IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 11, 2021, accepted November 21, 2021, date of publication November 23, 2021,
date of current version December 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130272

A Learning Objective Controllable Sphere-Based Method for Balanced and Imbalanced Data Classification

YEONTARK PARK AND JONG-SEOK LEE
Department of Industrial Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea

Corresponding author: Jong-Seok Lee (jongseok@skku.edu)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government Ministry of Science and ICT (MSIT) under Grant 2019R1A4A1024732 and Grant 2020R1A5A1019649; and in part by the Institute for Information and Communications Technology Planning & Evaluation (IITP) funded by the Korean Government (MSIT) under Grant 20210002920012002.

ABSTRACT Imbalanced data classification is one of the most important tasks in the field of machine learning because abnormality, which is usually of our interest, appears less frequently than normality in real-world systems. Learning classifiers from imbalanced data can be troublesome due to no absolute standard as to how much imbalance can be said to be imbalanced or balanced. To address this issue, this research proposes a new sphere-based classification method named LOCS (learning objective controllable sphere-based classifier), which is designed to maximize AUC (area under ROC curve). The AUC learning objective was adopted from the fact that it approximates the accuracy as class distribution becomes balanced. Therefore, the proposed method properly performs a classification task for both imbalanced and balanced data. It constructs a classification model by a single training, whereas existing cost-sensitive learning and resampling methods usually attempt different parameter settings. In addition, the learning objective can be easily modified within LOCS for each of application domains by setting different importance levels for positive and negative classes, respectively. Numerical experiments based on 25 real datasets with several investigational settings showed the effectiveness and the intended strengths of the proposed method.

INDEX TERMS Classification, class imbalance, sphere covering, learning objective, area under ROC curve.

I. INTRODUCTION

One of the most important challenges in the research field of machine learning and pattern recognition is the imbalanced data learning problem, which is an issue in various practical fields, such as software defect prediction [1], medical diagnosis [2], disaster information [3], industrial maintenance monitoring [4], financial trading order [5], and customer churn prevention [6]. The problem of learning from imbalanced data is attributable to the skewed distribution of the class. Rare instances represented by minority classes are relatively difficult to detect owing to infrequency and casualness [7]. However, these minority classes — such as cancer, fraud, and faults — are considered more important than the majority class in the various practical problems listed above, and the risk of misclassification of minority class instances is higher than the risk of misclassification of majority class instances correspondingly. Traditional classification techniques such as decision trees, support vector machines, and neural networks have been developed under the assumption that the class distribution of data is balanced [8]. As the majority class in imbalanced data is composed of a relatively larger number of instances than the minority class, it has an overwhelming influence on the minority class. Therefore, a relatively small number of minority classes are underestimated during the training process, which may lead to the failure of classifiers to accurately learn the pattern or distribution of minority classes [9]. This problem becomes more severe as the degree of imbalance increases, resulting in the failure to detect a minority class instance [10]. These traditional classification methods, trained with learning objectives that maximize overall accuracy [11], can become useless if they cannot be used for classification even though high accuracy is achieved. For example, a model that classifies all instances into majority classes with an accuracy of 99% cannot be used in the case of imbalanced data with ten minority classes and 1000 majority

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

classes. Therefore, a classification method that maximizes accuracy or minimizes error rate can be irrelevant as it results in low classification performance in imbalanced data classification [12]. A number of studies have been conducted to address this imbalanced data classification problem. These studies can be divided mainly into two approaches: data level approaches and algorithm level approaches [13]. The data level approach is a method of resizing data through resampling [14]. By doing so, the performance of the classifier can be improved by training after solving the imbalanced distribution of data. It can be divided mainly into over-sampling minority class instances and under-sampling majority class instances. Cost-sensitive learning (CSL), which sets misclassification costs differently for the majority and minority classes, is a representative algorithm level approach method.

Over-sampling is a method of reinforcing the influence of minority class instances, which consists of a process of randomly extracting and replicating instances from a set of minority instances in the data and adding them to the existing set of minority instances. In this method, the size of the minority instances set increases as much as the number of duplicated minority numbers shows a balanced distribution with the set of majority instances. A synthetic minority oversampling technique (SMOTE) generates synthetic data using more advanced over-sampling method [15]. This method does not replicate minority instances but creates new synthetic minority instances by interpolating between minority instances and their k-nearest neighbors and combining them with original minority instances to form a set of minority instances. Variants of SMOTE such as border-line SMOTE [16], adaptive synthetic sampling (ADASYN) [17], safe-level SMOTE [18], and MWMOTE [19] have been proposed since this study was published.

Under-sampling method is a method of randomly extracting instances from the majority class and removing them from the original set of majority instances. As a result, the total data size is reduced as much as the number of removed majority instances. Several strategies have been introduced for the rebalancing of minority and majority distribution by reducing majority instances more effectively. One of them is cluster-based under-sampling for the majority class that forms a cluster [20]. Furthermore, a study was conducted to substitute representatives of majority class instances with centers of clusters [21]. However, in general, resampling-based imbalance problem-solving techniques have a weakness in that original data cannot be used. Although resampling techniques create balanced distribution data for classifier training, it is difficult to be free from information loss caused by changes in the original distribution owing to synthetic data or by excluding informative majority instances [22].

Algorithm level approaches have been attempted to solve the imbalance problem without changing the data distribution. While the resampling approach focuses on a balanced class ratio of the original data, the algorithm level approach is designed to prevent damage to the original data and to use the original distribution, without changes, for training to solve the imbalance problem in the learning process rather than preprocessing. CSL is one of the most widely used algorithm level approaches to solving imbalance problems [10], [23], [24]. CSL is a method of minimizing the overall expected cost by allocating different misclassification costs to each class [10], [25], and the misclassification cost are usually determined by domain experts. In the past decades, CSL has received great attention as a problem-solving method for skewed class distribution [26], and many studies have proved that CSL is effective in addressing class imbalance problems [9], [13], [27], [28]. Representative studies that applied CSL to existing classification algorithms include cost-sensitive kNN [29]–[31], cost-sensitive SVM [32], [33], and cost-sensitive ANN [34], [35]. Although CSL has attempted to solve the imbalance problem by imposing different misclassification costs for each class using the domain knowledge of experts, it is usually difficult to determine the optimal cost for both the majority and minority classes [12], [14]. In addition, in the case of highly imbalanced data, the CSL may be biased towards a minority class given a high cost, and conversely, neglect majority classes, resulting in poor classification performance [36].

On the other hand, the sphere covering method for classifying instances using spheres has been developed as a method of finding prototypes that can represent instances of each class and determine a radius that evaluates the area that can be covered by the prototypes. The class cover problem was introduced in [37]. The class cover problem is to find a small number of sets covering, i.e. containing, points from one class without covering any points from the other class. Greedy sphere covering [38] used the class cover catch digraph to solve this problem and applied it in classification. In other words, the spatial area that can be covered by the prototype was regarded to be the distance to other class data located at the shortest distance using the nearest neighbor rule, which was set as the radius of the sphere. In addition, an instance containing as much data as possible was selected as a prototype. Interpretable prototype selection [39] has translated the optimization problem to select a minority of prototypes, including all possible training data, to a set cover optimization problem. To this end, a greedy algorithm was introduced that independently selects a prototype for each class. Randomized sphere cover (RSC) [40], [41] randomly selects the center (prototype) of the sphere from the training data and constructs a sphere with the radius as the shortest distance between the selected center and another class instance. A classification model consisting of a set of spheres constructed by repeating this process was introduced. This method constructs many spheres by repeatedly selecting centers randomly and allows α instances of the same class in each sphere. It is intended to improve classification accuracy by classifying test instances using spheres that only cover instances of the same class. However, in the case of binary classification, although the aforementioned spherical classification methods have been developed under the premise that

the class distribution is balanced and is suitable for increasing the accuracy, minority class instances cannot be accurately classified in imbalanced distribution since the number of minority instances is small, thereby decreasing the number of minority spheres. In other words, they are not suitable for the classification of imbalanced data as in the existing traditional classification methods.

The aforementioned methods have two drawbacks: first, additional parameters need to be set in advance, and second, the learning objective and the evaluation measure are inconsistent. In particular, the second drawback is a serious problem in machine learning. Sampling rates need to be determined in advance for resampling methods, and misclassification costs need to be determined in advance for CSL methods. In general, since the exact values of these parameters are unknown, a number of previous studies have tried different values, and the classifier with the best performance among the classifiers thus trained was selected based on G-mean, F1-score, or area under ROC curve (AUC), which is a measure used to evaluate a classifier for imbalanced data. At this moment, there is a discrepancy between the learning objective and the evaluation measure. This study proposes a new method in which the evaluation measure for classification performance evaluation of imbalanced data and the learning objective of the classifier are consistent in solving these problems. The proposed method is similar to the conventional RSC where the classification model is expressed as a sphere. However, there is a significant difference in the fact that the learning objective is set as AUC ROC and is designed to maximize its value during the learning process of the sphere, which is the main idea of this study. The contributions of this study in the research field of imbalanced data classification are as follows. First, a novel sphere-based classifier for the classification of imbalanced data is proposed. Second, the learning objective and evaluation measure for the classification of imbalanced data are matched as AUC. Third, the proposed algorithm has the advantage that the user can train the classifier by controlling the value of the true positive rate (TPR) and the false positive rate (FPR) according to the application problem.

The remainder of this paper is organized as follows. The RSC, which is the basis of the proposed classifier, and the classification performance evaluation indicator of imbalanced data will be reviewed in Section II. The algorithm of a learning objective controllable sphere-based classifier (LOCS) proposed in this study will be described with an illustrative example for ease of understanding in Section III. The proposed method will be tested with 25 real data sets, and its performance will be compared with the conventional methods in Section IV. Finally, this study will be concluded in Section V.

II. BACKGROUND

This section briefly describes the RSC classifier and the basic performance evaluation measures used in the imbalanced data classification as background information to explain the proposed method. As in other literature, "majority" and "negative", and "minority" and "positive" have been used interchangeably hereafter.

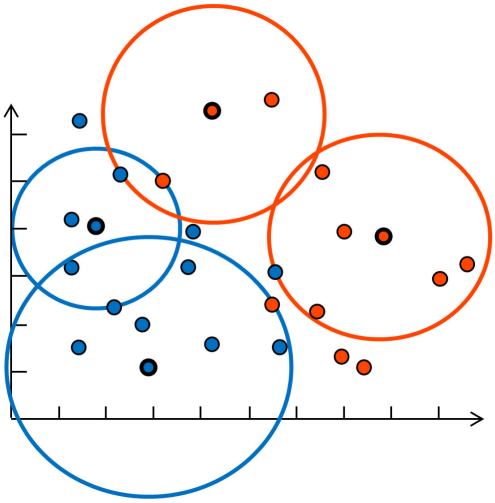


FIGURE 1. A set of spheres constructed by RSC from a binary class data. In this example, α was set to three. A constructed sphere thus contains at least three instances.

A. RANDOMIZED SPHERE COVER

The RSC is one of the sphere covering methods introduced in [40], [41]. RSC constructs sphere B_i from training data $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x_i}$ represents a vector of observation i, and y_i indicates the class of the ith instance. The sphere B_i has a specific class C_{B_i} and consists of a center $\boldsymbol{c_i}$ and a radius r_i. Therefore, the sphere is defined by the following 4-tuple, $B_i = \langle C_{B_i}, \boldsymbol{c_i}, r_i, X_{B_i} \rangle$, where $X_{B_i} = \{\boldsymbol{x} \in D \mid d(\boldsymbol{x}, \boldsymbol{c_i}) < r_i\}$ [37]. The radius r_i of the sphere B_i is defined as the distance between the center $\boldsymbol{c_i}$ and the closest instance which class is different from the class of $\boldsymbol{c_i}$, and it is given below.

$$r_i = \min_{\boldsymbol{x_j} \in \{X \setminus X_{B_i}\} \wedge y_j \neq C_{B_i}} d(\boldsymbol{x_j}, c_j), X = \{\boldsymbol{x} \in D\} \tag{1}$$

The αRSC algorithm uses an input parameter α, which represents the number of minimum instances within a sphere, which means that the sphere is not constructed if the number of instances included in the sphere is less than α when constructing a sphere. Informally, the training process of αRSC is as follows:

Repeat the process below until all training data is covered or discarded.

1) Randomly select an instance regarded as a center and add it to the set of covered instances.
2) Find the closest instance that has a different class with the instance selected as the center.
3) Set the distance between the closest instance and the center as the radius of the sphere.
4) Construct a sphere with the center and the radius.
5) Find all instances within the sphere in the training data.
6) If the number of instances inside the sphere is greater than α, add all of them to the set of covered instances

and the store sphere details (center, class, and radius). Otherwise, discard the instances.

The detailed pseudo code is described in [41]. Through the above training process, αRSC constructs the set of spheres, and for example, spheres are constructed as shown below in Fig. 1.

Through the set of spheres constructed during the training process, new instances are classified in the prediction stage according to the following rules.

- Classification rule 1: The test instance, which is covered by a sphere, takes the target class of the sphere. If there is more than one sphere of different target class covering the instance, the instance will take the target class of the sphere with the closest center.
- Classification rule 2: In the case where an instance is not covered by a sphere, the classifier selects the closest spherical edge.

Classification rule 2 is reasonable as it can classify test instances – mainly outliers – in areas not covered by spheres [40]. It is better for the spheres constructed in RSC to contain as many instances as possible, as this can lead to an increase in accuracy during the prediction stage. However, in the imbalanced data in which the number of minority instances is less than that of the majority instances, the number of spheres of the minority class constructed in the training stage will be less than that of the majority class. In addition, the fact that the number of minority class spheres is small implies that the probability that new minority class instances will be covered in the sphere in the prediction stage is reduced compared to the majority class instances, thereby degrading the classification performance. Therefore, to mitigate the minority class from being overwhelmed by the majority class, this research proposes to extend the radii of the constructed spheres to ensure that the AUC, an evaluation measure suitable for the classification of imbalanced data, is maximized. In other words, the evaluation measure and the learning objective will be matched by setting the evaluation measure, AUC, as a learning objective for the training of the classifier. This will induce the influence of the minority class spheres to increase. After completion of the training, the two classification rules of reasonable RSC will be applied as they are in the classification step.

B. EVALUATION MEASURES FOR BINARY CLASSIFICATION

In binary classification, the measure for evaluating the predictive performance of the classifier is generally computed based on the confusion matrix in Table 1. In this matrix, true positives (TP) is the number of positive instances classified correctly, false negatives (FN) is the number of positive instances classified incorrectly, false positives (FP) is the number of negative instances classified incorrectly, and true negatives (TN) is the number of negative instances classified correctly.

The accuracy defined in (2), below, using a confusion matrix, refers to the proportion of the total data that is correctly classified and used as a general performance measure of a classifier.

TABLE 1. Confusion matrix.

		Predicted	
		Positive	Negative
True	Positive	*TP*	*FN*
	Negative	*FP*	*TN*

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2}$$

However, in the case of imbalanced data, the accuracy cannot properly express the performance of the classifier as the positive class is overwhelmed by the negative class. For example, the accuracy of the classifier is 95% even if all instances are predicted as negative class for data composed of 5% positive class and 95% negative class. However, it can be said that if a classifier cannot detect positive class instances at all, then it is completely ineffective. Therefore, the classification performance needs be evaluated by two measures rather than one measure in the imbalanced data classification problem, and the two measures defined in (3) and (4) below are one of the most used pairs of measures.

$$TPR = \frac{TP}{TP + FN} \tag{3}$$

$$FPR = \frac{FP}{FP + TN} \tag{4}$$

TPR refers to the rate at which the classifier was correctly classified as positive class instances, also referred to as sensitivity. FPR refers to the rate at which the classifier was incorrectly classified as negative class instances. When evaluating the classification performance of imbalanced data, it is generally necessary to consider both TPR and FPR at the same time so that the receiver operating characteristics (ROC) graph makes it possible to organize and visualize the performance of the classifier [42]. The ROC graph can visualize the performance of the classifier based on the two indicators by plotting TPR and FPR on the vertical and horizontal axes, respectively. The area under this ROC curve is the AUC. Although the AUC calculation is complicated in the case of a soft classifier, the AUC value in the case of a hard classifier is defined below by (5). In the end, in the imbalanced data classification, the AUC makes it possible to evaluate the performance of the classifier as a single indicator instead of considering both TPR and FPR indicators at the same time.

$$AUC = \frac{1 + TPR - FPR}{2} \tag{5}$$

The AUC has been widely used as an indicator for evaluating the performance of classifiers in the classification of imbalanced data [12]. However, few studies have used this indicator as a learning objective for the training of a classifier [14]. As mentioned above, this study proposes a classifier training method that AUC itself as a learning objective.

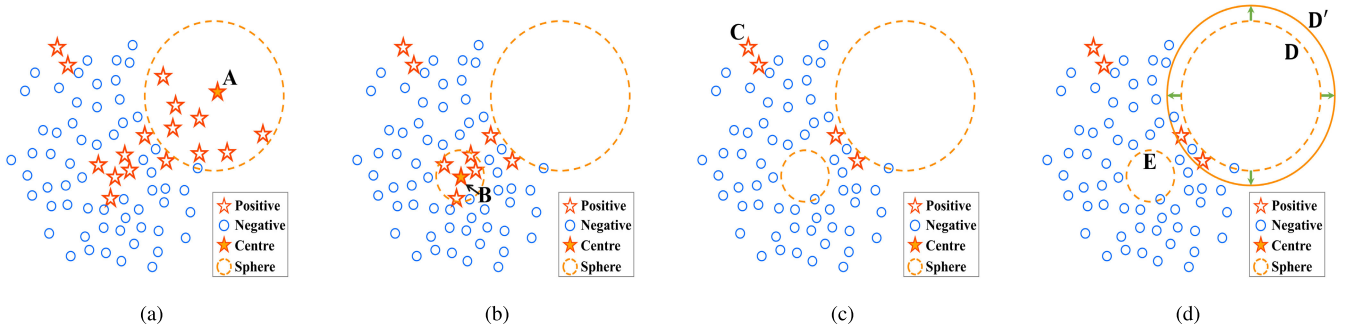


FIGURE 2. A binary class example to show constructing and expanding positive spheres. (a) The first sphere with A as the center, which has the largest number of covered instances among positive instances, is constructed. (b) The second sphere with B as the center is constructed. (c) After constructing two positive spheres, no more positive spheres are constructed because the input parameter α is three. (d) The existing positive sphere D is expanded to the sphere D′ where the AUC value is maximum. The other positive sphere E is not expanded.

III. PROPOSED METHOD

In this section, LOCS proposed in this study is described. As conventional classifiers were designed to increase the classification accuracy, it was recognized that they were weak in the classification of imbalanced data. Therefore, the main idea of the proposed algorithm is to directly set the AUC, which is a classification performance evaluation indicator in the imbalanced data classification, as an objective function to induce the classifier to maximize this and to design a sphere-based classifier that allows users to control the objective function according to their intentions by assigning different weights to TPR and FPR at the same time.

Algorithm 1 MakeSphere

Input: an instance $(\boldsymbol{x}, y)$, a set of instances $D = \{(\boldsymbol{x_i}, y_i)\}$

Output: $(r, \tilde{D})$ denotes the radius of a sphere and a set of instances in the sphere

1: Find $\min_{(\boldsymbol{x_i}, y_i) \in D} d(\boldsymbol{x}, \boldsymbol{x_i})$ such that $y \neq y_i$
2: Let $r = d(\boldsymbol{x}, \boldsymbol{x_i})$
3: Let $\tilde{D} = \{(\boldsymbol{x_j}, y_j) \mid d(\boldsymbol{x}, \boldsymbol{x_j}) < r,\ (\boldsymbol{x_j}, y_j) \in D\}$
4: **return** $(r, \tilde{D})$

A. LOCS

To construct a sphere in the training process of the classifier, a center and a radius are required, and a class representing the constructed sphere needs to be specified. Sphere class evidently follows the center class. New data can be classified according to classification rules through this set of spheres. In the same context as the previous studies, the radius of the sphere is defined as the distance from the center to the closest instance that has a different class from the center. In addition, the instances within the radius of the sphere are defined as covered instances. More formally, the radius $r = \min_{(\boldsymbol{x_i}, y_i) \in D} distance(\boldsymbol{c}, \boldsymbol{x_i})$ such that $y_c \neq y_i$, the set of covered instances $\tilde{D} = \{(\boldsymbol{x_j}, y_j) \mid distance(\boldsymbol{c}, \boldsymbol{x_j}) < r, (\boldsymbol{x_j}, y_j) \in D\}$, where $\boldsymbol{c}$ denotes the center, y_c denotes the class of the $\boldsymbol{c}$, and D denotes the dataset. Therefore, all covered instances within the radius have the same class as the center. The process of constructing a sphere when an instance $\boldsymbol{x}$ is selected as the center is as shown in Algorithm 1.

Spheres are constructed to cover as many instances as possible, ultimately to construct as few spheres as possible, and to reduce the computational time of the subsequent Algorithm 4. When an instance is selected as the center, the radius of the sphere to be constructed and the number of instances to be covered are reviewed to determine the center of the sphere that can cover most of the instances in the current situation, as described in lines 5-11 of Algorithm 2. Note that Algorithm 1 is invoked in Algorithm 2. If there are many covered instances inside the constructed sphere, this means that the class has a high density. The aim of constructing a dense sphere is to improve the classification performance of the final-trained classifier. As shown in lines 12-16 of Algorithm 2, even though a sphere containing as many instances as possible is constructed, the sphere is no longer constructed if the number of instances in it is smaller than the predetermined minimum sphere size α. Otherwise, it stores the data of the constructed sphere, center, and radius, and the instances covered by the constructed sphere are excluded from the training set T. This process is repeated until the algorithm termination condition is satisfied.

The process of constructing a set of spheres in which target class cl is positive is described through an illustrative example. The binary and imbalanced data containing 17 positive instances and 60 negative instances in which the two classes are divided into overlap states are shown in Fig. 2. All positive instances become candidates for the center to construct the first sphere. Instance A is the one with the largest number of instances covered by the radius of each central candidate. Therefore, the first positive sphere with instance A as the center can be constructed in a dotted line, as shown in Fig. 2 (a). As these instances covered by the constructed sphere are excluded from consideration, the center of the sphere containing the next largest number of instances becomes instance B, as shown in Fig. 2 (b). In this example, it was assumed that the minimum size of sphere α is set to three. As spheres with

Algorithm 2 PreCreateSpheres

Input: training set $T = \{(\boldsymbol{x_i}, y_i)\}$, minimum sphere size α, target clss $cl \in \{+, -\}$

Output: a set of spheres $S = \{(\boldsymbol{c_j}, r_j)\}$ where the elements denote center and radius of jth sphere

1: $S \leftarrow \emptyset$
2: **repeat**
3: Let $T_{cl} = \{(\boldsymbol{x_i}, y_i) \mid y_i = cl, (\boldsymbol{x_i}, y_i) \in T\}$
4: $r^* \leftarrow 0$, $\boldsymbol{c}^* \leftarrow \emptyset$, $maxCardinality \leftarrow 0$, $\tilde{D}^* \leftarrow \emptyset$
5: **for** each instance $(\boldsymbol{x_i}, y_i) \in T_{cl}$ **do**
6: $(r_i, \tilde{D}_i) \leftarrow$ **MakeSphere**$((\boldsymbol{x_i}, y_i), T)$
7: **if** $|\tilde{D}_i| > maxCardinality$ **then**
8: $r^* \leftarrow r_i$, $\boldsymbol{c}^* \leftarrow \boldsymbol{x_i}$,
9: $maxCardinality \leftarrow |\tilde{D}_i|$, $\tilde{D}^* \leftarrow \tilde{D}_i$
10: **end if**
11: **end for**
12: **if** $maxCardinality < \alpha$ **then**
13: **break**
14: **end if**
15: $S \leftarrow S \cup \{s = (\boldsymbol{c^*}, r^*)\}$
16: $T \leftarrow T \setminus \tilde{D}^*$
17: **end repeat**
18: **return** S

Algorithm 3 LOCS

Input: training set $T = \{(\boldsymbol{x_i}, y_i)\}$, minimum sphere size α, TPR importance w_1, FPR importance w_2

Output: a set of positive spheres S^+, a set of negative spheres S^-

1: $S^+ \leftarrow$ **PreCreateSpheres**$(T, \alpha, +)$
2: Compute the imbalance ratio IR of T
3: Let $\beta = IR \times \alpha$
4: $S^- \leftarrow$ **PreCreateSpheres**$(T, \beta, -)$
5: $(S^+, S^-) \leftarrow$ **PostExpandSphere**$(S^+, S^-, T, w_1, w_2,$ $GAparameters)$
6: **return** (S^+, S^-)

the cardinality of three or above can no longer be constructed, no more spheres are constructed after the construction of two spheres. Fig. 2 (c) shows that the center of the third sphere is instance **C**, and as two instances are covered, the sphere is not constructed.

The LOCS proposed in this study is described in Algorithm 3. As the spheres for $cl = +$ are constructed with the **PreCreateSphere()** function in Algorithm 2, the same process is repeated for $cl = -$ as shown in line 4 of Algorithm 3. In other words, spheres for negative class are constructed. However, for the purpose of preventing too many negative class spheres from being constructed, the minimum number of instances that the sphere needs to cover is computed as $\beta = \alpha \times IR$, which is the value obtained by multiplying α by imbalance ratio (IR) (line 3 of Algorithm 3). In the example of Fig. 2, since $\alpha = 3$, $IR = 3.5$, the minimum number of instances that will be covered by the sphere that will be used when constructing negative class spheres is $\beta = 11$.

It should be noted in advance that the **LOCS()** function calls the **PostExpandSpheres()** function described in Algorithm 4, in which the **EvaluateFitness()** function described in Algorithm 5 is invoked. Likewise, the **PredictClass()** function described in Algorithm 6 is invoked in the **EvaluateFitness()** function. After constructing the initial sphere for both positive and negative classes, the sphere will be expanded to maximize the AUC of the sphere classifier when the training is completed. This corresponds to line 5 of Algorithm 3. After line 4 of the **LOCS()** function has been completed, instances of the same class are covered in each of all spheres. Consider the sphere **D** in Fig. 2 (d). When there are instances of the same class as the sphere outside the sphere, it has been observed that there is an opportunity to increase the TPR by expanding the sphere and incorporating the instances into the sphere. This also means that there is an opportunity to increase AUC. In other words, in the case of imbalanced data with severe overlap, if there are class instances such as spheres outside the sphere but around the sphere boundary, expanding the sphere and incorporating them into the sphere can be better for increasing AUC. Fig. 2 (d) shows two positive spheres of class **D** and **E**. In the case of a positive sphere **D**, five new instances around the sphere can be covered if the radius is extended to **D**′. In this case, although FPR is increased by misclassifying three negative instances, an increase in TPR by covering two more positive instances will result in an increase in AUC. On the other hand, in the case of positive sphere **E**, there is no reason to expand the radius because there are only negative instances around it even if the radius is expanded. Consider the case where Sphere **D** is not extended to **D**′, but sphere **E** is extended to further cover the closest positive instances. In this case, the loss to FPR is greater than the gain from TPR due to the negative instances covered by the addition of sphere **E**. In other words, whether to expand the sphere, if so, how far to expand it will be determined using the AUC for the training set, and as AUC is a suitable measure for evaluating performance in imbalanced data classification as mentioned above, this is a reasonable approach to setting the learning objective.

The abovementioned is implemented in Algorithm 4. The genetic algorithm (GA), which is one of the most widely used evolutionary algorithms, is employed to find the new radius of all spheres that can maximize AUC. In the process of fining the optimally expanded radii of spheres in terms of AUC, lines 1-3 are intended to secure feasible regions. The lower bound of a solution is set to the radii of the given spheres before expansion $\boldsymbol{r} = (r_1^+, \ldots, r_a^+, r_1^-, \ldots, r_b^-)$, and the upper bound set to the distance from the farthest instance among instances with the same class as the center of each sphere to the distance to the center $(m_1^+, \ldots, m_a^+, m_1^-, \ldots, m_b^-)$. As shown in lines 4-12, Algorithm 4 precisely follows the general processes of GA, such as fitness evaluation, selection, crossover,

Algorithm 4 PostExpandSpheres

Input: a set of positive spheres S^+, a set of negative spheres S^-, training set $T = \{(\boldsymbol{x_i}, y_i)\}$, GA parameters(crossover probability, mutation probability, maximum number of generations, and population size), TPR importance w_1, FPR importance w_2

Output: updated sets of positive and negative spheres (S^{+*}, S^{-*})

Let $\boldsymbol{r} = (r_1^+, \ldots, r_a^+, r_1^-, \ldots, r_b^-)$, where $r_i^+ \in s_i^+$ and $r_i^- \in s_i^-$
lower $\leftarrow \boldsymbol{r}$
upper $\leftarrow (m_1^+, \ldots, m_a^+, m_1^-, \ldots, m_b^-)$, where $m_i^+ = \max d(\boldsymbol{c}_i^+ \in s_i^+, \boldsymbol{x}_j|_{y_j=+} \in T)$ and $m_i^- = \max d(\boldsymbol{c}_i^- \in s_i^-, \boldsymbol{x}_j|_{y_j=-} \in T)$
Generate an initial population ranging from **lower** to **upper**
while number of generations is less than maximum number of generations **do**
    Compute the fitness of each individual as radii of S^+ and S^- using **EvaluateFitness** with w_1, w_2
    Order the population and perform *selection* process
    Perform *crossover* with crossover probability
    Perform *mutation* with mutation probability and (**lower**, **upper**)
    Update the population for next generation
end while
Select the best individual corresponding to the best fitness
Update S^+ and S^- by replacing their radii with the best individual
Let S^{+*} and S^{-*} denote the updated sets of positive and negative spheres
return (S^{+*}, S^{-*})

Algorithm 5 EvaluateFitness

Input: a set of positive spheres S^+, a set of negative spheres S^-, training set $T = \{(\boldsymbol{x_i}, y_i)\}$, TPR importance w_1, FPR importance w_2

Output: evaluated fitness value($mAUC$)

Let $\boldsymbol{y} = (y_i)$
Let $\hat{\boldsymbol{y}} = (\hat{y}_i \leftarrow$ **PredictClass**$(\boldsymbol{x_i}, S^+, S^-))$
Compute $mAUC$ in Equation (7) using $\boldsymbol{y}, \hat{\boldsymbol{y}}, w_1, w_2$
return $mAUC$

and mutation. The updated set of spheres (S^{+*}, S^{-*}) returned by this algorithm has the same centers as the set of spheres (S^+, S^-) used as input but has different radii.

The process of evaluating the fitness of a solution in GA is described in Algorithm 5. The modified AUC (mAUC) of Equation (7) is computed using the sets S^+ and S^-, TPR importance w_1, FPR importance w_2, and the training set T, which are the inputs of the **EvaluateFitness()** function. In the case of $w_1 = 1, w_2 = 1$, it is the same as the

Algorithm 6 PredictClass

Input: test instance $\boldsymbol{x}$, a set of positive spheres S^+, a set of negative spheres S^-

Output: predicted class label $\hat{cl}$

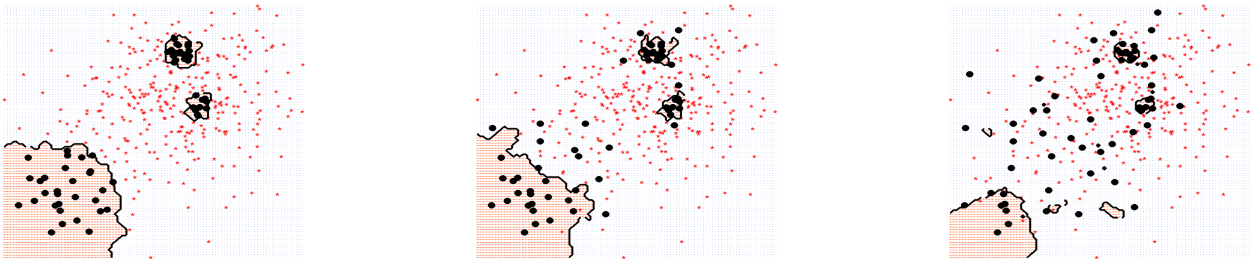
if $\boldsymbol{x}$ is covered by a sphere in S^+ **then**
    $\hat{cl} \leftarrow +$
else if $\boldsymbol{x}$ is covered by a sphere in S^- **then**
    $\hat{cl} \leftarrow -$
else if $\boldsymbol{x}$ is covered by spheres in both $S^+$and S^- **then**
    Find the closest center $\boldsymbol{c_i}$ and let s_i denote the sphere
    if $s_i \in S^+$ **then**
        $\hat{cl} \leftarrow +$
    else if $s_i \in S^-$ **then**
        $\hat{cl} \leftarrow -$
    end if
else if $\boldsymbol{x}$ is not covered by any sphere **then**
    Find the closest sphere face and let s_i denote the sphere
    if $s_i \in S^+$ **then**
        $\hat{cl} \leftarrow +$
    else if $s_i \in S^-$ **then**
        $\hat{cl} \leftarrow -$
    end if
end if
return $\hat{cl}$

AUC of Equation (5). w_1 and w_2 will be described in detail in Section III-C.

To compute AUC, the prediction class $\hat{\boldsymbol{y}}$ needs to be computed based on the given sphere sets. This is performed by the **PredictClass()** function described in Algorithm (7), which is the Classification rule 1 and Classification rule 2 mentioned in Section II-A. In brief, the class of the test instance follows the class of the spheres if the instance is only covered by the spheres of the same class; it follows the closest center class if covered by different classes of spheres, and it follows the class of the sphere with the closest distance to the sphere face if it is not covered by any sphere.

B. AUC LEARNING OBJECTIVE FOR BOTH BALANCED AND IMBALANCED DATA

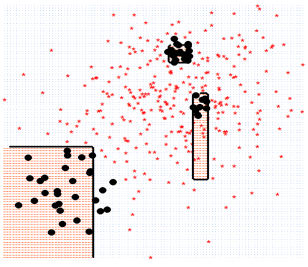
In this study, the proposed LOCS method sets AUC as the learning objective of classification. In fact, this works not only for imbalanced data classification but also for cases where the class distribution is balanced. This is because the AUC measure itself is insensitive to class distribution [14]. There are different degrees of class imbalance, and there is no absolute standard as to how much imbalance can be said to be imbalanced or balanced. Setting the AUC as the objective function for training the classifier works for both balanced and imbalanced data as the AUC approximates the accuracy as the number of positive instances TP+FN becomes closer to the number of negative instances FP+TN as shown in Equation (6). Therefore, if the classifier is trained



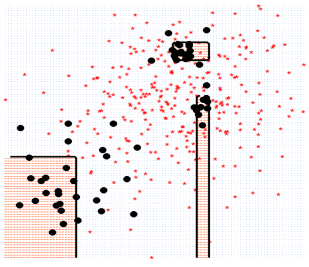
(a) *k*NN, overlapped positive 0%

(b) *k*NN, overlapped positive 25%

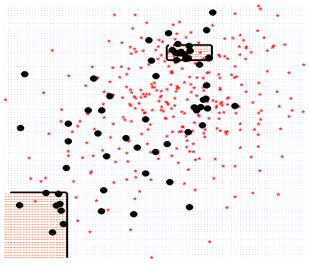
(c) *k*NN, overlapped positive 50%



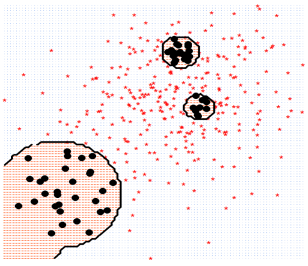
(d) CART, overlapped positive 0%



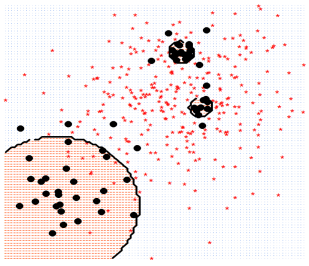
(e) CART, overlapped positive 25%



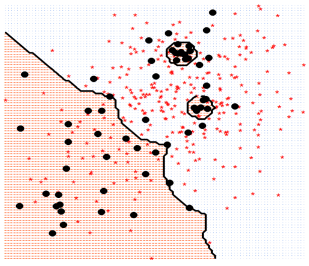
(f) CART, overlapped positive 50%



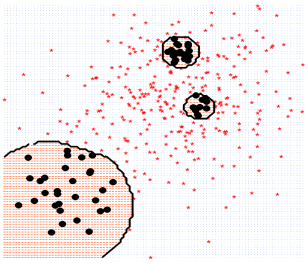
(g) LOCS(1:1),overlapped positive 0%



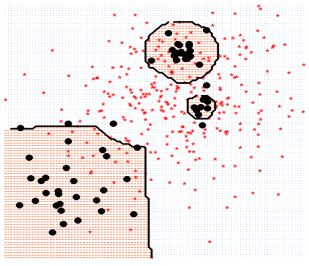
(h) LOCS(1:1),overlapped positive 25%



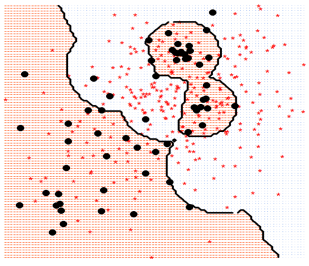
(i) LOCS(1:1),overlapped positive 50%



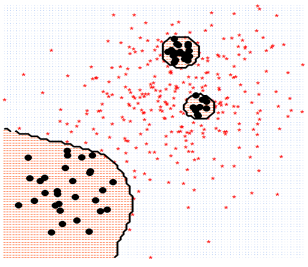
(j) LOCS(2:1),overlapped positive 0%



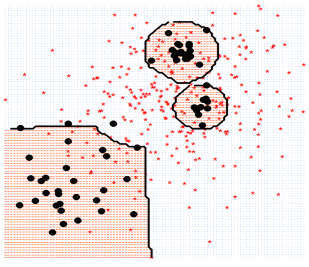
(k) LOCS(2:1),overlapped positive 25%



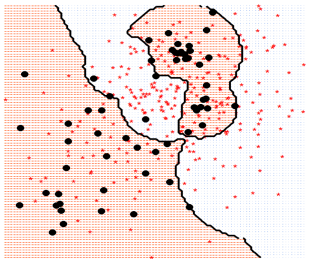
(l) LOCS(2:1),overlapped positive 50%



(m) LOCS(3:1),overlapped positive 0%



(n) LOCS(3:1),overlapped positive 25%



(o) LOCS(3:1),overlapped positive 50%

FIGURE 3. Variation of the decision boundaries for *k*NN, CART, LOCS (w_1:w_2 = 1:1), LOCS (w_1:w_2 = 2:1), LOCS (w_1:w_2 = 3:1) at varying overlap 0%, 25%, 50%.

to maximize the AUC, the advantage is that the degree of imbalance when training the classifier need not be considered as it can be used not only for the classification of imbalanced data but also for balanced data. Empirical evidence will be provided through an experiment in Section IV-B.

$$\begin{aligned} AUC &= \frac{1 + TPR - FPR}{2} \\ &= \frac{1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}}{2} \\ &\approx \frac{TP + FN + TP - FP}{2(TP + FN)} \\ &= \frac{TP + TN}{TP + FN + FP + TN} \\ &= Accuracy, \ as \ \frac{TP + FN}{FP + TN} \rightarrow 1 \end{aligned} \tag{6}$$

C. IMPORTANCE OF TPR AND FPR

In real-world problems, the importance of TPR and FPR may be different for each domain. For example, a precise classification of the positive class may be more important than misclassification of the negative class in the diagnosis of diseases in the medical field or for the detection of defects in the manufacturing industry. Another advantage of directly setting the AUC as an objective function is that although the AUC is composed of the same importance of TPR and FPR (1:1), the importance within the objective function can be freely controlled by assigning different weights such as (2:1) and (3:1) to TPR and FPR as needed by simply modifying the AUC. As shown in Equation (7), mAUC that assigns priorities w_1 and w_2 to TPR and FPR, respectively, is suggested as a learning objective of the proposed LOCS. As mentioned above, in the case of $w_1 = 1$, $w_2 = 1$, the mAUC is the same as the previously known AUC of Equation (5). Note that, with $w_2 = 1$, although the importance of TPR can increase and the TPR may increase correspondingly as w_1 is increased to a number greater than one, the FPR may also increase, and the AUC value of Equation (5) may decrease. With the proposed method, users can appropriately control the decision boundary of the classifier to suit their domain.

$$mAUC = \frac{1 + w_1 \cdot TPR - w_2 \cdot FPR}{2} \tag{7}$$

The decision boundary of the model generated by the proposed classifier was observed through a graphical example to examine whether the proposed method works as designed. The decision boundary of the proposed classifier and conventional classifiers is as shown in Fig. 3. The graphic example data was imbalanced data, with the number of positive and negative classes being 60 and 300, respectively. The positive class of the data consisted of three sub-clusters, and three datasets were prepared by designing such that 0%, 25%, and 50% of the number of positive instances overlap with the negative class. 'a%' means that $60 \times (a/100)$ positive instances are located in the negative class region. For comparison of decision boundaries, kNN ($k = 5$) and decision tree (CART) were selected as conventional classifiers. In addition, in the objective function of the proposed LOCS, the importance of TPR and FPR varied from w_1:w_2 = 1:1 to w_1:w_2 = 3:1 so as to observe how the decision boundary changes.

According to Fig. 3 (a) and (d), it was found that conventional classifiers (kNN, CART) classified well even in imbalanced data in areas where there was no overlap between the two classes. However, conventional classifiers could not properly classify positive classes as overlaps between classes increased when comparing Fig. 3 (b)(e) and (c)(f). However, in the case of LOCS, the decision boundary widened even in the class overlapped area to maximize the objective function mAUC as shown in Fig. 3 (g)(h)(i) where the overlap gradually increased. From another point of view, the change in the decision boundary based on the importance of LOCS is as shown in Fig. 3 (g)(j)(m), (h)(k)(n), (i)(l)(o). In particular, the decision boundary did not change significantly even when the TPR importance was increased because there was no overlap, as shown in Fig. 3 (g)(j)(m). On the other hand, the LOCS took a decision boundary towards the positive instances to bear the loss of FPR and increase the TPR in regions with severe overlap as the importance of TPR in the objective function increased, as shown in Fig. 3 (i)(l)(o). Therefore, to train the classifier by varying the importance of true positive in a specific situation, the decision boundary of the model can be precisely varied by adjusting the TPR importance of LOCS.

IV. NUMERICAL EXPERIMENTS

In this section, the performance of the proposed classifier LOCS was verified through experiments. The main aim of this experiment is to examine the design intent of the proposed method for various distributions of real-world data sets. The various environments of experiments conducted in this study are described in Section IV-A. The effect of maximizing the AUC for balanced data on the increase in accuracy was examined in Section IV-B. The performance of the proposed classifier LOCS was verified by comparing it with the conventional methodologies for imbalanced data classification such as resampling methods and CSL methods in Section IV-C. The importance of TPR and FPR within the mAUC of LOCS was varied as mentioned above to examine whether the performance changed as intended based on various real data sets in Section IV-D.

A. EXPERIMENTAL SETTINGS

The total experimental environments are described in this section. The characteristics of binary class real data sets to be used in the experiment, that is, IR, class description, total number of instances, number of attributes, number of positive instances, and number of negative instances, are as shown in Table 2. All datasets were obtained from UCI data repository [43] and KEEL data repository [44] and sorted in ascending order based on IR. The experimental data were selected by considering the various number of instances (214-5,820), attributes (7-32), and IRs (1.1-87.8). In this experiment, five data with less than two IR (#1-#5) were defined as balanced data, and 20 data with two IR or more (#6-#25) were defined

TABLE 2. Detail description of datasets.

#	Dataset	IR (n/p)	Class (p vs n)	# of instances	# of attr	# of pos	# of neg
1	Phishing [44]	1.1	(1) vs (0)	1352	9	650	702
2	Heart [41]	1.3	(1) vs (0)	270	13	120	150
3	Wisconsin [19]	1.9	(Malignant) vs (Benign)	683	9	239	444
4	Breast [19]	1.9	(1) vs (0)	683	9	239	444
5	Pima [19]	1.9	(1) vs (0)	768	8	268	500
6	glass0 [24]	2.1	(1) vs (reminders)	214	9	70	144
7	German [41]	2.3	(2) vs (1)	1000	24	300	700
8	yeast1 [24]	2.5	(NUC) vs (reminders)	1484	8	429	1055
9	glass0123vs456 [24]	3.2	(5,6,7) vs (1,2,3)	214	9	51	163
10	Absenteeism [44]	5.6	(3) vs (reminders)	740	20	112	628
11	turkiye student evaluation [14]	5.9	(13) vs (reminders)	5820	32	841	4979
12	ecoli3 [24]	8.6	(imU) vs (reminders)	336	7	35	301
13	ecoli067vs35 [24]	9.1	(imL, om) vs (cp, omL, pp)	222	9	22	200
14	yeast0256vs3789 [24]	9.1	(ME1, VAC, POX, ERL) vs (MIT, CYT, ME3, EXC)	1004	8	99	905
15	ecoli0347vs56 [24]	9.3	(om, omL) vs (cp, imL, imU, pp)	257	9	25	232
16	ecoli0147vs2356 [24]	10.6	(imS, imL, om, omL) vs (cp, im, imU, pp)	336	9	29	307
17	led7digit02456789vs1 [24]	11	(0,2,4,5,6,7,8) vs (9)	443	7	37	406
18	ecoli01vs5 [24]	11	(om) vs (cp, im)	240	9	20	220
19	yeast1vs7 [24]	14.3	(VAC) vs (NUC)	459	8	30	429
20	ecoli4 [24]	15.8	(om) vs (reminders)	336	7	20	316
21	yeast6 [24]	41.4	(EXC) vs (reminders)	1484	8	35	1449
22	winequality-white-3vs7 [45]	44	(3) vs (7)	900	11	20	880
23	poker-89vs6 [45]	58.4	(8,9) vs (6)	1485	10	25	1460
24	winequality-red-3vs5 [45]	68.1	(3) vs (5)	691	11	10	681
25	winequality-red-8 [14]	87.8	(8) vs (reminders)	1599	11	18	1581

as imbalanced data to be used in the experiment. For the datasets with originally more than two classes, we chose the classes with fewer instances as the positive class and integrated the other classes as the negative class. Therefore, the classification task involved discriminating specific classes from the others. In addition, all data were normalized.

The benchmarking classifiers for comparison were random sphere cover (RSC) [41], classification and regression tree (CART) [45], and support vector machine (SVM) [46]. Random over-sampling (ROS), random under-sampling (RUS), SMOTE [15], adaptive-SMOTE (AS) [47], under-sampling using sensitivity (USS) [48], and clustering and density-based hybrid (CDBH) [49] were used as resampling methods to solve the imbalance problem, and the cost-sensitive CART (CS-CART) and cost-sensitive SVM (CS-SVM) were used as CSL methods. SMOTE used k = 5, and the kernel of SVM was a radial basis function. Although LOCS was designed to maximize AUC, G-mean and F1-score [13], which are given below, with AUC were used for performance evaluation.

$$\text{G-mean} = \sqrt{TPR \times TNR} \tag{8}$$

$$\text{F1-score} = \frac{2TP}{2TP + FP + FN} \tag{9}$$

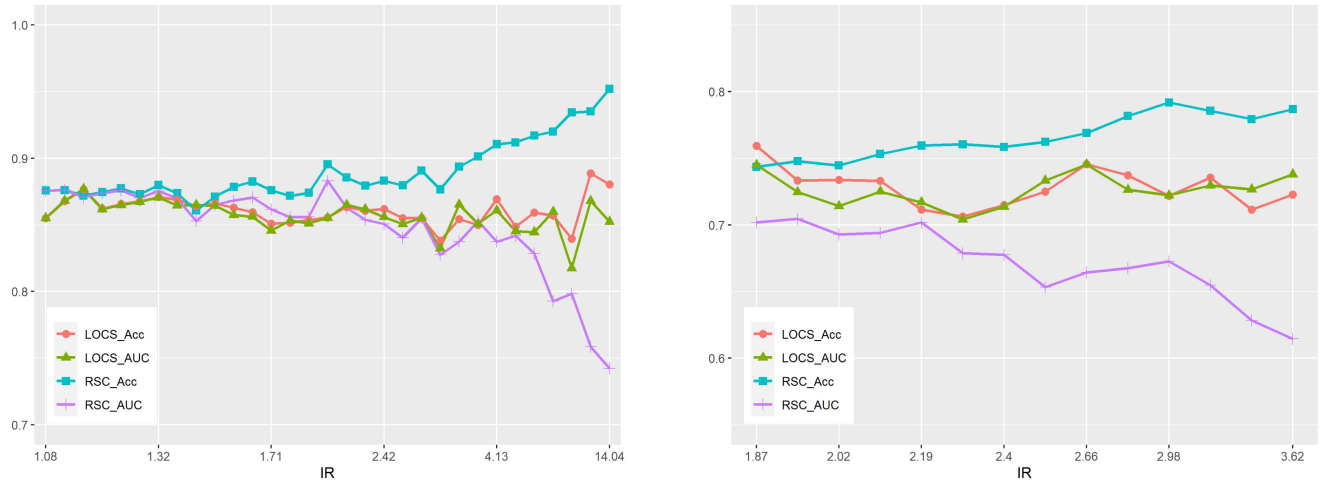
Fifty models were trained after assigning one to the majority instance and different values from 1 to 50 to the minority instance as misclassification cost for CSL. Among them, the cost with the best performance according to AUC, G-mean, and F1-score was respectively selected as the misclassification cost for the minority. For the parameter α of the sphere-based classifier RSC and the proposed method LOCS, the highest performance accoridng to the evaluation measures selected its value from 1 to 10 with consideration of various data sizes. The crossover probability, the mutation probability, and the population size, which are the GA parameters in the LOCS, were set to 0.8, 0.1, and 200, respectively. The 10-fold cross validation was applied in all experiments, the accuracy of Equation (2) was used for balanced cases, and the AUC of Equation (5), G-mean, and F1-score were used for imbalanced cases to evaluate the experimental results.

TABLE 3. Accuracy and AUC performances of LOCS in the balanced cases.

#	IR	CART (Acc)	SVM (Acc)	RSC (Acc)	LOCS (Acc)	LOCS (AUC)
1	1.1	0.876±0.017	0.905±0.017	0.880±0.029	0.861±0.037	0.864±0.024
2	1.3	0.804±0.063	0.819±0.066	0.819±0.083	0.781±0.073	0.812±0.056
3	1.9	0.944±0.014	0.969±0.018	0.974±0.015	0.960±0.016	0.970±0.014
4	1.9	0.952±0.028	0.968±0.018	0.975±0.017	0.965±0.016	0.973±0.019
5	1.9	0.746±0.059	0.771±0.048	0.755±0.062	0.747±0.067	0.745±0.046

B. BALANCED CASES

The performances of classifiers for relatively balanced data were compared. The accuracy of CART, SVM, RSC, LOCS, and AUC results of LOCS for data #1 to #5 with less than two IR are summarized in Table 3. In other words, LOCS (Acc) is the accuracy of the classifier trained with the proposed algorithm, and LOCS (AUC) is the AUC performance of the same classifier. For the five data, although there was a slight difference in accuracy performance for each of the four classifiers, it showed that there was no significant difference between the AUC value and the accuracy value of the LOCS that maximizes AUC. It experimentally supports the fact that the proposed algorithm works well for balanced data because the AUC maximization approximated the accuracy maximization as the number of positive instances approximated the number of negative instances, which implies as IR approximated one as the aforementioned theoretical analysis.



(a) phishing data. (b) pima data.

FIGURE 4. Comparison of accuracy and AUC performances between RSC and LOCS on phishing and pima data. We artificially increased the IR by repeatedly removing positive instances from data.

TABLE 4. AUC performances of resampling methods and LOCS in the imbalanced cases. R denotes rank of each method.

#	IR	ROS+RSC	R	RUS+RSC	R	SM+RSC	R	AS+RSC	R	USS+RSC	R	CDBH+RSC	R	LOCS	R
6	2.1	0.802 ±0.025	6	0.818 ±0.039	2	**0.821** ±0.023	1	0.798 ±0.016	7	0.807 ±0.006	5	0.81 ±0.015	3	0.809 ±0.028	4
7	2.3	0.636 ±0.007	6	0.667 ±0.011	2	0.667 ±0.013	3	0.627 ±0.001	7	0.648 ±0.002	5	0.658 ±0.008	4	**0.692** ±0.01	1
8	2.5	0.68 ±0.006	6	**0.709** ±0.002	1	0.698 ±0.001	4	0.678 ±0.004	7	0.703 ±0.013	3	0.697 ±0.008	5	0.707 ±0.008	2
9	3.2	0.939 ±0.023	4	0.932 ±0.007	7	**0.952** ±0.005	1	0.944 ±0.017	2	0.938 ±0.014	5	0.936 ±0.005	6	0.94 ±0.005	3
10	5.6	0.613 ±0.02	6	**0.667** ±0.031	1	0.639 ±0.001	4	0.588 ±0.023	7	0.637 ±0.049	5	0.641 ±0.002	3	0.648 ±0.001	2
11	5.9	0.571 ±0.003	7	**0.654** ±0.004	1	0.619 ±0.006	5	0.571 ±0.001	6	0.64 ±0.009	2	0.627 ±0.009	4	0.636 ±0.002	3
12	8.6	0.851 ±0.017	6	0.881 ±0.004	3	0.873 ±0.002	4	0.848 ±0.001	7	0.881 ±0.021	2	0.869 ±0.015	5	**0.899** ±0.002	1
13	9.1	0.886 ±0.022	3	0.854 ±0.011	6	0.888 ±0.002	2	0.886 ±0.039	4	0.845 ±0.056	7	**0.904** ±0.001	1	0.881 ±0.003	5
14	9.1	0.799 ±0.002	4	0.776 ±0.001	5	0.805 ±0.024	3	0.774 ±0.003	7	0.775 ±0.014	6	**0.808** ±0.001	1	0.807 ±0.008	2
15	9.3	0.905 ±0.043	3	0.869 ±0.029	6	0.893 ±0.007	4	0.889 ±0.009	5	0.867 ±0.025	7	0.905 ±0.023	2	**0.914** ±0.009	1
16	10.6	0.895 ±0.014	3	0.853 ±0.008	7	**0.901** ±0.029	1	0.893 ±0.008	4	0.865 ±0.009	6	0.896 ±0.002	2	0.881 ±0.042	5
17	11	0.828 ±0.076	6	0.824 ±0.011	7	0.839 ±0.055	5	0.841 ±0.014	4	0.874 ±0.002	2	0.87 ±0.033	3	**0.903** ±0.001	1
18	11	0.905 ±0.019	6	0.899 ±0.011	7	0.917 ±0.002	3	**0.928** ±0.014	1	0.926 ±0.014	2	0.915 ±0.002	4	0.915 ±0.005	5
19	14.3	0.699 ±0.008	6	0.734 ±0.038	2	0.707 ±0.053	5	0.684 ±0.042	7	0.714 ±0.035	4	0.731 ±0.042	3	**0.744** ±0.035	1
20	15.8	0.923 ±0.017	6	0.921 ±0.016	7	0.932 ±0.021	2	0.932 ±0.001	4	0.927 ±0.014	5	0.932 ±0.006	3	**0.949** ±0.012	1
21	41.4	0.809 ±0.012	6	0.871 ±0.015	3	0.807 ±0.03	7	0.827 ±0.01	5	0.882 ±0.015	2	0.85 ±0.03	4	**0.889** ±0.019	1
22	44	0.683 ±0.018	6	**0.735** ±0.024	1	0.663 ±0.016	7	0.72 ±0.021	2	0.716 ±0.001	4	0.693 ±0.003	5	0.718 ±0.021	3
23	58.4	0.861 ±0.018	5	0.854 ±0.006	6	0.877 ±0.035	4	**0.963** ±0.023	1	0.816 ±0.012	7	0.951 ±0.003	2	0.907 ±0.002	3
24	68.1	0.569 ±0.035	7	**0.752** ±0.011	1	0.638 ±0.035	5	0.657 ±0.037	4	0.701 ±0.045	3	0.63 ±0.005	6	0.714 ±0.002	2
25	87.8	0.558 ±0.017	7	0.743 ±0.049	3	0.612 ±0.001	6	0.672 ±0.002	5	**0.791** ±0.028	1	0.695 ±0.004	4	0.785 ±0.038	2
Average rank			5.45		3.9		3.8		4.8		4.15		3.5		2.4

Wilcoxon paired rank test statistic (p-value)	ROS+RSC	RUS+RSC	SM+RSC	AS+RSC	USS+RSC	CDBH+RSC
LOCS	201 (0.00006)	167 (0.0192)	182 (0.0027)	181 (0.0032)	191 (0.0006)	165 (0.024)

The robustness of the proposed method for changes in IR was compared with the conventional sphere-based classifier RSC through an experiment. A comparison of the accuracy and the AUC of RSC and LOCS by removing 20 and 10 positive instances from each phishing data (#1) and pima data (#5) repeatedly is shown in Fig. 4. As a result of repeatedly removing positive instances until no positive spheres were constructed, the IRs of the two data increased to 14.04 and 3.62. For both data, the accuracy of RSC increased as IR increased, while the AUC value of RSC decreased. This is because the classifier was trained mainly on negative classes to increase accuracy as the IR increased as expected. On the contrary, as the influence of positive instances gradually decreased, the AUC value continuously decreased. In contrast, LOCS was similar to the RSC in terms of both accuracy and AUC when the IR of data was low, and the AUC

TABLE 5. G-mean performances of resampling methods and LOCS in the imbalanced cases. R denotes rank of each method.

#	IR	ROS+RSC	R	RUS+RSC	R	SM+RSC	R	AS+RSC	R	USS+RSC	R	CDBH+RSC	R	LOCS	R
6	2.1	0.798 ±0.022	6	0.812 ±0.038	2	**0.818** ±0.016	1	0.788 ±0.019	7	0.8 ±0.007	5	0.805 ±0.018	3	0.804 ±0.027	4
7	2.3	0.606 ±0.013	6	0.659 ±0.014	2	0.656 ±0.031	3	0.595 ±0.001	7	0.645 ±0.019	5	0.651 ±0.009	4	**0.689** ±0.009	1
8	2.5	0.663 ±0.006	6	**0.706** ±0.001	1	0.69 ±0.001	5	0.656 ±0.001	7	0.7 ±0.008	3	0.695 ±0.008	4	0.705 ±0.007	2
9	3.2	0.937 ±0.024	4	0.929 ±0.008	7	**0.95** ±0.005	1	0.943 ±0.017	2	0.934 ±0.016	5	0.934 ±0.005	6	0.939 ±0.005	3
10	5.6	0.547 ±0.017	6	**0.66** ±0.036	1	0.598 ±0.017	5	0.479 ±0.02	7	0.629 ±0.055	4	0.632 ±0.009	3	0.639 ±0.001	2
11	5.9	0.458 ±0.004	6	**0.651** ±0.007	1	0.589 ±0.009	5	0.45 ±0.02	7	0.637 ±0.008	2	0.621 ±0.016	4	0.632 ±0.003	3
12	8.6	0.817 ±0.011	6	0.872 ±0.001	2	0.84 ±0.031	4	0.812 ±0.026	7	0.853 ±0.054	3	0.837 ±0.013	5	**0.891** ±0.003	1
13	9.1	0.87 ±0.022	3	0.839 ±0.008	6	0.876 ±0.036	2	0.852 ±0.075	4	0.813 ±0.081	7	**0.893** ±0.005	1	0.851 ±0.035	5
14	9.1	0.777 ±0.01	4	0.772 ±0.002	6	0.789 ±0.032	3	0.749 ±0.014	7	0.773 ±0.013	5	**0.803** ±0.001	1	0.796 ±0.003	2
15	9.3	0.891 ±0.055	3	0.864 ±0.03	5	0.884 ±0.008	4	0.856 ±0.02	7	0.86 ±0.028	6	0.899 ±0.027	2	**0.908** ±0.008	1
16	10.6	0.88 ±0.02	4	0.844 ±0.025	7	**0.891** ±0.038	1	0.882 ±0.005	3	0.859 ±0.008	6	0.887 ±0.001	2	0.874 ±0.012	5
17	11	0.74 ±0.174	7	0.788 ±0.042	6	0.791 ±0.116	5	0.792 ±0.048	4	0.867 ±0.003	2	0.834 ±0.072	3	**0.893** ±0.001	1
18	11	0.89 ±0.021	6	0.89 ±0.019	7	0.905 ±0.002	3	**0.918** ±0.017	1	0.918 ±0.013	2	0.903 ±0.002	4	0.903 ±0.005	5
19	14.3	0.606 ±0.039	6	0.721 ±0.041	2	0.636 ±0.055	5	0.584 ±0.09	7	0.701 ±0.049	4	0.705 ±0.048	3	**0.729** ±0.04	1
20	15.8	0.912 ±0.02	7	0.914 ±0.019	6	0.922 ±0.024	4	0.924 ±0.001	3	0.921 ±0.016	5	0.924 ±0.005	2	**0.943** ±0.015	1
21	41.4	0.77 ±0.011	7	0.867 ±0.003	3	0.771 ±0.06	6	0.801 ±0.016	5	0.878 ±0.018	2	0.834 ±0.058	4	**0.884** ±0.02	1
22	44	0.446 ±0.108	6	**0.704** ±0.052	1	0.428 ±0.078	7	0.59 ±0.008	4	0.683 ±0.031	2	0.556 ±0.027	5	0.622 ±0.071	3
23	58.4	0.827 ±0.08	6	0.845 ±0.007	5	0.859 ±0.046	4	**0.96** ±0.025	1	0.8 ±0.006	7	0.945 ±0.003	2	0.901 ±0.003	3
24	68.1	0.149 ±0.07	7	**0.61** ±0.009	1	0.296 ±0.071	5	0.34 ±0.066	4	0.557 ±0.082	2	0.291 ±0.001	6	0.545 ±0.002	3
25	87.8	0.175 ±0.05	7	0.696 ±0.048	3	0.287 ±0.028	6	0.468 ±0.06	5	**0.754** ±0.06	1	0.516 ±0.028	4	0.749 ±0.065	2
Average rank			5.65		3.7		3.95		4.95		3.9		3.4		2.45

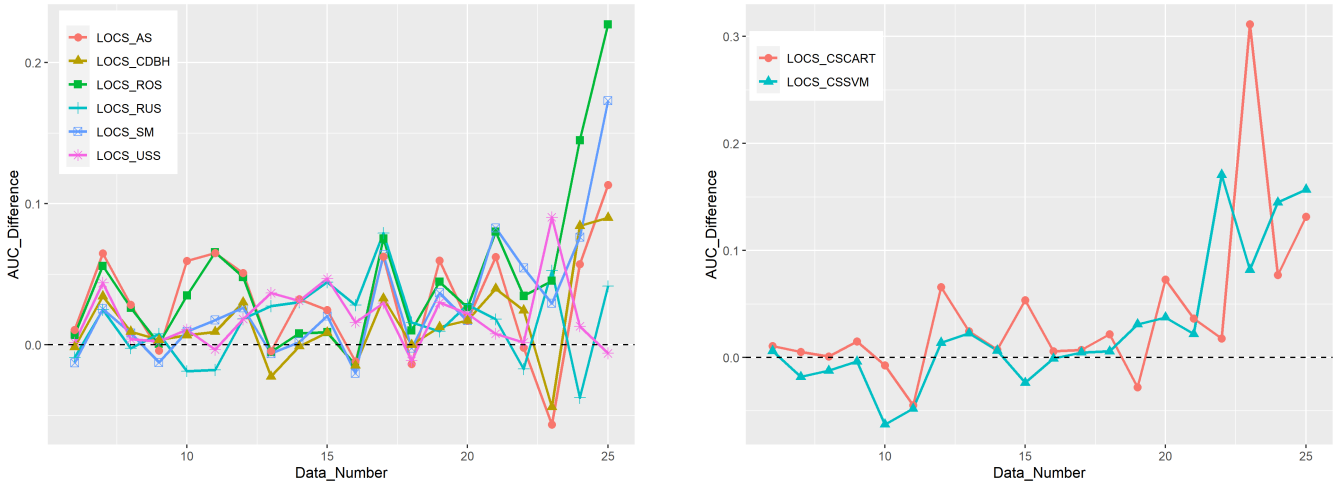
Wilcoxon paired rank test						
statistic (p-value)	ROS+RSC	RUS+RSC	SM+RSC	AS+RSC	USS+RSC	CDBH+RSC
LOCS	202 (0.00005)	151 (0.0897)	187 (0.0012)	189 (0.0009)	165 (0.024)	167 (0.0192)

TABLE 6. F1-score performances of resampling methods and LOCS in the imbalanced cases. R denotes rank of each method.

#	IR	ROS+RSC	R	RUS+RSC	R	SM+RSC	R	AS+RSC	R	USS+RSC	R	CDBH+RSC	R	LOCS	R
6	2.1	0.731 ±0.03	6	0.743 ±0.05	2	**0.755** ±0.019	1	0.726 ±0.023	7	0.733 ±0.019	4	0.738 ±0.023	3	0.732 ±0.032	5
7	2.3	0.478 ±0.013	6	0.55 ±0.008	2	0.534 ±0.036	4	0.467 ±0.002	7	0.527 ±0.004	5	0.544 ±0.02	3	**0.573** ±0.01	1
8	2.5	0.544 ±0.009	6	**0.585** ±0.002	1	0.571 ±0.001	4	0.539 ±0.006	7	0.577 ±0.016	3	0.571 ±0.009	5	0.582 ±0.01	2
9	3.2	0.889 ±0.03	3	0.856 ±0.01	7	**0.902** ±0.008	1	0.897 ±0.048	2	0.864 ±0.017	6	0.878 ±0.001	4	0.876 ±0.004	5
10	5.6	0.335 ±0.027	6	**0.371** ±0.036	1	0.36 ±0.007	4	0.29 ±0.041	7	0.345 ±0.05	5	0.362 ±0.017	3	0.365 ±0.001	2
11	5.9	0.262 ±0.006	6	**0.352** ±0.004	1	0.333 ±0.008	5	0.223 ±0.014	7	0.341 ±0.01	3	0.335 ±0.009	4	0.342 ±0.002	2
12	8.6	0.629 ±0.056	4	0.61 ±0.003	6	**0.658** ±0.01	1	0.623 ±0.003	5	0.593 ±0.035	7	0.63 ±0.06	3	0.655 ±0.03	2
13	9.1	**0.78** ±0.047	1	0.568 ±0.022	6	0.744 ±0.003	3	0.746 ±0.004	2	0.555 ±0.059	7	0.726 ±0.046	4	0.682 ±0.054	5
14	9.1	**0.614** ±0.003	1	0.435 ±0.009	6	0.583 ±0.049	2	0.53 ±0.041	4	0.416 ±0.008	7	0.516 ±0.015	5	0.578 ±0.026	3
15	9.3	0.78 ±0.071	2	0.565 ±0.061	6	**0.792** ±0.023	1	0.735 ±0.032	4	0.527 ±0.025	7	0.714 ±0.014	5	0.747 ±0.049	3
16	10.6	0.773 ±0.063	2	0.495 ±0.025	7	0.77 ±0.078	3	**0.775** ±0.011	1	0.509 ±0.023	6	0.696 ±0.057	4	0.644 ±0.05	5
17	11	0.608 ±0.149	5	0.466 ±0.005	7	0.652 ±0.053	4	**0.675** ±0.02	1	0.52 ±0.009	6	0.664 ±0.072	2	0.662 ±0.01	3
18	11	0.82 ±0.028	5	0.722 ±0.008	7	0.837 ±0.014	2	**0.84** ±0.005	1	0.772 ±0.077	6	0.825 ±0.022	4	0.826 ±0.007	3
19	14.3	**0.422** ±0.016	1	0.273 ±0.008	7	0.388 ±0.062	2	0.33 ±0.034	4	0.286 ±0.042	6	0.336 ±0.033	3	0.302 ±0.027	5
20	15.8	0.833 ±0.014	2	0.603 ±0.01	6	**0.853** ±0.014	1	0.825 ±0.012	3	0.6 ±0.035	7	0.82 ±0.042	5	0.822 ±0.007	4
21	41.4	**0.562** ±0.022	1	0.294 ±0.017	6	0.515 ±0.059	2	0.468 ±0.051	3	0.275 ±0.014	7	0.426 ±0.043	5	0.44 ±0.031	4
22	44	**0.373** ±0.061	1	0.106 ±0.001	6	0.309 ±0.031	2	0.268 ±0.018	3	0.105 ±0.021	7	0.245 ±0.112	4	0.179 ±0.063	5
23	58.4	0.752 ±0.067	3	0.122 ±0.007	6	0.774 ±0.086	2	**0.826** ±0.014	1	0.12 ±0.001	7	0.733 ±0.031	4	0.23 ±0.081	5
24	68.1	0.1 ±0.047	4	0.089 ±0.009	6	**0.167** ±0.054	1	0.132 ±0.005	2	0.087 ±0.026	7	0.114 ±0.018	3	0.091 ±0.001	5
25	87.8	0.105 ±0.043	4	0.081 ±0.016	5	**0.147** ±0.015	1	0.125 ±0.036	3	0.068 ±0.002	6	0.134 ±0.009	2	0.057 ±0.008	7
Average rank			3.45		5.05		2.3		3.7		5.95		3.75		3.8

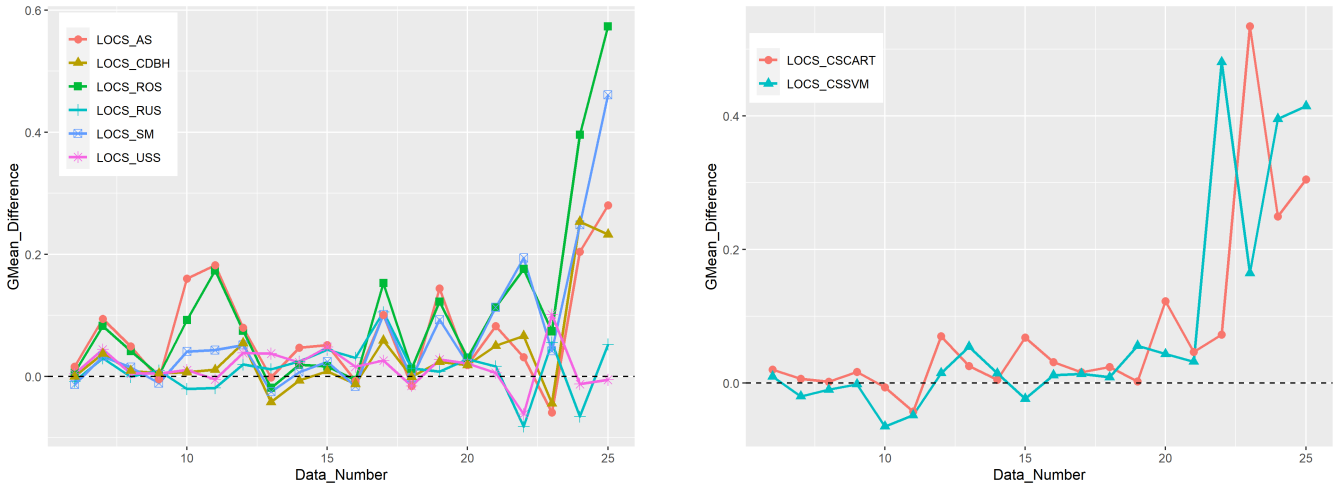
Wilcoxon paired rank test						
statistic (p-value)	ROS+RSC	RUS+RSC	SM+RSC	AS+RSC	USS+RSC	CDBH+RSC
LOCS	65 (0.1429)	188 (0.001)	30 (0.0037)	87 (0.5217)	204 (0.00002)	87 (0.5217)

performance remained robust even when the IR increased. As a result, it was found that setting the objective function as AUC in LOCS was effective in classifying both imbalanced and balanced data.



(a) (b)

FIGURE 5. AUC differences of existing methods from LOCS. (a) AUC differences of resampling methods from LOCS. (b) AUC differences of CSL methods from LOCS.



(a) (b)

FIGURE 6. G-mean differences of existing methods from LOCS. (a) G-mean differences of resampling methods from LOCS. (b) G-mean differences of CSL methods from LOCS.

C. IMBALANCED CASES

The performance of LOCS was compared with representative methods of two approaches, resampling approaches and CSL approaches, to solve the imbalanced classification problem. To this end, 20 imbalanced datasets ranging from glass0 (#6) to wine-quality-red-8 (#25) were used. The comparison of AUC, G-mean, and F1-score performances of the resampling methods and LOCS is summarized as shown in Table 4, Table 5, and Table 6, respectively. ROS, RUS, SMOTE, AS, USS, and CDBH were used as comparison methods, and RSC classifier was used as the base learner of each comparison method. The data number and IR, the rank of the performance in the right column of each method, and the average rank in the last row are as shown in the tables In addition, the results of the method with the highest performance for each data are emphasized in bold.

The AUC results are shown in Table 4. Although one of the resampling methods showed better results than LOCS in #6, #8, #9, #10, #11, #13, #14, #16, #18, #22, #23, #24, and #25 data, LOCS showed the best result in all other data. The average rank of all data showed that LOCS ranked 2.4, ROS, RUS, SM, AS, USS, and CDBH ranked 5.45, 3.9, 3.8, 4.8, 4.15, and 3.5, respectively. To make further comparisons, we conducted statistical analyses of the experimental results by adopting the Wilcoxon Signed-Rank test [50]. From the bottom of Table 4, we can see that LOCS has significantly better performances than others because the p-value are very small.

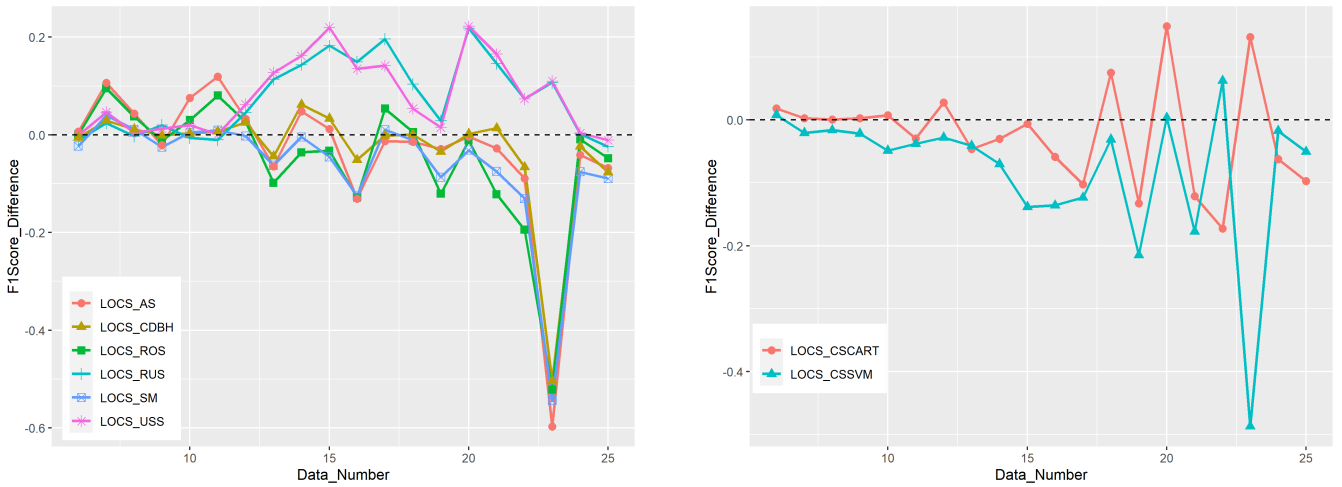


FIGURE 7. F1-score differences of existing methods from LOCS. (a) F1-score differences of resampling methods from LOCS. (b) F1-score differences of CSL methods from LOCS.

Likewise, the G-mean results and the F1-score results are shown in Table 5 and Table 6 respectively. The G-mean results are very similar to the AUC results. According to the average rank, the best performing method was LOCS, which was followed by CDBH, RUS, USS, SM, AS, and ROS in sequence. The Wilcoxon Signed-Rank tests supported the significant better performance of the proposed method. However, the F1-score results, which are shown in Table 6, were different from the AUC and G-mean results. The best performance came with SM which average rank was 2.3. LOCS was comparable with ROS, AS, and CDBH, which can be seen from the insignificant p-values of the Wilcoxon Signed-Rank tests. These are due to that F1-score evaluates *TP* and *FP*, i.e. number of instances, while AUC and G-mean evaluate the rates *TPR*, *FPR*, and *TNR*. Although *FP* is small, F1-score is small if *TP* is not large enough. This is why the F1-scores tend to decrease as IR increases, which can be seen from Table 6.

A demonstration of the AUC values of ROS, RUS, SM, AS, USS, and CDBH subtracted from the AUC values of LOCS in each data from the results of Table 4 is shown in Fig. 5 (a). If the difference is greater than zero, it implies that the performance of LOCS was better. Except for some data, the AUC value difference was found to be greater than 0 in most of the data, and in particular, the AUC difference was found to be greater in the data with a large IR. The same demonstrations for G-mean and F1-score are shown in Fig. 6 (a) and Fig. 7 (a) respectively. The G-mean graph provides the same interpretation with the AUC case, whereas the F1-score graph shows that LOCS was worse than RUS, SM, and AD for several datasets. Considering that the proposed method was designed to maximize AUC, we believe that these results are reasonable.

The results of comparing LOCS with CSL for the same 20 imbalanced data are shown in Table 7, Table 8, and Table 9, where each table corresponds to AUC, G-Mean, and F-score, respectively. CART and SVM, which are widely used, were selected as the base learners of CSL. As mentioned above, the classifier was trained by fixing the misclassification cost of the negative class to one and varying the misclassification cost of the positive class from 1 to 50. Among 50 classifiers, the classifier with the highest performance according to each evaluation measure was selected, and its performance was shown along with the cost. In the average rank of each method, LOCS ranked 1.5, CS-CART

TABLE 7. AUC performances of CSL methods and LOCS in the imbalanced cases. R and C denote rank and misclassification cost corresponding to highest AUC, respectively.

#	IR	CS_CART	R	C	CS_SVM	R	C	LOCS	R
6	2.1	0.798±0.010	3	25	0.802±0.002	2	2	**0.809**±0.028	1
7	2.3	0.687±0.001	3	3	**0.710**±0.01	1	2	0.692±0.01	2
8	2.5	0.706±0.005	3	3	**0.719**±0.002	1	2	0.707±0.008	2
9	3.2	0.925±0.004	3	3	**0.944**±0.007	1	3	0.940±0.005	2
10	5.6	0.655±0.001	2	50	**0.711**±0.003	1	8	0.648±0.001	3
11	5.9	0.681±0.004	2	7	**0.684**±0.006	1	6	0.636±0.002	3
12	8.6	0.833±0.018	3	33	0.885±0.003	2	3	**0.899**±0.002	1
13	9.1	0.857±0.040	3	15	0.859±0.041	2	11	**0.881**±0.003	1
14	9.1	0.799±0.001	3	6	0.800±0.001	2	5	**0.807**±0.008	1
15	9.3	0.860±0.001	3	2	**0.937**±0.003	1	6	0.914±0.009	2
16	10.6	0.875±0.023	3	2	**0.882**±0.002	1	10	0.881±0.042	2
17	11	0.896±0.001	3	2	0.898±0.001	2	2	**0.903**±0.001	1
18	11	0.893±0.016	3	3	0.909±0.019	2	3	**0.915**±0.005	1
19	14.3	**0.772**±0.029	1	6	0.713±0.019	3	15	0.744±0.035	2
20	15.8	0.876±0.040	3	20	0.912±0.034	2	14	**0.949**±0.012	1
21	41.4	0.853±0.013	3	32	0.867±0.001	2	14	**0.889**±0.019	1
22	44	0.700±0.054	2	4	0.547±0.001	3	8	**0.718**±0.02	1
23	58.4	0.596±0.032	3	27	0.825±0.001	2	10	**0.907**±0.002	1
24	68.1	0.637±0.001	2	4	0.569±0.036	3	11	**0.714**±0.002	1
25	87.8	0.653±0.072	2	13	0.628±0.036	3	31	**0.785**±0.038	1
Average rank		2.65				1.85			1.5

Wilcoxon paired rank test statistic (p-value)	CS_CART	CS_SVM
LOCS	178 (0.0049)	148 (0.114)

TABLE 8. G-mean performances of CSL methods and LOCS in the imbalanced cases. R and C denote rank and misclassification cost corresponding to highest G-mean, respectively.

#	IR	CS_CART	R	C	CS_SVM	R	C	LOCS	R
6	2.1	0.784±0.008	3	2	0.794±0.003	2	2	**0.804**±0.027	1
7	2.3	0.682±0.001	3	3	**0.708**±0.004	1	3	0.689±0.009	2
8	2.5	0.702±0.003	3	3	**0.715**±0.001	1	2	0.705±0.007	2
9	3.2	0.922±0.003	3	3	**0.940**±0.007	1	3	0.939±0.005	2
10	5.6	0.646±0.007	2	8	**0.704**±0.005	1	8	0.639±0.001	3
11	5.9	0.675±0.001	2	5	**0.680**±0.004	1	5	0.632±0.003	3
12	8.6	0.821±0.005	3	33	0.876±0.003	2	3	**0.891**±0.003	1
13	9.1	0.825±0.069	2	15	0.796±0.046	3	11	**0.851**±0.035	1
14	9.1	0.791±0.003	2	11	0.781±0.002	3	7	**0.796**±0.003	1
15	9.3	0.839±0.011	3	10	**0.931**±0.005	1	6	0.908±0.008	2
16	10.6	0.843±0.055	3	2	0.863±0.005	2	10	**0.874**±0.012	1
17	11	0.876±0.002	3	2	0.879±0.002	2	2	**0.893**±0.001	1
18	11	0.879±0.018	3	3	0.894±0.022	2	3	**0.903**±0.005	1
19	14.3	0.726±0.064	2	6	0.673±0.002	3	15	**0.729**±0.040	1
20	15.8	0.820±0.106	3	20	0.899±0.041	2	14	**0.943**±0.015	1
21	41.4	0.837±0.018	3	32	0.851±0.006	2	14	**0.884**±0.020	1
22	44	0.549±0.048	2	8	0.141±0.001	3	8	**0.622**±0.071	1
23	58.4	0.367±0.06	3	27	0.736±0.029	2	10	**0.901**±0.003	1
24	68.1	0.295±0.001	2	4	0.149±0.071	3	11	**0.545**±0.002	1
25	87.8	0.444±0.139	2	15	0.334±0.040	3	31	**0.749**±0.065	1
Average rank			2.6			2			1.4

Wilcoxon paired rank test statistic (p-value)	CS_CART	CS_SVM
LOCS	194 (0.0003)	157 (0.0532)

TABLE 9. F1-score performances of CSL methods and LOCS in the imbalanced cases. R and C denote rank and misclassification cost corresponding to highest F1-score, respectively.

#	IR	CS_CART	R	C	CS_SVM	R	C	LOCS	R
6	2.1	0.714±0.011	3	25	0.724±0.002	2	2	**0.732**±0.032	1
7	2.3	0.571±0.002	3	3	**0.594**±0.005	1	3	0.573±0.01	3
8	2.5	0.581±0.008	3	3	**0.598**±0.003	1	2	0.582±0.01	2
9	3.2	0.874±0.006	3	3	**0.898**±0.007	1	3	0.876±0.004	2
10	5.6	0.358±0.009	3	8	**0.414**±0.003	1	5	0.365±0.001	2
11	5.9	0.372±0.001	2	5	**0.38**±0.006	1	5	0.342±0.002	3
12	8.6	0.628±0.042	3	4	**0.683**±0.005	1	3	0.655±0.03	2
13	9.1	**0.728**±0.045	1	1	0.723±0.032	2	5	0.682±0.054	3
14	9.1	0.609±0.01	2	2	**0.648**±0.005	1	3	0.578±0.026	3
15	9.3	0.753±0.028	2	2	**0.885**±0.016	1	3	0.747±0.049	3
16	10.6	0.703±0.075	2	2	**0.78**±0.012	1	4	0.644±0.05	3
17	11	0.765±0.001	2	2	**0.786**±0.002	1	2	0.662±0.01	3
18	11	0.751±0.034	3	3	**0.857**±0.033	1	3	0.826±0.007	2
19	14.3	0.434±0.001	2	3	**0.517**±0.014	1	4	0.302±0.027	3
20	15.8	0.673±0.104	3	8	0.818±0.007	2	4	**0.822**±0.007	1
21	41.4	0.561±0.023	2	2	**0.617**±0.014	1	3	0.44±0.031	3
22	44	**0.352**±0.076	1	4	0.117±0.001	3	8	0.179±0.063	2
23	58.4	0.098±0.023	3	23	**0.717**±0.014	1	10	0.023±0.081	2
24	68.1	**0.153**±0.005	1	4	0.108±0.082	2	11	0.091±0.001	3
25	87.8	**0.155**±0.029	1	13	0.108±0.013	2	31	0.057±0.008	3
Average rank			2.25			1.35			2.4

Wilcoxon paired rank test statistic (p-value)	CS_CART	CS_SVM
LOCS	73 (0.2455)	16 (0.0003)

ranked 2.65, and CS-SVM ranked 1.85, in terms of AUC, indicating that LOCS had the best performance on average. The Wilcoxon Signed-Rank test results with the small p-values are shown at the bottom of Table 7. The ranking of three methods are the same in the G-mean table. However, the F1-score table shows a difference that the cost-sensitive SVM performed the best. When comparing the positive

TABLE 10. Classification performances of LOCS with different importance settings.

#	IR		RSC	LOCS (w_1:w_2=1:1)	LOCS (w_1:w_2=2:1)	LOCS (w_1:w_2=3:1)
6	2.1	AUC	0.787	0.848	0.815	0.764
		TPR	0.729	0.9	0.929	0.929
		FPR	0.154	0.204	0.298	0.4
7	2.3	AUC	0.616	0.663	0.673	0.666
		TPR	0.4	0.67	0.757	0.783
		FPR	0.167	0.344	0.411	0.451
8	2.5	AUC	0.666	0.711	0.687	0.682
		TPR	0.441	0.795	0.895	0.923
		FPR	0.108	0.373	0.521	0.558
9	3.2	AUC	0.956	0.95	0.947	0.947
		TPR	0.943	0.967	0.967	0.967
		FPR	0.031	0.067	0.073	0.073
10	5.6	AUC	0.58	0.672	0.658	0.639
		TPR	0.277	0.678	0.739	0.767
		FPR	0.116	0.333	0.424	0.489
11	5.9	AUC	0.551	0.614	0.597	0.602
		TPR	0.187	0.729	0.788	0.821
		FPR	0.085	0.5	0.595	0.617
12	8.6	AUC	0.772	0.898	0.891	0.858
		TPR	0.583	0.908	0.908	0.908
		FPR	0.04	0.113	0.126	0.192
13	9.1	AUC	0.846	0.902	0.909	0.889
		TPR	0.717	0.9	0.933	0.933
		FPR	0.025	0.095	0.115	0.155
14	9.1	AUC	0.764	0.822	0.774	0.732
		TPR	0.546	0.717	0.727	0.788
		FPR	0.018	0.073	0.178	0.325
15	9.3	AUC	0.858	0.904	0.912	0.869
		TPR	0.733	0.833	0.883	0.883
		FPR	0.017	0.026	0.06	0.146
16	10.6	AUC	0.854	0.889	0.862	0.857
		TPR	0.733	0.833	0.867	0.867
		FPR	0.026	0.055	0.143	0.153
17	11	AUC	0.861	0.904	0.904	0.848
		TPR	0.742	0.917	0.917	0.917
		FPR	0.02	0.108	0.108	0.221
18	11	AUC	0.918	0.909	0.877	0.859
		TPR	0.85	0.85	0.85	0.85
		FPR	0.014	0.032	0.095	0.132
19	14.3	AUC	0.649	0.71	0.7	0.725
		TPR	0.333	0.633	0.733	0.767
		FPR	0.035	0.213	0.334	0.317
20	15.8	AUC	0.922	0.964	0.937	0.907
		TPR	0.85	0.95	0.95	0.95
		FPR	0.006	0.022	0.076	0.136
21	41.4	AUC	0.769	0.888	0.873	0.875
		TPR	0.55	0.842	0.842	0.842
		FPR	0.012	0.065	0.096	0.091
22	44	AUC	0.645	0.683	0.7	0.728
		TPR	0.3	0.5	0.6	0.65
		FPR	0.009	0.134	0.2	0.194
23	58.4	AUC	0.841	0.857	0.844	0.847
		TPR	0.683	0.783	0.767	0.75
		FPR	0.001	0.068	0.078	0.057
24	68.1	AUC	0.495	0.659	0.659	0.643
		TPR	0	0.4	0.4	0.4
		FPR	0.01	0.082	0.082	0.113
25	87.8	AUC	0.572	0.759	0.716	0.712
		TPR	0.15	0.7	0.6	0.6
		FPR	0.006	0.182	0.167	0.176

misclassification cost, which showed the best performance in CSL, with the IR of each data, it was found that the cost that showed the best performance was not significantly related to IR. In addition, among the nine data (IR 11-IR 87.8) from led7digit02456789vs (#17) to winequality-red-8 (#25), the AUC and G-mean performances of LOCS were high in all data except for the data of yeast1vs7 (#19). This was shown in Fig. 5 (b) and Fig. 6 (b). These figures are the results of plotting the difference in AUC and G-mean values of LOCS and CSL methods in data order. As in the figures for comparing with the resampling methods, a value greater than zero indicated better performance of LOCS. It was found that the difference in performance between the LOCS and CSL methods is relatively large in data with a large IR.

D. DIFFERENT IMPORTANCE FOR TPR AND FPR

An experiment was conducted to observe how the TPR importance (w_1) and FPR importance (w_2) of Equation (7) affect the classification result. After fixing $w_2 = 1$, and training the classifier by varying w_1 from one to three, the performance has been summarized in Table 10. The performance of RSC was included in the table as a baseline. The results of #8 data show that, as w_1 of LOCS increased from one to three, TPR values increased to 0.795, 0.895, and 0.923 as intended. However, as the FPR values also increased to 0.373, 0.521, and 0.558, the AUC values decreased to 0.711, 0.687, and 0.682. This means that FP increased as the TPR importance increased to classify positive instances more accurately. The increase in the TPR value as the TPR importance increased was also found in data #6, #7, #10, #11, #13, #14, #15, #16, #19, and #22. In some cases, a further increase in AUC was found as the degree of increase in FPR was relatively small compared to the degree of increase in TPR with an increase in w_1. The TPR value did not change even when w_1 was increased in the case of #9, #12, #17, #18, #20, #21, and #24 data. On the other hand, it was found that the FPR value gradually increased and the AUC value gradually decreased.

V. CONCLUSION

In this study, a new classification method, called LOCS, was proposed to solve the imbalanced data learning problem based on a sphere-based classifier. The proposed algorithm was designed to set the AUC, which is a widely used evaluation measure in imbalanced data classification, as a learning objective to construct a sphere classifier that maximizes this parameter. The advantage of the proposed method is that it can be applied regardless of the degree of class imbalance because the closer the two classes are to the balanced state, the closer the AUC is to accuracy. In addition, it can be modified appropriately for the application domain and utilized by setting different importance levels for TPR and FPR. The effectiveness of LOCS was verified in numerical experiments based on 25 real datasets. The best performance was shown in 13 out of 20 imbalanced datasets in comparison experiments with conventional resampling approaches, and the best performance was shown in 12 out of 20 datasets in comparison experiments with conventional CSL methods. The CSL took a long time to learn because different misclassification costs had to be tried, while the proposed algorithm showed satisfactory performance with single learning. It was found that LOCS can produce robust classification results, even with changes in class distribution through experiments that varied in IR. In addition, experiments have shown that the proposed algorithm can be effectively used in practical domains by controlling the importance of TPR and FPR. Further studies are required to extend the proposed method to multi-class problems, because this study was limited to binary classification. A promising future research direction is to employ, instead of AUC, another metrics such as G-mean and F-scores as learning objectives in the proposed learning framework.

REFERENCES

[1] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, New York, NY, USA, 2014, pp. 1–10, doi: 10.1145/2601248.2601294.

[2] H. Parvin, B. Minaei-Bidgoli, and H. Alinejad-Rokny, "A new imbalanced learning and dictions tree method for breast cancer diagnosis," *J. Bionanosci.*, vol. 7, no. 6, pp. 673–678, 2013.

[3] S. Kim, H. Kim, and Y. Namkoong, "Ordinal classification of imbalanced data with application in emergency and disaster information services," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 50–56, Sep./Oct. 2016.

[4] E. Ramentol, I. Gondres, S. Lajes, R. Bello, Y. Caballero, C. Cornelis, and F. Herrera, "Fuzzy-rough imbalanced learning for the diagnosis of high voltage circuit breaker maintenance: The SMOTE-FRST-2T algorithm," *Eng. Appl. Artif. Intell.*, vol. 48, pp. 134–139, Feb. 2016.

[5] G. Amir and E. M. Masoud, "Investigating the performance of an order imbalance based trading strategy in a high-frequency trading," *Ind. Eng. Manage. Syst.*, vol. 19, no. 1, pp. 174–183, Mar. 2020.

[6] S. Lee, B. Koo, and K.-H. Jung, "Comparative study of dimension reduction methods for highly imbalanced overlapping churn data," *Ind. Eng. Manage. Syst.*, vol. 13, no. 4, pp. 454–462, Dec. 2014.

[7] 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.

[8] G. H. Nguyen, A. Bouzerdoum, and S. L. Phung, "Learning pattern classification tasks with imbalanced data sets," in *Pattern Recognition*. Vukovar, Croatia: InTech, 2009, pp. 193–208.

[9] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 7–19, Jun. 2004, doi: 10.1145/1007730.1007734.

[10] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, "Cost-sensitive learning," in *Learning From Imbalanced Data Sets*. Cham, Switzerland: Springer, 2018, pp. 63–78.

[11] Y.-J. Cho, H.-S. Lee, and C.-H. Jun., "Optimization of decision tree for classification using a particle swarm," *Ind. Eng. Manage. Syst.*, vol. 10, no. 4, pp. 272–278, 2011.

[12] J.-S. Lee and D. Zhu, "When costs are unequal and unknown: A subtree grafting approach for unbalanced data classification," *Decis. Sci.*, vol. 42, no. 4, pp. 803–829, Nov. 2011.

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

[14] J.-S. Lee, "AUC4.5: AUC-based C4.5 decision tree algorithm for imbalanced data classification," *IEEE Access*, vol. 7, pp. 106034–106042, 2019.

[15] 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.

[16] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Germany: Springer, 2005, pp. 878–887.

[17] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intelligence)*, Jun. 2008, pp. 1322–1328.
[18] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Berlin, Germany: Springer, 2009, pp. 475–482.
[19] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.
[20] Y.-P. Zhang, L.-N. Zhang, and Y.-C. Wang, "Cluster-based majority under-sampling approaches for class imbalance learning," in *Proc. 2nd IEEE Int. Conf. Inf. Financial Eng.*, Sep. 2010, pp. 400–404.
[21] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based under-sampling in class-imbalanced data," *Inf. Sci.*, vols. 409–410, pp. 17–26, Oct. 2017.
[22] V. García, J. S. Sánchez, A. I. Marqués, R. Florencia, and G. Rivera, "Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data," *Expert Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113026.
[23] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Prog. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, 2016.
[24] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. Cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics," *Expert Syst. Appl.*, vol. 39, no. 7, pp. 6585–6608, 2012.
[25] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 17, no. 1, 2001, pp. 973–978.
[26] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Proc. 6th Int. Conf. Data Mining (ICDM)*, Dec. 2006, pp. 970–974.
[27] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *Proc. Workshop Learn. Imbalanced Data Sets II (ICML)*, vol. 2, 2003, pp. 1–2
[28] N. V. Chawla, N. Japkowicz, and A. Kolcz, "Editorial: Special issue on learning from imbalanced data sets," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.
[29] D. J. Hand and V. Vinciotti, "Choosing K for two-class nearest neighbour classifiers with unbalanced classes," *Pattern Recognit. Lett.*, vol. 24, nos. 9–10, pp. 1555–1562, Jun. 2003.
[30] Z. Qin, A. T. Wang, C. Zhang, and S. Zhang, "Cost-sensitive classification with k-nearest neighbors," in *Knowledge Science, Engineering and Management*, M. Wang, Ed. Berlin, Germany: Springer, 2013, pp. 112–131.
[31] C. Qiu, L. Jiang, and C. Li, "Randomly selected decision tree for test-cost sensitive learning," *Appl. Soft Comput.*, vol. 53, pp. 27–33, Apr. 2017.
[32] S. Datta and S. Das, "Near-Bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs," *Neural Netw.*, vol. 70, pp. 39–52, Oct. 2015.
[33] S. Katsumata and A. Takeda, "Robust cost sensitive support vector machine," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, vol. 38. G. Lebanon and S. V. N. Vishwanathan, Eds. San Diego, CA, USA: PMLR, May 2015, pp. 434–443.
[34] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.
[35] B. Krawczyk and M. Woźniak, "Cost-sensitive neural network with ROC-based moving threshold for imbalanced classification," in *Intelligent Data Engineering and Automated Learning*, K. Jackowski, R. Burduk, K. Walkowiak, M. Wozniak, and H. Yin, Eds. Cham, Switzerland: Springer, 2015, pp. 45–52.
[36] J. A. Sáez, B. Krawczyk, and M. Wozniak, "Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets," *Pattern Recognit.*, vol. 57, pp. 164–178, Sep. 2016.
[37] A. H. Cannon and L. J. Cowen, "Approximation algorithms for the class cover problem," *Ann. Math. Artif. Intell.*, vol. 40, pp. 215–223, Mar. 2004.
[38] D. Marchette, "Class cover catch digraphs," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 2, pp. 171–177, 2010.
[39] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *Ann. Appl. Statist.*, vol. 5, no. 4, pp. 2403–2424, 2011, doi: 10.1214/11-AOAS495.
[40] R. Younsi and A. Bagnall, "An efficient randomised sphere cover classifier," *Int. J. Data Mining, Model. Manage.*, vol. 4, no. 2, pp. 156–171, 2012.
[41] R. Younsi and A. Bagnall, "Ensembles of random sphere cover classifiers," *Pattern Recognit.*, vol. 49, pp. 213–225, Jan. 2016.
[42] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[43] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml
[44] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, 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. 1–33, Jun. 2011.
[45] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.
[46] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, New York, NY, USA, 1992, pp. 144–152, doi: 10.1145/130385.130401.
[47] T. Pan, J. Zhao, W. Wu, and J. Yang, "Learning imbalanced datasets based on SMOTE and Gaussian distribution," *Inf. Sci.*, vol. 512, pp. 1214–1233, Feb. 2020.
[48] J. Zhang, T. Wang, W. W. Y. Ng, S. Zhang, and C. D. Nugent, "Undersampling near decision boundary for imbalance problems," in *Proc. Int. Conf. Mach. Learn. Cybern. (ICMLC)*, Jul. 2019, pp. 1–8.
[49] B. Mirzaei, B. Nikpour, and H. Nezamabadi-Pour, "CDBH: A clustering and density-based hybrid approach for imbalanced data classification," *Expert Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 114035.
[50] B. Derrick and P. White, "Comparing two samples from an individual Likert question," *Int. J. Math. Statist.*, vol. 18, no. 3, pp. 1–13, 2017.

YEONTARK PARK received the B.S. degree from Seoul National University and the M.S. degree in industrial engineering from Sungkyunkwan University, Suwon, South Korea, where he is currently pursuing the Ph.D. degree in industrial engineering. His research interests include design of learning algorithms for the class imbalance problem and their application to real domains.

JONG-SEOK LEE received the Ph.D. degree in industrial engineering from Iowa State University, USA. He has worked with the Research and Development Group, SAS Institute, for statistical software development. He is currently an Associate Professor with the Department of Industrial Engineering, Sungkyunkwan University, where he is leading the Info Science Laboratory as a POSCO Fellow Professor. His research papers have been published in the *INFORMS Journal on Computing*, *Information Sciences*, *Decision Sciences*, and other professional journals. His research interests include machine learning, data mining, and their application to service and manufacturing industries.