# Imputation-Based Ensemble Techniques for Class Imbalance Learning

Roozbeh Razavi-Far, *Senior Member, IEEE*, Maryam Farajzadeh-Zanajni, *Student Member, IEEE*, Boyu Wang, Mehrdad Saif, *Senior Member, IEEE*, and Shiladitya Chakrabarti

**Abstract**—Correct classification of rare samples is a vital data mining task and of paramount importance in many research domains. This article mainly focuses on the development of the novel class-imbalance learning techniques, which make use of oversampling methods integrated with bagging and boosting ensembles. Two novel oversampling strategies based on the single and the multiple imputation methods are proposed. The proposed techniques aim to create useful synthetic minority class samples, similar to the original minority class samples, by estimation of missing values that are already induced in the minority class samples. The re-balanced datasets are then used to train base-learners of the ensemble algorithms. In addition, the proposed techniques are compared with the commonly used class imbalance learning methods in terms of three performance metrics including AUC, F-measure, and G-mean over several synthetic binary class datasets. The empirical results show that the proposed multiple imputation-based oversampling combined with bagging significantly outperforms other competitors.

**Index Terms**—Class imbalance learning, oversampling, ensembles learning, missing data imputation

## 1 Introduction

In imbalanced datasets, the distribution of the samples between distinct classes are not equal. In the class imbalanced (CI) condition, the standard classifiers tend to favor majority class samples, and, consequently, assign incorrect labels to the minority class samples. Class imbalanced learning (CIL) is a challenge for the standard classifiers, which mainly deal with almost equal number of samples from each class. Generally, the majority class is labeled as the "negative" class and represents the natural state. The minority class, which is often labeled as "positive", has less number of samples, and it is usually the class of interest. For example, they represent data samples correspond to the cancerous people in medical records, fraud records of a bank or a faulty condition in an aircraft operational condition.

In real-world applications such as medical diagnosis, credit card fraud detection, fault diagnosis of industrial systems or face recognition [1], training a classifier under the CI condition is extremely important since misclassifiying the minority class samples may lead to severe problems. For instance, consider a situation where a person is affected by a dangerous disease and classified as healthy, or a faulty operational condition of a safety-critical system is classified as a normal state. CI is an integral part of the real world problems and enormously common in practice [2], [3], [4], [5], [6]. Various solutions have been proposed in the literature to deal with the CI problem [3], [4], [7], [8]. The provided solutions can be generally categorized into three approaches: data-level, algorithm-level and cost-sensitive approaches [3], [4]. Data-level approaches mainly focus on the pre-processing step, more specifically on over/under sampling methods [8], [9] to modify the distribution of samples and, therefore, provide equal number of samples for each class. Algorithm-level approaches concentrate on modifying learning algorithms in order to shift its bias in a way that it tends to favor the minority class samples during the training session. Various learning algorithms are modified in order to address CI distributions [10]. Cost-sensitive learning techniques taking into account unequal misclassification costs for the majority and minority classes, so that the significance of the minority (majority) class is enhanced (diminished) during learning phase. Tuning the misclassification cost in cost-sensitive algorithms [7] or weighting the samples in Weighted Extreme Learning Machine (WELM) [11] have been proposed to tackle the CI problem. Among cost-sensitive learning techniques, cost-sensitive ensembles such as AdaCost and Cost-Sensitive Boosting (CSB2) have been widely used for CIL [12], [13], [14].

Moreover, combination of data-level approaches and ensemble of classifiers can also be considered among the most popular approaches to solve the CI problem [4], [8], [15], [16], [17], [18], [19]. Galar et al. [4] reviewed different ensemble algorithms integrated with sampling techniques and showed their efficiency in handling class imbalance problems due to their versatility. These sampling-based ensemble algorithms make use of the potentials of both sampling methods and ensemble learning to enhance the classification accuracy. This combination attracted many researchers to propose some

- *R. Razavi-Far, M. Farajzadeh-Zanajni, M. Saif, and S. Chakrabarti are with the Department of Electrical and Computer Engineering, University of Windsor, Ontario N9B 3P4, Canada. E-mail: {roozbeh, msaif}@uwindsor.ca, {mfarajzadeh.z, shiladitya.chakarabarti}@gmail.com.*
- *B. Wang is with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA. E-mail: boyuwang@seas.upenn.edu.*

*Manuscript received 25 Feb. 2019; revised 8 Sept. 2019; accepted 29 Oct. 2019. Date of publication 6 Nov. 2019; date of current version 1 Apr. 2021. (Corresponding author: Roozbeh Razavi-Far.)*
*Recommended for acceptance by Q. He.*
*Digital Object Identifier no. 10.1109/TKDE.2019.2951556*

1041-4347 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

sampling-based ensemble techniques for CIL. The commonly used methods in this subcategory include RUSBoost [2], SMOTEBag [20] and SMOTEBoost [9].

The focus of this paper is mainly on oversampling methods that could be combined with ensemble schemes for CIL. The oversampling methods, unlike under-sampling ones, have no risk for loosing informative data samples [22]. Moreover, these methods can also be employed to balance the skewed-class data distribution in the pre-processing step, independent of the classification algorithms.

Here, in order to overcome the class imbalance problem, two novel oversampling methods are developed. These novel oversampling strategies make use of missing data imputation techniques to generate new synthetic minority class samples. These schemes initially induce some random missing over the minority class samples, and, then, estimate the missing scores by means of missing data imputation techniques resulting in the generation of new synthetic minority class samples. These novel imputation-based oversampling strategies are then devised in the ensemble schemes, i.e., boosting and bagging algorithms, resulting in various novel ensembles-based schemes for CIL. These novel imputation-based ensemble schemes are based on the bagging and boosting algorithms. The proposed techniques are evaluated and compared with twelve commonly used CIL techniques in classifying several synthetic binary class imbalanced datasets.

This paper is organized as follows: Section 2 describes the related works and existing CIL approaches and explains the sampling-based ensemble techniques. In Section 3, two novel oversampling strategies are proposed, that are based on single and multiple imputation of missing data. Moreover, a brief description of the missing data imputation techniques are provided in this section. Section 3 also presents the proposed CIL methods, which make use of the two novel imputation-based sampling strategies and integrated to bagging and boosting for class imbalance learning. Section 4 presents the experimental results. Several comparisons between the proposed CIL techniques and other widely used CIL methods are also provided. This section also presents the results of statistical significant tests. Section 5 summarizes and concludes the paper.

## 2 Related Works

As above mentioned, numerous research works have been proposed to address the learning under class imbalance conditions. This section first introduces the class imbalance problem, and, then, briefly reviews the relevant works for class imbalance learning. There has been a large number of practical problems that can be handled through supervised learning and fall under the class imbalanced condition. There has been an increasing interest for class imbalance learning as a result of the increased collection of the class imbalanced datasets in real-world applications [23], [24], [25].

An imbalanced binary class dataset contains more number of majority class samples than the minority class samples. Typical classifiers usually assume that class labels are equally distributed within the training subset. This assumption biases their classification performances towards the majority class samples [21]. In [25], the suitable performance metrics for CIL have been reviewed, the reasons for the performance drops in CI conditions have been studied, and various remedies have been introduced to rectify the undesirable impacts of class imbalance.

The primarily goal of this paper is on the development of efficient CIL techniques that can handle different class imbalance ratios. To this aim, first, existing solutions in three different categories of the CIL approaches (e.g., data-level, algorithm-level and cost-sensitive approaches) are briefly explained in this section. Moreover, the commonly used methods corresponding to each category are briefly presented. The CI problem generally can be solved either in the pre-processing step or during the learning phase of the classifiers [4]. Fig. 1 illustrates the taxonomy of the major techniques to overcome the class imbalance problem. The proposed techniques are also added to this chart, marked with '*' and explained in the next section.

**Algorithm 1.** SMOTE [9]

INPUTS: $\mathcal{S}^{min}$ is a subset of minority class samples, $\delta$ is the oversampling rate, $\psi$ is the # of nearest neighbours
DEFINITIONS:
$m^{min}$ is the $\#$ of samples of $\mathcal{S}^{min}$
**for** $i = 1, \ldots, m^{min}$ **do**
  **for** $j = 1, \ldots, \delta$ **do**
    1. Find a set $\psi$-nearest neighbors for a sample $x_i^{min}$

$$\mathcal{N}_i = \left\{ x_p \in \mathcal{S}^{min} | \, d_{ip} \leq d_{iq} \, \forall \, x_q \notin \mathcal{N}_i \right\}_{p=1}^{\psi}$$

    where $d_{ip}$ ($d_{iq}$) stands for the Euclidean distance between $x_i$ and $x_p$ ($x_i$ and $x_q$).
    2. Select a nearest neighbor of $x_i^{min}$, $x_j^{min} \in \mathcal{N}_i$
    3. Select a random number $\gamma$, where $0 \leq \gamma \leq 1$.
    4. Create a new synthetic sample

$$x_{i,j}^{new} = x_i^{min} + (x_j^{min} - x_i^{min}) \times \gamma$$

  **end**
**end**
5. Return $\delta m^{min}$ newly generated samples:

$$\mathcal{S}_{os}^{min} = \left\{ x_{i,j}^{new} | \, \forall i, j : \, i = 1, \ldots, m^{min}, \, j = 1, \ldots, \delta \right\}$$

### 2.1 Data-Level Approaches

Data-level approaches refer to those CIL methods that are independent from the classification algorithms. These methods try to reduce the effect of the skewed class distribution by addition of the minority class samples, e.g., random oversampling (ROS) or eliminating the available majority class samples, e.g., random undersampling (RUS). ROS and RUS are simple methods to produce a class balanced distribution. However, they are not 'smart', since they randomly choose their sampling subset, that may lead to over-fitting during the learning phase or loss of important information. There exist some informed sampling techniques such as the synthetic minority oversampling technique (SMOTE) [9]. SMOTE has been widely used to overcome the weakness of RUS and ROS.

SMOTE is a very common sampling technique and selected as representative of data-level approaches to balance the proportion of the data samples among different classes. SMOTE makes use of the nearest neighbors technique to

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

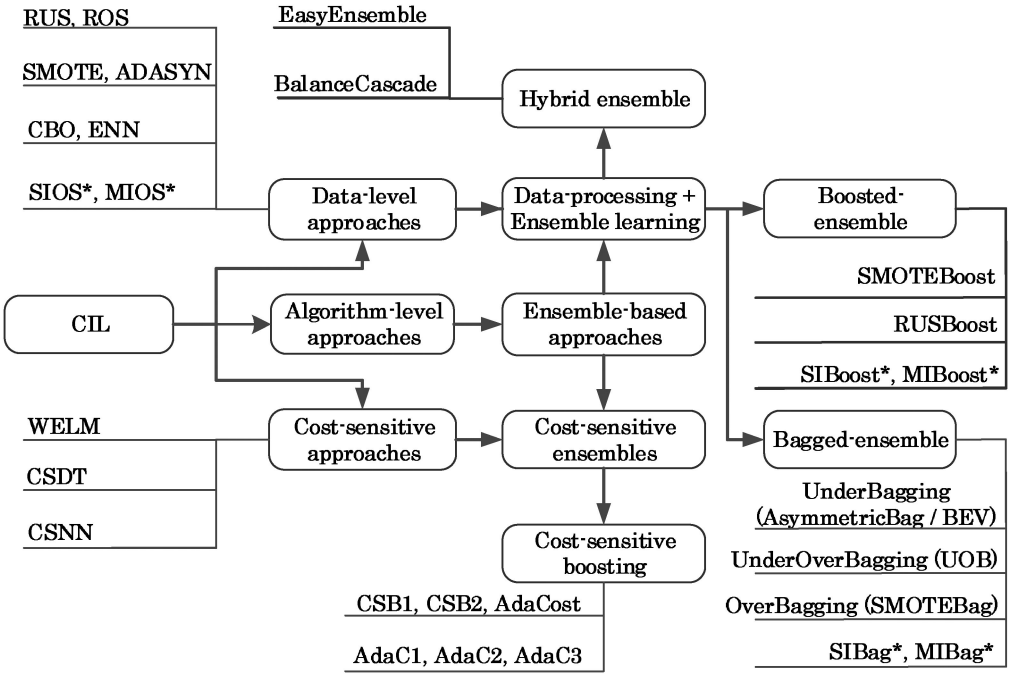



Fig. 1. The taxonomy of the major techniques to overcome the class imbalance problem. The proposed novel CIL techniques are marked with asterisks [4], [21].

create some synthetic samples. The number of generated minority class samples depends on the required oversampling ratio. Given minority class samples as $\mathcal{S}^{min}$ and an oversampling rate of $\delta$, SMOTE is presented in Algorithm 1 [3], [9].

Resampling techniques have been widely used for CIL. In [24], various oversampling technqiues are validated and compared for class imbalance learning. Various state-of-the-art data-level approaches have been proposed for class imbalanced learning including Edited Nearest Neighbor (ENN) [26], SMOTE and Tomek Links (SMOTE-TL) [27], and ADAptive SYNthetic algorithm (ADASYN) [28]. ENN classifies samples by kNN, and, then, discards those that are misclassified [26]. ENN aims to eliminate outliers and those samples overlapping among different classes. SMOTE-TL makes use of SMOTE to oversample minority class samples and, then, discards Tomek links [27]. ADASYN adaptively resamples the minority class samples that are more difficult to learn [28]. It randomly picks $\psi$ nearest neighbors around those minority class samples and generates new synthetic samples in the middle of minority class samples and their selected nearest neighbors. These methods are also used in this work for the sake of comparison.

### 2.2 Algorithm-Level Approaches

There exists another group of machine learning techniques, so called algorithm-level approaches, to alleviate the class imbalance problem [4], [10]. These techniques aim to adapt classification algorithms in order to bias the learning toward the minority class [29]. These approaches modify learning algorithms to reward the minority class and penalize the majority class during the training session. Various efforts have been made to modify kernel mechanisms in the learning process such as kernel-based support vector machines (KSVMs).

Extreme Learning Machines (ELM) have been widely used for many applications. Weighted ELM (WELM) has been lately proposed to tackle the CI problem [11]. WELM can also belong to the cost-sensitive approaches, since it uses a misclassification cost in a weighting scheme. Here, WELM is used as a commonly used algorithm-level approach, which also makes use of the cost information (see Fig. 1).

Ensemble-based approaches have also been extensively used for CIL [4]. As it is illustrated in Fig. 1, ensemble-based methods can be categorized into two major groups of (a) ensemble learning along with data processing and (b) cost-sensitive ensembles. Each of these groups including commonly used methods are presented in Fig. 1. Ensemble learning along with data sampling approaches are explained in the following and cost-sensitive ensembles are explained under cost-sensitive approaches. Ensemble learning along with data sampling approaches can be categorized into three subgroups: (1) Boosted ensemble techniques that embed data processing techniques into boosting algorithms such as SMOTEBoost [30], RUSBoost [2], Tomek Link and Boosting (TLboost) [21] and LexiBoost [31], (2) Bagged ensembles can be divided into OverBagging [20], UnderBagging [32], and UnderOverBagging [20] strategies. Bagged ensembles also include other algorithms, e.g., SMOTEBag [20], AsymmetricBag [33] and Bagging Ensemble Variation (BEV) [34], and (3) Hybrid ensembles that merge both boosting and bagging algorithms to form an ensemble of ensembles such as EasyEnsemble (EE) and BalancedCascade [35], [36]. Hybrid ensembles are computationally more expensive compared to the boosted and bagged ensembles. A critical review on this topic has been performed and efficient ensemble techniques for CIL are compared in [4]. Among various ensemble-based approaches for CIL (see Fig. 1), RUSBoost [2], SMOTEboost and SMOTEbag [20], [30] are the most common and robust techniques [4]. Moreover, RUSBoost is a simpler method

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

with less computational complexity. Therefore, RUSBoost and SMOTEBoost from the boosted ensembles, SMOTEBag and UnderOverbagging from the bagged ensembles, and EE from the hybrid ensembles are considered in this work.

#### 2.2.1 Boosted Ensembles

The two popular members of this category, RUSBoost and SMOTEBoost, are actually a combination of AdaBoost.M2 with RUS and SMOTE, respectively. AdaBoost.M2 algorithm adjusts distribution weights of the samples, by assigning a higher weight to the misclassified samples including the representative samples of the minority class and also adjusts voting weights of the individual base-classifiers. This scheme also helps to avoid the loss of potentially useful samples, which can be occurred in RUS. For the sake of brevity, only the pseudo-code of SMOTEBoost is presented in Algorithm 2 [2], [30]. However, SMOTE which is applied in the step 2 of this algorithm can be replaced by RUS to perform RUSBoost. The class imbalanced datset $\mathcal{S}$ is considered as $\mathcal{S} = \{(x_i, y_i)|x_i \in X,\ y_i \in Y = \{0,1\}\}_{i=1}^{m}$ and, $I(\varphi)$ is considered as the indicator function of event $\varphi$ for the rest of the paper.

**Algorithm 2.** SMOTEBoost [30]

INPUTS: $\mathcal{S}$ is a CI dataset, $h$ is the base-classifier, $T$ is the maximum # of iterations, $\psi$ is the number of nearest neighbors, $m'$ is the percentage of the total samples of the minority class
DEFINITIONS: $m$ is the # of samples
1. Initialize the weight of each sample $w_1(i) = \frac{1}{m}$
**for** $t = 1, \ldots, T$ **do**
2. Generating synthetic samples from minority class by means of SMOTE until $m'$ condition satisfies.
3. Construct a base-classifier $h_t$ based on the new distribution $w_t(i)$
4. Compute $\epsilon_t$:

$$\epsilon_t = \sum_{i=1}^{m} w_t(i) I(h_t(x_i) \neq y_i)$$

5. Update the weight distribution $w_t(i)$ for all samples $x_i, 1 \leq i \leq m$:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} \frac{1}{2(1-\epsilon_t)}, & h_t(x_i) = y_i \\ \frac{1}{2\epsilon_t}, & h_t(x_i) \neq y_i \end{cases}$$

**end**
6. Return the final hypothesis:

$$H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) I(h_t(x) = y)$$

#### 2.2.2 Bagged Ensembles

Generally, these methods make use of the bagging algorithm to create an ensemble model that is integrated with resampling techniques (i.e., under-sampling of the majority class samples or oversampling of minority class samples). SMOTEBag and UnderOverBagging (UOB) are under this category [20]. SMOTEbag enhances the diversity among the base-classifiers by changing the ratio of bootstrap replicates and generating various synthetic samples by SMOTE, over the bagging iterations. To construct these subsets, SMOTEbag resamples from the majority class samples with replacement at rate 100 percent. It uses a percentage value $\beta\%$ (ranging from 10 to 100) to control the number of newly generated synthetic minority class samples. It initially resamples some original minority class samples and, then, uses SMOTE to create new synthetic minority class samples. Similar to SMOTEbag, UOB also exploits an ensemble model that is based on bagging. UnderOverbagging reduces the number of majority class samples and increases the number of minority class samples to create a re-balanced subsets $\mathcal{S}_t$ that are then used during the learning phase of the base-classifiers of the ensemble. UOB process is presented in Algorithm 3 [20]. For the complete descriptions of the SMOTEbag and UOB, the reader can refer to [20].

**Algorithm 3.** UnderOverBagging [20]

INPUTS: $\mathcal{S}$ is a CI dataset, $T$ is the number of base-classifier
DEFINITIONS: $\mathcal{S}^{maj}$ ($\mathcal{S}^{min}$) is the subset of the majority (minority) class samples, $\mathcal{S}_r^{maj}$ ($\mathcal{S}_r^{min}$) is the resampled subset of the majority (minority) class samples, $\mathcal{SWR}(A, B)$ is the function of sampling with replacement to draw $B$ samples from the given set $A$.
**for** $t = 1, \ldots, T$ **do**
1. $\beta = \frac{t}{T}$
2. Resample from the majority class samples with replacement as follows:

$$\mathcal{S}_r^{maj} = \mathcal{SWR}(\mathcal{S}^{maj}, \beta \cdot m^{maj}).$$

3. Resample from original minority class samples with replacement at the percentage of $\theta = \frac{m^{maj}}{m^{min}} \cdot \beta$

$$\mathcal{S}_r^{min} = \mathcal{SWR}(\mathcal{S}^{min}, \theta \cdot m^{min}).$$

4. Construct $\mathcal{S}_t$ subset by means of generated samples as follows: $\mathcal{S}_t = \left\{\mathcal{S}_r^{maj} \cup \mathcal{S}_r^{min}\right\}$
5. Train a base-classifier $h_t \rightarrow Y$ using $\mathcal{S}_t$
**end**
6. Find the composite hypothesis:

$$H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y).$$

#### 2.2.3 Hybrid Ensembles

These ensembles unlike the bagged and boosted ensembles perform a dual ensemble learning. This has been done by merging bagging and boosting along with a pre-processing approach. Hybrid ensembles such as EasyEnsemble (EE) and BalanceCascade (BC) [35] make use of the bagged scheme to form the ensemble, however, instead of an individual base learner, they train AdaBoost inside each bag, and, thus, forming an ensemble of ensembles. More specifically, EE constructs a bag of balanced subsets by resorting to random oversampling, and, then, trains ensemble of base learners, whose outcomes are combined by means of the majority of votes [35].

### 2.3 Cost-Sensitive Level Approaches

These approaches consider the costs associated with the different classes into account [37]. The learning algorithm then assigns a high misclassification cost to the minority class samples reinforcing their importance during the training session; on the contrary, the cost corresponding to the majority class samples is decreased. In [38], cost-sensitive

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

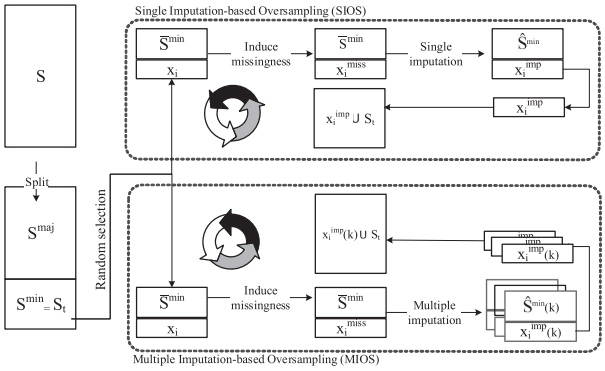



Fig. 2. The general diagram of the single and multiple imputation-based oversampling techniques.

neural networks are trained for CIL. Cost-sensitive level approaches are mainly categorized into two groups including direct methods and meta-learning approaches [21], [25].

Cost-sensitive ensembles include various cost-sensitive boosting techniques such as AdaCost, AdaC1, AdaC2, and AdaC3 [12], [13], [14]. In boosted ensembles, the classifiers generally focus on learning the samples that are misclassified by previous classifiers or those samples that are more difficult to learn than those samples that are already correctly classified in previous iterations. The importance of classifying a sample can be determined by assigning a weight value, which is updated for each sample once the learning process at that iteration is done. AdaC2 is a boosting algorithm that also considers a cost value associated with each sample, which represents the importance of the correct classification of that sample. In other words, AdaC2 exploits the concept of cost to assign higher classification costs for the minority class samples $C^{min}$ in comparison with the cost related to the majority class samples $C^{maj}$ to highlight their importance during CIL. Thus, AdaC2 updates the weight calculation formula in the boosting algorithm by introducing the cost values [12]. The other cost-sensitive boosting algorithm, considered in this study, is CSB2. Unlike AdaC2, CSB2 does not modify the weight update parameter in the boosting algorithm. However, it introduces the cost sensitivity in the process of updating the weight and distribution of the samples [13].

In this work, twelve CIL techniques from different categories along with the C4.5 base learner (BaseLine) are selected and applied to all CI datasets. These include BaseLine, SMOTE, ADASYN, SMOTE-TL, ENN, WELM, EE, SMOTEBoost, RUSBoost, SMOTEBag, UnderOverBag, AdaC2, and CSB2. The attained results by these techniques are compared with the proposed CIL approaches in Section 4 for performing a fair comparison.

## 3 The Proposed CIL Approaches

This section proposes novel single and multiple imputation-based over-sampling methods that are combined with the bagging and boosting algorithms. The proposed oversampling methods are divided into two categories w.r.t. the type of imputation methods employed in the core of these algorithms. These categories include (1) single imputation-based oversampling (SIOS) techniques and (2) multiple imputation-based oversampling (MIOS) techniques. Moreover, four novel types of ensembles are proposed by merging SIOS and MIOS with boosting and bagging to make SIBoost, SIBag, MIBoost and MIBag, respectively (see Fig. 1). These novel algorithms are introduced in this section.

The proposed methods not only exploit the benefits of the oversampling techniques, but can also be merged with the ensemble algorithms in order to enhance the classification perfromance under the class imbalance condition. In other words, imputation-based over-sampling techniques try to generate synthetic minority class samples, similar to the original minority class samples, and provide suitable re-balanced subsets for training the base-classifiers. These classifiers make use of the original minority class samples along with the newly generated samples during the training phase. Moreover, this approach could benefit the process of classifying the imbalance data samples in presence of missing values in the minority class samples. This is a worthwhile direction for the future research.

In imputation-based class imbalance learning, the basic idea is to create an incomplete subset by intentionally inducing missing values into the original minority class samples; and, then, make use of both original and incomplete minority class samples to re-estimate and impute the induced missing values in order to create new minority class samples. These newly produced samples change the distribution of the class labels, aim to re-balance the dataset, and will be used during the learning process of the ensemble.

In addition, it is also noticeable that the proposed oversampling technique has the potential to be applied solely on an imbalanced dataset during the preprocessing step in order to re-balance the dataset, so that any standard classifier (e.g., Naive Bayes or decision tree) could properly classify the attained re-balanced dataset.

### 3.1 Imputation-Based Oversampling

The proposed oversampling methods attempt to address the class imbalanced learning issue at the most fundamental data level by providing a class re-balanced data distribution. The imputation-based oversampling actually alters the minority class sample size by producing new samples as a result of an estimation procedure. The estimation function is based on either single imputation (SI) or multiple imputation (MI) of the missing values that are randomly induced on the existing minority class samples. Fig. 2 illustrates the general overview of the single and multiple imputation-based oversampling techniques that are SIOS (top panel) and MIOS (bottom panel), respectively. These techniques are applied to all CI datasets for oversampling the minority class samples.

#### 3.1.1 Single Imputation-Based Oversampling (SIOS)

Given a binary class imbalance dataset $\mathcal{S}$, one can split it to a subset of majority class samples and a subset of minority class samples. Having the subset of the minority class samples in hand, SIOS technique iteratively estimates new samples by means of the original samples in the feature space. SIOS pseudo-code is presented in Algorithm 4. In Algorithm 4, the number of samples that needs to be generated by SIOS is initially computed and considered as the

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

target number $\tau^{min}$. Then, the existing minority class subset and its number of samples are saved as $\mathcal{S}_t$ and $m_t^{min}$, respectively. If $m_t^{min}$ is less than the target number $\tau^{min}$, then SIOS iteratively generates new samples one by one until a balance distribution is achieved. In step 3, one sample from the minority class subset is randomly selected. In other words, the minority class subset is divided into two parts; a minority class sample $x_i$ and the rest $\bar{\mathcal{S}}^{min}$. After that, missing values are randomly induced over 20-50 percent of the features of $x_i$ forming $x_i^{mis}$. The incomplete sample $x_i^{mis}$ along with the rest of the minority class samples $\bar{\mathcal{S}}^{min}$ are fed to the single imputation function, *SI*. This function, which is actually the core of SIOS technique, tries to estimate the missing values of $x_i^{mis}$ by means of the information from the rest of complete minority class samples $\bar{\mathcal{S}}^{min}$. SIOS procedure is presented in the top panel of Fig. 2.

**Algorithm 4.** SIOS

INPUTS: $\mathcal{S}^{min}$ is the subset of minority class samples.
$\rho$ is the oversampling rate.
DEFINITIONS: $m^{min}$ is the # of minority class samples.
$(\bar{A}, a) = split(A, 1)$ randomly selects a sample from $A$ and places it in $a$, and the rest of samples are saved in $\bar{A}$.
$a^{mis} = missing(a)$ randomly induces missing values on the sample $a$ and returns an incomplete sample $a^{mis}$.
$\hat{A} = SI(a, A)$ the function receives an incomplete sample $a$ and the subset of original minority class samples $A$, where $a \notin A$. It calls a single imputation (EM, BPCA or LWLA), estimates the missing values and returns the imputed sample along with the original subset.
1. Initialize the target rate $\tau^{min} = \rho.m^{min}$
2. Copy the minority class subset $\mathcal{S}_t = \mathcal{S}^{min}$, and its # of observations $m_t^{min} = m^{min}$

**while** $m_t^{min} < \tau^{min}$ **do**
3. Randomly select a minority class sample $x_i$, $(\bar{\mathcal{S}}^{min}, x_i) = split(\mathcal{S}^{min}, 1)$
4. Induce random missingness $x_i^{mis} = missing(x_i)$
5. Call *SI* function to estimate $x_i^{mis}$ by knowing $\bar{\mathcal{S}}^{min}$ as follows:

$$\hat{\mathcal{S}}^{min} = SI(x_i^{mis}, \bar{\mathcal{S}}^{min})$$

6. Select $x_i^{imp}$ from $\hat{\mathcal{S}}^{min}$
7. Append the generated sample $\mathcal{S}_t = \mathcal{S}_t \cup x_i^{imp}$
8. Update the # of samples $m_t^{min} = m_t^{min} + 1$

**end**
9. Return $\mathcal{S}_t$

In this study, three different imputation methods named Expectation Maximization (EM) [39], Bayesian Principle Component Analysis (BPCA) [40] and Local Weighted Linear Approximation (LWLA) [41] are considered for single imputation *SI*. Although the goal of these *SI* techniques is the same, their strategy in imputing the $x_i^{mis}$ are completely different and, thus, lead to different results $x_i^{imp}$. EM is a widely used statistical approach to impute missing values based on key parameters (i.e., mean and covariance) of the data model. BPCA is another commonly used single imputation strategy, which makes use of correlations among the data. BPCA benefits from the global covariance structure of the data, however, it does not consider the strong local similarity structure of the data. Therefore, a local imputation strategy, called LWLA, is also used in this work to capture the local structure of the data. LWLA adaptively tailors each target missing feature to a set of candidate features by resorting to a weighting function, which focuses on the nearest points in the approximation. LWLA linearly estimates the missing values of the target features without a need for pre-selection of the candidate features and parameter selection. LWLA can efficiently integrate the local structure with the global distribution for all selected features around the target feature. This makes LWLA an optimal scheme to simultaneously extract global and local information by means of the Taylor series approximation theory. For detailed explanations about missing data imputation by means of EM, BPCA and LWLA, the reader can refer to the following references [39], [40], [41].

Upon estimation of the missing scores in $x_i^{mis}$, the newly imputed sample $x_i^{imp}$ is introduced. In step 7, the newly generated sample $x_i^{imp}$ is merged with the minority class subset, and, then, the number of minority class samples is increased. The process of generating synthetic samples is continued until a balanced dataset is achieved. MIOS procedure is presented in the bottom panel of Fig. 2.

**Algorithm 5.** MIOS

INPUTS: $\mathcal{S}^{min}$, $\rho$, $k$.
DEFINITIONS:
$\hat{A}(\kappa) = MI(a, A, k)$ the function receives an incomplete sample $a$, the subset of original minority class samples $A$ ($a \notin A$), and the number of imputations $k$. It calls a multiple imputation function (MCMC or MICE), estimates the missing values multiple times and returns $k$ datasets each of which contains $\kappa$th imputed sample along with the original subset ($1 < \kappa \leq k$).
1. Initialize the target rate $\tau^{min} = \rho.m^{min}$
2. Copy the minority class subset $\mathcal{S}_t = \mathcal{S}^{min}$, and the # of samples $m_t^{min} = m^{min}$

**while** $m_t^{min} < \tau^{min}$ **do**
3. Randomly select a minority class sample $x_i(\bar{\mathcal{S}}^{min}, x_i) = split(\mathcal{S}^{min}, 1)$
4. Induce random missingness $x_i^{mis} = missing(x_i)$
5. Call $MI$ function to estimate $x_i^{mis}$, $k$ times, using $\bar{\mathcal{S}}^{min}$ as follows:

$$\hat{\mathcal{S}}^{min}(\kappa) = MI(x_i^{mis}, \bar{\mathcal{S}}^{min}, k)$$

6. Select $k$ imputed samples from the completed datasets $\hat{\mathcal{S}}^{min}(\kappa)$, then, save them into $S_i^{imp}$
7. Append the generated samples $\mathcal{S}_t = \mathcal{S}_t \cup S_i^{imp}$
8. Update the # of samples $m_t^{min} = m_t^{min} + k$

**end**
9. Return $\mathcal{S}_t$

### 3.1.2 Multiple Imputation-Based Oversampling (MIOS)

In this work, multiple imputation-based oversampling (MIOS) method is also proposed, which aims to generate synthetic minority class samples. The MIOS pseudo-code is provided in Algorithm 5. Similar to SIOS, MIOS receives a minority class subset and an oversampling rate as inputs. This algorithm needs to set another input parameter as the number of imputation $k$. The value of $k$ ($k$ is set to 5 by default) defines the number of estimated samples that needs to be generated based on each incomplete sample. In other words, each missing value induced in $x_i^{mis}$, can be filled

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

with $k$ different imputed values. So, in step 5, the *MI* function is called to generate $k$ different samples based on $x_i^{mis}$ and the existing complete minority class samples. MI generates multiple estimations, creates multiple complete datasets for each missing value. The outcomes of *MI* are saved into $\hat{\mathcal{S}}^{min}(\kappa)$, which refers to the $\kappa$th completed dataset, where $1 \leq \kappa \leq k$. Then, the imputed samples in these completed datasets are selected and saved in a set $S_i^{imp}$, and added to the minority class samples in the next step $\mathcal{S}_t = \mathcal{S}_t \cup S_i^{imp}$.

The major distinction of SIOS and MIOS is the oversampling method used in the core of these techniques referred as *SI* and *MI*, respectively. In this work, two state-of-the-art multiple imputation techniques; Multivariate Imputation by Chained Equations (MICE) [42] and Markov Chain Monte Carlo (MCMC) [43] are used for *MI*, that aim to estimate multiple values for each missing score. MICE specifies a conditional distribution for each missing feature and assumes the joint distribution; it then makes use of the Gibbs sampler to iteratively sample from the conditional distributions [42]. MCMC missing data imputation creates pseudo-random samples from multidimensional probability distributions via Markov chains [43].

### 3.2 Imputation-Based Ensembles

In this step, two novel types of ensemble are proposed that are based on integration of SIOS with ensemble learning, to form SIBoost and SIBag that can be compared with the famous methods such as SMOTEBoost and SMOTEBag. Similarly, the other proposed oversampling method, called MIOS, is also integrated with boosting and bagging ensembles in order to create new ensemble techniques, MIBoost and MIBag. The detailed explanations of these techniques are provided in this section.

#### 3.2.1 Imputation-Based Boosting

In the boosting approach (see Algorithm 6), an initial weight is assigned to each sample based on the distribution of the samples in each class. Then, *SIOS* is applied on the minority class subset to generate new minority class samples. The *SIOS* procedure is explained in Algorithm 4. Increasing the number of minority class samples alters the class distribution and introduces a new training set that is used to train the base-classifier $h_t$ of the ensembles. After that the error $\epsilon_t$, which is affected by the number of misclassified samples is calculated and the new weight distribution $w_{t+1}$ is computed and assigned to each sample. The updated weights $w_{t+1}$ assist in classifying those samples that are difficult-to-learn, i.e., misclassified by $h_t$ in the previous iteration. SIBoost creates various minority class samples by means of *SIOS* and iteratively trains a set of base learners $h_t$, $t = 1, \ldots, T$ to form an ensemble, and combines their outcomes to attain the final decision $H(x)$.

In these experiments, SIBoost makes use of three single imputation methods (EM, BPCA and LWLA) in the core of the algorithm, resulting in three variants of imputation-based boosting named SIBoost-EM, SIBoost-BPCA and SIBoost-LWLA.

MIBoost is constructed the same as SIBoost except the step 2 of the Algorithm 6. In the step 2, MIBoost makes use of the multiple imputation-based oversampling method, called *MIOS*. Moreover, by default, the number of imputation $k$ is set to 5 in *MIOS*. For the sake of simplicity, the MIBoost algorithm is considered as an extension of the Algorithm 6 and its difference with SIBoost is highlighted at the end of the algorithm.

**Algorithm 6.** SIBoost

INPUTS: $\mathcal{S}$ is a CI dataset, $h$ is the base-classifier, $T$ is the maximum # of iterations, $m'$ is the percentage of total minority class samples.

DEFINITIONS: $m$ is the # of samples.

1. Initialize the weight of each sample $w_1(i) = \frac{1}{m}$

**for** $t = 1, \ldots, T$ **do**

2. Generate new synthetic minority class samples using SIOS until $m'$ condition satisfies.

3. Construct a base-classifier $h_t$ based on the new distribution $w_t(i)$.

4. Compute $\epsilon_t$ as follows:

$$\epsilon_t = \sum_{i=1}^{m} w_t(i) I(h_t(x_i) \neq y_i)$$

5. Update the weight distribution $w_t(i)$ for all samples $x_i, 1 \leq i \leq m$:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} \frac{1}{2(1-\epsilon_t)}, & h_t(x_i) = y_i \\ \frac{1}{2\epsilon_t}, & h_t(x_i) \neq y_i \end{cases}$$

**end**

6. Return the final hypothesis:

$$H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) I(h_t(x) = y)$$

**MIBoost** calls *MIOS* in step 2, instead of *SIOS*.

Similarly, MIBoost makes use of two multiple imputation methods (MICE and MCMC) in the core of the algorithm, resulting in two variants of imputation-based boosting named MIBoost-MICE and MIBoost-MCMC.

#### 3.2.2 Imputation-Based Bagging

The pseudo-code of SIBag is shown in Algorithm 7. At each iteration, SIBag performs a sampling with replacement $\mathcal{SWR}$ on both subsets of majority class samples $\mathcal{S}^{maj}$ and minority class samples $\mathcal{S}^{min}$. SIBag draws some samples from these subsets and forms $\mathcal{S}_r^{maj}$ and $\mathcal{S}_r^{min}$ subsets, respectively. Upon calculation of the oversampling rate $\rho$, *SIOS* is used for generating new minority class samples by means of $\mathcal{S}_r^{min}$ until the $\rho$ condition is satisfied. The result of the oversampling $\mathcal{S}_{os}^{min}$, is then merged with the original subset of the minority class samples $\mathcal{S}_r^{min}$ and the majority class samples $\mathcal{S}_r^{maj}$. The combination of these samples $\mathcal{S}_t = \left\{\mathcal{S}_r^{maj} \cup \mathcal{S}_{os}^{min}\right\}$ is then iteratively used to construct the base-classifiers $h_t$ during the training session. Once the base-classifiers are trained based on various re-balanced subsets, the final hypothesis $H(x)$ can then be obtained by combination of the outcomes of these constructed base-classifiers.

MIBag is constructed the same as SIBagt except the step 5 of the Algorithm 7. In the step 5, MIBag makes use of the multiple imputation-based oversampling method, called *MIOS*, in which $k$ is set to 5 by default. For the sake of simplicity, the MIBag algorithm is considered as an extension of

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Specifications of the Datasets Used in the Experiments

| Names | # of Samples | # of Features | IR | Names | # of Samples | # of Features | IR | Names | # of Samples | # of Features | IR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| bupa | 345 | 6 | 1.38 | ecoli1 | 336 | 7 | 3.36 | yeast1vs7 | 459 | 7 | 14.3 |
| flowmeterA | 87 | 37 | 1.48 | new-thyroid1 | 215 | 5 | 5.14 | glass4 | 214 | 9 | 15.47 |
| ecoli0vs1 | 220 | 7 | 1.86 | new-thyroid2 | 215 | 5 | 5.14 | ecoli4 | 336 | 7 | 15.8 |
| wisconsin | 683 | 9 | 1.86 | ecoli2 | 336 | 7 | 5.46 | pageblocks13vs4 | 472 | 10 | 15.86 |
| pima | 768 | 8 | 1.87 | segment0 | 2308 | 19 | 6.02 | glass016vs5 | 184 | 9 | 19.44 |
| iris0 | 150 | 4 | 2 | glass6 | 214 | 9 | 6.38 | shuttlec2vsc4 | 129 | 9 | 20.5 |
| glass0 | 214 | 9 | 2.06 | yeast3 | 1484 | 8 | 8.1 | yeast1458vs7 | 693 | 8 | 22.1 |
| yeast1 | 1484 | 8 | 2.46 | ecoli3 | 336 | 7 | 8.6 | glass5 | 214 | 9 | 22.78 |
| haberman | 306 | 3 | 2.78 | pageblocks0 | 5472 | 10 | 8.79 | yeast2vs8 | 482 | 8 | 23.1 |
| vehicle2 | 846 | 18 | 2.88 | yeast2vs4 | 514 | 8 | 9.08 | yeast4 | 1484 | 8 | 28.1 |
| vehicle1 | 846 | 18 | 2.9 | yeast05679vs4 | 528 | 8 | 9.35 | yeast1289vs7 | 947 | 8 | 30.57 |
| vehicle3 | 846 | 18 | 2.99 | vowel0 | 988 | 13 | 9.98 | yeast5 | 1484 | 8 | 32.73 |
| glass0123vs456 | 214 | 9 | 3.2 | glass016vs2 | 192 | 9 | 10.29 | ecoli0137vs26 | 281 | 7 | 39.14 |
| vehicle0 | 848 | 18 | 3.25 | shuttlec0vsc4 | 1829 | 9 | 13.87 | yeast6 | 1484 | 8 | 41.4 |

the Algorithm 7 and its difference with SIBag is highlighted at the end of the algorithm.

In these experiments, SIBag (MIBag) makes use of different single (multiple) imputation methods including EM, BPCA and LWLA (MICE and MCMC) in the core of the algorithm, resulting in various imputation-based bagging: SIBag-EM, SIBag-BPCA and SIBag-LWLA (MIBag-MICE and MIBag-MCMC).

**Algorithm 7.** SIBag

INPUTS: $\mathcal{S}, T$

**for** $t = 1, \ldots, T$ **do**

1. $\beta = \frac{t}{T}$
2. Resample from the majority class samples with replacement at 100 percent as follows:

$$\mathcal{S}_r^{maj} = \mathcal{SWR}(\mathcal{S}^{maj}, m^{maj})$$

3. Resample from original minority class samples with replacement at the percentage of $\theta = \frac{m^{maj}}{m^{min}} \cdot \beta$

$$\mathcal{S}_r^{min} = \mathcal{SWR}(\mathcal{S}^{min}, \theta \cdot m^{min})$$

4. Set $\rho = \frac{m^{maj}}{m^{min}} \cdot (1 - \beta)$
5. Create new synthetic minority class samples by means of SIOS as follows:

$$\mathcal{S}_{os}^{min} = SIOS(\mathcal{S}_r^{min}, \rho)$$

6. Construct $\mathcal{S}_t$ by merging subsets:

$$\mathcal{S}_t = \{\mathcal{S}_r^{maj} \cup \mathcal{S}_{os}^{min}\} \quad (1)$$

7. Train a base-classifier $h_t \rightarrow Y$ using $\mathcal{S}_t$

**end**

8. Find the composite hypothesis:

$$H(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y)$$

**MIBag** calls *MIOS* in step 5, where $k$ is set to 5:

$$\mathcal{S}_{os}^{min} = MIOS(\mathcal{S}_r^{min}, \rho, k)$$

## 4 Experimental Results

In order to assess the performance of the proposed novel algorithms, different binary class datasets with different class imbalance ratios and different specifications are used. Hence, the proposed algorithms including SIBoost, SIBag, MIBoost and MIBag strategies along with the commonly used CIL techniques are applied on several datsets. The classification results in terms of F-Measure (F-m), Geometric Mean (G-mean) and Area Under the Curve (AUC) are obtained, and, then, these results are evaluated and compared with each other to make sure if the proposed methods could improve the performance. The classifiers employed in the bagged and the boosted ensembles are C4.5. In addition, decision tree is also considered as a base classifier to learn the data samples that are already re-balanced by the commonly used and the proposed novel oversampling techniques. The base learner C4.5 (BaseLine) has two hyperparameters including the minimum number of samples per leaf and the pruning confidence. These parameters are tuned using random search within the cross-validation scheme. In this work, the nested variation of distribution optimally balanced stratified cross-validation (DOB-SCV) [21], [44], [45] is used to evaluate the model performance and to tune the C4.5 hyperparameters. DOB-SCV is a more suitable validation strategy for imbalanced datasets, which can minimize covariate shift. In all these experiments, the number of folds is set to five.

### 4.1 Datasets

This study contains 42 different imbalanced datsets in which the class imbalance ratio vary from 1.38 to 41.4. This ratio shows the proportion of the majority class samples to the minority class samples. These datasets are available online in the KEEL and UCI repositories. They are widely used by many researchers to evaluate CIL methods. The specification of these datasets are provided in Table 1. Table 1 shows the name, number of samples, number of features and the imbalance ratio (IR) value for each dataset.

### 4.2 Performance Metrics

Various performance metrics can be considered for assessing the effectiveness of these CIL techniques. Among all, accuracy is the most common measure used for evaluating standard classifiers. However, accuracy is dependent to the class distribution and is not a good indicator for the skewed-class data. Because of this, other frequently used metrics such as F-measure, G-mean and AUC are considered in this work to

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

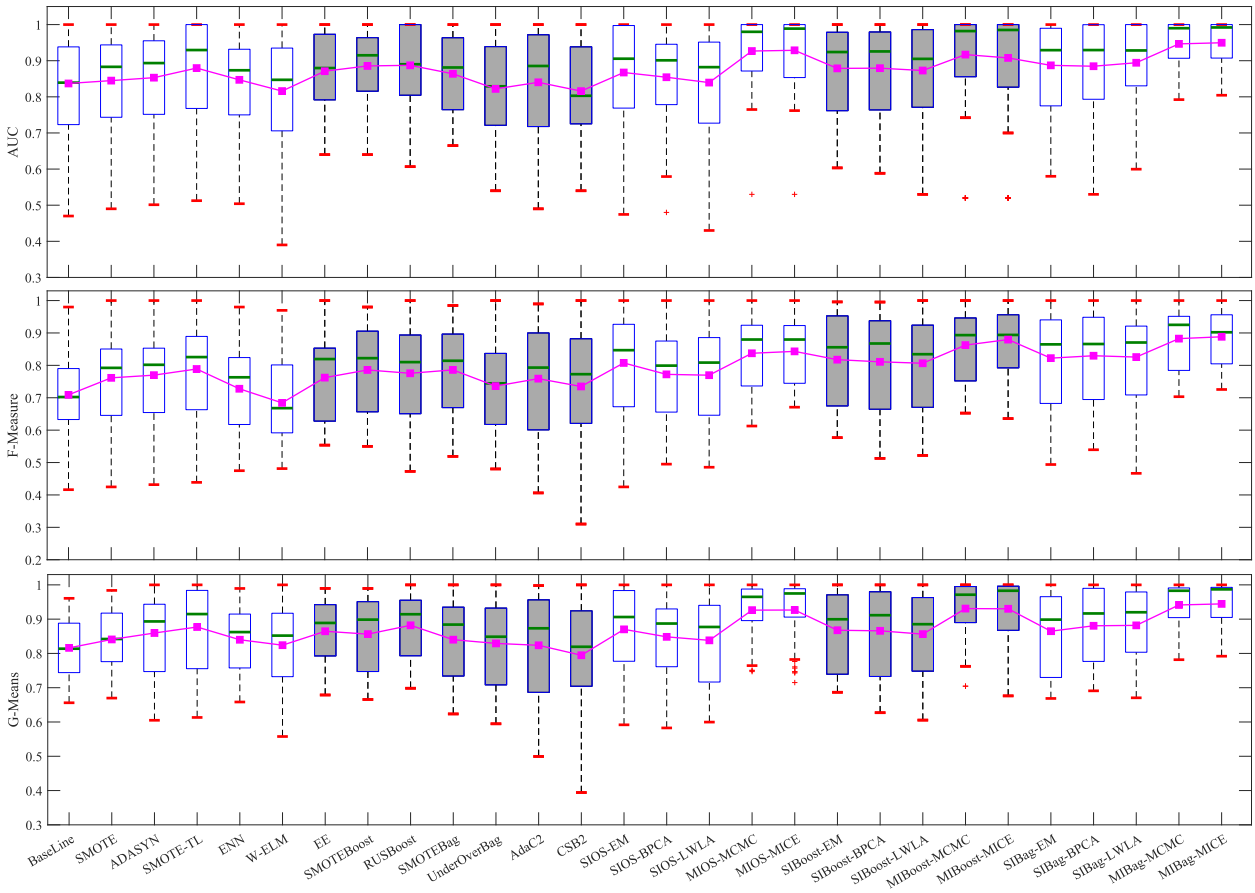



Fig. 3. Comparison of all CIL methods, i.e., the commonly used techniques and the proposed novel methods in terms of AUC, F-measure, and G-mean.

perform a standard and proper assessment of the techniques under class imbalance condition.

Given a binary class imbalance dataset $\mathcal{S} = \{(x_i, y_i)|x_i \in X, y_i \in Y\}_{i=1}^{m}$, the predicted class labels are saved in $\hat{Y}$. Comparing the actual classes $Y$ with the predicted ones $\hat{Y}$ results in calculating various measures including the amount of correctly and incorrectly classified samples named as True Positive (TP) and False Positive (FP); also the number of correctly and incorrectly rejected samples referred as True Negative (TN) and False Negative (FN), respectively. Once the confusion matrix is constructed, the performance metrics, F-measure and G-mean can be calculated as follows:

$$F - measure = \frac{(2 \times precision \times recall)}{precision + recall} \tag{2}$$

$$G - mean = \sqrt{TPR \times TNR}, \tag{3}$$

in which precision $= TP/(TP + FP)$ True Positive Rate (TPR) or recall $= TP/(TP + FN)$ and True Negative Rate (TNR)$= TN/(FP + TN)$. In addition, to provide a comprehensive assessment over the performance of different classifiers, AUC is also taken into account [46].

### 4.3 Empirical Results

Different CIL methods that are representative of data-level approaches (i.e., SMOTE, ADASYN, SMOTE-TL, ENN), algorithm-level approaches (i.e., WELM), hybrid ensembles (i.e., EE), boosted ensembles (i.e., SMOTEBoost and RUSBoost), bagged ensembles (i.e., SMOTEBag and UnderOverBagging) and cost-sensitive ensembles (i.e., AdaC2 and CSB2) along with the proposed methods are applied to various datasets (see Fig. 1). Three different performance metrics (PMs) are considered in order to perform a comparison over the performance of these CIL methods.

Fig. 3 presents performance of these CIL methods along with the BaseLine w.r.t. the evaluation metrics; F-measure, G-mean and AUC. The box-plot presentation of each CIL method aims to summarize the statistical characteristics of the attained performance measures. Each box shows distribution of the attained performance measure by each CIL technique. The mean values of the box-plots, showing the averaged performance, are connected to each other to provide a smooth comparison between different CIL methods. Besides, for the sake of a better comparison and to provide an explicit distinction, the boxplots are colored in white and gray. In Fig. 3, it can be seen that the proposed methods could successfully increase the classification performance. The smallest boxes, which show the most stable methods, belong to those techniques that make use of multiple imputation. In addition, MIOS could generally outperform SIOS, and also, MIBoost and MIBag could achieve better performance compared to SIBoost and SIBag, respectively.

Moreover, the averaged performances of CIL methods are compared with the BaseLine. Table 2 contains the summary

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
The Averaged Performance Measures (AUC, F-measure, and G-mean) Attained by Each CIL Technique Over all Datasets

| CIL Techniques | AUC | F-measure | G-mean | Overall Rank |
|---|---|---|---|---|
| **Data Level Approaches** | | | | |
| SMOTE | 0.79 | 5.25 | 2.46 | 21.3 |
| ADASYN | 1.61 | 6.08 | 4.33 | 18.3 |
| SMOTE-TL | 4.26 | 7.97 | 6.08 | 12 |
| ENN | 1.0 | 1.88 | 2.36 | 23 |
| **Algorithm Level Approaches** | | | | |
| WELM | -2.1 | -2.44 | 0.78 | 26 |
| **Ensemble-Based Approaches** | | | | |
| EE | 3.39 | 5.31 | 4.84 | 17.3 |
| SMOTEBoost | 4.85 | 7.68 | 4.01 | 14.3 |
| RUSBoost | 5.02 | 6.65 | 6.58 | 10.6 |
| SMOTEBag | 2.68 | 7.7 | 2.38 | 18 |
| UnderOverBag | -1.5 | 2.69 | 1.29 | 24.3 |
| **Cost-Sensitive Approaches** | | | | |
| AdaC2 | 0.32 | 5.0 | 0.75 | 24 |
| CSB2 | -2.1 | 2.58 | -2.15 | 26.3 |
| **Imputation-based Algorithms** | | | | |
| SIOS-EM | 3.04 | 9.84 | 5.4 | 13.3 |
| SIOS-BPCA | 1.73 | 6.32 | 3.21 | 18.6 |
| SIOS-LWLA | 0.25 | 6.06 | 2.19 | 22.3 |
| MIOS-MCMC | 8.98 | 12.8 | 11.0 | 5.3 |
| MIOS-MICE | 9.18 | 13.42 | 11.03 | 4.3 |
| SIBoost-EM | 4.18 | 10.86 | 5.14 | 12 |
| SIBoost-BPCA | 4.23 | 10.19 | 4.97 | 12.3 |
| SIBoost-LWLA | 3.58 | 9.75 | 4.01 | 15.3 |
| MIBoost-MCMC | 7.79 | 15.35 | 11.46 | 4 |
| MIBoost-MICE | 7.07 | 17.06 | 11.39 | 4.3 |
| SIBag-EM | 5.01 | 11.33 | 4.86 | 10.6 |
| SIBag-BPCA | 4.77 | 12.07 | 6.44 | 9 |
| SIBag-LWLA | 5.73 | 11.66 | 6.56 | 7.6 |
| MIBag-MCMC | 10.96 | 17.35 | 12.51 | 2 |
| MIBag-MICE | 11.27 | 17.92 | 12.85 | 1 |

*The results are shown as percentage increments or decrements w.r.t. the BaseLine.*

of the results obtained through applying all CIL techniques for classifying all class imbalanced datasets with varying IR in terms of AUC, F-measure and G-mean. The attained results show the percentage increments or decrements w.r.t. the baseline learner C4.5 on average. These CIL techniques are later ranked in terms of each performance metric. The averaged ranks over all performance metrics are then calculated and reported in the rightmost column of the table. MIBag-MICE and MIBag-MCMC are ranked as top two methods on average over all datasets since they outperform other CIL techniques in terms of each performance measure. Moreover, considering Table 2 and Fig. 3, one may conclude that multiple-imputation-based oversampling integrated with ensemble learning could provide better classification performances compared to other CIL methods.

Furthermore, these 42 datasets are sorted based on IR and divided into three different categories of equal depth, indicating Low (IR: 1.38 to 3.25), Medium (IR: 3.36 to 13.87) and High (IR: 14.3 to 41.4) IR datasets. The classification performances attained by each CIL method over all datasets are then calculated for each category of Low, Medium and High IR. The obtained performance measurements are then averaged and compared to the base learner.

The attained results are reported in Table 3 as percentage increments or decrements w.r.t. the base learner in terms of AUC, F-measure and G-mean. The bold entries in Table 3 stand for the top two highest performance increments that are mostly correspond to the combination of multiple imputation-based oversampling techniques and ensemble learners. In other words, MIOS integrated with boosting and bagging outperform other competitors in all categories of low, medium and high imbalance ratios. This can be also seen in box plots of the Fig. 3 as the smallest boxes with highest average performance belong to MIBag, MIBoost, and MIOS techniques. Moreover, in order to investigate the efficiency of MIOS-based approaches on all datasets, statistical significant tests are performed that are explained in details in the next subsection.

### 4.4 Statistical Analysis

In order to provide a certain reassurance and validate the experimental results and analysis, a non-parametric statistical test, i.e., Friedman's test, is employed to compare these CIL methods ($\eta$ methods) over different datasets ($\varphi$ datasets) [47]. In Friedman's comparison, first each CIL method is ranked separately on each dataset, where $r_p^q$ denotes the rank of $q$th CIL method on $p$th dataset (in case of ties, average ranks are shared.). Then, the average rank of the CIL method over all $\varphi$ datasets is calculated as $R_q = 1/\varphi \sum_{p=1}^{\varphi} r_p^q$. Under the null hypothesis, states that all the CIL methods have equal performance (i.e., $R_q$ should be equal), the following Friedman statistic $F_F$ will be distributed based on $F$-distribution with $\eta - 1$ numerator degree of freedom and $(\eta - 1)(\varphi - 1)$ denominator degree of freedom:

$$F_F = \frac{(\varphi - 1)\chi_F^2}{\varphi(\eta - 1) - \chi_F^2} \tag{4}$$

where

$$\chi_F^2 = \frac{12\varphi}{\eta(\eta+1)} \left[ \sum_{q=1}^{\eta} R_q^2 - \frac{\eta(\eta+1)^2}{4} \right]. \tag{5}$$

Table 4 presents the summary of the results attained through the Friedman's comparison. According to this table, the null hypothesis $H_0$ of equal performance among these CIL methods in terms of each evaluation metric is rejected at the level of significant $\alpha = 0.05$. Since $H_0$ is rejected, a post-hoc test, Nemenyi test [48] is employed to compare the performance of all these CIL methods together, in order to see if the performance of these $q$th CIL method is significantly better than the $q'$th CIL method. If the difference between the average rank of these CIL techniques $q$ and $q'$ over all datasets is greater than the critical difference (CD), then, they are significantly different. CD can be computed as follows, where the critical value $cv_\alpha$ is based on the Studentized range statistic [48].

$$CD = cv_\alpha \sqrt{\frac{\eta(\eta+1)}{6\varphi}}. \tag{6}$$

The CD diagrams along with the average rank of these CIL methods in terms of AUC, F-measure and G-mean are illustrated in Fig. 4, respectively. Groups of methods that are not significantly different at the significance level $\alpha$= 0.05 are connected. According to the Fig. 4, multiple imputation oversampling combined with the bagged and the boosted ensembles are positioned as top four among the others, and, ranked as the most accurate techniques. Moreover, the proposed

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
The Averaged Performance Measures (AUC, F-measure and G-mean) Attained by Each CIL Technique Over all Datasets w.r.t. Their Ratios (i.e., Low IR, Medium IR and High IR)

| | Low | | | Medium | | | High | | |
|---|---|---|---|---|---|---|---|---|---|
| CIL Techniques | AUC | F-measure | G-mean | AUC | F-measure | G-mean | AUC | F-measure | G-mean |
| **Data Level Approaches** | | | | | | | | | |
| SMOTE | 2.10 | 6.26 | 3.55 | 0.25 | 9.09 | 0.79 | 0.03 | 0.40 | 3.04 |
| ADASYN | 2.43 | 8.56 | 7.21 | 1.51 | 9.12 | 0.96 | 0.88 | 0.57 | 4.84 |
| SMOTE-TL | 3.60 | 10.70 | 8.61 | 7.33 | 12.08 | 3.01 | 1.86 | 1.16 | 6.65 |
| ENN | -0.73 | 5.49 | 3.78 | 2.60 | 0.83 | 1.30 | 1.16 | -0.67 | 2.02 |
| **Algorithm Level Approaches** | | | | | | | | | |
| WELM | 1.36 | -5.73 | 2.05 | -2.65 | 0.30 | 0.00 | -5.03 | -2.01 | 0.29 |
| **Ensemble-Based Approaches** | | | | | | | | | |
| EE | -0.25 | 5.86 | 5.53 | 5.18 | 8.97 | 1.98 | 5.25 | 1.12 | 7.01 |
| SMOTEBoost | -0.25 | 9.86 | 6.29 | 7.71 | 12.06 | 2.89 | 7.10 | 1.13 | 2.85 |
| RUSBoost | 2.21 | 6.65 | 6.37 | 6.52 | 11.98 | 2.32 | 6.35 | 1.35 | 11.05 |
| SMOTEBag | 0.28 | 9.44 | 6.63 | 3.05 | 12.87 | 2.67 | 4.71 | 0.80 | -2.13 |
| UnderOverBag | -0.81 | 0.98 | 6.72 | 0.90 | 6.67 | 2.09 | -4.58 | 0.45 | -4.91 |
| **Cost-Sensitive Approaches** | | | | | | | | | |
| AdaC2 | 0.53 | 5.69 | 6.73 | 1.03 | 9.32 | 0.05 | -0.60 | 0.00 | -4.53 |
| CSB2 | 0.32 | 8.28 | 5.66 | -3.02 | 0.49 | 0.09 | -3.64 | -1.01 | -12.21 |
| **Imputation-based Algorithms** | | | | | | | | | |
| SIOS-EM | 2.34 | 10.98 | 8.66 | 6.26 | 12.88 | 2.80 | 0.50 | 2.69 | 4.74 |
| SIOS-BPCA | 0.49 | 2.98 | 5.63 | 4.26 | 13.01 | 2.36 | 0.43 | 2.98 | 1.65 |
| SIOS-LWLA | -0.90 | 10.97 | 6.07 | 1.20 | 4.37 | 2.34 | 0.44 | 2.87 | -1.82 |
| MIOS-MCMC | 10.16 | 11.58 | 8.45 | 9.13 | 12.99 | 3.67 | 7.65 | 3.12 | 11.64 |
| MIOS-MICE | 10.41 | 11.73 | 8.63 | **9.15** | 13.01 | 3.47 | 7.98 | 3.10 | 11.87 |
| SIBoost-EM | 1.64 | 11.35 | 7.93 | 5.63 | 12.89 | 1.88 | 5.27 | 2.79 | 5.61 |
| SIBoost-BPCA | 1.68 | 11.61 | 8.21 | 5.64 | 12.31 | 1.89 | 5.38 | 2.99 | 4.81 |
| SIBoost-LWLA | 0.16 | 10.63 | 7.80 | 5.20 | 12.76 | 1.68 | 5.40 | 2.86 | 2.54 |
| MIBoost-MCMC | 6.30 | 11.33 | **9.07** | 9.12 | 13.01 | **5.87** | **8.57** | **3.65** | 10.74 |
| MIBoost-MICE | 3.45 | 10.91 | **9.10** | 9.10 | 13.25 | 4.33 | **8.68** | **3.89** | 11.26 |
| SIBag-EM | 2.55 | 11.33 | 7.65 | 7.61 | 13.31 | 1.80 | 4.89 | 2.54 | 5.14 |
| SIBag-BPCA | 2.90 | 11.37 | 8.08 | 7.41 | 13.26 | 1.07 | 4.02 | 2.70 | 10.17 |
| SIBag-LWLA | 4.75 | 11.28 | 8.18 | 8.02 | 13.14 | 1.02 | 4.44 | 2.85 | 10.49 |
| MIBag-MCMC | **10.46** | **12.02** | 8.78 | 9.13 | **13.56** | 5.35 | 8.01 | 3.19 | **12.66** |
| MIBag-MICE | **11.19** | **12.07** | 8.80 | **9.17** | **13.61** | **5.41** | 8.14 | 3.21 | **12.73** |

*The results are shown as percentage increments or decrements w.r.t. the BaseLine. The bold entries stand for the top two highest performances.*

method, i.e., MIBag-MICE, could achieve the best rank and significantly outperform other competitors in terms of AUC, F-measure and G-mean.

For the sake of reliable conclusion, the performance of the proposed methods under categories of SIBag, SIBoost, MIBag and MIBoost are needed to be carefully studied and compared w.r.t. their predecessors that are actually bagging, boosting and sampling techniques including SIOS and MIOS. The comparison is made to investigate the effect of different components of the algorithms. The goal is to find that, which combination of the re-sampling techniques and the ensemble learners (i.e., SIBag, SIBoost, MIBag and MIBoost) could improve the performance compared to either ensembles (i.e., boosting and bagging), or re-sampling methods (i.e., SIOS and MIOS). To this aim, Fig. 5 is provided to present the average amount of F-measure, G-mean and AUC for boosted (e.g., Adaboost) and bagged CIL methods without applying the proposed oversampling methods, and also after applying SIOS and MIOS oversampling methods. Besides, integrations of SIOS and MIOS with ensemble learning techniques are also considered into account. In Fig. 5, the standard deviation of the performance measures is highlighted in different background colors. Fig. 5 shows that regardless of the type of single imputation used in the *SI* function, SIBoost and SIBag approaches have almost similar performances. This statement is also true for multiple imputation methods. In other words, the type of imputation strategy used in Algorithms 4 (i.e., EM, BPCA and LWLA) and in Algorithms 5 (i.e., MCMC and MICE) has a very limited impact on the performance. Actually, the efficiency of the proposed techniques are mostly related to the oversampling strategies SIOS and MIOS; which is also integrated to the boosted and the bagged ensembles such as SIBoost, SIBag, MIBoost and MIBag. Multiple imputation-based oversampling, MIBoost and MIBag could achieve better performances compared to single imputation-based over sampling, SIBoost and SIBag, respectively. Hence, MIOS could be considered as an effective component that could enhance the

TABLE 4
Summary of the Friedman Statistics $F_F$ and Critical Value in Terms of Each Evaluation Metric

| Evaluation Metric | $F_F$ | Critical value ($\alpha = 0.05$) |
|---|---|---|
| AUC | 101 | 1.5 |
| F-measure | 425 | |
| G-mean | 207 | |

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

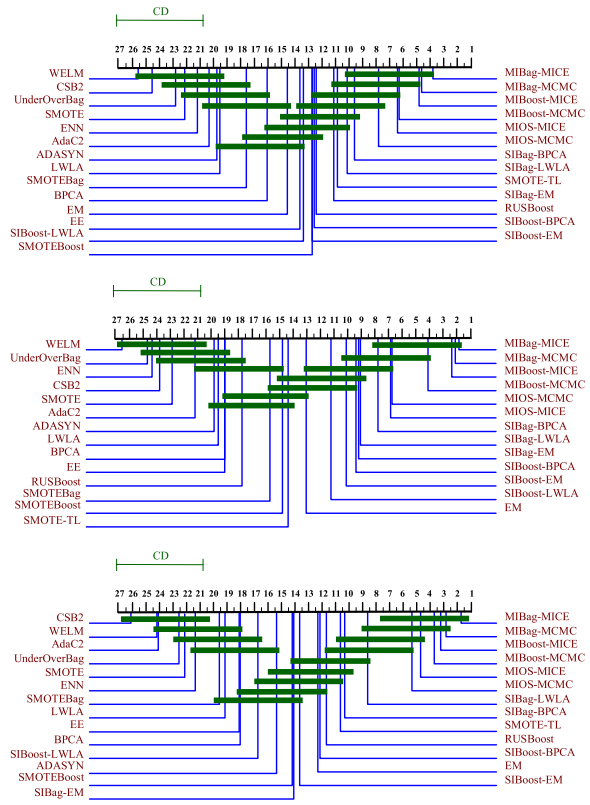



Fig. 4. Critical difference diagram for the Nemenyi test. Comparison of all CIL methods against each other in terms of AUC (panel on the top), F-measure (middle), and G-mean (bottom). The averaged rank of each CIL technique is shown (i.e., rank 1 indicates the method with the highest performance). Groups of methods that are not significantly different at $\alpha = 0.05$ are connected.

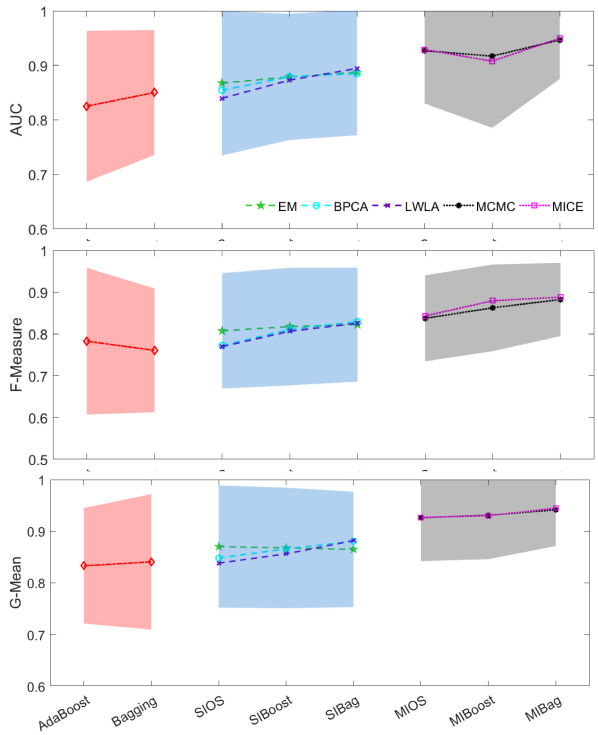



Fig. 5. Comparison of the proposed methods, MIBag, MIBoost, SIBag, and SIBoost along with different core components (i.e., MIOS, SIOS, Adaboost, and Bagging), over different single and multiple imputation methods in terms of AUC, F-measure, and G-mean.

TABLE 5
Summary of Computational Complexity of Ensemble Algorithms

| Algorithm | Complexity |
|---|---|
| SMOTEBoost | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |
| RUSBoost | $\mathcal{O}(T\xi(d))$ |
| SMOTEBag | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |
| UnderOverBag | $\mathcal{O}(T\xi(d))$ |
| ADAC2 | $\mathcal{O}(T\xi(d))$ |
| CSB2 | $\mathcal{O}(T\xi(d))$ |
| EasyEnsemble | $\mathcal{O}(T\xi(d))$ |
| SIBoost | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |
| MIBoost | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |
| SIBag | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |
| MIBag | $\mathcal{O}(T(\xi(d) + m'\varsigma(d)))$ |

accuracy. The better performance of MIOS compared to SIOS can be as a result of the greater randomness in estimation of missing scores. Multiple imputation, which estimates several values for each missing score, reflects the added uncertainty due to the fact that imputed values are not actual values. Multiple imputation then generates multiple datasets by adding a random value to estimation aiming to restore randomness lost in the imputation process. Thus, multiple imputation brings the uncertainty of the estimation into account and reveals a possible range for the estimation of the missing scores, resulting in a more diverse synthetic minority class samples during the oversampling (MIOS). SIOS, on the other hand, makes use of single imputation strategies, which merely estimate one value for each missing score, and, thus, do not reflect any uncertainty into account in returning one synthetic minority class sample. In addition, the bagged ensembles combined with oversampling methods outperform the boosted ones. The best performance can be achieved by the integration of MIOS with the bagged ensembles.

### 4.5 Computational Complexity

Table 5 summarizes the computational complexity of the ensemble algorithms for CIL, where we have assumed that the complexity of updating a base learner and making a prediction is $\mathcal{O}(\xi(d))$, and the complexity of generating $m'$ synthetic samples is $\mathcal{O}(m'\varsigma(d))$, where $d$ is the feature dimension. Note that we have omitted $m$ in the complexities as we assume that it is the same for all the algorithms for simplicity.[1] Roughly speaking, the algorithms are clustered into two groups – one generating synthetic data points and one not. The algorithms in the former group have a higher computational cost. For example, it takes $\mathcal{O}(m'm^{min}d)$ (i.e., $\varsigma(d) = m^{min}d$) for SMOTE-based ensembles to generate $m'$ synthetic data points. Similarly, the computational complexity of our approaches depends on the imputation algorithms. For example, the computational complexity of BPCA based algorithm is $\mathcal{O}(m'm^{min}d^2)$ [49], which is higher than SMOTE-based ensembles. However, we would emphasize that the main novelty and contribution of this work is to propose a general

1. In practice, RUSBoost may train base learners with fewer samples, and, therefore, has lower complexity.

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

imputation-based framework for CIL, rather than the imputation algorithms themselves. In other words, our approaches can readily accommodate any existing imputation method for CIL. Lower computational cost can be achieved by using more efficient algorithms. For instance, much lower computational costs can be achieved when approximation method is applied to BPCA [50]. More important, empirical evaluations presented in the previous subsections show that our approaches significantly outperform other competitors.

## 5 Conclusion

To sum up, the proposed CIL methods show satisfactory results and could improve the average performance under the class imbalance condition with different imbalance ratios. Generally the performance achieved by bagged ensembles are higher than the boosted ones. In addition, the combination of the bagged ensemble and multiple imputation techniques, which result in MIBag could outperform other CIL techniques. While multiple imputation methods could achieve better performance compared to their peer single imputation method, the obtained results show that the performance of MIBag-MICE are significantly better than other CIL methods in terms of F-measure, G-mean and AUC. While many datastes in real world applications are not complete and may contain some missing values, we aim to develop algorithms in our future work that can help for class imbalance learning in the presence of missing data. This work could be extended for multiclass problems, which looks to be a valuable direction for future research.

## Acknowledgments

The authors gratefully acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 401226689].

## References

[1] A. D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1–14, Aug. 2018.

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

[3] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, pp. 1263–1284, Sep. 2009.

[4] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst. Man Cybern. Syst. Part C*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[5] X. Zhang and B. Hu, "A new strategy of cost-free learning in the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2872–2885, Dec. 2014.

[6] 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.

[7] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Nete. Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, Aug. 2018.

[8] H. Cao, X. L. Li, D. Y. K. Woon, and S. K. Ng, "Integrated oversampling for imbalanced time series classification," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2809–2822, Dec. 2013.

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

[10] G. Wu and E. Y. Chang, "KBA: Kernel boundary alignment considering imbalanced data distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 786–795, Jun. 2005.

[11] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

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

[13] K. Ting, "A comparative study of cost-sensitive boosting algorithms," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 983–990.

[14] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, Oct. 2014.

[15] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28. no. 12, pp. 3353–3366, Dec. 2016.

[16] 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.

[17] S. Wang, L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.

[18] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1532–1545, Jun. 2016.

[19] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 206–219, Jan. 2013.

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

[21] S. Gomez, L. Hernandez-Callejo, B. Martinez, and A. Sanchez-Esguevillas, "Exploratory study on class imbalance and solutions for network traffic classification," *Neurocomputing*, vol. 343, pp. 100–119, 2019.

[22] C. Lin *et al.*, "Minority oversampling in kernel adaptive subspaces for class imbalanced datasets," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 950–962, May 2018.

[23] 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, 2017.

[24] O. Loyola-Gonzlez, J. Martnez-Trinidad, M. García-Borroto, and J. A. Carrasco-Ochoa, "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases," *Neurocomputing*, vol. 175, pp. 935–947, 2016.

[25] V. Lpez, A. Fernndez, S. Garca, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, 2013.

[26] D. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.

[27] G. Batista, A. Bazzan, and M. Monard, "Balancing training data for automated annotation of keywords: A case study," in *Proc. 2nd Brazilian Workshop Bioinf.*, 2003, pp. 35–43.

[28] H. He, Y. Bai, E. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 1322–1328.

[29] C. Ling, V. Sheng, and Q. Yang, "Test strategies for cost-sensitive decision trees," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1055–1067, Aug. 2006.

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

[31] S. Datta, S. Nag, and S. Das, "Boosting with lexicographic programming: Addressing class imbalance without cost tuning," *IEEE Trans. Knowl. Data Eng.*, early access, Jan. 22, 2019, doi: 10.1109/TKDE.2019.2894148.

[32] R. Barandela, R. Valdovinos, and J. Sanchez, "New applications of ensembles of classifiers," *Pattern Anal. App.*, vol. 6, pp. 245–256, 2003.

[33] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1088–1099, Jul. 2006.

[34] C. Li, "Classifying imbalanced data using a bagging ensemble variation," in *Proc. 45th Annu. Southeast Regional Conf.*, 2007, pp. 203–208.

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.

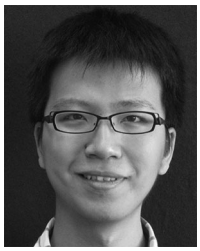
[35] X. Liu, J. Wu, and Z. Zhou, "Exploratory under-sampling for class-imbalance learning," in *Proc. 39th Proc. IEEE Int. Conf. Data Mining*, 2006, pp. 965–969.
[36] X. Y. Liu, J. Wu, and Z. H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst. Man Cybern B*, vol. 39, no. 2, pp. 539–550, Apr. 2009.
[37] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 155–164.
[38] 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.
[39] T. Schneider, "Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values," *J. Climate*, vol. 14, pp. 853–871, 2001.
[40] O. Shigeyuki, S. Masa-aki, T. Ichiro, M. Morito, M. Ken-Ichi, and I. Shin, "A Bayesian missing value estimation method for gene expression profile data," *Bioinf.*, vol. 19, pp. 2088–2096, 2003.
[41] C.-C. Liu, D.-Q. Dai, and H. Yan, "The theoretic framework of local weighted approximation for microarray missing value estimation," *Pattern Recognit.*, vol. 43, no. 8, pp. 2993–3002, 2010.
[42] T. E. Raghunathan, J. M. Lepkowski, J. V. Hoewyk, and P. Solenberger, "A multivariate technique for multiply imputing missing values using a sequence of regression models," *Survey Methodology*, vol. 27, no. 1, pp. 85–95, 2001.
[43] D. Rubin, *Multiple Imputation for Nonresponse in Surveys*, Hoboken, NJ, USA: Wiley, 2009.
[44] J. Moreno-Torres, J. Saez, and F. Herrera, "Study on the impact of partition-induced dataset shift on *k*-fold cross-validation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1304–1312, Aug. 2012.
[45] V. Lopez, A. Fernandez, and F. Herrera, "On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed," *Inf. Sci.*, vol. 257, pp. 1–13, 2014.
[46] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.
[47] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
[48] P. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Princeton University, Princeton, NJ, USA, 1963.
[49] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[50] S. Nakajima, M. Sugiyama, and S. D. Babacan, "On Bayesian PCA: Automatic dimensionality selection and analytic solution," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.*, 2011, pp. 497–504.



**Roozbeh Razavi-Far** (S'08–M'17) received the BSc degree in electrical and computer engineering, the MSc and PhD degrees from the Amirkabir University of Technology, and achieved a second PhD degree from Politecnico di Milano. He was a postdoctoral fellow with the Université libre de Bruxelles, Belgium, an NSERC postdoctoral fellow, and a lecturer at University of Windsor, Canada. He is currently a lecturer, academic advisor, and adjunct professor with the Department of Computer Engineering, University of Windsor. His research interests include machine learning, data mining, big data analytics, and computational intelligence. He is a senior member of the IEEE.



**Maryam Farajzadeh-Zanjani** (S'15) received the BSc degree in computer engineering from Khayyam University, in 2008, the MSc degree in computer engineering from Eastern Mediterranean University (EMU), in 2012, and the second MSc degree in computer engineering from the University of Windsor, Canada, in 2017. She is currently working toward the PhD degree in the Department of Computer Engineering, University of Windsor. Her research interests include data mining, machine learning, and computational intelligence. She is a student member of the IEEE.



**Boyu Wang** received the BEng degree in information engineering from Tianjin University, China, the MSc degree in computer engineering from the University of Macau, China, and the PhD degree in computer science from McGill University, Canada. He was a postdoctoral researcher at Princeton University. He is a postdoctoral researcher with the Department of Computer and Information Science, University of Pennsylvania. His research interests include machine learning, and reinforcement learning.



**Mehrdad Saif** (S'84–M'84–SM'05) received the BSc, MSc, and the DEng degrees in 1982, 1984, and 1987, respectively. In 1987, he joined the School of Engineering Science, Simon Fraser University. From 2002 to 2011, he was the director of the School of Engineering Science. He has been the dean of the Faculty of Engineering, University of Windsor since 2011. He has published about 300 journal and conference papers plus an edited book. He is a senior member of the IEEE.



**Shiladitya Chakrabarti** received the BSc and MEng degrees (honors) in computer engineering from the University of Windsor, Canada. He is currently working toward the MSc degree in computer engineering at the University of Windsor. His research interests include data mining, machine learning, and pattern recognition.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: UNIVERSIDAD DE CORDOBA. Downloaded on May 30,2024 at 20:26:16 UTC from IEEE Xplore. Restrictions apply.