Journal of Intelligent Information Systems (2023) 61:299–324
https://doi.org/10.1007/s10844-022-00763-z

MaMiPot: a paradigm shift for the classification of imbalanced data

Hossein Ghaderi Zefrehi[1] **· Hakan Altınçay**[1]

Received: 5 June 2022 / Revised: 1 November 2022 / Accepted: 3 November 2022 /
Published online: 7 December 2022

Abstract
The most frequently used approach for imbalance learning is resampling. In this technique, the minority class is oversampled. Most of the algorithms used for this purpose are variants of the well-known algorithm named as SMOTE, which is based on creating synthetic samples on the lines connecting selected minority instances. The variants are mainly designed to alleviate the shortcomings of SMOTE where, the idea of generating synthetic samples to avoid the bias on the majority class is preserved. This study aims at following a different path. Instead of balancing, the proposed approach is based on repositioning the samples. In particular, the classifier is enforced to learn the decision regions of the minority class by repositioning the majority class samples. Consequently, the regions in which positive instances exist will not be outnumbered the majority class samples. Hence the classifier labels these regions as the minority class. To tackle the outliers, the minority samples are repositioned in small steps to avoid distorting the original distribution. The potential of the proposed repositioning scheme is also evaluated as a preprocessing algorithm for SMOTE. The experiments that are conducted on 52 datasets from the KEEL repository have shown that the proposed approach is highly effective, when evaluated in terms of F-score, G-mean and AUC.

Keywords Imbalance learning · Balancing · Repositioning · SMOTE · Binary classification

1 Introduction

In binary classification, the number of samples in one class may be much smaller than the other, leading to the so-called class imbalance problem (He & Ma, 2013; Fernández et al., 2018a). The class with smaller number of samples is generally labeled as the minority (or,

✉ Hossein Ghaderi Zefrehi
hossein.zefrehi@emu.edu.tr

Hakan Altınçay
hakan.altincay@emu.edu.tr

1 Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus, via Mersin 10, Turkey

positive) class and the other as majority (or, negative). When other distribution-dependent factors such as availability of rare instances representing infrequent sub-concepts and overlaps between minority and majority classes are combined with the bias on the majority class due to imbalance, the test samples are generally classified as the majority class (Napierala & Stefanowski, 2016). Most of the conventionally used learners achieve poor performance on the minority class since they compute decision regions by considering the number of classification errors on the whole training data (Jo & Japkowicz, 2004). The imbalance problem is generally tackled using two main approaches (Gong & Kim, 2017). The first is based on *resampling* the minority samples to balance the training data (Bej et al., 2021). After balancing by resampling, some parts of the feature space where the minority samples were outnumbered by the negatives will be learned as positive decision regions. Numerous techniques have been proposed for synthetic sample generation, most of which are variants of a well known algorithm named as synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002; Fernández et al., 2018b; Kovács, 2019). In this approach, new instances are generated on lines connecting existing minority samples. Particularly, given a minority point named as *seed*, a sample from its k-nearest minority samples is randomly selected. Then, a synthetic sample is generated on the line joining these two points. This process is repeated until the desired number of synthetic samples are obtained. Such data-level approaches are also combined with ensembling techniques to improve the generalization performances (Ghaderi Zefrehi & Altınçay, 2020; Galar et al., 2013; Haixiang et al., 2017; Díez-Pastor et al., 2015). As an alternative approach, algorithm-level modifications of existing classifiers is addressed. For instance, in cost-sensitive decision trees, the cost of misclassifying a minority class sample is set to be higher than a majority sample (Ling et al., 2006). The penalty of misclassifying minority class samples by a support vector machine classifier is set to be higher than that of the majority class (Veropoulos et al., 1999). Similarly, a weighted cross-entropy loss is employed for XGBoost to consider misclassified positive samples as larger loss compared to the misclassified negatives (Wang et al., 2020).

SMOTE has been criticized to have several shortcomings (Barua et al., 2014), which can be itemized as follows: (1) The samples in dense clusters will have the neighbors that are likely to be from the same clusters. In such cases, the synthetic samples will be near replicas of the seed sample lying in dense clusters. (2) The relative positions and distances of the minority samples are not considered in k-nearest-based neighbor selection. Because of this, the selected samples may be in incorrect regions. For instance, the seed sample may be noisy or an outlier and hence located in a region that is far away from the other minority samples. In such a case, the synthetic samples may also be away from the other minority examples. (3) The nearest neighbor of a seed may be in a different cluster. In this type of cases, the synthetic sample may be located in space of the majority class. (4) Due to the linear interpolation-based sample generation, true distribution of the minority samples will be distorted (Halimu & Kasem, 2021; Xie et al., 2022). If a classifier takes into account the variance of the minority samples, incorrect variance estimates will lead to additional challenges for the classifiers during testing (Blagus & Lusa, 2013). Moreover, the training samples will not be independent after balancing, violating independence assumption, if employed by the classifier (Blagus & Lusa, 2013). (5) Using an a priori fixed k value is another weakness since variations of the density of samples in different regions is ignored. (6) All samples are not equally important from oversampling point of view. For instance, some samples such as those located close to the decision boundary are harder-to-classify but SMOTE does not differentiate between the samples (Barua et al., 2014; Napierala & Stefanowski, 2016).

The solutions for binary imbalanced classification tasks are generally easier to formulate when compared to multi-class setting. For instance, samples can be categorized by taking into account their positions with respect to the decision boundary, such as boundary sample, safe or outlier (Napierala & Stefanowski, 2016). Since there are two classes, each class is either minority or majority. However, multi-class problems are more challenging, mainly due to having more complex relations between the classes than the binary case (Krawczyk, 2016; Lango & Stefanowski, 2022). Moreover, learning the decision boundaries is expected to be more difficult in the multi-class case. Binary imbalanced classification has attracted more interest and many variants of SMOTE have been proposed for this purpose, each addressing some of its shortcomings (Fernández et al., 2018b; Kovács, 2019). Most of these techniques employ alternative strategies for selecting and/or weighting the minority samples during resampling. Two well-known and frequently-mentioned methods are Borderline-SMOTE and ADASYN (Han et al., 2005; He et al., 2008). In Borderline-SMOTE, a subset of minority samples is considered as seed samples. For instance, if all k-nearest neighbors of a minority sample are from the majority class, it is assumed to be noise and it is not considered during oversampling. A sample in the remaining set is considered as a seed only if most of its nearest samples are from the majority class. This criterion is expected to be mainly satisfied by the borderline samples and hence the learner will more accurately learn the decision boundary. A variant of Borderline-SMOTE which takes into account the neighboring majority samples as well in generating synthetic samples is also proposed (Han et al., 2005). On the other hand, ADASYN assigns different weights to the minority samples (He et al., 2008). The likelihood that a sample will be selected as a seed is proportional with its weight. The weight of a sample is computed by taking into account the number of majority samples in its nearest sample set. Using weights, it is aimed at focusing on the samples that are harder to classify. However, noisy samples that are located in majority class region may also get large weights, leading to introducing additional noisy samples (Barua et al., 2014). SMOTE may generate synthetic samples that overlap with the majority class space, leading to expanded minority clusters. Data cleaning is an alternative technique used for post-processing the balanced datasets to correct the distortions of the class clusters. For instance, in SMOTE-ENN, any instance that is not correctly labeled by employing three nearest neighbors is deleted (Batista et al., 2004). Similarly, SMOTE-TomekLinks discards the instances that form Tomek links (Batista et al., 2004). CCR is another data-cleaning based approach where, before oversampling, the neighborhood of each minority sample is cleaned by removing majority class instances (Koziarski & Wożniak, 2017).

The ultimate goal of oversampling is to enforce the learner to label the feature space including minority samples as the positive class, although some of these regions might be outnumbered by the negative samples. In fact, this can be achieved using an alternative approach. Rather than increasing the numbers of positive samples in the subspace where they appear, we propose to *reposition* the negative samples towards their centroid so as to reduce the number of majority samples in the subspace where the minority samples exist. Similarly, the positives are repositioned but for small steps to effectively deal with different types of minority samples such as borderline or outlying instances. This approach has several advantages compared to the oversampling-based approaches. For instance, small displacements will keep the positive borderline samples close to their original positions and hence do not distort the boundary. After repositioning, the outliers that are more likely to be in majority class space will not mislead the learner as their original forms since they will be closer to the other positives. Moreover, irrelevant positive regions are not generated since the minority instances are slightly modified. Consequently, the assumptions made by some classifiers about the original class distribution such as normality of feature values are not violated. The hard-to-classify minority samples problem is also addressed by reducing the number of negatives in the corresponding spaces.

The first step of the proposed MAjority and MInority rePOsitioning Technique (MaMiPot) is to train the learner on the given dataset to identify the misclassified samples, that is the set of false positives (sFP) and false negatives (sFN). The initial set of the seed samples is computed as the union of sFP and sFN. The algorithm repositions the seed samples in sFP using the centroid of the true negatives by linear interpolation. Similarly, the samples in sFN are repositioned by employing centroid of the correctly classified positive samples. The learner is re-trained by employing the updated training set and evaluated on the original training data to check for improvement in the performance metric. The procedure is repeated until further performance gain is not achieved.

The following section presents a brief review of the resampling-based techniques to alleviate the aforementioned shortcomings of SMOTE. The complete description of the proposed imbalance learning approach is presented in Section 3. The experimental work done on 52 datasets is presented in Section 4. The conclusions drawn are given in Section 5.

2 Literature review

More than hundred variants of SMOTE have been proposed after it was published in year 2003 (Fernández et al., 2018b; Kovács, 2019). These methods mainly differ in the approaches utilized in selecting and/or weighting the minority seed samples, the methods for generating new samples to replace linear interpolation and, elimination of existing noisy samples and incorrect synthetic samples that are located in the majority class region. In this brief review, some of these variants are presented to highlight the solutions they proposed for one or more shortcomings of SMOTE.

In SMOTEFUNA, the furthest sample from the seed is selected and the synthetic sample is generated within the cuboidal space, allowing to produce diverse synthetic instances (Tarawneh et al., 2020). This technique is argued to have several advantages. For instance, the synthetic samples are not expected to be close to their seeds in the cases when the seed is from a dense cluster. Hence, the shortcoming mentioned in item (1) is improved. Moreover, since only the furthest instance is considered, setting the value of tuning parameter k is not needed.

In order to avoid employing noisy instances as seed that is mentioned in item (2), DBSMOTE utilizes a clustering algorithm (Bunkhumpornpat et al., 2012). The samples that do not lie in a cluster are labeled as noisy instances and they are not utilized in oversampling. In MSMOTE, if all k neighbors of a minority sample belong to the majority class, it is labeled as a noisy instance and ignored (Liang et al., 2009). MWMOTE checks the neighbors of all minority samples (Barua et al., 2014). The samples which do not have any minority example in their neighbor set are discarded. Hence, such samples will not be considered as either seed or neighbor during linear interpolation. In Safe-Level SMOTE, the safety of the two samples selected for linear interpolation is computed by checking their neighbors (Bunkhumpornpat et al., 2009). The synthetic sample is generated closer to the instance that is more safe.

As a solution to the weakness mentioned in item (3), (Hu et al., 2014) proposed to use a classifier to evaluate the validity of each synthetic sample. A classifier is initially trained using the training samples. Two randomly selected minority samples are employed to generate a new synthetic sample by linear interpolation. This sample is accepted if the confidence score of the classifier is within a predetermined confidence interval. In GSMOTE, a safe area is defined for each minority sample (Douzas & Bação, 2019). Safe area represents the feature space where the generated samples are not noisy. In SMOTEFUNA algorithm, the distances of each synthetic sample from the nearest minority and nearest majority sample are utilized to evaluate its validity (Tarawneh et al., 2020). In particular, if a synthetic sample is closer to the negative nearest neighbor, it is discarded. In RBO, the synthetic samples are generated by computing Gaussian radial basis function-based distributions of samples in both positive and negative classes (Koziarski et al., 2019). The difficult regions are identified by using these potential surfaces and oversampled. ADPCHFO proposed by Tao *et al.* applies oversampling by employing samples within the same clusters (Tao et al., 2020). Density peak clustering is applied to cluster the minority samples. Using clusters allows generation of samples within valid regions. In order to avoid overfitting due to duplicate samples, a heuristic filter is applied to eliminate overlapping instances.

GSMOTE addresses the shortcoming mentioned in item (4) by using geometric regions rather than linear interpolation for generating synthetic instances (Douzas & Bação, 2019). For each minority sample, a safe radius is computed. Based on this value, sample generation takes place in a truncated hyper-spheroid. Xie *et al.* proposed the use of Gaussian distribution around the seed samples for generating synthetic instances (Xie et al., 2022). For a given seed, a random direction that originates from the seed is initially selected. The new sample is generated in that direction where the distance is computed using a Gaussian distribution whose parameters are computed from the training data. In ROSE, synthetic instances are sampled from a kernel density estimate that is centered at the selected samples (Menardi & Torelli, 2014). For both GSMOTE and ROSE, setting the value of tuning parameter k is not needed. Hence, the shortcoming mentioned in item (5) is avoided. SWIM employs the density of the well-represented majority class for generating new samples (Bellinger et al., 2020). In particular, the synthetic samples have approximately the same density as the seed.

In order to address the shortcoming mentioned in (6), ADASYN selects the seeds by considering the weights computed for each minority samples. The weight is proportional with the number of neighbors from the majority class. Hence, the learner is expected to focus on the hard-to-classify samples. In order to put more emphasis on hard-to-classify samples, SDSMOTE identifies the borderline samples by defining the degree of support on the minority samples (Li et al., 2014). In MWMOTE, the hard-to-learn samples are selected by considering the distance of each minority sample to its closest majority instance (Barua et al., 2014).

Many other variants of SMOTE have also been proposed. For instance, SMOTE-IPF addresses the shortcoming in item (2) by removing both the noisy samples in the original data and noisy synthetic samples (Sáez et al., 2015). TRIM-SMOTE proposes preprocessing the minority instances to generate multiple seed sets to be utilized for generating synthetic instances (Puntumapon & Waiyamai, 2012). In Polynom-fit-SMOTE method, synthetic instances are generated using polynomial fitting (Gazzah et al., 2008). MCT generates synthetic samples using cloning where the instances on the borderline are cloned less than those that are internal to the minority class region (Jiang et al., 2015). SMOTE-D considers the distances between each seed and its k-nearest neighbors to calculate the number synthetic instances between each sample pair (Torres et al., 2016). Cluster-SMOTE aims at generating artificial samples withing the major clusters of the minority class (Cieslak et al., 2006). CURE-SMOTE is another clustering-based approach (Li & Fan, 2017). The minority samples are clustered to identify small clusters. These are assumed to be noisy and removed. MDO generates synthetic samples in dense regions of the minority class and hence reduces the risk generating instances in majority class space (Abdi & Hashemi, 2016). In ANS, the number of neighbors considered, i.e. k is not fixed but dynamically calculated for each minority sample (Siriseriwan & Sinapiromsaran, 2017). In kmeans-SMOTE, the training data is clustered in an unsupervised way (Douzas et al., 2018). The clusters including small proportion of minority samples are not selected for oversampling. Moreover, dense minority clusters are oversampled more than sparse clusters.

Balancing is also addressed by analyzing the types of samples in the minority class. For instance, the minority samples are categorized as safe, rare, borderline and outlier and, the weights of a one-class classifier are adjusted based on the types of samples (Krawczyk et al., 2014). Sampling by analyzing the neighborhood of the positive samples is also proposed (Błaszczyński & Stefanowski, 2015). A bagging-based ensemble of classifiers is constructed where each boostrap is formed by mainly including hard-to-learn samples. In particular, the probability of drawing an instance depends on its difficulty of classification that is quantified using the number of neighbors from the majority class.

3 The proposed approach: MaMiPot

The proposed approach is based on repositioning the training samples. In order to understand the main idea behind this approach, consider the toy example given in Fig. 1. Due to class imbalance, most of the positives (represented by red diamonds) are expected to be misclassified, as shown on the left part of the figure. In other words, the decision region for the minority class covers much smaller space than the actual space where the positive samples appear. Such solutions generally lead to low classification accuracy on the positive samples (i.e. sensitivity) but high accuracy on the negatives (i.e. specificity). As shown in the upper right part of the figure, the solution by SMOTE is to generate synthetic minority samples to enlarge the positive decision region, leading to a more acceptable solution. As an alternative approach, the idea behind MaMiPot is presented in the lower right part of the figure. The positive and negative samples are repositioned for the same purpose of enlarging the minority decision region. The repositioned instances are shown using the same symbol of the class but not filled with the corresponding color. Since the majority samples overlapping with the minority are moved away from the minority region, minimizing the classification accuracy will correspond to labeling a wider space as positive. Moreover, since the distribution of minority samples is not distorted, the borderline is computed more accurately. Although the motivation behind moving the majority samples is intuitively reasonable, repositioning of the minority instances needs further clarification. As mentioned above, one of the main challenges in SMOTE is outliers. These samples may lead to generating synthetic instances in the majority class region and hence severely mislead the learner. To alleviate this

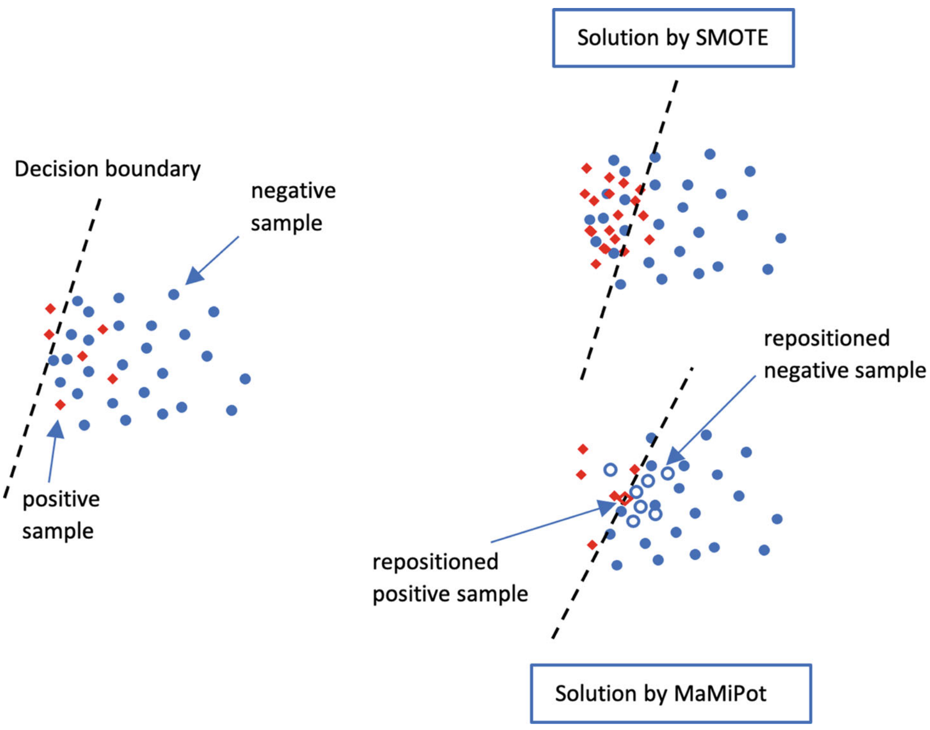


Fig. 1 Comparison of SMOTE and MaMiPot on a toy example

problem, the proposed approach repositions the minority samples as well. However, in order not to distort the distribution of the minority samples, the amount of repositioning is set to be in smaller steps for the minority class.

There are two main issues that need to be addressed for the implementation of the idea. These are selection of the samples and the directions to which they will be moved. In MaMiPot, the decision boundary computed on the original training data is employed for guiding the selection and repositioning tasks. The details of the proposed method is presented in Algorithm 1. The first step of the algorithm is to train a classifier. In Step 2, it the classifier is tested using the training data. The next step, Step 3 is to obtain the decision boundary by computing the best-fitting decision threshold, τ_{opt}. The performance metric is a design parameter of the algorithm. Hence, τ_{opt} should be defined by taking into account the metric, M. Using the selected threshold, the value of the metric, M_{opt} is recorded. Selection of τ_{opt} is explained in detail in the following context.

Using the decision boundary obtained, the training samples are partitioned into four sets. sTP denotes the set of correctly classified minority samples (i.e. true positives). sFN represents the set of misclassified minority samples (i.e. false negatives). The set sTN includes correctly classified majority samples (i.e. true negatives). sFP denotes the set of misclassified majority samples (i.e. false positives). The next step is to compute the centroids of the correctly classified minority samples denoted by μ_P and the correctly classified majority samples denoted by μ_N.

```
Input: $T$: Training data ($P$ minority and $N$ majority samples), $C$: Classifier, $M$:
       Performance metric, $\alpha_P$: Positive (minority) repositioning factor, $\alpha_N$: Negative
       (majority) repositioning factor, $\gamma$: cluster size threshold
1  Train the classifier $C$ using the training data $T$
2  Test $C$ using $T$
3  Compute the threshold, $\tau_{opt} = \frac{1}{2}\left(\frac{P}{P+N} + 0.5\right)$ and record the value of $M$ computed
   using $\tau_{opt}$, denoted by $M_{opt}$
4  Compute the sets $sTP$, $sFP$, $sTN$ and $sFN$ using $\tau_{opt}$
5  Calculate the centroid of samples in $sTP$, denoted by $\mu_P$
6  Calculate the centroid of samples in $sTN$, denoted by $\mu_N$
7  $iter \leftarrow 0$, $T_{opt} \leftarrow T$
8  while $iter < 3$ do
9      $iter \leftarrow iter + 1$
10     //Repositioning minority samples in $sFN$:
11     if $|sTP| > \gamma$ & $|sFN| > 0$ then
12         for each $x_i$ in $sFN$ do
13             Move $x_i$ towards $\mu_P$ using $\alpha_P$:
14                 $\hat{x}_i = \alpha_P * \mu_P + (1 - \alpha_P) * x_i$
15     //Repositioning majority samples in $sFP$:
16     if $|sTN| > \gamma$ & $|sFP| > 0$ then
17         for each $y_i$ in $sFP$ do
18             Move $y_i$ towards $\mu_N$ using $\alpha_N$:
19                 $\hat{y}_i = \alpha_N * \mu_N + (1 - \alpha_N) * y_i$
20     Define $T_{new}$ as the union of $sTP$, $sTN$ and the repositioned sets $\widehat{sFN}$ and $\widehat{sFP}$
21     Train the classifier $C_{new}$ using $T_{new}$
22     Test $C_{new}$ using the original training data $T$ and, using $\tau_{opt}$ record $M_{new}$
23     if $M_{new} > M_{opt}$ then
24         $M_{opt} = M_{new}$
25         $T_{opt} = T_{new}$
26         $iter \leftarrow 0$
Output: The repositioned training set, $T_{opt}$
```

Algorithm 1 MaMiPot.

The samples in sFN and sFP are the candidates to be repositioned. In the first part of the iterations starting on line 11, the misclassified minority samples in sFN are repositioned towards the centroid of the correctly classified minority samples using the positive repositioning factor, α_P. $\widehat{sFN}$ is computed as the updated set after repositioning. In this study, we assumed that the correctly classified samples form a single cluster.

It should be noted that, at this step only the locations in the set sFN changed. As a condition for making the repositioning, we check whether the number of correctly classified samples is above a threshold denoted by γ. The motivation for this condition can be explained as follows. The centroid is computed using only the correctly classified minority samples. In MaMiPot algorithm, the centroids are used as the representatives for the potential regions to be enlarged. By moving samples towards these points, it is aimed to enforce the learner to focus on that part of the feature space. Because of this, for a centroid to convey a potential for the learner to discover additional minority regions, a predefined number of samples denoted by γ are required to be already correctly classified. For instance, if all minority samples are misclassified, none of them is repositioned.

In the second part of the iterations starting on line 16, the misclassified majority samples are repositioned towards the centroid of correctly classified majority samples using the negative repositioning factor, α_N. The new positions are recorded, obtaining the updated set, $\widehat{sFP}$. As in the case of minority samples, we check whether the number of correctly classified majority samples is above the threshold, γ. The new training set, T_{new} containing the correctly classified samples (i.e. sTP and sTN) and, repositioned samples (i.e. $\widehat{sFN}$ and $\widehat{sFP}$), is used to construct a new classifier, C_{new} as presented on line 21. This classifier is tested on the original training set, T to evaluate whether the performance is improved. If the performance is improved, T_{new} is recorded as T_{opt} and the iterations are repeated.

During the iterations, the centroids are not updated. The main reason can be explained as follows: The ultimate goal of the algorithm is to reposition the samples but not the class. Allowing changes in the direction of repositioning may lead to translation of the classes in the feature space which is not desired. Actually, such operation will not be rewarded since the repositioning done in each iteration is evaluated using the original training set on Line 22 of the algorithm.

The repositioning factors α_P and α_N represent the percentage of repositioning on the lines connecting the minority and majority instances, respectively. In order not to distort the distribution of minority samples, α_P should be kept small. In other words, the minority samples must be repositioned in small steps. On the other hand, larger displacements should be allowed by employing large α_N values. In fact, α_P is mainly used to deal with the outliers. During the iterations, they are moved towards the centroid computed for the minority class. Figure 2 presents the execution of MaMiPot, where $\alpha_N = 0.9$ and $\alpha_P = 0.1$. The figure on the left shows the dataset. The minority samples are presented using the symbol '$\diamond$' filled by red color. The repositioned instances are plotted using non-filled symbols. As it can be seen, the positive and negative samples in the overlapping regions are moved towards the

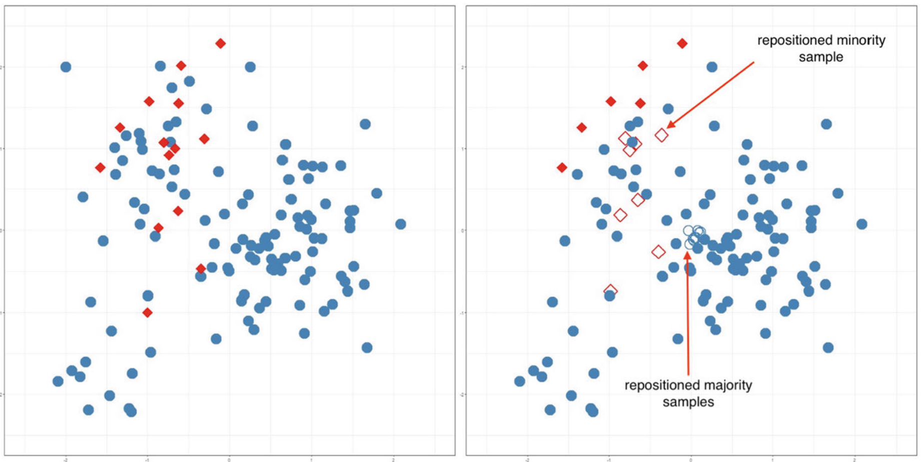


Fig. 2 Repositioning of positive and negative samples using MaMiPot. The design parameters were set as $\alpha_N = 0.9$ and $\alpha_P = 0.1$

corresponding centroids. Consequently, the classifier is expected to increase the number of true positives. It can be also be seen that the minority samples are moved slightly.

If three consecutive updates do not lead to further performance improvement, the iterations are terminated and T_{opt} is returned as the best-fitting training set. Remember that the centroids μ_P and μ_N are kept fixed during the iterations. Since α_N is selected as a large value, after the first iteration, the repositioned negatives will be close to μ_P. As a matter of fact, the remaining room for repositioning is small for the majority samples after the first iteration. On the other hand, since the minority samples are repositioned in small steps, they can continue to reposition as the number of iterations increase. Consequently, the first priority of the algorithm is to solve the problems due to imbalance by repositioning the negatives. If a performance gain is not achieved any more, by running for two more iterations, MaMiPot tries the lower priority option and targets at achieving further improvements by repositioning the minority samples. When $\alpha_P = 0.1$, the distance between a minority sample and its centroid will be reduced by $1 - (0.9)^3 = 0.27$ (i.e. 27%) after three iterations which is much less than the repositioning of majority samples in the first iteration (i.e. 0.90). This is consistent with our main target of avoiding altering the distribution of positive samples. Alternative values for the number of iterations can be empirically evaluated for the task under concern.

The performance metric is another design parameter of the algorithm. Accuracy is not an appropriate measure in imbalance learning. Since the number of minority samples is only a few in general, very high accuracy values can be obtained even by classifying all samples as negative. Alternatively, metrics that measure the performance separately for different classes are employed for this purpose. However, since the performances on the positive and negative classes are competing, trade-off forms of these metrics are generally utilized. These metrics are F-score, G-mean and AUC. F-score is defined as the harmonic mean of precision and recall where precision is the ratio of correctly classified positive instances and the number of all samples that are classified as positive (i.e. TP / (TP+FP)) and recall is the percentage of correctly classified positives (i.e. sensitivity, TP/ (TP+FN)) as follows:

$$F\text{-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{1}$$

G-mean, which is defined as the geometric mean of sensitivity (i.e. recall) and specificity (i.e the classification accuracy of the negative class) is calculated as

$$G\text{-mean} = \sqrt{Sensitivity \times Specificity}. \tag{2}$$

AUC is the third metric used that is computed as the area under the receiver operating characteristic curve. As mentioned above, the main idea of MaMiPot is to reposition the misclassified minority and majority samples. As a matter of fact, AUC can be used as a design parameter only if the samples to be repositioned are determined using a heuristic approach. For instance, majority samples that appear in highest P ranks and minority samples appearing in lowest N ranks can be selected. Alternatively, as done in several variants of SMOTE, the neighborhoods of the minority samples or density of the minority samples in different clusters can be considered. On the other hand, F-score or G-mean can be employed directly since they are threshold-dependent. In this study, in order not to bias the results by re-running the algorithm for all metrics, the algorithm is run only for F-score and the performance scores of all three metrics are reported. In a recent study, the following threshold is proposed for F-score (Collell et al., 2018):

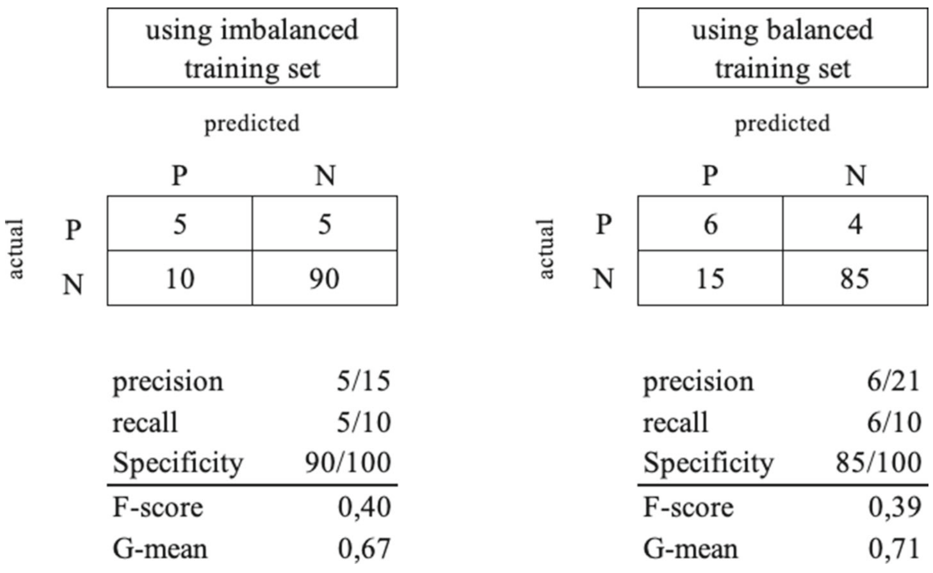
$$\tau_{opt} = \frac{1}{2}\left(\frac{P}{P+N} + 0.5\right) \tag{3}$$

where 0.5 is the maximum possible threshold value for maximizing F-score and $P/(P+N)$ is the a-priori probability of the minority class (Lipton et al., 2014). The midway between these two term is proposed as the threshold for F-score. In our simulations, we used this threshold value.

In general, since the minority samples are outnumbered by the majority, we expect low recall values (Erenel & Altınçay, 2013). In other words, the classification performance on the positive samples is generally low. The main reason for this is the bias in favor of the majority class samples. On the other hand, precision will be higher when compared to employing a balanced dataset. This can be explained as follows: After balancing, the decision region for the minority enlarges, including more correctly classified minority samples and hence higher recall. However, the number of false positives will also increase. Due to the imbalance, we expect a larger increase in FP than TP, leading to a smaller precision score. Let the *balancing factor,* β be defined as the proportion of the synthetic samples to the difference in the class sizes, $\beta = N_{syn}/(N_{maj} - N_{min})$ where N_{syn} denotes the number of synthetic samples that are generated in the case of N_{maj} majority and N_{min} minority instances. In order to tackle the low recall problem, SMOTE oversamples with balancing factor equal to one whereas MaMiPot aims to alleviate the imbalance problem by moving the negatives away from the positives. When the minority class is small, if the performance metric is selected as recall, a large degradation in precision may occur. In order to avoid this, the algorithm should be run using a trade-off metric such as F_1 and G-mean.

For a better understanding of the way MaMiPot will contribute the performance scores for different metrics, both class-imbalance and characteristics of the metrics should be taken into consideration. As it was emphasized in the recent study by (Soleymani et al., 2020), both AUC and G-mean favor increasing the number of true positives, even for much higher number of misclassified instances from the majority class. This can be understood by considering the toy example presented in Fig. 3. Assume that there are 10 minority and 100 test samples. Using a learner that is trained on the training set which is not balanced, let the number of true positives and true negatives be 5 and 90, respectively. After oversampling the minority class, we expect a higher true positive value. However, this can be achieved at the expense of true negatives. Let the number of true positives be increased by only one whereas the number of true negatives is decreased by 5. This corresponds an increase in sensitivity from $5/10$ to $6/10$ and a decreased specificity from $90/100$ to $85/100$. When compared to employing the original training data, although the total number of misclassified samples increased by $5 - 1 = 4$, G-mean increases from 0.67 to 0.71. Hence, G-mean favors increasing the number of true positives.

MaMiPot improves the performance on the minority class mainly by repositioning the majority samples. However, this is done in a global way. In particular, the algorithm does not focus on any specific region by evaluating the clustering of samples. Similarly, in SMOTE, the whole feature space of the minority class benefits from oversampling. On the other hand, in some variants of SMOTE, oversampling is applied in a selected subspace. For instance, kmeans-SMOTE considers the density of minority samples to select clusters to be oversampled. Since MaMiPot is not cluster or region-based, some minority samples may not benefit from repositioning. Oversampling the minority class has the potential to be a



using imbalanced training set

actual \ predicted	P	N
P	5	5
N	10	90

precision	5/15
recall	5/10
Specificity	90/100
F-score	0,40
G-mean	0,67

using balanced training set

actual \ predicted	P	N
P	6	4
N	15	85

precision	6/21
recall	6/10
Specificity	85/100
F-score	0,39
G-mean	0,71

Fig. 3 An example to illustrate the effect of balancing in terms of the metric values

complementary tool for such samples. In this setting, MaMiPot will be utilized as a pre-processing step before oversampling, providing several advantages for the oversampler. For instance, due to repositioning of the majority samples by MaMiPot, a smaller balancing factor than one is expected to be better-fitting, leading to less distortion on the minority distribution when compared to SMOTE and most of its variants. Moreover, utilizing a small balancing factor will result in a similar imbalance for both training and test data, which is another important concern in imbalance learning (Pozzolo et al., 2015). By repositioning outlying minority samples towards the centroid of the correctly classified instances, the risk of generating synthetic samples in invalid regions is reduced.

Precision replaces the specificity in the case of F-score (Soleymani et al., 2020). Since the minority class includes small number of samples in general, if the increase in true positive rate due oversampling is achieved at the expense of precision due to increased number of false positives, F-score value will degrade. As shown in Fig. 3, F-score decreases from 0.40 to 0.39. Consequently, when F-score is the performance metric, SMOTE is expected to contribute MaMiPot only if it is run using a small balancing factor. The left part of Fig. 4 presents the execution of SMOTE on the data given in Fig. 2. The dataset obtained after running MaMiPot using $\alpha_N = 0.9$ and $\alpha_P = 0.1$, followed by oversampling using SMOTE for $\beta = 0.5$ is illustrated on the right. It can be seen in the figure that, since many majority samples are repositioned, a smaller number of synthetic samples is expected to be sufficient for achieving a reasonable recall value.

4 Experimental work

In order to evaluate the proposed approach, the experiments are conducted using three classifiers, namely support vector machine (SVM) with radial basis function kernel, naive Bayes (NB) and decision tree (J48) which are widely employed for imbalanced datasets (Haixiang

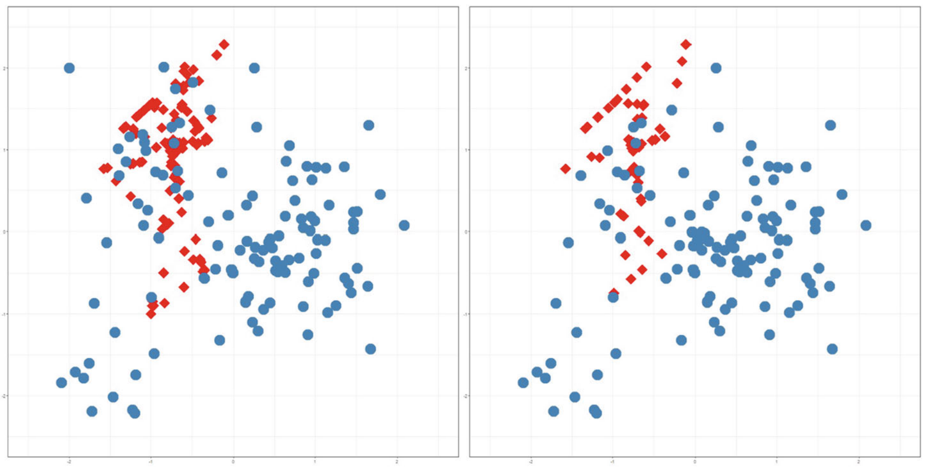


Fig. 4 Balancing the data using SMOTE (left figure) and balancing positive samples after repositioning by MaMiPot and using $\beta = 0.5$ (right figure). The design parameters of MaMiPot were set as $\alpha_N = 0.9$ and $\alpha_P = 0.1$

et al., 2017). The computational cost of SVM is high. Based on the empirical evidence from preliminary experiments, we fixed its parameters as $\sigma = 0.25$ and $C = 0.25$ in all simulations. The default settings are utilized for J48, i.e. confidence factor = 0.25 and minimum instances per leaf is 2. The experiments are conducted on KEEL collection (Alcalá-Fdez et al., 2011). Fourteen datasets having imbalance ratios less than 5 are discarded. All of the remaining 52 datasets having the imbalance ratios between 5.14 and 129.44 are considered. The datasets are presented in Table 1.

For each dataset, the experiments are repeated 20 times and the average scores are reported. In each experiment, 5×2-fold cross validation is applied. In particular, the data is randomly split into two equal-sized parts five times. Both parts are then utilized as either training or test data. The performance of the proposed method is compared with SMOTE and nineteen extensions namely, ADASYN (He et al., 2008), Borderline-SMOTE1 (Han et al., 2005), DBSMOTE (Bunkhumpornpat et al., 2012), GSMOTE (Douzas & Bação, 2019), SMOTEFUNA (Tarawneh et al., 2020), MWMOTE (Barua et al., 2014), ANS (Siriseriwan & Sinapiromsaran, 2017), CCR (Koziarski & Wożniak, 2017), CURE-SMOTE (Li & Fan, 2017), kmeans-SMOTE (Douzas et al., 2018), SMOTE-D (Torres et al., 2016), MDO (Abdi & Hashemi, 2016), MCT (Jiang et al., 2015), SMOTE-ENN (Batista et al., 2004), SMOTE-TomekLinks (Batista et al., 2004), Polynom-fit-SMOTE (Gazzah et al., 2008), SMOTE-IPF (Sáez et al., 2015), TRIM-SMOTE (Puntumapon & Waiyamai, 2012) and Cluster-SMOTE (Cieslak et al., 2006).

In the first set of experiments, MaMiPot is run using five balancing factors, $\beta \in \{0, 0.25, 0.50, 0.75, 1\}$. As mentioned above, α_P and α_N which are in the interval (0, 1) represent the percentage of repositioning on the lines connecting the minority and majority instances to the centroids of the corresponding classes. To reposition the minority instances in small steps, we set $\alpha_P = 0.1$. On the other hand, we set $\alpha_N = 0.9$ to allow larger displacements for the majority class. A large value of α_N is expected to allow MaMiPot terminate in small number of iterations. Since the minority class is rare in many datasets, we set $\gamma = 1$. When $\beta = 0$, only MaMiPot is applied. For $\beta > 0$, SMOTE is applied after MaMiPot where the number of synthetic samples is determined by β. The results obtained

Table 1 The datasets selected from KEEL repository

Rank	Data set	#Positives	IR	Rank	Data set	#Positives	IR
1	abalone19	32	129.44	27	glass-0-1-6_vs_2	17	10.29
2	yeast6	35	41.40	28	ecoli-0-6-7_vs_5	20	10.00
3	ecoli-0-1-3-7_vs_2-6	7	39.14	29	vowel0	90	9.98
4	yeast5	44	32.73	30	yeast-0-5-6-7-9_vs_4	51	9.35
5	yeast-1-2-8-9_vs_7	30	30.57	31	ecoli-0-3-4-7_vs_5-6	25	9.28
6	yeast4	51	28.10	32	ecoli-0-3-4-6_vs_5	20	9.25
7	yeast-2_vs_8	20	23.10	33	glass-0-4_vs_5	9	9.22
8	glass5	9	22.78	34	ecoli-0-2-6-7_vs_3-5	22	9.18
9	yeast-1-4-5-8_vs_7	30	22.10	35	ecoli-0-1_vs_2-3-5	24	9.17
10	shuttle-c2-vs-c4	6	20.50	36	ecoli-0-4-6_vs_5	20	9.15
11	glass-0-1-6_vs_5	9	19.44	37	yeast-0-2-5-6_vs_3-7-8-9	99	9.14
12	abalone9-18	42	16.40	38	yeast-0-2-5-7-9_vs_3-6-8	99	9.14
13	page-blocks-1-3_vs_4	28	15.86	39	yeast-0-3-5-9_vs_7-8	50	9.12
14	ecoli4	20	15.80	40	glass-0-1-5_vs_2	17	9.12
15	glass4	13	15.46	41	ecoli-0-2-3-4_vs_5	20	9.10
16	yeast-1_vs_7	30	14.30	42	ecoli-0-6-7_vs_3-5	22	9.09
17	shuttle-c0-vs-c4	123	13.87	43	yeast-2_vs_4	51	9.08
18	ecoli-0-1-4-6_vs_5	20	13.00	44	ecoli-0-3-4_vs_5	20	9.00
19	cleveland-0_vs_4	13	12.62	45	page-blocks0	559	8.79
20	ecoli-0-1-4-7_vs_5-6	25	12.28	46	ecoli3	35	8.60
21	glass2	17	11.59	47	yeast3	163	8.10
22	glass-0-1-4-6_vs_2	17	11.06	48	glass6	29	6.38
23	ecoli-0-1_vs_5	20	11.00	49	segment0	329	6.02
24	glass-0-6_vs_5	9	11.00	50	ecoli2	52	5.46
25	led7digit-0-2-4-5-6-7-8-9_vs_1	37	10.97	51	new-thyroid2	35	5.14
26	ecoli-0-1-4-7_vs_2-3-5-6	29	10.59	52	new-thyroid1	35	5.14

are presented in Fig. 5. The rightmost column presents the specificity, recall and precision scores. As expected, increasing the minority samples by applying SMOTE results in decreased precision but increased recall, when compared to using the original training set. Due to reducing the bias on the majority class using oversampling, specificity which denotes the performance on the majority samples decreases as β increases.

The leftmost column in Fig. 5 presents the performance scores obtained for three trade-off metrics and several balancing factors. It can be seen in the figure that, applying oversampling after MaMiPot degrades the average F-score for NB. On the other hand, G-mean improves for all three classifiers when $\beta = 0.25$. Similarly, a notable improvement is observed for AUC in the case of J48. For $\beta > 0.25$, there is not any notable improvement for SVM and NB. When all metrics are considered, it can be argued that $\beta = 0.25$ is a reasonable setting.

The average F-score, G-mean and AUC scores obtained using 52 KEEL datasets are presented in Table 2. The results obtained for both $\beta = 0$ and oversampling using $\beta = 0.25$ are reported. For each classifier and metric, the highest score is presented in boldface and the second highest score is underlined. The table shows that, using $\beta = 0.25$ our method

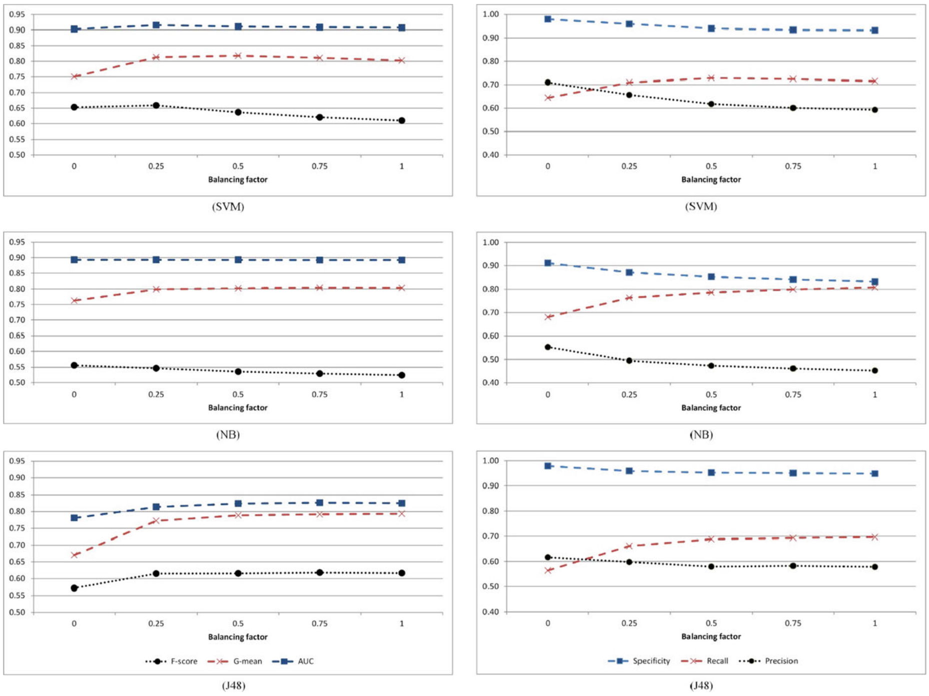


Fig. 5 Average F-score, G-mean and AUC values (in leftmost column) and, sensitivity (recall), specificity and precision scores (in rightmost column) obtained by MaMiPot ($\beta = 0$) and oversampling after MaMiPot using different balancing factor values, $\beta > 0$. Each row corresponds to a different classifier

achieves the highest average scores for all three metrics for SVM. Similarly, it provides better scores than all references for F-score and G-mean using NB. Another observation is that, for all three performance metrics, the scores obtained for SVM are superior to those obtained for both NB and J48. For G-mean, the importance of applying oversampling after repositioning by MaMiPot is evident from the scores obtained using all classifiers. On the other hand, when the average performance over all folds is considered, MaMiPot is not among the top-ranked systems for J48.

For a deeper evaluation of MaMiPot for different β, the performance scores are compared in terms of the average ranks over all folds for different values. In particular, a rank score is assigned for each β value and each fold. Then, the sums of these ranks over all folds are calculated for each β. After sorting in ascending order, the overall ranks are recorded. Table 3 presents the overall ranks corresponding to each classifier and performance metric pair. The last column presents the row-wise averages. $\beta = 0.25$ is found to be the top-ranked when the ranking performance is evaluated for all metrics and classifiers. It can be seen in the table that, when G-mean and AUC are considered, J48 benefits more from higher β values when compared to SVM and NB.

Taking into account the ranking scores presented in Table 3, we set $\beta = 0.25$ in the following context. Table 4 presents the overall rank of each classifier and performance metric pair. The last column presents the row-wise averages. It can be seen in Table 4 that, MaMiPot using $\beta = 0.25$ is the top-ranked technique when the ranking performance is

Table 2 The average performance scores obtained using SVM, NB and J48

	SVM			NB			J48		
	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC
SMOTE	0.6239	0.7753	0.9080	0.5309	0.7928	0.8946	0.6152	0.7939	0.8251
ADASYN	0.5965	0.7580	0.9038	0.5058	0.7839	0.8907	0.6067	0.7897	0.8211
Borderline-SMOTE1	0.6292	0.7598	0.9084	0.5374	0.7873	0.8919	**0.6228**	0.7816	0.8192
DBSMOTE	0.6234	0.7581	0.9071	0.5396	0.7764	0.8815	0.6165	0.7674	0.8246
GSMOTE	0.6288	0.8017	0.9127	0.5200	0.7891	**0.9021**	0.6108	0.8120	**0.8384**
SMOTEFUNA	0.6157	0.7402	0.9034	0.5188	0.7453	0.8901	0.6126	0.7618	0.8145
MWMOTE	0.6070	0.7587	0.9026	0.4987	0.7658	0.8689	0.5993	0.7665	0.8096
ANS	0.5948	0.6874	0.8958	0.5077	0.7017	0.8548	0.6021	0.7235	0.7971
CCR	0.5736	0.7638	0.8921	0.4281	0.5742	0.8721	0.5779	0.7019	0.7978
CURE-SMOTE	0.5925	0.7017	0.8958	0.4762	0.6899	0.8574	0.5865	0.7155	0.7957
kmeans-SMOTE	0.5666	0.6336	0.8952	0.4914	0.6790	0.8605	0.6001	0.7026	0.7933
SMOTE-D	0.6058	0.7472	0.9049	0.5078	0.7579	0.8866	0.6075	0.7760	0.8152
MDO	0.5920	0.7060	0.8957	0.4430	0.6521	0.8699	0.6072	0.7299	0.8148
MCT	0.6165	0.7691	0.9083	0.4997	0.7681	0.8710	0.6057	0.7588	0.8032
SMOTE-ENN	0.6067	0.7835	0.9061	0.5257	0.7855	0.8942	0.6059	**0.8132**	0.8326
SMOTE-TomekLinks	0.6228	0.7737	0.9078	0.5311	0.7924	0.8938	0.6146	0.7941	0.8244
Polynom-fit-SMOTE	0.6255	0.7534	0.9082	0.5118	0.7238	0.8794	0.6079	0.7261	0.8139
SMOTE-IPF	0.6225	0.7733	0.9080	0.5299	0.7921	0.8942	0.6157	0.7939	0.8252
TRIM-SMOTE	0.6237	0.7288	0.9022	0.5399	0.7585	0.8805	0.6124	0.7458	0.8042
Cluster-SMOTE	0.6046	0.7428	0.9038	0.4989	0.7392	0.8692	0.6011	0.7583	0.8049
MaMiPot ($\beta = 0$)	0.6528	0.7503	0.9030	**0.5558**	0.7626	0.8928	0.5720	0.6705	0.7809
MaMiPot ($\beta = 0.25$)	**0.6587**	**0.8129**	**0.9153**	0.5462	**0.7991**	0.8929	0.6151	0.7722	0.8138

evaluated for all metrics and classifiers. It should be noted that, when evaluated using F-score, MaMiPot is top-ranked for SVM and second ranked for both NB and J48. The ranks achieved using J48 are low for G-mean and AUC. In a recent study, the performances of

Table 3 The rankings of MaMiPot for different β values

	SVM			NB			J48			Avg.
	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC	Rank
MaMiPot ($\beta = 0.25$)	1	2	1	2	1	2	1	4	4	**2.00**
MaMiPot ($\beta = 0.50$)	3	1	3	3	2	3	3	3	3	2.67
MaMiPot ($\beta = 0.75$)	4	3	4	4	3	4	2	2	1	3.00
MaMiPot ($\beta = 0.00$)	2	5	2	1	5	1	5	5	5	3.44
MaMiPot ($\beta = 1.00$)	5	4	5	5	4	5	4	1	2	3.89

The averages of all nine ranks are presented in the last column

Table 4 The rankings of different schemes according to their fold-based performance scores for SVM, NB and J48

	SVM			NB			J48			Avg.
	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC	Rank
MaMiPot	1	1	2	2	1	7	2	10	13	**4.33**
SMOTE-IPF	6	6	7	7	3	5	4	3	4	5.00
Borderline-SMOTE1	4	10	3	4	6	4	1	7	7	5.11
SMOTE	9	7	4	8	4	2	6	5	3	5.33
SMOTE-TomekLinks	5	5	8	6	2	6	8	4	5	5.44
GSMOTE	8	2	1	14	7	1	16	2	1	5.78
SMOTE-ENN	16	3	11	9	5	3	20	1	2	7.78
TRIM-SMOTE	2	14	10	1	8	8	3	14	16	8.44
DBSMOTE	10	13	9	3	11	17	9	11	6	9.89
Polynom-fit-SMOTE	3	12	6	10	16	19	10	17	11	11.56
MCT	7	4	5	16	10	10	15	16	21	11.56
SMOTE-D	13	16	16	13	13	9	14	9	10	12.56
SMOTEFUNA	11	17	13	12	15	16	5	13	12	12.67
Cluster-SMOTE	12	15	15	15	14	12	13	12	15	13.67
MDO	17	11	12	21	18	14	7	15	9	13.78
MWMOTE	15	9	18	18	12	13	17	8	14	13.78
ADASYN	19	18	19	19	9	11	18	6	8	14.11
ANS	14	19	17	5	17	18	12	18	17	15.22
kmeans-SMOTE	20	21	14	11	19	15	11	20	20	16.78
CCR	21	8	20	20	21	20	21	21	19	19.00
CURE-SMOTE	18	20	21	17	20	21	19	19	18	19.22

The averages of all nine ranks are presented in the last column

seven classifiers are evaluated on datasets that are grouped according to their characteristics (Napierala & Stefanowski, 2016). Considering the set of 5 neighboring samples, rare and outlying instances are defined to have one or zero minority instances in their sets of 5 neighbors, respectively. A sample is labeled as safe if at most one of its neighbors is from the majority class. It is shown that SVM performs better than J48 on datasets which include safe and borderline instances (Napierala & Stefanowski, 2016). It can be argued that, by repositioning the majority samples away from the minority class, training sets are transformed into safer forms, leading to better performance for SVM, when compared to J48.

It is well known that decision trees are unstable classifiers whereas SVM and NB are stable (Ting et al., 2011). Training unstable classifiers like decision trees is challenging when the training set size is small because a small perturbation in the training set may result in a highly different model, leading to substantial differences in the performance scores obtained on unseen test sets (Skurichina & Duin, 2002; Dietterich, 2000). As a matter of fact, the standard deviations in performance scores obtained from different folds are expected to be larger for unstable classifiers. In fact, variations in the predicted labels due to small perturbations on the training sets is effectively used in constructing classifier ensembles (Breiman, 1996; Bauer & Kohavi, 1999). In this approach, the predictions of unstable

Table 5 The average standard deviations over all folds for MaMiPot

	F-score	G-mean	AUC
SVM	0.0326	0.0234	0.1005
NB	0.0372	0.0316	0.1128
J48	0.2560	0.1929	0.1404

learners such as decision trees are combined to obtain more accurate decisions (Skurichina & Duin, 2000). Table 5 presents the average standard deviations for 20 repetitions over all folds for MaMiPot. It can be seen that the standard deviations are much larger for J48. For a better evaluation of relative performances of the schemes considered, 5-fold cross-validation is also performed. In this approach, 80% of the data is employed during training instead of 50%. The average scores and the corresponding ranks are presented in Tables 6 and 7, respectively. It can be seen that, for J48, the average ranks of MaMiPot are improved from 10 to 3 for G-mean and 13 to 10 for AUC. For stable classifiers SVM and NB, only the rank for G-mean is increased from 1 to 3 in the case of NB. Taking into account the size of datasets employed and the observations listed above, we argue that the inferior rankings achieved using J48 can be attributed to its unstable behavior on small datasets.

Table 6 The average performance scores obtained using SVM, NB and J48 using 5-fold cross validation

	SVM			NB			J48		
	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC
SMOTE	0.6477	0.8049	0.9136	0.5390	**0.7994**	0.9044	0.6223	0.7939	0.8326
ADASYN	0.6150	0.7845	0.9088	0.5053	0.7836	0.9011	0.6099	0.7852	0.8276
Borderline_SMOTE1	0.6605	0.8005	0.9168	0.5391	0.7925	0.8991	0.6354	0.7876	0.8323
DBSMOTE	0.6550	0.7925	0.9144	0.5499	0.7793	0.8903	0.6327	0.7696	0.8378
GSMOTE	0.6538	0.8332	0.9199	0.5308	0.7972	**0.9138**	0.6306	0.8165	**0.8532**
SMOTEFUNA	0.6459	0.7616	0.9102	0.5305	0.7460	0.8984	0.6419	0.7744	0.8392
MWMOTE	0.6210	0.7806	0.9147	0.5265	0.7912	0.8934	0.6186	0.7803	0.8266
ANS	0.6492	0.7427	0.9119	0.5547	0.7735	0.8836	0.6358	0.7503	0.8117
CCR	0.5752	0.7473	0.8804	0.4394	0.5580	0.8785	0.6236	0.7358	0.8228
CURE_SMOTE	0.6330	0.7328	0.9031	0.4929	0.6979	0.8622	0.6207	0.7221	0.8115
kmeans_SMOTE	0.6309	0.6855	0.9103	0.5240	0.7098	0.8691	0.6460	0.7400	0.8220
SMOTE_D	0.6371	0.7892	0.9114	0.5144	0.7570	0.8996	0.6288	0.7873	0.8290
MDO	0.6113	0.7245	0.8972	0.4575	0.6636	0.8767	0.6269	0.7388	0.8350
MCT	0.6376	0.8003	0.9161	0.5237	0.7819	0.8904	0.6319	0.7711	0.8186
SMOTE_ENN	0.6300	0.8130	0.9141	0.5353	0.7864	0.9024	0.6278	**0.8234**	0.8518
SMOTE_TomekLinks	0.6476	0.8025	0.9134	0.5336	0.7942	0.9058	0.6349	0.8019	0.8367
polynom_fit_SMOTE	0.6578	0.7857	0.9142	0.5300	0.7421	0.8877	0.6213	0.7190	0.8254
SMOTE_IPF	0.6462	0.8017	0.9138	0.5381	0.7983	0.9076	0.6203	0.7910	0.8355
TRIM_SMOTE	0.6656	0.7652	0.9166	0.5449	0.7815	0.8952	0.6468	0.7707	0.8267
Cluster_SMOTE	0.6313	0.7752	0.9136	0.4899	0.7384	0.8835	0.6332	0.7782	0.8244
MaMiPot	**0.6815**	**0.8358**	**0.9236**	0.5616	0.7927	0.8992	**0.6529**	0.7948	0.8327

Table 7 The rankings of different schemes according to their fold-based performance scores for SVM, NB and J48 using 5-fold cross validation

	SVM			NB			J48			Avg. Rank
	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC	
MaMiPot	1	1	2	2	3	7	2	3	10	**3.44**
TRIM_SMOTE	2	8	3	3	1	6	1	4	5	3.67
ANS	4	15	11	1	2	10	4	6	11	7.11
GSMOTE	13	2	1	16	11	1	16	2	2	7.11
SMOTE_TomekLinks	8	3	9	10	7	4	14	5	7	7.44
Borderline_SMOTE1	6	7	5	11	9	8	6	11	9	8.00
DBSMOTE	5	11	10	4	5	18	9	9	3	8.22
SMOTE_IPF	11	6	7	8	4	2	17	8	14	8.56
SMOTEFUNA	7	16	14	5	8	15	5	7	4	9.00
SMOTE	9	5	8	9	6	3	20	14	12	9.56
SMOTE_ENN	19	10	13	13	10	5	19	1	1	10.11
polynom_fit_SMOTE	3	9	4	7	12	19	13	21	8	10.67
kmeans_SMOTE	10	21	12	6	15	13	3	17	13	12.22
MCT	14	4	6	14	14	12	11	16	21	12.44
SMOTE_D	15	13	18	15	16	11	12	13	15	14.22
MWMOTE	17	12	17	17	13	9	18	10	18	14.56
Cluster_SMOTE	16	14	15	19	17	17	10	12	17	15.22
MDO	18	17	16	18	20	16	7	19	6	15.22
CURE_SMOTE	12	19	19	12	19	21	8	18	20	16.44
ADASYN	20	18	20	20	18	14	21	15	16	18.00
CCR	21	20	21	21	21	20	15	20	19	19.78

The averages of all nine ranks are presented in the last column

The KEEL repository includes datasets having a wide range of imbalance ratios. Similarly, the numbers of positive samples and dimensionalities of the feature vectors are different. In order to investigate the relative performance of different techniques for datasets having different characteristics, the average ranks of different KEEL subsets are computed. The results are presented in Table 8. The scores reported for $IR \geq 10$ are obtained using the datasets for which the imbalance ratio is greater than or equal to ten. Similarly the column labeled as $P > 30$ is for the list of datasets where the number of positive samples is above 30. In last two columns, the datasets are partitioned according to the number of available features, d. MaMiPot is top-ranked for highly imbalanced datasets. Similarly, it is the best-performing scheme when the number of positive samples is small. This is inline with the main target that was set as alleviating the imbalance problem without altering the distribution of the minority class.

As another comparison of different approaches, Nemenyi test is performed as shown in Figs. 6, 7 and 8, respectively for SVM, NB and J48 (Demšar, 2006). The black lines connect the methods for which the difference of mean ranks is smaller than the critical difference (CD). The test is done in a pairwise manner, for three classifiers and three different performance metrics. In part (a), F-scores are compared. Parts (b) and (c) compare G-mean and AUC scores, respectively. It can be observed that significantly better scores than most of

Table 8 The ranks on different subsets of KEEL datasets

	$IR \geq 10$	$IR < 10$	$P > 30$	$P \leq 30$	$d \geq 9$	$d < 9$
MaMiPot	3.00$^{(1)}$	9.78	6.56$^{(3)}$	3.33$^{(1)}$	7.44	4.00$^{(1)}$
SMOTE-IPF	6.44$^{(5)}$	5.33$^{(1)}$	5.00$^{(1)}$	6.44	3.89$^{(2)}$	7.89$^{(5)}$
Borderline-SMOTE1	4.11$^{(4)}$	9.56	7.00$^{(5)}$	4.89$^{(2)}$	7.22	5.67$^{(3)}$
SMOTE	4.11$^{(3)}$	8.89$^{(5)}$	7.22	5.22$^{(4)}$	6.22$^{(5)}$	6.00$^{(4)}$
SMOTE-TomekLinks	7.22	5.44$^{(2)}$	5.33$^{(2)}$	6.67	3.67$^{(1)}$	8.33
GSMOTE	3.22$^{(2)}$	9.33	7.89	5.00$^{(3)}$	5.67$^{(4)}$	5.33$^{(2)}$
SMOTE-ENN	8.22	7.33$^{(4)}$	7.44	9.11	5.11$^{(3)}$	9.67
TRIM-SMOTE	11.00	6.78$^{(3)}$	6.89$^{(4)}$	8.33	9.33	9.33
DBSMOTE	6.67	13.33	12.67	6.33$^{(5)}$	10.22	10.11
Polynom-fit-SMOTE	12.89	10.22	12.44	10.44	10.67	12.33
MCT	14.22	9.22	10.78	12.56	10.56	12.67
SMOTE-D	12.33	12.33	11.11	13.78	8.33	15.22
SMOTEFUNA	9.33	15.78	15.22	10.33	14.33	12.67
Cluster-SMOTE	13.89	11.67	11.78	15.00	14.00	13.22
MWMOTE	15.11	10.67	12.00	15.44	12.22	13.44
MDO	16.56	11.11	12.00	14.78	20.00	8.00
ADASYN	8.11	16.11	14.33	13.22	10.67	14.56
ANS	18.33	9.89	11.78	16.44	18.89	11.22
kmeans-SMOTE	18.00	13.67	15.56	16.67	18.78	14.11
CCR	19.78	16.33	19.56	17.56	17.22	17.89
CURE-SMOTE	18.44	18.22	18.44	19.44	16.56	19.33

The numbers in parentheses indicate the order of top 5 schemes

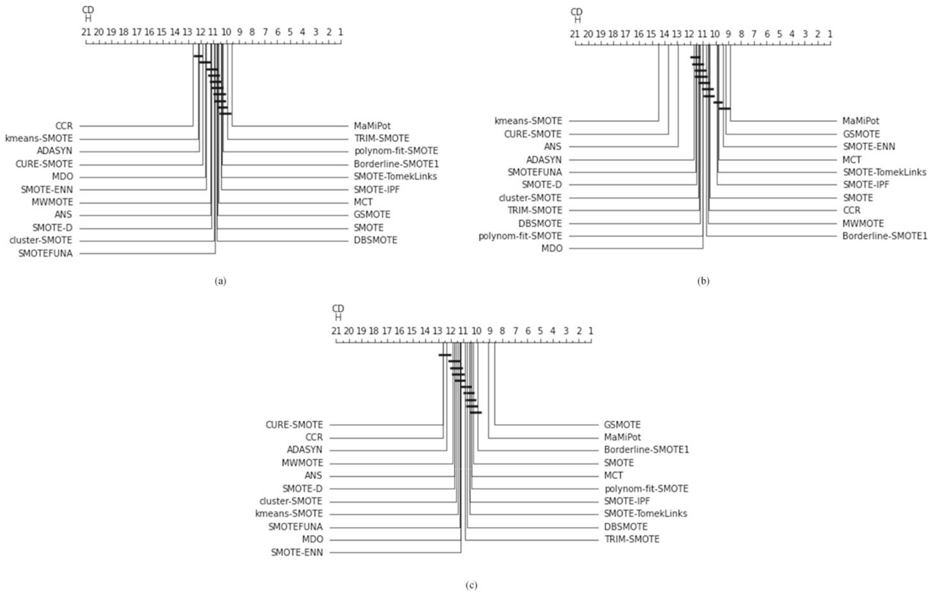


Fig. 6 Statistical test results for SVM: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores

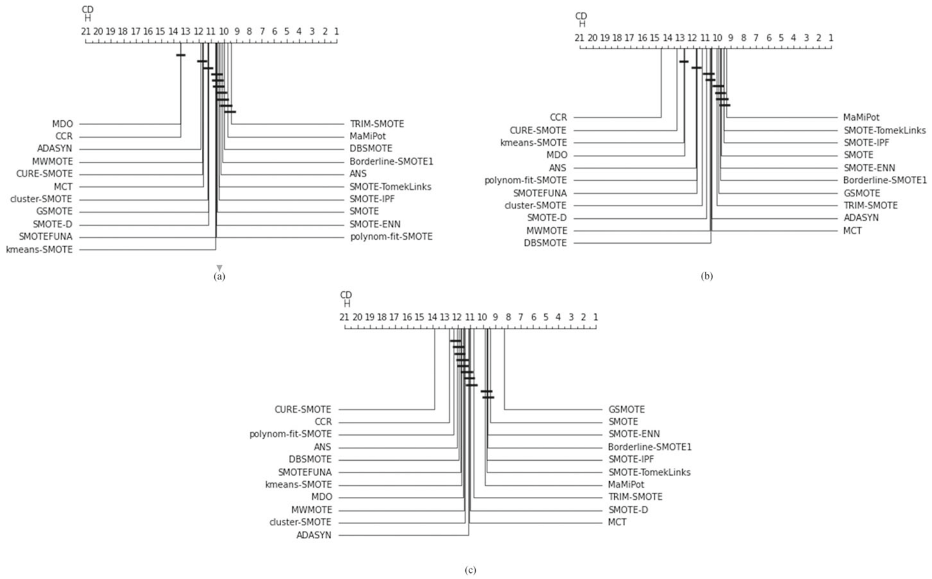


Fig. 7 Statistical test results for NB: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores

the reference techniques are obtained for majority of the classifier/metric pairs, especially for SVM and NB. Although MaMiPot is not among top-ranked schemes for G-mean and AUC when J48 is used, it can be seen in Fig. 8 that the only method that is consistently in top five for all three metrics is SMOTE-IPF. However, this technique is not ranked in top

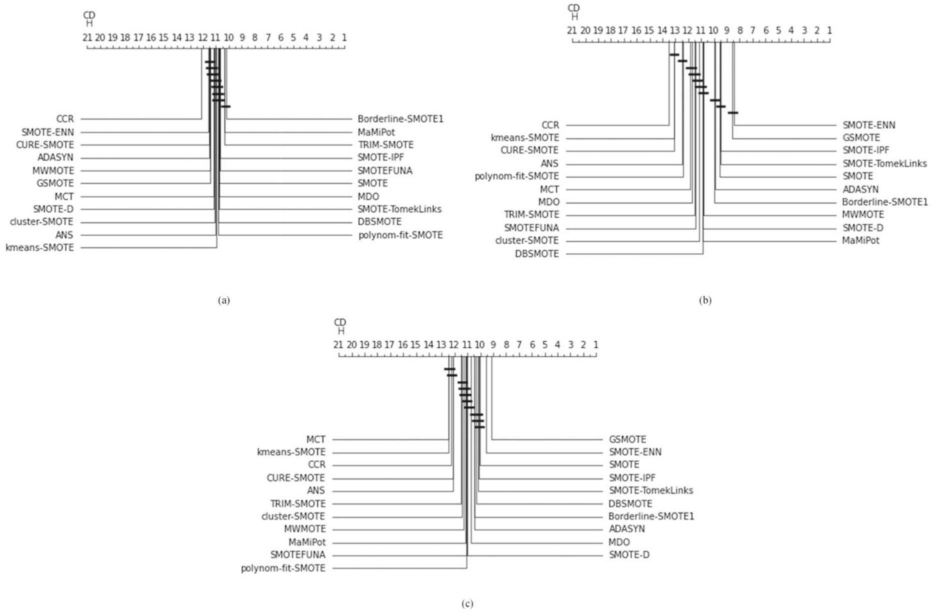


Fig. 8 Statistical test results for J48: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores

Table 9 The ties of the method providing the highest average rank over all folds for each classifier and performance metric pair

Classifier	Perf. Measure	Winner	Tying Methods and the p-values
SVM	F-score	MaMiPot	–
SVM	G-mean	MaMiPot	–
SVM	AUC	GSMOTE	–
NB	F-score	TRIM-SMOTE	MaMiPot (0.062)
NB	G-mean	MaMiPot	SMOTE-IPF (0.654), SMOTE-TomekLinks (0.717)
NB	AUC	GSMOTE	–
J48	F-score	Borderline-SMOTE1	MaMiPot (0.900), TRIM-SMOTE (0.900)
J48	G-mean	SMOTE-ENN	GSMOTE (0.900)
J48	AUC	GSMOTE	–

five for any metric in the case of SVM. Table 9 presents the ties of the method that is top ranked according to Table 2 for each classifier and performance metric. Pairwise comparisons between individual systems are performed using Nemenyi post-hoc test and the pairs having $p -$ value $\geq$ 0.05 are reported. The corresponding p-values are also provided. It can be seen that MaMiPot is tied with the winner, TRIM-SMOTE in the case of NB and F-score. Similarly, MaMiPot is tied with the winner, borderline-SMOTE1 in the case of J48 and F-score.

5 Conclusions and future work

In this study, repositioning of the instances is proposed as an alternative approach for imbalance learning. The main motivation behind the proposed approach was to effectively learn the minority decision regions without distorting the true distribution. The proposed approach is evaluated on 52 datasets, covering a wide range of imbalance ratios. The experiments conducted using three different classifiers have shown that MaMiPot achieves the best average ranking score, when evaluated in terms of three different performance metrics. The relative performance of MaMiPot and reference techniques were also compared for six groups of datasets which were formed by taking into account the imbalance ratios, numbers of positive samples and numbers of features. Among these groups, it is observed that MaMiPot is particularly effective on datasets that have higher imbalance ratio and on those having smaller number of positive samples.

In our experimental work, we considered MaMiPot as a preprocessing algorithm only for SMOTE. As a further study, the performance of MaMiPot should also be assessed for the variants of SMOTE. As it can be seen on lines 14 and 19 of the algorithm, the misclassified samples are moved towards the centroids denoted by μ_P and μ_N. Various alternatives can be considered for this purpose. For instance, the samples may be clustered as done in some variants of the SMOTE and the samples in sFP and sFN are moved towards the nearest cluster. Alternatively, given a majority sample in sFP, a randomly selected sample in sTN can be selected. In other words, misclassified majority samples may be moved towards a randomly selected true negative instance.

MaMiPot should be evaluated by using parameters other than the imbalance ratio. For instance, the datasets can be partitioned into subsets, each of which is dominated by either safe, borderline, outlier or rare samples. After categorizing the datasets, MaMiPot should be run on each group to further explore its strengths and weaknesses. Moreover, MaMiPot can be modified to reposition the majority samples in different ways by taking into account the types of neighboring minority samples.

Author Contributions Hakan Altınçay conceived the original idea. He planned the experiments, analyzed the results, and wrote the main manuscript text. Hossein Ghaderi Zefrehi developed the theory and carried out the experiments All authors discussed the results and contributed to the final manuscript.

Availability of supporting data The data is available at https://sci2s.ugr.es/keel/datasets.php

Declarations

Competing interests The authors declare that they have no conflict of interest.

References

Abdi, L., & Hashemi, S. (2016). To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE Transactions on Knowledge and Data Engineering*, *28*(1), 238–251. https://doi.org/10.1109/TKDE.2015.2458858.

Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL Data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*, 255–287.

Barua, S., Islam, M. M., Yao, X., & Murase, K. (2014). MWMOTE–Majority weighted minority over-sampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*(2), 405–425. https://doi.org/10.1109/TKDE.2012.232.

Błaszczyński, J., & Stefanowski, J. (2015). Neighbourhood sampling in bagging for imbalanced data. *Neurocomputing*, *150*, 529–542. https://doi.org/10.1016/j.neucom.2014.07.064.

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor Newsl*, *6*(1), 20–29. https://doi.org/10.1145/1007730.1007735.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, *36*, 105–139. https://doi.org/10.1023/A:1007515423169.

Bej, S., Davtyan, N., Wolfien, M., Nassar, M., & Wolkenhauer, O. (2021). LoRAS: An oversampling approach for imbalanced datasets. *Machine Learning*, *110*, 279–301. https://doi.org/10.1007/s10994-020-05913-4.

Bellinger, C., Sharma, S., Japkowicz, N., & Zaïane, O. R. (2020). Framework for extreme imbalance classification: SWIM—sampling with the majority class. *Knowledge and Information Systems*, *62*, 841–866. https://doi.org/10.1007/s10115-019-01380-z.

Blagus, R., & Lusa, L. (2013). SMOTE For high-dimensional class-imbalanced data. *BMC Bioinformatics*, *14*, 106. https://doi.org/10.1186/1471-2105-14-106.

Breiman, L. (1996). Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, Berkeley https://www.bibsonomy.org/bibtex/265f179a69a81cebd376b94f71f35b31d/brefeld.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-Level-SMOTE: Safe-level synthetic minority over-sampling technique for handling the class imbalanced problem. In T. Theeramunkong, B. Kijsirikul, N. Cercone, & T. B. Ho (Eds.) *Advances in Knowledge Discovery and Data Mining* (pp. 475–482). Berlin: Springer, https://doi.org/10.1007/978-3-642-01307-2_43.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). DBSMOTE: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, *36*(3), 664–684. https://doi.org/10.1007/s10489-011-0287-y.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953.

Cieslak, D., Chawla, N., & Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *2006 IEEE International Conference on Granular Computing,* pp. 732–737. https://doi.org/10.1109/GRC.2006.1635905.

Collell, G., Prelec, D., & Patil, K. R. (2018). A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing*, *275*, 330–340. https://doi.org/10.1016/j.neucom.2017.08.035.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30. http://jmlr.org/papers/v7/demsar06a.html.

Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C., & Kuncheva, L. I. (2015). Random balance: Ensembles of variable priors classifiers for imbalanced data. *Knowledge-Based Systems*, *85*, 96–111. https://doi.org/10.1016/j.knosys.2015.04.022.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, Boosting, and Randomization. *Machine Learning*, *40*, 139–157. https://doi.org/10.1023/A:1007607513941.

Douzas, G., & Bação, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. Information Sciences, 501. https://doi.org/10.1016/j.ins.2019.06.007.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on K-Means and SMOTE. *Information Sciences*, *465*(C), 1–20. https://doi.org/10.1016/j.ins.2018.06.056.

Erenel, Z., & Altınçay, H. (2013). Improving the precision-recall trade-off in undersampling-based binary text categorization using unanimity rule. *Neural Computing and Applications*, *22*(S1), 83–100. https://doi.org/10.1007/s00521-012-1056-5.

Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B., & Herrera, F. (2018a). Learning from Imbalanced Data Sets. Springer. https://doi.org/10.1007/978-3-319-98074-4.

Fernández, A., Garcia, S., Herrera, F., & Chawla, N. (2018b). SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905. https://doi.org/10.1613/jair.1.11192.

Galar, M., Fernández, A., Tartas, E. B., & Herrera, F. (2013). EUSBOost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, *46*(12), 3460–3471. https://doi.org/10.1016/j.patcog.2013.05.006.

Gazzah, S., Essoukri, B., & Amara, N. (2008). New oversampling approaches based on polynomial fitting for imbalanced data sets. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems,* pp. 677–684. https://doi.org/10.1109/DAS.2008.74.

Ghaderi Zefrehi, H., & Altınçay, H. (2020). Imbalance learning using heterogeneous ensembles. Expert Systems with Applications 142. https://doi.org/10.1016/j.eswa.2019.113005.

Gong, J., & Kim, H. (2017). RHSBoost: Improving classification performance in imbalance data. *Computational Statistics & Data Analysis*, *111*, 1–13. https://doi.org/10.1016/j.csda.2017.01.005.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239. https://doi.org/10.1016/j.eswa.2016.12.035.

Halimu, C., & Kasem, A. (2021). A novel ensemble method for classification in imbalanced datasets using split balancing technique based on instance hardness (sbal_IH). *Neural Computing and Applications*, *33*(17), 11,233–11,254. https://doi.org/10.1007/s00521-020-05570-7.

Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. *Adv Intell Comput*, *3644*, 878–887. https://doi.org/10.1007/11538059_91.

He, H., & Ma, Y. (2013). Imbalanced Learning: Foundations, Algorithms, and Applications, 1st edn. Wiley-IEEE Press.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence),* pp. 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969.

Hu, J., He, X., Yu, D. J., Yang XB, & Yang, H. B. S. J. Y. (2014). A new supervised over-sampling algorithm with application to protein-nucleotide binding residue prediction. *PLOS ONE*, *9*(9), 1–10. https://doi.org/10.1371/journal.pone.0107676.

Jiang, L., Qiu, C., & Li, C. (2015). A novel minority cloning technique for cost-sensitive learning. *Internations Journal of Pattern Recognition and Artificial Intelligence*, *29*, 1551,004:1–1551,004:18. https://doi.org/10.1142/S0218001415510040.

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *SIGKDD Explor Newsl*, *6*(1), 40–49. https://doi.org/10.1145/1007730.1007737.

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, *83*, 105,662. https://doi.org/10.1016/j.asoc.2019.105662.

Koziarski, M., & Wożniak, M. (2017). CCR: A combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer Science*, *27*(4), 727–736. https://doi.org/10.1515/amcs-2017-0050.

Koziarski, M., Krawczyk, B., & Woźniak, M. (2019). Radial-based oversampling for noisy imbalanced data classification. *Neurocomputing*, *343*, 19–33. https://doi.org/10.1016/j.neucom.2018.04.089.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232.

Krawczyk, B., Woźniak, M., & Herrera, F. (2014). Weighted one-class classification for different types of minority class examples in imbalanced data. In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM),* pp. 337–344. https://doi.org/10.1109/CIDM.2014.7008687.

Lango, M., & Stefanowski, J. (2022). What makes multi-class imbalanced problems difficult? an experimental study. *Expert Systems with Applications*, *199*, 116,962. https://doi.org/10.1016/j.eswa.2022.116962.

Li, K., Zhang, W., Lu, Q., & Fang, X. (2014). An improved SMOTE imbalanced data classification method based on support degree. In *International Conference on Identification, Information and Knowledge in the Internet of Things,* pp. 34–38. https://doi.org/10.1109/IIKI.2014.14.

Li, M., & Fan, S. (2017). CURE-SMOTE Algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinform*, *18*(1), 169,1–169,18. https://doi.org/10.1186/s12859-017-1578-z.

Liang, Y., Hu, S., Ma, L., & He, Y. (2009). MSMOTE: Improving classification performance when training data is imbalanced. Computer Science and Engineering, International Workshop on 2:13–17. https://doi.org/10.1109/WCSE.2009.756.

Ling, C., Sheng, V., & Yang, Q. (2006). Test strategies for cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering*, *18*(8), 1055–1067. https://doi.org/10.1109/TKDE.2006.131.

Lipton, Z., Elkan, C., & Naryanaswamy, B. (2014). Optimal Thresholding of Classifiers to Maximize F1 Measure. In *Machine learning and knowledge discovery in databases : European Conference, ECML PKDD : Proceedings ECML PKDD (Conference),* 8725. https://doi.org/10.1007/978-3-662-44851-9_15.

Menardi, G., & Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, *28*, 92–122. https://doi.org/10.1007/s10618-012-0295-5.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information System*, *46*(3), 563–597. https://doi.org/10.1007/s10844-015-0368-1.

Pozzolo, A. D., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *IEEE Symposium series on computational intelligence*, *SSCI2015*, (pp. 159–166). South Africa: Cape Town. https://doi.org/10.1109/SSCI.2015.33.

Puntumapon, K., & Waiyamai, K. (2012). A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling. In *Advances in Knowledge Discovery and Data Mining,* Springer Berlin Heidelberg, pp. 371–382. https://doi.org/10.1007/978-3-642-30220-6_31.

Sáez, J. A., Luengo, J., Stefanowski, J., & Herrera, F. (2015). SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, *291*, 184–203.

Siriseriwan, W., & Sinapiromsaran, K. (2017). Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling. *Songklanakarin Journal of Science and Technology*, *39*, 565–576. https://doi.org/10.14456/sjst-psu.2017.70.

Skurichina, M., & Duin, R. P. W. (2000). Boosting in linear discriminant analysis. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, *Springer-Verlag*, (pp. 190–199). Heidelberg: Berlin. https://doi.org/10.1007/3-540-45014-9_18.

Skurichina, M., & Duin, R. P. W. (2002). Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications*, *5*(2), 121–135.

Soleymani, R., Granger, E., & Fumera, G. (2020). F-measure curves: a tool to visualize classifier performance under imbalance. *Pattern Recognition*, *100*, 107,146. https://doi.org/10.1016/10.1016/j.patcog.2019.107146.

Tao, X., Li, Q., Guo, W., Ren, C., He, Q., Liu, R., & Zou, J. (2020). Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering. *Information Sciences*, *519*, 43–73. https://doi.org/10.1016/j.ins.2020.01.032.

Tarawneh, A. S., Hassanat, A. B. A., Almohammadi, K., Chetverikov, D., & Bellinger, C. (2020). SMOTE-FUNA: Synthetic minority over-sampling technique based on furthest Neighbour algorithm. *IEEE Access*, *8*, 59,069–59,082. https://doi.org/10.1109/ACCESS.2020.2983003.

Ting, K. M., Wells, J. R., Tan, S. C., Teng, S. W., & Webb, G. I. (2011). Feature-subspace aggregating: ensembles for stable and unstable learners. *Machine Learning*, *82*(3), 375–397. https://doi.org/10.1007/s10994-010-5224-5.

Torres, F. R., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2016). SMOTE-D a deterministic version of SMOTE In J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, V. Ayala Ramirez, J. A. Olvera-López, & X. Jiang (Eds.) *Pattern recognition*, (pp. 177–188). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39393-3_18.

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of International Joint Conference Artificial Intelligence*.

Wang, C., Deng, C., & Wang, S. (2020). Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters*, *136*, 190–197. https://doi.org/10.1016/j.patrec.2020.05.035.

Xie, Y., Qiu, M., Zhang, H., Peng, L., & Chen, Z. (2022). Gaussian distribution based oversampling for imbalanced data classification. *IEEE Transactions on Knowledge & Data Engineering*, *34*(02), 667–679. https://doi.org/10.1109/TKDE.2020.2985965.