A Bootstrap-Based Iterative Selection for Ensemble Generation

Dayvid V. R. Oliveira, Thyago N. Porpino, George D. C. Cavalcanti and Tsang Ing Ren
Centro de Informática
Universidade Federal de Pernambuco, Brazil
http://www.cin.ufpe.br/~viisar
{dvro, tnp, gdcc, tir}@cin.ufpe.br

***Abstract*—We propose a bootstrap-based iterative method for generating classifier ensembles called *Iterative Classifier Selection Bagging* (ICS-Bagging). Each iteration of ICS-Bagging has two phases: i) bootstrap sampling to generate a pool of classifiers; and, ii) selection of the best classifier of the pool using a fitness function based on the ensemble accuracy and diversity. The selected classifier is added to the final ensemble. The bootstrap sampling runs on each iteration and updates the probability of sampling per class based on the class error rate. This process is repeated until the number of classifiers in the final ensemble is reached. For the specific case of imbalanced datasets, we also propose the SMOTE-ICS-Bagging, a variation of the ICS-Bagging that runs SMOTE at the beginning of each iteration in order to reduce the class imbalance before data sampling. We compared the proposed techniques with Bagging, Random Subspace and SMOTEBagging, using 15 imbalanced datasets from KEEL. The results show the proposed techniques outperform all other techniques in accuracy. Ranking diagrams revealed that the proposed algorithms achieved the highest ranking in accuracy, outperforming SMOTEBagging, a renowned ensemble generation method for imbalanced datasets.**

***Keywords*—Ensemble Generation, Multiple Classifier Systems, Bagging, Imbalanced Datasets, SMOTE.**

I. Introduction

In many real-world classification datasets, the instances of one class are greatly outnumbered by the instances of the other classes. This imbalance gives rise to the so called class imbalance problem, which is the problem of learning a class that has a small set of examples when compared to the other classes. In recent years, the class imbalance problem has emerged as one of the great challenges of data mining [1], and it is very common in practice, being present in areas such as medical diagnosis [2], fraud detection [3], and face recognition [4].

On imbalanced datasets, the underrepresented class, called the minority or positive class, is usually the class of interest (i.e. higher misclassification cost), which makes it essential that the few instances available for this class be appropriately learned by the model. This is not usually the case for standard classification learning algorithms, as most of them do not take class distribution into account and have a strong bias towards the majority class. This results in poor learning of the minority class. As this class is so important in these types of problems, it makes sense to explicitly deal with the class imbalance problem.

Numerous methods exist to treat imbalanced datasets [5] [6] [7] [8], [7]. These methods can be categorized into 3 major groups: (1) data sampling, in which the dataset is preprocessed so that a standard learning method can be used [9]; (2) algorithmic modification, which involves adapting standard learning methods to take class distribution into account; (3) cost-senstive learning [10], that uses approaches at the data level, at the algorithmic level, or both.

Ensemble methods [11] [12] [13] (also known as Multiple Classifier Systems) are an important area of research in machine learning, and have been proven in practice to increase the accuracy of classifier systems [13]. These methods consist of training not just one classifier (i.e. expert), but several, and combining their outputs (i.e. expert opinions) in the hope that the results will be better than any classifier, or at least trying to avoid the choice of the worst classifier.

The process to generate good classifiers for an ensemble is not trivial, as the criteria for a "good" ensemble is not only the accuracy of its classifiers, but that they are as uncorrelated as possible. To heuristically measure this degree of "uncorrelatedness", the concept of diversity [14] [15] is used. Since the training of each classifier already optimizes for accuracy, the ensemble designer must choose how to improve diversity in the ensemble. There are several methods that do this, and they can be divided in: (1) training data manipulation [16] [17] [18], (2) randomization [19] and (3) different models/architectures [20]. Ensemble methods are also frequently adapted to imbalanced domains [8], either by preprocessing the data before training each classifier [21] [22] or by using a cost-sensitive framework.

In this paper, we proposed a technique for generating classifier ensembles called Iterative Classifier Selection Bagging (ICS-Bagging). ICS-Bagging is a bootstrap-based iterative method composed of two steps: first, a bootstrap sampling generates a pool of classifiers, and, after, the best classifier of the pool is selected using a fitness function based on the ensemble accuracy and diversity.

The main contributions of this paper are two new proposed methods: (1) ICS-Bagging: an ensemble generation method that chooses what classifiers should be added in the final ensemble by using ensemble accuracy and diversity; (2) SMOTE-ICS-Bagging (SICS-Bagging): a variant of ICS-Bagging that explicitly deals with the class imbalance problem. Both methods are shown to beat the state-of-the-art according to a recent survey [8].

This paper is organized as follows. Section 2 presents the background for the proposed methods, going into more detail

about imbalanced datasets solutions and ensemble methods. Section 3 presents ICS-Bagging and SICS-Bagging. Next, in Section 4, we detail the experiments, and in Section 5 we present the conclusions.

II. Background

In the field of classification, imbalanced datasets are a common occurrence and dealing with such datasets is difficult problem [23] [1]. The main characteristic of such datasets is the fact that it has an imbalance in the class distribution.

Most of the standard learning algorithms expect a balanced training set. Therefore, good models for standard classification are not necessarily the best for imbalanced datasets. Some of the reasons for this are [7]:

- The use of performance metrics that do not take class imbalance into account, such as the accuracy rate, may provide an advantage to the majority class.
- Classification rules that predict the minority class may be too specific, and so they are discarded in favor of more general rules.
- Very small clusters of minority class examples can be identified as noise, and therefore they could be wrongly discarded by the classifier. Also, a few noisy examples can degrade the identification of the minority class, since it has fewer examples to train with.

Multiple methods exist to deal with the class imbalance problem, [7] categorizes these methods into 3 major groups:

- **Data sampling**: training data is modified in such a way to produce a more or less balanced class distribution that allow classifiers to perform in a similar manner to standard classification.
- **Algorithmic modification**: adapts standard learning algorithms to be more attuned to class imbalance issues.
- **Cost-sensitive learning** [10]: This type of solutions incorporate approaches at the data level, at the algorithmic level, or at both levels combined, considering higher costs for the misclassification of examples of the positive class with respect to the negative class, and therefore, trying to minimize higher cost errors.

A. Data sampling

One of the simpler alternatives (listed above) for dealing with imbalanced datasets, is preprocessing the data so as to diminish or eliminate the imbalance. In this vein, there are 3 options: (1) oversampling the minority class [24], (2) undersampling the majority class [25] [26], (3) combining oversampling and undersampling [27]. In this paper we focus on SMOTE [24], which is an oversampling technique that was shown to be very effective as a preprocessing step for dealing with the class imbalance problem [8].

SMOTE [24] is an oversampling technique that creates synthetic examples by interpolating between minority instances that are close to each other. Synthetic instances are created along the line segment between each minority class instance and one of its k nearest neighbors (from the same class). The procedure is basically this: for each minority class example, randomly select one of its k nearest neighbors (from the same class), take the difference between the two vectors, multiply the difference by a random number (between 0 and 1) and add it to the minority class example.

The intuition on why SMOTE improves performance on imbalanced datasets is that it provides more related minority class samples to learn from, thus allowing a learning algorithm to create broader decision regions, leading to more coverage of the minority class. This of course, has an implicit locality bias, that is, it assumes that interpolating two nearby samples of the same class, will generate a point in the same class. That is not always the case, so one of the problems of SMOTE is its sensitivity to the complexity of the dataset.

B. Ensemble generation

1) Bagging: Bagging [16] is an ensemble meta-algorithm that was designed to improve the accuracy and stability of supervised machine learning models. Its main concept is bootstrap aggregation, where the training set for each classifier is constructed by random uniform sampling (with replacement) instances from the original training set (usually keeping the size of the original data) [28]. The classifiers output is then combined in some way (for classification problems, majority vote is generally used). The bootstrap sampling process generates a different training set for each classifier, which naturally increases ensemble diversity.

2) Boosting: Boosting [17] [29] is a general method for improving the accuracy of any given learning algorithm [30]. The objective of boosting methods is to produce a very accurate (i.e. "strong") classifier by combining rough and somewhat inaccurate (i.e. "weak") classifiers. The boosting method works by iteratively training each classifier, by feeding it a weighted training set. The weights in each instance of the training set are higher for instances often misclassified by the preceding classifiers. This effectively forces each classifier to focus on the current "hardest" examples. After the ensemble is generated, a simple combination scheme may be used (e.g. majority voting).

C. Ensemble Generation for Imbalanced Datasets

The use of ensembles for dealing with imbalanced datasets is one common solution [7], and [8] roughly categorizes these techniques into: (1) bagging-based; (2) boosting-based; (3) hybrid; and (4) cost-sensitive ensembles. Out of these types, we're interested in comparing our proposed technique with the bagging-based SMOTEBagging, since it obtained slightly better results than the best (more robust) techniques tested in [8] and [7].

In SMOTEBagging [21], the bootstrapping procedure used in the original Bagging, is modified so that even more diversity is introduced in the ensemble, this is done by using SMOTE. This technique, at first, uses SMOTE for oversampling the minority class, and then its resampling rate is updated in each iteration (i.e. for each classifier that is trained), from lower to higher values (e.g. 10% - 100%). This ratio defines the number of minority instances to be additionally resampled in each iteration [28].

III. Proposed Algorithms

This section presents the *Iterative Classifier Selection Bagging* (ICS-Bagging) and the *SMOTE Iterative Classifier Selection Bagging* (SICS-Bagging).

A. ICS-Bagging

ICS-Bagging is a bootstrap-based iterative methods for generating classifier ensembles. Each iteration generates a set of classifiers and selects the best classifier to the ensemble. The bootstrap sampling uses a probability of sampling from each class, with this probability being derived from the class error rate. Figure 1 presents the architecture of the ICS-Bagging algorithm.

Follows the explanation of each step:

Preprocessing: Before generating the classifiers, a preprocessing technique might be applied to the training set. This preprocessing can consist of removing or generating features, removing outliers, noisy and redundant data, or generating new data. ICS-Bagging does not have this step, it is only used in the SICS-Bagging and is mentioned later in this section.

Generate K classifiers: This step generates K classifiers using bootstrap sampling (with replacement). In the first step, the weights are the same for all classes, after the first step the weights are updated to prioritize the class that has a higher error rate.

The motivation for using the weights to guide the bootstrap process is that the new generated classifiers are trained with instances that increase the accuracy of the class with the higher error rate.

The motivation for generating $K > 1$ classifiers is to expand a region of search, increasing the probability of finding a classifier that considerably increase the classification accuracy and diversity.

Add the best classifier to the pool: For each of the K generated classifiers the following steps are performed to find the best classifier. Algorithm 1 presents the mechanism to find the classifier to be inserted into the pool.

Algorithm 1 Find Best Classifier

Require: $\mathcal{V}$: validation set
Require: $\mathcal{C}$: list of classifiers
Require: $\mathcal{P}$: pool
Require: *fitness*: a fitness function
1: $best_{index} \leftarrow -1$
2: $best_{value} \leftarrow -1$
3: **for all** $i \in SizeOf(C)$ **do**
4: Add C_i to $\mathcal{P}$
5: $fit_i \leftarrow$ *fitness*$(\mathcal{P})$
6: **if** $fit_i > best_{value}$ **then**
7: $best_{value} \leftarrow fit_i$
8: $best_{index} \leftarrow i$
9: **end if**
10: Remove C_i from $\mathcal{P}$
11: **end for**
12: **return** $\mathcal{C}_{best_{index}}$

In Algorithm 1, C is the list of K generated classifiers, $\mathcal{V}$ is the validation set (in this work, we used the traning set as the validation set) and $\mathcal{P}$ is the current pool of classifiers.

For all classifiers in C, the classifier is added to the pool (Line 4), and the the fitness of the pool is calculated (Line 5). The *fitness* is given by

$$\textit{fitness} = \alpha \times \textit{ACC} + (1 - \alpha) \times \textit{DIV} \tag{1}$$

where *ACC* is the classification accuracy of the pool, *DIV* is the diversity metric, and α is the balance parameter that regulates the objective function between high classification accuracy and high diversity, and has a range of 0.51 to 0.99.

If the pool achieves the highest fitness with this classifier, the index of this classifier is saved in $best_{index}$ (Line 6 - 9). The classifier is removed from the pool (Line 10) and the process starts again with another classifier, until the best classifier is returned (Line 12).

For the classification of a test sample, any combination rule could be used. In this paper, we used the majority vote rule [12] to combine the outputs of the classifiers in the pool.

Any classification and diversity metric can be used in Equation 1, but both need to be normalized (between 0 and 1). In this paper we used the AUC as classification accuracy because of the imbalanced datasets issue, and the Entropy Measure E [14] as the diversity metric because of the simplicity and running speed.

The Entropy Measure E is a non-pairwise diversity measure that has it's highest value when half classifiers correctly classify a pattern and the other half doesn't. If all classifiers have the same agree on a classification, the ensemble is not considered diverse. The Entropy Measure E is described as

$$E = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{(L - \lceil \frac{L}{2} \rceil)} \textit{min}\{l(z_j), L - l(z_j)\} \tag{2}$$

where L is the number of classifiers of the ensemble, N is the number of samples to be classified, $l(z_i)$ is a function that returns the number of classifiers that correctly classify the sample z_i. This diversity metric varies from 0 to 1, where 1 is the highest diversity and 0 is the lowest, therefore, there is no need for normalization when using this metric.

The motivation for adding only one of the K generated classifiers is because the error rate of each class might change when the best classifier is inserted in the pool, which means, the pool now has different samples to prioritize in order to increase classification rate and diversity.

|pool| = N: If the pool already contains the desired number of classifiers the pool is returned.

Update the weight of each class: Since the error rate of each class might have changed after inserting the new classifier in the pool, the weights need to be updated using Equation 3,

$$weight_{class} = \frac{error_{class}}{\sum_{c \in classes} error_c} \tag{3}$$

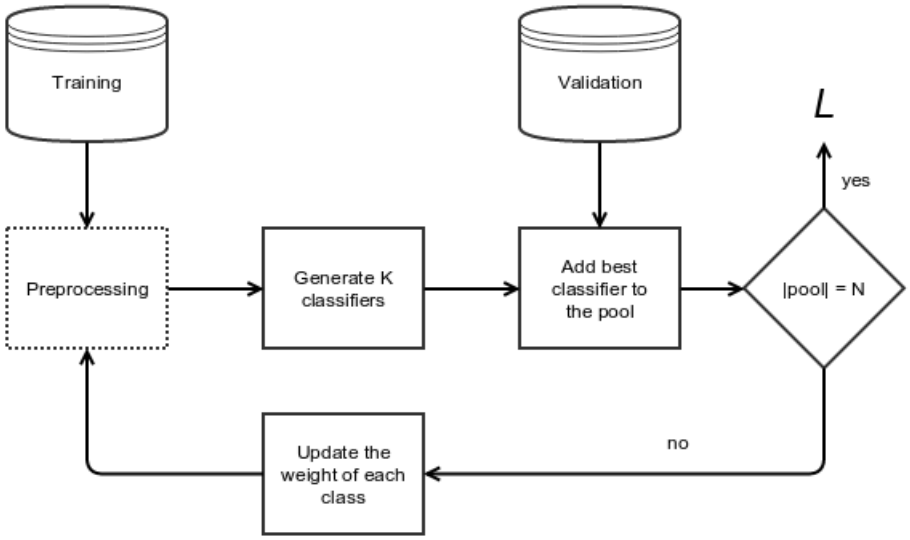

Fig. 1. Architecture of ICS-Bagging and SICS-Bagging. Where L is the final ensemble, and Preprocessing is a step from the SICS-Bagging

where $weight_{class}$ is the weight of the class, $error_{class}$ is the classification error rate of the class, and $\sum_{c \in classes} error_c$ is the sum of the error rate of all classes.

As previously stated, the motivation for updating the weights is to increase the probability of training the new K classifiers with samples from the class with higher error rate of the pool.

Return the pool: The final pool of classifiers L is returned.

B. SMOTE-ICS-Bagging

SMOTE-ICS-Bagging (SICS-Bagging) is a variation of ICS-Bagging in which the Synthetic Minority Over-sampling Technique (SMOTE) is used as a preprocessing phase before generating the K classifiers. This step is performed to increase diversity and to reduce the imbalance ratio when performing bootstrap sampling.

IV. Experiments

This section presents the methodology used in the experiments, and the results of ICS-Bagging and SICS-Bagging.

A. Methodology

The ICS-Bagging and SICS-Bagging methods are evaluated using 15 imbalanced datasets from KEEL [31]. The datasets are binary and have incremental imbalance ratio (IR), given by the number of samples of the majority class divided by the number of samples of the minority class. The datasets used in this study are summarised in Table I that shows the number of samples, the number of attributes, the classes distribution and the IR.

TABLE I. Datasets characteristics

Label	Name	Patterns	Features	% (min., maj.)	IR
1	Glass1	214	9	(35.51, 64.49)	1.82
2	Pima	768	8	(34.84, 66.16)	1.90
3	Iris0	150	4	(33.33, 66.67)	2.00
4	Yeast1	1484	8	(28.91, 71.09)	2.46
5	Vehicle2	846	18	(28.37, 71.63)	2.52
6	Vehicle3	846	18	(28.37, 71.63)	2.52
7	Ecoli1	336	7	(22.92, 77.08)	3.36
8	Ecoli2	336	7	(15.48, 84.52)	5.46
9	Glass6	214	9	(13.55, 86.45)	6.38
10	Yeast3	1484	8	(10.98, 89.02)	8.11
11	Ecoli3	336	7	(10.88, 89.12)	8.19
12	Vowel0	13	988	(9.10, 90.9)	9.98
13	Glass4	214	9	(6.07, 93.93)	15.47
14	Ecoli4	336	7	(5.95, 94.05)	15.8
15	Page-blocks13vs4	472	10	(5.93, 94.07)	15.85

The datasets are partitioned using the *5-fold cross-validation* procedure, which means that the datasets are divided in 5 folds (each one with 20% of the samples) and the experiments are performed 5 times, each time with one of the folds as the test set and the remaining four folds as the training set. This partitioning is performed respecting class proportion.

The evaluation metrics are: classification accuracy and diversity. For the classification accuracy, the metric used was the Area Under the ROC Curve (AUC). This metric was chosen because it is one of the most suitable metrics when dealing with imbalanced datasets. For the diversity, the Entropy Measure E was used. This non-paired diversity metric was used because it is not biased by the AUC (already considered in the fitness function), and for simplicity, because it has the ideal range (from 0 to 1), where 1 is the highest diversity possible.

In order to evaluate the results, we used the statistical paired test *One Sided Wilcoxon Signed Rank Test* [32], comparing ICS-Bagging and SICS-Bagging with the other techniques in this experiment. The level of significance used was $\alpha = 0.1$.

The techniques and parameters used in this experiments are presented in Table II.

TABLE II. ALGORITHMS AND PARAMETERS

Algorithm	Parameters
ICS-Bagging	$N = 40$, *repetition = True* $K = 5, K_{smote} = 5$
SICS-Bagging	$N = 40$, *repetition = True* $K = 5, K_{smote} = 5$
SMOTEBagging	$N = 40$, *repetition = True* $K_{smote} = 5$
Bagging	$N = 40$, *repetition = True*
RandomSubspace	$N = 40$, $features_{max} = 0.5$

The base classifier used in the experiments was the Decision Tree Classifier with maximum depth of 9, and minimum number of samples required to be at a leaf node of 1. The combination scheme used is a simple majority vote.

B. Results

Tables III and IV present the average and standard deviation for, respectively, the AUC and Entropy Measure E. The best result in each dataset is highlighted in bold. The last lines present the results of the Wilcoxon Test, the *p-value* and the result when compared with ICS-Bagging and SICS-Bagging. The symbol "+" when a proposed technique outperformed, the symbol "−" when it was outperformed, and "=" when nothing can be concluded with statistical support (*NA* means *not applicable*).

1) Classification Accuracy: Table III shows that ICS-Bagging and SICS-Bagging outperformed all techniques in classification accuracy. ICS-Bagging outperformed Bagging (*p-value* $= 0.0055$), Random Subspace (*p-value* $= 0.0003$), and the top ensemble technique for imbalanced datasets, SMOTEBagging (*p-value* $= 0.0044$). SICS-Bagging was even better, outperforming all techniques in classification accuracy, including ICS-Bagging.

SICS-Bagging was designed as an improvement of ICS-Bagging, with SMOTE being applied before each iteration in order to increase the likelihood of creating balanced classifiers (following the SMOTEBagging approach). The objective was achieved, and SICS-Bagging was the best technique in 9 out of 15 datasets, and achieved the highest average AUC (0.8703), followed by ICS-Bagging (0.8655).

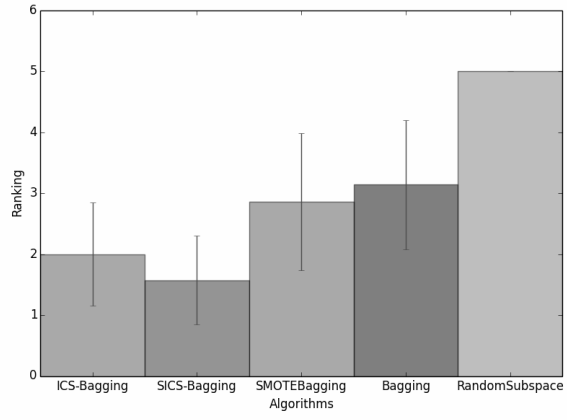

Fig. 2. Average AUC ranking of the ensemble techniques

Figure 2 and Table V show the average and standard deviation AUC ranking of the ensemble techniques in the experiments. The two techniques proposed in this paper achieved the two highest performances. SICS-Bagging achieved the first place with a ranking of 1.53, and ICS-Bagging the second place with a ranking of 1.93.

Table III and Figure 2 show that ICS-Bagging and SICS-Bagging had an excelent performance in classification accuracy with imbalanced datasets, outperforming SMOTEBagging, which was considered one of the best ensemble techniques for imbalanced domains in [8].

2) Diversity: Table IV shows that ICS-Bagging outperformed all techniques in diversity, except for the Random Subspace. Random Subspace achieved a high diversity because it only selects a subset of the features for each classifier, and it was confirmed to be a more effective diversity generator than selecting a subset of samples for each classifier. This can be confirmed when Random Subspace is compared with Bagging (with no preprocessing).

Statistically, SICS-Bagging outperformed only SMOTE-Bagging, but, on average, it achieved the third highest diversity, losing only to Random Subspace and ICS-Bagging. SICS-Bagging was outperformed by ICS-Bagging in diversity because the SMOTE preprocessing improves the AUC of all classifiers, but does not necessarily generate diverse classifiers. This can be confirmed because SMOTEBagging did not improve diversity over Bagging (on average).

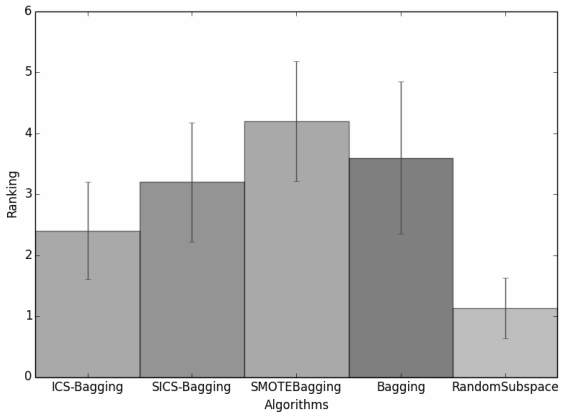

Fig. 3. Average Entropy Measure E ranking of the ensemble techniques

Figure 3 and Table VI show the average and standard deviation ranking of the ensemble techniques in the experiments. ICS-Bagging achieved the second highest diversity, and SICS-Bagging the third highest. The highest diversity ranking was achieved by Random Subspace, which achieved one of the lowest performance in classification accuracy. This indicates that a high Entropy E value does not result in a high classification accuracy when there is a high loss of information (which happens when only a subset of the features is selected).

The purpose of a diversity measure is to predict the accuracy of the ensemble on new data, that is, it is a metric that tries to predict the generalization power of the ensemble. These results show that the Entropy E was not strongly correlated to classification accuracy when using simple majority vote, this scenario could be different if another combination rule was used, or if we used dynamic ensemble selection.

TABLE III. AVERAGE, STANDARD DEVIATION AUC, AND WILCOXON SIGNED RANK TEST

Dataset	ICS-Bagging-40		SICS-Bagging-40		SMOTEBagging-40		Bagging-40		RandomSubspace-40	
glass1	**0.7923**	**0.0500**	0.7829	0.0193	0.7816	0.0770	0.7521	0.0529	0.5133	0.0267
pima	**0.7390**	**0.0194**	0.7377	0.0189	0.6988	0.0307	0.7215	0.0227	0.5766	0.0345
iris0	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**	0.9700	0.0600
yeast1	0.7309	0.0343	0.7204	0.0237	**0.7314**	**0.0282**	0.6810	0.0187	0.5288	0.0342
vehicle2	0.9592	0.0168	0.9630	0.0159	**0.9669**	**0.0127**	0.9594	0.0225	0.5596	0.0403
vehicle3	0.7368	0.0188	**0.7566**	**0.0162**	0.7300	0.0083	0.6600	0.0226	0.5008	0.0106
ecoli1	0.8722	0.0497	**0.8795**	**0.0444**	0.8642	0.0414	0.8480	0.0454	0.5086	0.0172
ecoli2	0.8855	0.0495	**0.8861**	**0.0439**	0.8670	0.0529	0.8714	0.0464	0.5000	0.0000
glass6	0.8917	0.0190	**0.8965**	**0.0643**	0.8886	0.0361	0.8865	0.0624	0.7440	0.1694
yeast3	**0.9089**	**0.0273**	0.9045	0.0248	0.9055	0.0095	0.8434	0.0337	0.5000	0.0000
ecoli3	0.8047	0.0967	**0.8223**	**0.0590**	0.7525	0.0827	0.7724	0.0677	0.5000	0.0000
vowel0	0.9483	0.0573	0.9594	0.0546	0.9455	0.0547	0.9589	0.0540	0.4994	0.0011
glass4	0.8750	0.1877	**0.8825**	**0.1915**	0.8708	0.0982	0.7467	0.1654	0.5000	0.0000
ecoli4	0.8405	0.1143	**0.8655**	**0.0721**	0.8155	0.1194	0.8671	0.1326	0.5000	0.0000
page-blocks-1-3_vs_4	**0.9978**	**0.0045**	**0.9978**	**0.0045**	**0.9978**	**0.0045**	0.9766	0.0414	0.6378	0.0612
Average	0.8655	0.0497	**0.8703**	**0.0435**	0.8544	0.0438	0.8363	0.0526	0.5693	0.0303
P-value (ICS-Bagging)	NA		0.9421		0.0044		0.0055		0.0003	
Result (ICS-Bagging)	NA		−		+		+		+	
P-value (SICS-Bagging)	0.0579		NA		0.0054		0.0008		0.0003	
result (SICS-Bagging)	+		NA		+		+		+	

TABLE IV. AVERAGE, STANDARD DEVIATION ENTROPY MEASURE E, AND WILCOXON SIGNED RANK TEST

Dataset	ICS-Bagging-40		SICS-Bagging-40		SMOTEBagging-40		Bagging-40		RandomSubspace-40	
glass1	0.4445	0.0560	0.4314	0.0570	0.4177	0.0464	0.3786	0.0531	**0.4845**	**0.0473**
pima	0.4284	0.0133	0.4130	0.0197	0.4303	0.0174	0.4131	0.0196	**0.5242**	**0.0212**
iris0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.0080**	**0.0041**
yeast1	**0.4342**	**0.0230**	0.3926	0.0256	0.3779	0.0200	0.3410	0.0029	0.3866	0.0067
vehicle2	0.1006	0.0148	0.0809	0.0039	0.0849	0.0059	0.0886	0.0126	**0.1502**	**0.0163**
vehicle3	**0.3900**	**0.0219**	0.3774	0.0248	0.3757	0.0239	0.3661	0.0177	**0.3900**	**0.0352**
ecoli1	0.1542	0.0361	0.1555	0.0479	0.1474	0.0340	0.1456	0.0337	**0.3431**	**0.0296**
ecoli2	0.1556	0.0440	0.1377	0.0347	0.1174	0.0203	0.1216	0.0286	**0.2441**	**0.0156**
glass6	0.1108	0.0550	0.0894	0.0235	0.0708	0.0305	0.0613	0.0148	**0.1189**	**0.0189**
yeast3	0.0767	0.0076	0.0717	0.0110	0.0619	0.0086	0.0825	0.0086	**0.1592**	**0.0158**
ecoli3	0.1187	0.0364	0.1200	0.0339	0.1089	0.0361	0.1246	0.0290	**0.1842**	**0.0260**
vowel0	0.0346	0.0176	0.0309	0.0148	0.0237	0.0084	0.0299	0.0069	**0.0844**	**0.0066**
glass4	0.0684	0.0299	0.0623	0.0288	0.0557	0.0289	0.0944	0.0312	**0.1123**	**0.0449**
ecoli4	0.0460	0.0228	0.0476	0.0049	0.0253	0.0125	0.0333	0.0151	**0.1087**	**0.0341**
page-blocks-1-3_vs_4	0.0052	0.0065	0.0020	0.0035	0.0012	0.0013	0.0269	0.0090	**0.0461**	**0.0081**
Average	0.1712	0.0257	0.1608	0.0223	0.1533	0.0196	0.1538	0.0189	**0.2230**	**0.0220**
P-value (ICS-Bagging)	NA		0.0018		0.0006		0.0320		0.9982	
Result (ICS-Bagging)	NA		+		+		+		−	
P-value (SICS-Bagging)	0.9982		NA		0.0078		0.1501		0.9996	
Result (SICS-Bagging)	−		NA		+		=		−	

TABLE V. RANKING OF AUC CLASSIFICATION ACCURACY

Algorithm	Mean	Std	Ranking
ICS-Bagging	1.93	0.85	2
SICS-Bagging	1.53	0.72	1
SMOTEBagging	2.73	1.18	3
Bagging	3.20	1.05	4
Random Subspace	5.00	0.00	5

TABLE VI. RANKING OF ENTROPY MEASURE E DIVERSITY

Algorithm	Mean	Std	Ranking
ICS-Bagging	2.40	0.80	2
SICS-Bagging	3.20	0.98	3
SMOTEBagging	4.20	0.98	5
Bagging	3.60	1.25	4
Random Subspace	1.13	0.50	1

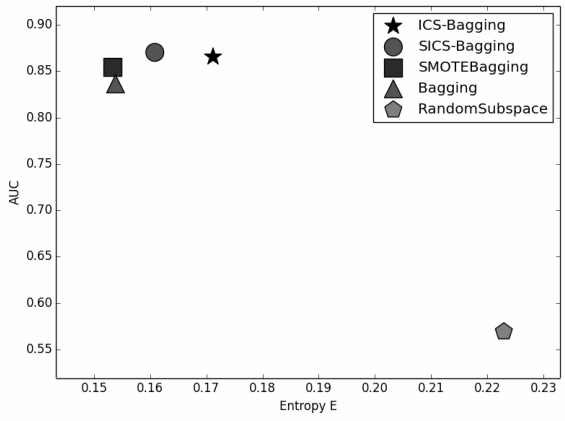

Fig. 4. Dispersion (Diversity vs. AUC) of the ensemble algorithms used in the experiment.

3) Diversity vs. Classification: Figure 4 presents the dispersion graph (Diversity vs. AUC) of the ensemble techniques used in this experiment. This figure shows that, on average, SICS-Bagging achieved the highest classification accuracy and the third highest diversity. ICS-Bagging had the second highest classification accuracy and the second highest diversity.

V. CONCLUSION

This paper proposed a new method of generating ensembles called Iterative Classifier Selection Bagging (ICS-Bagging), and its extension for imbalanced datasets SMOTE Iterative Classifier Selection Bagging (SICS-Bagging). An experimental study concluded that both ICS-Bagging and SICS-Bagging

obtain state-of-the-art results in the datasets tested (according with [8]). Future works include: (1) comparing the proposed techniques with other methods; (2) testing other preprocessing techniques and (3) using other diversity metrics.

ACKNOWLEDGMENT

The authors would like to thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, in portuguese), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, in portuguese) and FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco, in portuguese).

REFERENCES

[1] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.

[2] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural networks*, vol. 21, no. 2, pp. 427–436, 2008.

[3] D. A. Cieslak, N. V. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets," in *Granular Computing, International Conference on*. IEEE, 2006, pp. 732–737.

[4] Y.-H. Liu and Y.-T. Chen, "Total margin based adaptive fuzzy support vector machines for multiview face recognition," in *Systems, Man and Cybernetics, International Conference on*, vol. 2. IEEE, 2005, pp. 1704–1711.

[5] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the International Conference on Machine Learning*, 2007, pp. 935–942.

[6] H. He and E. A. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[7] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.

[8] M. Galar, A. Fernández, E. B. Tartas, H. B. Sola, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 4, pp. 463–484, 2012.

[9] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *Machine Learning*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3201, pp. 39–50.

[10] C. Elkan, "The foundations of cost-sensitive learning," in *International Joint Conference on Artificial Intelligence*, vol. 17, no. 1, 2001, pp. 973–978.

[11] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the International Workshop on Multiple Classifier Systems*, ser. MCS. Springer, 2000, pp. 1–15.

[12] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[13] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[14] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[15] E. K. Tang, P. N. Suganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65, no. 1, pp. 247–271, 2006.

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

[17] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[18] T. K. Ho, "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 8, pp. 832–844, 1998.

[19] L. K. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 10, pp. 993–1001, 1990.

[20] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the International Conference on Machine Learning*, 2004, p. 18.

[21] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Computational Intelligence and Data Mining, Symposium on*. IEEE, 2009, pp. 324–331.

[22] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases*. Springer, 2003, pp. 107–119.

[23] D. V. R. Oliveira, G. R. Magalhaes, G. D. C. Cavalcanti, and T. I. Ren, "Improved self-generating prototypes algorithm for imbalanced datasets," in *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, vol. 1. IEEE, 2012, pp. 904–909.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[25] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proceedings of the International Conference on Machine Learning*, vol. 97. Morgan Kaufmann, 1997, pp. 179–186.

[26] G. E. Batista, A. C. Carvalho, and M. C. Monard, "Applying one-sided selection to unbalanced datasets," in *Proceedings of the Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence*. Springer, 2000, pp. 315–325.

[27] J. Stefanowski and S. Wilk, "Selective pre-processing of imbalanced data for improving classification performance," in *Data Warehousing and Knowledge Discovery*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5182, pp. 283–292.

[28] J. Błaszczyński, J. Stefanowski, and Ł. Idkowiak, "Extending bagging for imbalanced data," in *Proceedings of the International Conference on Computer Recognition Systems*. Springer, 2013, pp. 269–278.

[29] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.

[30] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation and Classification*. Springer, 2003, pp. 149–171.

[31] J. Alcalá, 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," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, pp. 255–287, 2010.

[32] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, pp. 80–83, 1945.