Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches

Miriam Seoane Santos, Jastin Pompeu Soares, Pedro Henriques Abreu, Hélder Araújo and João Santos
CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
Department of Electrical and Computer Engineering, University of Coimbra, Portugal
IPO-Porto Research Centre, Porto, Portugal

***Abstract*—Although cross-validation is a standard procedure for performance evaluation, its joint application with oversampling remains an open question for researchers farther from the imbalanced data topic. A frequent experimental flaw is the application of oversampling algorithms to the entire dataset, resulting in biased models and overly-optimistic estimates. We emphasize and distinguish overoptimism from overfitting, showing that the former is associated with the cross-validation procedure, while the latter is influenced by the chosen oversampling algorithm. Furthermore, we perform a thorough empirical comparison of well-established oversampling algorithms, supported by a data complexity analysis. The best oversampling techniques seem to possess three key characteristics: use of cleaning procedures, cluster-based example synthetization and adaptive weighting of minority examples, where Synthetic Minority Oversampling Technique coupled with Tomek Links and Majority Weighted Minority Oversampling Technique stand out, being capable of increasing the discriminative power of data.**

I. Introduction

Imbalanced Data (ID) occurs when there is a considerable difference between the class priors of a given problem. Considering a binary classification problem, a dataset is said to be imbalanced if there exists an under-represented concept (a minority class) when compared to the other (a majority class) [1]. Prediction models built from imbalanced datasets are most often biased towards the majority concept, which is especially critical when there is a higher cost of misclassifying the minority examples, such as diagnosing rare diseases [2].

Approaches to handle imbalanced scenarios can be mainly divided into data-level approaches, where the data is preprocessed in order to achieve a balanced dataset for classification, and algorithmic-level approaches, where the classifiers are adapted to deal with the characteristic issues of imbalanced data [3]–[6]. By far, data-level approaches are the most commonly used, as they have proven to be efficient, are simple to implement and completely classifier-independent [2], [7]. Data-level strategies fall into two main categories, undersampling and oversampling: the former consists in removing majority examples while the latter replicates the minority examples. Researchers often invest in oversampling procedures since they are capable of balancing class distributions without ruling out potentially critical majority examples [8].

Cross-validation (CV) is a standard procedure to evaluate classification performance; yet, its joint application with oversampling raises some questions for researchers farther from the imbalanced data community. Some researchers not familiarised with the topic tend to misunderstand some aspects of a standard experimental setup in imbalanced domains. One of their frequent misconceptions relates to the joint-use of CV and oversampling algorithms: oversampling seems to be applied to the entire original data, and only then the cross-validation and model evaluation is performed [9]–[12]. This misconception naturally leads to building biased models and producing overoptimistic error estimates (examples of these situations will be illustrated in Section III).

In traditional CV, the entire dataset is initially partitioned into k folds, where k-1 folds are used to train the prediction model and the left-out fold is used for testing. The folds then rotate so that all folds are used for training and testing the model, and the final performance metrics are averaged across the k estimates of each test fold. This process assures that k independent sets are used to test the model, simulating unseen data: the test set is never seen during the training of the model, to avoid overfitting the data. Incorrectly applying oversampling while performing CV may derive into two main issues: overoptimism and overfitting, as we proceed to explain.

Regarding the issue of overoptimism, consider Approach 1 (CV after Oversampling) and Approach 2 (CV during Oversampling) as depicted in Fig. 1. In the first approach (Approach 1) we design a cross-validation setup prone to overoptimism: the entire dataset is first oversampled to achieve a 50-50 distribution between classes and the cross-validation is applied afterwards. In this scenario, it is possible that copies of the same patterns appear in both the training and test sets, making this design subjected to overoptimism (Fig. 1 - CV after Oversampling). In the second approach (Approach 2), the oversampling procedure is performed during cross-validation: the dataset is first divided into k stratified partitions and only the training set (corresponding to $k-1$ partitions) is oversampled (Fig. 1 - CV during Oversampling). In this scenario, the patterns included in the test set are never oversampled or seen by the model in the training stage, thus allowing a proper evaluation of the model's capability to generalize from the training data.

Regarding the issue of overfitting, some researchers directly associate it to all oversampling procedures, while others refer to the overoptimistic results of a CV approach as "overfitting", which confuses both concepts and hinders their identification. For this reason, we here distinguish both ideas and explain how they relate to CV and oversampling approaches, providing some examples:

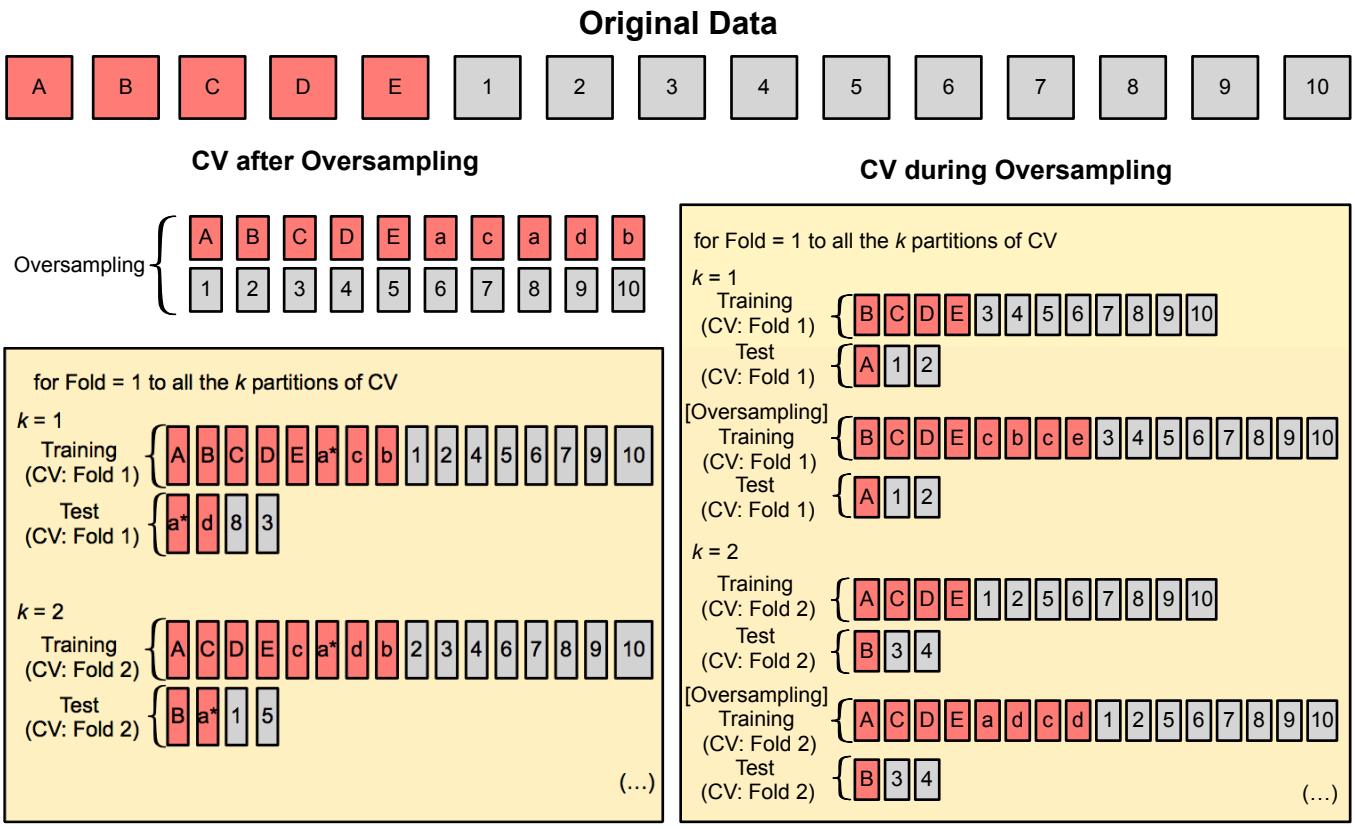

Fig. 1. Different cross-validation approaches: CV after oversampling (left) and CV during oversampling (right). When the cross-validation is implemented after oversampling is applied, similar patterns may appear in both training and test partitions (marked in the schema with an asterisk), leading to overoptimistic error estimates. When the cross-validation is applied during oversampling, only the training patterns are considered both for generating new patterns and training the model, avoiding overoptimism. In both approaches, similar or exact copies may appear in the training partitions, leading to overfitting, which is surpassed by an appropriate choice of oversampling technique.

- Overfitting occurs when the classifier is "tightly fitted" to the training data points, and therefore loses its generalization ability for the test data. Because of this, the classification performance is lower in the test set when compared to the training set. In this context, overfitting is usually associated to oversampling techniques that generate exact replicas of training data patterns (e.g. Random Oversampling - ROS), causing an overfit of the model in its learning stage.
- Overoptimism occurs when exact or similar replicas of a given pattern exist in both the training and test sets (well represented in Fig. 1 - CV after Oversampling). In this case, the classification performance in the test sets will be similar to the one obtained in the training sets, not because the model is able to correctly generalize to the test data, but rather because there are similar patterns in both training and test partitions. In this context, overoptimism is associated to incorrect implementations of cross-validation approaches, when oversampling is used.

As an example, consider that we divided a dataset into five equal folds. If we considered four partitions for training and applied the ROS algorithm, exact replicas of existing minority patterns would be generated: the classifier could be so exaggeratedly fitted to the training data that it would misclassify the test patterns (overfitting occurs). On the other hand, imagine that we considered all five partitions to perform oversampling, creating similar patterns rather than exact replicas. Although we are not using a technique prone to overfitting, we are considering all the data points in the oversampling procedure and therefore the probability that similar patterns are both in the training and test partitions increases (Fig. 1). In this case, we are in the presence of an overoptimistic approach.

The importance of a proper cross-validation approach in imbalanced domains was first emphasized by Blagus and Lusa [13]. They have evaluated the bias introduced in Classification and Regression Trees (CART) when cross-validation and sampling techniques – random undersampling, random oversampling and Synthetic Minority Oversampling Technique (SMOTE) – are jointly used. The results showed that incorrect CV achieved overly-optimistic estimates for random oversampling and SMOTE, while random undersampling produced accurate predictions, resilient to the change of CV procedure. Although this work provides an interesting take on the problem, some questions remained unanswered from the experimental setup. The number of real-world datasets used was rather small (10 datasets) and there was not much variability in terms of sample size. Therefore, although authors claimed that a higher bias (overoptimistic effects) was observed for smaller datasets, the lack of variability does not allow a complete analysis: in this work, we use a larger number of real-world datasets

(86 datasets) to provide a thorough evaluation of this topic. Blagus and Lusa [13] also refer that the bias is marginal when the prediction task is "easy", without supporting this claim with any type of complexity measures: we therefore explore well-established data complexity measures to characterize the difficulty of each dataset. Furthermore, the following novel analyses are included:

- Determine whether Imbalance Ratio (IR) influences the classification bias (overoptimistic effects);
- Evaluate incorrect versus correct CV approaches from a complexity perspective, by analysing the data complexity in training and test partitions;
- Analyse a higher number of oversampling algorithms, in order to compare their inner procedure, determine how they handle data complexity and assess which are more subjected to overfitting and which provide the highest classification improvement.

Motivated by the topics presented above, the purpose of this work is as follows:

(i) To fully characterize the risk of overoptimism when CV and oversampling algorithms are used, extending the work of Blagus and Lusa [13], as previously described;
(ii) To distinguish the problem of overoptimism from the overfitting problem, including a novel analysis on the risk of overfitting and on the influence of data complexity on classification results;
(iii) To study the behavior of 15 well-established oversampling algorithms and their influence on classification performance, providing a thorough analysis of their inner procedure.

In this way, the contribution of this research is two-fold. First, it details important aspects on how to properly address imbalanced data problems, so that researchers farther from the imbalance topic or new researchers in the field truly understand the nature of the problem and acknowledge the most correct validation procedures and promising resampling techniques. Secondly, for researchers familiarised with the imbalanced data field, it provides a thorough empirical analysis of a comprehensive set of oversampling techniques, focusing on their behavior/inner procedure and strengths/faults, supported by a data complexity analysis.

The structure of the manuscript is as follows: Section II presents some background knowledge on oversampling techniques, complexity measures and performance measures. Then, Section III presents recent works that make use of overoptimistic cross-validation procedures. The experimental setup is described in Section IV, while the experimental results are discussed in Section V. Finally, Section VI summarises the conclusions of the work and refers to some directions for future work.

II. Background Knowledge

This section reviews some background information that supports the different stages of this work, regarding oversampling algorithms, complexity metrics and performance metrics.

A. Oversampling Algorithms

1) ROS: Random Oversampling (ROS) is the simplest of oversampling techniques, where the existing minority examples are replicated until the class distribution is balanced. This approach is often criticised since it does not introduce any new information to the data (the oversampled examples are mere copies of the original data points) and may lead to overfitting (even if CV is performed properly) [14].

2) SMOTE: Synthetic Majority Oversampling Technique (SMOTE) works by generating synthetic minority examples along the line segments joining randomly chosen G minority examples and their k-nearest minority class neighbors [15]. G is the number of minority examples to oversample in order to obtain the desired balancing ratio between the classes, and along with the value of k, it can be specified by the user. SMOTE will then generate a new synthetic sample $\mathbf{s}$ according to $\mathbf{s} = \mathbf{x} + \varphi(\mathbf{x} - \mathbf{v})$, where $\mathbf{x}$ is the minority sample to oversample, $\mathbf{v}$ is one of its chosen nearest neighbors and φ is called a *gap*, in this case, a random number between 0 and 1. By generating similar examples to the existing minority points, SMOTE creates larger and less specific decision boundaries that increase the generalization capabilities of classifiers, therefore increasing their performance.

3) ADASYN: Instead of producing an equal number of synthetic minority instances for each minority example, the Adaptive Synthetic Sampling Approach (ADASYN) algorithm, proposed by He et al. [16], specifies that minority examples harder to learn are given a greater importance, being oversampled more often. ADASYN determines a weight (w_i) for each minority example, defined as the normalized ratio of majority examples N_i among its k nearest neighbors: $w_i = \frac{N_i}{k \times z}$ where z is a normalization constant. Then, the number of synthetic data points to generate for each minority example is specified as $g_i = w_i \times G$, being G the total necessary number of synthetic minority samples to produce according to the required amount of oversampling. The oversampling procedure is the same as SMOTE; the only difference is that harder minority examples are replicated more often.

4) Borderline-SMOTE: Based on the same idea of providing a more clear decision boundary, Han et al. [17] suggested two new variations of SMOTE – Borderline-SMOTE1 and Borderline-SMOTE2 – in which only the minority examples near the borderline are considered for oversampling. Borderline-SMOTE first considers the division of the minority examples into three mutually exclusive sets: noise, safe and danger. This division is made by considering the number of majority examples m' found among each minority example's k nearest neighbors. Thus being, if $m' = k$, all the nearest neighbors of a minority data point p_i are majority examples, and p_i is considered noise; conversely, if $\frac{k}{2} > m' \geq 0$, p_i is considered safe while if $k > m' \geq \frac{k}{2}$, p_i is surrounded by more majority examples than minority ones (or surrounded by exactly the same number), and therefore is considered danger. The "danger" data points are considered the minority borderline examples, and only them are oversampled, following a SMOTE-like procedure. For Borderline-SMOTE1 new synthetic examples are created along the line between

the danger examples and their minority nearest neighbors; Bordeline-SMOTE2 uses the same procedure as Borderline-SMOTE1, but further considers the nearest majority example of each danger data point to produce one more synthetic example: the distance between each danger point and its nearest majority neighbour is multiplied by a *gap* between 0 and 0.5 so that the new point falls closer to the minority class, thus strengthening the minority borderline examples.

5) Safe-Level-SMOTE: Contrary to Borderline-SMOTE, the technique proposed by Bunkhumpornpat et al. [18], called Safe-Level-SMOTE, only synthesizes minority examples around safe regions. To specify a safe region, a coefficient named *safe level ratio* (sl_{ratio}) is defined, which is the ratio between the number of minority examples found among each minority example's (p) k nearest neighbors, sl_p, and the number of minority examples found among a randomly chosen neighbor's (n) k-neighborhood, sl_n. Depending on the sl_{ratio} of a given minority example, five different scenarios may be applied to the SMOTE-based generation: if both sl_p and sl_n are 0, no oversampling occurs; if $sl_p > 0$ and $sl_n = 0$, then the SMOTE's *gap* is set to 0 (the minority example is duplicated); if $sl_{ratio} = 1$, the *gap* is as in the original formulation of SMOTE ($rand(0, 1)$); if $sl_{ratio} > 1$, the *gap* is set to $rand(0, \frac{1}{sl_{ratio}})$ so that the new example is generated closer to the minority example p and finally, if $sl_{ratio} < 1$, the *gap* is set to $rand(1 - sl_{ratio}, 1)$ so that, conversely, the new example is generated closer to the nearest neighbor n.

6) SMOTE+TL: SMOTE + Tomek Links (SMOTE+TL) also works on the basis of creating clear safe regions, by applying Tomek links after the data is oversampled with SMOTE [14]. A Tomek link is defined as a pair of examples from different classes, one from the minority class and the other from the majority class, (x_i, x_j), that are each other's closest neighbors [19]. In this technique, SMOTE is first applied to oversample the minority examples; then, the Tomek links are identified and both data points of each pair are removed.

7) SMOTE+ENN: Similar to SMOTE+TL, SMOTE+ENN first generates synthetic examples from the minority class (through SMOTE), from which a process of data cleaning follows, using the Wilson's Edited Nearest Neighbour Rule (ENN). ENN removes any example (either minority or majority examples) whose class differs from at least two of its three nearest neighbors [20]. By removing the examples that are misclassified by its three nearest neighbors, SMOTE+ENN provides a deeper data cleaning than SMOTE+TL [14].

8) ADOMS: Adjusting the Direction Of the synthetic Minority claSS examples (ADOMS) algorithm combines SMOTE with Principal Component Analysis (PCA) to produce new synthetic minority examples along the first principal component of the data surrounding each minority example [21]. For each minority example to replicate, ADOMS searches for its k-nearest minority class neighbors and performs PCA to determine the first principal component axis of the local data. The generation of the new example is done in a SMOTE-like fashion, but instead of being placed along the line that joins a minority example and one of its k nearest neighbors, it is placed along the first principal component axis of its k-neighborhood.

9) CBO: Jo and Japkowicz [22] propose an oversampling approach that simultaneously handles the between-class imbalance (imbalance between different classes) and the within-class imbalance, where a single class may comprise sub-clusters that hinder the learning process of algorithms. Their approach is called Cluster-Based Oversampling (CBO) and uses k-means clustering to guide the oversampling procedure. First, k-means is applied to each class to find the existing sub-clusters; then, the majority class is oversampled - each sub-cluster of the majority class is inflated until it reaches the size of the largest majority sub-cluster. Finally, the minority class is oversampled: each sub-cluster is oversampled until it reaches the size N_{maj}/N_{cmin}, where N_{maj} is total number of majority examples after oversampling and N_{cmin} is the number of minority class clusters. Different oversampling approaches may be coupled with CBO algorithm: this work makes use of the Random Oversampling (CBO + ROS), as proposed by Jo and Japkowicz in the original paper [22] and SMOTE (CBO + SMOTE), as discussed by He and Garcia [1].

10) AHC: Cohen et al. propose an oversampling approach based on Agglomerative Hierarchical Clustering (AHC) [23]. In this approach, the minority examples are clustered using AHC with both the single and complete linkage rules in succession, so that the produced clusters may vary. Then, fine-grained clusters are retrieved from all levels of the generated dendrograms and their centroids (prototypes) are determined. The process of synthetic data generation is based on introducing the computed cluster prototypes as new samples from the minority class, until a complete balance is achieved.

11) MWMOTE: Similarly to ADASYN and Borderline-SMOTE, the Majority Weighted Minority Oversampling Technique (MWMOTE) also works on the basis of generating synthetic samples in specific regions, where the minority examples are harder to learn [24]. MWMOTE starts by identifying the harder-to-learn minority examples (S_{imin}), so that each is given a selection weight (S_w), according to their distance to the nearest examples belonging to the majority class. These weights are then converted into selection probabilities, S_p, that will be used in the oversampling stage. To generate the new synthetic samples, the complete set of minority class examples S_{min} is clustered into M groups. Then, a minority example x from S_{imin} is selected according to the probability S_p, and another random minority example in S_{min} that belongs to the same cluster of x is used to generate a new synthetic sample in the same way as SMOTE. This approach is performed as many times as required, according to the necessary number N of synthetic samples to be generated for complete balance.

12) SPIDER: Stefanowski and Wilk propose an algorithm that uses the characteristics of examples to drive their oversampling: Selective Pre-Processing of Imbalance Data (SPIDER) [25]. SPIDER comprises two stages: first, each example is categorized into "safe" or "noisy", according to the correct or incorrect classification result returned by its k-neighborhood, respectively ($k = 3$ in the original formulation).

Then, an amplification strategy must be specified by the

user: either "weak amplification", "weak amplification with relabeling" or "strong amplification". If weak amplification is chosen, the noisy minority examples are amplified (copied) as many times as there are safe majority examples in their k-neighborhood ($k = 3$). "Weak amplification with relabeling" allies the amplification of noise minority examples described before with a relabeling procedure: noisy majority examples surrounded by noisy minority examples (considering $k = 3$), are relabeled to the minority class. The "strong amplification" technique processes both the noisy and safe minority examples. It starts by amplifying the safe examples by producing as many copies as there are safe majority examples in their 3-nearest neighborhood and then considers the noisy minority examples and reclassifies them according to a larger neighborhood ($k = 5$). If an example is correctly classified, it suffers a standard weak amplification; otherwise, it is more strongly amplified, by considering a 5-nearest neighborhood. Finally, for any type of amplification chosen, the noisy examples of the majority class are removed (in the case of "weak amplification with relabeling", only the un-relabelled noisy majority examples are removed).

SPIDER2 is a modification of SPIDER that performs the pre-processing of minority and majority examples in two separate stages [26]. It maintains the choice to perform a weak or strong amplification for the minority examples; while for the majority examples it is possible to decide whether relabeling is required or not. SPIDER2 starts by categorizing the majority examples into "safe" or "noisy" and if the relabeling option is chosen, the noisy majority examples are relabeled; otherwise, they are removed. Then, the minority examples are also divided into "safe" or "noisy" and the amplification proceeds according to the chosen technique (weak or strong), which are the same as above.

B. Data Complexity Measures

Ho and Basu [27] proposed several complexity measures that regard essentially three properties of datasets: geometry/topology, class overlapping and boundary separability (Table I).

TABLE I
COMPLEXITY MEASURES DESCRIPTION.

Measure	Description	Higher Data Complexity
F1	Highest value of Fisher's Discriminative Ratio (among all features)	$--$
F2	Highest volume of overlap between classes (among all features)	$++$
F3	Maximum feature efficiency (among all features)	$--$
L1	Minimised error of a linear classifier (linear SVM)	$++$
L2	Error rate (training set) of a linear classifier (linear SVM)	$++$
N1	Fraction of points on boundary by MST	$++$
N2	Ratio of average intra-class and inter-class scatter	$++$
N3	Error rate of nearest neighbour classifier (KNN, k=1)	$++$
L3	Nonlinearity of linear classifier (linear SVM)	$++$
N4	Nonlinearity of a nearest neighbour classifier (KNN, k=1)	$++$

1) Geometry and Topology: L3 and N4 measure the nonlinearity of a linear classifier and a nearest-neighbour classifier, respectively. L3 returns the error of a Support Vector Machine (SVM) [28] with linear kernel in a test set created by linear interpolation of randomly selected pairs of examples from the same class. N4 constructs a test set in the same way as for L3 and returns the test error for a nearest-neighbor classifier. Higher values of these measures indicate more complex classification problems.

2) Overlapping of Individual Feature Values: F1, F2 and F3 focus on the ability of a single feature to distinguish between classes [27]. F1 measures the highest discriminative power of all features in data (higher discriminative power indicates lower complexity), F2 measures the highest volume of overlap between the classes' conditional distributions (if there is no overlap in at least one feature, F2 will be zero), and F3 measures feature efficiency, the fraction of points where the values spanned by each class do not overlap (higher fractions indicate easier classification problems).

3) Class Separability: L1, L2, N1, N2 and N3 focus on the characteristics of the boundary between classes [29]. L1 and L2 measure to what extent the training data is linearly separable using an SVM with linear kernel [30]: if a classification problem is linearly separable, then L1 is zero and L2 is the training set error rate. N1 measures the fraction of points connected to the opposite class by an edge in a Minimum Spanning Tree (MST) and it can achieve high values when the classes are interspersed (higher complexity) or when the class boundary has a narrower margin than the intra-class distances (lower complexity). However, for the datasets used in this research, we observed that the first scenario is often the case, and for that reason we have associated higher values of N1 to a higher complexity in Table I. N2 measures the trade-off between the within-class spread and the between-class spread. In an easy classification problem, the within-class scatter should be low and the between-class scatter should be high; nevertheless, the denominator (between-class scatter) greatly influences N2 values: we, therefore, consider that higher values of N2 (smaller between-class scatter) traduce more complex scenarios. Finally, N3 measure is the error rate of a 1-nearest neighbor classifier (higher N3 values are associated to a higher complexity).

Additional information on the presented complexity measures is available in the extended version of the paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf).

C. Performance Metrics for Imbalanced Scenarios

Accuracy (ACC) measures the percentage of correctly classified examples and is computed as $ACC = \frac{TP+TN}{TP+FN+FP+TN}$, where TP and TN are the true positives and true negatives and FP and FN are the false positives and false negatives. Given that ACC is biased towards the majority class [28], alternative metrics should be considered, such as Sensitivity, Specificity, Precision, F-Measure, G-mean and the Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) [1]. Sensitivity (SENS) is calculated as $SENS = \frac{TP}{TP+FN}$ and measures the percentage of positive

examples correctly classified, while Specificity (SPEC) refers to the percentage of negative examples correctly identified and can be computed as $SPEC = \frac{TN}{TN+FP}$. Precision (PREC) corresponds to the percentage of positive examples correctly classified, considering the set of all the examples classified as positive, $PREC = \frac{TP}{TP+FP}$. F-measure, G-mean and AUC represent the trade-off between some of the metrics described above. F-measure (F-1) shows the compromise between sensitivity and precision, obtained through their harmonic mean, $F\text{-}1 = \frac{2 \times PREC \times SENS}{PREC+SENS}$ while G-mean represents the geometric mean of both classes' accuracies, $G\text{-}mean = \sqrt{SENS \times SPEC}$. At last, AUC makes use of the ROC curve to exhibit the trade-off between the classifier's TP and FP rates [31].

III. Related Works

This section presents a series of related works aiming to show that the less the work is related to learning from imbalanced data, the more likely the cross-validation (CV) procedure is poorly designed. Thus, related works were divided into three main categories: "Learning from imbalanced data", "Comparing approaches in a specific context" and "Solving a classification problem". The "Learning" category includes research works focused on performing extensive experiments to evaluate diverse sampling techniques [32]–[38]. Typically, these works include a large number of publicly available datasets and a comprehensive set of learners and sampling algorithms. "Comparison" category works perform a comparison of oversampling approaches in a specific context: these works normally include a lower number of datasets and sampling strategies. "Classification" category comprises works where the main objective is to solve a particular classification problem and the imbalanced nature of data is not the focus. The extended version of this paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf) provides additional information on related works, including a table that summarises their main characteristics. All works included in the "Learning" category, except one, perform a well-designed CV procedure, where the training and test partitions are determined before any oversampling technique is applied. As we move towards research works whose objective is not to provide a general review on ways to deal with the data imbalance problem, we find a larger number of works where the CV procedure is not appropriate: the complete dataset is oversampled and the partition into training and test is performed afterwards. This is more evident if we consider the research works where the main objective is to ease a classification task, rather than studying different approaches to surpass the issues of imbalanced datasets. It is possible that these researchers were not completely familiarised with imbalanced data domains and respective approaches; thus, when faced with a specific imbalanced context, they resorted to the state-of-the-art oversampling approach (namely, SMOTE) to solve the issue, but they understood it as a form of preprocessing, which created a greater propensity for misconception during its application. We therefore conclude that the less the work is related to learning from imbalanced learning, the more likely the CV procedure is poorly designed.

IV. Experimental Setup

The experimental setup used in this work comprises three main approaches: Baseline, Approach 1 and Approach 2 (Fig. 2). For the results presented as "Baseline", the collected

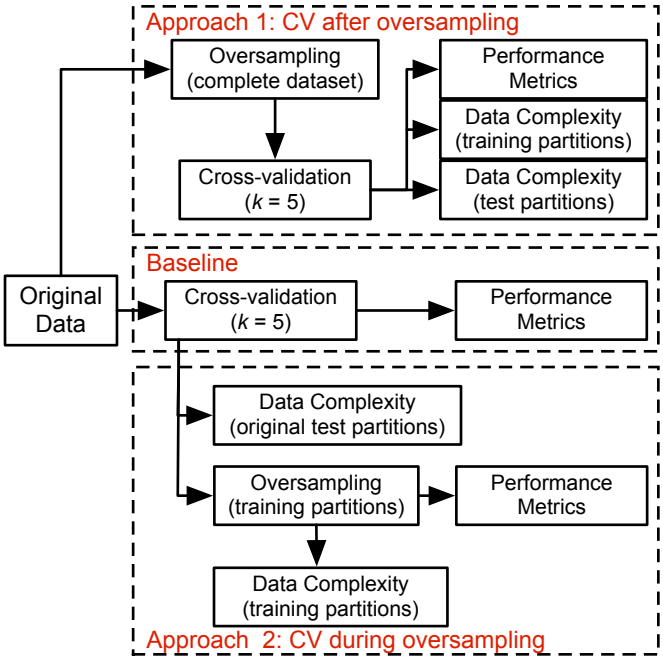

Fig. 2. Experimental setup architecture.

datasets are first divided into five stratified folds ($k = 5$ folds is the maximum that allowed a proper stratification) and the classifiers are applied afterwards, without any type of oversampling. The data complexity measures and performance metrics for the original training and test sets are then retrieved. In Approach 1, the original datasets are oversampled and the CV and performance evaluation are performed afterwards. The data complexity measures (for oversampled training and test sets) are then retrieved. In Approach 2, oversampling is performed during CV: the original datasets are first divided into five folds (same folds as for the Baseline), and only the training partitions are oversampled. The classifiers are then trained with the oversampled training folds and tested in the respective original test folds. In this case, the data complexity measures are only determined for the oversampled training sets, since the data complexity of the test sets is the same as obtained from the Baseline method.

With this setup we aim to perform 3 main analysis: (i) compare the differences in classification performance between Approaches 1 and 2, in order to explain the risk of overoptimistic error estimates, (ii) distinguish between overoptimistic and overfitting approaches in imbalanced scenarios that consider oversampling and (iii) determine which oversampling approach is the most appropriate to solve the imbalance problem, obtaining the best average results for all the different contexts (datasets) considered in this study.

Regarding the process of data collection, the 86 datasets used in this work were collected from two online repositories, UCI Machine Learning Repository (http://archive.ics.uci.edu/

ml) and KEEL – Knowledge Extraction based on Evolutionary Learning (http://www.keel.es). The choice criteria included the following parameters: complete datasets, regarding binary classification problems, with a variable sample size, number of features and IR. Their main characteristics are summarised in Table II. As a note worth mentioning, although the variability of datasets considered in this work is considerable, especially in comparison to the precursor work of Blagus and Lusa [13], we would like to point out that the conclusions drawn in this research refer to datasets with low dimensionality (4-34 features). The interested readers may find additional information on the data collection stage in the extended version of this paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf).

V. Results and Discussion

In this section, we start by comparing Approaches 1 and 2 regarding their risk of overoptimism/overfitting. Then, we move to an analysis on data complexity and the inner characteristics of each oversampling method.

A. Approach 1 versus Approach 2: Evaluating the risk of overoptimism and overfitting

To evaluate the issues of overoptimism and overfitting regarding the joint-use of CV and oversampling approaches, we started by comparing the performance results of Approach 1 and Approach 2 (please refer to the extended version of this paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf). The results confirmed that the performance outputted by Approach 1 is more optimistic: the mean test values of the various performance metrics (AUC, G-mean, F-1 and SENS) was always higher in Approach 1 (see extended version). Since the behavior observed for both Approaches is consistent for all performance metrics, we will refer only to the AUC values in the following analyses, in order to provide a base of comparison with previous works, which largely use AUC. Fig. 3 shows the AUC values (training and test partitions) obtained for the original datasets (Baseline) and for the oversampled datasets, considering both Approaches 1 and 2. Furthermore, Table III presents the absolute differences between AUC values of training and test partitions, for both Approaches (listed in descending order of differences in Approach 2). The p-values derived from a Mann-Whitney test are also included and confirm that the train-test differences between Approaches 1 and 2 are significantly different. In the extended version of this paper, the interested reader may find additional statistical tests for training and test partitions individually, and a table of 10 representative datasets for which the classification results are discussed in detail.

As shown in Fig. 3, the training results are similar, which suggests that the major difference between both approaches relies on the characteristics of the test sets. In Approach 1, it is the overoptimism problem (rather than the overfitting) that is identified, given that the difference between training and test results is not considerable (Table III). In this scenario, the test sets have similar characteristics to the training sets (are balanced and may contain exact replicas or similar patterns to the training points). From Table III, it can be observed that for Approach 1, the best methods often include CBO and ROS. CBO+Random and ROS create exact replicas of existing data points, and since the division (i.e., CV) is performed after the oversampling procedure in the entire dataset, the probability that exact replicas exist in both the training and test sets increases, thus producing better results. In the case of CBO+SMOTE, although SMOTE creates synthetic examples, it does so by inflating the clusters defined by k-means algorithm, which may reduce the data variability introduced in the dataset. Therefore, patterns in the test sets may also be similar to the ones comprised in the training sets.

In Approach 2, the difference between the results of the training and test sets is more accentuated: in this scenario, the test sets follow the same distribution as the original dataset and its patterns are never considered in the oversampling or training phases. As a result, overoptimism does not appear in this scenario. However, some overfitting effects may occur. Considering the presence of overfitting as a difference around 0.1 between the training and test AUCs [39], it can be observed that the great majority of oversampling methods cannot be responsible for overfitting effects. However, some methods seem to be introducing overfitting (Table III). CBO+Random, which obtains the worst results, seems to be the method responsible for the highest amount of overfitting, followed by Borderline-SMOTE and CBO+SMOTE. ROS, SMOTE+ENN and Safe-Level-SMOTE, although in a lighter scale, also seem to have some generalization issues, where the difference between training and test AUCs also comes close to 0.1. The same cannot be observed for Approach 1 given that the overoptimism problem highly dominates the results, preventing the identification of these overfitting effects. The tendency of CBO+Random, CBO+SMOTE and ROS to overfit the data is somewhat intuitive: as they create exact replicas (CBO+Random and ROS) or very similar replicas (CBO+SMOTE by creating synthetic examples in defined clusters) to the existing training patterns, the models tend to overfit these training patterns and fail to generalize to different ones. Further on, Section V-C performs a detailed analysis of Approach 2, reviewing the advantages and disadvantages of each oversampling algorithm, allowing the understanding of why Borderline-SMOTE and Safe-Level-SMOTE may present generalization issues (which also explains their poor performance). The major issue of these methods is that the definition of danger/borderline examples (Borderline-SMOTE) and safe examples (Safe-Level-SMOTE) may fail in certain scenarios and prejudice the classification task (inability to generalize). Finally, SMOTE+ENN shows a training/test difference of 0.096, which is considerable when compared to its analogous SMOTE+TL (0.091) and precursor SMOTE (0.087). Both methods (SMOTE+ENN and SMOTE+TL) were developed to surpass the issues of overgeneralization of SMOTE. However, as will be discussed further on (Section V-C), the ability of SMOTE to create larger decision boundaries seems to be a major strength, whereas its successor procedures seem to create a higher risk of overfitting the training data. This may be due to excessive cleaning applied after SMOTE. In the case of SMOTE+TL, the issue is not critical (0.091), as only the Tomek Links are removed. For SMOTE+ENN, the

TABLE II
CHARACTERISTICS OF IMBALANCED DATASETS.

Dataset	Size	Features	IR	*Dataset*	Size	Features	IR
bupa	345	6	1.38	vowel0	988	13	9.98
pageblocks-1-3vs4	472	10	1.57	ecoli-0-6-7vs5	220	7	10.00
glass1	214	9	1.82	glass-0-1-6vs2	192	9	10.29
ecoli-0vs1	220	7	1.86	ecoli-0-1-4-7vs2-3-5-6	336	7	10.59
wisconsin	683	9	1.86	led7digit-0-2-4-5-6-7-8-9vs1	443	7	10.97
pima	768	8	1.87	ecoli-0-1vs5	240	7	11.00
cmc1vs2	961	9	1.89	glass-0-6vs5	108	9	11.00
iris0	150	4	2.00	glass-0-1-4-6vs2	205	9	11.06
glass0	214	9	2.06	glass2	214	9	11.59
german	1000	20	2.33	ecoli-0-1-4-7vs5-6	332	7	12.28
yeast1	1484	8	2.46	cleveland-0vs4	173	14	12.31
haberman	306	3	2.78	ecoli-0-1-4-6vs5	280	7	13.00
vehicle2	846	18	2.88	shuttle-c0-vs-c4	1829	9	13.87
vehicle1	846	18	2.90	yeast-1vs7	459	8	14.30
vehicle3	846	18	2.99	glass4	214	9	15.46
glass-0-1-2-3vs4-5-6	214	9	3.20	ecoli4	336	7	15.8
transfusion	748	4	3.20	abalone9-18	731	8	16.4
vehicle0	846	18	3.25	dermatology-6	358	34	16.9
ecoli1	336	7	3.36	thyroid-3vs2	703	21	18.00
newthyroid1	215	5	5.14	glass-0-1-6vs5	184	9	19.44
ecoli2	336	7	5.46	pageblocks-1vs3-4-5	5144	10	21.27
balance_scaleBvsR	337	4	5.88	shuttle-6vs2-3	230	9	22.00
balance_scaleBvsL	337	4	5.88	yeast-1-4-5-8vs7	693	8	22.10
segment0	2308	19	6.02	pageblocks-1-2vs3-4-5	5473	10	22.69
glass6	214	9	6.38	glass5	214	9	22.78
yeast3	1484	8	8.10	yeast-2vs8	482	8	23.10
ecoli3	336	7	8.60	letter-U	20000	16	23.60
pageblocks0	5472	10	8.79	flare-F	1066	11	23.79
ecoli-0-3-4vs5	200	7	9.00	car-good	1728	6	24.04
yeast-2vs4	514	8	9.08	pageblocks-1vs4-5	5116	10	24.20
ecoli-0-6-7vs3-5	222	7	9.09	car-vgood	1728	6	25.58
ecoli-0-2-3-4vs5	202	7	9.10	letter-Z	20000	16	26.25
glass-0-1-5vs2	172	9	9.12	kr-vs-k-zero-onevsdraw	2901	6	26.63
yeast-0-3-5-9vs7-8	506	8	9.12	yeast4	1484	8	28.10
yeast-0-2-5-6vs3-7-8-9	1004	8	9.14	winequality-red-4	1599	11	29.17
yeast-0-2-5-7-9vs3-6-8	1004	8	9.14	poker-9vs7	244	10	29.50
ecoli-0-4-6vs5	203	7	9.15	yeast-1-2-8-9vs7	947	8	30.57
ecoli-0-1vs2-3-5	244	7	9.17	abalone-3vs11	502	8	32.47
ecoli-0-2-6-7vs3-5	224	7	9.18	yeast5	1484	8	32.73
glass-0-4vs5	92	9	9.22	kr-vs-k-threevseleven	2935	6	35.23
ecoli-0-3-4-6vs5	205	7	9.25	winequality-red-8vs6	656	11	35.44
ecoli-0-3-4-7vs5-6	257	7	9.28	abalone-17vs7-8-9-10	2338	8	39.31
yeast-0-5-6-7-9vs4	528	8	9.35	abalone-21vs8	581	8	40.50

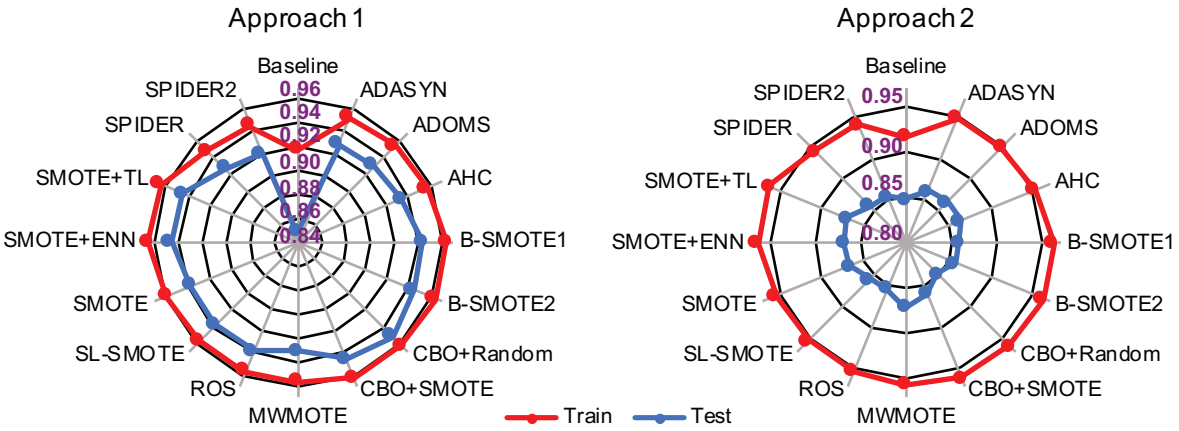

Fig. 3. Performance metrics (average) achieved for the original datasets (Baseline) and for the oversampled datasets, considering both Approaches 1 and 2.

issue is aggravated (0.096) due to its deeper data-cleaning procedure. Such cleaning aims to simplify the training data and ease the definition of less complex class boundaries, although the results suggest that this may not be advantageous for all scenarios: such simplification may jeopardize generalization. Focusing on the test AUC results, MWMOTE and SMOTE+TL seem to be the best oversampling methods (Fig. 3).

TABLE III
AUC DIFFERENCES BETWEEN TRAINING AND TEST PARTITIONS.

	Train-Test Difference		Man-Whitney
Algorithm	**Approach 1**	**Approach 2**	***p*-value**
CBO+Random	0.011	0.112	9.26E-23
Borderline-SMOTE2	0.019	0.104	8.34E-18
Borderline-SMOTE1	0.020	0.104	1.31E-17
CBO+SMOTE	0.016	0.099	1.18E-17
ROS	0.018	0.097	2.17E-17
SMOTE+ENN	0.019	0.096	6.07E-16
Safe-Level-SMOTE	0.019	0.095	2.54E-16
SMOTE+TL	0.020	0.091	1.32E-14
AHC	0.023	0.089	1.36E-12
SPIDER	0.022	0.088	3.18E-17
SMOTE	0.023	0.087	2.83E-41
ADASYN	0.024	0.086	4.53E-12
ADOMS	0.024	0.085	7.92E-12
SPIDER2	0.025	0.084	6.93E-08
MWMOTE	0.025	0.084	2.69E-12
Baseline	0.069	0.069	9.94E-01

B. Data Complexity Analysis

In order to better support the existence of overoptimism in Approach 1 (CV after oversampling), we have investigated the complexity of the training and test partitions for all datasets, in both Approaches 1 and 2. We hypothesize that the overoptimism is related to the difference between training and test partitions as explained in what follows. When oversampling is applied before CV, the test and training partitions will have a similar structure, and therefore their complexity is similar - the classification is more straightforward, given that the algorithm learns from similar contexts. When oversampling is performed during CV, the test and training partitions, as previously explained, have a different structure, which hinders the classification task.

Fig. 4 shows the difference (in module) between the complexity of the training and test partitions, on average, for each approach. This is performed for all oversampling algorithms, and the differences in complexity are also related to the mean test AUCs for each algorithm. For the original (Baseline) partitions, the AUC values and differences in complexity are the same for both approaches.

From Fig. 4, it can be observed that the results are consistent with our reasoning: the difference in complexity in Approach 2 is higher than for Approach 1. In some cases, algorithms SPIDER and SPIDER2 show an antagonistic behavior to the other methods, which may be due to their process of generating new data (that differs from the remaining algorithms). In the implementation used in this work, SPIDER uses a weak amplification strategy, where the minority class examples are replicated according to the existence of majority data points marked as "safe" among their k nearest neighbors. Given a complex dataset, where there are only a few "safe" examples, the minority examples are never oversampled. For SPIDER2, we have used a strong amplification strategy with relabeling, where the neighborhood to be considered is extended to $k+2$, and the class of the original majority examples marked as "noisy" is directly changed. Additionally, SPIDER and SPIDER2 are the only methods that do not guarantee an equal class distribution, i.e., it is not guaranteed that the resulting dataset, after oversampling, is balanced. These differences from the other methods could be the origin of their erratic behavior regarding both the results of the classification and complexity measures. The intrinsic characteristics of each oversampling algorithm will be further discussed in Section V-C.

We continue this section by addressing the questions raised by Blagus and Lusa [13] that were not fully answered in their experimental setup (please check Section I). Thus being, we analysed the mean test AUC results for ROS and SMOTE methods (the two oversampling methods used by Blagus and Lusa [13]), for all datasets ordered by their sample size and IR: from the simulation results, no relation was found with either one. For this reason, and due to space restrictions, this analysis is not included herein, but it is fully detailed in the extended version of the paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf). In terms of complexity, we have chosen to present the F1 metric (Fig. 5). The results using other complexity measures followed the same tendency, yet F1 seems the most straightforward to understand: it measures the highest discriminative power considering all the features in the dataset – if at least one feature has a high discriminative capability (its values allow to distinguish between classes), then the classification task is "easy". Fig. 5 shows that the complexity of the classification task is what most influences the overoptimistic behavior of poorly designed CV procedures: the less complex the classification task is, the smaller is the difference between the CV setups (Approach 1 and Approach 2). Indeed, when the classification task is easier, the decision boundary is clearer and Approach 2 achieves higher classification results. Thus the difference between both approaches is not so discrepant.

We have further conducted a regression and clustering analysis based on all the complexity measures obtained from the training data. For the regression analysis, we obtained a regression model that could accurately predict the test AUC based solely on the complexity measure of the corresponding training partitions (R^2 of 0.72), where the highest values of the coefficient of determination were obtained for SMOTE+TL (0.807), MWMOTE (0.798) and SMOTE+ENN (0.795). The clustering analysis (using k-means clustering) produced a division where the top 70 datasets with the best test AUC results are grouped, the majority are produced with MWMOTE, SMOTE+TL and SMOTE+ENN. These results are detailed in the extended version of the paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf).

C. Analysis of oversampling algorithms: Approach 2

After determining the most suitable CV scheme in imbalanced scenarios (CV during oversampling - Approach 2), we focus on analyzing the most appropriate oversampling methods for imbalanced contexts. To that end, three different strategies were considered. In the first strategy, we analyze the average test AUC values including all classifiers (Strategy 1). In the second strategy, we rank the AUC values by the oversampling technique, for each classifier. Then, the average rank is computed for each oversampling technique (Strategy 2). Finally, the third strategy considers the ranking of AUC values by oversampling technique, for each classifier and dataset. Then,

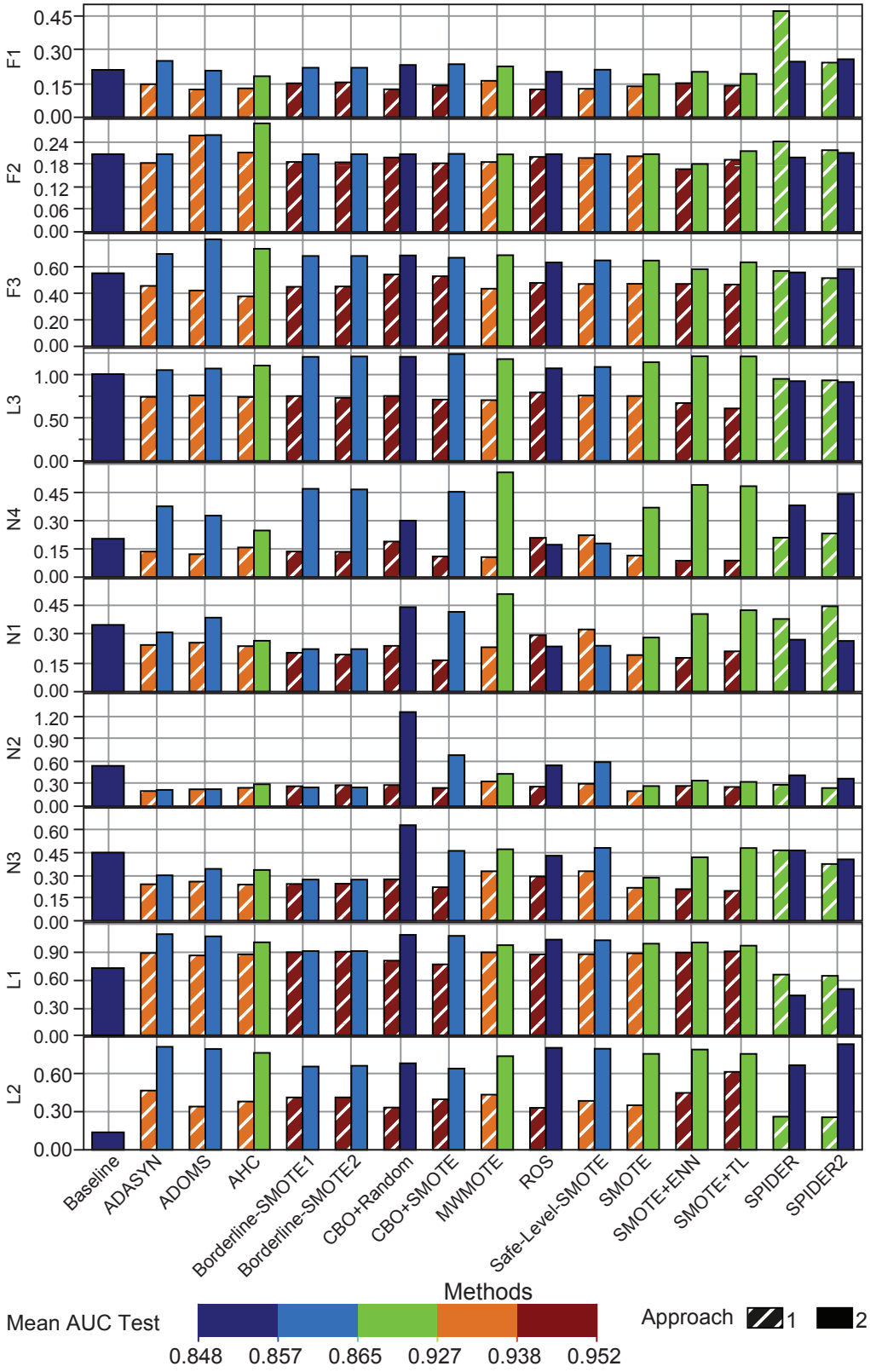

Fig. 4. Differences (in module) between the complexity measures for all oversampling techniques, considering both Approaches 1 and 2.

the average rank is computed for each oversampling technique (Strategy 3). The results of each strategy are summarised in Table IV.

Table IV shows that all the implemented techniques are

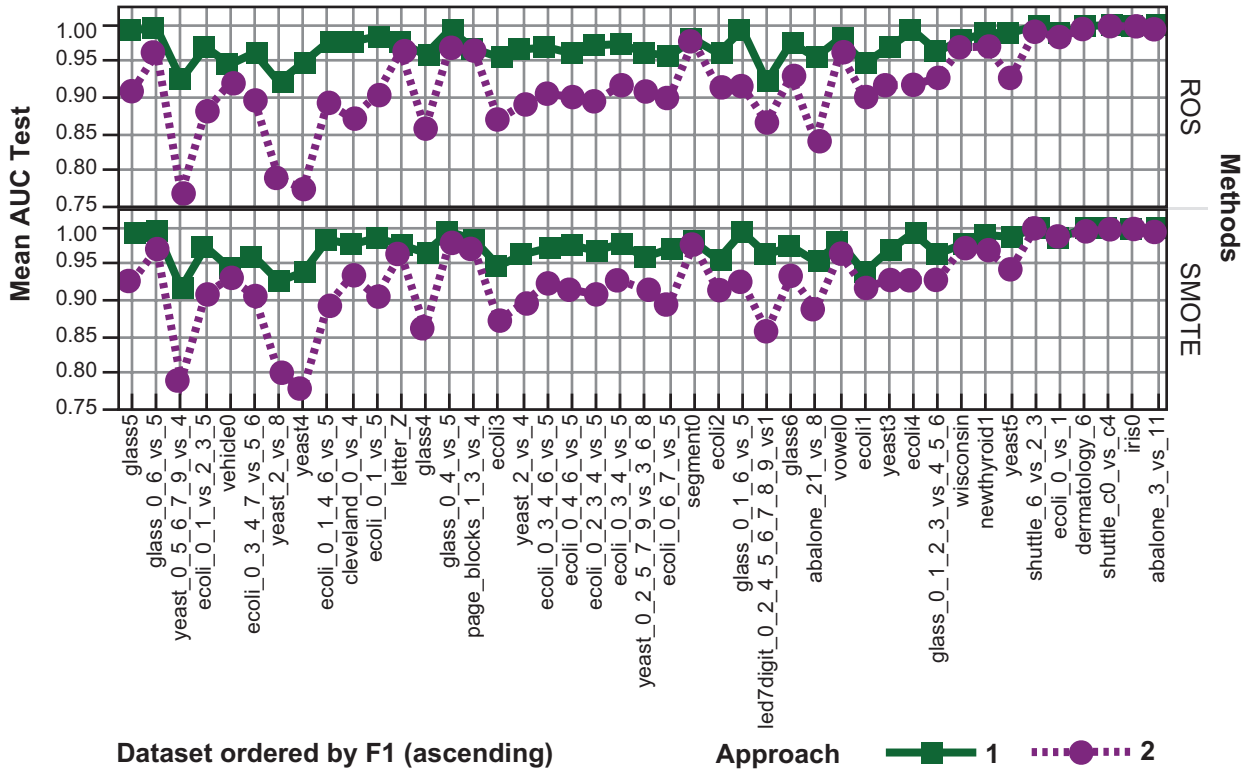

Fig. 5. Differences between test AUCs of Approach 1 and Approach 2: datasets are ordered by their original F1 complexity measure (highest discriminative power among all features). Due to space restrictions, only the datasets with highest F1 values are represented, although a complete analysis is performed in the extended version of this paper.

TABLE IV
OVERSAMPLING METHODS (PLUS BASELINE) ORDERED BY PERFORMANCE, ACCORDING TO EACH TESTED STRATEGY.

	Strategy		
Rank	**1**	**2**	**3**
1st	SMOTE+TL (0.871±0.052)	SMOTE (3.000±1.265)	SMOTE+TL (6.535±4.094)
2nd	MWMOTE (0.871±0.053)	SMOTE+TL (3.167±1.941)	MWMOTE (7.199±4.332)
3rd	SMOTE (0.868±0.054)	MWMOTE (4.333±4.844)	SMOTE+ENN (7.201±4.072)
4th	SMOTE+ENN (0.867±0.054)	SMOTE+ENN (5.000±2.828)	SMOTE (7.222±3.460)
5th	AHC (0.865±0.055)	AHC (6.000±2.280)	ADOMS (7.606±4.195)
6th	ADOMS (0.864±0.057)	ADOMS (7.000±2.000)	AHC (8.088±3.817)
7th	ADASYN (0.862±0.059)	ADASYN (8.000±2.098)	ADASYN (8.215±4.223)
8th	CBO+SMOTE (0.860±0.058)	SL-SMOTE (8.000±6.753)	SL-SMOTE (8.411±4.075)
9th	B-SMOTE1 (0.858±0.060)	CBO+SMOTE (9.333±5.317)	B-SMOTE1 (8.743±4.119)
10th	B-SMOTE2 (0.858±0.060)	ROS (9.833±5.076)	B-SMOTE2 (8.743±4.119)
11th	SL-SMOTE (0.857±0.061)	B-SMOTE1 (10.667±1.966)	CBO+SMOTE (8.745±4.475)
12th	SPIDER (0.856±0.059)	B-SMOTE2 (10.667±1.966)	ROS (9.019±4.034)
13th	ROS (0.855±0.063)	SPIDER (11.000±1.673)	SPIDER (9.412±4.665)
14th	SPIDER2 (0.855±0.059)	SPIDER2 (11.833±2.137)	SPIDER2 (9.569±4.764)
15th	CBO+Random (0.849±0.063)	CBO+Random (13.500±2.811)	CBO+Random (9.821±4.419)
16th	Baseline (0.848±0.066)	Baseline (13.667±3.830)	Baseline (11.471±4.981)

B and SL are equivalent to Borderline and Safe-Level, respectively.

better than using the original dataset without any type of processing (Baseline). Also, all considered strategies output the same set of winners, SMOTE+TL, SMOTE+ENN, MWMOTE and SMOTE, although their ranks may vary. SMOTE+TL, followed by MWMOTE, are considered the best oversampling methods. The same is true for the worst oversampling techniques, where CBO+Random, SPIDER, SPIDER2 and ROS are found on the bottom positions. In light of these results, we herein provide a detailed discussion on the intrinsic characteristics and behavior of the different oversampling methods used. We compare each method in what concerns their inner procedure and how they are able to address the datasets' complexity and improve the classification results, also highlighting their main advantages and disadvantages.

1) ROS and SMOTE: ROS is the simplest of the oversampling techniques: a random subset of minority examples is replicated until the desired balance is reached. Nevertheless, this technique is subjected to overfitting due to the replication (creation of exact copies) of minority examples. The fact that ROS creates exact copies of existing examples, leads to a generation of very similar partitions in Approach 1 (and consequent overoptimism), while in Approach 2, as explained in Fig. 3, ROS is mostly subjected to overfitting. This is also

supported by Fig. 4, where ROS is among the best methods in Approach 1 (between 0.938 and 0.952), while for Approach 2 it provides the worst AUC results (between 0.848 and 0.857).

SMOTE smooths the problem of creating exact copies of existing minority examples by creating synthetic minority instances using the k-neighborhood of minority examples. However, the minority class is augmented without considering the structure of data: all minority examples have the same probability of being oversampled, regardless of their neighborhood, which leads to the following issues [7], [24], [40]:

- By considering a neighborhood composed only of minority examples, the new synthetic examples may be generated in overlapping areas (problem of overgeneralization);
- Since no distinction between minority examples is performed (e.g. by evaluating their majority neighborhood), SMOTE-like methods can also augment noise regions, by oversampling noisy examples (minority examples surrounded by majority examples, that are most likely noise).

Nevertheless, it seems that the ability of SMOTE to generate larger decision boundaries is still a major strength, even with its susceptibilities. In fact, SMOTE is found among the best oversampling methods, as shown in Fig. 4, which justifies why it is a renowned oversampling method, widely used in several research areas [40], [41].

2) SMOTE+TL and SMOTE+ENN: SMOTE+TL and SMOTE+ENN combine oversampling with a cleaning procedure that alleviates SMOTE's problem of overgeneralization: they are able to remove examples that lie on overlapping regions (as detailed in Section II-A). However, since SMOTE is applied prior to the cleaning procedure, some of the same issues from SMOTE remain:

- All minority examples have the same probability of being oversampled, causing some unnecessary ("safe") examples to be oversampled;
- Noise minority regions could be augmented and remain after the cleaning procedure: after oversampling, they may not be identified by Tomek Links or ENN as examples to remove, since their neighborhood is changed.

Nevertheless, what is true for noise regions, is also true for small disjuncts. SMOTE+TL and SMOTE+ENN, by applying SMOTE as a first step, may be inflating unnecessary noise regions, but may also be inflating important, underrepresented, minority points. Overall, our results show that combining SMOTE with these cleaning methods turns out to be a superior approach than most (Table IV): SMOTE creates larger and less specific decision boundaries, that are afterwards simplified by Tomek Links and ENN by removing several borderline examples while also alleviating the issue of small disjuncts. However, some caution must be taken regarding the cleaning procedure: as discussed from Table III, for some datasets, an excessive cleaning may be the cause of overfitting.

3) CBO+Random and CBO+SMOTE: CBO was first thought as a way of handling both the between-class imbalance as well as the within-class imbalance (small disjuncts). CBO is able to attend to the structure of data by performing clustering on both classes individually (both minority and majority example are oversampled). Nevertheless, CBO needs a procedure for the generation of new examples, and each has its hitches:

- CBO+Random is more prone to overfitting: since random oversampling is performed within clusters, the probability that similar instances are oversampled more often is even greater than for ROS alone, as discussed in Fig. 3 and Table III;
- CBO+SMOTE eases the problem of overgeneralization given that SMOTE is performed within clusters; however, it no longer takes advantage of SMOTE's ability to create larger decision regions, which explains why its performance is considerably lower than SMOTE's (Table IV): applying SMOTE within clusters increases the probability that similar instances are generated, which can also result in overfitting, as discussed from Table III.

Finally, for both techniques, the definition of the most appropriate number of clusters is a problem. In this work, to find the optimal k number of clusters for each class, we have used three evaluation criteria: Calinski-Harabasz [42], Davies-Bouldin [43] and Silhouette [44], and a range of $k = 2, ..., 20$. Our CBO computes for each criterion five times and extracts the mode of these five runs. Finally, after determining the optimal k according to each criterion, the mode is computed again to obtain the final optimal k for a given class.

4) Borderline-SMOTE and ADASYN: Defining a taxonomy of minority examples (noise, safe and danger) allows Borderline-SMOTE to operate only on the examples of interest: the synthetic minority examples will be created in a SMOTE-like fashion, along the line that joins each danger example to its k nearest minority neighbors, thus strengthening the borderline examples. Nevertheless, as Borderline-SMOTE uses the same procedure as SMOTE to oversample minority examples, it may suffer from the same issues mentioned above. Additionally, another problem with Borderline-SMOTE technique is in the way danger/borderline examples are identified (see Section II-A): in some contexts, the $k > m' \geq \frac{k}{2}$ criterion may fail, and in those cases there is no oversampling in important regions near the decision boundary, which will prejudice the classification task [24], as discussed in Table III. We assume that this issue may affect some of the datasets in our study, since that, although Borderline-SMOTE aims to provide a more clear decision boundary, it does not figure among the best approaches (Table IV).

ADASYN considers the majority neighborhood of the minority examples to guide the oversampling procedure: the minority examples are assigned different weights according to the number of majority examples in their neighborhood. Adaptively assigning weights to the minority examples is a way to smooth the above-mentioned issues of Borderline-SMOTE; however, the definition of parameters for weight assignment may be inappropriate to correctly distinguish the importance of minority examples for classification. As mentioned in Section II-A, the weight of each minority example is proportional to the number of majority examples in its k-neighborhood, which causes two main issues [24]:

- ADASYN may oversample unnecessary noisy examples: noisy examples are typically surrounded by the majority

class, and therefore their weight will be high;
- ADASYN may fail to oversample important minority examples close to the decision boundary, which is the most important concept to learn, if all their k-nearest neighbors are from the minority class.

Considering different weights for different minority examples is a way of defining the structure of minority data (although ignoring the structure of majority data). If additionally, the criterion to define those weights fails for some datasets, ADASYN loses its main advantage. This is consistent with the results provided in Table IV, where ADASYN is found in the 7th position, slightly above the middle of the table, although far from the top winners.

5) Safe-Level-SMOTE and ADOMS: Safe-Level-SMOTE also considers a weighted scheme to oversample the minority examples in safe regions. The weight assignment is more sophisticated than ADASYN's, since that rather than looking only to the majority neighborhood of each minority example, Safe-Level-SMOTE also considers the structure of minority data points: the weights defined by sl_{ratio} allow Safe-Level-SMOTE to place new instances near those considering "safer", easing the problem of small disjuncts while avoiding the augmentation of noise regions. However, for specific scenarios, Safe-Level-SMOTE may generate inconsistent examples [7]: if a minority example is an outlier, inside a well-defined majority cluster, then its sl_{ratio} will be 0, causing the *gap* for SMOTE synthetization to be 1, thus creating a new minority instance in the exact location of a majority point. This may explain its susceptibility to overfitting (Table III) and its poor performance (Table IV).

Rather than placing synthetic examples along the line between a minority example and one of its k minority neighbors (as SMOTE), ADOMS considers the local minority distribution along the example to oversample, through the computation of the first principal component of the defined k-neighborhood (Section II-A). Therefore, ADOMS takes advantage of SMOTE's ability to define larger decision regions, while considering the local structure. However, ADOMS seems to fall behind SMOTE in the three considered strategies from Table IV: we hypothesize that some of the instances placed by ADOMS create more overlapping than SMOTE's: SMOTE generates instances along a line joining two minority instances, yet ADOMS may place its instances in sparser projections [21]. Since, as in SMOTE, the distribution of majority examples is not considered, the generation procedure might not be appropriate for all scenarios.

6) SPIDER and SPIDER2: SPIDER combines the local-oversampling of noisy, difficult, minority examples with a cleaning procedure that removes (or relabels) noisy majority examples. The original SPIDER algorithm processed both minority and majority examples at the same time, sometimes severely modifying the majority class. To address this issue, a new version was proposed, SPIDER2, that alleviates the degradation of the minority class by processing minority and majority examples separately. The major issues of these methods are as follows:

- The process that leads to the amplification of minority examples does not distinguish between borderline or noisy examples (if they are "not-safe", they are all considered "noise"). Therefore, these "unsafe" minority examples are all given the same importance to classification: SPIDER/SPIDER2 can either be oversampling difficult examples so that they are not misclassified, but at the same time, they can be augmenting undesired noise regions;
- Both methods perform replication of examples rather than synthetization, which adds no new information;
- When "relabeling" is chosen, SPIDER/SPIDER2 perform an oversampling procedure similar to SMOTE, except that instead of generating new instances in the neighborhood of minority examples, it relabels their majority neighbors: however, relabeling examples might not be an appropriate approach in some domains.

Although SPIDER and SPIDER2 aim to define a taxonomy of minority examples, they do not distinguish between two important minority concepts, "borderline" and "noisy", addressing them as equals. Also, the fact that these methods consider replication of existing examples rather than the synthetization of new ones, is surely responsible for their lower positions on Table IV, along similar methods with the same inner procedure (CBO+Random and ROS). Finally, they are the only methods for which it is not possible to establish the amount of oversampling, which, in this work, has been established to accomplish a perfect balance in the training sets (50%-50% distribution). Since the remaining methods were optimised to achieve perfect balance, it was expected that SPIDER/SPIDER2 might provide somewhat erratic results, as discussed in Section V-B.

7) AHC and MWMOTE: Through clustering, AHC is able to consider the structure of both minority and majority classes, which is a great advantage over most oversampling algorithms that focus mostly on local properties rather than the whole data structure. Also, specifying the number of clusters is not an issue, since all levels of the resulting dendrograms are considered. However, this originates its major disadvantage: the process becomes very computationally expensive. AHC's ability to take into account the structure of data seems to be one of the reasons why it figures among the best approaches (Table IV), which is also confirmed in similar approaches (e.g. ADASYN, MWMOTE).

As discussed in Section II-A, MWMOTE is the most complete method, and its inner procedure is able to surpass most issues explained above. MWMOTE aims to provide i) an improved way of selecting the minority examples for oversampling, by being more meticulous on the way the importance of minority examples for classification is defined and ii) an improved way of generating new synthetic examples, avoiding the issues of SMOTE-based synthesization. To that end, MWMOTE considers filtering, a weighted scheme based on a taxonomy of minority examples, and a SMOTE-like cluster-based synthesization of examples:

- MWMOTE starts by filtering the initial minority set to find the examples that are surrounded by the majority class, thus avoiding that noisy points are oversampled;
- Then, MWMOTE defines the importance of each minority example for classification, taking into account three

main factors:

(i) minority examples closer to the decision boundary should have a higher weight than those farther from it;
(ii) minority examples within sparse minority clusters should have a higher weight than those on dense minority clusters (which alleviates the problem of small disjuncts);
(iii) minority examples closer to a dense majority cluster should have a higher weight than those closer to a sparse majority cluster.

- Finally, MWMOTE reduces the issues of SMOTE-like synthesization by considering a cluster-based oversampling approach: the generation of new minority examples is performed using only minority neighbors of the same clusters.

By combining strong features of other algorithms (filtering, clustering, adaptive weighting), MWMOTE performs a more guided oversampling procedure, that considers not only the distribution of majority examples around minority examples to define their importance, but also the structure of minority and majority examples (through clustering). This behavior is what makes MWMOTE one of the top approaches, and outstanding in dealing with several difficulty factors that arise in real-world datasets [45], namely overlapping (through filtering), noisy data (through a weighting scheme) and small disjuncts (through clustering). In the extended version of this paper (https://eden.dei.uc.pt/~pha/Long-version-CIM.pdf), readers may find a comprehensive table that resumes the main characteristics of the oversampling algorithms implemented in this work. The key factors that distinguish algorithms from each other are presented in a synthesised way and their greatest advantages and disadvantages are highlighted.

Taking into account the characteristics of the inner procedure of each method, and in light of the performance results discussed in the previous sections, it seems that the best oversampling methods are those that combine three main characteristics:

1) Cluster-based oversampling, so that the structure/distribution of both the minority and majority examples is considered: this approach seems to be superior to considering only the majority neighborhood of individual minority examples or filtering out some minority/majority examples;
2) Adaptive weighting of minority examples: defining a proper taxonomy of minority examples (borderline, safe, noise, and rare/small disjuncts) is crucial so that the importance of each example for classification is properly addressed: more important examples should be oversampled more often;
3) Cleaning procedures, to overcome some issues that rise naturally during oversampling, namely the generation of synthetic examples in overlapping areas.

VI. Conclusions and Future Work

The goal of this work was essentially threefold:

1) To emphasize the risk of overoptimism related to the joint use of CV and oversampling, extending the work of Blagus and Lusa [13];
2) To distinguish the problem of overoptimism from the overfitting problem and study the influence of the datasets' complexity generated by oversampling algorithms on the classification task;
3) To determine the performance of the state-of-the-art oversampling strategies, in order to provide some insight on the ones that reveal the best behavior.

Attending to these sub-objectives, there are three main conclusions to be derived:

(i) The cross-validation procedure after the oversampling (Approach 1) leads to overoptimistic results and makes this approach inappropriate for imbalanced domains. Approach 2 – performing oversampling in the training sets at each iteration of a cross-validation procedure – is the correct way of validating results in imbalanced scenarios. The overoptimism is not related to the data's sample size or imbalance ratio, but rather to the complexity of the prediction task, where the maximum discriminative power of all features (complexity measure F1) seems to be a good predictor of this effect;
(ii) While Overoptimism is greatly associated with inappropriate validation setups, Overfitting (significant differences in performance between training and test sets) is mostly related to the oversampling algorithm used, where algorithms that create exact replicas of existing patterns are the most prejudicial (e.g. CBO+Random). The difference in complexity of the training and test sets is lower in Approach 1 and is the rational behind its overoptimistic behavior: the training and test sets have a similar structure, that is, they are balanced and might contain exact replicas or similar data points to the training data;
(iii) Among the implemented oversampling methods, SMOTE+TL and MWMOTE achieve the best results, with average test AUC values of 0.871 (considering all classifiers). These techniques change the overlapping areas in the data and increase their discriminative power. Overall, the best oversampling techniques possess three key characteristics: use of cleaning procedures, cluster-based synthetization of examples and adaptive weighting of minority examples.

Furthermore, we have performed a regression and clustering analysis which confirmed that the complexity produced by the oversampling algorithms is related to the classification results, in a quasi-linear way. As concluding remarks, we would like to emphasize some lessons learned which could be beneficial to new researchers in the field:

- Oversampling algorithms have distinctive inner procedures that are better suited to particular characteristics of data (e.g. CBO inflates small disjuncts, SMOTE-TL and SMOTE-ENN deal with class overlapping, Safe-Level-SMOTE and Borderline-SMOTE prioritize safe and borderline concepts in data). Thus being, analyzing data

complexity measures may provide useful insights to guide the choice of appropriate oversampling methods;

- Stratified CV is the state-of-the-art validation approach for performance evaluation and should be carefully designed in imbalanced domains. Nevertheless, even a correct CV may cause partition-induced covariate shift during the learning stage [40], which can lead to loss in performance or under-estimation of results. A promising approach to surpass the issues of dataset shift is the Distribution Optimally Balanced stratified cross-validation (DOB-SCV) [46], which is worthy of investigation in future works in the field.

As future work, several undersampling and other new oversampling techniques could be included in the analysis, in order to determine the complexity changes they make in the original datasets. In this context, the novel R package "imbalance" could be of interest, given that it includes the implementation of recent resampling algorithms in the literature [47]. Also, one could focus on specific sub-problems of imbalanced data (e.g. small disjuncts, overlapping, lack of data) and study their identification in multidimensional data and/or ways to surpass them using preprocessing techniques. Finally, future work could also consider an extension of this research for datasets with higher dimensionality.

REFERENCES

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

[2] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6585–6608, June 2012.

[3] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, June 2004.

[4] R. Mollineda, R. Alejo, and J. Sotoca, "The class imbalance problem in pattern classification and learning," in *II Congreso Español de Informática (CEDI 2007). ISBN*, Sep. 2007, pp. 978–84.

[5] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, Apr. 2012.

[6] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, May 2013.

[7] T. Maciejewski and J. Stefanowski, "Local neighbourhood extension of SMOTE for mining imbalanced data," in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, Apr. 2011, pp. 104–111.

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

[9] P. Fergus, P. Cheung, A. Hussain, D. Al-Jumeily, C. Dobbins, and S. Iram, "Prediction of preterm deliveries from EHG signals using machine learning," *PloS One*, vol. 8, no. 10, p. e77154, Oct. 2013.

[10] K. U. Rani, G. N. Ramadevi, and D. Lavanya, "Performance of synthetic minority oversampling technique on imbalanced breast cancer data," in *3rd International Conference on Computing for Sustainable Global Development*. IEEE, Mar. 2016, pp. 1623–1627.

[11] U. R. Acharya, V. K. Sudarshan, S. Q. Rong, Z. Tan, C. M. Lim, J. E. Koh, S. Nayak, and S. V. Bhandary, "Automated detection of premature delivery using empirical mode and wavelet packet decomposition techniques with uterine electromyogram signals," *Computers in Biology and Medicine*, vol. 85, pp. 33–42, May 2017.

[12] K. Oppedal, K. Engan, T. Eftestol, M. Beyer, and D. Aarsland, "Classifying alzheimer's disease, lewy body dementia, and normal controls using 3d texture analysis in magnetic resonance images," *Biomedical Signal Processing and Control*, vol. 33, pp. 19–29, Mar. 2017.

[13] R. Blagus and L. Lusa, "Joint use of over-and under-sampling techniques and cross-validation for the development and assessment of prediction models," *BMC Bioinformatics*, vol. 16, no. 1, pp. 1–10, Nov. 2015.

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

[15] 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, pp. 321–357, June 2002.

[16] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks*. IEEE, June 2008, pp. 1322–1328.

[17] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Advances in Intelligent Computing*, pp. 878–887, Aug. 2005.

[18] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," *Advances in Knowledge Discovery and Data Mining*, pp. 475–482, Apr. 2009.

[19] I. Tomek, "Two modifications of CNN," *IEEE Trans. Systems, Man and Cybernetics*, vol. 6, pp. 769–772, Nov. 1976.

[20] D. L. Wilson, "Asymptotic properties of nearest neighbour rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, July 1972.

[21] S. Tang and S. Chen, "The generation mechanism of synthetic minority class examples," in *IEEE International Conference on Information Technology and Applications in Biomedicine*. IEEE, May 2008, pp. 444–447.

[22] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, June 2004.

[23] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler, "Learning from imbalanced data in surveillance of nosocomial infection," *Artificial Intelligence in Medicine*, vol. 37, no. 1, pp. 7–18, May 2006.

[24] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, Nov. 2014.

[25] J. Stefanowski and S. Wilk, "Selective pre-processing of imbalanced data for improving classification performance," *Lecture Notes in Computer Science*, vol. 5182, pp. 283–292, Sep. 2008.

[26] K. Napierała, J. Stefanowski, and S. Wilk, "Learning from imbalanced data in presence of noisy and borderline examples," in *Rough Sets and Current Trends in Computing*. Springer, June 2010, pp. 158–167.

[27] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289–300, Mar. 2002.

[28] P. H. Abreu, M. S. Santos, M. H. Abreu, B. Andrade, and D. C. Silva, "Predicting breast cancer recurrence using machine learning techniques: A systematic review," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, pp. 1–40, Dec. 2016.

[29] T. K. Ho, "Geometrical complexity of classification problems," *Proceedings of the 7th Course on Ensemble Methods for Learning Machines at the International School on Neural Nets "E.R. Caianiello"*, pp. 1–15, Feb. 2004.

[30] A. Orriols-Puig, N. Macia, and T. K. Ho, "Documentation for the data complexity library in C++," *Universitat Ramon Llull, La Salle*, vol. 196, pp. 1–40, Dec. 2010.

[31] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, Apr. 1982.

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

[33] R. Alejo, J. Monroy-de Jesús, J. H. Pacheco-Sánchez, E. López-González, and J. A. Antonio-Velázquez, "A selective dynamic sampling back-propagation approach for handling the two-class imbalance problem," *Applied Sciences*, vol. 6, no. 7, pp. 1–17, July 2016.

[34] W. A. Rivera and P. Xanthopoulos, "A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets," *Expert Systems with Applications*, vol. 66, pp. 124–135, Dec. 2016.

[35] J. A. Sáez, B. Krawczyk, and M. Woźniak, "Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets," *Pattern Recognition*, vol. 57, pp. 164–178, Sep. 2016.
[36] G. Douzas and F. Bacao, "Self-organizing map oversampling (SOMO) for imbalanced data set learning," *Expert Systems with Applications*, vol. 82, pp. 40–52, Oct. 2017.
[37] S. Shilaskar, A. Ghatol, and P. Chatur, "Medical decision support system for extremely imbalanced datasets," *Information Sciences*, vol. 384, pp. 205–219, Apr. 2017.
[38] J. Liu, Y. Li, and E. Zio, "A SVM framework for fault detection of the braking system in a high speed train," *Mechanical Systems and Signal Processing*, vol. 87, pp. 401–409, Mar. 2017.
[39] J. Luengo, A. Fernández, S. García, and F. Herrera, "Addressing data complexity for imbalanced data sets: Analysis of SMOTE-based oversampling and evolutionary undersampling," *Soft Computing*, vol. 15, no. 10, pp. 1909–1936, Oct. 2011.
[40] 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, Nov. 2013.
[41] M. S. Santos, P. H. Abreu, P. J. García-Laencina, A. Simão, and A. Carvalho, "A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients," *Journal of Biomedical Informatics*, vol. 58, pp. 49–59, Dec. 2015.
[42] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, June 1974.
[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, Apr. 1979.
[44] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2009, vol. 344.
[45] J. Stefanowski, *Dealing with Data Difficulty Factors While Learning from Imbalanced Data*. Springer International Publishing, June 2016, pp. 333–363.
[46] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera, "Study on the impact of partition-induced dataset shift on *k*-fold cross-validation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304–1312, June 2012.
[47] I. Cordón, S. García, A. Fernández, and F. Herrera, *Imbalance: Preprocessing Algorithms for Imbalanced Datasets*, Feb. 2018, R package version 1.0.0. [Online]. Available: https://cran.r-project.org/web/packages/imbalance