IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 November 2022, accepted 25 November 2022, date of publication 28 November 2022,
date of current version 5 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225456

RESEARCH ARTICLE

MPSUBoost: A Modified Particle Stacking Undersampling Boosting Method

SANG-JIN KIM AND DONG-JOON LIM
Department of Industrial Engineering, Sungkyunkwan University, Suwon 16419, South Korea
Corresponding author: Dong-Joon Lim (tgno3@skku.edu)

This work was supported by the National Research Foundation of Korea, South Korea, under Grant 2020R1F1A1066629.

ABSTRACT Class imbalance problems are prevalent in the real world. In such cases, traditional supervised algorithms tend to have difficulty in recognizing minority data because the models are likely to maximize prediction accuracy by simply ignoring minority data. To address the class imbalance problem, various approaches have been tried, including data preprocessing techniques, cost-sensitive learning, and ensemble modeling. Recently, several hybrid models combining sampling methods with boosting have been proposed, such as RUSBoost, LIUBoost, and CUSBoost. In this study, a novel under-sampling-based boosting method named MPSUBoost is proposed to handle the class imbalance problem. The proposed method is an integration of modified PSU and AdaBoost. The performance benchmark testing conducted on 35 highly imbalanced datasets indicated that the proposed method provided performance improvement over three existing methods (RUSBoost, LIUBoost, and CUSBoost). Moreover, we verified that the samples obtained by MPSUBoost effectively represented the given majority data, which led to a competitive advantage in the imbalanced data, particularly when true positives are imperative.

INDEX TERMS Imbalanced data, undersampling, boosting, data mining, ensemble modeling.

I. INTRODUCTION

In data mining and machine learning, a variety of supervised algorithms have been proposed for classification problems. However, there is a major obstacle known as the data imbalance problem between classes when using such algorithms in real situations [1], [2], [3]. When the sample volume in one class overwhelms the sample volume in other classes, traditional classification models tend to maximize prediction accuracy by being biased to the majority class. In real world problems when the identification of minor classes is the main purpose of the algorithm, such as anomaly detection [4] and spam mail detection [5], the classification of minority classes is more important than the classification of the majority class. In this respect, traditional methods suffer from class imbalance problems, particularly when the misclassification of minority classes causes a high cost for the users.

Various attempts have been made to solve the data imbalance problem [6]. They can be broadly classified into three categories depending on how they handle the problem. The first approach is to artificially balance the distribution between classes in the training data used for training models. Specifically, this method selects fewer samples from the majority class (undersampling) or generates additional samples of the minority class (oversampling) to balance the ratio of the majority class to the minority class. The second approach seeks to modify existing algorithms, optimizing the performance of existing learning algorithms by modifying methods to be more appropriate for specific real-world situations. Cost-sensitive learning, which assigns a greater misclassification cost to the minority class examples than to the majority class examples, is in this group. Due to the greater weight placed on minority samples, the models can reduce the classifiers' bias towards the majority class and focus on the minority class to minimize the total misclassification cost. The third type of approach is feature selection, which aims to optimize the performance of a classifier by selecting a subset of proper features. In particular, feature selection might be a powerful approach to handle most imbalanced classification problems with high-dimensional data [7]. Feature selection is focused on handling features that are too disparate, thereby allowing a classifier to perform optimally. In addition, some attempts have been made to improve the classification performance by applying two or more of the methods mentioned above. For instance, an ensemble approach combines multiple base classifiers with sampling techniques [8] such

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

that weak classifiers learned from balanced training data can be combined. In general, this hybrid approach demonstrates powerful results in addressing imbalanced classification problems [9], [10], [11].

Reducing the impact of the majority class inevitably leads to a loss of information [12]. As a result, the recognition rate of the majority class would be decreased, even though the model has power of explanation on the minority class [13]. Underrepresenting the majority class can result in excessive false positives, which is undesirable in many real-world problems. For example, when detecting spam mail [14], false positives may mean that important emails, such as a due date notice for a loan payment, are blocked by being misclassified as loan advertisements. Another example would be a bot access prevention algorithm [15] where misclassifying users as bots and banning them could lead websites to lose loyal customers.

In this paper, we propose a novel undersampling-and-boosting-based algorithm referred to as Modified Particle Stacking Undersampling AdaBoost (MPSUBoost). The proposed model is a hybrid method based on a sampling method PSU [16] applied to AdaBoost. As elaborated in the following sections, PSU is a deterministic undersampling method; therefore, it is limited to having fixed starting points for the sampling procedure. To complement this, we modify the algorithm so that the proposed method can utilize a variety of samples to minimize the loss of information while securing the recognition rate of the majority class.

The remaining paper is structured as follows. Section 2 provides an overview of prior works that discuss how to handle the class imbalance problem. Section 3 summarizes limitations of prior works and the motivation of this study. Section 4 describes the MPSUBoost method with computational procedures. Section 5 provides the experimental results and compares the performance of MPSUBoost to other algorithms. Section 6 discusses the classification performance of MPSUBoost in connection with data distribution. Section 7 presents a summary of the results and directions for future work.

II. RELATED WORKS

A. DATA-LEVEL APPROACHES

In this group of approaches, sampling methods are classified differently according to whether samples are deleted from the majority class (undersampling) or artificially generated and duplicated from the minority class (oversampling).

1) UNDERSAMPLING

There exist a number of undersampling methods based on a variety of sampling criteria. The simplest form of undersampling is random under-sampling (RUS). RUS randomly chooses samples from the majority class to make a balanced dataset. However, RUS leads to a loss of information from the unselected majority samples due to the characteristics of randomly selected samples. To reduce information loss, several undersampling methods have been proposed, such as the Condensed Nearest Neighbor rule (CNN), Tomek Links (TL), NearMiss (NM), and Cluster Centroids (CC) [17], [18], [19], [20].

Recently, Jeon and Lim [16] proposed a novel undersampling technique referred to as PSU, which is a distance-based technique that splits majority data into multiple partitions based on the distance from the centroid. PSU selects a sample from each partition such that the sample must be the farthest from other samples that are already selected. By maximizing the distance sum between resampled samples, loss of information and data redundancy will be minimized. However, due to its deterministic nature, PSU has a limitation in reducing the loss of information. PSU selects only the same samples from the majority class no matter how many times sampling processes are performed.

The aforementioned undersampling methods have been used in various real-world applications, such as analyzing tweet sentiment data, detecting web attacks, and predicting credit card default [21], [22], [23].

2) OVERSAMPLING

Several oversampling methods have been devised to handle class imbalance problems. In [24], a synthetic minority oversampling technique (SMOTE) was proposed as a method of creating a new minority class sample by interpolating two points that are close to each other among the minority class samples. Simply replicating the same samples of the minority class multiple times can increase the probability of an over-fitting issue. SMOTE can prevent this issue by creating new samples instead of replicating existing ones. Nonetheless, SMOTE has some drawbacks. It is possible to introduce outliers by artificially synthesizing minority class samples without considering the position of the samples of the majority class. In addition, if the data is high-dimensional, the computational cost will increase dramatically.

MSMOTE [25] has been proposed to overcome the above-mentioned shortcomings. This method considers the distribution of minor classes. After dividing the minority class into three categories (border, safe, and latent noise), samples belonging to the latent noise category are not used to synthesize artificial samples.

Oversampling methods have been used in class imbalance problems, such as survival prediction of heart failure patients, emotion classification in a YouTube dataset, and intrusion detection [26], [27], [28].

B. ALGORITHM-LEVEL APPROACHES

In most class imbalance problems, information about the minority class is more important than information about the majority class. Algorithm-level approaches focus on classifying the minority class well by modifying existing learning algorithms to mitigate bias towards the majority class. Cost-sensitive learning is a representative modified method of an algorithm-level approach.

Methods of cost-sensitive learning modify weights of minority samples by assigning the importance of the minority class to be greater than the importance of the majority class.

AdaCost [29] is a cost-sensitive boosting algorithm. AdaCost adds misclassification costs to AdaBoost's weight update mechanism. The misclassification cost indicates that the costs of failing to properly classify the samples in the data are assigned differently depending on their class (i.e., minority or majority). This cost assignment leads AdaCost to put more emphasis on the correct classification of the minority class than on the majority class. Similarly, Sun et al. [30] proposed three cost-sensitive learning classifications based on boosting methods: AdaC1, AdaC2, and AdaC3. All three methods use modified weight update formulas of AdaBoost to bias their respective weighting strategies. These methods attempt to handle the imbalance problem by assigning relatively high importance to samples of the minority class. However, to obtain the individual importance of each sample, knowledge of domain experts is required beforehand. In the real world, it is often difficult to obtain the required domain expertise.

Algorithm-level approaches have been applied in several class imbalance problems, such as classifying risk types of human papillomavirus, face recognition, and predicting product failures [31], [32], [33].

C. ENSEMBLE APPROACHES

Longadge and Dongre [34] claimed that applying two or more approaches gives a better solution to the class imbalance problem. In general, methods from ensemble approaches are combinations of a data-level method with the traditional boosting or bagging method. This approach keeps the respective advantages of both methods by combining them and can function as a robust classifier. In the real world, it often shows strong performance in coping with imbalance problems, so methods of this group are often used.

SMOTEBoost, RUSBoost, and EasyEnsemble [10] are typical ensemble approaches. SMOTEBoost [9] introduced a combined sampling and boosting algorithm to deal with the imbalance problem. SMOTEBoost merges an effective oversampling technique (SMOTE) with AdaBoost [35], resulting in a highly effective ensemble approach to learn from imbalanced data. SMOTE is an oversampling method that synthesizes samples of the minority class to create balanced data. SMOTEBoost applies SMOTE to create the necessary balanced data to train weak classifiers in each iteration. However, SMOTE has higher computational cost than other undersampling methods, and SMOTEBoost has high computational complexity because it performs SMOTE in each iteration. Also, SMOTEBoost is vulnerable to noisy data because SMOTE may synthesize new noisy samples from real noisy samples.

RUSBoost [11] combines RUS with AdaBoost to deal with class imbalance problems. RUSBoost uses RUS to obtain balanced training data, which is then used for training weak classifiers in the boosting phase. Even though RUSBoost is a relatively simple method relying on randomness, Seiffert et al. [11] showed that its performance is competitive with those of other ensemble methods. It also has the advantage of lower computational cost in comparison to many oversampling-based methods. In this respect, when applying a model to big data, RUSBoost may be appropriate. However, it has the drawback of causing a loss of information due to its randomness-dependent property.

To minimize the loss of information, CUSBoost [36] was proposed as a cluster-based undersampling method integrated with a boosting method. CUSBoost uses cluster-based undersampling (CUS) in the boosting phase. CUS is a sampling method that clusters the majority samples into k clusters using a k-means clustering algorithm. It then uses random undersampling on each of the created clusters to select majority samples. Similarly, locality-informed undersampling boosting (LIUBoost) [37] attempts to minimize the loss of information by incorporating a cost term for every sample, based on hardness, into the weight update formula.

III. MOTIVATION

The existing treatments of imbalanced data each have their own drawbacks. Undersampling methods suffer from a loss of information due to discarding some of the given sample distribution. This leads to an underrepresented majority class, which results in poor predictive performance.

To supplement information loss, various hybrid methods (e.g., RUSBoost, LIUBoost, and CUSBoost) have been developed by integrating sampling methods with boosting models. These methods repeat sampling in each iteration to train multiple weak classifiers. Nevertheless, RUSBoost and LIUBoost largely rely on randomness, and it is likely that the underrepresented majority class eventually yields poor classification performance [37].

CUSBoost adopts CUS to reduce its randomness by extracting majority samples from each cluster. However, it requires a predetermined number of clusters to be identified, which is not only arbitrary but also inappropriate when the dataset is not suitable for clustering.

When integrating the sampling method with the boosting approach, the overall predictive performance is largely determined by the sampling method adopted. Among several undersampling methods, PSU has shown a relatively low level of information loss on highly imbalanced datasets [16]. In this respect, PSU-based boosting methods are expected to yield better performance than existing methods. However, PSU was originally designed as a deterministic method. This indicates that applying the original PSU algorithm to construct a boosting method will lead to the issues described below.

First, all weak classifiers would learn on the same dataset, which is against the intention of the sampling-based boosting method. Second, PSU is vulnerable to outliers, which can deteriorate the predictive performance when combined with a boosting method. PSU is designed to select the farthest majority sample from the previously chosen samples; thus, it may include outliers in the final set of samples. Additionally, informative samples might be disregarded or underrepresented by extreme points. This problem becomes significant when integrating PSU with a boosting method; the outliers would be selected in every iteration, and they will end

up preventing weak classifiers from learning an appropriate data distribution.

This study customizes the PSU algorithm while maintaining the original property of minimizing information loss. The proposed method, referred to as MPSUBoost, is based on a modified PSU algorithm integrated with AdaBoost. We particularly focus on maximizing the data representability of the minority class.

IV. METHODOLOGY

A. RETHINKING EXISTING PERSPECTIVES

To extend the conventional PSU algorithm so that it can be integrated with an ensemble, we propose two modification ideas: multiple starting points and median distance measure.

1) STARTING POINTS

The PSU algorithm begins with the centroid of the majority class to split the majority data class into partitions. Due to its deterministic nature, PSU results in a final sample set that is unique to the starting point, namely the centroid. This indicates that if one wants to obtain diverse sample sets, multiple starting points have to be determined. Figure 1 illustrates different sample sets obtained from (a) the centroid, (b) some points close to each other, and (c) PSU samples. Note that PSU was performed five times on the starting points (black points) to obtain the aggregated final sample sets (blue points). Unsurprisingly, the conventional PSU algorithm (a) yielded a unique set of samples; therefore, the same samples were simply selected five times. In (b) and (c), in contrast, different starting points resulted in correspondingly distinct sample sets, and a pool of diverse samples was obtained in aggregate.

The key question is then how to choose ideal starting points to construct an ensemble model. When the starting points are close to each other, PSU would end up with samples extracted from a similar majority region, as seen in (b). This may not be appropriate for ensemble modeling as a diverse set of samples are generally sought to take advantage of divergent decision boundaries. The excluded region of the majority class can cause significant information loss, which would lead to poor performance of a strong classifier.

To utilize PSU's strength of data representability, we propose using PSU samples as individual starting points on which PSU is repeated to extract multiple sample sets. As shown in (c), samples from the original PSU, namely the blue points in (a), were then used as starting points for different iterations of PSU to yield the final sample sets. In this way, a boosting ensemble can be constructed with a diverse set of samples that better represent the original data distribution of the majority class.

2) DISTANCE MEASURE

Another unique characteristic of PSU is that it uses the farthest distance from the previously chosen samples to select subsequent samples. Although this approach is intended to reduce data redundancy, it is possible for outlying data points to be overrepresented because they tend to be located far from the overall data distribution. This may ultimately deteriorate

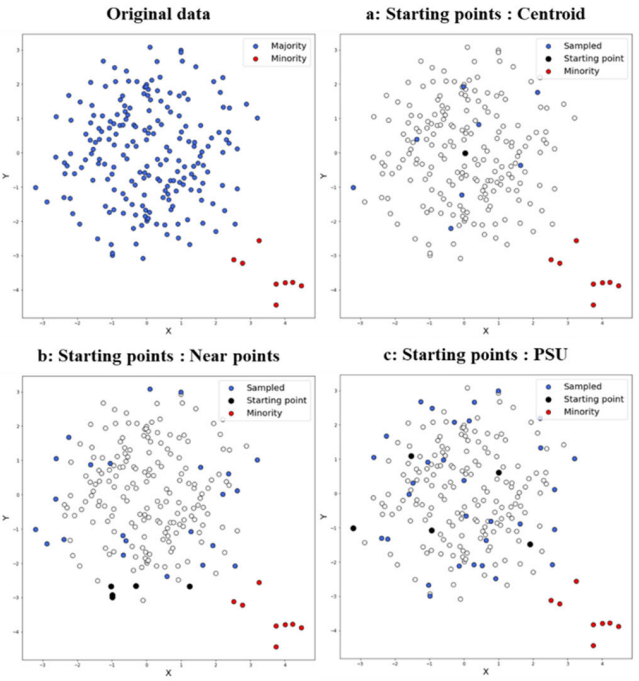

FIGURE 1. Sampling results from different starting points.

the overall predictive performance. Specifically, the same outliers will be included in the final sample set, regardless of different starting points, as they will be most likely recognized as 'representative points.' This inadvertently reduces the resulting ensemble into a strong classifier consisting of a few similar weak classifiers.

To alleviate the impact of outlying data points in PSU, we suggest using a median distance to select samples. If median samples in each partition were selected instead of the farthest samples, PSU would be less likely to choose outlying data points because those points are unlikely to be median points in each partition. Consequently, PSU would proceed conservatively and become robust to outliers. Figure 2 compares four different undersampling methods applied to the same original data. The blue points are the sampled majority data points from the corresponding undersampling methods repeated five times. It is seen in RUS, CUS, and PSU that data points in the isolated cluster (bottom right corner) are not represented, while extreme data points are included in the sample set. In contrast, the proposed method based on starting points from PSU and the median distance could sample some points in the separate cluster as well as a point that seems less extreme, while also avoiding points extremely far from the overall data distribution. We expect that the proposed method would be able to identify distinct data segments from representative starting points, while the tendency of selecting extreme points would be lessened by using the median distance.

B. MPSUBoost

Concerning the two methodological modifications (multiple starting points and median distance measure), we propose a new undersampling and boosting-based algorithm,

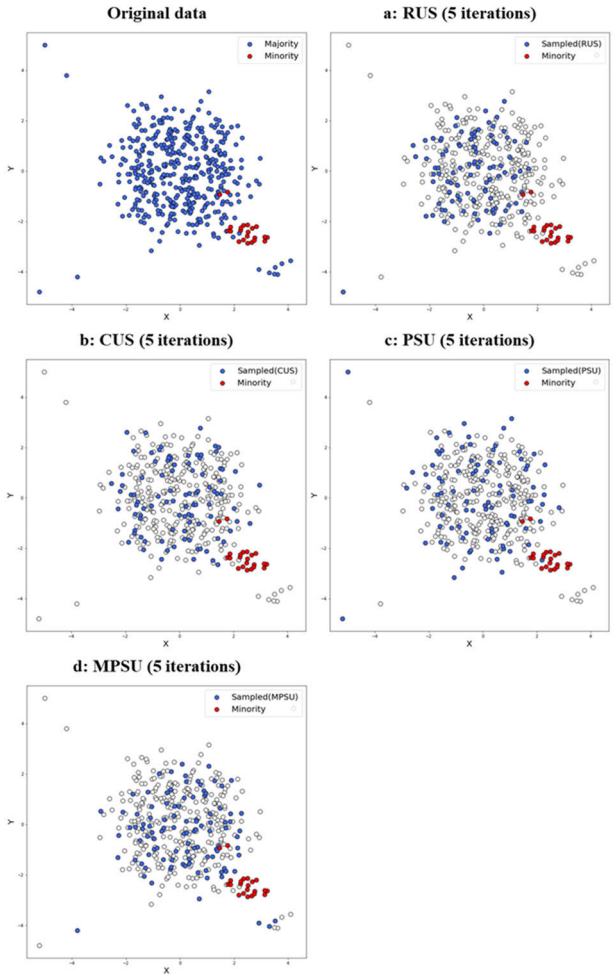

FIGURE 2. Comparison of various undersampling methods.

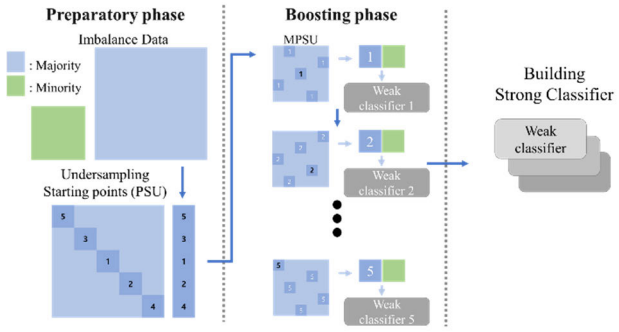

FIGURE 3. MPSUBoost procedures.

i.e., MPSUBoost. The algorithmic procedure of the proposed method is illustrated in Figure 3 and specified in Algorithm 1. In the preparatory phase, the original PSU method is first implemented to obtain representative samples of the majority class (D^R). Each data point (D_t^R) in D^R is then used as a starting point in the following model construction phase. Specifically, when a sample is selected from D^R, the distances from the sample to majority data points are calculated as (1). $Dist_t$ is sorted in ascending order and then grouped into partitions (S_i) from which one of each representative sample (X_i^{re}) is selected. Note here that when a sample is selected from a partition, the sample must be a median point in the partition. As discussed already, the aim of this modification from the original PSU method is to prevent extreme data points from being included in the final sample set. Combined with using multiple starting points, it is expected that a diverse set of samples that better represent the original data distribution can be obtained; this sample set should be less affected by sparsely distributed outlying points.

$$Dist_t = d_2\left(D_t^R, X_1^{Maj}\right), \ldots, d_2(D_t^R, X_M^{Maj}) \tag{1}$$

With the sample set obtained for each iteration, MPSUBoost builds the weak classifier (h_t). The weak classifier learns on the resampled majority data (D_t^{re}) and minority data (D^{Minor}) with their corresponding weights, i.e., w_t^{re} and w_t^{Minor}, respectively. The weak classifier (h_t) is built in each iteration and the weight of each data point ($w_{t+1}(i)$) is updated by calculating the error (ϵ_t) and the weight update parameter (α_t).

$$\epsilon_t = \sum_{i=1}^{N} w_t(i) \times I(h_t(X_i) \neq y_i) \tag{2}$$

$$\alpha_t = \frac{\epsilon_t}{1-\epsilon_t} \tag{3}$$

$$w_{t+1}(i) = w_t(i) \times exp(-y_i \alpha_t h_t(X_i)) \tag{4}$$

Within this process, the weights of the misclassified data points increase, while the weights of the correctly classified data points decrease. After performing all iterations, the weak classifiers ($h_1, \ldots, h_t$) are combined to create a strong classifier.

It is worth noting that MPSUBoost inherits the deterministic nature of PSU, that is, the methodological extensions applied to MPSUBoost guarantee the same unique strong classifier will be obtained regardless of the experimental setting. This property can be an advantage, particularly when reproducible results are imperative and/or the sensitivity of the model with additional data points has to be verified.

V. EXPERIMENTAL EVALUATION

A. DATASETS

Three well-known methodologies (RUSBoost, CUSBoost, LIUBoost) are compared with MPSUBoost by applying them to the classification problem on highly imbalanced datasets. A total of 35 datasets were used for performance evaluation; all were obtained from the KEEL repository [38]. Note that nine multi-class datasets were divided into 35 binary-class problems. Table 1 summarizes the description of the datasets.

B. EXPERIMENTAL SETUP

The proposed model is an ensemble approach that addresses the class imbalance problem. To see the relative performance of the proposed method compared with other well-known ensemble methods, we designed an experiment as follows.

- A stratified random split was conducted to perform hold-out validation. Each dataset was randomly partitioned into two sets: a training set (70%) and a test set (30%). The ratio of each class was maintained by using a stratified random split. Hold-out validation is repeated

Algorithm 1 MPSUBoost

Input

a) Majority class data $D^{Major} = \{(X_1^{Maj}, y_1^{Maj}), \ldots, (X_M^{Maj}, y_M^{Maj})\}$

b) Minority class data $D^{Minor} = \{(X_1^{Min}, y_1^{Min}), \ldots, (X_m^{Min}, y_m^{Min})\}$

c) Iteration count T

d) Number of data $N = M + m$

Initialization: $w_t(i) = \frac{1}{N}$, *where* $i = 1, \ldots, N$;

1. Calculate the centroid of majority class data:
   $\mathrm{C} = (X_1^{Maj} + X_2^{Maj} + \ldots + X_M^{Maj})/\mathrm{M}$
2. Calculate L2- norm between C and majority class data:
   $D_2 = d_2\left(C, X_1^{Maj}\right), \ldots, d_2\left(C, X_M^{Maj}\right)$
3. Sort D_2 and group them into T partitions:
   $S = [s_1, \ldots, s_T]$
4. Set X_1' to be the last data point in s_1
   for $l = 2$ **to** T **do**
5. Set X_l' to be the farthest data point in s_l in the resampled dataset: $\{X_2', \ldots, X_{\mathrm{T}}'\}$
   end for
6. Construct the resampled dataset: $D^R = \{X_1', \ldots, X_T'\}$
   for $t = 1$ **to** T **do**
7. Orderly select majority data from D^R: D_t^R
8. Calculate L2-norm between D_t^R and majority class data: $Dist_t = d_2\left(D_t^R, X_1^{Maj}\right), \ldots, d_2(D_t^R, X_M^{Maj})$
9. Sort $Dist_t$ and group them into m partitions: $S = [S_1, \ldots, S_m]$
   for $l = 1$ **to** m **do**
10. Set X_l^{re} to be the median data point in S_i
    end for
11. Construct a resampled dataset: $D_t^{re} = \{X_1^{re}, \ldots, X_m^{re}\}$
12. Fit the classifier h_t based on the resampled dataset D_t^{re} *and* D^{Minor} with their weights w_t^{re} *and* w_t^{Minor}
13. Calculate the error $\epsilon_t = \sum_{i=1}^{N} w_t(i) \times I(h_t(X_i) \neq y_i)$ and weight update parameter $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$
14. Update data weight $w_{t+1}(i) = w_t(i) \times \exp(-y_i \alpha_t h_t(X_i))$

   end for

Output

The strong classifier: $sign(\sum_{t=1}^{T} \alpha_t h_t(X_i))$

100 times to obtain the average performance to further reduce variations in the random splits.

- For a fair comparison, the base classifier of the comparison methods was controlled to be classification and regression trees (CART), which were generated using the Scikit-learn package [39]. Also, the maximum number of weak classifiers for each model is set to 100. Each model was a combination of a sampling method and AdaBoost, and the ratio of the majority class to the minority class sampled from each iteration was set to 1:1. In addition, common hyperparameter settings were used: the number of neighbors was set to 5 for LIUBoost and the number of clusters was set to 10 for CUSBoost.
- Since we focus the classification performance on the minority class, balanced metrics (F1-score, AUC, MCC, G-mean, Precision, and Recall) were used as performance measures.

TABLE 1. Data description.

Dataset	Number of instances	Number of attributes	Minor ratio
ecoli-0-3-4_vs_5	200	8	9.0
ecoli-0-6-7_vs_3-5	222	8	9.1
ecoli-0-2-3-4_vs_5	202	8	9.1
glass-0-4_vs_5	92	10	9.2
ecoli-0-4-6_vs_5	203	7	9.2
ecoli-0-3-4-6_vs_5	205	8	9.2
ecoli-0-2-6-7_vs_3-5	224	8	9.2
ecoli-0-3-4-7_vs_5-6	257	8	9.3
ecoli-0-6-7_vs_5	220	7	10.0
ecoli-0-1-4-7_vs_2-3-5-6	336	8	10.6
ecoli-0-1_vs_5	240	7	11.0
glass-0-6_vs_5	108	10	11.0
ecoli-0-1-4-7_vs_5-6	332	7	12.3
shuttle-c0-vs-c4	1829	10	13.9
glass4	214	10	15.5
page-blocks-1-3_vs_4	472	11	15.9
zoo-3	101	17	19.2
glass-0-1-6_vs_5	184	10	19.4
shuttle-c2-vs-c4	129	10	20.5
shuttle-6_vs_2-3	230	10	22.0
glass5	214	10	22.8
yeast-2_vs_8	482	9	23.1
kr-vs-k-zero-one_vs_draw	2901	7	26.6
kr-vs-k-one_vs_fifteen	2244	7	27.8
yeast4	1484	9	28.1
winequality-red-4	1599	12	29.2
poker-9_vs_7	244	11	29.5
yeast-1-2-8-9_vs_7	947	9	30.6
yeast5	1484	9	32.7
winequality-red-8_vs_6	656	12	35.4
winequality-red-8_vs_6-7	855	12	46.5
poker-8-9_vs_6	1485	11	58.4
shuttle-2_vs_5	3316	10	66.7
kr-vs-k-zero_vs_fifteen	2193	7	80.2
poker-8-9_vs_5	2075	11	82.0

C. RESULTS

Table 2 summarizes the results of the experiment. The average performances and comparative mean ranks of the four methods are presented. Note that mean ranks were obtained by averaging performance ranks of the 35 datasets. The best average performance and mean rank corresponding to each measure are highlighted in bold.

Overall, MPSUBoost was found to have competitive performance with the other methods; it was only surpassed by RUSBoost in terms of average recall but outperformed in all other measures. It is noteworthy that MPSUBoost performed better than the other methods, particularly in terms of the

TABLE 2. Performance benchmark.

Dataset	MPSUBoost						RUSBoost						LIUBoost						CUSBoost					
	F1-score	AUC	MCC	G-mean	Precision	Recall	F1-score	AUC	MCC	G-mean	Precision	Recall	F1-score	AUC	MCC	G-mean	Precision	Recall	F1-score	AUC	MCC	G-mean	Precision	Recall
ecoli-0-3-4_vs_5	0.8082	0.8955	0.7966	0.8858	0.8349	0.8117	0.7594	0.8957	0.7457	0.8871	0.7368	0.8333	0.7097	0.8564	0.6864	0.8451	0.7008	0.7533	0.7889	0.8934	0.7755	0.8846	0.7977	0.8150
ecoli-0-6-7_vs_3-5	0.7029	0.8349	0.6776	0.8192	0.7310	0.7043	0.6901	0.8303	0.6619	0.8145	0.7083	0.7000	0.7155	0.8465	0.6901	0.8347	0.7293	0.7314	0.6827	0.8282	0.6551	0.8115	0.7011	0.6957
ecoli-0-2-3-4_vs_5	0.8215	0.9043	0.8112	0.8962	0.8439	0.8283	0.7548	0.8870	0.7400	0.8777	0.7437	0.8133	0.7157	0.8600	0.6938	0.8492	0.7072	0.7600	0.7958	0.8976	0.7818	0.8899	0.7986	0.8217
glass-0-4_vs_5	0.9279	0.9894	0.9242	0.9892	0.8770	1	0.9232	0.9882	0.9198	0.988	0.8718	1	0.8846	0.9830	0.8789	0.9827	0.8040	1	0.9210	0.9869	0.9168	0.9866	0.8682	0.9967
ecoli-0-4-6_vs_5	0.7841	0.9011	0.7707	0.8930	0.7712	0.8333	0.7396	0.8992	0.7216	0.8934	0.6784	0.8483	0.7097	0.8673	0.6873	0.8577	0.6790	0.7800	0.7475	0.8846	0.7312	0.8754	0.7297	0.8083
ecoli-0-3-4-6_vs_5	0.7744	0.9063	0.7583	0.9005	0.7327	0.8500	0.7585	0.9038	0.7424	0.8983	0.7093	0.8517	0.7306	0.8775	0.7099	0.8694	0.6999	0.7967	0.7737	0.9007	0.7579	0.8939	0.7446	0.8367
ecoli-0-2-6-7_vs_3-5	0.7056	0.8372	0.6798	0.8173	0.7273	0.7071	0.6773	0.8263	0.6510	0.8069	0.6997	0.6929	0.7151	0.8455	0.6916	0.8330	0.7369	0.7286	0.6826	0.8257	0.6567	0.8072	0.7119	0.6900
ecoli-0-3-4-7_vs_5-6	0.7453	0.8716	0.7228	0.8622	0.7379	0.7800	0.6832	0.8393	0.6548	0.8267	0.6760	0.7300	0.6507	0.8257	0.6176	0.8119	0.6307	0.7062	0.7363	0.8580	0.7133	0.8467	0.7499	0.7500
ecoli-0-6-7_vs_5	0.7538	0.8929	0.7385	0.8864	0.7240	0.8233	0.7347	0.8884	0.7163	0.8817	0.6879	0.8200	0.7508	0.8978	0.7347	0.8921	0.7050	0.8367	0.7603	0.8939	0.7446	0.8876	0.7325	0.8233
ecoli-0-1-4-7_vs_2-3-5-6	0.6431	0.8066	0.6224	0.7821	0.6869	0.6489	0.6218	0.8035	0.5955	0.7832	0.6353	0.6544	0.5107	0.7549	0.4668	0.7295	0.4789	0.5833	0.6359	0.8013	0.6121	0.7771	0.6730	0.6389
ecoli-0-1_vs_5	0.7472	0.8764	0.7371	0.8636	0.7601	0.7800	0.7232	0.8903	0.7087	0.8825	0.6714	0.8233	0.6012	0.8336	0.5775	0.8191	0.5370	0.7317	0.7486	0.8748	0.7362	0.8634	0.7582	0.7767
glass-0-6_vs_5	0.7801	0.9042	0.7771	0.8778	0.7619	0.8433	0.6840	0.8737	0.6739	0.8323	0.6337	0.8067	0.4288	0.7475	0.4091	0.6329	0.2995	0.6567	0.7821	0.8972	0.7796	0.8559	0.7715	0.8233
ecoli-0-1-4-7_vs_5-6	0.6983	0.8483	0.6803	0.8333	0.7005	0.7288	0.6463	0.8312	0.6273	0.8158	0.6388	0.7113	0.5388	0.8039	0.5088	0.7884	0.4622	0.6825	0.6711	0.8417	0.6505	0.8259	0.6554	0.7225
shuttle-c0-vs-c4	0.9992	0.9999	0.9992	0.9999	0.9984	1	0.9996	1	0.9996	1	0.9992	1	0.9997	1	0.9997	1	0.9995	1	0.9996	1	0.9996	1	0.9992	1
glass4	0.5677	0.8452	0.5610	0.8267	0.4909	0.7525	0.5151	0.8242	0.5059	0.8064	0.4286	0.7225	0.2668	0.6364	0.2305	0.4583	0.2052	0.3825	0.5210	0.8354	0.5111	0.8142	0.4144	0.7475
page-blocks-1-3_vs_4	0.9129	0.9899	0.9124	0.9895	0.8538	0.9912	0.7125	0.9571	0.7277	0.9554	0.6014	0.9712	0.4049	0.6751	0.4229	0.4239	0.2771	0.4175	0.9120	0.9839	0.9113	0.9830	0.8641	0.9788
zoo-3	0.1849	0.6576	0.1691	0.4792	0.1183	0.4850	0.1556	0.6138	0.1254	0.4059	0.1014	0.4100	0.1081	0.5245	0.0545	0.1380	0.0715	0.1250	0.0981	0.5228	0.0331	0.2403	0.0631	0.2650
glass-0-1-6_vs_5	0.5695	0.8466	0.5709	0.8200	0.5128	0.7467	0.5324	0.8711	0.5470	0.8415	0.4176	0.8300	0.4769	0.8846	0.5187	0.8284	0.3180	0.8867	0.5614	0.8390	0.5669	0.8091	0.5154	0.7467
shuttle-c2-vs-c4	0.9240	0.9468	0.9293	0.9380	0.9858	0.8950	0.8967	0.9249	0.9050	0.9080	0.9900	0.8500	0.8967	0.9249	0.9050	0.9080	0.9900	0.8500	0.9259	0.9447	0.9302	0.9231	0.9798	0.8900
shuttle-6_vs_2-3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
glass5	0.5318	0.8196	0.5334	0.7592	0.4775	0.6867	0.4975	0.8634	0.5199	0.8399	0.4093	0.8133	0.4702	0.9264	0.5210	0.9095	0.3118	0.9667	0.4975	0.8262	0.5047	0.7753	0.4343	0.7167
yeast-2_vs_8	0.6651	0.7592	0.7020	0.7088	1	0.5183	0.6604	0.7589	0.6962	0.7085	0.9874	0.5183	0.6651	0.7592	0.7020	0.7088	1	0.5183	0.6651	0.7592	0.7020	0.7088	1	0.5183
kr-vs-k-zero-one_vs_draw	0.6723	0.8957	0.6787	0.8882	0.6087	0.8155	0.4862	0.9123	0.5231	0.9103	0.3405	0.8929	0.3232	0.5113	0.3371	0.0308	0.2150	0.0261	0.6756	0.8926	0.6806	0.8849	0.6160	0.8084
kr-vs-k-one_vs_fifteen	1	1	1	1	1	1	0.9978	0.9999	0.9977	0.9999	0.9958	1	0.7127	0.5232	0.7058	0.0571	0.7645	0.0470	1	1	1	1	1	1
yeast4	0.3233	0.7584	0.3310	0.7264	0.2299	0.5893	0.3027	0.7876	0.3245	0.7760	0.1965	0.6733	0.3130	0.8042	0.3399	0.7959	0.2017	0.7067	0.3228	0.7775	0.3361	0.7603	0.2191	0.6367
winequality-red-4	0.1452	0.5943	0.1171	0.4940	0.1023	0.2875	0.1439	0.5988	0.1178	0.5034	0.0987	0.2981	0.1915	0.6210	0.1667	0.5316	0.1418	0.3125	0.1356	0.5909	0.1075	0.4903	0.0924	0.2875
poker-9_vs_7	0.2116	0.6529	0.2060	0.4708	0.1695	0.3750	0.1723	0.6188	0.1579	0.4073	0.1359	0.3250	0.0166	0.4919	-0.0335	0.0194	0.0099	0.0150	0.1953	0.6213	0.1809	0.3896	0.1611	0.2950
yeast-1-2-8-9_vs_7	0.1566	0.5850	0.1351	0.4490	0.1561	0.2433	0.1581	0.5904	0.1353	0.4542	0.1411	0.2544	0.1792	0.5779	0.1566	0.4061	0.1933	0.1878	0.1507	0.5896	0.1288	0.4602	0.1344	0.2589
yeast5	0.5764	0.8759	0.5937	0.8629	0.5028	0.7815	0.5041	0.9228	0.5501	0.9162	0.3860	0.9023	0.4228	0.9193	0.4881	0.8933	0.2718	0.9131	0.5763	0.8862	0.5944	0.8758	0.4847	0.8038
winequality-red-8_vs_6	0.2034	0.6854	0.2083	0.6127	0.1346	0.4500	0.1829	0.6976	0.1973	0.6375	0.1156	0.4960	0.1570	0.6293	0.1486	0.5260	0.1044	0.3500	0.1996	0.6880	0.2064	0.6169	0.1308	0.4580
winequality-red-8_vs_6-7	0.1153	0.6267	0.1186	0.5131	0.0697	0.3480	0.1131	0.6341	0.1198	0.5402	0.0679	0.3740	0.1162	0.6185	0.1157	0.4933	0.0721	0.3260	0.1119	0.6280	0.1160	0.5223	0.0673	0.3540
poker-8-9_vs_6	0.0370	0.5121	0.0091	0.3310	0.0210	0.1800	0.0337	0.5058	0.0040	0.3128	0.0191	0.1600	0.0276	0.4998	-0.0012	0.0348	0.0172	0.0150	0.0349	0.5072	0.0055	0.3183	0.0198	0.1662
shuttle-2_vs_5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
kr-vs-k-zero_vs_fifteen	1	1	1	1	1	1	0.9860	0.9995	0.9876	0.9995	0.9808	1	0.6270	0.6818	0.6311	0.4262	0.5562	0.3675	0.9984	1	0.9985	1	0.9973	1
poker-8-9_vs_5	0.0314	0.5285	0.0172	0.3444	0.0172	0.2013	0.0283	0.5208	0.0118	0.3269	0.0155	0.1825	0.0178	0.4970	-0.0039	0.1614	0.0103	0.0662	0.0299	0.5234	0.0142	0.3255	0.0165	0.1850
Mean performance	**0.6206**	**0.8300**	**0.6140**	**0.7863**	**0.6097**	0.7053	0.5850	0.8274	0.5804	0.7839	0.5577	**0.7131**	0.5159	0.7630	0.5074	0.6428	0.4829	0.5779	0.6097	0.8229	0.6011	0.7744	0.5972	0.6947
Mean rank	**1.4000**	**1.8286**	**1.4857**	**1.8857**	**1.4571**	**1.8286**	2.8571	2.2286	2.8000	2.1429	2.8571	1.9429	3.1143	2.9429	3.0857	3.0000	3.0571	2.8571	2.0857	2.4000	2.1143	2.3714	2.1143	2.3143

TABLE 3. ***Post-hoc*** **test (Wilcoxon) results (*p*-value) compared with MPSUBoost.**

Measure	Benchmark Methods		
	RUSBoost	LIUBoost	CUSBoost
F1-score	0.0000 *	0.0000 *	0.0002 *
AUC	0.2418	0.0023 *	0.0045 *
MCC	0.0000 *	0.0000 *	0.0002 *
G-mean	0.5317	0.0006 *	0.0190
Precision	0.0000 *	0.0000 *	0.0031 *
Recall	0.6735	0.0034 *	0.0551

F1-score and MCC, which are sensitive to distinguishing the minority class.

To show whether the differences in the performance measures were statistically significant, the Friedman chi-square test was first conducted on the rank values. As a result, it was confirmed that the methods performed unequally (p-value < 0.001) for all the measures. A *post-hoc* Wilcoxon rank test was then performed as a means of pairwise comparison of the methods. Note that the significance level of each pairwise comparison was corrected using the Bonferroni correction [40], [41] with the adjusted alpha risk of 0.0083 ($\approx$0.05/6). Table 3 summarizes the *post-hoc* test results (p-value) of benchmark methods compared with MPSUBoost. p-values smaller than the adjusted alpha risk are marked with an asterisk. Overall, MPSUBoost showed distinctly superior classification performance on the highly imbalanced datasets.

In a relative sense, smaller improvements have been observed in AUC, G-mean, and Recall compared to the other measures. It is interesting to note that those measures commonly put less emphasis on the true positive rate, which is critical in a highly imbalanced dataset. On the contrary, MPSUBoost clearly surpassed other methods in terms of the F1-score, MCC, and Precision. It is therefore recommended to use MPSUBoost, particularly when false positives cause a significant cost with a highly imbalanced dataset.

VI. DISCUSSION

It can be seen from the experimental results that MPSUBoost performs well in terms of the F1-score, namely the harmonic mean of Precision and Recall, compared to the other methods. Specifically, MPSUBoost performed better relative to the other methods in terms of Precision than in terms of Recall. This implies that the decision boundary constructed by MPSUBoost may be tightly enveloping the minority class in a way that true positives could be maximized. On the other hand, the majority samples are fully represented by the sampled data points so that weak classifiers could learn on an unbiased training space consistently in all iterations. This property allows the weak classifiers to attain comprehensive insights for classification, whereas those from other methods are likely to be biased toward the given representation of the majority region. As a result, MPSUBoost can perform

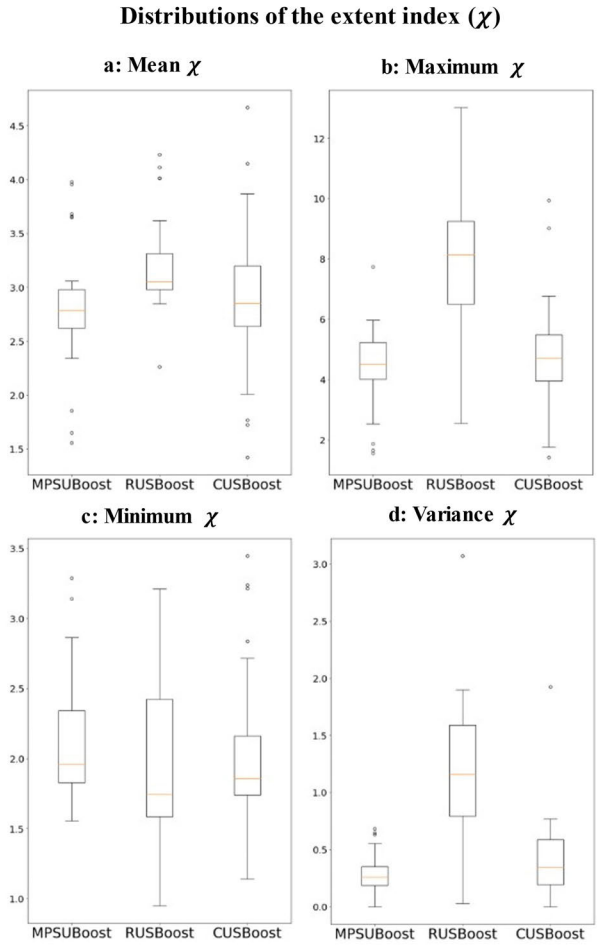

FIGURE 4. **Distributions of the extent index (χ) in Euclidian distance.**

a generalizable classification for majority points, while a reliable decision can be made on minority points. However, this methodological characteristic might leave some relevant points unrecognized; nonetheless, it may still have a competitive advantage in scenarios where false negatives are less of a concern but accurately identifying positives is imperative.

The wide range of the decision region secured for the majority class possibly comes from the inherent properties of MPSUBoost. In particular, it is intended to select a sample for each partition; hence, MPSUBoost can always secure a certain radial distance from a starting point to a sample selected in the final partition. Although there is a chance to pick a sample that is distant from the minority region, even if that is done, it is likely that a sample selected in the previous partition would have been close to the minority region. One could argue that the conventional PSU algorithm where the farthest distance is adopted for sample selection further enlarges the majority region; however, it should be recognized that this extreme approach may also increase the likelihood of choosing outlying majority points. Unlike MPSUBoost, RUSBoost and LIUBoost randomly miss the chance of selecting samples close to the minority region. Similarly, in the case of CUSBoost, a certain distance from the minority region to samples, namely cluster centroids, has

to be reserved. Therefore, obtaining some false positives is unavoidable.

To examine the size of majority regions established by MPSUBoost compared with other methods, we introduce the extent index (χ), which is an average distance between vertices of the convex hull [42] of samples. As the index measures perimeters as well as diagonals, the wider the area of the sample space, the higher the index.

Figure 4 contains distributions of the extent indices obtained throughout the experiment (in all iterations) using MPSUBoost, RUSBoost, and CUSBoost. Note that the Euclidean distance[1] is used in the figure and the sample space obtained from LIUBoost is the same as RUSBoost; therefore this is omitted. It is verified that MPSUBoost retained a certain sample space size that is larger than the other two methods (see its relatively large minimum size in c), and such spaces are sustained over multiple iterations (see its relatively small variance in d). This serves as the basis for the argument that MPSUBoost tends to yield constantly wide majority regions (represented by the samples), allowing weak classifiers to learn on representative under-sampled majority points. In contrast, RUSBoost shows highly fluctuating sizes of its sample space (see its relatively large maximum size in b and small minimum size in c resulting from the high level of variance in d). Unsurprisingly, this is attributed to the randomness adopted by the method, where majority points are underrepresented at some times and overrepresented at other times. This may have led the feature space to include biased and outlying distributions of majority points in some boosting iterations.

VII. CONCLUSION

In this paper, a new under-sampling-based boosting algorithm, named MPSUBoost, was proposed to address the data imbalance problem. To overcome the deterministic nature of the conventional PSU algorithm, we have applied two modification ideas: multiple starting points and median distance measure. The performance benchmark conducted on 35 highly imbalanced datasets demonstrated that the proposed method provided statistically significant improvement over competing methods by effectively selecting majority samples. In addition, we verified that the samples obtained by MPSUBoost are sufficiently and consistently representative of the given majority data.

The unique characteristics of the proposed method can become weaknesses, depending on the modeling situation. For example, when the cost of false negatives is greater than the cost of false positives, MPSUBoost would not be an ideal option. In addition, care must be taken when the degree of complexity is high, such as situations with a wide indecision region in the feature space where majority and minority points are randomly distributed. In such a case, the tendency of securing an extensive majority region would increase the likelihood of misclassifying minority points.

[1]We find consistent patterns using different distance measures; see Figure 5 and Figure 6 in the Appendix for the cases where the Minkowski and Manhattan distances are used, respectively.

In the current setting, sampling median points in each partition is expected to yield high Precision. However, aggressive sampling might be more appropriate, especially if majority points are widely distributed as multiple clusters. The existing distance metric may then be too conservative to properly capture majority points located in the outer feature space.

As a direction for future work, methodological extensions could be considered to advance the proposed method. Above all, false negatives could be reduced by considering both majority and minority regions in such a way that the representation of majority points is less affected by indecision regions (if any). One can also consider an integration of the proposed method with dynamic distance measures such that an optimal choice of samples suitable for each partition could be flexibly utilized to avoid the underrepresentation of majority points. Furthermore, MPSUBoost should be compared with over-sampling-based boosting methods and/or other ensemble approaches. One could also explore optimal parameters to build base classifiers along with the sampling ratio. Finally, case studies are needed to reflect real-world data imbalance issues to validate the proposed method.

APPENDIX

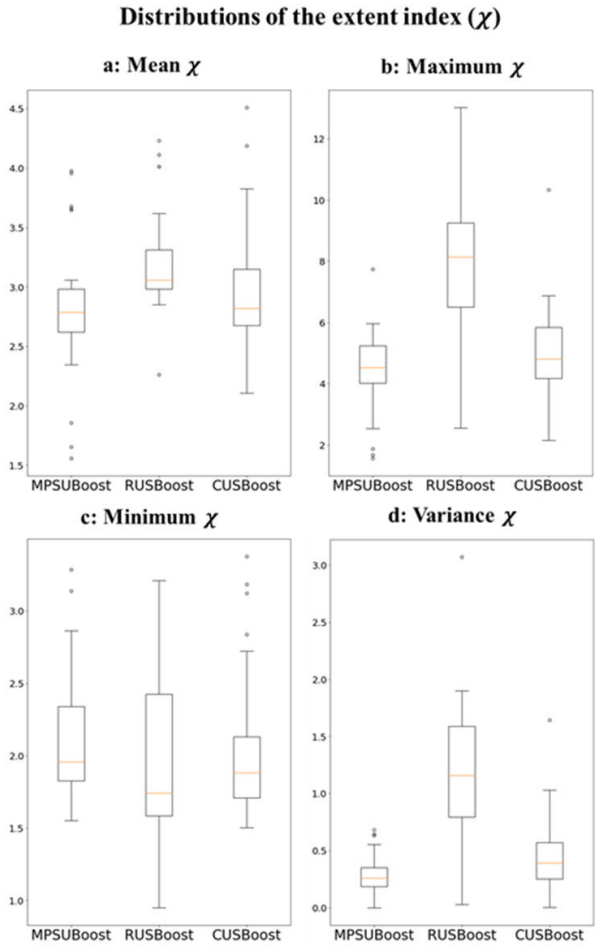

FIGURE 5. Distributions of the extent index (χ) in Minkowski distance.

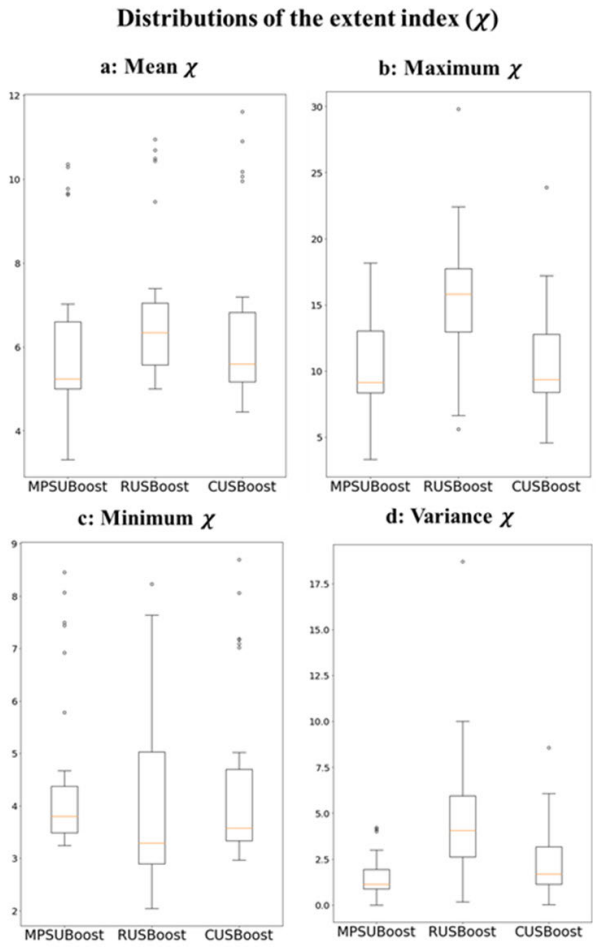

FIGURE 6. Distributions of the extent index (χ) in Manhattan distance.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009.

[3] Q. Yang and X. Wu, "10 challenging problems in data mining research," *Int. J. Inf. Technol. Decision Making*, vol. 5, no. 4, pp. 597–604, 2006.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[5] N. Jindal and B. Liu, "Review spam detection," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 1189–1190.

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

[7] M. Wasikowski and X. Chen, "Combating the small sample class imbalance problem using feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1388–1400, Oct. 2010.

[8] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *J.-Jpn. Soc. Artif. Intell.*, vol. 14, nos. 771–780, p. 1612, 1999.

[9] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Princ. data Mining Knowl. Discovery*, 2003, pp. 107–119.

[10] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 539–550, Apr. 2008.

[11] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[12] G. E. Batista, R. C. Prati, and M. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, 2004.

[13] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Mar. 2009, pp. 324–331.

[14] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," 2001, *arXiv:cs/0109015*.

[15] F. Morstatter, L. Wu, T. H. Nazer, K. M. Carley, and H. Liu, "A new approach to bot detection: Striking the balance between precision and recall," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2016, pp. 533–540.

[16] Y.-S. Jeon and D.-J. Lim, "PSU: Particle stacking undersampling method for highly imbalanced big data," *IEEE Access*, vol. 8, pp. 131920–131927, 2020.

[17] H. Altınçay and C. Ergün, "Clustering based under-sampling for improving speaker verification decisions using AdaBoost," in *Joint IAPR Int. Workshops Stat. Techn. Pattern Recognit. (SPR) Struct. Syntactic Pattern Recognit. (SSPR)*, 2004, pp. 698–706.

[18] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.

[19] I. Tomek, "Two Modifications of CNN," *IEEE Trans. Syst., Man, Cybern.*, vol. 6, no. 11, pp. 769–772, 1976.

[20] I. Mani and I. Zhang, "kNN approach to unbalanced data distributions: A case study involving information extraction," in *Proc. Workshop Learn. From Imbalanced Datasets*, vol. 126, 2003, pp. 1–7.

[21] J. Prusa, T. M. Khoshgoftaar, D. J. Dittman, and A. Napolitano, "Using random undersampling to alleviate class imbalance on tweet sentiment data," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2015, pp. 197–202.

[22] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Detecting web attacks using random undersampling and ensemble learners," *J. Big Data*, vol. 8, no. 1, pp. 1–20, Dec. 2021.

[23] M. Alam, "An investigation of credit card default prediction in the imbalanced datasets," *IEEE Access*, vol. 8, pp. 201173–201198, 2020.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[25] S. Hu, Y. Liang, L. Ma, and Y. He, "MSMOTE: Improving classification performance when training data is imbalanced," in *Proc. 2nd Int. Workshop Comput. Sci. Eng.*, 2009, pp. 13–17.

[26] A. Ishaq, S. Sadiq, M. Umer, S. Ullah, S. Mirjalili, V. Rupapara, and M. Nappi, "Improving the prediction of heart failure Patients' survival using SMOTE and effective data mining techniques," *IEEE Access*, vol. 9, pp. 39707–39716, 2021.

[27] P. Sarakit, T. Theeramunkong, and C. Haruechaiyasak, "Improving emotion classification in imbalanced Youtube dataset using SMOTE algorithm," in *Proc. 2nd Int. Conf. Adv. Inform., Concepts, Theory Appl. (ICAICTA)*, Aug. 2015, pp. 1–5.

[28] Z. Li and G. Yan, "A spark platform-based intrusion detection system by combining MSMOTE and improved AdaBoost algorithms," in *Proc. IEEE 9th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2018, pp. 1046–1049.

[29] W. Fan, S. J. Stolfo, and J. Zhang, "AdaCost: Misclassification cost-sensitive boosting," in *Proc. 16th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, vol. 1999, pp. 97–105.

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

[31] S.-B. Park, S. Hwang, and B.-T. Zhang, "Mining the risk types of human papillomavirus (HPV) by AdaCost," in *Proc. Int. Conf. Database Expert Syst. Appl.*, 2003, pp. 403–412.

[32] F. D. Frumosu, A. R. Khan, H. Schiøler, M. Kulahci, M. Zaki, and P. Westermann-Rasmussen, "Cost-sensitive learning classification strategy for predicting product failures," *Expert Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113653.

[33] Y. Zhang and Z. H. Zhou, "Cost-sensitive face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1758–1769, Oct. 2010.

[34] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," 2013, *arXiv:1305.1707*.

[35] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, Aug. 1995.
[36] F. Rayhan, S. Ahmed, A. Mahbub, R. Jani, S. Shatabda, and D. M. Farid, "CUSBoost: Cluster-based under-sampling with boosting for imbalanced classification," in *Proc. 2nd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solution (CSITSS)*, Dec. 2017, pp. 1–5.
[37] S. Ahmed, "LIUBoost: Locality informed under-boosting for imbalanced data classification," in *Emerging Technologies in Data Mining and Information Security*. Cham, Switzerland: Springer, 2019, pp. 133–144.
[38] J. Alcalc-Fdez, S. Garcia, L. Sanchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Log. Soft Comput.*, vol. 17, pp. 1–36, Jun. 2011.
[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2012.
[40] R. A. Armstrong, "When to use the Bonferroni correction," *Ophthalmic Physiol. Opt.*, vol. 34, no. 5, pp. 502–508, 2014.
[41] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 152–161, 2016.
[42] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

SANG-JIN KIM received the bachelor's degree in systems management engineering from Sungkyunkwan University, South Korea. He is currently a Junior Researcher with the Department of Industrial Engineering, Sungkyunkwan University. His current research interests include data mining, machine learning, boosting, and imbalanced data.

DONG-JOON LIM received the B.S. and M.S. degrees in industrial engineering from Sungkyunkwan University, South Korea, and the Ph.D. degree in engineering and technology management from Portland State University, USA. He is an Assistant Professor with the Systems Management Engineering Department, Sungkyunkwan University, South Korea. His current research interests include technological forecasting, optimization modeling, productivity analysis, and data mining. He is also a Developer of an open-source R package "DJL" which implements various decision support tools related to econometrics and technometrics. His academic honors include the Emerald Literati Network Award (outstanding author), the ENI Award (finalist for renewable and non-conventional energy), the Marie Brown Award, and various fellowships from PSU, SKKU, A&P, etc.