Hindawi
Complexity
Volume 2023, Article ID 2176891, 19 pages
https://doi.org/10.1155/2023/2176891

Research Article

IMBoost: A New Weighting Factor for Boosting to Improve the Classification Performance of Imbalanced Data

SeyedEhsan Roshan, Jafar Tanha, Farzad Hallaji, and Mohammad-reza Ghanbari

Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

Correspondence should be addressed to Jafar Tanha; tanha@tabrizu.ac.ir

Received 9 September 2023; Revised 23 October 2023; Accepted 27 October 2023; Published 11 November 2023

Academic Editor: Giacomo Fiumara

Imbalanced datasets pose significant challenges in the field of machine learning, as they consist of samples where one class (majority) dominates over the other class (minority). Although AdaBoost is a popular ensemble method known for its good performance in addressing various problems, it fails when dealing with imbalanced data sets due to its bias towards the majority class samples. In this study, we propose a novel weighting factor to enhance the performance of AdaBoost (called IMBoost). Our approach involves computing weights for both minority and majority class samples based on the performance of classifier on each class individually. Subsequently, we resample the data sets according to these new weights. To evaluate the effectiveness of our method, we compare it with six well-known ensemble methods on 30 imbalanced data sets and 4 synthetic data sets using ROC, precision-eecall AUC, and G-mean metrics. The results demonstrate the superiority of IMBoost. To further analyze the performance, we employ statistical tests, which confirm the excellence of our method.

1. Introduction

Imbalanced problems have occurred in various fields such as fault detection [1], anomaly detection [2], credit risk prediction [3], and cancer diagnoses [4]. Imbalanced datasets have been one of the challenging issues in the field of machine learning, where samples belonging to one class, majority or negative, outnumber the other class, minority or positive. Therefore, most classifiers have difficulties in dealing with such problems since they tend to bias towards majority samples, resulting in weak performance in the classification of minority class samples. In recent years, various methods have been developed to cope with imbalanced problems that can be categorized into four taxonomies, namely, the data level, algorithm level, and hybrid approaches.

Data-level methods aim to balance classes and include under-, over-, and hybrid-sampling methods. Undersampling involves removing majority class samples until the minority and majority classes are balanced. This simplifies training and improves run time and storage [5]. However, the main drawback of this method is the removal of potentially useful samples. Oversampling, on the other hand, duplicates or adds minority class samples [6]. While this increases the training set size and time, it might also add meaningless samples to the data set. Hybrid-sampling methods combine multiple sampling techniques to overcome the limitations of individual sampling methods. [7]. Algorithm-level methods modify the algorithm to remove its bias towards the majority classes [8]. Cost-sensitive learning is a commonly used approach in handling imbalanced data sets. It involves assigning a higher cost to misclassified samples from the minority class, forcing the classifiers to classify the minority samples correctly [9]. Defining costs of misclassification is challenging in these methods and is disadvantageous [10]. Hybrid methods are a combination of the aforementioned techniques, with ensemble methods being one of the most common approaches in this category [8]. Ensemble learning algorithms demonstrate strong performance in addressing balanced problems; however, they encounter challenges when faced with skewed data sets due to their primary focus on maximizing accuracy. To

address this limitation, the literature proposes the integration of ensemble learning algorithms with other approaches, including algorithm-level and data-level techniques, to effectively handle imbalanced data sets. By combining these methods, ensemble learning algorithms can better adapt to imbalanced issues and improve their classification performance. [11].

Overall, ensemble methods train base learners and combine them and the results show that a combination of base classifiers outperforms individual classifiers [12]. Bagging [13] and boosting [14] are the most common ensemble methods. In the bagging technique, classifiers are trained in parallel using training sets known as "bags." These bags are created by sampling the original data set with replacement, a method called bootstrap sampling [13]. On the other hand, boosting algorithms work iteratively by placing a stronger emphasis on difficult samples. This is accomplished by increasing the weights assigned to samples that are incorrectly classified by the base classifier at each iteration [14].

Boosting is a widely recognized approach in ensemble learning, known for its ability to create a strong and robust classifier by combining multiple weak classifiers [15]. The strength of boosting lies in its serial learning aspect, which leads to excellent generalization. In essence, weak classifiers are learned sequentially with the objective of reducing the errors made by previously trained classifiers. Over time, various boosting approaches have been proposed. AdaBoost is a popular boosting technique used in ensemble schemes. It follows an iterative process where a set of weak classifiers is trained on weighted data. The outputs of these weak classifiers are combined through a weighted summation to produce the final boosted classifier. Despite the good performance of AdaBoost, its performance against unbalanced data sets should be improved due to giving the same weight to minor and major samples [16]. Moreover, the weak performance of base classifiers in AdaBoost can affect the performance of AdaBoost against imbalanced problems. In this work, we propose a new approach to improve the performance of AdaBoost in dealing with imbalanced problems. To do this, unlike traditional AdaBoost, we compute the performance of classifiers on minor and major samples separately. Furthermore, as part of the boosting process, the training data are resampled based on the weights assigned to each sample. Therefore, a new training data set is created by sampling (with replacement). In the new data set, samples with higher weights are repeated multiple times. This repetition allows the classifier to bias towards these particular samples, leading to a better learning process. By giving more importance to these weighted samples, the classifier can focus on capturing their patterns and characteristics, potentially resulting in improved performance. In addition, the initial weights of positive and negative samples are calculated based on their distribution. To validate the effectiveness of our proposed approach, we perform comprehensive experiments and compare our approach with state-of-the-art methods.

The remaining sections of the paper are structured as follows. Section 2 provides a review of existing methods for imbalanced data sets. Section 3 explains the proposed method. Sections 4 and 5 present the experimental setup and results, and conclusions are finally presented in Section 6.

2. Related Works

The advantages of ensemble learning motivate researchers to focus on the feasibility of using boosting algorithms. The base concept of ensemble learning refers to using multiple learning models instead of a single model to get better performances. Recently, a lot of ensemble-based methods have been proposed to cope with imbalanced problems. In this section, we shortly discuss these methods.

Chawla et al. [17] have proposed a method to oversample the data of minority class called the SMOTE boost method for learning from imbalanced data sets. For the first time, the issue of using the majority class to make the data set balanced was raised in [18]. RUSBoost is an ensemble method that combines data sampling and boosting to enhance classification performance in the presence of imbalanced training data. It addresses the limitations of SMOTEBoost by addressing the complexity and time-consuming nature of data sampling.

Undersampling is one of the popular methods to address imbalanced problems. Liu et al. [19] proposed a method in which many majority class samples were ignored. The main idea of this work was to sample several subsets from the majority class and then train a learner using each of them and finally combine the outputs of those learners.

To improve the performance of the boosting algorithm for imbalanced data, Iosifidis et al. [20] proposed a novel cost-sensitive boosting approach (called AdaCC) that dynamically adjusts the misclassification costs over the boosting rounds in response to the model's performance. This approach is more effective than using a fixed misclassification cost matrix.

Yuan and Ma [21] proposed a novel approach to addressing imbalanced data sets by combining oversampling with AdaBoost and retraining the weights of the classifiers using optimization techniques such as genetic algorithms. This approach allows for the direct optimization of targeted performance measures, such as G-mean and F-measure, and can improve the overall performance of the AdaBoost algorithm on imbalanced data sets.

Mostafaei and Tanha [22] proposed a new technique for addressing imbalanced data sets. The technique involves undersampling the majority class using peak clustering. In addition, the paper proposed a boosting-based algorithm called OUBoost, which combines the peak undersampling technique with SMOTE. It selectively chooses useful examples from the majority class and generates synthetic examples from the minority class, thereby indirectly modifying the update weights.

Ke et al. [23] proposed a method called majority resampling via subclass clustering (MRSC). This approach tried to address the issue of imbalanced data sets. It utilized a clustering algorithm to cluster the majority class instances into numerous groups or subclasses. Subsequently, a new training set was created by combining these subclasses with

the minority class data. The imbalanced ratio (IR) of the new multiclass data set had lower IR compared to the original data set. Finally, the classifier was trained using the created data set.

Roshan and Asadi [10] proposed a multiobjective approach to address imbalanced problems. They utilized NSGA-II to undersample the majority samples. Finally, the solutions obtained from the Pareto front were used to train classifiers.

Zhai et al. [24] introduced a novel approach to address the shortcomings of SMOTE, particularly its limited diversity and the overlapping nature of generated minority samples. Their method incorporated the use of generative adversarial networks (GANs) and combined an oversampling technique with a two-class imbalanced data classification approach. The oversampling method consisted of an enhanced GAN model, while the classification method involved fusing classifiers using fuzzy integral.

Puri and Kumar [25] introduced an enhanced hybrid bag-boost model that incorporates a novel resampling technique. This technique comprises two main components, namely, K-means SMOTE for oversampling and edited nearest neighbor (ENN) for undersampling to eliminate noise from the data set. This technique operated in three steps as follows: first, it clustered the data set using K-means clustering, then it applied SMOTE within each cluster to address class imbalance by generating synthetic instances of the minority class, and finally, it employed ENN to eliminate instances that contribute to noise.

Wang et al. [26] introduced a hybrid strategy called Majority-to-Minority Resampling (MMR) and a boosting algorithm called Majority-to-Minority Boosting (MMBoost) for classification tasks. MMR was developed to tackle class imbalance by taking samples from the majority class to augment the minority class. This adaptive resampling approach aimed to mitigate information loss. In addition, the MMBoost algorithm adjusts weights of the sampled instances to further enhance classification performance.

Arafa et al. [27] proposed a preprocessing method called Reduced Noise-SMOTE (RN-SMOTE) to address unbalanced problems. The RN-SMOTE method consisted of several steps. First, it used SMOTE to oversample the training data. However, since SMOTE added noise to minority classes, DBSCAN was applied to identify and remove noise. Once the noise was eliminated, the clean synthetic instances were combined with the original data. Then, RN-SMOTE was employed to rebalance the data set again using SMOTE before feeding it into the underlying classifier.

Li et al. [28] presented a novel ensemble method that combined ensemble learning techniques with a new undersampling method called binary PSO instance selection. The proposed method aimed to address the challenges of imbalanced classification problems. By leveraging the strengths of ensemble learning and the undersampling technique, the method effectively selected a suitable combination of majority class samples to create a new data set that incorporates minority class samples. The approach utilized a multiobjective strategy to optimize the performance of imbalanced classification while preserving the integrity of the original data set.

Dong and Qian [29] introduced DBRF, a density-based random forest algorithm, to enhance prediction performance for imbalanced problems. DBRF focuses on identifying challenging boundary samples and employs a density-based approach to augment them. Two distinct random forest classifiers are then built to model the augmented boundary samples and the original data set separately. The final output is determined using a bagging technique, which combines the predictions from the two random forest classifiers.

Gu et al. [30] proposed the DCI-ISSA equilibrium ensemble method to address class imbalance issues. It introduced two techniques as follows: data center interpolation (DCI) for creating balanced data sets and the improved sparrow search algorithm (ISSA) for parameter optimization in random forest (RF). These techniques improved classification performance and adapted to different imbalanced data sets.

Zhao et al. [31] introduced a weighted hybrid ensemble method, called WHMBoost, designed to classify imbalanced data in binary classification tasks. WHMBoost combined two data sampling methods and two base classifiers within a boosting algorithm framework. Each sampling method and base classifier were assigned specific weights to leverage their complementary advantages and enhance performance.

Morais and Vasconcelos [32] introduced the k-influential neighborhood for oversampling (k-INOS) algorithm, which aimed to enhance the robustness of oversampling algorithms when dealing with noisy examples in the minority class. The k-INOS algorithm followed a three-step process. First, it detected and removed instances in the minority class that were likely to be noise. Then, an oversampling algorithm was applied to the undersampled data set. Finally, the previously removed minority examples were reintroduced into the data set after oversampling, ensuring that they were not used to augment the minority class directly.

In [33], an ensemble algorithm called BPSO-Ada Boost-KNN for addressing multiclass imbalanced data classification was proposed. The algorithm combines feature selection and boosting techniques within the ensemble framework. In this model, BPSO (binary particle swarm optimization) is employed as the feature selection algorithm using AUCarea as the fitness measure. Finally, a boosting classifier was built in which KNN was chosen as the base classifier.

Wang and Sun [16] presented a method to enhance the AdaBoost algorithm by introducing weighted vote parameters for the weak classifiers. The proposed approach determined the weighted vote parameters based not only on the global error rate but also on the classification accuracy rate of the positive class, which is the primary focus of interest.

Piao et al. [34] introduced a cost-sensitive ensemble learning method for classifying imbalanced data, utilizing a support vector machine (SVM) as the base learner within the AdaBoost. The method was developed to rebalance the weights of AdaBoost, influencing the training of the base learner. This weighting strategy focused on increasing the

sample weight of misclassified minority instances while decreasing the sample weight of misclassified majority instances in each round, effectively equalizing their distributions. By employing this method, the classification performance on imbalanced data can be improved. Moreover, other methods had been developed that adjusted weights to instances according to their labels, such as AdaC1-2-3 [35], AdaCost [36], and CSB1-2 [37].

Hao and Huang [38] proposed an algorithm that begins by obtaining a group of k-base classifiers through K-fold cross validation with a set of training samples. A subclassifier was then obtained using a voting method. Concurrently, weight coefficients for each subclassifier were derived based on the training error and adjustments were made to the data distribution of the training samples.

Furthermore, multiple base classifier groups contributed to obtaining more diverse subclassifiers. Finally, these subclassifiers were combined with weights to create an integrated enhanced classifier.

3. IMBoost

3.1. Original AdaBoost. Before delving into our method for improving AdaBoost to deal with imbalanced problems, we first review AdaBoost. Let $S = \{(x_1, y_2), \ldots, (x_m, y_m)\}$ be a training set, where each sample x_i belongs to sample space X and each label y_i belongs to label space Y.

The concept behind AdaBoost is to utilize a weak learner to make precise predictions by iteratively training the weak learner on different distributions of the training samples. The algorithm for AdaBoost is presented in Algorithm 1. AdaBoost assigns weights to the training samples based on their classification errors and increases or decreases the weights of the samples that are classified incorrectly or correctly, respectively.

3.2. IMBoost. In our work, we focus on binary imbalanced problems, where the minority class is labeled as $y_{\min} = +1$ and the majority class is labeled as $y_{\text{maj}} = -1$. We consider a weak learner that accepts the training set as input along with a distribution D over $\{1, \ldots, m\}$, which is our training set, and m is number of samples. Given this input, the weak learner computes a weak hypothesis h that maps x to R. The sign of $h(x)$ is used to determine the predicted label to be assigned to sample x. If $h(x)$ is positive, the predicted label is assigned as the positive class, and if $h(x)$ is negative, the predicted label is assigned as the negative class.

In general, AdaBoost is not well suited for imbalanced classification problems. This is because the minority class samples are underrepresented and AdaBoost tends to bias towards the majority class. Consequently, the overall performance of the classifier can negatively be impacted. Therefore, it is important that the weights of minority and majority class samples are adjusted according to classifier performance on them. In this paper, we propose an improved version of AdaBoost that revised the weighting mechanism of AdaBoost for binary imbalanced data sets.

In our method, at first, we set the initial weights to $1/N_{\min}$ and $1/N_{\text{maj}}$ for minority and majority class samples, respectively, where $N_{\min}$ is the number of samples belonging to the minority class, N_{maj} is the number of samples belonging to the majority class, $i_{\min}$ and i_{maj} are ith example that belongs to minority class samples and ith example that belongs to majority class samples, respectively.

Similar to original AdaBoost, we train a weak classifier $h_t(x)$ using the distribution over the training set.

While in AdaBoost, the weighted classification error ε is computed over all the training set, we calculate ε for both minority and majority class samples (equation (1)).

$$\text{Error rate of minortiy class samples: } \varepsilon_{\min} = \sum_i I\left(y_{i_{\min}} \neq h_t(x_i)\right),$$
$$\text{Error rate of majority class samples: } \varepsilon_{\text{maj}} = \sum_i I\left(y_{i_{\text{maj}}} \neq h_t(x_i)\right), \tag{1}$$

where I is an indicator that yields a value of 1 if the condition inside the parentheses is true and 0 if it is false.

Then, the weights of h_t on minority and majority samples, denoted by $\alpha_{t_{\min}}$ and $\alpha_{t_{\min}}$, are computed based on $\varepsilon_{\min}$ and ε_{maj}. Intuitively, $\alpha_{t_{\min}}$ and $\alpha_{t_{\min}}$ measure the significance of h_t on minority and majority sample classes; $\alpha_{t_{\min}}$ and $\alpha_{t_{\min}}$ get larger if $\varepsilon_{\min}$ and ε_{maj} get smaller. $\alpha_{t_{\min}}$ and $\alpha_{t_{\min}}$ are calculated as follows:

$$\alpha_{t_{\min}} = \frac{1}{2} log \frac{1 - \varepsilon_{\min}}{\varepsilon_{\min}},$$
$$\alpha_{t_{\text{maj}}} = \frac{1}{2} log \frac{1 - \varepsilon_{\text{maj}}}{\varepsilon_{\text{maj}}}. \tag{2}$$

Once the error rate and hypothesis weight are computed, the weights of the training samples are then modified accordingly, where $\exp(-\alpha_t \times y_i \times h_t(x_i))$ and $\exp(-\alpha_t \times y_i \times h_t(x_i))$ are the weight update factor for minority and majority class samples, respectively. The misclassified samples by h_t get higher weights, and correctly classified samples get lower weights. Subsequently, the training data undergo a resampling process based on the updated weights of the samples. At this stage, the weights of both minority and majority class samples are reinitialized. In the updated data set, samples with higher weights have a higher likelihood of being repeated multiple times. As a result, the classifier becomes biased towards these samples, leading to improved learning performance. Finally, the final classifier $H(x)$ is obtained by combining all weak classifiers using their weights. The pseudocode of our method is presented in Algorithm 2.

3.3. Forward Stagewise Additive Modeling (FSAM). Friedman et al. [39] presented a statistical interpretation of the binary AdaBoost algorithm and demonstrated that the two-class AdaBoost algorithm can be considered as a forward stagewise additive modeling using the exponential loss function as follows:

Input: training set $S = \{(x_1, y_2), \ldots, (x_m, y_m)\}$, where x_i is the feature vector and y_i is the label; number of iterations: T. Output: final classifier $H(x)$.

(1) Initialize weights $D_1(i) = 1/m$, for $i = 1, \ldots, m$

(2) For $t = 1$ to T:

(a) Train a weak classifier $h_t(x)$ using the distribution D_t

(b) Compute the training errorrate ε_t with respect to D_t given by:

$\varepsilon_t = \sum_{i=1}^{m} D_t(i) \times I(y_i \neq h_t(x_i))$

(c) Computetheweighto $f h_t$:

$\alpha_t = 1/2 \times log\ (1 - \varepsilon_t/\varepsilon_t)$

(d) Update the weights by:

$D_{t+1}(i) = (D_t(i) \times \exp(-\alpha_t \times y_i \times h_t(x_i)))/Z_t$

where Z_t is a normalization factor

(3) Output the final classifier:

$H(x) = \text{sign}(\sum_{i=1}^{t} \alpha_t f_t(x))$

Algorithm 1: The pseudocode of AdaBoost.

Input: training set $S = \{(x_1, y_2), \ldots, (x_m, y_m)\}$, where x_i is the feature vector and y_i is the label; number of iterations: T. Output: final classifier $H(x)$.

(1) Initialize weights $D_1(i_{\min}) = 1/N_{\min}$ and $D_1(i_{\text{maj}}) = 1/N_{\text{maj}}$, for $i = 1, \ldots, m$

(2) Fort $= 1$ to T:

(a) Train a weakclassifier $h_t(x)$ using the distribution D_t

(b) Compute the training error rates $\varepsilon_{\min}$ and ε_{maj} (with respect to D_t) given:

$\varepsilon_{\min} = \sum_i I(y_{i_{\min}} \neq h_t(x_i))$

$\varepsilon_{\text{maj}} = \sum_i I(y_{i_{\text{maj}}} \neq h_t(x_i))$

(c) Compute the weight of h_t:

$\alpha_{t_{\min}} = 1/2 \log 1 - \varepsilon_{\min}/\varepsilon_{\min}$

$\alpha_{t_{\text{maj}}} = 1/2 \log 1 - \varepsilon_{maj}/\varepsilon_{maj}$

$\alpha_t = \alpha_{t_{\min}} + \alpha_{t_{\text{maj}}}$

(d) Update the weights by:

$D_{t+1}(i_{\min}) = D_t(i_{\min}) \times \exp(-\alpha_{t_{\min}} \times y_i \times h_t(x_i))/Z_t$

$D_{t+1}(i_{\text{maj}}) = D_t(i_{\text{maj}}) \times \exp(-\alpha_{t_{\text{maj}}} \times y_i \times h_t(x_i))/Z_t$

$*$ where Z_t is a normalization factor

(e) $R-$ sample the training data according to weights and initialze the weights of samples as step1.

(3) Output the final classifier:

$H(x) = \text{sign}(\sum_{i=1}^{t} \alpha_t f_t(x))$

Algorithm 2: The pseudocode of the proposed method.

$$L(y, f) = \exp\ (-yf(x)). \tag{3}$$

In this part, we show that our method can be interpreted as forward stagewise additive modeling employing the exponential loss.

Given the training set, we try to find $f(x) = f_1(x), \ldots, f_K(x)$ such that

$$\min_{f(x)} \sum_{i=1} L(y_i, f(x_i)),$$
$$\text{subject to } f_1(x) + \cdots + f_K(x) = 0. \tag{4}$$

We assume that $f(x)$ has the following form:

$$f(x) = \sum_{m=1}^{M} \beta^{(m)} g^{(m)}(x), \tag{5}$$

where $\beta^{(m)}$ is the coefficient, and $g^{(m)}(x)$ is the basis function. We need $g(x)$ to satisfy symmetric constraint as follows:

$$g_1(x) + \ldots + g_K(x) = 0, \tag{6}$$

where g: $x \longrightarrow y \{y_{\min} \text{ or } y_{\text{maj}}\}$.

FSAM aims to build a predictive model by incrementally adding simple base models to the existing model without adjusting the parameters and coefficients of models that have already been added. The pseudocode of FSAM is presented in Algorithm 3.

Now, using exponential loss, we are trying to find β and $g^{(m)}(x)$ to solve the following equation:

(1) Initialize $f_0(x) = 0$
(2) For $m = 1$ to M:
  (a) $(\beta^m, \gamma^m) = \text{argmin}_{\beta,\gamma} \sum_{i-1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
  (b) $f_m(x) = f_{m-1}(x) + \beta b(x; \gamma_m)$

Algorithm 3: The pseudocode of FSAM.

Considering: $b(x_i; \gamma) \longrightarrow G(x_i)$,

$$\left(\beta^{(m)}, G^{(m)}\right) = \text{argmin}_{\beta,G} \sum_{i} \exp \cdot \left(-y_i \left(f_{m-1}(x_i) + \beta G(x_i)\right)\right). \tag{7}$$

Assuming y_i to be $y_{i_{\min}}$ and $y_{i_{\text{maj}}}$, we expand equation (7) as follows:

$$\left(\beta^{(m)}, G^{(m)}\right) = \text{argmin}_{\beta,G} \sum_{i_{\min}}^{N_{\min}} exp\ \left(-y_{i_{\min}} f_{m-1}(x_i) - y_{i_{\min}} \beta_{\min} G(x_i)\right) + \sum_{i_{\text{maj}}}^{N_{\text{maj}}} \exp\ \left(-y_{i_{\text{maj}}} f_{m-1}(x_i) - y_{i_{\text{maj}}} \beta_{\text{maj}} G(x_i)\right), \tag{8}$$

and

$$w_{i_{\min}} = \exp\left(-y_{i_{\min}} f_{m-1}(x_i)\right), \quad w_{i_{\text{maj}}} = \exp\left(-y_{i_{\text{maj}}} f_{m-1}(x_i)\right), \tag{9}$$

where $w_{i_{\min}}$ and $w_{i_{\text{maj}}}$ are the weights of minority and majority class samples, respectively. It worth noting that when the data set is balanced, (8) reduces to (7). Moreover, note that in (8), $w_{i_{\min}}$ and $w_{i_{\text{maj}}}$ are not a function of G and β and they depend on f_m and change at each iteration.

We can expand equation (8) as follows:

$$\begin{aligned}\left(\beta^{(m)}, G^{(m)}\right) &= \text{argmin}_{\beta,G} \sum_{i_{\min}}^{N_{\min}} \exp\ \left(-y_{i_{\min}} \beta_{\min} G(x_i)\right) w_{i_{\min}} + \sum_{i_{\text{maj}}}^{N_{\text{maj}}} \exp\left(-y_{i_{\text{maj}}} \beta_{\text{maj}} G(x_i)\right) w_{i_{\text{maj}}} \\ &= \left(\sum_{I\left(y_{i_{\min}} = G(x_i)\right)} w_{i_{\min}} \exp\left(-\beta_{\min}\right) + \sum_{I\left(y_{i_{\min}} \neq G(x_i)\right)} w_{i_{\min}} \exp\left(\beta_{\min}\right) \right) \\ &+ \left(\sum_{I\left(y_{i_{\text{maj}}} = G(x_i)\right)} w_{i_{\text{maj}}} \exp\left(-\beta_{\text{maj}}\right) + \sum_{I\left(y_{i_{\text{maj}}} \neq G(x_i)\right)} w_{i_{\text{maj}}} \exp\left(\beta_{\text{maj}}\right). \right)\end{aligned} \tag{10}$$

Now, we minimize (10). To do this, first, we assume $\beta_{\min}$ and β_{maj} are fixed, and we just minimize over G. Therefore, for fixed $\beta_{\min}$ and β_{maj}, $G(x)$ is as follows:

$$G = \text{argmin}_G \sum_i w_{i_{\min}} I\left(y_{i_{\min}} \neq G(x_i)\right) + \sum_i w_{i_{\text{maj}}} I\left(y_{i_{\text{maj}}} \neq G(x_i)\right). \tag{11}$$

Next, we want to minimize $\beta_{\min}$ and β_{maj}. We keep β_{maj} as the constant and take derivative with respect to $\beta_{\min}$ and set to zero as follows:

$$\begin{aligned}
&-\sum_{I\left(y_{i_{\min}}=G(x_i)\right)} w_{i_{\min}} \exp\left(-\beta_{\min}\right) + \sum_{I\left(y_{i_{\min}}\neq G(x_i)\right)} w_{i_{\min}} \exp\ \left(\beta_{\min}\right) = \\
&-\sum_i \left(1 - I\left(y_{i_{\min}} \neq G(x_i)\right)\right) w_{i_{\min}} \exp\left(-\beta_{\min}\right) + \sum_i I\left(y_{i_{\min}} \neq G(x_i)\right) w_{i_{\min}} \exp\ \left(\beta_{\min}\right) = 0, \\
&\xrightarrow{\text{err}_{\min} = \sum_{i_{\min}} I\left(y_{i_{\min}} \neq G(x_i)\right)} \beta_{\min} = \frac{1}{2}\log\frac{1-\text{err}_{\min}}{\text{err}_{\min}}
\end{aligned} \tag{12}$$

For β_{maj}, we do the same calculation as (12). Thus, β_{maj} is calculated as follows:

$$\beta_{\text{maj}} = \frac{1}{2} log \frac{1-\text{err}_{\text{maj}}}{\text{err}_{\text{maj}}}. \tag{13}$$

Finally, the model and weights are updated according to equations (14) and (15), respectively.

$$f_m(x) = f_{m-1}(x) + \beta^{(m)} G^{(m)}(x), \tag{14}$$

$$\begin{aligned}
w_{i_{\min}} &= \exp\left(-y_{i_{\min}} \beta_{\min} f_{m-1}(x_i)\right), \\
w_{i_{\text{maj}}} &= \exp\left(-y_{i_{\text{maj}}} \beta_{\text{maj}} f_{m-1}(x_i)\right).
\end{aligned} \tag{15}$$

4. Experimental Setups

Our goal is to demonstrate that the proposed method can improve the performance of AdaBoost over imbalanced data sets. To do this, we conducted comprehensive experiments and compare our method with the state-of-the-art method on different data sets with different IR. In the following section, first, information about the data sets and the tuned parameters are presented and then the used metrics are presented.

4.1. Data Sets. In this study, four synthetic data sets and 30 imbalance data sets available at KEEL [40] are used to evaluate the proposed method. The distribution of these synthetic data sets is presented in Figure 1. Tables 1 and 2 show the features of data sets including the number of attributes, number of samples, and proportion of number of majority class samples to minority ones (IR). In experiments, the area under ROC and the precision-recall curve and geometric mean (G-Mean) estimations are obtained using 5-fold cross validation; each dataset is partitioned into five folds, with each fold containing an equal number of samples. Subsequently, the learning algorithms are trained on four of the folds and tested on one fold for each iteration.

4.2. Tuning of Parameters. The performance of an algorithm can be greatly influenced by the assigned parameter values. Modifying parameter values in algorithms can yield different solutions. This subsection focuses on parameter tuning, where values are selected in a manner that facilitates comparison with other methods under similar conditions.

To compare IMBoost with other methods, six methods, i.e., OUBoost, AdaCC SMOTEBoost, AdaCost, AdaC1, and AdaBoost are used. In Table 3, the related parameters for all methods are presented. Other parameters are adjusted based on the original paper.

4.3. Metrics. In this section, we introduce the utilized metrics, including the area under ROC and the precision-recall curve and geometric mean (G-Mean)

4.3.1. ROC and Precision-Recall AUC. Both ROC and precision-recall AUC (PR AUC) are used for imbalanced problems; however, PR AUC is more robust under imbalanced data sets [41]. ROC AUC measures the area under ROC curve, while PR AUC measures the area under precision and recall curve (equation (16)).

The ROC AUC represents the area under this curve and is a measure of how well your model can distinguish between minor and major instances. On the other hand, PR AUC focuses on the performance of the model within the positive class and is particularly relevant when dealing with imbalanced data sets.

$$\begin{aligned}
\text{ROC} - \text{AUC} &= \text{AUC}(\text{FPR}, \text{TPR}), \\
\text{PR} - \text{AUC} &= \text{AUC}(\text{precision}, \text{recall}).
\end{aligned} \tag{16}$$

4.3.2. G-Mean. G-mean (geometric mean) is an important metric for evaluating imbalanced classification problems. It focuses on measuring the balance between the performance of a classifier in both the majority and minority classes. It helps prevent overfitting of the majority class and

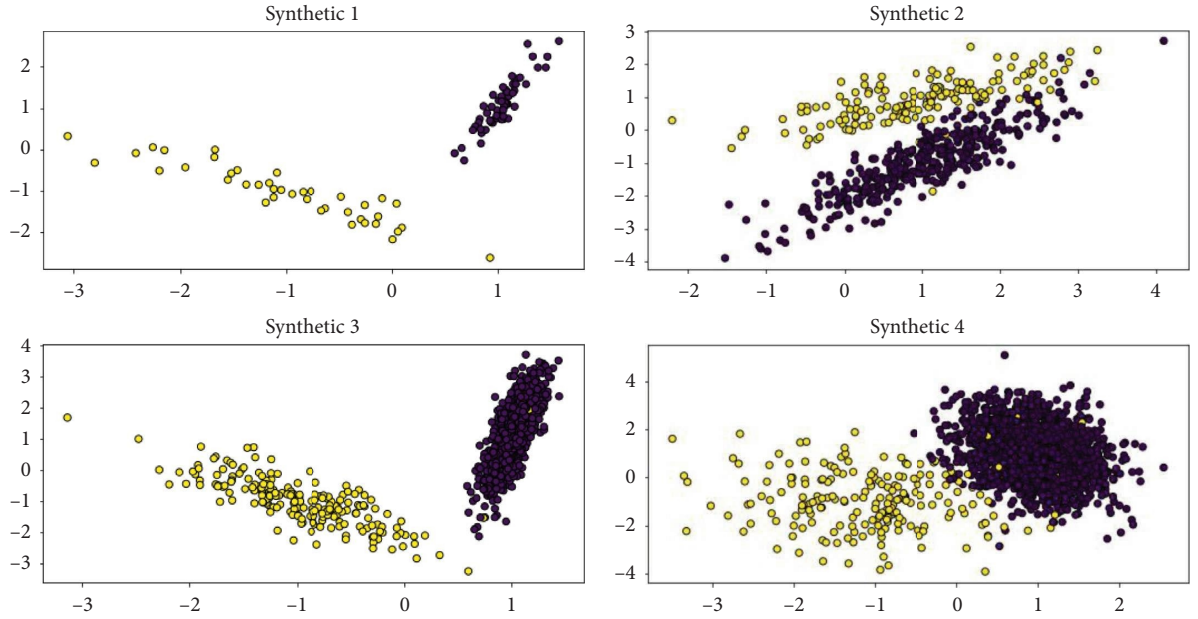

Figure 1: The distribution of synthetic data sets.

Table 1: The attributes of synthetic data sets.

Data sets	# of features	# of samples	IR
Synthetic 1	2	100	1.5
Synthetic 2	2	500	2.33
Synthetic 3	2	1000	4
Synthetic 4	2	2000	9

Table 2: Attributes of data sets.

#	Data set	# of features	# of samples	(IR)
1	glass1	9	214	1.82
2	ecoli-0_vs_1	7	220	1.86
3	Wisconsin	9	683	1.86
4	iris0	4	150	2
5	glass0	9	214	2.06
6	haberman	3	306	2.76
7	Vehicle3	18	846	2.99
8	glass-0-1-2-3_vs_4-5-6	9	214	3.2
9	Ecoli1	7	336	3.36
10	new-thyroid1	5	215	5.14
11	Ecoli2	7	336	5.46
12	segment0	19	2308	6.02
13	glass6	9	214	6.38
14	yeast3	8	1484	8.1
15	ecoli3	7	336	8.6
16	yeast-2_vs_4	8	514	9.08
17	yeast-0-5-6-7-9_vs_4	8	528	9.35
18	vowel0	13	988	9.98
19	glass2	9	214	11.59
20	glass4	9	214	15.47
21	ecoli4	7	336	15.8
22	page-blocks-1-3_vs_4	10	472	15.86
23	abalone9-18	8	731	16.4
24	yeast-1-4-5-8_vs_7	8	693	22.1
25	glass5	9	214	22.78
26	yeast-2_vs_8	8	482	23.1
27	yeast4	8	1484	28.1
28	yeast-1-2-8-9_vs_7	8	947	30.57
29	yeast5	8	1484	32.73
30	yeast6	8	1484	41.4

Table 3: The tuned parameters.

Methods	Parameters
IMBoost	Decision tree = C4.5 Depth of each tree = 5 Number of trees = 10
OUBoost	Decision tree = C4.5 Depth of each tree = 5 Number of trees = 10
SMOTEBoost	Decision tree = C4.5 Depth of each tree = 10 k-Nearest neighbors = 3 Number of trees = 100 Quantity = balance
AdaCost AdaC1 AdaBoost	Decision tree = C4.5 Depth of each tree = 10 Number of trees = 100
AdaCC	Decision tree = C4.5 Depth of each tree = 1 Number of trees = 100

underfitting of the minority class. A low G-mean indicates poor classification of positive samples, even if negative samples are classified correctly. G-mean is calculated using the following equation:

$$G - \text{Mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TN}}{\text{TN} + \text{FP}}}. \tag{17}$$

5. Results and Discussion

In this section, the results of the method are provided and compared. First of all, we investigate the performance of method on synthetic data sets. Then, we compare the outcome of the methods on 30 data sets.

5.1. Results on Synthetic Data Sets. To facilitate a more comprehensive comparison, we created four synthetic data sets, each with varying imbalanced ratios (IRs) and incorporating noise and outliers. In Tables 4–6, we present the performance of various methods based on ROC AUC, PR AUC, and G-mean. The results reveal that our method achieves the lowest ROC AUC score on synthetic 1 but attains the highest ROC AUC scores on synthetic data sets 2 and 3. In Figure 2, we depict the performance of different methods on synthetic data sets. It is evident that our method outperforms the others, especially on data sets with the high IR.

However, when considering the PR AUC, our method falls short on synthetics 2 and 4. Furthermore, when using the G-mean metric, our method exhibits slightly worse performance on synthetic 4 compared to other methods.

5.2. Results on KEEL Data Sets. The efficiency of IMBoost is reported in terms of ROC AUC, PR AUC, and G-mean in Tables 7–9. In Tables 7–9, each row corresponds to the results of OUBoost, AdaCC, SMOTEBoost, AdaCost, AdaC1, and AdaBoost over each data set, where the average of mentioned measures for 5-fold cross validation is reported for each method. The last row of the tables explains the average and the frequency of each method achieving the first rank.

In Table 7, with respect to the ROC AUC metric, our method achieves the first rank 13 times, while OUBoost and AdaCC hold the second-best rank. Figure 3 compares the average of obtained results on different data sets, and it depicts that the proposed method performs better than OUBoost, AdaCC, SMOTEBoost, AdaCost, AdaC1, and AdaBoost. Although OUBoost and AdaCC achieve more first rank in comparison with SMOTEBoost, SMOTEBoost outperformed them on average. ROC AUC serves as a standard evaluation measure, encompassing both positive (minority) and negative (majority) samples.

Consequently, precision-recall AUC (PR AUC) emerges as a valuable metric to assess the performances of methods, specifically concerning the minority class. In Table 8, we evaluate the performance of various methods based on PR AUC. Table 8 depicts that our method attains the top rank 12 times, while AdaCC secures the first rank in 8 data sets, highlighting the excellence of our method. However, when we consider the average results, there is not a significant disparity in the performance among these methods. In Figure 4, the performances of models are compared in terms of their averages.

Moreover, Table 9 depicts that our method outperforms other methods in terms of the G-mean metric. According to the results, our method obtains the first rank 13 times, indicating its superior performance. Figure 5 further illustrates that our method performs well on average.

In addition, Figure 6 depicts the performance of the top four methods, namely, our method, OUBoost, AdaCC, and SMOTEBoost, based on the IR. The regression line associated with each classifier illustrates how the performance of the classifier decreases with an increase in the IR. However, our method shows a significant reduction in performance with a steep slope. After ecoli3 (with an IR of 8.6), it demonstrates higher performance. In other words, our method performs well on data sets with a higher IR compared to the other methods.

Based on the results obtained from both the synthetic and KEEL data sets, we can conclude that our method exhibits superior performance on data sets with the high IR.

5.3. The Effect of Classifiers. In this section, we will discuss the impact of the number of classifiers on our method. Table 10 presents the results of our method with different numbers of classifiers, displaying the average results and the number of times it achieved the first rank. According to the findings, our method performs well with 10 classifiers and the addition of extra classifiers does not significantly improve its average performance. However, it is worth noting that our method with 20 and 30 classifiers outperforms the aforementioned methods on average.

Table 4: The average value of ROC AUC belongs to our method and six state-of-the-art methods on synthetic data sets.

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
Synthetic 1	91.20	**100.00**	92.26	**100.00**	**100.00**	**100.00**	98.00
Synthetic 2	**98.59**	89.92	98.45	89.92	89.03	89.03	86.56
Synthetic 3	**99.06**	98.75	96.44	98.43	98.75	98.75	95.21
Synthetic 4	97.90	92.68	**100.00**	96.72	92.12	92.12	90.10

Bold values represent the best result in each synthetic dataset.

Table 5: The average value of PR AUC belongs to our method and six state-of-the-art methods on synthetic data sets.

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
Synthetic 1	**94.00**	77.48	82.03	75.26	89.03	89.03	81.25
Synthetic 2	93.74	98.00	98.00	93.24	**98.75**	**98.75**	95.32
Synthetic 3	**100.00**	86.86	82.65	69.14	82.12	82.12	80.12
Synthetic 4	82.40	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	83.25

Bold values represent the best result in each synthetic dataset.

Table 6: The average value of G-mean belongs to our method and six state-of-the-art methods on synthetic data sets.

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
Synthetic 1	**98.58**	89.88	92.24	89.17	88.88	88.88	85.0
Synthetic 2	**98.81**	98.74	98.44	98.12	98.74	98.74	96.25
Synthetic 3	**97.13**	92.39	96.43	93.39	91.87	91.87	91.25
Synthetic 4	99.21	**100.0**	**100**	**100.0**	**100.0**	**100.0**	98.85

Bold values represent the best result in each synthetic dataset.

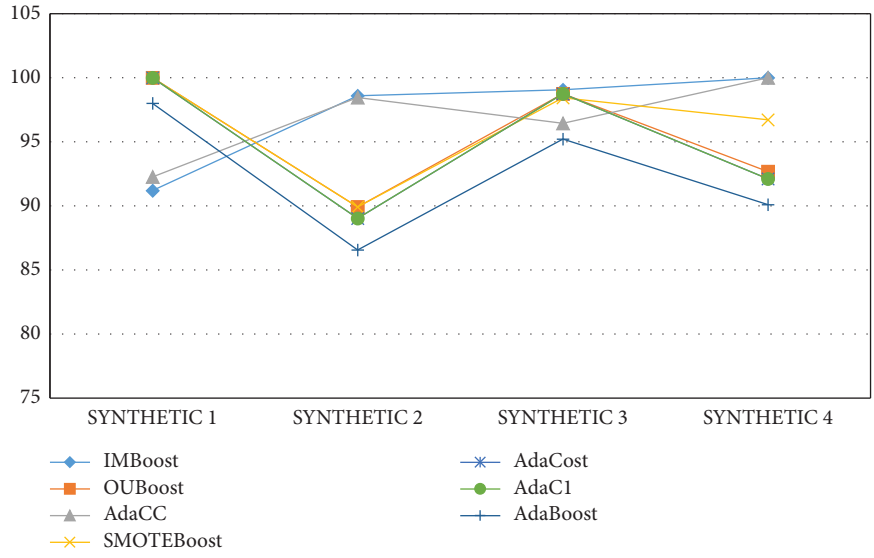

Figure 2: ROC AUC vs. IR.

Furthermore, it is important to note that our method with different numbers of classifiers still performs poorly on specific data sets, namely, glass1, Wisconsin, glass0, vehicle3, ecoli2, glass2, page-blocks-1-3_vs_4, and abalone9-18. However, when utilizing 20 classifiers, our method demonstrates improved performance on haberman and glass-0-1-2-3_vs_4-5-6. In addition, with 50 classifiers, it performs well on vowel0 and ecoli4, and with 100 classifiers, it achieves good results on new-thyroid1.

Figure 7 illustrates the impact of the number of classifiers, with the x-axis representing the number of classifiers and the y-axis indicating the average results across data sets. As depicted, the performance of our method decreases with an increase in the number of classifiers. These results suggest that adding more classifiers is not suitable for enhancing the performance of IMBoost.

5.4. Time Complexity. For further analysis, we investigate the time complexity of our method and compare it with other methods. Table 11 provides a comparison of the time complexity of the methods, with the execution time presented in seconds. It is noteworthy that our method, which includes 10 classifiers, exhibits the worst execution time among the methods. In

Table 7: Comparison of the ROC AUC of different methods (the better ones are bold).

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
glass1	71.68	**76.83**	74.24	71.64	75.59	75.59	71.95
ecoli-0_vs_1	**97.64**	**97.64**	96.23	97.59	96.56	96.56	95.16
wisconsin	95.00	**96.14**	95.70	93.73	93.71	93.71	92.43
iris0	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
glass0	75.03	**81.18**	77.0	80.92	79.53	79.53	76.65
haberman	56.11	**56.96**	55.25	56.66	55.9	51.83	51.12
vehicle3	69.23	**70.81**	63.83	72.63	66.79	68.06	65.93
glass-0-1-2-3_vs_4-5-6	87.19	**91.86**	88.86	91.62	90.93	90.93	92.46
ecoli1	**87.41**	86.37	86.27	83.96	83.60	83.41	81.60
new-thyroid1	94.92	90.04	**97.14**	95.15	93.17	93.17	92.30
ecoli2	86.92	88.31	**89.92**	85.34	85.2	85.2	84.20
segment0	99.26	99.26	**99.47**	99.03	99.21	99.21	98.63
glass6	**91.75**	90.52	90.52	85.77	82.98	82.98	79.96
yeast3	**90.94**	86.45	89.39	87.76	86.45	76.44	83.61
ecoli3	**81.33**	78.5	77.60	76.68	72.23	70.8	72.89
yeast-2_vs_4	**91.50**	82.59	86.54	87.84	83.67	83.61	82.85
yeast-0-5-6-7-9_vs_4	**77.40**	66.77	56.04	74.56	66.62	67.73	65.36
vowel0	94.16	93.16	**95.38**	95.33	91.77	91.77	92.66
glass2	64.79	**67.95**	64.36	63.07	62.63	62.13	60.10
glass4	**85.08**	84.50	84.50	83.75	84.25	84.25	80.10
ecoli4	86.21	84.52	84.36	**89.05**	81.70	81.70	86.55
page-blocks-1-3_vs_4	99.32	99.66	**99.88**	99.77	98.10	98.10	97.88
abalone9-18	76.77	64.78	77.04	**79.98**	67.12	67.12	72.79
yeast-1-4-5-8_vs_7	**59.05**	49.92	57.45	51.68	55.83	55.15	55.54
glass5	**95.85**	89.75	80.0	89.75	89.75	89.75	89.75
yeast-2_vs_8	**81.31**	67.39	76.63	74.68	71.41	71.41	66.74
yeast4	**76.23**	57.56	67.75	66.92	58.66	61.63	64.56
yeast-1-2-8-9_vs_7	62.49	56.61	**63.04**	55.19	56.28	57.29	61.69
yeast5	93.26	85.59	**94.16**	85.55	84.61	84.89	84.3
yeast6	78.32	75.43	**82.42**	76.03	73.38	77.67	71.64
Average/1st rank	**83.53/13**	80.56/9	81.69/9	81.72/3	79.58/1	79.38/1	79.04/1

Table 8: Comparison of the PR AUC of different methods (the better ones are bold).

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
glass1	50.94	57.43	57.09	54.51	**62.15**	57.09	56.02
ecoli-0_vs_1	**94.68**	92.96	91.71	93.89	93.66	93.66	91.98
wisconsin	**90.51**	91.60	90.99	88.58	88.64	88.64	87.54
iris0	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
glass0	**62.22**	55.69	57.80	53.08	61.97	61.97	58.23
haberman	28.74	30.10	**34.36**	29.41	30.34	27.05	25.31
vehicle3	37.50	32.69	43.90	**45.81**	40.21	41.14	36.24
glass-0-1-2-3_vs_4-5-6	79.82	79.69	78.26	76.67	**79.95**	**79.95**	76.87
ecoli1	**68.07**	59.71	65.38	64.30	63.80	62.15	59.58
new-thyroid1	**96.87**	80.65	95.21	80.82	81.31	81.31	82.07
ecoli2	55.53	52.99	**60.93**	52.54	55.37	55.37	48.72
segment0	96.50	97.61	97.61	**97.70**	96.91	96.91	96.07
glass6	74.66	51.12	**76.67**	64.80	65.29	65.29	64.51
yeast3	52.61	56.21	**57.81**	53.11	54.10	52.58	55.41
ecoli3	**38.13**	37.76	37.14	36.47	31.24	31.24	30.30
yeast-2_vs_4	56.81	51.86	57.41	**59.38**	58.87	56.31	55.24
yeast-0-5-6-7-9_vs_4	**29.42**	22.81	28.15	29.28	24.25	25.81	23.16
vowel0	**86.78**	83.16	88.06	82.91	79.88	79.88	80.20
glass2	**27.82**	19.06	23.13	24.19	21.52	21.52	19.87
glass4	59.33	48.99	**65.19**	47.32	55.74	55.74	55.32
ecoli4	53.41	49.65	59.87	**62.73**	53.33	53.33	58.45
page-blocks-1-3_vs_4	82.44	89.28	**96.66**	94.28	94.28	94.28	93.68
abalone9-18	28.87	**29.62**	32.45	26.18	25.02	24.71	24.12
yeast-1-4-5-8_vs_7	06.55	06.66	06.04	**08.71**	06.70	06.70	05.92

Table 8: Continued.

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
glass5	68.33	**83.79**	74.26	61.86	74.26	74.26	74.26
yeast-2_vs_8	27.33	24.15	**36.58**	19.11	31.98	31.98	31.08
yeast4	13.63	13.06	13.47	**14.57**	08.32	10.62	09.65
yeast-1-2-8-9_vs_7	05.20	11.25	06.19	05.61	**11.91**	05.77	06.89
yeast5	**61.00**	43.02	49.83	51.94	53.63	51.41	56.28
yeast6	**26.93**	24.16	22.17	20.85	25.37	25.37	24.57
Average/1st rank	55.35/12	52.55/3	56.81/8	53.35/7	54.33/4	53.73/2	52.91/1

Table 9: Comparison of the G-mean of different methods (the better ones are bold).

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
glass1	70.56	72.75	73.66	71.50	**74.90**	**74.90**	70.83
ecoli-0_vs_1	**97.61**	96.53	96.19	96.90	96.48	96.48	97.24
wisconsin	94.94	95.59	**95.67**	93.76	93.64	93.64	92.34
iris0	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
glass0	74.44	**91.63**	76.03	89.63	90.74	90.74	75.87
haberman	**55.03**	51.58	24.98	48.00	51.13	44.11	43.81
vehicle3	68.38	38.26	43.39	**66.89**	62.84	65.60	60.20
glass-0-1-2-3_vs_4-5-6	87.07	75.27	88.26	71.36	79.33	79.33	**92.27**
ecoli1-5	**87.08**	80.99	85.99	85.99	83.12	82.92	80.84
new-thyroid1	94.81	92.21	**97.03**	91.59	92.92	92.92	91.61
ecoli2	86.50	82.46	**89.72**	87.28	84.69	84.69	83.52
segment0	99.26	98.75	99.46	**99.54**	99.21	99.21	98.63
glass6	**91.62**	78.73	90.17	87.48	81.63	81.63	78.29
yeast3	**90.88**	79.31	89.23	83.28	85.83	65.59	82.36
ecoli3	**80.75**	74.72	74.42	75.86	68.10	66.36	68.84
yeast-2_vs_4	**91.45**	82.08	85.82	86.74	82.19	82.19	81.32
yeast-0-5-6-7-9_vs_4	**76.07**	59.04	15.70	70.73	59.36	61.07	57.68
vowel0	93.94	93.24	**95.12**	94.29	91.33	91.33	92.19
glass2	**53.47**	36.19	49.88	60.91	46.35	46.35	46.98
glass4	**82.68**	76.37	81.6	77.24	82.47	82.47	76.11
ecoli4	85.18	83.31	80.87	**85.32**	78.00	78.00	83.73
page-blocks-1-3_vs_4	99.32	99.55	**99.88**	99.77	98.03	98.03	97.81
abalone9-18	**74.49**	63.33	71.34	66.93	57.87	57.87	66.98
yeast-1-4-5-8_vs_7	**52.47**	34.95	47.01	32.38	27.59	22.56	35.08
glass5	**95.66**	93.90	68.28	79.75	79.75	79.75	79.75
yeast-2_vs_8	78.06	55.04	71.66	**78.41**	58.11	58.11	51.07
yeast4	**73.36**	54.74	55.2	65.87	42.60	49.72	53.97
yeast-1-2-8-9_vs_7	54.05	51.57	**56.22**	37.62	22.13	35.53	49.37
yeast5	92.76	79.48	**94.07**	88.31	83.48	83.51	82.23
yeast6	**84.76**	67.80	80.39	64.13	67.20	71.96	66.08
Average/1st rank	**82.13/16**	74.88/2	75.41/9	78.39/5	74.27/2	73.95/2	74.86/2

contrast, AdaC1 demonstrates the lowest average execution time, followed by AdaBoost as the second best. However, their performance is not satisfactory compared to other methods.

Based on the results, it is evident that increasing the number of classifiers in our method leads to higher time complexity. Therefore, since a larger number of classifiers does not contribute to an improvement in the performance of our method, but increases the time complexity, it is not advisable to increase the number of classifiers.

5.5. Nonparametric Statistical Tests. To assess the presence of a significant difference between the methods, statistical tests are conducted on the obtained results. To conduct these tests, we employ a two-stage approach proposed by Desmar [42].

In the initial stage, a statistical test based on the ranking of algorithms according to their performance is conducted. The null hypothesis assumes that the algorithms have equal performance. Rejecting the null hypothesis indicates statistically significant differences in the performance of the algorithms.

In the preceding step, the algorithm with the highest rank is identified as the "control algorithm." It is subsequently compared to other algorithms in pairwise comparisons using various nonparametric post hoc statistical tests, such as Holm [43], Hochberg [44], and Hommel [45].

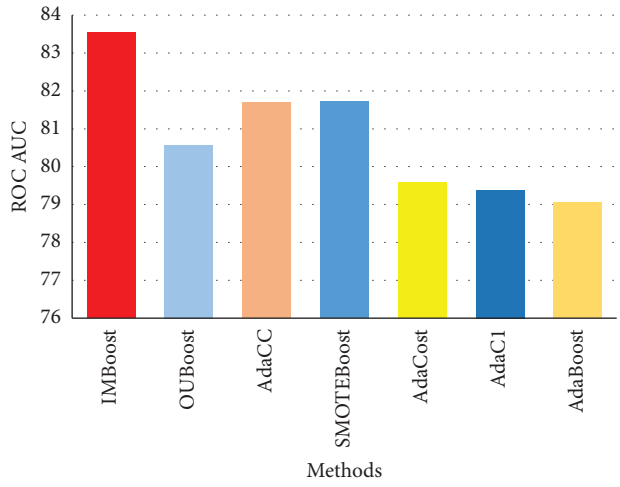

Figure 3: The average results of ROC AUC over methods.

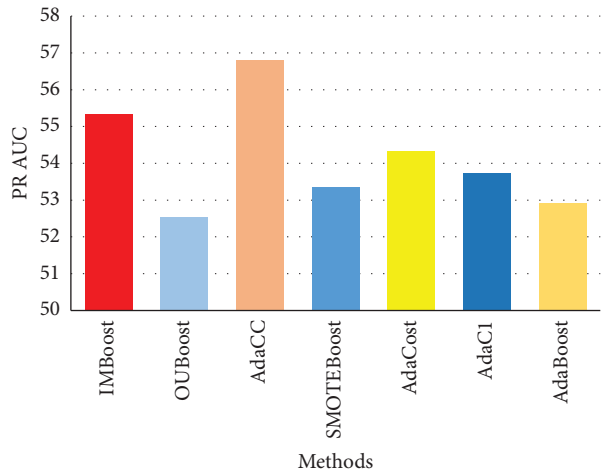

Figure 4: The average results of PR AUC over methods.

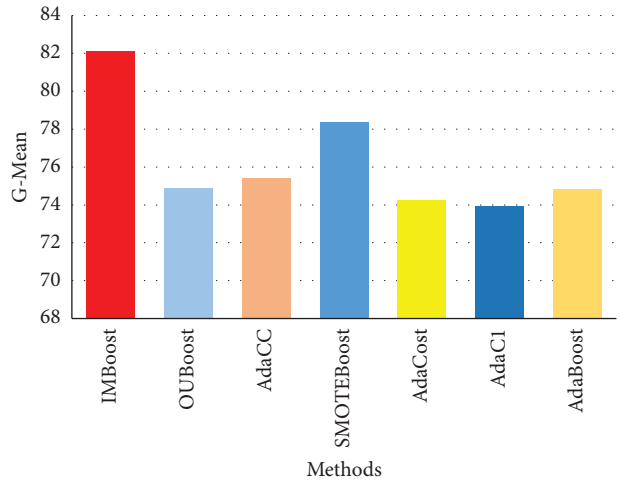

Figure 5: The average results of G-mean over methods.

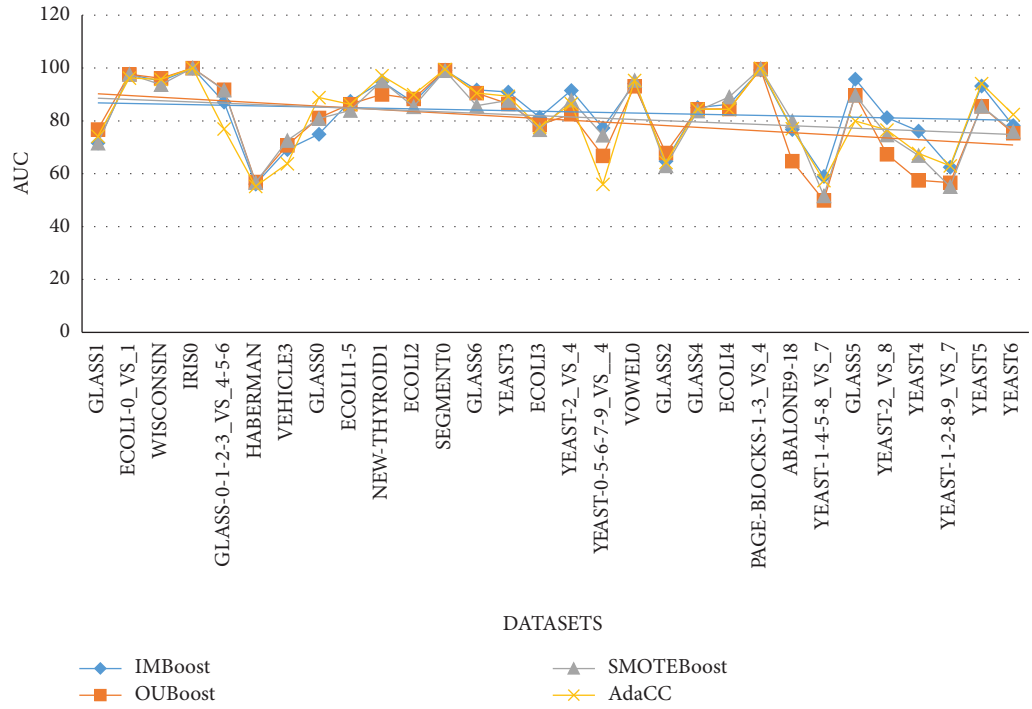

Figure 6: Regression line to fit the AUC and the IR.

Table 10: The average value of AUC belongs to our method with different number of classifiers.

Data set	10	20	30	40	50	70	80	100	150
glass1	**71.68**	62.72	61.1	63.56	61.77	62.68	64.31	52.41	63.29
ecoli-0_vs_1	97.64	96.25	97.65	98.64	**98.66**	96.98	97.31	89.74	95.65
wisconsin	**95**	94.06	93.62	94.05	94.66	94.22	92.33	94.34	92.9
iris0	**100**	**100**	99.5	**100**	**100**	99	**100**	93	98.5
glass0	**75.03**	63.41	72.56	71.68	55.92	61.35	57.84	62.18	56.05
haberman	56.11	**58.63**	53.21	56.25	55.14	51.76	53.13	44.82	53.06
vehicle3	**69.23**	69.1	67.25	62.74	64.04	54.95	60.47	56.91	53.76
glass-0-1-2-3_vs_4-5-6	87.19	**94.92**	91.08	88.16	90.03	87.09	83.4	85.94	90.26
ecoli1	**87.41**	80.79	83.13	83.7	66.53	61.51	71.52	79.34	72.27
new-thyroid1	94.92	93.25	92.06	94.04	89.88	95.23	90.43	**96.62**	90.43
ecoli2	**86.92**	82.09	85.97	84.87	83.23	80.92	72.57	75.39	70.63
segment0	**99.26**	98.71	98.5	98.71	97.64	97.24	97.54	98.07	96.66
glass6	91.75	84.05	**91.8**	88.82	90.13	91.17	90.36	79.77	90.36
yeast3	**90.94**	89	87.23	82.83	77.53	73.29	72.27	68.81	57.4
ecoli3	81.33	82.42	**82.76**	69.67	77.53	67.04	66.6	72.79	66.34
yeast-2_vs_4	**91.5**	89.65	87.98	89.55	84.13	81.8	88.17	80	83.66
yeast-0-5-6-7-9_vs_4	**77.4**	72.73	75.38	67.88	70.93	75.45	59.86	62.31	64.68
vowel0	94.16	94.44	94.22	89.33	**96.54**	92.26	92.54	92.6	90.69
glass2	64.79	62.64	64.17	**67.8**	58.51	57.45	58.41	61.69	52.03
glass4	85.08	85.18	**82.72**	85	80.95	80.39	77.69	71.03	87.31
ecoli4	86.21	86.35	86.51	**89.17**	86.64	87.41	86.95	80.81	84.36
page-blocks-1-3_vs_4	**99.32**	99.21	96.73	97.63	98.76	94.24	86.83	87.29	82.2
abalone9-18	**76.77**	72.31	76.61	68.56	65.59	72.15	61.62	63.52	54.33
yeast-1-4-5-8_vs_7	59.05	61.8	58.63	**66.73**	52.23	50.84	52.98	54.24	53.97
glass5	**95.85**	95.12	94.39	92.19	83.9	89.75	89.75	79.89	91.7
yeast-2_vs_8	81.31	**81.55**	80.13	75.35	66.46	71.57	61.41	58.35	51.53
yeast4	76.23	73	77.13	68.5	77.43	**77.53**	63.73	63.94	49.21
yeast-1-2-8-9_vs_7	62.49	64.24	66.57	50.09	**68.42**	62.41	62.79	59.17	63.57
yeast5	**93.26**	92.43	91	86.14	86.45	87.32	88.16	81.8	78.47
yeast6	78.32	80.75	79.47	**83.44**	75.52	74.06	77.73	71.89	72.25
Average/1st rank	**83.53/15**	82.02/4	82.30/3	80.5/5	78.5/4	77.63/1	75.95/1	73.95/1	73.58/0

Based on the number of trees, bold values represent the best result in each synthetic dataset.

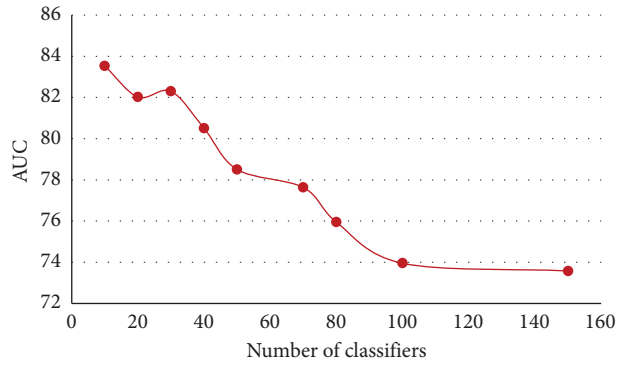

Figure 7: The effect of number of classifiers on our method performance.

Table 11: The time complexity of our method compared to 5 state-of-the-art methods.

Data set	IMBoost	OUBoost	AdaCC	SMOTEBoost	AdaCost	AdaC1	AdaBoost
glass1	0.680148	0.311335	0.814370	0.402105	0.006436	0.006353	0.008269
ecoli-0_vs_1	0.711841	0.40913	0.949293	0.617847	0.01316	0.008869	0.006247
wisconsin	2.334817	0.176958	0.969636	0.480909	0.007351	0.006742	0.005681
iris0	0.555059	0.00806	0.854523	0.308105	0.00906	0.005942	0.005438
glass-0-1-2-3_vs_4-5-6	0.701264	0.144232	0.958188	0.477068	0.005527	0.005469	0.006646
haberman	0.903955	0.188589	0.958188	0.383278	0.231583	0.069137	0.248758
vehicle3	3.001377	0.292414	0.499649	1.43819	0.597517	0.63525	0.774126
glass0	0.682704	0.176531	0.846364	0.416247	0.008579	0.007279	0.006417
ecoli1	0.995574	0.11928	0.929662	0.533222	0.009847	0.005569	0.00906
new-thyroid1	0.894101	0.139003	0.994520	0.479737	0.019937	0.025586	0.005324
ecoli2	0.995574	0.128287	0.951977	0.4296	0.005504	0.005569	0.006852
segment0	7.356576	0.340925	0.998483	3.546493	0.015306	0.015353	0.018524
glass6	0.670612	0.159746	0.978450	0.378248	0.005642	0.005635	0.006109
yeast3	4.550099	0.219679	0.960383	1.484923	0.197325	0.088621	0.05314
ecoli3	0.97971	0.136388	0.938859	0.455026	0.009862	0.008136	0.006105
yeast-2_vs_4	1.731223	0.128514	0.969571	0.467827	0.008302	0.006939	0.005811
yeast-0-5-6-7-9_vs_4	1.661534	0.101199	0.192670	0.560591	0.00937	0.008313	0.005915
vowel0	3.49743	0.164919	0.993348	1.59756	0.018093	0.020025	0.010061
glass2	0.677648	0.133019	0.948784	0.451205	0.00899	0.009994	0.006205
glass4	0.676413	0.102536	0.985117	0.44172	0.008662	0.009171	0.005702
ecoli4	0.981345	0.107385	0.984244	0.58351	0.007583	0.008247	0.007122
page-blocks-1-3_vs_4	2.69899	0.112687	0.998870	0.63347	0.011839	0.006191	0.006757
abalone9-18	2.172811	0.11797	0.963277	0.948631	0.014928	0.014531	0.016362
yeast-1-4-5-8_vs_7	2.135579	0.12574	0.920830	0.719552	0.077993	0.044838	0.008405
glass5	0.669755	0.131695	0.990418	0.46357	0.008244	0.005624	0.006244
yeast-2_vs_8	1.58693	0.118113	0.981532	0.554371	0.009059	0.007046	0.006161
yeast4	4.226671	0.152353	0.764113	1.209537	0.02398	0.024508	0.026595
yeast-1-2-8-9_vs_7	2.955871	0.19862	0.933641	0.910121	0.174815	0.030466	0.012869
yeast5	4.137762	0.159	0.985930	1.086509	0.016726	0.013052	0.006679
yeast6	4.187912	0.20405	0.977180	1.31572	0.035213	0.014928	0.0077483
Average	2.00037614	0.166945	0.906402	0.792496	0.052548	0.037446	0.043511

Table 12 presents the ranking results based on the ROC AUC using the Friedman test. The p value, which is smaller than the significance level, indicates the rejection of the null hypothesis, demonstrating significant differences in performance among the algorithms used. Our method ranks first among the other algorithms in terms of ROC AUC and is, therefore, designated as the "control algorithm." Subsequently, this algorithm is pairwise compared to the other employed algorithms using post hoc statistical tests, including Holm, Hochberg, and Hommel. The results of these tests are reported in Table 13.

Table 12: Average rankings of the algorithms based on the ROC AUC (P value computed by Friedman test = 0 and Friedman statistic = 49.175).

Methods	Ranking
IMBoost	2.1333
AdaCC	3.15
SMOTEBoost	3.667
OUBoost	4.6333
AdaCost	4.7
AdaC1	4.7667
AdaBoost	5.25

Table 13: Post hoc comparison table for $\alpha = 0.05$ based on the ROC AUC.

Algorithm	p value	Holm/Hochberg/Hommel	Hypothesis
SMOTEBoost vs. IMBoost	0.027024	0.025	Rejected
OUBoost vs. IMBoost	0.000007	0.016667	Rejected
AdaC1 vs. IMBoost	0.000002	0.01	Rejected
AdaCost vs. IMBoost	0.000004	0.0125	Rejected
AdaBoost vs. IMBoost	0	0.008333	Rejected
AdaCC vs. IMBoost	0.068345	0.05	Not rejected

Table 14: The obtained results of Wilcoxon signed-rank test based on the ROC AUC. W^+ corresponds to our method and W^- to AdaCC.

Comparison	W^+	W^-	Hypothesis	p value
IMBoost vs. AdaCC	279	156	Not rejected	0.1901

Table 15: Average rankings of the algorithms based on the PR AUC (P value computed by Friedman test = 0 and Friedman statistic = 25.557143).

Methods	Ranking
IMBoost	3.3
AdaCC	2.88
SMOTEBoost	4.13
OUBoost	4.58
AdaCost	3.65
AdaC1	4.2
AdaBoost	5.25

Table 16: Post hoc comparison table for $\alpha = 0.05$ based on the PR AUC.

Algorithms	p value	Holm/Hochberg/Hommel	Hypothesis
SMOTEBoost vs. AdaCC	0.025023	0.016667	Rejected
OUBoost vs. AdaCC	0.002305	0.01	Rejected
AdaC1 vs. AdaCC	0.018247	0.0125	Rejected
AdaCost vs. AdaCC	0.169283	0.025	Rejected
AdaBoost vs. AdaCC	0.000022	0.008333	Rejected
IMBoost vs. AdaCC	0.455053	0.05	Not rejected

Table 17: The obtained results of Wilcoxon signed-rank test based on the PR AUC. W^- corresponds to our method and W^+ to AdaCC.

Comparison	W^+	W^-	Hypothesis	p value
IMBoost vs. AdaCC	290	145	Not rejected	1207

Table 18: Average rankings of the algorithms based on G-mean (P value computed by Friedman test = 0 and Friedman statistic = 50.067857).

Methods	Ranking
IMBoost	2.5
AdaCC	3.1
SMOTEBoost	3.4667
OUBoost	3.5167
AdaCost	4.9
AdaC1	4.9167
AdaBoost	5.6

Table 13, at a significance level of 0.05, shows that the control algorithm outperforms the other algorithms pairwise, except for AdaCC. To gain insight into how our method performs compared to AdaCC, we conducted a pairwise comparison using the Wilcoxon signed-rank test at a significance level of 0.05. As indicated in Table 14, the p value exceeds the 0.05 significance level. In other words, the observed difference between the sample change and the expected change is not substantial enough to be considered statistically significant. Consequently, concerning the ROC AUC metric, there is only a marginal difference between our method and AdaCC. However, when considering the average performance across all data sets, our method surpasses AdaCC, and on most of the data sets, IMBoost demonstrates superior performance. In the identical experiments using the PR AUC, as shown in Table 15, AdaCC secures the top position and is designated as the "control algorithm." However, the outcomes of the Holm, Hochberg, and Hommel tests, as presented in Table 16, reveal that AdaCC

Table 19: Post hoc comparison table for $\alpha = 0.05$ based on G-mean.

Algorithm	p value	Holm/Hochberg/Hommel	Hypothesis
SMOTEBoost vs. IMBoost	0.083081	0.025	Rejected
OUBoost vs. IMBoost	0.068345	0.016667	Rejected
AdaC1 vs. IMBoost	0.000015	0.01	Rejected
AdaCost vs. IMBoost	0.000017	0.0125	Rejected
AdaBoost vs. IMBoost	0	0.008333	Rejected
AdaCC vs. IMBoost	0.282059	0.05	Not rejected

Table 20: The obtained results of Wilcoxon signed-rank test based on G-mean. W^+ corresponds to our method and W^- to AdaCC.

Comparison	W^+	W^-	Hypothesis	p value
IMBoost vs. AdaCC	338	97	Rejected	0.008008

does not outperform our method. In Table 17, the Wilcoxon signed-rank test, conducted at a significance level of 0.05, indicates no significant difference between our method and AdaCC.

In addition, we conducted the Friedman test based on G-mean and the results are presented in Table 18. According to these results, our method claims the first rank and is designated as the "control algorithm." Moreover, the outcomes of the Holm, Hochberg, and Hommel tests in Table 19 indicate that our method performs better than the other methods at a significance level of 0.05, with the exception of AdaCC. Consequently, the Wilcoxon test is used at a significance level of 0.05. As shown in Table 20, based on the G-mean metric, our method outperforms the other methods.

Overall, the evaluation conducted using the Friedman test reveals that the proposed method attains a higher rank compared to other techniques, with AdaCC securing the second rank. Subsequent post hoc tests confirm that the employed algorithms exhibit varying performance in terms of G-mean, and our method demonstrates the best performance.

6. Conclusion

Most studies addressing imbalanced problems commonly employ over- and undersampling techniques. However, these methods may introduce noisy data or discard important information, respectively. In this paper, we propose a novel method to enhance the performance of AdaBoost for imbalanced data sets. Our approach involves initializing the weights of minority and majority samples based on their distribution. Subsequently, the weights are updated according to the error of classifier on minority and majority samples separately. The data set is then resampled based on these updated weights, and this process is iterated.

To evaluate the effectiveness of our method, we compare it with six ensemble methods on 34 data sets. The performance of these methods is measured using the ROCAUC, PR AUC, and G-mean metrics. The results based on these metrics demonstrate that IMBoost outperforms the others, consistently achieving the highest rank on most data sets. In addition, our method exhibits strong performance on data sets with high imbalance ratios. According to statistical tests, which included both PR and ROC AUC measures, no significant difference was observed between AdaCC and IMBoost. However, when we applied the G-mean measure, it solidified our method's excellence. However, it is important to note that our method has a drawback in terms of time complexity since the increase in samples leads to higher computational requirements.

For future studies, extending IMBoost to multiclass data sets or semisupervised problems could be explored. In addition, applying metaheuristics and comparing the results with our method could be considered as a promising further research.

Data Availability

The data are publicly available.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] X. Gong and W. Qiao, "Imbalance fault detection of direct-drive wind turbines using generator current signals," *IEEE Transactions on Energy Conversion*, vol. 27, no. 2, pp. 468–476, 2012.

[2] J. Kong, W. Kowalczyk, S. Menzel, and T. Bäck, "Improving imbalanced classification by anomaly detection," in *Parallel Problem Solving from Nature--PPSN XVI: 16th International Conference, PPSN 2020*, pp. 512–523, Springer, Leiden, Netherlands, 2020.

[3] A. Namvar, M. Siami, F. Rabhi, and M. Naderpour, "Credit Risk Prediction in an Imbalanced Social Lending Environment," 2018, https://arxiv.org/abs/1805.00801.

[4] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *Journal of Biomedical Informatics*, vol. 90, Article ID 103089, 2019.

[5] M. A. U. H. Tahir, S. Asghar, A. Manzoor, and M. A. Noor, "A classification model for class imbalance dataset using genetic programming," *IEEE Access*, vol. 7, pp. 71013–71037, 2019.

[6] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, "SMOTE-RSB ∗: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory," *Knowledge and Information Systems*, vol. 33, no. 2, pp. 245–265, 2012.

[7] D. Colton and M. Hofmann, "Sampling techniques to overcome class imbalance in a cyberbullying context," *Journal of Computer-Assisted Linguistic Research*, vol. 3, no. 1, pp. 21–40, 2019.

[8] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *J. Big Data*, vol. 7, no. 1, pp. 70–47, 2020.

[9] C. X. Ling and V. S. Sheng, "Cost-sensitive learning and the class imbalance problem," *Encyclopedia of Machine Learning*, vol. 2011, pp. 231–235, 2008.

[10] S. E. Roshan and S. Asadi, "Improvement of Bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization," *Engineering Applications of Artificial Intelligence*, vol. 87, Article ID 103319, 2020.

[11] B. Krawczyk, M. Galar, Ł. Jeleń, and F. Herrera, "Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy," *Applied Soft Computing*, vol. 38, pp. 714–726, 2016.

[12] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, Chapman & Hall/CRC, London, UK, 1st edition, 2012.

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

[14] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Icml*, vol. 96, pp. 148–156, 1996.

[15] O. Sagi and L. Rokach, "Ensemble learning: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, 2018.

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

[17] N. V Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 107–119, Springer, Berlin, Heidelberg, 2003.

[18] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: a hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[19] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 39, no. 2, pp. 539–550, 2009.

[20] V. Iosifidis, S. Papadopoulos, B. Rosenhahn, and E. Ntoutsi, "AdaCC: cumulative cost-sensitive boosting for imbalanced classification," *Knowledge and Information Systems*, vol. 65, no. 2, pp. 789–826, 2023.

[21] B. Yuan and X. Ma, "Sampling+ reweighting: boosting the performance of AdaBoost on imbalanced datasets," in *Proceedings of the The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Brisbane, QLD, June 2012.

[22] S. H. Mostafaei and J. Tanha, "OUBoost: boosting based over and under sampling technique for handling imbalanced data," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 10, pp. 3393–3411, 2023.

[23] S.-W. Ke, C.-F. Tsai, Y.-Y. Pan, and W.-C. Lin, "Majority resampling via sub-class clustering for imbalanced datasets," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 15, pp. 1–16, 2023.

[24] J. Zhai, J. Qi, and S. Zhang, "Imbalanced data classification based on diverse sample generation and classifier fusion," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 3, pp. 735–750, 2022.

[25] A. Puri and M. Kumar Gupta, "Improved hybrid bag-boost ensemble with K-means-SMOTE--ENN technique for handling noisy class imbalanced data," *The Computer Journal*, vol. 65, no. 1, pp. 124–138, 2022.

[26] G. Wang, J. Wang, and K. He, "Majority-to-minority resampling for boosting-based classification under imbalanced data," *Applied Intelligence*, vol. 53, no. 4, pp. 4541–4562, 2023.

[27] A. Arafa, N. El-Fishawy, M. Badawy, and M. Radad, "RN-SMOTE: Reduced noise smote based on DBSCAN for enhancing imbalanced data classification," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5059–5074, 2022.

[28] J. Li, Y. Wu, S. Fong et al., "A binary PSO-based ensemble under-sampling model for rebalancing imbalanced training data," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 7428–7463, 2022.

[29] J. Dong and Q. Qian, "A density-based random forest for imbalanced data classification," *Future Internet*, vol. 14, no. 3, p. 90, 2022.

[30] Q. Gu, J. Tian, X. Li, and S. Jiang, "A novel Random Forest integrated model for imbalanced data classification problem," *Knowledge-Based Systems*, vol. 250, Article ID 109050, 2022.

[31] J. Zhao, J. Jin, S. Chen, R. Zhang, B. Yu, and Q. Liu, "A weighted hybrid ensemble method for classifying imbalanced data," *Knowledge-Based Systems*, vol. 203, Article ID 106087, 2020.

[32] R. F. A. B. De Morais and G. C. Vasconcelos, "Boosting the performance of over-sampling algorithms through under-sampling the minority class," *Neurocomputing*, vol. 343, pp. 3–18, 2019.

[33] G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, "BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification," *Engineering Applications of Artificial Intelligence*, vol. 49, pp. 176–193, 2016.

[34] C. Piao, N. Wang, and C. Yuan, "Rebalance weights AdaBoost-SVM model for imbalanced data," *Computational Intelligence and Neuroscience*, vol. 2023, Article ID 4860536, 26 pages, 2023.

[35] 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.

[36] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: misclassification cost-sensitive boosting," *Icml*, vol. 99, pp. 97–105, 1999.

[37] 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.

[38] L. Hao and G. Huang, "An improved AdaBoost algorithm for identification of lung cancer based on electronic nose," *Heliyon*, vol. 9, no. 3, Article ID e13633, 2023.

[39] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[40] J. Derrac, S. Garcia, L. Sanchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of

algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, 2015.

[41] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, Pittsburgh, PE, USA, August 2006.

[42] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[43] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, pp. 65–70, 1979.

[44] Y. Hochberg, "A sharper Bonferroni procedure for multiple tests of significance," *Biometrika*, vol. 75, no. 4, pp. 800–802, 1988.

[45] G. Hommel, "A stagewise rejective multiple test procedure based on a modified Bonferroni test," *Biometrika*, vol. 75, no. 2, pp. 383–386, 1988.