Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Radial-Based oversampling for noisy imbalanced data classification

Michał Koziarski[a], Bartosz Krawczyk[b,*], Michał Woźniak[c]

[a] Department of Electronics, AGH University of Science and Technology, Al. Mickiewicza 30, Kraków 30–059, Poland
[b] Department of Computer Science, Virginia Commonwealth University, 401 West Main Street, P.O. Box 843019, Richmond, VA 23284-3019, USA
[c] Department of Systems and Computer Networks, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, Wrocław 50–370, Poland

ARTICLE INFO

*Article history:*
Received 29 November 2017
Revised 17 March 2018
Accepted 21 April 2018
Available online 4 February 2019

*Keywords:*
Pattern classification
Machine learning
Imbalanced data
Oversampling
Radial basis functions
Noisy data

ABSTRACT

Imbalanced data classification remains a focus of intense research, mostly due to the prevalence of data imbalance in various real-life application domains. A disproportion among objects from different classes may significantly affect the performance of standard classification models. The first problem is the high imbalance ratios that pose a serious learning difficulty and require usage of dedicated methods, capable of alleviating this issue. The second important problem which may appear is noise, which may be accompanying the training data and causing strong deterioration of the classifier performance or increase the time required for its training. Therefore, the desirable classification model should be robust to both skewed data distributions and noise. One of the most popular approaches for handling imbalanced data is oversampling of the minority objects in their neighborhood. In this work we will criticize this approach and propose a novel strategy for dealing with imbalanced data, with particular focus on the noise presence. We propose *Radial-Based Oversampling* (RBO) method, which can find regions in which the synthetic objects from minority class should be generated on the basis of the imbalance distribution estimation with radial basis functions. Results of experiments, carried out on a representative set of benchmark datasets, confirm that the proposed guided synthetic oversampling algorithm offers an interesting alternative to popular state-of-the-art solutions for imbalanced data preprocessing.

## 1. Introduction

Most of the classification algorithms assume that there are no significant disproportions among objects from different classes. Nevertheless, in many practical tasks, we may observe that objects from one class (so-called *majority class*) significantly outnumber the objects from remaining classes (*minority class*). Such a problem is known as imbalanced data classification [1,2], where an unequal number of objects from the examined classes plays a key role during the classifier learning. Various approaches have been proposed in the literature to tackle this challenging difficulty embedded in the nature of data. Usually, the researchers are focusing on maximizing the correct minority class classification. At the same time performance on the majority class cannot be neglected. Imbalance data classification has been mentioned as one of the most important challenges in machine learning [3], due to its presence in many real-life scenarios, as anomaly detection [4], fault diagnosis [5] or face recognition [6] to enumerate only a few.

Class imbalance makes the learning task more complex, but the disproportion between object quantity among different classes is not the sole source of difficulty. Another potential factor of high impact lies in minority class objects potentially forming small distributed clusters [7] without an uniform structure. We may also face the problem that the number of objects from minority class is insufficient to train a classification model capable of generalization, which can cause the *overfitting* [8]. All these problems related to imbalanced data have been intensively researched over the last decade [9–12].

Another important problem, which usually accompanies the imbalanced data classification, is the presence of noise [13], which could cause the strong deterioration of a classifier's performance and/or draw out the time required for its training. Therefore, the desirable classifier training model over an imbalance dataset should be also robust to noise.

We distinguish three strategies for handling imbalanced data during the classifier learning step: [14]:

1. *Inbuilt mechanisms*, which adapt the existing classifiers to the mentioned problem and bias them towards favoring the minority class [15,16].
2. *Data preprocessing methods*, which try to equalize the number of the objects from different classes [17,18].

[*] Corresponding author.
*E-mail addresses:* michal.koziarski@agh.edu.pl (M. Koziarski), bkrawczyk@vcu.edu (B. Krawczyk), michal.wozniak@pwr.edu.pl (M. Woźniak).

https://doi.org/10.1016/j.neucom.2018.04.089
0925-2312/© 2019 Elsevier B.V. All rights reserved.

3. *Hybrid methods*, which integrate the preprocessing techniques with the inbuilt mechanisms [19].

In this work we will focus on the data preprocessing approach, which generates new synthetic objects from minority class (*oversampling*) and/or removes objects from the majority class (*undersampling*). Basic methods from this family do not take into consideration the imbalanced data distribution and as a result, they may lead to an increased difficulty of the classification task, e.g., in the case in which easily separable subpopulation of the objects (a particular feature subspace) is injected by objects from the opposite class. Therefore, whether we aim to produce new minority class objects or to remove objects from majority class, a knowledge about imbalance distribution should guide the sampling procedure. In this paper we will focus on the oversampling strategies. Among them, SMOTE (*Synthetic Minority Over-sampling Technique*) algorithm [9] is the best known, but ADASYN (*ADAptive SYNthetic sampling*) [20], which takes into consideration which objects are difficult to learn, and RAMO (*Ranked Minority Oversampling*) [21] should also be mentioned. The main drawback of SMOTE is that it does not take into consideration the majority objects from the neighborhood when it generates new synthetic objects. Such a behavior may increase the overlapping of classes and introduce an additional noise. Therefore, several extensions have been proposed, such as Borderline-SMOTE [22], which generates synthetic minority class objects close to the decision border. Safe-level SMOTE [10] and LN-SMOTE [23] aim at reducing the risk of introducing synthetic objects inside regions of the majority class.

In this work we will focus on alleviating the mentioned drawbacks of SMOTE-based methods by proposing a significant extension of the method initially reported in [24]. We will consider imbalance distribution estimation based on the radial basis functions, which will be used by the proposed *Radial-Based Oversampling* (RBO) method to find regions in which the synthetic objects from minority class should be generated.

The main contributions of this paper are as follows:

- Proposition of Radial-Based Oversampling method for handling imbalanced data, which is able to choose appropriate regions, in which synthetic objects from minority classes should be generated.
- Introduction of artificial objects based on information from both minority and majority classes.
- Explanation on how the potential-based oversampling alleviates drawbacks of popular SMOTE-based methods.
- Showcasing capabilities of the proposed method to handle challenging imbalanced data with label and feature noise presence.
- Experimental evaluation of the proposed algorithm on the basis of diverse benchmark datasets and a detailed comparison with the state-of-the-art methods for imbalanced classification.

The remainder of this manuscript is organized as follows. Section 2 gives an overview of learning from imbalanced and noisy datasets. Section 3 describes in detail the proposed RBO algorithm, while Section 4 discusses the experimental set-up and the obtained results. Section 5 summarizes our research findings, while the final section concludes the paper and provides our insights into future works.

## 2. Learning from imbalanced and noisy data

In this section, we will present a short overview of the challenges in learning classifiers from data characterized by both class imbalance and noise presence.

### 2.1. Imbalanced data

Despite more than twenty years of progress, learning from imbalanced data is still among the contemporary challenges faced by machine learning [17,25]. This can be attributed to constant emergence of new real-world applications that have skewed data problem embedded in them [26]. Additionally, we have gained a deeper insight into what causes learning algorithms to fail when dealing with imbalanced problems. The imbalanced ratio, commonly targeted as the main factor, turns out not to be the sole source of learning difficulties [17]. It is easy enough to create a dataset that will have extremely high imbalance ratio, while at the same time pose little challenge to most of classifiers. Such a situation can occur when (i) we have enough objects from both classes to train a good classifier; and/or (ii) when we have well-separated objects between classes.

The former situation is connected with small training sample size. In many problems we are not able to gather sufficient number of minority class objects [27]. When their quantity is low, we are not able to train a competent classifier. Small minority training sets may not reflect the true object distribution of this class, misleading the classifier and preventing it from obtaining generalization capabilities. This leads to dataset shift, when the train set does not reflect the test set [28].

The latter of discussed situations is connected with class overlapping. The higher the overlapping ratio, the more difficult it becomes to properly recognize the minority class. Standard classifiers (ones without inbuilt mechanism for becoming skew-insensitive) will always favor the majority class in highly overlapping regions of the feature space. Minority class structure is not always uniform. Many researchers proposed to internally distinguish among minority class objects based on the difficulty they pose for a classifier. The simplest approach is to separate into easy objects (ones that can be properly recognized without any special techniques dedicated to imbalanced data) and difficult objects (ones that require additional attention in order to be properly utilized by a classifier) [29]. Some researchers proposed to further extend this taxonomy into four types of minority class objects [7]: safe (same as easy objects), borderline (objects partially overlapping with the majority class, located in the uncertainty region of the feature space), disjuncts (minority class structures located within the majority class, characterized by both small sample size and overlapping) [30], and outliers (singular objects located far from main minority class structures, either rare [31] or noisy [32]). Recent works show that incorporating such an information into data preprocessing or classifier design may lead to significant improvements in obtained learning models [33]. This shows the importance of understanding which minority class objects are the most important ones - a methodology that stands behind the RBO algorithm introduced in this paper.

### 2.2. Noisy data

The presence of noise in data may lead to predictive performance deterioration, prolonging the training time, and increasing the complexity of outputted classification models. It is obvious that there is a strong correlation between the final model quality and the quality of supplied training data.

Data noise has two main sources [13]: errors introduced by measurement tools and random errors introduced by processing or by experts when the data is gathered [34]. Based on the properties of the training set that are affected, we distinguish two types of noise: label noise and feature noise.

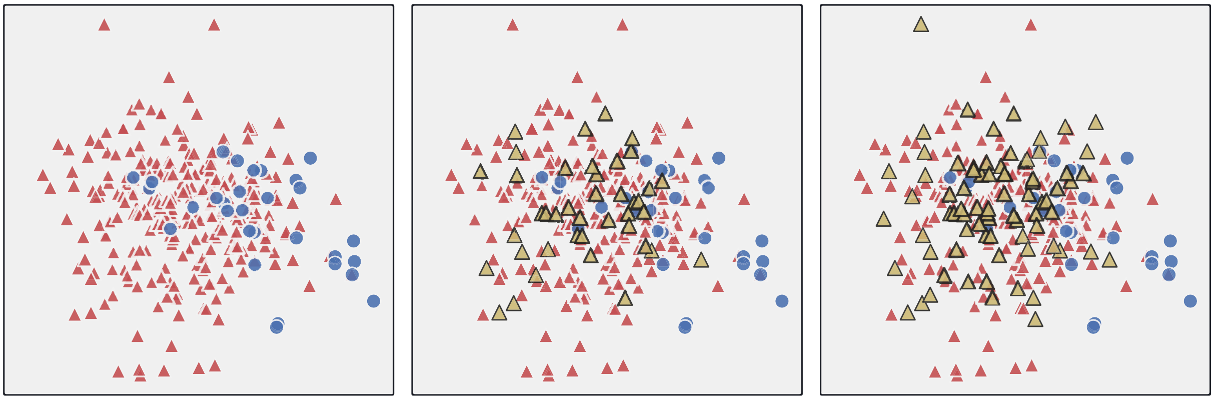


**Fig. 1.** An example of a noise affecting class labels under class imbalance. (*Left*) Initial dataset. (*Center*) Dataset with 10% of minority objects affected by label noise. (*Right*) Label noise ratio increased to 20%. Noisy objects are highlighted. Each consecutive scenario poses additional difficulties for classifier, as we lose information regarding the minority class, leading to increase of imbalance ratio and reduced (or even lacking) presence of this class in some regions of the feature space.

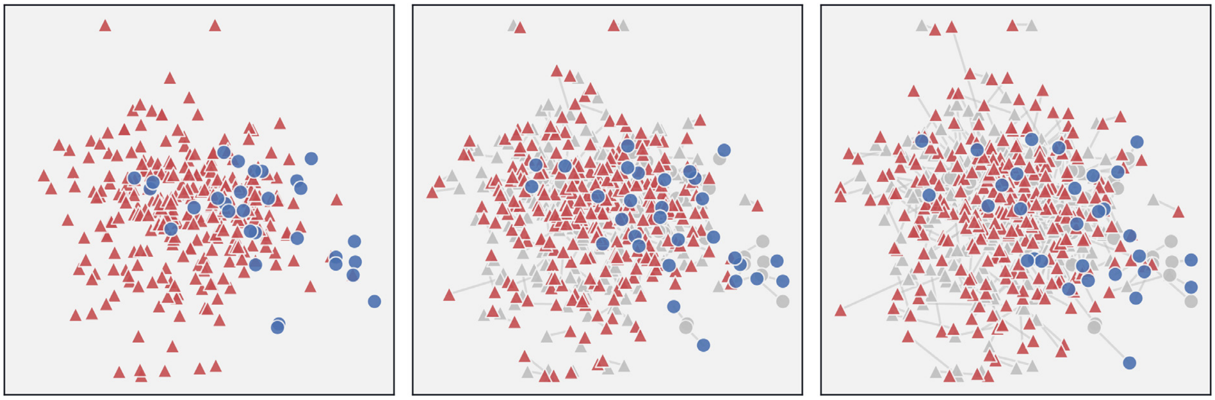


**Fig. 2.** An example of a noise affecting feature values under class imbalance. (*Left*) Initial dataset. (*Center*) Dataset with feature values affected by a Gaussian feature noise with strength set to 20%. (*Right*) Feature noise strength increased to 40%. Gray objects indicate the original position of each object, while gray lines indicate shift vectors that were result of attribute noise introduction. Each consecutive scenario poses additional difficulties for classifier, as class distributions shift, not reflecting anymore the true ones, overlapping between classes increase, and we lose homogeneous regions belonging to a specific class.

### *2.2.1. Label noise*

Label noise occurs when an object is being provided in the training set with an incorrect class label (Fig. 1) [35]. Lugosi [36] noticed that "*the teacher may sometimes lie*" and he proposed learning with an unreliable teacher approach based on the estimated *posterior* probability maximization and nearest neighbor classification. Class noise can be attributed to several causes, such as subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each object. Two types of class label noise may be distinguished [37]:

- contradictory objects [38], when duplicate objects have different class labels;
- mislabeled objects that are assigned to another class than the actual one [39].

Of course, we have to be very careful when deciding which object will be considered as having noisy class label. Assuming statistical dependencies between attribute values and class label, we must be aware that objects with the same attribute values may appear with different labels. This is due to the fact that used attributes have not enough discriminant power to distinguish objects among classes.

In many human activities, such as medical diagnosis, the class noise may be caused by:

- Human error, because we cannot assume that a human expert is infallible. The cost of such misclassification may be very high, e.g., in [40] authors stated that clinical mistakes (i.e., incorrect diagnosis etc.) which could have been avoided, cause each year 44,000 - 98,000 patents deaths in USA alone.
- Machine error, which may be caused by design faults or momentary error.
- Digitalization error, caused by incorrectly inputting a class label.
- Archiving error, caused by missing or incorrectly copied information.

The recent and extensive review on class noise may be found in [41]

### *2.2.2. Feature noise*

Feature noise might also be caused by a human during the data imputation or by a machine, e.g., because of sensor/measurement error (Fig. 2). We may distinguish the following types of feature noise:

- Erroneous feature values, which are usually caused by incorrect imputation or quality of measurement tools.
- Missing feature values, caused by the incorrect imputation or by the case that for a given record an acquisition of this specific attribute value was worthless. Several hints how handling missing attribute values may be found in [42].
- Incomplete features, usually caused by incorrect imputation or that appear during record digitalization or archiving.

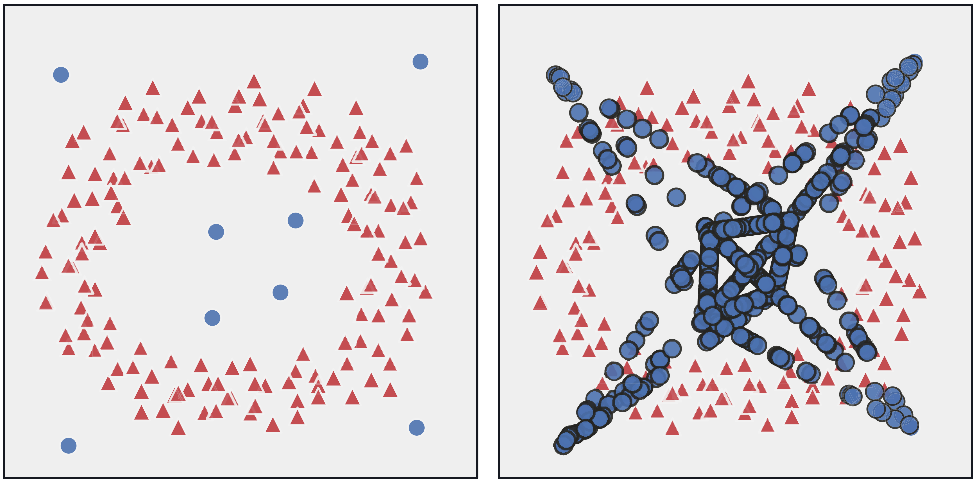


**Fig. 3.** An example of a difficult dataset. Neighborhood-based methods are not resilient to small number of minority objects, disjoint data distributions, or presence of the outliers. On the left: original imbalanced dataset. On the right: data after the oversampling with SMOTE. Synthetically generated samples overlap the original majority distribution.

## 3. Radial-based approach to oversampling

Neighborhood-based oversampling strategies, such as SMOTE and its derivatives, are by far the most prevalent approaches to dealing with the imbalance on the data level. However, despite their popularity, this family of methods does not remain without its own shortcomings. In the remainder of this section, we will discuss some of disadvantages of the neighborhood-based oversampling strategies, putting a particular emphasis on the issue of noisy data. Afterwards, we propose an alternative method for finding the regions of interest, based on the potential estimation with radial basis functions. Finally, we present a guided oversampling strategy, named *Radial-Based Oversampling* (RBO), in which potential estimation is employed to synthetically generate new minority objects.

### *3.1. Shortcomings of neighborhood-based approaches*

Conceptually simplest oversampling-based approach to dealing with the imbalanced data is random oversampling (ROS). When applying ROS new objects are being generated by duplicating randomly chosen, existing objects. The drawback of this approach is that it leads to the minority objects being grouped in small areas, in which the original objects were placed. This can present a problem for some of the classifiers, especially those prone to overfitting. SMOTE algorithm and its derivatives were proposed specifically to alleviate this issue. Instead of duplicating existing objects, SMOTE aimed at synthesizing new ones. In the original SMOTE the synthetic objects are being placed on the lines connecting the existing minority objects with its nearest minority neighbors. This lines act as a regions of interest, in which the creation of new objects is being considered. Compared to ROS, this approach leads to a more spread-out clusters of minority objects, leading to a reduced risk of overfitting. However, SMOTE makes an implicit assumption that the produced regions of interest are indeed suitable for the oversampling. We argue that oftentimes this is not the case. An example of a dataset for which the regions of interest produced by SMOTE might be inappropriate is presented in Fig. 3. Due to the presence of spread-out, minority outliers SMOTE's regions of interest overlap with the original majority class distribution. This behavior is intuitively harmful, as synthetically generated, overlapping objects are not representative of the original data distribution. So while the expansion of the regions of interest compared to ROS is desirable, neighborhood-based approaches might often expand them in unwarranted directions, not taking into the account the majority class distribution. This issue is prevalent in case of noisy data, for which the likelihood of observing disjoint outliers is high, especially if the noise occurs on the label level.

Some recent works suggest using kernel functions [43,44] or density-based [45] solutions to achieve a more informative oversampling and overcome local instance-level difficulties. Kernel-based solutions perform a high-dimensional mapping into an artificial feature space that should lead to a better separation between classes and thus easier oversampling without the risk of increasing class overlapping. Density-based methods try to avoid using neighborhood-based introduction of artificial instances and focus instead of detecting regions of given density for minority class. Despite being a step forward from neighborhood-based methods, these techniques still suffer from a number of drawbacks. Kernel-based methods increase the dimensionality of the artificial feature space, which may further emphasize the issue of limited access to minority class instances (leading to the problem of small sample size and high dimensionality). Additionally, kernel-based approaches do not take into account individual difficulty of instances, thus allowing for noisy or rare objects to strongly influence the mapping process. Density-based solutions use only density from the minority class, thus practically ignoring the role of majority class and suffering for similar limitations as neighborhood-based approaches when facing difficult data distributions.

### *3.2. Potential estimation with radial basis functions*

An essential step in oversampling strategies that synthetically generate new objects is establishing the regions of interest, in which the new objects should be introduced. SMOTE and its derivatives achieve that by connecting the nearby minority objects and generating the new objects alongside those connections. While conceptually very simple, this approach does not take into the account neither the position of existing majority class objects, nor the distance between the connected minority objects. As a result, generated regions of interest can overlap areas with a high density of the majority class objects. Intuitively, this is often not a desired behavior, since the goal of the oversampling is to generate majority objects reflecting the whole data distribution.

To mitigate this issue, we propose an alternative approach for approximating the regions of interest. Instead of generating binary regions between the minority objects, we propose using a real-valued potential surface, with potential at each point in space representing our preference towards that point belonging to either

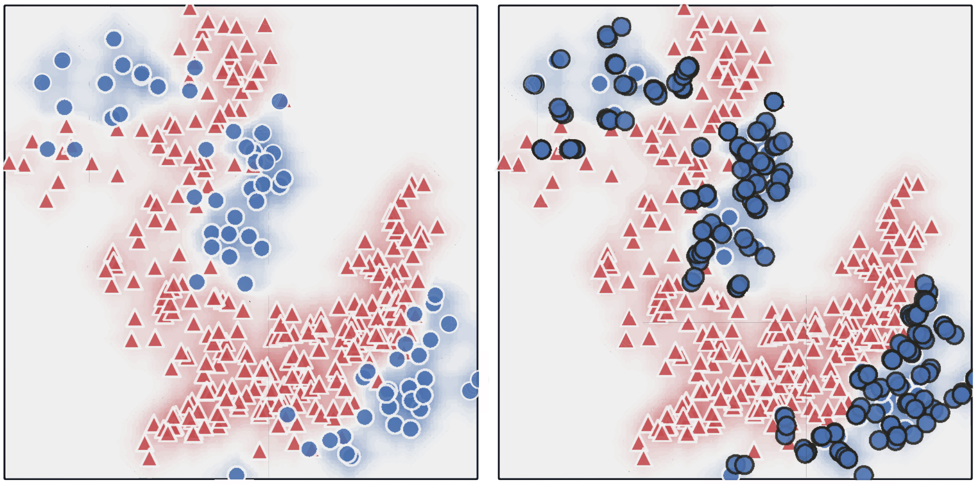


**Fig. 4.** On the left: visualized potential function for the original dataset. On the right: the same potential with an addition of the synthetic samples generated by the *Radial-Based Oversampling* algorithm.

minority or majority class. To calculate that potential, we assign a Gaussian radial basis function (RBF) to every object in our training dataset, with the polarity dependent on its class. Given a set of majority objects $K$, a set of minority objects $\kappa$, and a parameter $\gamma$ representing a spread of a single RBF, we define the potential in a point $x$ as

$$\Phi(x, K, \kappa, \gamma) = \sum_{i=1}^{|K|} e^{-\left(\frac{\|K_i - x\|_1}{\gamma}\right)^2} - \sum_{j=1}^{|\kappa|} e^{-\left(\frac{\|\kappa_j - x\|_1}{\gamma}\right)^2}. \tag{1}$$

where $K_1$ stands for the $i$th object from majority class and $kappa_j$ denotes $j$th object form minority class respectively.

An example of a potential surface for a two-dimensional dataset is presented in Fig. 4. Minority objects, as well as associated with them negative potentials are represented by a blue color, whereas majority objects and associated with them positive potentials are represented by a red color. Potentials close to zero were indicated by a gray color.

Compared to the regions of interest generated by SMOTE, potential surface has several advantages. First of all, it is resilient to the presence of the outliers, as well as a small number of minority objects combined with disjoint distributions. While in those cases SMOTE is likely to generate synthetic objects over the clusters of a majority objects, using the potential surface would result in a smaller, constrained regions of a negative potential. Secondly, compared to the regions of interest used by SMOTE, potential surface contains more information that can be used to guide more sophisticated placement strategies. For instance, depending on the task at hand and the preference towards either the precision or the recall, synthetic minority objects might be placed either at the regions with a potential close to zero, or those with a very high potential. The former might be interpreted as an uncertain area, and placing new objects in it should move the decision border in favor of the minority class, whereas the latter represents a higher certainty, and should lead to a more conservative decision border.

### 3.3. Imbalanced data oversampling

Based on a potential surface, various strategies of imbalanced data resampling can be designed. In this paper, we propose an oversampling algorithm that creates artificial objects as a product of an iterative optimization algorithm. We decided to optimize the absolute value of the potential, which corresponds to placing the synthetic minority objects in regions of high uncertainty, close to the predicted decision border. We considered two alternative optimization criteria, namely the maximization and the minimization of potentials. The final choice was motivated by the intuitive soundness of objects generated in the local minima, especially when compared to the considered alternatives, as well as the visual examination of the datasets oversampled with the different optimization criteria. In all the cases, we were choosing a random minority object as a starting point for the optimization. The maximization of the potential, if not constrained by the distance, would lead to oversampling over the clusters of majority points, the exact behavior we find disadvantageous in SMOTE. On the other hand, the minimization of the potential would often be immediately stuck in a local minima, since the existing minority objects were often associated with an especially low potential.

The optimization procedure itself is based on a simple hill climbing algorithm. Starting by generating a new object in a position associated with a randomly chosen minority object, at every iteration we translate the object in a random direction and preserve the translation only if the absolute value of the potential decreases. The procedure lasts for a number of iterations given as a parameter of the algorithm, with a small probability of an early stopping, also given as a parameter. The addition of the early stopping possibility is motivated by the idea of covering the considered region of interest more evenly, instead of placing them only on the decision borders. Importantly, during the oversampling, in the proposed version of the algorithm the newly generated, synthetic minority instances are not being considered during the potential calculation. We observed that synthetic instances could cause a significant drift in the original potential and deemed this behavior potentially dangerous, especially in the case of an extreme imbalance.

Complete pseudocode of the described oversampling strategy is presented in Algorithm 1. In the proposed variant, the oversampling was performed up to the point of balancing the class distributions. An example of an originally imbalanced dataset after applying the RBO was presented in Fig. 4. As can be seen, in case of a singular outliers, new objects are being generated in a close proximity, and the overlapping of the majority clusters is limited. On the other hand, in the remaining cases synthetically generated objects are being spread across the regions of interest, with a preference towards the difficult areas. It can be argued that in case of a singular outliers, instead of limiting the distance within which the synthetic objects can be generated, the outliers

**Algorithm 1** Radial-Based Oversampling.

```
Input: collections of majority objects $K$ and minority objects $\kappa$
Parameters: spread of radial basis function $\gamma$, optimization step size, number of iterations per synthetic sample, probability of early stopping $p$
Output: collection of synthetic minority objects $S$

function RBO($K$, $\kappa$, $\gamma$, step size, iterations, $p$):
initialize empty collection $S$
while $|\kappa| + |S| < |K|$ do
    point $\leftarrow$ randomly chosen object from $\kappa$
    for $i \leftarrow 1$ to iterations do
        break with probability $p$
        direction $\leftarrow$ randomly chosen standard basis vector in a $n$-dimensional Euclidean space, with $n$ being the number of features
        sign $\leftarrow$ randomly chosen value from $\{-1, 1\}$
        translated $\leftarrow$ point $+$ direction $\times$ sign $\times$ step size
        if $|\Phi(translated, K, \kappa, \gamma)| < |\Phi(point, K, \kappa, \gamma)|$ then
            point $\leftarrow$ translated
        end if
    end for
    add point to $S$
end while
return $S$
```

could be ignored completely and regarded as a noise. However, especially in the case of extreme imbalance, such behavior could lead to omitting important instances. Therefore, instead of making an assumption about the nature of the outliers on the level of the oversampling algorithm, if necessary we propose using data cleaning prior to oversampling.

Characteristics of the proposed RBO algorithm can also be favorable when dealing with noisy data. Most of existing oversampling schemes suffers from a degraded performance when the noise level is being increased. Let us explain this phenomenon using SMOTE as an exemplary algorithm. Label noise will lead to mixed objects in the neighborhood, while feature noise will lead to increased overlapping between classes. As SMOTE creates new artificial objects based on a selected neighbors, both types of noise will affect it significantly. Label noise will lead to either too small neighborhood that will not allow for sufficient empowering of the minority class presence in this region (when majority objects in overlapping areas are mislabeled as minority ones), or to overextended neighborhood that will introduce scattered artificial objects (when local minority objects are mislabeled as majority ones). Feature noise will shift the location of objects in the training set, thus leading to incorrect placement of new objects (as they are placed along the lines joining two selected minority objects). These limitations are shared by most of oversampling techniques from SMOTE family. While recently SMOTE-IPF [46] implementation was proposed in order to tackle noisy data, we must note that it combines standard SMOTE with computationally costly filtering. Therefore, it seems more interesting to investigate oversampling methodologies that have inbuilt mechanism for handling the presence of noise. RBO falls into this category, as by using potentials we are able to identify which regions should be subject to oversampling. Therefore, shifts in the training set will not affect RBO procedure as much as reference methods. Additionally, RBO generates new objects and positions them in an less random manner than SMOTE, eliminating the problem of artificial objects being spread over the feature space. An illustrative example of differences between SMOTE and RBO on noisy data is given in Fig. 5.

**Table 1**
Details of the datasets used during the preliminary (top) and the final (bottom) analysis.

| No. | Name | IR | Features | Samples |
|---|---|---|---|---|
| 1 | pima | 1.87 | 8 | 768 |
| 2 | yeast1 | 2.46 | 8 | 1484 |
| 3 | haberman | 2.78 | 3 | 306 |
| 4 | vehicle2 | 2.88 | 18 | 846 |
| 5 | led7digit02456789vs1 | 10.97 | 7 | 443 |
| 6 | yeast1vs7 | 14.30 | 7 | 459 |
| 7 | winequalityred4 | 29.17 | 11 | 1599 |
| 8 | poker9vs7 | 29.50 | 10 | 244 |
| 9 | abalone3vs11 | 32.47 | 8 | 502 |
| 10 | winequalitywhite9vs4 | 32.60 | 11 | 168 |
| 1 | glass1 | 1.82 | 9 | 214 |
| 2 | glass0 | 2.06 | 9 | 214 |
| 3 | vehicle1 | 2.90 | 18 | 846 |
| 4 | vehicle3 | 2.99 | 18 | 846 |
| 5 | glass0123vs456 | 3.20 | 9 | 214 |
| 6 | vehicle0 | 3.25 | 18 | 846 |
| 7 | ecoli1 | 3.36 | 7 | 336 |
| 8 | newthyroid1 | 5.14 | 5 | 215 |
| 9 | ecoli2 | 5.46 | 7 | 336 |
| 10 | segment0 | 6.02 | 19 | 2308 |
| 11 | glass6 | 6.38 | 9 | 214 |
| 12 | yeast3 | 8.10 | 8 | 1484 |
| 13 | ecoli3 | 8.60 | 7 | 336 |
| 14 | yeast2vs4 | 9.08 | 8 | 514 |
| 15 | vowel0 | 9.98 | 13 | 988 |
| 16 | glass016vs2 | 10.29 | 9 | 192 |
| 17 | glass2 | 11.59 | 9 | 214 |
| 18 | ecoli4 | 15.80 | 7 | 336 |
| 19 | pageblocks13vs4 | 15.86 | 10 | 472 |
| 20 | abalone918 | 16.40 | 8 | 731 |
| 21 | yeast1458vs7 | 22.10 | 8 | 693 |
| 22 | yeast4 | 28.10 | 8 | 1484 |
| 23 | yeast1289vs7 | 30.57 | 8 | 947 |
| 24 | poker89vs6 | 58.40 | 10 | 1485 |

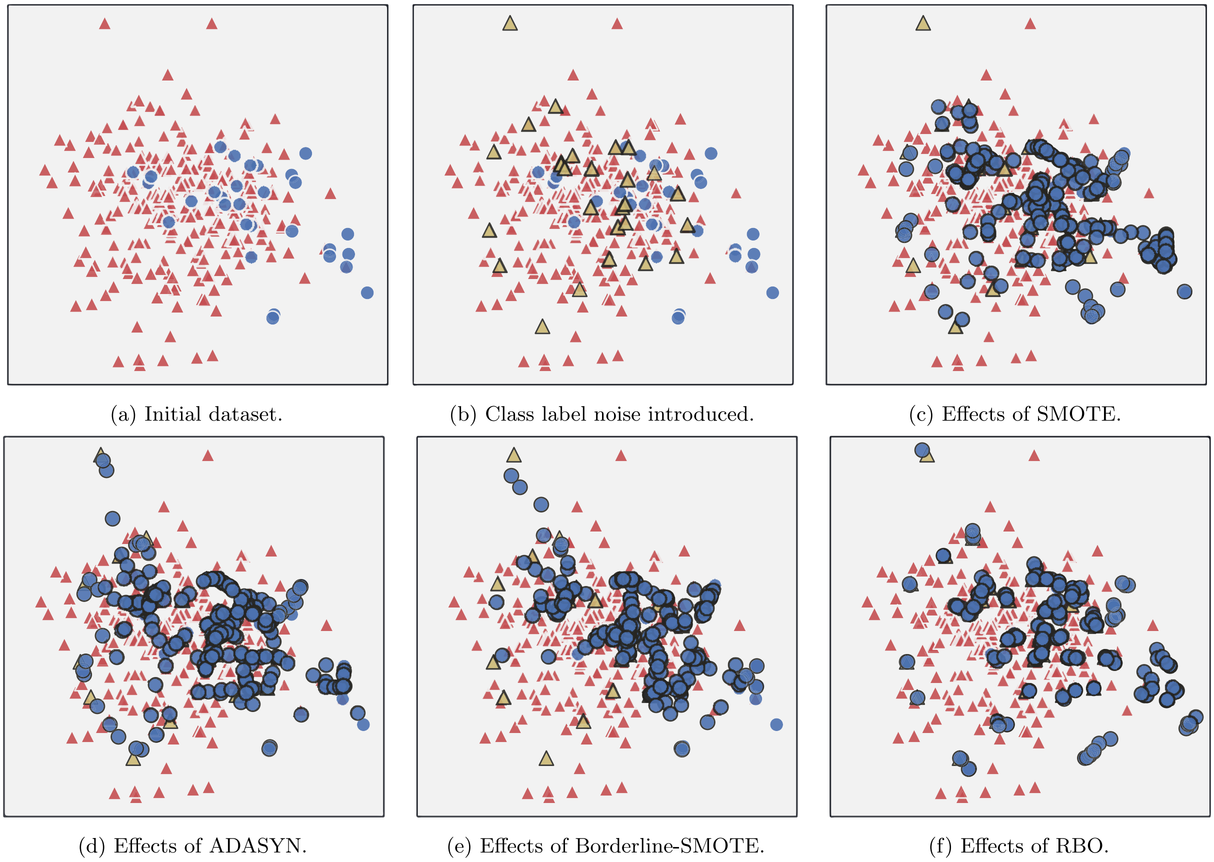
## 4. Experimental study

A series of experiments was conducted to empirically evaluate the performance of the RBO algorithm. During the first experiment, we tested the impact of RBO's parameters on its performance during the task of imbalanced data classification. Then we compared the RBO algorithm with the state-of-the-art approaches of dealing with imbalanced data under the standard conditions, that is without the presence of the noise. Finally, we evaluated how the presence of the noise, both at the level of labels and features, affects the RBO's performance compared to the state-of-the-art alternatives. In the remainder of this section we describe our experimental set-up and present the achieved results.

### 4.1. Set-up

*Data.* Overall, 34 two-class datasets with a varying level of imbalance were used throughout the experiments. Out of them, 10 were randomly chosen for the preliminary analysis, while the remaining 24 were used for the remainder of the experiments, the final analysis. The details of the datasets used during the preliminary and the final analysis are presented in Table 1. In addition to the imbalance ratio (IR), the number of features and contained samples was specified for each dataset. As can be seen, chosen datasets vary in size, number of features and degree of imbalance, which ensures a reliable assessment of the performance. All of the datasets were taken from the KEEL [47] imbalanced data repository, and as such are publicly available.

For the cross-validation, the datasets were partitioned into a 5 × 2 folds [48]. Prior to resampling and classification, categorical features were encoded as integers. Afterwards, all the features



(a) Initial dataset.

(b) Class label noise introduced.

(c) Effects of SMOTE.

(d) Effects of ADASYN.

(e) Effects of Borderline-SMOTE.

(f) Effects of RBO.

**Fig. 5.** An example of influence of class label noise on imbalanced data oversampling. SMOTE and ADASYN are strongly affected, as they have no mechanisms to reduce the influence of noisy neighbors on introduced artificial objects. Borderline SMOTE is able to detect obvious outliers, but is still prone to generating synthetic minority objects over clusters of majority objects. The proposed RBO approach is less affected due to its potential-based selection of regions to oversample.

were normalized to a (0, 1) range based on the original range of the training data. No further preprocessing was applied.

*Performance metrics.* Since the classification accuracy is not a proper performance metric in case of a highly imbalanced data, during the conducted experimental study we instead considered the most widely used measures for the imbalanced binary classification, namely: precision, recall, G-mean, F-measure and AUC.

*Classifiers.* To ensure the validity of the obtained results four classification algorithms, representing different methodologies, were used as a base learners, namely: CART decision tree, k-nearest neighbors (k-NN), support vector machine (SVM) with RBF kernel and Naive Bayes (NB). The implementations of the classification algorithms provided in the scikit-learn [49] machine learning library were used, together with their default parameters.

*Reference methods.* Excluding the preliminary analysis, performance of the RBO algorithm was compared with the state-of-the-art, data-level methods of dealing with the imbalance in data. Specifically, for the comparison we used basic SMOTE [50], as well as several of its extensions: Borderline-SMOTE (Bord) [51] and SMOTE in combination with the data cleaning methods [52], Tomek links (SMOTE+TL) [53] and Edited Nearest-Neighbor Rule (SMOTE+ENN) [54]. Furthermore, we considered the ADASYN [55] algorithm, as well as the Neighborhood Cleaning Rule (NCL) [56]. Finally, as a reference we evaluated the baseline case, in which no data resampling was applied prior to the classification (Base). Throughout the experimental analysis, the implementations of the reference methods provided in the imbalanced-learn [57] library were used. For each of the methods, the default parameters provided in the imbalanced-learn were employed.

*Implementation and reproducibility.* RBO algorithm, as well as the experiments described in this paper, were implemented in the Python programming language. Complete code, sufficient to repeat the experiments, was made available at.[1] Together with the code, we provided the partitioning of the datasets into the cross-validation folds, as well as the complete results of the conducted experimental analysis.

### 4.2. Experiment 1: preliminary analysis

The goal of this experimental analysis was evaluating the impact of the RBO's parameters on its performance. Specifically, we considered the influence of the spread of a radial basis function $\gamma$, as well as the effective distance by which the original minority objects can be translated, represented by the number of iterations of the algorithm. We conducted our analysis on a subset consisting of 10 datasets. We disabled the probability of early stopping, and fixed the step size of the algorithm at 0.001. At the same time, we considered a combination of $\gamma$ values in $\{0.001, 0.01, \ldots, 10.0\}$ and the number of iterations in $\{1000, 2000, 4000, \ldots, 16,000\}$.

Visualization of this part of the experimental evaluation are presented in Fig. 6. Due to the space constraints, only the values of AUC for SVM classifier were shown. However, very similar trends were observed for the remaining classifiers and the rest of the combined performance metrics, that is F-measure and G-mean. Complete visualizations of the preliminary analysis, containing the remaining metrics and classifiers, were provided at the

[1] https://github.com/michalkoziarski/RBO.

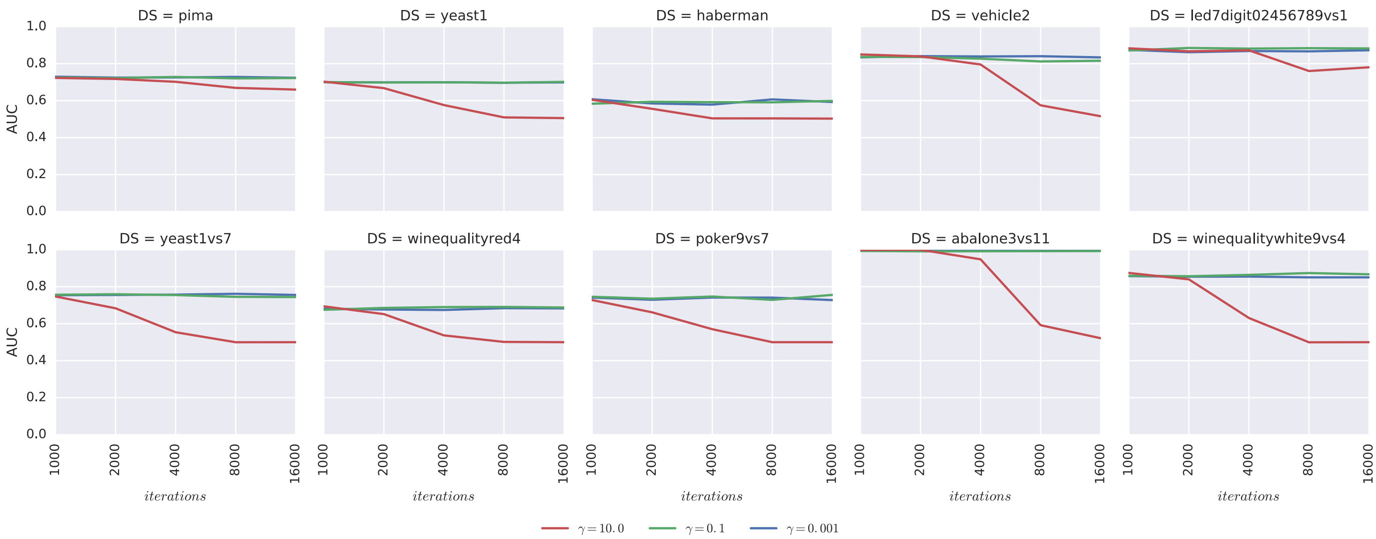


**Fig. 6.** Impact of the number of *iterations* on the AUC, for various datasets (DS) and $\gamma$ values. SVM was used as a classifier.

website. For simplicity, only the values of $\gamma$ in {0.001, 0.1, 10.0} were presented, since the omitted, intermediate values did not change drawn conclusions.

As can be seen, for a small values of $\gamma$ performance of the algorithm saturates quickly, and increasing the number of iterations has an insignificant influence on the AUC. Furthermore, RBO remains relatively stable to the choice of $\gamma$ as long as it does not take especially high values, larger than 1. However, when combined with a high number of iterations, choosing a large value of $\gamma$, such as 10, significantly decreases the algorithms performance.

Based on the results of the preliminary analysis, for the reminder of the experimental study we set the number of iterations to 5000, at the same time decreasing the algorithms step size to 0.0001. Furthermore, at avoid the incorrect combination of translation distance and $\gamma$, we performed a parameter selection of the latter, considering the values in $\{0.001, 0.01, \ldots, 10.0\}$. Selection was performed based on the AUC achieved on a validation set, divided from the original training data. Finally, we introduced a small probability of early stopping, equal to 0.001.

### 4.3. Experiment 2: imbalanced data without noise

During this experimental analysis we evaluated RBO against state-of-the-art, data-level approaches to imbalanced data classification. Compared to the original submission [24] in which RBO was first presented, we significantly extended the test conditions. Specifically, we used a larger pool of datasets, included more reference methods, added another classification algorithm and extended the performance metrics with AUC. Furthermore, we performed a statistical analysis of the achieved results to assess their significance.

In Table 2 we present the average rankings of the algorithms computed with a Friedman test. Furthermore, we denoted the methods that achieved a significantly different results than RBO according to the Shaffer's post-hoc test at a significance level of 0.05. As can be seen, compared to the reference methods, RBO achieved the best performance in combination with SVM and NB classifiers. In particular, RBO achieved the highest average rank of all of the considered methods for every combined metric (that is: F-measure, AUC and G-mean), as well as the precision, when used together with the NB classifier. On the other hand, RBO achieved relatively poor results when combined with the CART and k-NN classifiers, leading to the average ranks lower than those of most of the reference methods. However, it should be noted that the differences were rarely statistically significant, especially in the cases, in which RBO performed poorly.

### 4.4. Experiment 3: imbalanced data with label noise

During the third part of the experimental evaluation we measured the influence of the label noise on the classification performance. We artificially simulated the noise by modifying the original data according to the schema proposed by Zhu et al. [39]: given the pair of classes (*X*, *Y*), *X* being the majority class and *Y* the minority class, and a noise level *x*%, an object with label *X* has a probability of *x*% to be incorrectly labeled as *Y*. In case of imbalanced data high levels of noise can alter the relation between the classes, that is the original minority class can become the majority class after applying the noise. Since oversampling techniques were originally used by generating new minority objects up to the point of achieving the class balance, it is unclear how this procedure should be altered if majority class was switched due to applying the noise. Because of that we limited ourselves to the noise levels in {0.0, 0.05, 0.1, 0.15, 0.20}, for which the majority class did not switch for any of the datasets.

We present the results of this stage of the experimental study in Tables 3–6, separately for each of the considered classifiers. Similarly to the part of the experimental study with no noise present, the best performance was observed for the NB classifier. In that case the highest average ranks were observed for all of the combined metrics, on most of the considered noise levels. In general, compared to the case with no distortions, low to medium levels of noise caused the RBO to achieve relatively higher ranks for all of the classifiers except k-NN. In several cases the increase was significant enough that RBO achieved the highest rank out of the considered methods, both in case of the CART and the SVM. The opposite trend was observed when RBO was combined with the k-NN classifier, in which case the decrease in relative performance was observed when the label noise was added.

### 4.5. Experiment 4: imbalanced data with feature noise

In the final stage of the experimental evaluation we measured the impact of the feature noise on the RBO's performance. We followed the procedure introduced by Sáez et al. [35] and modified the original observations by adding a random value to each feature. The value followed a Gaussian distribution with zero mean and

**Table 2**
Average rankings for the baseline case, in which no noise was explicitly added to the data. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Metric | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| CART | Precision | 4.0000 | 4.7917 | 6.3333$_{+}$ | 4.0417 | 4.5833 | **3.1667** | 5.9167$_{+}$ | **3.1667** |
| | Recall | 6.9375 | 4.2500 | **2.3542**$_{-}$ | 4.4792 | 4.0625 | 5.1250 | 3.3542 | 5.4375 |
| | F-measure | 5.7083 | 5.0000 | 4.2083 | 4.1458 | 4.2708 | 4.9167 | **3.6250** | 4.1250 |
| | AUC | 6.7083 | 4.0417 | **2.9583** | 4.6458 | 4.0625 | 5.0417 | 3.5417 | 5.0000 |
| | G-mean | 6.7917 | 4.1667 | **2.9167**$_{-}$ | 4.4792 | 3.9792 | 5.0000 | 3.5000 | 5.1667 |
| k-NN | Precision | **2.0625**$_{-}$ | 5.2500 | 6.0000 | 5.2500 | 5.4167 | 4.3750 | 2.6458$_{-}$ | 5.0000 |
| | Recall | 7.8958$_{+}$ | **2.7917** | 3.5000 | 3.6250 | 3.2500 | 4.2500 | 6.2708 | 4.4167 |
| | F-measure | 5.1250 | 3.9167 | 5.3750 | **3.7083** | 4.4167 | 4.2917 | 4.7083 | 4.4583 |
| | AUC | 7.4167$_{+}$ | **2.6458** | 4.7917 | 3.0833 | 3.4583 | 3.9583 | 5.9167 | 4.7292 |
| | G-mean | 7.3750$_{+}$ | **2.5000**$_{-}$ | 4.7083 | 3.4583 | 3.2500 | 3.7083 | 6.0833 | 4.9167 |
| SVM | Precision | 6.8958$_{+}$ | 4.1250 | 4.1667 | 5.1667 | 3.4167 | **2.8958** | 5.5000 | 3.8333 |
| | Recall | 7.7292$_{+}$ | 3.4792 | **2.8125** | 3.1667 | 3.6458 | 4.3125 | 7.1667$_{+}$ | 3.6875 |
| | F-measure | 7.6875$_{+}$ | 3.7500 | 3.5000 | 3.7917 | 3.2917 | **3.1458** | 7.2083$_{+}$ | 3.6250 |
| | AUC | 7.6875$_{+}$ | 3.7917 | **2.5833** | 3.5417 | 3.6875 | 3.7500 | 7.0000$_{+}$ | 3.9583 |
| | G-mean | 7.7292$_{+}$ | 3.7500 | **2.6667** | 3.7083 | 3.4167 | 3.6875 | 6.9583$_{+}$ | 4.0833 |
| NB | Precision | 3.9375 | 4.4583 | 6.4167$_{+}$ | 5.2917$_{+}$ | 3.9167 | 4.2083 | 4.6458 | **3.1250** |
| | Recall | 5.0000 | 4.6042 | **2.6250**$_{-}$ | 4.6458 | 4.2292 | 5.1667 | 4.7292 | 5.0000 |
| | F-measure | 4.4167 | 4.4583 | 5.8750$_{+}$ | 4.6250 | 4.0000 | 5.0417 | 4.6667 | **2.9167** |
| | AUC | 4.6875 | 4.4167 | 5.2917$_{+}$ | 4.8750 | 4.2083 | 4.9583 | 4.6458 | **2.9167** |
| | G-mean | 4.9583 | 4.3750 | 5.9583$_{+}$ | 4.4167 | 3.9167 | 4.4583 | 4.9167 | **3.0000** |

**Table 3**
Average rankings for the case of *label noise*, for the *CART* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.00 | 4.0000 | 4.7917 | 6.3333$_{+}$ | 4.0417 | 4.5833 | **3.1667** | 5.9167$_{+}$ | **3.1667** |
| | 0.05 | 3.0000 | 5.3750 | 5.8333$_{+}$ | 5.7083$_{+}$ | 4.6250 | **2.5000** | 5.5417$_{+}$ | 3.4167 |
| | 0.10 | 2.8333 | 5.5000$_{+}$ | 5.8750$_{+}$ | 6.0000$_{+}$ | 4.9583 | **1.6667** | 6.1250$_{+}$ | 3.0417 |
| | 0.15 | 3.0833 | 5.2500 | 5.6250 | 5.7500$_{+}$ | 3.9167 | **1.7083** | 7.0417$_{+}$ | 3.6250 |
| | 0.20 | 2.9167 | 5.8333 | 5.1667 | 5.1667 | 3.8333 | **1.7083**$_{-}$ | 7.2083$_{+}$ | 4.1667 |
| Recall | 0.00 | 6.9375 | 4.2500 | **2.3542**$_{-}$ | 4.4792 | 4.0625 | 5.1250 | 3.3542 | 5.4375 |
| | 0.05 | 6.3542 | **3.2292**$_{-}$ | 3.3333$_{-}$ | 4.6042 | 3.5417$_{-}$ | 5.6042 | 3.2917$_{-}$ | 6.0417 |
| | 0.10 | 5.9167 | 4.0625 | 3.7083 | 4.2500 | 4.9792 | 5.5625 | **2.5000**$_{-}$ | 5.0208 |
| | 0.15 | 5.5000 | 4.2083 | 3.2292 | 4.6458 | 4.4375 | 6.5417 | **2.2917**$_{-}$ | 5.1458 |
| | 0.20 | 4.6667 | 4.3125 | 3.4583 | 4.6458 | 5.2917 | 7.2917$_{+}$ | **1.5417**$_{-}$ | 4.7917 |
| F-measure | 0.00 | 5.7083 | 5.0000 | 4.2083 | 4.1458 | 4.2708 | 4.9167 | **3.6250** | 4.1250 |
| | 0.05 | 3.7917 | 4.9583 | 5.5417 | 5.6667 | 4.2917 | **3.7083** | 4.0417 | 4.0000 |
| | 0.10 | 3.2083 | 5.6250$_{+}$ | 5.7083$_{+}$ | 5.4583$_{+}$ | 5.4583$_{+}$ | **2.7500** | 4.6250 | 3.1667 |
| | 0.15 | 3.3750 | 4.9583 | 5.3750 | 5.5417 | 4.3333 | **2.9583** | 5.9583$_{+}$ | 3.5000 |
| | 0.20 | **2.9167** | 5.6667 | 4.8750 | 4.8750 | 4.2500 | 3.0833 | 6.3333$_{+}$ | 4.0000 |
| AUC | 0.00 | 6.7083 | 4.0417 | **2.9583** | 4.6458 | 4.0625 | 5.0417 | 3.5417 | 5.0000 |
| | 0.05 | 4.6667 | 4.2500 | 4.5833 | 5.4583 | **3.7500** | 3.9583 | 4.0000 | 5.3333 |
| | 0.10 | 3.6250 | 5.1667 | 5.3958 | 5.3958 | 5.6250 | **3.1667** | 3.9583 | 3.6667 |
| | 0.15 | 4.0833 | 4.8333 | 4.7917 | 5.4583 | 4.3750 | **3.1667** | 5.5417 | 3.7500 |
| | 0.20 | 3.1667 | 5.8750 | 4.5833 | 5.2500 | 4.5000 | **3.1250** | 5.3750 | 4.1250 |
| G-mean | 0.00 | 6.7917 | 4.1667 | **2.9167**$_{-}$ | 4.4792 | 3.9792 | 5.0000 | 3.5000 | 5.1667 |
| | 0.05 | 4.9583 | 3.9583 | 4.1250 | 5.5833 | **3.8333** | 4.1667 | 3.8750 | 5.5000 |
| | 0.10 | 3.9583 | 5.1250 | 4.7500 | 5.1667 | 5.5000 | **3.4583** | 4.0417 | 4.0000 |
| | 0.15 | 4.2917 | 4.4583 | 4.5000 | 5.2917 | 4.0417 | 4.0417 | 5.5000 | **3.8750** |
| | 0.20 | **3.2500** | 5.5000 | 4.2083 | 4.8750 | 4.1667 | 3.7917 | 5.9583 | 4.2500 |

standard deviation equal to $\frac{1}{5} \times \textit{noise level} \times (\max - \min)$, with *max* and *min* being the upper and lower limits of a given feature, respectively. We considered the noise levels in $\{0.0, 0.1, \ldots, 0.5\}$.

We present the results of this stage of the experimental study in Tables 7–10, once again separately for each of the considered classifiers. Opposite to the case of the label noise, presence of feature noise made RBO achieve a comparatively worse results than the reference methods, for most of the classifiers and noise levels. Notably, when the data was affected by a significant amount of feature noise, performance of RBO combined with RBO decreased significantly, changing from being the best out of the evaluated methods to one of the worst. The only exception to this trend was the SVM classifier, for which an improvement in performance compared to the reference methods was observed for low to medium levels of feature noise.

## 5. Lessons learned

In this section we will summarize the research findings and observations that could be drawn from the experimental analysis:

- *RBO vs. other oversampling approaches.* The proposed RBO algorithm was designed in order to alleviate the limitations of existing SMOTE-based methods that took into account the minority class neighborhood, but ignored the information coming from the majority class distribution. This plays a crucial role when dealing with datasets characterized by high class overlapping or

**Table 4**
Average rankings for the case of *label noise*, for the *k-NN* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.00 | **2.0625**$_-$ | 5.2500 | 6.0000 | 5.2500 | 5.4167 | 4.3750 | 2.6458$_-$ | 5.0000 |
| | 0.05 | **1.6667**$_-$ | 5.9583 | 6.3750 | 4.9167 | 5.4167 | 3.4583 | 2.7083$_-$ | 5.5000 |
| | 0.10 | **1.6250**$_-$ | 5.8333 | 5.8333 | 6.0000 | 4.8750 | 2.9167$_-$ | 2.8750$_-$ | 6.0417 |
| | 0.15 | **1.7917**$_-$ | 5.2917 | 5.9167 | 6.5833 | 4.5000 | 2.4583$_-$ | 3.8750 | 5.5833 |
| | 0.20 | **1.9583**$_-$ | 5.2083 | 5.6250 | 6.5833 | 4.0000 | 2.2083$_-$ | 4.9583 | 5.4583 |
| Recall | 0.00 | 7.8958$_+$ | **2.7917** | 3.5000 | 3.6250 | 3.2500 | 4.2500 | 6.2708 | 4.4167 |
| | 0.05 | 7.9375$_+$ | 2.9167 | 4.2083 | **2.8542** | 3.1458 | 4.7083 | 5.9583 | 4.2708 |
| | 0.10 | 7.8958$_+$ | **2.7083** | 3.8542 | 3.1250 | 3.2292 | 5.4167 | 5.6042 | 4.1667 |
| | 0.15 | 7.7500$_+$ | **2.9167** | 3.4375 | 3.6042 | 3.6458 | 6.2500$_+$ | 4.8750 | 3.5208 |
| | 0.20 | 7.4167$_+$ | **2.8333** | 4.0000 | 3.2708 | 3.5833 | 6.6667$_+$ | 4.2083 | 4.0208 |
| F-measure | 0.00 | 5.1250 | 3.9167 | 5.3750 | **3.7083** | 4.4167 | 4.2917 | 4.7083 | 4.4583 |
| | 0.05 | 3.7083 | 5.3750 | 6.1667 | 4.0833 | 4.7083 | 3.5833 | **3.2500** | 5.1250 |
| | 0.10 | 3.1667$_-$ | 5.3750 | 5.8750 | 5.5000 | 4.2917 | 3.5833 | **2.6667**$_-$ | 5.5417 |
| | 0.15 | **2.9583**$_-$ | 4.7500 | 5.5833 | 6.1667 | 4.2083 | 3.2500 | 3.7917 | 5.2917 |
| | 0.20 | **2.5000**$_-$ | 4.8750 | 5.4583 | 6.0833 | 3.9167 | 3.2500 | 4.7083 | 5.2083 |
| AUC | 0.00 | 7.4167$_+$ | **2.6458** | 4.7917 | 3.0833 | 3.4583 | 3.9583 | 5.9167 | 4.7292 |
| | 0.05 | 5.9167 | 4.3750 | 6.1667 | 3.2083 | 4.0000 | **2.9583**$_-$ | 4.2500 | 5.1250 |
| | 0.10 | 4.2083 | 4.8333 | 5.8333 | 5.1250 | 3.7500 | **3.2500**$_-$ | 3.5417 | 5.4583 |
| | 0.15 | 3.7917 | 4.2083 | 5.2500 | 6.1667 | 4.0833 | **3.0833** | 4.2083 | 5.2083 |
| | 0.20 | **3.3333** | 4.4167 | 5.1250 | 5.7917 | 3.7500 | 3.5000 | 5.0000 | 5.0833 |
| G-mean | 0.00 | 7.3750$_+$ | **2.5000**$_-$ | 4.7083 | 3.4583 | 3.2500 | 3.7083 | 6.0833 | 4.9167 |
| | 0.05 | 6.2917 | 4.2500 | 6.0417 | 3.3333 | 3.8333 | **2.8750** | 4.4167 | 4.9583 |
| | 0.10 | 4.3750 | 4.6667 | 5.7500 | 5.1667 | 3.6667 | **3.2500** | 3.7500 | 5.3750 |
| | 0.15 | 3.8750 | 4.0417 | 5.2500 | 6.0417 | 3.8750 | **3.3333** | 4.7917 | 4.7917 |
| | 0.20 | 3.6667 | 4.4167 | 4.9167 | 5.8750 | **3.5000** | 3.5417 | 5.3750 | 4.7083 |

**Table 5**
Average rankings for the case of *label noise*, for the *SVM* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.00 | 6.8958$_+$ | 4.1250 | 4.1667 | 5.1667 | 3.4167 | **2.8958** | 5.5000 | 3.8333 |
| | 0.05 | 6.8750$_+$ | 3.9583 | 4.6667 | 5.1250 | 3.2917 | **2.6042** | 5.3542 | 4.1250 |
| | 0.10 | 6.6667$_+$ | 4.0417 | 4.4583 | 5.5833$_+$ | 3.5000 | **3.2292** | 5.1042 | 3.4167 |
| | 0.15 | 6.8125$_+$ | 4.0417 | 4.8750 | 5.2917 | 3.1667 | **2.9167** | 5.3125 | 3.5833 |
| | 0.20 | 5.9583 | 4.2917 | 4.0417 | 5.2083 | **2.7500** | 4.0000 | 5.5417 | 4.2083 |
| Recall | 0.00 | 7.7292$_+$ | 3.4792 | **2.8125** | 3.1667 | 3.6458 | 4.3125 | 7.1667$_+$ | 3.6875 |
| | 0.05 | 7.7708$_+$ | 3.2292 | 2.9583 | **2.5625** | 3.5417 | 5.5833$_+$ | 6.8542$_+$ | 3.5000 |
| | 0.10 | 7.7500$_+$ | 2.7917 | 3.7083 | **2.5208** | 3.5000 | 6.0833$_+$ | 6.7083$_+$ | 2.9375 |
| | 0.15 | 7.7292$_+$ | 2.7917 | 3.3750 | **2.2708** | 4.3750 | 6.3542$_+$ | 5.8542$_+$ | 3.2500 |
| | 0.20 | 7.4583$_+$ | 3.0208 | 3.2083 | **2.2083** | 4.5833 | 6.7708$_+$ | 5.4167$_+$ | 3.3333 |
| F-measure | 0.00 | 7.6875$_+$ | 3.7500 | 3.5000 | 3.7917 | 3.2917 | **3.1458** | 7.2083$_+$ | 3.6250 |
| | 0.05 | 7.7292$_+$ | 3.7917 | 3.6250 | 4.0000 | 3.4167 | **2.9375** | 6.8750$_+$ | 3.6250 |
| | 0.10 | 7.7500$_+$ | 3.5000 | 3.7500 | 4.5000 | **3.1667** | 3.6458 | 6.4375$_+$ | 3.2500 |
| | 0.15 | 7.6458$_+$ | 3.2083 | 4.0833 | 4.5417 | 3.2917 | 3.8125 | 6.3333$_+$ | **3.0833** |
| | 0.20 | 7.0417$_+$ | 3.4583 | 3.7500 | 4.3750 | **2.9583** | 4.5417 | 6.4167$_+$ | 3.4583 |
| AUC | 0.00 | 7.6875$_+$ | 3.7917 | **2.5833** | 3.5417 | 3.6875 | 3.7500 | 7.0000$_+$ | 3.9583 |
| | 0.05 | 7.7292$_+$ | 3.2500 | 3.7083 | 3.6042 | **3.2083** | 3.9375 | 7.1042$_+$ | 3.4583 |
| | 0.10 | 7.7083$_+$ | 3.2500 | 3.7083 | 4.2083 | **2.9583** | 4.1042 | 6.7292$_+$ | 3.3333 |
| | 0.15 | 7.6875$_+$ | **2.9167** | 3.9583 | 3.9167 | 3.1667 | 4.6042 | 6.7500$_+$ | 3.0000 |
| | 0.20 | 7.1250$_+$ | 3.0000 | 3.5833 | 3.8750 | **2.8750** | 5.4792$_+$ | 6.7292$_+$ | 3.3333 |
| G-mean | 0.00 | 7.7292$_+$ | 3.7500 | **2.6667** | 3.7083 | 3.4167 | 3.6875 | 6.9583$_+$ | 4.0833 |
| | 0.05 | 7.7292$_+$ | **3.2917** | 3.5833 | 3.5417 | **3.2917** | 4.1458 | 7.1250$_+$ | **3.2917** |
| | 0.10 | 7.7500$_+$ | **3.0417** | 3.7083 | 4.2083 | **3.0417** | 4.2292 | 6.7292$_+$ | 3.2917 |
| | 0.15 | 7.6875$_+$ | 2.8333 | 4.2083 | 3.9167 | 3.2083 | 4.6875 | 6.7500$_+$ | **2.7083** |
| | 0.20 | 7.1667$_+$ | **2.7917** | 3.7500 | 3.7500 | 2.9167 | 5.7083$_+$ | 6.6250$_+$ | 3.2917 |

small minority class disjuncts. Obviously, not all of imbalanced datasets display such characteristics. For standard datasets the current way of introducing new objects may be to limited to local areas, thus not empowering the minority class enough. This is both advantage and drawback of RBO, depending on the context. It also serves as an insight into why on a set of diverse benchmarks RBO does not always outperforms SMOTE-based methods. It should be viewed as a specific-purpose algorithm to be deployed when we expect to deal with specific learning difficulties embedded in the nature of data.

- *Role of base classifier in RBO*. RBO, as all of preprocessing methods for imbalanced data, can work with any type of classifier. We have examined our method with four different popular learners, showing that it can work well with all of them. However, it is interesting to observe that RBO works especially well with NB classifier, offering a significantly better improvement in

**Table 6**
Average rankings for the case of *label noise*, for the *NB* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.00 | 3.9375 | 4.4583 | 6.4167$_+$ | 5.2917$_+$ | 3.9167 | 4.2083 | 4.6458 | **3.1250** |
| | 0.05 | 3.3958 | 4.4583 | 6.0417$_+$ | 5.8333$_+$ | 5.0417 | 3.3750 | 4.6875 | **3.1667** |
| | 0.10 | **3.0208** | 4.9583 | 5.4167 | 5.9583$_+$ | 4.7500 | 3.9583 | 4.3542 | 3.5833 |
| | 0.15 | **3.2708** | 4.6250 | 4.7500 | 6.4167$_+$ | 4.9167 | 3.8333 | 4.6875 | 3.5000 |
| | 0.20 | **3.2500** | 4.5417 | 4.3750 | 6.0417$_+$ | 4.8750 | 4.0000 | 5.2917 | 3.6250 |
| Recall | 0.00 | 5.0000 | 4.6042 | **2.6250**$_-$ | 4.6458 | 4.2292 | 5.1667 | 4.7292 | 5.0000 |
| | 0.05 | 5.5625 | 3.7708 | 4.4375 | 3.8750 | **3.2708** | 4.7083 | 5.5208 | 4.8542 |
| | 0.10 | 5.8958 | **3.3958** | 4.1875 | 3.7083 | 3.4167 | 4.6458 | 5.7917 | 4.9583 |
| | 0.15 | 5.9375 | **3.0417** | 4.8958 | 3.5625 | 3.6875 | 4.6458 | 5.4375 | 4.7917 |
| | 0.20 | 6.3125 | 3.2500 | 4.1667 | **3.1875** | 3.9167 | 4.6042 | 5.7292 | 4.8333 |
| F-measure | 0.00 | 4.4167 | 4.4583 | 5.8750$_+$ | 4.6250 | 4.0000 | 5.0417 | 4.6667 | **2.9167** |
| | 0.05 | 3.8542 | 4.6250 | 5.5833$_+$ | 5.5000$_+$ | 4.7500 | 3.8750 | 4.8542 | **2.9583** |
| | 0.10 | 3.7292 | 4.8333 | 4.9167 | 5.7917$_+$ | 4.7917 | 4.1667 | 4.3542 | **3.4167** |
| | 0.15 | 3.6875 | 4.0833 | 4.5833 | 6.0000 | 4.6667 | **3.6667** | 5.3542 | 3.9583 |
| | 0.20 | **3.3750** | 4.1667 | 3.8333 | 5.5833 | 4.4583 | 4.3333 | 5.8750 | 4.3750 |
| AUC | 0.00 | 4.6875 | 4.4167 | 5.2917$_+$ | 4.8750 | 4.2083 | 4.9583 | 4.6458 | **2.9167** |
| | 0.05 | 4.7708 | 4.2083 | 5.2917$_+$ | 5.3750$_+$ | 4.7917 | 4.0000 | 4.4792 | **3.0833** |
| | 0.10 | 4.3333 | 4.7500 | 4.7500 | 5.5833 | 4.5833 | 4.3333 | 4.2917 | **3.3750** |
| | 0.15 | 4.0417 | 3.7083 | 4.5417 | 5.9167$_+$ | 4.9167 | 4.2500 | 5.1250 | **3.5000** |
| | 0.20 | 3.7500 | 4.2917 | **3.5417** | 5.1250 | 4.7083 | 4.9167 | 5.2917 | 4.3750 |
| G-mean | 0.00 | 4.9583 | 4.3750 | 5.9583$_+$ | 4.4167 | 3.9167 | 4.4583 | 4.9167 | **3.0000** |
| | 0.05 | 4.8958 | 4.2083 | 5.9167$_+$ | 4.7917 | 4.5000 | 3.5000 | 5.1458 | **3.0417** |
| | 0.10 | 5.1042 | 4.6667 | 5.4167$_+$ | 5.2917$_+$ | 4.0417 | 3.5417 | 4.8542 | **3.0833** |
| | 0.15 | 4.7708 | 3.7083 | 5.0417 | 5.4167 | 4.5833 | 3.6667 | 5.1875 | **3.6250** |
| | 0.20 | 4.4167 | 4.0833 | **3.9583** | 4.8333 | 4.3750 | 4.7083 | 5.4167 | 4.2083 |

**Table 7**
Average rankings for the case of *feature noise*, for the *CART* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.0 | 4.0000 | 4.7917 | 6.3333$_+$ | 4.0417 | 4.5833 | **3.1667** | 5.9167$_+$ | **3.1667** |
| | 0.1 | **3.0000** | 4.9583 | 5.6667 | 4.7083 | 4.8333 | 3.7083 | 5.3750 | 3.7500 |
| | 0.2 | **2.8333** | 4.7917 | 5.7500 | 4.9583 | 4.2917 | 4.1250 | 5.2917 | 3.9583 |
| | 0.3 | **2.9583** | 5.0833 | 4.6667 | 4.8333 | 5.4583 | 4.0833 | 5.3333 | 3.5833 |
| | 0.4 | **2.9583** | 5.5417 | 4.9167 | 4.9583 | 5.4167 | 3.7083 | 4.6667 | 3.8333 |
| | 0.5 | **2.8750** | 5.2083$_+$ | 4.6250 | 5.1667$_+$ | 5.4583$_+$ | 4.4583 | 5.2917$_+$ | 2.9167 |
| Recall | 0.0 | 6.9375 | 4.2500 | **2.3542**$_-$ | 4.4792 | 4.0625 | 5.1250 | 3.3542 | 5.4375 |
| | 0.1 | 7.1042 | 3.8542 | **3.0625**$_-$ | 4.7292 | 3.3125$_-$ | 3.9375 | 4.4375 | 5.5625 |
| | 0.2 | 7.2083 | 3.7292$_-$ | **3.0625**$_-$ | 3.8125$_-$ | 3.4167$_-$ | 4.0625 | 4.6042 | 6.1042 |
| | 0.3 | 7.3542 | 3.0417$_-$ | 4.3333 | 3.6875 | **2.9375**$_-$ | 4.0208 | 4.9375 | 5.6875 |
| | 0.4 | 7.2292 | 3.1458$_-$ | 4.5833 | 3.4375$_-$ | **3.0625**$_-$ | 3.7708 | 4.9792 | 5.7917 |
| | 0.5 | 7.6042 | 3.1042$_-$ | 4.5417 | 2.8750$_-$ | **2.7917**$_-$ | 4.0417 | 5.2292 | 5.8125 |
| F-measure | 0.0 | 5.7083 | 5.0000 | 4.2083 | 4.1458 | 4.2708 | 4.9167 | **3.6250** | 4.1250 |
| | 0.1 | 5.2500 | 4.5000 | 4.3333 | 4.8333 | 4.2083 | 4.5417 | **4.1250** | 4.2083 |
| | 0.2 | 5.3750 | 4.4167 | 4.9583 | 4.3750 | **3.7917** | 4.2917 | 3.9583 | 4.8333 |
| | 0.3 | 5.4583 | 4.1250 | 4.7500 | 4.3750 | 4.1667 | **4.0000** | 4.4167 | 4.7083 |
| | 0.4 | 6.4583 | 4.0417 | 4.4167 | 4.2500 | 3.8333 | **3.5000** | 4.5833 | 4.9167 |
| | 0.5 | 6.6667 | 3.5833 | 4.3750 | **3.4583** | 4.1250 | 4.1667 | 4.9583 | 4.6667 |
| AUC | 0.0 | 6.7083 | 4.0417 | **2.9583** | 4.6458 | 4.0625 | 5.0417 | 3.5417 | 5.0000 |
| | 0.1 | 6.4583 | 4.3333 | **3.3750** | 4.6250 | 3.5000 | 4.0417 | 4.5000 | 5.1667 |
| | 0.2 | 6.3750 | 4.1250 | 3.6667 | 4.0833 | **3.5417**$_-$ | 3.9583 | 4.5000 | 5.7500 |
| | 0.3 | 7.1250 | 3.3333 | 4.0417 | 3.9167 | **3.2500**$_-$ | 4.0000 | 4.9167 | 5.4167 |
| | 0.4 | 7.1250 | 3.5000 | 4.3333 | 3.8750 | **3.1667**$_-$ | 3.3333$_-$ | 5.1667 | 5.5000 |
| | 0.5 | 7.4167$_+$ | 3.2500 | 4.1667 | **3.0417**$_-$ | 3.3333 | 4.2083 | 5.4167 | 5.1667 |
| G-mean | 0.0 | 6.7917 | 4.1667 | **2.9167**$_-$ | 4.4792 | 3.9792 | 5.0000 | 3.5000 | 5.1667 |
| | 0.1 | 6.8333 | 4.0417 | 3.5000 | 4.6667 | **3.3333** | 3.7917 | 4.6667 | 5.1667 |
| | 0.2 | 6.7500 | 3.8333 | 3.7083$_-$ | 4.0833 | **3.2917**$_-$ | 3.8333 | 4.6250 | 5.8750 |
| | 0.3 | 7.0833 | 3.0417$_-$ | 4.3333 | 3.8750 | **2.9583**$_-$ | 4.0000 | 5.0833 | 5.6250 |
| | 0.4 | 7.2083 | 3.2917$_-$ | 4.5000 | 3.7083 | **3.1250**$_-$ | 3.2500$_-$ | 5.3750 | 5.5417 |
| | 0.5 | 7.5417 | 3.2500$_-$ | 4.2500 | **2.9167**$_-$ | 3.0000$_-$ | 3.9583 | 5.6250 | 5.4583 |

**Table 8**
Average rankings for the case of *feature noise*, for the *k-NN* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.0 | **2.0625**$_{-}$ | 5.2500 | 6.0000 | 5.2500 | 5.4167 | 4.3750 | 2.6458$_{-}$ | 5.0000 |
| | 0.1 | **1.6875**$_{-}$ | 5.6667 | 5.3750 | 5.5417 | 5.5417 | 4.5833 | 3.0625 | 4.5417 |
| | 0.2 | **2.2708**$_{-}$ | 5.6250 | 4.7917 | 5.4167 | 5.9167 | 4.5833 | 2.6458$_{-}$ | 4.7500 |
| | 0.3 | **2.2292**$_{-}$ | 5.6250 | 4.2500 | 5.5000 | 5.8333 | 4.7083 | 3.2292 | 4.6250 |
| | 0.4 | **2.1667**$_{-}$ | 6.5000 | 4.4583 | 5.2083 | 5.2917 | 5.1667 | 2.6667 | 4.5417 |
| | 0.5 | **2.0417** | 5.3333 | 4.3333 | 5.9167$_{+}$ | 5.9583$_{+}$ | 5.0000 | 3.6250 | 3.7917 |
| Recall | 0.0 | 7.8958$_{+}$ | **2.7917** | 3.5000 | 3.6250 | 3.2500 | 4.2500 | 6.2708 | 4.4167 |
| | 0.1 | 7.8542$_{+}$ | **2.6250** | 4.2083 | 3.4792 | 2.9583 | 4.1458 | 6.3542 | 4.3750 |
| | 0.2 | 7.8750$_{+}$ | **2.4792**$_{-}$ | 5.0000 | 3.1042 | 2.6250$_{-}$ | 3.5833 | 6.4583 | 4.8750 |
| | 0.3 | 7.8750$_{+}$ | **2.2708**$_{-}$ | 5.6042 | 3.1250 | 2.8750 | 3.2708 | 6.2917 | 4.6875 |
| | 0.4 | 7.8750$_{+}$ | **2.3333**$_{-}$ | 5.6667 | 2.8750 | 2.3750$_{-}$ | 3.6458 | 6.4583 | 4.7708 |
| | 0.5 | 7.8750$_{+}$ | **2.2292**$_{-}$ | 5.7500 | 3.1250$_{-}$ | 2.2500$_{-}$ | 3.1875$_{-}$ | 6.2500 | 5.3333 |
| F-measure | 0.0 | 5.1250 | 3.9167 | 5.3750 | **3.7083** | 4.4167 | 4.2917 | 4.7083 | 4.4583 |
| | 0.1 | 5.0833 | 4.3333 | 5.0417 | **4.1250** | 4.1667 | 4.5417 | 4.5000 | 4.2083 |
| | 0.2 | 5.3750 | 4.0000 | 5.4583 | **3.7917** | 4.3750 | 4.1667 | 4.5833 | 4.2500 |
| | 0.3 | 5.6667 | 4.1667 | 4.7500 | 4.3750 | 4.6250 | **4.0833** | 4.1250 | 4.2083 |
| | 0.4 | 5.6250 | 4.6250 | 4.4583 | 3.7083 | **3.6667** | 5.0833 | 4.4583 | 4.3750 |
| | 0.5 | 6.0833 | **3.3333** | 4.9583 | 4.5417 | 4.3750 | 4.3750 | 4.2500 | 4.0833 |
| AUC | 0.0 | 7.4167$_{+}$ | **2.6458** | 4.7917 | 3.0833 | 3.4583 | 3.9583 | 5.9167 | 4.7292 |
| | 0.1 | 7.2917$_{+}$ | **2.9792** | 4.7292 | 3.5417 | 3.0000 | 4.2083 | 5.7500 | 4.5000 |
| | 0.2 | 7.0000 | 3.0417 | 5.3750 | 3.2500 | **2.9167** | 3.3750 | 6.0833 | 4.9583 |
| | 0.3 | 7.1667$_{+}$ | **2.6667** | 5.8333 | 3.5000 | 3.4167 | 3.0417 | 5.7500 | 4.6250 |
| | 0.4 | 7.3333$_{+}$ | 2.8750$_{-}$ | 5.4583 | 3.2500 | **2.5833**$_{-}$ | 3.5000 | 5.9167 | 5.0833 |
| | 0.5 | 7.2917$_{+}$ | **2.3333**$_{-}$ | 5.5417 | 3.9167 | 3.0417 | 3.0000 | 5.9167 | 4.9583 |
| G-mean | 0.0 | 7.3750$_{+}$ | **2.5000**$_{-}$ | 4.7083 | 3.4583 | 3.2500 | 3.7083 | 6.0833 | 4.9167 |
| | 0.1 | 7.2083$_{+}$ | 2.8750 | 4.8750 | 3.6667 | **2.7917** | 4.2083 | 5.9167 | 4.4583 |
| | 0.2 | 7.0417 | 2.9583 | 5.4583 | 3.5000 | **2.7500**$_{-}$ | 3.1667 | 6.1250 | 5.0000 |
| | 0.3 | 7.2500$_{+}$ | **2.7083** | 5.5833 | 3.4583 | 3.3750 | 3.0000 | 5.9583 | 4.6667 |
| | 0.4 | 7.3333$_{+}$ | 2.9583 | 5.2917 | 3.4167 | **2.4583**$_{-}$ | 3.4583 | 6.0833 | 5.0000 |
| | 0.5 | 7.3333$_{+}$ | **2.4583**$_{-}$ | 5.5000 | 4.0833 | 2.9167 | 2.9167 | 5.9167 | 4.8750 |

**Table 9**
Average rankings for the case of *feature noise*, for the *SVM* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.0 | 6.8958$_{+}$ | 4.1250 | 4.1667 | 5.1667 | 3.4167 | **2.8958** | 5.5000 | 3.8333 |
| | 0.1 | 6.8958$_{+}$ | 3.7917 | 4.3750 | 4.9583 | 3.6667 | **3.0208** | 5.7083 | 3.5833 |
| | 0.2 | 6.9167$_{+}$ | 3.9583 | 4.5000 | 4.6667 | 3.5833 | **3.1458** | 5.3958 | 3.8333 |
| | 0.3 | 6.9167$_{+}$ | 3.9583 | 4.1250 | 4.8750 | **3.5000** | 3.5208 | 5.4792 | 3.6250 |
| | 0.4 | 6.9167$_{+}$ | 4.1250 | 4.3750 | 4.4583 | **3.3750** | 3.5625 | 5.4375 | 3.7500 |
| | 0.5 | 6.9167$_{+}$ | 3.7917 | 3.8750 | 4.8333 | 4.0000 | **2.9375** | 5.6042 | 4.0417 |
| Recall | 0.0 | 7.7292$_{+}$ | 3.4792 | **2.8125** | 3.1667 | 3.6458 | 4.3125 | 7.1667$_{+}$ | 3.6875 |
| | 0.1 | 7.7500$_{+}$ | 3.4167 | **3.0417** | 3.2292 | 3.5000 | 4.5208 | 7.1458$_{+}$ | 3.3958 |
| | 0.2 | 7.7708$_{+}$ | 4.0000 | **2.8750** | 3.1458 | 3.3542 | 4.6250 | 7.0833$_{+}$ | 3.1458 |
| | 0.3 | 7.7708$_{+}$ | 3.6458 | **2.8750** | 3.1250 | 3.6458 | 4.6667 | 6.8958$_{+}$ | 3.3750 |
| | 0.4 | 7.7708$_{+}$ | 3.4792 | **3.0625** | 3.3333 | 3.1875 | 4.3333 | 6.8125$_{+}$ | 4.0208 |
| | 0.5 | 7.7708$_{+}$ | 3.5625 | 3.2708 | **2.9375** | 3.3542 | 4.7292 | 6.8958$_{+}$ | 3.4792 |
| F-measure | 0.0 | 7.6875$_{+}$ | 3.7500 | 3.5000 | 3.7917 | 3.2917 | **3.1458** | 7.2083$_{+}$ | 3.6250 |
| | 0.1 | 7.7500$_{+}$ | 3.7917 | 3.3750 | 4.0000 | 3.3750 | 3.3125 | 7.1042$_{+}$ | **3.2917** |
| | 0.2 | 7.7292$_{+}$ | 3.6667 | 3.3750 | 4.1667 | **3.1667** | 3.5625 | 6.9583$_{+}$ | 3.3750 |
| | 0.3 | 7.7292$_{+}$ | 3.5833 | 3.1667 | 4.1250 | 3.4583 | 3.8125 | 7.0833$_{+}$ | **3.0417** |
| | 0.4 | 7.7708$_{+}$ | 3.4583 | 3.4167 | 3.4167 | **3.3750** | 3.8958 | 7.0417$_{+}$ | 3.6250 |
| | 0.5 | 7.7708$_{+}$ | 3.4583 | 3.2500 | 3.8750 | **3.1250** | 3.6875 | 7.0417$_{+}$ | 3.7917 |
| AUC | 0.0 | 7.6875$_{+}$ | 3.7917 | **2.5833** | 3.5417 | 3.6875 | 3.7500 | 7.0000$_{+}$ | 3.9583 |
| | 0.1 | 7.7500$_{+}$ | 3.9167 | **2.8750** | 4.1250 | 3.3750 | 3.6042 | 6.9375$_{+}$ | 3.4167 |
| | 0.2 | 7.7708$_{+}$ | 3.6875 | 3.3125 | 3.4583 | **3.1250** | 4.2708 | 6.8750$_{+}$ | 3.5000 |
| | 0.3 | 7.7708$_{+}$ | 3.6250 | **3.0833** | 3.7500 | 3.1667 | 4.3125 | 7.0000$_{+}$ | 3.2917 |
| | 0.4 | 7.7708$_{+}$ | 3.2917 | 3.2500 | 3.9167 | **3.1667** | 4.0208 | 7.0417$_{+}$ | 3.5417 |
| | 0.5 | 7.7708$_{+}$ | 3.4375 | **3.1250** | 3.5000 | 3.1458 | 4.2708 | 7.0833$_{+}$ | 3.6667 |
| G-mean | 0.0 | 7.7292$_{+}$ | 3.7500 | **2.6667** | 3.7083 | 3.4167 | 3.6875 | 6.9583$_{+}$ | 4.0833 |
| | 0.1 | 7.7500$_{+}$ | 3.8333 | **2.7917** | 4.0833 | 3.4167 | 3.7292 | 6.9375$_{+}$ | 3.4583 |
| | 0.2 | 7.7708$_{+}$ | 3.8333 | 3.2083 | 3.5000 | **3.0417** | 4.2292 | 6.9167$_{+}$ | 3.5000 |
| | 0.3 | 7.7708$_{+}$ | 3.5833 | 3.1667 | 3.9583 | 3.2500 | 4.2292 | 6.9167$_{+}$ | **3.1250** |
| | 0.4 | 7.7708$_{+}$ | 3.2500 | 3.0833 | 4.1250 | **3.0000** | 4.0208 | 7.1250$_{+}$ | 3.6250 |
| | 0.5 | 7.7708$_{+}$ | **3.2083** | 3.2500 | 3.8333 | **3.2083** | 4.1458 | 7.0417$_{+}$ | 3.5417 |

**Table 10**
Average rankings for the case of *feature noise*, for the *NB* classifier. Methods that achieved significantly different results (according to Shaffer's post-hoc test) than RBO where denoted in subscript: with + sign for methods compared to which RBO achieved a better results, and - sign for methods compared to which RBO achieved worse results.

| | Level | Base | SMOTE | ADASYN | Bord | SMOTE+TL | SMOTE+ENN | NCL | RBO |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.0 | 3.9375 | 4.4583 | 6.4167$_{+}$ | 5.2917$_{+}$ | 3.9167 | 4.2083 | 4.6458 | **3.1250** |
| | 0.1 | **2.4375**$_{-}$ | 4.4583 | 6.3333 | 5.7083 | 4.1250 | 3.8333 | 4.1458 | 4.9583 |
| | 0.2 | **2.2708**$_{-}$ | 4.6250 | 5.8750 | 6.0417 | 5.0000 | 3.6667 | 3.5625 | 4.9583 |
| | 0.3 | **2.5625** | 5.0000 | 5.5000 | 5.2500 | 4.2917 | 4.5833 | 4.1042 | 4.7083 |
| | 0.4 | **2.6458** | 4.9167 | 5.0000 | 5.7917 | 4.8750 | 4.5417 | 3.7708 | 4.4583 |
| | 0.5 | **2.7708**$_{-}$ | 5.2083 | 4.9167 | 5.2083 | 5.2083 | 4.0833 | 3.4375 | 5.1667 |
| Recall | 0.0 | 5.0000 | 4.6042 | **2.6250**$_{-}$ | 4.6458 | 4.2292 | 5.1667 | 4.7292 | 5.0000 |
| | 0.1 | 6.0625 | 4.0208 | **2.6250** | 4.9167 | 4.1250 | 4.8333 | 5.1250 | 4.2917 |
| | 0.2 | 6.6250$_{+}$ | 3.4792 | **2.5208** | 5.0417 | 3.8958 | 5.0417 | 5.2917 | 4.1042 |
| | 0.3 | 6.7708$_{+}$ | 3.6458 | **3.0625** | 4.6042 | 3.8542 | 4.6875 | 5.1667 | 4.2083 |
| | 0.4 | 6.9167$_{+}$ | 3.7083 | 3.8333 | 4.2500 | **3.3958** | 4.4375 | 5.3542 | 4.1042 |
| | 0.5 | 7.0625$_{+}$ | **3.2500** | 4.0000 | 4.2083 | 3.7083 | 4.0625 | 5.2708 | 4.4375 |
| F-measure | 0.0 | 4.4167 | 4.4583 | 5.8750$_{+}$ | 4.6250 | 4.0000 | 5.0417 | 4.6667 | **2.9167** |
| | 0.1 | **3.0208** | 4.4167 | 5.4583 | 5.4167 | 4.2917 | 4.7917 | 3.6042 | 5.0000 |
| | 0.2 | **3.1458** | 4.4167 | 5.0417 | 5.5417 | 4.7917 | 4.6250 | 3.3958 | 5.0417 |
| | 0.3 | 3.6042 | 4.7083 | 4.7500 | 4.7083 | 4.5833 | 5.0833 | **3.5625** | 5.0000 |
| | 0.4 | 4.1458 | 4.4167 | 4.7500 | 5.1667 | 4.6250 | 4.9583 | **3.4375** | 4.5000 |
| | 0.5 | 4.6042 | 4.4167 | 4.7500 | 4.8333 | 4.6667 | 4.3333 | **3.3125** | 5.0833 |
| AUC | 0.0 | 4.6875 | 4.4167 | 5.2917$_{+}$ | 4.8750 | 4.2083 | 4.9583 | 4.6458 | **2.9167** |
| | 0.1 | **3.1042** | 4.3750 | 4.3333 | 5.8750 | 4.5417 | 5.0000 | 3.6042 | 5.1667 |
| | 0.2 | 4.0625 | 4.2500 | 3.7083 | 5.7500 | 4.7500 | 5.0833 | **3.6042** | 4.7917 |
| | 0.3 | 4.7708 | 4.4583 | **3.5208** | 5.2917 | 4.4583 | 5.2083 | 3.9792 | 4.3125 |
| | 0.4 | 5.1458 | 4.6667 | **3.6250** | 5.3750 | 4.4167 | 4.7083 | 4.2292 | 3.8333 |
| | 0.5 | 5.6875 | 4.1250 | **3.8750** | 4.8750 | 4.3750 | 4.0833 | 4.3958 | 4.5833 |
| G-mean | 0.0 | 4.9583 | 4.3750 | 5.9583$_{+}$ | 4.4167 | 3.9167 | 4.4583 | 4.9167 | **3.0000** |
| | 0.1 | **3.4375** | 4.0417 | 5.3333 | 5.6250 | 4.0000 | 4.5833 | 3.8125 | 5.1667 |
| | 0.2 | 4.4375 | **4.1250** | 4.2500 | 5.5417 | 4.2500 | 4.2917 | 4.1458 | 4.9583 |
| | 0.3 | 5.0625 | 4.0833 | **3.9167** | 5.2917 | 3.9583 | 4.4167 | 4.6458 | 4.6250 |
| | 0.4 | 5.6875 | 4.3333 | 4.0000 | 5.0833 | 4.3333 | **3.8333** | 4.8958 | **3.8333** |
| | 0.5 | 6.1042 | 4.3750 | 4.1667 | 4.6250 | 4.0417 | **3.5417** | 4.5625 | 4.5833 |

performance than any other examined preprocessing algorithm. This can be contributed to the way in which RBO introduces artificial objects. As their location is based on both minority and majority class potentials, the risk of disrupting class probabilities is reduced. Therefore, RBO allows NB for a better estimation of its parameters, while minimizing the risk of dataset shift.

- *RBO for imbalanced data with class label noise*. RBO starts displaying its usefulness of handling class label noise even with small noise ratios (starting from 5%). It tends to outperform all other oversampling approaches, regardless of the base classifier used. The only solution that is competitive to RBO is SMOTE+ENN. This can be explained by the fact that ENN cleans dataset, thus reducing the impact of noisy instances. At the same time we must point out that any cleaning procedure can be applied with RBO, thus potentially leading to its further improvements. Therefore, our future efforts will concentrate on developing a new data cleaning algorithm that will use potential information, to provide a good synergy with RBO philosophy.
- *RBO for imbalanced data with feature noise*. RBO does not display as constant performance with feature noise as with class label noise. Here, we can observe stronger dependencies on the base classifier. It works well with NB and SVM, while suffering from reduced performance when coupled with CART and *k*-NN. Here, preprocessing algorithms combined with data cleaning methods tend to perform better than RBO, leading to similar conclusions about future extension of our algorithm as discussed in the label noise paragraph. However, when compared with SMOTE, RBO tends to perform on par, or even superb to it. This shows that the potential-based oversampling may also be an useful direction for handling feature noise, but additional improvements are required in this area.

## 6. Conclusions

We have discussed the issue of learning efficient classifiers from imbalanced and noisy imbalanced data. We have concentrated on data preprocessing algorithms, specifically oversampling-based, for alleviating the skewed distributions. The limitations of existing oversampling methods that are based on popular SMOTE approach were discussed. Their main drawback was identified as relying on neighborhood analysis which does not take into account the majority class instances, local data characteristics and learning difficulty in the given area of the feature space. Additionally, we have pointed out the challenges behind the presence of noise in either class labels or features. The strong negative impact of noise on SMOTE-based algorithms was discussed and illustrated using a 2D example.

Based on our observations, we have introduced a new oversampling algorithm for imbalanced and noisy data that alleviates the limitations of its predecessors. Instead of relying on neighborhood search, we proposed to use radial-based functions to estimate local distributions of both minority and majority instances. The calculation of joint potentials in each area of the feature space allowed us to detect regions that posed the highest difficulty to the classifier and introduce new objects in this specific region. This has lead to more guided oversampling procedure that does not increase class overlapping and is able to better allocate artificial objects than uniform solutions (e.g., SMOTE). Additionally, by using density-based oversampling we were able to increase the robustness of our method to noise. Experimental results on diverse set of benchmark datasets with and without noise proved that our RBO algorithm could be an useful alternative for SMOTE-based solutions, especially when dealing with complex data distributions.

Results obtained in this paper encourage us to continue future works on alternative approaches to oversampling imbalanced data. We are going to extend RBO algorithm to multi-class problems, as well as embed it into hybrid architectures with inbuilt mechanisms.

## Acknowledgments

This work was supported by the Polish National Science Center under the grant no. UMO-2015/19/B/ST6/01597 as well as the PLGrid Infrastructure.

## References

[1] Y. Sun, A.K.C. Wong, M.S. Kamel, Classification of imbalanced data: A review, Int. J. Pattern Recogn. Artif. Intell. 23 (4) (2009) 687–719.
[2] S. Wang, L.L. Minku, X. Yao, A systematic study of online class imbalance learning with concept drift, CoRR abs/1703.06683 (2017).
[3] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[4] W. Khreich, E. Granger, A. Miri, R. Sabourin, Iterative boolean combination of classifiers in the ROC space: an application to anomaly detection with hmms, Pattern Recogn. 43 (8) (2010) 2732–2752.
[5] Z. Yang, W.H. Tang, A. Shintemirov, Q.H. Wu, Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 39 (6) (2009) 597–610.
[6] Y.-H. Liu, Y.-T. Chen, Face recognition using total margin-based adaptive fuzzy support vector machines, IEEE Trans. Neural Netw. 18 (1) (2007) 178–192.
[7] K. Napierala, J. Stefanowski, Identification of different types of minority class examples in imbalanced data, in: Proceedings of the Hybrid Artificial Intelligent Systems, in: Lecture Notes in Computer Science, 7209, Springer Berlin Heidelberg, 2012, pp. 139–150.
[8] X.-w. Chen, M. Wasikowski, Fast: A ROC-based feature selection metric for small samples and imbalanced data classification problems, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 124–132.
[9] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[10] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-Level-SMOTE: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Proceedings of the Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference 2009, Bangkok, Thailand, 2009, pp. 475–482.
[11] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: One-sided selection, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 179–186.
[12] A. Cano, D.T. Nguyen, S. Ventura, K.J. Cios, ur-caim: improved CAIM discretization for unbalanced and balanced data, Soft. Comput. 20 (1) (2016) 173–188.
[13] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artif. Intell. Rev. 22 (3) (2003) 177–210, doi:10.1007/s10462-004-0751-8.
[14] V. Lopez, A. Fernandez, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (7) (2012) 6585–6608.
[15] D.A. Cieslak, T.R. Hoens, N.V. Chawla, W.P. Kegelmeyer, Hellinger distance decision trees are robust and skew-insensitive, Data Min. Knowl. Discov. 24 (1) (2012) 136–158.
[16] A. Cano, A. Zafra, S. Ventura, Weighted data gravitation classification for standard and imbalanced data, IEEE Trans. Cybern. 43 (6) (2013) 1672–1687.
[17] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progress Artif. Intell. 5 (4) (2016) 221–232.
[18] C. Bellinger, C. Drummond, N. Japkowicz, Beyond the boundaries of SMOTE - a framework for manifold-based synthetically oversampling, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, Part I, 2016, pp. 248–263.
[19] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, in: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, 2009, pp. 324–331.
[20] H. He, Y. Bai, E.A. Garcia, S. Li, Adasyn: adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of the International Joint Conference on Neural Networks, Part of the IEEE World Congress on Computational Intelligence,Hong Kong, China, 2008, pp. 1322–1328. June 1–6, 2008
[21] S. Chen, H. He, E.A. Garcia, RAMOBoost: ranked minority oversampling in boosting, IEEE Trans. Neural Netw. 21 (10) (2010) 1624–1642.
[22] H. Han, W. Wang, B. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of the Advances in Intelligent Computing, International Conference on Intelligent Computing Hefei, China, Part I, 2005, pp. 878–887.
[23] T. Maciejewski, J. Stefanowski, Local neighbourhood extension of SMOTE for mining imbalanced data, in: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, Paris, France, 2011, pp. 104–111.
[24] M. Koziarski, B. Krawczyk, M. Woźniak, Radial-based approach to imbalanced data oversampling, in: Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 318–327.
[25] P. Branco, L. Torgo, R.P. Ribeiro, A survey of predictive modeling on imbalanced domains, ACM Comput. Surveys 49 (2) (2016) 31:1–31:50.
[26] H. Guo, Y. Li, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.
[27] M. Wasikowski, X. Chen, Combating the small sample class imbalance problem using feature selection, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1388–1400.
[28] J.G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N.V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recogn. 45 (1) (2012) 521–530.
[29] S. Ertekin, J. Huang, L. Bottou, C.L. Giles, Learning on the border: active learning in imbalanced data classification, in: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, 2007, pp. 127–136. November 6–10
[30] C. Bellinger, S. Sharma, O.R. Zaïane, N. Japkowicz, Sampling a longer life: Binary versus one-class classification revisited, in: Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications, LIDTA@PKDD/ECML Skopje, Macedonia, 2017, pp. 64–78. 22 September 2017,
[31] E. Jamison, I. Gurevych, Needle in a haystack: Reducing the costs of annotating rare-class instances in imbalanced datasets, in: Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 28, Cape Panwa Hotel, Phuket, Thailand, 2014, pp. 244–253. December 12–14
[32] T. Razzaghi, P. Xanthopoulos, O. Seref, Constraint relaxation, cost-sensitive learning and bagging for imbalanced classification problems with outliers, Optim. Lett. 11 (5) (2017) 915–928.
[33] P. Skryjomski, B. Krawczyk, Influence of minority class instance types on SMOTE imbalanced data oversampling, in: Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications, LIDTA@PKDD/ECML Skopje, Macedonia, 2017, pp. 7–21. 22 September 2017
[34] R.Y. Wang, V.C. Storey, C.P. Firth, A framework for analysis of data quality research, IEEE Trans. Knowl. Data Eng. 7 (4) (1995) 623–640, doi:10.1109/69.404034.
[35] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowl. Inf. Syst. 38 (1) (2014) 179–206.
[36] G. Lugosi, Learning with an unreliable teacher, Pattern Recogn. 25 (1) (1992) 79–87, doi:10.1016/0031-3203(92)90008-7.
[37] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, INFFC: an iterative class noise filter based on the fusion of classifiers with noise sensitivity control, Inf. Fus. 27 (2016) 19–32.
[38] M.A. Hernández, S.J. Stolfo, Real-world data is dirty: Data cleansing and the merge/purge problem, Data Min. Knowl. Discov. 2 (1) (1998) 9–37, doi:10.1023/A:1009761603038.
[39] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, in: Proceedings of the Twentieth International Conference on International Conference on Machine Learning, in: ICML'03, AAAI Press, 2003, pp. 920–927.
[40] M.S. Donaldson, J.M. Corrigan, L.T. Kohn, et al., To ERR is Human: Building a Safer Health System, 6, National Academies Press, 2000.
[41] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn. Syst. 25 (5) (2014) 845–869, doi:10.1109/tnnls.2013.2292894.
[42] J.W. Grzymala-Busse, W.J. Grzymala-Busse, Handling Missing Attribute Values, Springer US, Boston, MA, pp. 37–57.
[43] B. Tang, H. He, Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, 2015, pp. 664–671. May 25–28, 2015
[44] M. Pérez-Ortiz, P.A. Gutiérrez, P. Tiño, C. Hervás-Martínez, Oversampling the minority class in the feature space, IEEE Trans. Neural Netw. Learn. Syst. 27 (9) (2016) 1947–1961.
[45] M. Gao, X. Hong, S. Chen, C.J. Harris, Probability density function estimation based over-sampling for imbalanced two-class problems, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 2012, pp. 1–8. June 10–15, 2012
[46] J.A. Sáez, J. Luengo, J. Stefanowski, F. Herrera, SMOTE-IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, Inf. Sci. 291 (2015) 184–203.
[47] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework., J. Mult. Valued Log. Soft. Comput. 17 (2011).
[48] E. Alpaydin, Combined 5 × 2 cv F test for comparing supervised classification learning algorithms, Neural Comput. 11 (8) (1999) 1885–1892.
[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.
[50] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[51] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of the International Conference on Intelligent Computing, Springer, 2005, pp. 878–887.

[52] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorat. Newslett. 6 (1) (2004) 20–29.
[53] I. Tomek, Two modifications of CNN, IEEE Trans. Syst. Man Cybern. 6 (1976) 769–772.
[54] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[55] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1322–1328.
[56] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Proceedings of the Conference on Artificial Intelligence in Medicine in Europe, Springer, 2001, pp. 63–66.
[57] G. Lemaitre, F. Nogueira, C.K. Aridas, Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning, J. Mach. Learn. Res. 18 (17) (2017) 1–5.

**Michał Koziarski** received M.Sc. degree in computer science from the Wrocaw University of Science and Technology, Poland, in 2016. Currently, he is a Ph.D. student at the Department of Electronics of the AGH University of Science and Technology, Poland. His research interests include computer vision, neural networks and imbalanced data classification.

**Bartosz Krawczyk** is an assistant professor in the Department of Computer Science, Virginia Commonwealth University, Richmond VA, USA, where he heads the Machine Learning and Stream Mining Lab. He obtained his MSc and PhD degrees from Wroclaw University of Science and Technology, Wroclaw, Poland, in 2012 and 2015 respectively. His research focuses on machine learning, data stream mining, ensemble learning, class imbalance, one-class classifiers, and interdisciplinary applications of these methods. He has authored 50+ JCR journal papers and 100+ contributions to conferences. Dr. Krawczyk received prestigious awards for his scientific achievements like IEEE Richard Merwin Scholarship, IEEE Outstanding Leadership Award, and Best PhD Thesis Award from Polish Artificial Intelligence Society among others. He served as a Guest Editor in four journal special issues and as a chair of twelve special session and workshops. He is a member of Program Committee for over 50 international conferences and a reviewer for 30 journals.

**Michał Woźniak** is a professor of computer science at the Department of Systems and Computer Networks, Wrocaw University of Science and Technology, Poland. He received M.Sc. degree in biomedical engineering from the Wrocław University of Technology in 1992, and Ph.D. and D.Sc. (habilitation) degrees in computer science in 1996 and 2007, respectively, from the same university. In 2015 he was nominated as the professor by President of Poland. His research focuses on machine learning, compound classification methods, classifier ensembles, data stream mining, and imbalanced data processing. Prof. Woźniak has been involved in research projects related to the above-mentioned topics and has been a consultant of several commercial projects for well-known Polish companies and public administration. He has published over 260 papers and three books. His recent one Hybrid classifiers: Method of Data, Knowledge, and Data Hybridization was published by Springer in 2014. Prof. Woźniak was awarded with numerous prestigious awards for his scientific achievements as IBM Smarter Planet Faculty Innovation Award (twice) or IEEE Outstanding Leadership Award, and several best paper awards of the prestigious conferences. He serves as program committee chairs and member for the numerous scientific events and prepared several special issues as the guest editor. He is the member of the editorial board of the high ranked journals as *Information Fusion (Elsevier), Applied Soft Computing (Elsevier), and Engineering Applications of Artificial Intelligence (Elsevier)*. Prof. Woźniak is a senior member of the IEEE.