on the training and testing sets.

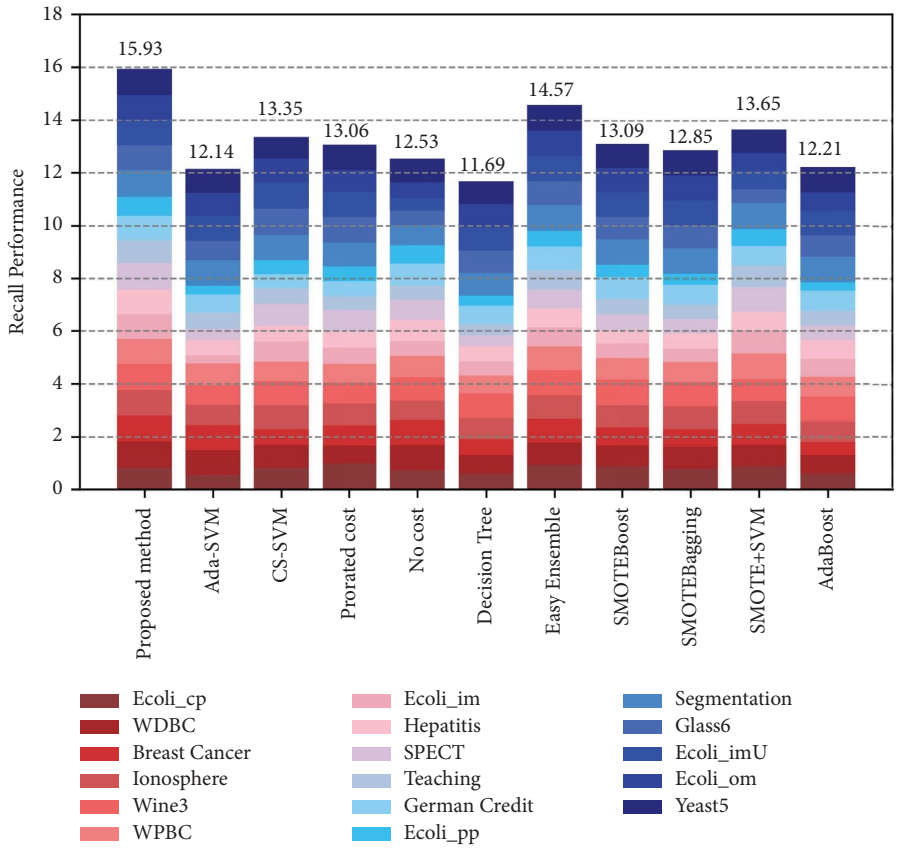

Figure 7: Comparison of different methods' performance on all datasets in terms of recall.

Table 4: P values of the Shapiro-Wilk test for model performance measures in the first group dataset.

Model	G-mean	$F1$-score	P-mean	AUC	Accuracy	Recall
Proposed	**0.002**	**0.028**	**0.036**	**0.013**	**0.032**	**0.001**
Ada-SVM	**0.025**	0.153	**0.205**	**0.018**	**0.026**	0.120
CS-SVM	0.719	0.523	0.716	**0.009**	0.933	0.102
Prorated cost	0.188	0.109	**0.049**	**0.007**	0.185	0.225
No cost	0.142	0.213	**0.037**	0.140	0.554	0.410
Decision tree	0.073	0.094	0.088	0.060	**0.007**	0.093
Easy ensemble	**0.011**	**0.040**	**0.047**	**0.001**	**0.005**	**0.024**
SMOTEBoost	**0.032**	0.167	0.146	**0.004**	**0.022**	0.060
SMOTEBagging	**0.039**	**0.047**	**0.031**	**0.001**	**0.013**	0.117
SMOTE + SVM	0.275	0.117	0.193	**0.033**	0.134	0.757
AdaBoost	0.159	0.289	0.290	**0.002**	**0.001**	0.419

The bold values represent data that obey normal distribution.

Table 5: P values for all performance measures in the first group datasets.

Indicators/Model		Ada-SVM	CS-SVM	Prorated cost	No cost	Decision tree	Easy ensemble	SMOTEBoost	SMOTEBagging	SMOTE + SVM	AdaBoost
G-mean	T	0.834	0.455	0.350	**0.035**	0.850	0.147	0.293	0.284	0.065	0.905
	Wilcox	0.361	0.117	0.278	**0.007**	0.877	0.134	0.278	0.293	**0.008**	0.959
$F1$-score	T	0.906	0.658	0.519	**0.018**	0.430	0.240	0.114	0.057	0.328	0.198
	Wilcox	0.796	0.931	0.667	**0.029**	0.418	0.134	0.088	**0.034**	0.459	0.134
P-mean	T	0.608	0.502	0.407	**0.009**	0.818	0.378	0.264	0.134	0.496	0.441
	Wilcox	0.642	0.718	0.380	**0.014**	0.770	0.249	0.235	0.095	0.570	0.310
AUC	T	0.849	0.715	0.735	0.068	0.094	0.287	0.297	0.276	0.268	0.882
	Wilcox	0.931	0.642	0.418	0.052	0.052	0.068	0.134	0.143	0.174	0.438
Accuracy	T	0.094	0.665	0.752	0.270	**0.028**	0.051	**0.019**	**0.006**	0.277	0.051
	Wilcox	0.270	0.823	0.959	0.148	0.060	0.056	**0.022**	**0.005**	0.125	0.029
Recall	T	**0.000**	**0.001**	**0.000**	**0.000**	**0.000**	**0.022**	**0.001**	**0.001**	**0.001**	**0.000**
	Wilcox	**0.000**	**0.003**	**0.002**	**0.000**	**0.000**	**0.011**	**0.000**	**0.001**	**0.000**	**0.000**

The bold values represent the parts of the tests that are significant.

Table 6: G-mean, $F1$-score, accuracy, AUC, and recall performance comparison of different algorithms on Pageblock.

Datasets	IR	Methods	G-mean	$F1$-score	Accuracy	AUC	Recall
pa1	1	Proposed	0.8684	0.8644	0.8700	0.9439	0.8320
		Ada-SVM	0.8674	0.8658	0.8690	0.9432	0.8480
		CS-SVM	0.6823	0.7429	0.7380	0.9407	0.7720
		Prorated cost	0.6747	0.6667	0.7290	0.8726	0.7000
		No cost	0.4675	0.3610	0.5980	0.8633	0.2340
		Decision tree	0.9315	0.9314	0.9320	0.9327	0.9260
		Easy ensemble	0.9409	0.9412	0.9410	0.9789	0.9440
		SMOTEBoost	**0.9488**	**0.9492**	**0.9490**	0.9801	0.9520
		SMOTEBagging	0.9486	0.9498	**0.9490**	**0.9878**	**0.9620**
		SMOTE + SVM	0.4285	0.4335	0.5180	0.6513	0.4240
		AdaBoost	0.9399	0.9399	0.9400	0.9749	0.9380
pa2	2	Proposed	0.9027	0.8626	0.9049	0.9504	0.8971
		Ada-SVM	0.8421	0.7975	0.8696	0.9339	0.7735
		CS-SVM	0.7523	0.7040	0.8147	0.9614	0.7029
		Prorated cost	0.6715	0.5865	0.7745	0.9577	0.6618
		No cost	0.5907	0.5378	0.6265	0.6950	0.6059
		Decision tree	0.9370	0.9187	0.9461	0.9375	0.9118
		Easy ensemble	0.9461	0.9151	0.9402	0.9819	**0.9647**
		SMOTEBoost	0.9365	0.9141	0.9422	0.9767	0.9206
		SMOTEBagging	**0.9515**	**0.9297**	**0.9520**	**0.9940**	0.9500
		SMOTE + SVM	0.6435	0.6408	0.6838	0.8086	0.5515
		AdaBoost	0.9287	0.9104	0.9412	0.9791	0.8941

Table 6: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.8545	0.7849	0.8902	0.9440	0.7922
		Ada-SVM	0.8131	0.7523	0.8902	0.9383	0.7095
		CS-SVM	0.6985	0.5843	0.6814	0.9626	0.8784
		Prorated cost	0.6512	0.5482	0.7333	0.9461	0.6902
		No cost	0.6393	0.5648	0.8402	0.8787	0.4196
pa3	3	Decision tree	0.9357	0.9080	0.9539	0.9366	0.9020
		Easy ensemble	0.9374	0.8732	0.9304	0.9759	**0.9529**
		SMOTEBoost	0.9333	0.8819	0.9382	0.9789	0.9255
		SMOTEBagging	**0.9465**	**0.9197**	**0.9598**	**0.9912**	0.9216
		SMOTE + SVM	0.5464	0.5365	0.6275	0.7605	0.4392
		AdaBoost	0.9082	0.8698	0.9363	0.9805	0.8588
		Proposed	0.7869	0.6927	0.8873	0.9239	0.6776
		Ada-SVM	0.7408	0.6567	0.8824	0.8878	0.5794
		CS-SVM	0.5781	0.5289	0.6363	0.9310	0.7998
		Prorated cost	0.6737	0.5037	0.7824	0.9522	0.6878
		No cost	0.5779	0.4492	0.8108	0.8316	0.3773
pa4	4	Decision tree	0.8913	0.8330	0.9333	0.8965	0.8288
		Easy ensemble	**0.9417**	0.8356	0.9225	0.9759	**0.9755**
		SMOTEBoost	0.8939	0.8166	0.9235	0.9659	0.8484
		SMOTEBagging	0.9227	**0.8631**	**0.9431**	**0.9825**	0.8920
		SMOTE + SVM	0.5572	0.5579	0.6163	0.6974	0.4828
		AdaBoost	0.8788	0.8216	0.9304	0.9707	0.8037
		Proposed	0.8711	0.8115	0.9400	0.9510	0.7845
		Ada-SVM	0.8468	0.7770	0.9290	0.9379	0.7431
		CS-SVM	0.8068	0.6097	0.7640	0.9675	0.9214
		Prorated cost	0.7045	0.5217	0.8330	0.9664	0.6720
		No cost	0.5971	0.4310	0.8220	0.9021	0.4661
pa5	5	Decision tree	0.8909	0.8471	0.9520	0.8966	0.8135
		Easy ensemble	**0.9509**	0.8267	0.9310	0.9858	**0.9822**
		SMOTEBoost	0.9186	0.8357	0.9420	0.9805	0.8859
		SMOTEBagging	0.9398	**0.8859**	**0.9610**	**0.9876**	0.9100
		SMOTE + SVM	0.5798	0.4998	0.6781	0.8644	0.3828
		AdaBoost	0.8925	0.8359	0.9460	0.9719	0.8203
		Proposed	0.8883	0.7462	0.9120	0.9463	0.8603
		Ada-SVM	0.8244	0.7151	0.9200	0.9613	0.7197
		CS-SVM	0.7235	0.5276	0.8050	0.9463	0.7158
		Prorated cost	0.7640	0.5348	0.8340	0.9558	0.7867
		No cost	0.6683	0.4150	0.6980	0.7337	0.6571
pa6	6	Decision tree	0.8776	0.8207	0.9510	0.8837	0.7897
		Easy ensemble	**0.9353**	0.7947	0.9300	**0.9836**	**0.9441**
		SMOTEBoost	0.9266	**0.8541**	**0.9570**	0.9547	0.8879
		SMOTEBagging	0.9096	0.8436	0.9550	0.9797	0.8532
		SMOTE + SVM	0.7270	0.7181	0.7404	0.8113	0.6511
		AdaBoost	0.8794	0.8301	0.9540	0.9610	0.7901
		Proposed	0.8678	0.5990	0.8670	0.9081	0.8800
		Ada-SVM	0.7219	0.6043	0.9300	0.9381	0.5400
		CS-SVM	0.8292	0.6184	0.8370	0.9441	0.8500
		Prorated cost	0.8715	0.6052	0.8880	0.9546	0.8600
		No cost	0.5428	0.2467	0.6540	0.6753	0.4900
pa9	9	Decision tree	0.9146	0.8585	0.9720	0.9178	0.8500
		Easy ensemble	**0.9474**	0.7797	0.9460	**0.9902**	**0.9500**
		SMOTEBoost	0.9356	0.8453	0.9670	0.9789	0.9000
		SMOTEBagging	0.9273	0.8595	0.9720	0.9796	0.8800
		SMOTE + SVM	0.5115	0.5285	0.6011	0.7108	0.5189
		AdaBoost	0.9225	**0.8819**	**0.9770**	0.9862	0.8600
		Proposed	0.9140	0.5916	0.9150	0.9492	0.9152
		Ada-SVM	0.8029	0.6881	0.9650	0.9597	0.6606
		CS-SVM	0.7527	0.6198	0.9590	0.9687	0.5924
		Prorated cost	0.7228	0.5062	0.9330	0.9625	0.5833
		No cost	0.7578	0.3637	0.8270	0.8811	0.7379
pa16	16	Decision tree	0.8344	0.7576	0.9730	0.8492	0.7091
		Easy ensemble	**0.9454**	0.6876	0.9490	0.9910	**0.9455**
		SMOTEBoost	0.9065	0.7452	0.9660	0.9319	0.8470
		SMOTEBagging	0.9029	**0.8130**	0.9770	**0.9912**	0.8318
		SMOTE + SVM	0.3773	0.4681	0.5446	0.6409	0.5012
		AdaBoost	0.8712	0.8034	**0.9780**	0.9539	0.7758

Table 6: Continued.

Datasets	IR	Methods	*G*-mean	*F*1-score	Accuracy	AUC	Recall
		Proposed	0.8287	0.2537	0.7850	0.8307	**0.8933**
		Ada-SVM	0.6681	0.3863	0.9530	0.8689	0.5133
		CS-SVM	0.6654	0.3926	0.8350	0.9409	0.6333
		Prorated cost	0.6131	0.3140	0.9370	0.9399	0.4400
		No cost	0.6843	0.3137	0.9250	0.9497	0.5200
pa36	36	Decision tree	0.6880	0.5366	0.9760	0.7415	0.4933
		Easy ensemble	0.8867	0.3611	0.9180	**0.9737**	0.8800
		SMOTEBoost	**0.9007**	**0.7519**	**0.9850**	0.9341	0.8400
		SMOTEBagging	0.7590	0.5822	0.9760	0.9489	0.6000
		SMOTE + SVM	0.7446	0.7204	0.7657	0.8861	0.6609
		AdaBoost	0.7919	0.6537	0.9830	0.9354	0.6533
		Proposed	0.8130	0.4421	0.9540	0.9045	0.7000
		Ada-SVM	0.6850	0.4578	0.9700	0.9126	0.4900
		CS-SVM	0.5073	0.1274	0.5630	0.8521	0.6400
		Prorated cost	0.8231	0.4430	0.9570	0.9505	0.7500
		No cost	0.6989	0.2909	0.9370	0.8405	0.5400
pa40	40	Decision tree	0.7079	0.5511	0.9810	0.7609	0.5300
		Easy ensemble	**0.8688**	0.3042	0.9040	0.9608	**0.8400**
		SMOTEBoost	0.7504	0.4129	0.9590	0.8809	0.6000
		SMOTEBagging	0.6223	0.4462	0.9760	**0.9770**	0.4140
		SMOTE + SVM	0.6696	**0.6396**	0.7029	0.7905	0.5237
		AdaBoost	0.6286	0.5091	**0.9820**	0.8368	0.4200
		Proposed	**0.8744**	0.3286	0.9130	0.9388	**0.8500**
		Ada-SVM	0.6705	0.2812	0.9330	0.6749	0.5500
		CS-SVM	0.7896	0.3793	0.9070	0.9126	0.7333
		Prorated cost	0.4940	0.2819	0.7980	0.9486	0.4500
		No cost	0.8095	0.2633	0.8750	0.9403	0.8000
pa50	50	Decision tree	0.7232	0.6276	0.9880	0.7735	0.5500
		Easy ensemble	0.8276	0.2396	0.9020	0.9503	0.8000
		SMOTEBoost	0.6937	0.4715	0.9790	0.9446	0.5167
		SMOTEBagging	0.7603	0.6500	**0.9890**	0.9579	0.6167
		SMOTE + SVM	0.6315	0.5722	0.7202	0.8514	0.4941
		AdaBoost	0.7810	**0.6595**	0.9880	**0.9667**	0.6333

The bold values represent the parts of the tests that are significant.

Table 7: *P* values for recall performance measures in the high IR of the Pageblock dataset.

Indicators/Model		Ada-SVM	CS-SVM	Prorated cost	No cost	Decision tree	Easy ensemble	SMOTEBoost	SMOTEBagging	SMOTE + SVM	AdaBoost
Recall	T	**0.000**	0.055	**0.016**	**0.001**	**0.023**	0.110	0.394	0.170	**0.000**	0.060
	Wilcox	**0.004**	0.097	**0.018**	**0.006**	**0.030**	0.125	0.565	0.250	**0.002**	**0.035**

The bold values represent the parts of the tests that are significant.

dataset. This also implies that our approach seems to be more effective with small datasets. However, we found that the recall value of the proposed model presented in Table 7 is significantly better than accuracy-oriented algorithms such as No-Cost SVM, decision tree, and AdaBoost. It also outperforms the recall values of cost-sensitive classifiers such as Ada-SVM, prorated, and SMOTE + SVM. The proposed model does not have statistically significant differences in recall values with CS-SVM, Easy Ensemble, SMOTEBoost, and SMOTEBagging state-of-the-art algorithms.

The proposed, Ada-SVM, CS-SVM, prorated cost, no cost, decision tree, easy ensemble, SMOTEBoost, SMOTEBagging, SMOTE + SVM, and AdaBoost algorithms used to classify the Ecoli series dataset take approximately 100 s, 100 s, 30 s, 30 s, 5 s, 0.03 s, 50 s, 3 s, 3 s, 20 s, and 1 s, respectively. The time required for the Pageblock dataset of size 1000 is 1700 s, 1700 s, 200 s, 200 s, 40 s, 6 s, 160 s, 7 s, 7 s, 120 s, and 3 s.

6. Discussion and Conclusion

Unlike other classification methods that only assign different costs to different categories to achieve cost sensitivity, the proposed model built a process that enabled SVM to self-adaptively update the cost value of each sample by integrations. The misclassified positive samples were assigned higher cost values, while other misclassified negative and correctly classified instances were given lower cost items to decrease their effectiveness in training. The optimization of each base classifier can be reached by these automatically

updated cost values, which are following the selfadaptively weight vector that was decided by our new weighting mechanism.

Through theoretical justifications and empirical studies, the proposed approach had higher recall than others, and there were no differences with other classification measures at the 0.05 level. When dealing with imbalanced datasets, the findings demonstrated that the suggested method outperformed alternative methods statistically significantly. Through extensive experiments on different IR datasets, our method guarantees good results on the classification of a few classes on both high and low IR datasets. In some datasets, the mean recall metric of the proposed method could be one after five-fold cross-validation, while a set of other metrics could be maintained at an average level. We also found that our model underperformed in terms of overall accuracy compared with that of some models. This phenomenon stems from the purpose that we chase, our need is for higher recall rather than overall accuracy. This is momentous for the practical issue of reducing the identification overhead when working with a small number of classes. Because our method can achieve a recall value of 1 in some datasets, this good feature can be an effective aid in practical work, reducing a large amount of burden for manual work in identifying minorities, such as medical diagnosis. This is when the overall accuracy becomes less important than recall.

The advantage of P-mean in assessing the classification impact of a skewed dataset is not evident in this study; however, P-mean can be used to appraise the cost-sensitive classifier. It is worth investigating whether the advantage of this assessment metric can be proved in additional experimental instances. Furthermore, we analyzed and confirmed the results of previous studies about class structure and imbalance ratio. They indeed can have a serious impact on classification performance. The classification effect of all classifiers would be reduced when the sample overlap is excessively high; however, our method can also have the best recall performance among many classifiers. As an important reason for the classification disaster of the imbalanced dataset, the structure of classes can be studied in more depth in future work. The high IR would bring a bad classification performance.

Our model has its limitations. For instance, it works better on small datasets; this may be attributed to the fact that SVM, which is the base classifier of our proposed model, is more suitable for classification problems on small datasets. As regards the longer time that our algorithm needs for classification as compared to others, if we continue to update the program code in the future without changing the model, the time computational complexity of our method will be considerably reduced. Substantial research can be conducted in the future, including parameter evaluation and improvement. The impact of kernel functions on imbalanced classification or imbalanced multiple classification problems is also worth investigating. Because of its enormous application potential, this challenging topic will continue to receive extensive attention.

Appendix

A. Training Error Bounds for the AdaBoost

Proof

$$
\begin{aligned}
&\frac{1}{N}\sum_{i} I\left(G\left(x_i\right) \neq y_i\right),\\
&\leq \frac{1}{N}\sum_{i} \exp\left(-y_i f\left(x_i\right)\right),\\
&= Z_1 \sum_{i} \omega_{2i} \prod_{m=2}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right),\\
&= Z_1\left(\frac{1}{b}\sum_{\{i\,|\,M_2\}} b\omega_{2i} \prod_{m=2}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) + \sum_{\{i\,|\,M_1+M_3\}} \omega_{2i} \prod_{m=2}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right)\right),\\
&= Z_1 Z_2\left(\frac{1}{b^2}\sum_{\{i|\,M_2\}} b\omega_{3i} \prod_{m=3}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) + \sum_{\{i\,|\,M_1+M_3\}} \omega_{3i} \prod_{m=3}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right)\right),\\
&= \cdots,\\
&= Z_1 Z_2 \cdots Z_{M-1}\left(\frac{1}{b^{M-1}}\sum_{\{i|M_2\}} b\omega_{mi} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) + \sum_{\{i|M_1+M_3\}} \omega_{mi} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right)\right),\\
&\leq Z_1 Z_2 \cdots Z_M = \prod_{m=1}^{M} Z_M.
\end{aligned}
\tag{A.1}
$$

□

B. Training Error Bounds for the Binary Classification Problem AdaBoost

Proof

$$\begin{aligned} Z_m &= \sum_{\{i|M1\}} \omega_{mi} \exp\left(-\alpha_m\right) + \sum_{\{i|M2\}} b \cdot \omega_{mi} \exp\left(\alpha_m\right) + \sum_{\{i|M3\}} \omega_{mi} \exp\left(\alpha_m\right), \\ &= \left(1 - e_m\right)\exp\left(-\alpha_m\right) + (b+1)e_m \exp\left(\alpha_m\right), \\ &= (b+2)\sqrt{e_m\left(1-e_m\right)}, \\ &= \left(1 + \frac{b}{2}\right)\sqrt{1 - 4\gamma_m^2}. \end{aligned} \tag{B.1}$$

According to the Taylor series of $\exp(x)$ and $(1-x)\ 1/2$ at $x = 0$, we have

$$\begin{aligned} &\sqrt{1 - 4\gamma_m^2} \\ &\exp\left(-2\gamma_m^2\right) = 1 - 2\gamma_m^2 + 2\gamma_m^4 - \frac{4}{3}\gamma_m^6 + o\left(\gamma_m^6\right). \end{aligned} \tag{B.2}$$

If there exists $\gamma > 0$ with $\gamma_m > \gamma$ for all m, we can obtain

$$\left(1 + \frac{b}{2}\right)^M \prod_{m=1}^{M} \sqrt{1 - 4\gamma_m^2} \le \left(1 + \frac{b}{2}\right)^M \exp\left(-2\sum_m \gamma_m^2\right) \le \left(1 + \frac{b}{2}\right)^M \exp\left(-2M\gamma^2\right). \tag{B.3}$$

□

Data Availability

The datasets used in this study are from the UCI ML repository (http://archive.ics.uci.edu/ml/datasets).

Conflicts of Interest

The authors declare that they have no conflicts of interest

Acknowledgments

This work was supported by the 2022 Heilongjiang University Graduate Student Innovative Research Project (No. YJSCX2022-250HLJU).

References

[1] C. Cortes and V. N. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[3] N. Japkowicz and S. Stephen, "The class imbalance problem: The class imbalance problem: A systematic study1 systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[4] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[5] G. Kushankur, C. Bellinger, R. Corizzo, B. Krawczyk, and N. Japkowicz, "On the combined effect of class imbalance and concept complexity in deep learning," in *Proceedings of the 2021 IEEE International Conference on Big Data (Big Data)*, pp. 4859–4868, Orlando, FL, USA, December, 2021.

[6] S. Shilaskar, A. Ghatol, and P. Chatur, "Medical decision support system for extremely imbalanced datasets," *Information Sciences*, vol. 384, pp. 205–219, 2017.

[7] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30, no. 2/3, pp. 195–215, 1998.

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

[9] K. Ezawa, M. Singh, and S. Norton, "Learning goal-oriented Bayesian networks for telecommunications risk management," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 139–147, Bari, Italy, July, 1996.

[10] C. Cardie and N. Howe, "Improving minority class prediction using case-specific feature weights," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 57–65, Nashville, TN, USA, July 1997.

[11] J. Gou, Y. Xu, D. Zhang, Q. Mao, L. Du, and Y. Zhan, "Two-phase linear reconstruction measure-based classification for face recognition," *Information Sciences*, vol. 433-434, pp. 17–36, 2018.

[12] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[13] K. Shin, J. Han, S. Kang, and Mi-Mote, "MI-MOTE: Multiple imputation-based minority oversampling technique for imbalanced and incomplete data classification," *Information Sciences*, vol. 575, pp. 80–89, 2021.

[14] X. Wang, J. Xu, T. Zeng, and L. Jing, "Local distribution-based adaptive minority oversampling for imbalanced data classification," *Neurocomputing*, vol. 422, pp. 200–213, 2021.

[15] Z. Qu, H. Liu, Z. Wang, J. Xu, P. Zhang, and H. Zeng, "A combined genetic optimization with AdaBoost ensemble model for anomaly detection in buildings electricity consumption," *Energy and Buildings*, vol. 248, Article ID 111193, 2021.

[16] J. Sun, H. Li, H. Fujita, B. Fu, and W. Ai, "Class-imbalanced dynamic financial distress prediction based on Adaboost-SVM ensemble combined with SMOTE and time weighting," *Information Fusion*, vol. 54, pp. 128–144, 2020.

[17] H. Jiang, B. Zou, C. Xu, J. Xu, and Y. Y. Tang, "SVM-Boosting based on Markov resampling: SVM-Boosting based on Markov resampling: Theory and algorithmheory and algorithm," *Neural Networks*, vol. 131, pp. 276–290, 2020.

[18] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomputing*, vol. 343, pp. 50–64, 2019.

[19] B. B. Hazarika and D. Gupta, "Density-weighted support vector machines for binary class imbalance learning," *Neural Computing and Applications*, vol. 33, no. 9, pp. 4243–4261, 2021.

[20] D. Gupta, B. Richhariya, and P. Borah, "A fuzzy twin support vector machine based on information entropy for class imbalance learning," *Neural Computing and Applications*, vol. 31, no. 11, pp. 7153–7164, 2019.

[21] D. Gupta and B. Richhariya, "Entropy based fuzzy least squares twin support vector machine for class imbalance learning," *Applied Intelligence*, vol. 48, no. 11, pp. 4212–4231, 2018.

[22] P. Borah and D. Gupta, "Robust twin bounded support vector machines for outliers and imbalanced data," *Applied Intelligence*, vol. 51, no. 8, pp. 5314–5343, 2021.

[23] B. Krawczyk, M. Woźniak, and G. Schaefer, "Cost-sensitive decision tree ensembles for effective imbalanced classification," *Applied Soft Computing*, vol. 14, pp. 554–562, 2014.

[24] U. Gupta and D. Gupta, "Lagrange twin-bounded support vector machine based on l2-norm," in *Advances in Intelligent Systems and Computing*, vol. 740, 2019.

[25] G. J. Karakoulas and J. Shawe-Taylor, "Optimizing classifiers for imbalanced training sets," in *Advances in Neural Information Processing Systems*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds., pp. 253–259, MIT Press, Cambridge, MA, USA, 1999.

[26] F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *Journal of Machine Learning Research*, vol. 7, pp. 1713–1741, 2006.

[27] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6585–6608, 2012.

[28] W. Lee, C. H. Jun, and J. S. Lee, "Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification," *Information Sciences*, vol. 381, pp. 92–103, 2017.

[29] N. H. C. Lima, A. D. D. Neto, and J. D. De Melo, "Creating an ensemble of diverse support vector machines using Adaboost," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1802–1806, Atlanta, GA, USA, June, 2009.

[30] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting,"

Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.

[31] Q. Fu, B. Jing, P. He, S. Si, and Y. Wang, "Fault Feature Selection and Diagnosis of Rolling Bearings Based on EEMD and Optimized Elman_AdaBoost Algorithm," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5024–5034, 2018.

[32] J. Shi, D. Wang, J. Wang et al., "Comparative analysis of the complete mitochondrial genomes of three geographical topmouth culter (Culter alburnus) groups and implications for their phylogenetics," *Bioscience Biotechnology and Biochemistry*, vol. 81, no. 3, pp. 482–490, 2017.

[33] W. Fan, S. Stolfo, J. Zhang, and P. Chan, "AdaCost: misclassification cost-sensitive boosting," in *Proceedings of the 16th International Conference on Machine Learning*, pp. 97–105, San Francisco; CA, USA, June, 1999.

[34] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets," *Information Sciences*, vol. 354, pp. 178–196, 2016.

[35] A. Taherkhani, G. Cosma, and T. M. McGinnity, "AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning," *Neurocomputing*, vol. 404, pp. 351–366, 2020.

[36] M. Tahir and A. Khan, "Protein subcellular localization of fluorescence microscopy images: Protein subcellular localization of fluorescence microscopy images: Employing new statistical and Texton based image features and SVM based ensemble classificationmploying new statistical and Texton based image features and SVM based ensemble classification," *Information Sciences*, vol. 345, pp. 65–80, 2016.

[37] L. I. Kuncheva and C. J. Whitaker, "Using diversity with three variants of boosting: aggressive, conservative, and inverse," in *Proceedings of the 3rd International Workshop on Multiple Classifier Systems*, Cagliari, Italy, June, 2002.

[38] K. Bache and M. Lichman, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, CA, USA, 2013.

[39] B. B. Hazarika and D. Gupta, "Density weighted twin support vector machines for binary class imbalance learning," *Neural Processing Letters*, vol. 54, no. 2, pp. 1091–1130, 2022.

[40] S. C. Prasad, P. Anagha, and S. Balasundaram, "Robust pinball twin bounded support vector machine for data classification," *Neural Processing Letters*, pp. 1–23, 2022.

[41] X. Tao, Q. Li, W. Guo et al., "Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification," *Information Sciences*, vol. 487, pp. 31–56, 2019.

[42] W. Wang and D. Sun, "The improved AdaBoost algorithms for imbalanced data classification," *Information Sciences*, vol. 563, pp. 358–374, 2021.

[43] C. Li, X. Q. Ding, and Y. S. Wu, "Revised AdaBoost algorithm - ad AdaBoost," *Chinese Journal of Computers*, vol. 30, no. 1, pp. 103–109, 2007.

[44] X. Li, L. Wang, and E. Sung, "AdaBoost with SVM-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, 2008.

[45] B. Liu, C. Liu, Y. Xiao, L. Liu, W. Li, and X. Chen, "AdaBoost-based transfer learning method for positive and unlabelled learning problem," *Knowledge-Based Systems*, vol. 241, Article ID 108162, 2022.

[46] G. Yao, X. Hu, and G. Wang, "A novel ensemble feature selection method by integrating multiple ranking information combined with an SVM ensemble model for enterprise credit risk prediction in the supply chain," *Expert Systems with Applications*, vol. 200, Article ID 117002, 2022.

[47] R. Wei, J. Jiang, H. Xu, X. Sun, and Y. Chen, "Fault diagnosis of aircraft actuators based on AdaBoost-ASVM," in *Lecture Notes in Electrical Engineering*, vol. 644, pp. 135–147, LNEE, 2022.

[48] Z. Mehmood and S. Asghar, "Customizing SVM as a base learner with AdaBoost ensemble to learn from multi-class problems: A hybrid approach AdaBoost-MSVM," *Knowledge-Based Systems*, vol. 217, Article ID 106845, 2021.

[49] S. Wan, X. Li, Y. Yin, and J. Hong, "Milling chatter detection by multi-feature fusion and Adaboost-SVM," *Mechanical Systems and Signal Processing*, vol. 156, Article ID 107671, 2021.

[50] S. Mahajan, A. Raina, X. Z. Gao, and A. Kant Pandit, "Plant recognition using morphological feature extraction and transfer learning over SVM and AdaBoost," *Symmetry*, vol. 13, no. 2, pp. 356–16, 2021.

[51] J. Li, L. Sun, and R. Li, "Nondestructive detection of frying times for soybean oil by NIR-spectroscopy technology with Adaboost-SVM (RBF)," *Optik*, vol. 206, Article ID 164248, 2020.

[52] C. Jin, X. Kong, J. Chang, H. Cheng, and X. Liu, "Internal crack detection of castings: a study based on relief algorithm and Adaboost-SVM," *The International Journal of Advanced Manufacturing Technology*, vol. 108, no. 9-10, pp. 3313–3322, 2020.

[53] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, QC, Canada, August 1999.

[54] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods," *Journal of Machine Learning Research*, vol. 5, pp. 725–775, 2004.

[55] C. Y. Yang, J. S. Yang, and J. J. Wang, "Margin calibration in SVM class-imbalanced learning," *Neurocomputing*, vol. 73, no. 1-3, pp. 397–411, 2009.

[56] X. Liu, J. Wu, and Z. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[57] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: improving prediction of the minority class in boosting," *Lecture Notes in Artifificial Intelligence*, vol. 2838, pp. 107–119, 2003.

[58] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.