Machine Learning
https://doi.org/10.1007/s10994-024-06563-6

Local neighborhood encodings for imbalanced data classification

Michał Koziarski[1] · **Michał Woźniak**[1]

Received: 5 July 2022 / Revised: 14 March 2024 / Accepted: 26 April 2024

Abstract
This paper aims to propose *Local Neighborhood Encodings* (LNE)-a hybrid data preprocessing method dedicated to skewed class distribution balancing. The proposed LNE algorithm uses both over- and undersampling methods. The intensity of the methods is chosen separately for each fraction of minority and majority class objects. It is selected depending on the type of neighborhoods of objects of a given class, understood as the number of neighbors from the same class closest to a given object. The process of selecting the over- and undersampling intensities is treated as an optimization problem for which an evolutionary algorithm is used. The quality of the proposed method was evaluated through computer experiments. Compared with SOTA resampling strategies, LNE shows very good results. In addition, an experimental analysis of the algorithms behavior was performed, i.e., the determination of data preprocessing parameters depending on the selected characteristics of the decision problem, as well as the type of classifier used. An ablation study was also performed to evaluate the influence of components on the quality of the obtained classifiers. The evaluation of how the quality of classification is influenced by the evaluation of the objective function in an evolutionary algorithm is presented. In the considered task, the objective function is not *de facto* deterministic and its value is subject to estimation. Hence, it was important from the point of view of computational efficiency to investigate the possibility of using for quality assessment the so-called proxy classifier, i.e., a classifier of low computational complexity, although the final model was learned using a different model. The proposed data preprocessing method has high quality compared to SOTA, however, it should be noted that it requires significantly more computational effort. Nevertheless, it can be successfully applied to the case as no very restrictive model building time constraints are imposed.

Keywords Machine learning · Imbalanced data · Oversampling · Undersampling · Evolutionary algorithm

Editors: Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Shuo Wang.

Extended author information available on the last page of the article

1 Introduction

The problem of imbalanced data classification is one of the important subareas of machine learning research. This is dictated by the fact that most real-world decision problems are of this nature, i.e., a significant disparity between the object counts of the different classes. In the case of a two-class classification task to which this work is devoted, the relationship is quite obvious, i.e., one class with a large number of objects is called the majority class, and a class with a small number of instances is called the minority class. The critical point here is that, as a rule, the cost of incorrectly classified objects of the minority class is much higher than the error made on majority class instances. We should be aware that in the case of multiclass problems, such relation is no longer obvious, and minority-majority relations may occur only between some pairs of classes. Moreover, a given class may be a majority class concerning selected classes, simultaneously a minority class for others. This problem will not be considered in this paper and we will focus on the binary classification task. For imbalanced data, it is observed that canonical classifier learning methods that do not explicitly optimize the quality criterion prefer models biased towards the majority class. However, in the cases in which we can provide a learning criterion that can account for the different cost of errors for each class, the problem is obtaining such information, e.g., in the form of a loss matrix, from the user (Branco et al., 2016). The question is whether indeed the disparity between classes is the most significant difficulty or whether other factors affecting the difficulty of the data may be crucial since it is not difficult to imagine a decision problem that, even despite the significant disparity between classes, is easy to classify using traditional methods (see Fig. 1).

It seems that difficulties related to the classification of imbalanced data should be sought in characteristics of conditional probability distributions of classes. Napierala and Stefanowski (2016) noticed that, especially in the case of minority classes, they tend to form small, disconnected clusters, which combined with the small number of objects in the minority class, causes additional difficulty in correctly learning the models. One may find several taxonomies related to evaluating the difficulty of classifying minority class objects. As a rule, one distinguishes between the fraction of objects that do not present problems with correct classification (usually known as safe examples) and the remaining instances as unsafe. A popular taxonomy is to determine the difficulty of objects of a minority class

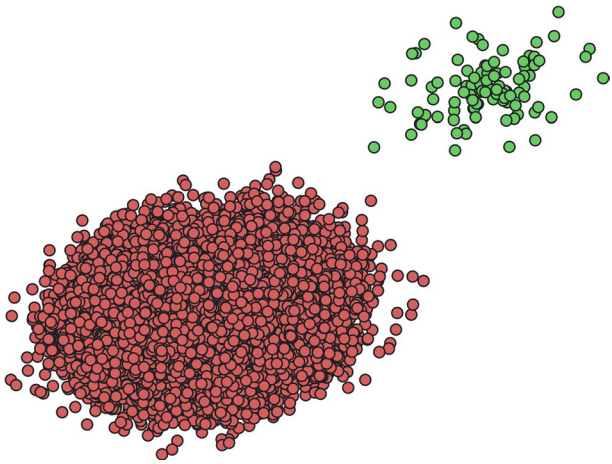

Fig. 1 Easy classification task for imbalance ratio 1:1000

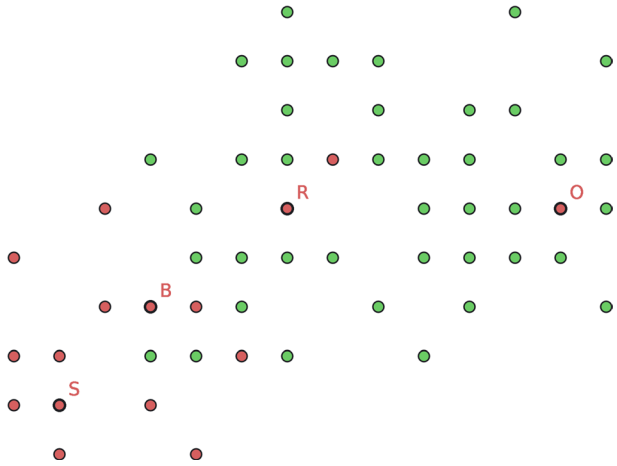

Fig. 2 Examples of different minority class instances: "S" stands for safe, "B" for borderline, "R" for rare, and "O" for outliers

using the number of objects of the minority class found among the five nearest neighbors of a given object from this class. This leads to a division into:

- Examples lying near each other can be considered as *safe* ones.
- Instances close to the border between classes, where examples of different classes may overlap, are identified as *borderline* examples.
- Groups of a few examples of a class in areas of other classes are known as *rare* examples.
- *Outliers* are completely isolated examples of a concrete class.

Figure. 2 shows a sample illustration of different types of minority class objects.

This paper will present the Local Neighborhood Encodings (LNE) algorithm, a hybrid algorithm for preprocessing imbalanced data. LNE uses both oversampling and undersampling methods. The intensity of these methods is separate for each fraction of minority and majority class objects and is chosen according to the neighborhood type of the class instance, defined as the number of neighbors of the same class closest to the given object. An evolutionary algorithm is used to solve this optimization task.

In brief, the most important contributions of this work are:

- Proposition of LNE - a hybrid preprocessing method for imbalanced data.
- Formulation of an optimization task and representation for a selected evolutionary algorithm.
- Analysis of the effect of classification task characteristics such as imbalance ratio or size of each object type fractions on the quality of the proposed method.
- Discussion on the data set characteristics returned by LNE.
- Experimental evaluation of the quality of the LNE for different types of classifiers and the impact of using a simple proxy classification model required for feature value estimation in the optimization algorithm.
- Experimental evaluation of the proposed approach based on diverse benchmark datasets and a detailed comparison with the state-of-art approaches
- Discussion on an ablation study to demonstrate how the various components of the LNE affect its quality.

2 Related work

The problem of imbalanced data is encountered in a significant fraction of real-world problems and in recent years, the main focus has been on the analysis of tabular data. However, a growing number of works indicate that this problem is also very important for image classification since many of the datasets are characterized by this type of property, making generalizations of deep models difficult (Johnson & Khoshgoftaar, 2019; Kim et al., 2021).

The preference for the majority class by classification models can be mitigated by modifying the training data to bring the minority class size in line with the majority class size or by modifying the classifier's learning or decision-making process to account for class disparity and thus increase the sensitivity of decisions towards the minority class.

We may divide the techniques used to deal with data imbalance into two categories. In the first approach (also called algorithm-level solutions), the problem of data imbalance is considered at the stage of the learning classifier, e.g., considering the different cost of error between different classes in the learning criterion. In contrast, the second approach (data-level solutions) does not interfere with the learning of the classifier itself but modifies the learning set before the learning process starts to compensate for the differences in the number of examples from each class.

In this paper, we will mainly focus on the second approach. Hence let us characterize the most important techniques associated with them. In general, these methods try to remove objects from the majority class (undersampling) or increase the size of the minority class (oversampling). In this regard, it should be noted that these actions may be purely random, or the process of removing objects of the majority class or adding instances of the minority class may result from an analysis of the distributions of the different fractions of objects.

Randomized preprocessing methods are easy to implement and have low computational complexity. However, it is essential to realize that in some cases, they may have an adverse effect on the dataset. Random Undersampling (RUS) can lead to the rejection of important instances for creating the correct class boundary or lie in specific subregions of the target class. On the other hand, random oversampling can lead to the reallocation of noisy instances and thus inappropriately affect the actual class distribution. The most straightforward sampling-based approach to data imbalance is Random Oversampling (ROS). Using ROS, new objects are generated by replicating randomly selected existing objects. The disadvantage of this approach is that it leads to the clustering of minority objects in small areas where the original instances were located. This can be a problem for some classifiers, especially those that tend to overfit. An interesting approach is Lee proposed oversampling method (Lee, 2000), which produces noisy replicas of minority class objects while keeping the majority class unchanged.

Although randomized methods often have good results, it should be noted that many authors try to develop strategies that add synthetic minority examples or remove majority instances in a guided manner.

For nearly two decades, the most popular method for creating new instances of a minority class has been the SMOTE algorithm (Fernández et al., Jan 2018). It involves randomly generating objects between minority class instances. Although SMOTE works well in many practical applications, it has been noted that it can lead to a change in the distribution of the minority class, resulting in overfitting of the classifier. Therefore, several modifications have been proposed, specifically turning around the fact that minority class objects are not generated in areas potentially belonging to the majority class. For instance, *Borderline* SMOTE (Han et al., 2005) generates synthetic minority class objects near the

decision boundary. In contrast, SafeLevelSMOTE (Bunkhumpornpat et al., 2009) and LN-SMOTE (Maciejewski & Stefanowski, 2011) avoid generating minority class instances in areas where objects belonging to the majority class dominate. Other popular methods include ADASYN (He et al., 2008), which generates synthetic objects by taking into account which areas are difficult to classify and therefore increases the number of minority class instances generated in those areas. Also, Sáez et al. (2016) proposed that the fraction of objects subject to oversampling for each particular problem should be chosen according to their difficulty type. At the same time, the rest should be left unmodified.

It is also worth noting RBO (*Radial-Based Oversampling*) (Koziarski et al., 2017), which estimates the mutual distribution of minority and majority class objects using potential functions to select the area of minority class object generation. Koziarski also proposed *Potential Anchoring* algorithm (Koziarski, 2021a), which also uses potential functions to ensure invariant probability distributions across classes during the resampling process. On the other hand, CCR (*Combined Cleaning and Resampling*) (Koziarski & Woźniak, 2017), combines two techniques - cleaning the area around minority class objects by removing majority class objects from their neighborhood and generating synthetic minority class objects in that area.

As mentioned, RUS carries the risk of removing important objects from the majority class, which may lead to constructing a classifier that ignores less dense clusters of the majority class. Several methods have been proposed to avoid this tendency. They try to analyze the local mutual distributions of the minority and majority classes during undersampling. For example, ENN (*Edited Nearest Neighbor*) removes majority examples if it finds the homogeneity of a given sample neighborhood. RBU (*Radial Based Undersampling*) (Koziarski, 2020) employs a previously introduced concept of mutual class potential. SMUTE (*Synthetic Minority Undersampling Technique*) utilizes the data interpolation used in SMOTE to reduce the number of observations from the majority class (Koziarski, 2021b).

Many methods try to combine both techniques, i.e., over and undersampling. For example, CSMOUTE (*Combined Synthetic Oversampling and Undersampling Technique*) (Koziarski, 2021b) combines the mentioned SMUTE method with SMOTE. On the other hand, Galar et al. proposed a technique for combining *under-* and *oversampling* with the classifier ensemble (Galar et al., 2012). Other authors are keen on using data preprocessing techniques by combining them with algorithm-level solutions, mainly based on techniques having their roots in classifier ensembles built on perturbed learning sets. These methods can be overviewed in Fernández et al. (2018).

Some of the works try to treat the guided data preprocessing as an optimization task. Kim et al. (2016) proposed a hybrid method using a clustering technique and genetic algorithms (GA) based on the artificial neural networks model to balance imbalanced data distribution. Barandela et al. (2005) employed a genetic algorithm to balance data distribution and perform feature selection simultaneously. On the other hand, Khoshgoftaar et al. (Jan 2010) proposed using an evolutionary algorithm for the undersampling task. Garcia and Herrera also used the evolutionary algorithm to propose a family of undersampling techniques EUS(*Evolutionary Undersampling*) (García & Herrera, 2009). Later this concept was extended by Wojciechowski, who applied the multicriteria optimization algorithm NSGA-2 (Deb et al., 2002) to the undersampling task (Wojciechowski, 2021). Multicriteria optimization approach was additionally considered by Węgier et al. in the tasks of building ensembles (Węgier et al., 2022) and providing interpretable decision trees (Węgier et al., 2023). Hualong et al. (2013) used ant colony optimization to improve imbalanced DNA microarray data classification performance. Metaheuristic algorithms are also used successfully for oversampling algorithms. The examples include GenSample (Karia et al., 2019), based on and, ACO Resampling (Li et al., 2020), which uses *Ant Colony Optimization*.

When using data balancing methods, one must also ask what degree of balancing one wants to achieve. Most works try to reach an equal number of minority and majority class instances. However, for some classifiers, such as decision trees, it has been shown that this approach does not give the best results (Weiss & Provost, 2003). Moreover, Khoshgoftaar et al. have shown experimentally that when the imbalance ratio is large, the balancing process should be stopped for IR values between 2:1 and 3:1 (Khoshgoftaar et al., 2007). Although this characteristic is well known, most authors seem to ignore it in their studies.

A more comprehensive review of work related to data balancing methods or not discussed so-called algorithm-level solutions might be found in the review papers (Branco et al., 2016; Krawczyk, 2016).

3 Local neighborhood encodings

Our approach is inspired by the categorization of observation types proposed by Napierała and Stefanowski (Napierala & Stefanowski, 2016), and a further study by Sáez et al. (2016), in which selective oversampling of different observation types was analyzed. In a nutshell, the proposed categorization was based on five nearest neighbour taxonomy presented in the previous section. The aforementioned study by Sáez et al. later used this categorization in an approach in which only the observations from specific types are used for oversampling. They used the same division into safe, borderline, rare and outlier observations, and exhaustively

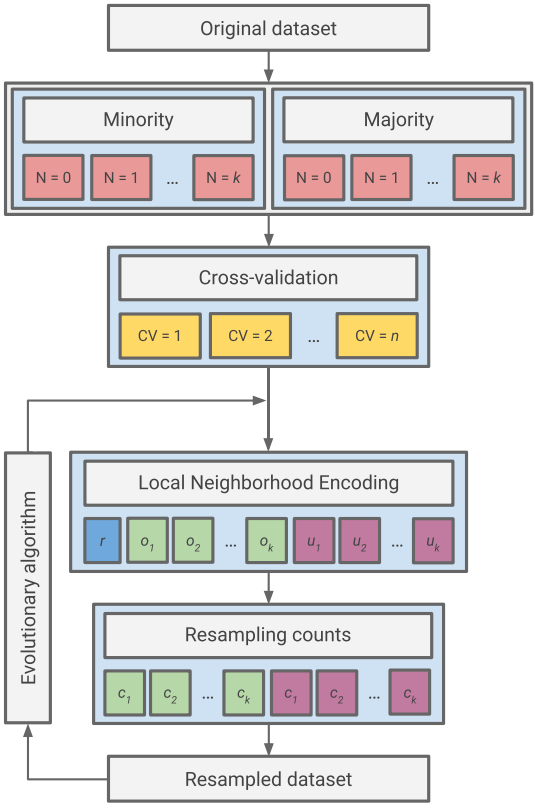

Fig. 3 Schematic drawing of the proposed approach. Firstly, the original dataset is divided into bags based on the number of same class nearest neighbors, and afterwards into several cross-validation folds. Then, evolutionary algorithm is used to optimize Local Neighborhood Encodings coding the number of observations with specific number of same class nearest neighbors that will be over- and undersampled. Finaly LNE resamples the original dataset

evaluated all 16 (2^4) combinations of different minority observation types that can be used. They were able to experimentally demonstrate that limiting oversampling to specific observation types can improve the performance of the imbalanced data classification algorithms.

We extend the idea of selective resampling. First, we take advantage of the fact that some previous studies have shown that a combination of oversampling and undersampling can improve performance compared to an approach using only one (Koziarski, 2021a, 2021b). For this reason, we will combine these techniques and consider the intensity of each process as the parameter being optimized. Second, the optimization process will also determine the oversampling intensity of each type of minority class observation. Moreover, based on the observations pointed out, among others, by Khoshgoftaar et al. (2007), the imbalance ratio of the final set will also be treated as a parameter to be determined in our algorithm.

To implement the above ideas we designed an approach based on the evolutionary optimization of a *Local Neighborhood Encoding*, a real-valued vector of numbers encoding the number of observations from specific types that will be either created via oversampling or removed via undersampling. We present the high-level overview of the approach in Fig. 3, and a detailed pseudo code in Algorithm 1.

Algorithm 1 Local neighborhood encodings

Input:
$\mathcal{X} = \mathcal{X}_{maj} \cup \mathcal{X}_{min}$ - the collection of original observations, where $\mathcal{X}_{maj}$ and $\mathcal{X}_{min}$ denote subsets of majority and minority observations
clf - classifier
f - optimization criterion
Parameters:
k - number of nearest neighbors used for observation difficulty calculation
ps - population size
$iterations$ - number of iterations of the optimization algorithm
n_f - number of cross-validation folds
r_{max} - maximum oversampling ratio
Output:
$\mathcal{X}'$ - the collection of resampled observations

function LNE($\mathcal{X}_{maj}$, $\mathcal{X}_{min}$, clf, f):
  $nn^v \leftarrow$ vector with nn^v_i containing the number of same class observations in a k-neighborhood of $\mathcal{X}_i$
  $\mathcal{F} \leftarrow$ collection of n_f cross-validation folds, with $\mathcal{F}_i = (\mathcal{X}_{train}, nn^v_{train}, \mathcal{X}_{valid})$
  $\mathcal{P} \leftarrow$ starting population, randomly initialized from $[0; 1]^{ps \times (2k+1)}$
  for i in 1..$iterations$ **do**
    for j in 1..ps **do**
      $r, o_1, ..., o_k, u_1, ..., u_k \leftarrow \mathcal{P}_j$ split into oversampling ratio r, oversampling coefficient o_m for $x_n \in \mathcal{X}_{min} : nn^v_n = m$, and undersampling coefficient u_m for $x_n \in \mathcal{X}_{maj} : nn^v_n = m$
      $n_{over} \leftarrow r \cdot r_{max} \cdot (|\mathcal{X}_{maj}| - |\mathcal{X}_{min}|)$: total number of oversampled observations
      $c^o_m \leftarrow n_{over} \cdot \frac{o_m}{\sum_{l=1}^{k} o_l}$: oversampling count c^o for observation type m
      $c^u_m \leftarrow (1 - u_m) \cdot |x_n \in \mathcal{X}_{maj} : nn^v_n = m|$: undersampling count c^u for observation type m
      for $(\mathcal{X}_{train}, nn^v_{train}, \mathcal{X}_{valid})$ in $\mathcal{F}$ **do**
        $\mathcal{X}'_{train} \leftarrow \mathcal{X}_{train}$ resampled using counts $c^o_1, ..., c^o_k, c^u_1, ..., c^u_k$
        fit clf using $\mathcal{X}'_{train}$
        evaluate clf on $\mathcal{X}_{valid}$ w.r.t. f
      end for
      $S_j \leftarrow$ average criterion f over all folds
    end for
    $\mathcal{P}_{best} \leftarrow \arg\max_{S_j} \mathcal{P}_j$
    perform optimization step on $\mathcal{P}$ w.r.t. S and update $\mathcal{P}$
  end for
  $\mathcal{X}' \leftarrow \mathcal{X}$ resampled using $\mathcal{P}_{best}$
  return $\mathcal{X}'$

Let us first describe how we will encode the strength of resampling for each observation type. It is worth noting that there is a major practical distinction between the over- and undersampling, as the former is unbounded. In principle, we can generate synthetic observations *ad infinitum*. On the other hand, in the case of undersampling, there is a clear bound equal to the number of the original observations. Because of that, the two of them will have to be encoded differently. Specifically, the approach we propose is based on coding all of the information required to produce the resampling counts (that is the numbers of observations from specific types that will be either over- or undersampled) as a $2k + 1$ element vector, with k being the parameter describing the size of neighborhood used for type calculation.

$$\left[r, o_1, ..., o_k, u_1, ..., u_k\right]$$

The first element r of this vector encodes the strength of oversampling, i.e., the total number of observations generated via oversampling will be equal to $n_{over} = r \cdot r_{max} \cdot \left(|\mathcal{X}_{maj}| - |\mathcal{X}_{min}|\right)$, with r_{max} being a hyperparameter bounding the number of oversampled observations (which in practice we will set to be fairly high, i.e., equal to 5, to allow for an oversampling strength search within this bound), and $\mathcal{X}_{maj}$ and $\mathcal{X}_{min}$ being the collections of majority and minority observations, respectively. Next k elements $o_1, o_2, ..., o_k$ encode the relative strength of oversampling from particular observation types, with the resulting number of observations generated based on a specific observation type m defined as $c^o_m = n_{over} \cdot \frac{o_m}{\sum_{l=1}^{k} o_l}$. Finally, the last k elements $u_1, u_2, ..., u_k$ encode the proportion of majority class observations with a type m that will be removed during undersampling. Note that with such an approach, all elements of the encoding vector can be bound within the range [0; 1].

Given such an encoding scheme, we can formulate the resampling process as an optimization procedure of the encoding for the given dataset. Proposed solutions will be evaluated on the cross-validation folds, with the encoding used to obtain observation type-dependent resampling counts for a specific training fold. Based on the resulting counts, resampling will be performed on the training fold, an estimator fitted on it, and the performance evaluated on the test fold. The final performance score will be the average of scores across folds. Specifically, we will evaluate the quality of a given encoding using 3×2 cross-validation (Raschka, 2018), and any desired target metric as an optimization criterion. While, in principle, various oversampling, undersampling, and optimization algorithms can be used, in this paper we will employ SMOTE (Chawla et al., 2002) for oversampling (with random observations from a currently considered observation type used as the starting points), random undersampling as the undersampling algorithm, and Differential Evolution (Price et al., 2006) as the optimization algorithm. Once the optimization procedure is finished, we will use the resulting encoding to resample the whole training dataset.

Finally, it is worth noting that one advantage of the algorithm being formulated in such a way is that the encodings are, to some extent, interpretable: they describe not only the overall degree of balancing (with the dataset being either balanced completely, only partially, or overbalanced, with the old minority class becoming the new majority), the proclivity towards either over- or undersampling, and towards focusing the resampling on specific types of observations. In particular, the last one can be viewed as desirable since the idea of focusing the resampling on specific observation types is present in a number of existing approaches, such as Borderline-SMOTE (Han et al., 2005) and Safe-Level-SMOTE (Bunkhumpornpat et al., 2009). However, while the approaches are out of necessity contradictory, there is little justification to prefer any of them *a*

priori, not in the context of a specific dataset. An analysis of the encodings produced during the optimization for a larger dataset body could shed some light on the trends associated with focusing the resampling on specific observation types: this idea will be later revisited in Sect. 4.5.

4 Experimental study

To evaluate the proposed approach's usefulness and properties, we conducted an experimental study. It aims to answer the following research questions:

RQ1: How does LNE compare with state-of-the-art resampling strategies?
RQ2: What design choices are responsible for the performance of LNE?
RQ3: Can LNE be sped up using less computationally expensive proxy estimators?
RQ4: Can the solutions found by LNE provide any generalizable insights into imbalanced data resampling?

4.1 Set-up

Data. Conducted experimental study was based on the binary imbalanced datasets provided in the KEEL repository (Alcalá-Fdez et al., 2011), with a total of 60 datasets used. Their details were presented in Table 1. In addition to the imbalance ratio (IR), the number of samples and the number of features, for each dataset we computed the data difficulty index (DI) (Koziarski, 2021a) using $m = 5$ nearest neighbors, which is a [0; 1] bounded functions measuring the difficulty of a given dataset. Prior to resampling and classification, each dataset was preprocessed: categorical features were encoded as integers, and afterwards all features were standarized by removing the mean and scaling to unit variance.

Classification. Four different classification algorithms were used throughout the experimental study: CART decision tree, k-nearest neighbors classifier (KNN), support vector machine (SVM) and multi-layer perceptron (MLP). The implementations of the classification algorithms provided in the scikit-learn machine learning library (Pedregosa et al., 2011) were utilized. Used hyperparameters of the classification algorithms were presented in Table 2.

Reference resampling methods. We considered several other state-of-the-art resampling strategies. We based our choice on a recent ranking constructed by (Kovács, 2019), out of which we selected the following best-performing methods: SMOTE (Chawla et al., 2002), Polynomial Fitting SMOTE (pf-SMOTE) (Gazzah et al., 2008), Oversampling with Rejection (Lee) (Lee et al., 2015), Synthetic Minority Oversampling Based on Sample Density (SMOBD) (Cao et al., 2011), Partially Guided Oversampling (G-SMOTE) (Sandhan & Choi, 2014), Learning Vector Quantization-based SMOTE (LVQ-SMOTE) (Nakamura et al., 2013), Assembled SMOTE (A-SMOTE) (Zhou et al., 2013) and SMOTE combined with Tomek Links (SMOTE-TL) (Batista et al., 2004). The implementations of the reference methods provided in the smote-variants library (Kovács, 2019) were utilized. Used hyperparameters of the resampling algorithms were presented in Table 2. For all of the reference methods we adjusted the proportion of oversampling using 3×2 cross-validation, selecting values of oversampling proportion from $\{0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$, with 1.0 indicating resampling up to the point of achieving balanced class distributions.

Evaluation. For every dataset, we reported the results averaged over the 5×2 cross-validation folds (Alpaydin, 1999).

Let us define the used metrics. Firstly, let's define the confusion matrix, which summarizes the number of instances from each class classified correctly or incorrectly as the remaining classes (see Table 3).

On the basis of the confusion matrix, we may define

$$recall = \frac{TP}{TP + FN} \tag{1}$$

that is also known as *sensitivity*.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$specificity = \frac{TN}{TN + FP} \tag{3}$$

Throughout the experimental study we reported the values of AUC, balanced accuracy (BAC), G-mean, and F_β score (F-beta)

$$G - mean = \sqrt{precision + recall} \tag{4}$$

$$BAC = \frac{sensitivity + specificity}{2} \tag{5}$$

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{6}$$

The parameter β can be tuned for different trade-offs between both components. Nevertheless, using such metrics could be dangerous because β should be appropriately set. Brzezinski et al. (2018) showed that inappropriate parameter setting for $F_\beta score$ may favor the majority class for the imbalanced data classification task. During the experiments, β has been chosen individually for each dataset and equal to its imbalance ratio (Stapor et al., 2021). In our experiments we also used AUC under the precision-recall curve that is computed on predicted probabilities.

Implementation and reproducibility. The experiments described in this paper were implemented in the Python programming language. Complete code, sufficient to repeat the experiments, as well as complete results in a CSV format, and tables showing average performance on each dataset, separately for every classifier and performance metric, were made publicly available at.[1] We used the *scikit-learn* (Pedregosa et al., 2011) library to implement the experimental protocol, performance metrics, and classifiers.

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

Table 1 Summary of the characteristics of datasets used throughout the experimental study

Name	DI	IR	Samples	Features	Name	DI	IR	Samples	Features
kddcup-buffer_overflow_vs_back	0.10	73.43	2233	41	cleveland-0_vs_4	0.60	12.31	173	13
kddcup-rootkit-imap_vs_back	0.17	100.14	2225	41	poker-8–9_vs_6	0.64	58.40	1485	10
ecoli2	0.20	5.46	336	7	haberman	0.67	2.78	306	3
page-blocks0	0.22	8.79	5472	10	yeast-0–5-6–7-9_vs_4	0.68	9.35	528	8
page-blocks-1–3_vs_4	0.23	15.86	472	10	zoo-3	0.68	19.20	101	16
glass0	0.27	2.06	214	9	yeast-0–3-5–9_vs_7–8	0.68	9.12	506	8
ecoli-0–1_vs_2–3-5	0.28	9.17	244	7	flare-F	0.69	23.79	1066	11
kr-vs-k-zero_vs_eight	0.29	53.07	1460	6	abalone-21_vs_8	0.70	40.50	581	8
ecoli-0–1-4–7_vs_2–3-5–6	0.29	10.59	336	7	yeast-1_vs_7	0.71	14.30	459	7
ecoli1	0.33	3.36	336	7	poker-8_vs_6	0.72	85.88	1477	10
yeast3	0.35	8.10	1484	8	poker-9_vs_7	0.75	29.50	244	10
ecoli-0–6-7_vs_5	0.35	10.00	220	6	yeast4	0.76	28.10	1484	8
yeast-2_vs_4	0.39	9.08	514	8	glass-0–1-4–6_vs_2	0.78	11.06	205	9
glass1	0.40	1.82	214	9	glass-0–1-6_vs_2	0.80	10.29	192	9
ecoli-0–6-7_vs_3–5	0.40	9.09	222	7	glass2	0.81	11.59	214	9
yeast5	0.40	32.73	1484	8	abalone9–18	0.82	16.40	731	8
ecoli-0–2-6–7_vs_3–5	0.42	9.18	224	7	yeast-1–2-8–9_vs_7	0.82	30.57	947	8
glass-0–1-6_vs_5	0.42	19.44	184	9	glass-0–1-5_vs_2	0.83	9.12	172	9
ecoli-0–1-3–7_vs_2–6	0.43	39.14	281	7	abalone-17_vs_7–8-9–10	0.84	39.31	2338	8
yeast-2_vs_8	0.46	23.10	482	8	yeast-1–4-5–8_vs_7	0.89	22.10	693	8
vehicle1	0.48	2.90	846	18	abalone-20_vs_8–9-10	0.89	72.69	1916	8
pima	0.48	1.87	768	8	winequality-red-3_vs_5	0.90	68.10	691	11
glass4	0.49	15.46	214	9	winequality-red-4	0.92	29.17	1599	11
ecoli3	0.50	8.60	336	7	winequality-red-8_vs_6	0.92	35.44	656	11
yeast-0–2-5–6_vs_3–7-8–9	0.51	9.14	1004	8	winequality-white-3_vs_7	0.93	44.00	900	11
glass5	0.51	22.78	214	9	winequality-white-3–9_vs_5	0.94	58.28	1482	11
yeast6	0.54	41.40	1484	8	winequality-red-8_vs_6–7	0.94	46.50	855	11

Table 1 (continued)

Name	DI	IR	Samples	Features	Name	DI	IR	Samples	Features
yeast1	0.54	2.46	1484	8	abalone-19_vs_10-11-12-13	0.96	49.69	1622	8
vehicle3	0.55	2.99	846	18	poker-8–9_vs_5	0.97	82.00	2075	10
winequality-white-9_vs_4	0.60	32.60	168	11	abalone19	0.97	129.44	4174	8

Table 2 Parameters of the classification and the sampling algorithms used throughout the experimental study

Algorithm	Parameters
CART	Criterion: Gini impurity
KNN	k-nearest neighbors = 3
SVM	Kernel: RBF
	C = 1.0
MLP	Hidden neurons = 100
	Activation: ReLU
	Optimizer: Adam
	Learning rate = 0.001
LNE	$k = 4$
	$r_{max} = 5.0$
	CV = 3×2
	Population size = 100
SMOTE	k-nearest neighbors = 5
pf-SMOTE	Topology: star
Lee	k-nearest neighbors = 5
	Rejection level = 0.5
SMOBD	$\eta_1 = 0.5$
	Noise threshold $t = 1.8$
G-SMOTE	k-nearest neighbors = 5
LVQ-SMOTE	k-nearest neighbors = 5
	n clusters = 10
A-SMOTE	k-nearest neighbors = 5
	Population parameter = 2
SMOTE-TL	k-nearest neighbors = 5

Table 3 Confusion matrix for two-class classification task

	Predicted positive	Predicted negative
Positive class	True positive (*TP*)	False negative (*FN*)
Negative class	False positive (*FP*)	True negative (*TN*)

4.2 Comparison with reference methods

We began the experimental analysis by comparing the proposed approach to several state-of-the-art reference resampling methods. Average ranks together with the results of statistical comparison using Friedman test combined with Shaffer's post-hoc, reported at a significance level $\alpha = 0.05$, were presented in Table 4. As can be seen, when compared with respect to BAC, G-mean, and F-beta metrics, the proposed LNE approach outperformed the reference methods in every case, usually achieving statistically significantly better results, demonstrating the general usefulness of the proposed approach. However, it is worth mentioning that we observed no statistically significant differences when the comparison was made using AUC (under the precision-recall curve). It is not entirely clear what caused this behavior. One possible explanation is that the results computed with a metric using probability scores instead of binarized predictions are generally more stable

and less susceptible to data imbalance. The other reason is that using probability-based metrics is more difficult to optimize and/or more susceptible to overfitting, reducing the final performance of the model.

To better illustrate the performance of the proposed approach we also conducted pairwise comparisons on individual datasets, comparing the results on all 10 cross-validation folds using Wilcoxon rank-sum test at a significance level $\alpha = 0.10$. The number of datasets for which LNE achieved statistically significantly better or worse results was presented in Fig. 4. As can be seen, there is a clear discrepancy depending on the performance metric chosen, with using F-beta (which should penalize the predictions biased towards the majority class most heavily) leading to most wins, BAC and G-mean to a medium amount, and AUC to the least; with the number of losses low in every case.

4.3 Ablation study

Having established that the proposed LNE approach can outperform the reference methods, we next tried to examine what specific design choices led to this outperformance. We performed an ablation study to compare LNE with its two different, simplified variants:

- LNE_{NRS} that is LNE with *no ratio selection*, for which the balancing was always performed up achieving a perfectly balanced distribution (or, in other words, for which the optimization of balancing ratio was disabled).
- LNE_{NTS} that is LNE with *no type selection*: a variant in which the resampling was based on all available observation types, and only the balancing ratio was optimized.

Notably, while LNE_{NRS} differed only in assigning the resampling counts but still used $2k + 1$ values as the encoding. LNE_{NTS} variant was encoded using only two values: oversampling and undersampling ratio.

Note that due to a high computational complexity in this comparison we excluded the MLP classifier because of its long training time.

Let's present the average ranks and p values obtained when comparing the ablation variants with the baseline LNE: LNE - 1.83, LNE_{NRS} - 2.20, and LNE_{NTS} - 1.96. To determine whether the rank differences are statistically significant, the Friedman paired posthoc Shaffer test was performed, and the following results were obtained: LNE vs. LNE_{NRS} (p-value < 0.001), LNE vs. LNE_{NTS} (p-value $= 0.015$). As can be seen, the baseline LNE using both mechanisms (type and ratio selection) achieved statistically significantly better results than the ablation variants. The discrepancy was most significant in comparison to the ablation variant with no ratio selection, indicating that it was the major contributor to the outperformance of LNE over the reference algorithms. In contrast, type selection was a minor contribution: we suspect this is because the variant with no type selection had only two optimizable parameters, thus being much less prone to overfitting. Still, type selection can still be potentially used as a vehicle for meta-analysis of the algorithm's behavior, which will be discussed in Sect. 4.5. Finally, it is worth noting that we aggregated the results over all classifiers and performance metrics to illustrate the general trends better. Still, they were not consistent for particular classifier and performance metric combinations in every case.

Table 4 Average ranks of the evaluated methods calculated for all of the considered datasets

	Metric	SMOTE	pf-SMOTE	Lee	SMOBD	G-SMOTE	LVQ-SMOTE	A-SMOTE	SMOTE-TL	LNE
CART	AUC	5.38	4.76	4.82	4.94	5.04	5.58	5.48	4.82	**4.18**
	BAC	5.18 $_+$	6.05 $_+$	5.08 $_+$	5.04 $_+$	5.79 $_+$	4.68 $_+$	5.05 $_+$	5.29 $_+$	**2.82**
	G-mean	5.11 $_+$	6.44 $_+$	5.25 $_+$	5.36 $_+$	6.04 $_+$	3.95 $_+$	5.18 $_+$	5.52 $_+$	**2.16**
	F-beta	5.46 $_+$	6.72 $_+$	5.25 $_+$	5.48 $_+$	6.51 $_+$	3.55 $_+$	5.12 $_+$	5.55 $_+$	**1.38**
KNN	AUC	5.12	**3.53**	5.48	5.14	5.38	6.56 $_+$	4.85	4.28	4.67
	BAC	4.72 $_+$	5.58 $_+$	5.12 $_+$	5.04 $_+$	7.16 $_+$	5.87 $_+$	4.72 $_+$	4.46 $_+$	**2.33**
	G-mean	4.65 $_+$	6.02 $_+$	5.08 $_+$	4.82 $_+$	7.55 $_+$	5.63 $_+$	4.93 $_+$	4.53 $_+$	**1.80**
	F-beta	4.86 $_+$	6.54 $_+$	5.11 $_+$	4.98 $_+$	7.83 $_+$	5.23 $_+$	4.96 $_+$	4.40 $_+$	**1.08**
SVM	AUC	4.71	5.52	**4.54**	4.78	5.34	4.78	4.72	4.66	5.95
	BAC	4.68	6.94 $_+$	4.98 $_+$	5.25 $_+$	6.20 $_+$	3.46	5.27 $_+$	4.81	**3.42**
	G-mean	4.73	6.82 $_+$	5.03 $_+$	5.16 $_+$	6.28 $_+$	3.34	5.32 $_+$	5.02 $_+$	**3.32**
	F-beta	5.15 $_+$	8.02 $_+$	5.15 $_+$	5.35 $_+$	6.34 $_+$	3.18 $_+$	5.38 $_+$	5.03 $_+$	**1.39**
MLP	AUC	4.61	5.96	**4.25**	4.67	5.57	5.55	4.38	4.40	5.62
	BAC	4.88 $_+$	6.94 $_+$	4.75 $_+$	5.01 $_+$	6.00 $_+$	4.48 $_+$	5.36 $_+$	4.83 $_+$	**2.75**
	G-mean	4.97 $_+$	7.15 $_+$	4.37 $_+$	5.12 $_+$	6.28 $_+$	4.15	5.52 $_+$	4.72 $_+$	**2.73**
	F-beta	4.95 $_+$	7.69 $_+$	4.99 $_+$	5.34 $_+$	6.56 $_+$	3.70 $_+$	5.42 $_+$	4.86 $_+$	**1.48**

Best performance was denoted with bold font. Methods compared to which LNE achieved a statistically significantly better results (according to Shaffer's post-hoc test) were denoted with a + sign

Fig. 4 Win-loss-tie comparison between individual reference methods and LNE, with the number of datasets on which LNE achieved statistically significantly better performance denoted with green, statistically significantly worse with red, no no statistically significant differences with yellow (Color figure online)

4.4 Using proxy estimator during optimization

So far, we could establish that LNE tends to outperform the reference methods. However, this comes at a high computational cost, especially for classifiers based on a computationally expensive training procedure, such as MLPs. Note that this cost is high when evaluating each offspring of each iteration of the optimization algorithm. The question is whether it is possible to reduce this cost, e.g., by using a proxy classifier that will not require too much computational resources to train. Such an approach would use the proxy classifier only in the optimization stage. However, the final model would be trained using the optimized parameters and the target classification model. In summary, the question, in this case, is whether the results would remain competitive compared to the reference methods.

To test this hypothesis, we conducted an experiment using the CART decision tree as the proxy classifier (i.e., CART was used during the optimization procedure). We used the obtained encoding to fit the final MLP model. We chose CART based on its training time,

but in principle, any other classification algorithm could be used. We presented the results in Table 5. As can be seen, when compared to the original results in Table 4, the performance degraded for all of the performance metrics except F-beta: for BAC and G-mean LNE still achieved statistically better results than some of the reference approaches, but their number was significantly lower. On the other hand, for the AUC using proxy estimator actually produced statistically significantly worse results than some of the reference methods. No noticeable changes were observed for F-beta. Overall, this leads to a conclusion that while the behavior was dependent on the choice of the performance metric, in most cases, using proxy estimators tends to decrease the performance of LNE, even though it reduces the training time.

4.5 Analysis of the obtained encodings

Finally, having established the general usability of the proposed approach, we can proceed with the question of whether the solutions found by LNE provide any generalizable insights into imbalanced data resampling. We performed a meta-analysis of 9600 solution vectors obtained during the conducted experiments (for 60 different datasets, 10 cross-validation folds per dataset, four classification algorithms, and four optimization criteria).

We began with the analysis of global resampling properties, that is, trends regarding the level of balancing obtained during the resampling process and the preference towards resampling due to either over- or undersampling. We present the distributions of IR values on datasets after resampling in Fig. 5, with IR $= 1$ corresponding to the case in which the resulting resampled dataset was perfectly balanced and IR < 1 corresponding to the case in which the old minority case became the new majority case. Analogically, we show the distributions of the adjusted oversampling-to-undersampling (O/U) ratios, defined as O/U ratio $= \frac{\#oversampled+1}{\#undersampled+1}$ (with denominator incremented by 1 to avoid division by zero errors, and nominator to preserve the property of O/U ratio being equal to 1 when the amount of oversampled and undersampled observations was the same) in Fig. 6.

Several observations can be made based on the presented results. Firstly, the choice of the optimization criterion had a visible influence on the overall trends, more so than the choice of the classification algorithm. In the case of IR after resampling, for BAC and G-mean, the output datasets after resampling were, on average, roughly balanced; for F-beta overbalancing occurred, meaning that the old minority class became the new majority; and the opposite was the case for AUC, for which the resampling was least intense. A similar observation can be made for the adjusted O/U ratio. While for all of the classification algorithm and performance metric combinations, the average O/U ratio was greater than one, indicating that the preferred mode of operation was complementing strong oversampling with weak undersampling, the oversampling-to-undersampling ratio was the highest for AUC and the lowest for F-beta. However, it should be noted that we observed a high variance for both IR after resampling and O/U ratio, indicating the optimal choice is highly dataset specific. Still, on average, the results suggest that supplementing oversampling with weaker undersampling can be beneficial. However, exact resampling strength and the ratio of oversampling-to-undersampling should be selected individually for a given dataset.

It is also worth considering whether the LNE's preference for setting a target IR is due to the characteristics of the measures used. In Brzeziński et al. (2020), the distributions of selected metrics are presented concerning IR. Based on the analysis presented, it appears that g-mean and BAC retain their shape (i.e., the distribution of possible values)

Table 5 Average ranks of the evaluated methods calculated for all of the considered datasets in the case of the proxy estimator, with CART used in the optimization process and MLP used as a final classifier

	Metric	SMOTE	pf-SMOTE	Lee	SMOBD	G-SMOTE	LVQ-SMOTE	A-SMOTE	SMOTE-TL	LNE
MLP	AUC	4.56	5.99	**4.20** $_{-}$	4.67	5.57	5.52	4.30 $_{-}$	4.30 $_{-}$	5.90
	BAC	4.71	6.81 $_{+}$	4.53	4.96	5.80 $_{+}$	4.37	5.18	4.67	**3.98**
	G-mean	4.75	6.98 $_{+}$	4.20	4.92	6.06 $_{+}$	**4.02**	5.33	4.50	4.23
	F-beta	4.98 $_{+}$	7.69 $_{+}$	4.96 $_{+}$	5.31 $_{+}$	6.54 $_{+}$	3.72 $_{+}$	5.39 $_{+}$	4.89 $_{+}$	**1.52**

The best performance was denoted with bold font. Methods compared to which LNE achieved statistically significantly better results (according to Shaffer's posthoc test) were denoted with a + sign, and worse results with − sign

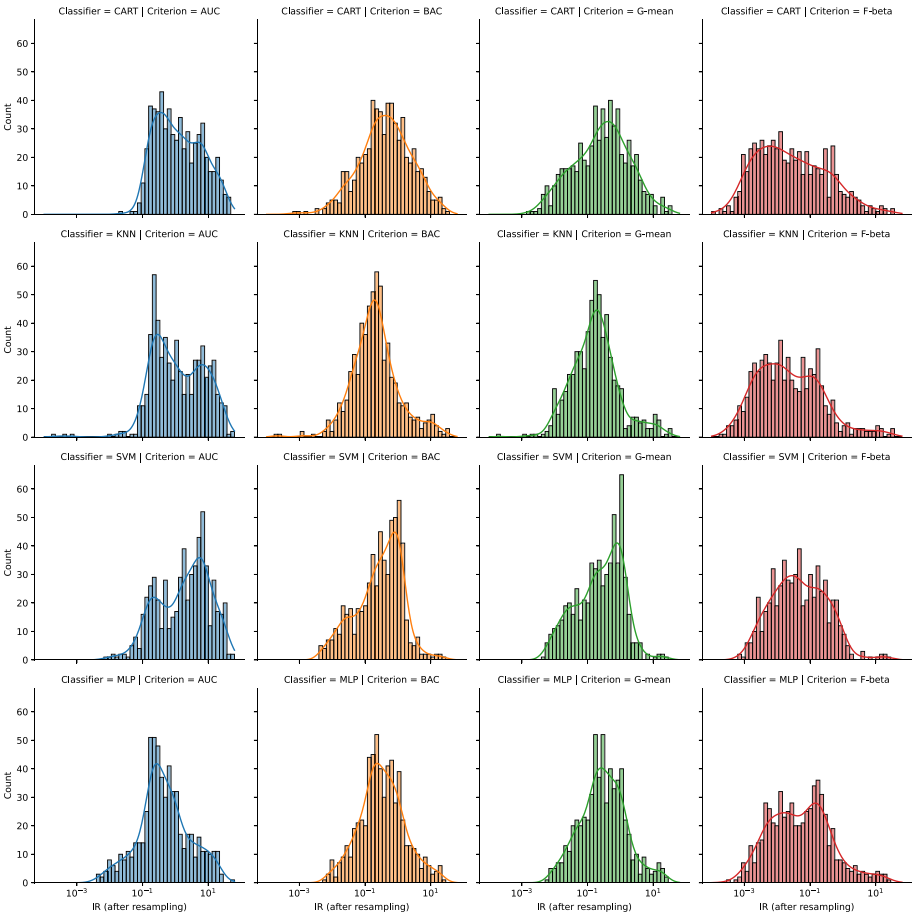

Fig. 5 Distributions of IR values after resampling. Note log scale

regardless of IR, which may be the reason that their use as an optimization criterion will lead to reasonably balanced class distributions. In the case of the metric If F-beta, where the value of *recall* was chosen proportionally to IR, one might suspect that since this metric indicates that *recall* is β times more important than *precision*, it seems natural that the model will force the overrepresentation of minority class instances in the training set.

However, it should also be recalled that the results obtained are characterized by a large value of standard deviation, so it seems only fair to conclude that the target IR value should be chosen individually for each task, taking into account the preferred metric, as well as the classification model used.

The second question that we tried to address was whether some simple dataset characteristics could be used to predict the preferred resampling properties *a priori* without needing explicit evaluation on the target dataset. To this end we examined the relationship between the IR before and after the resampling (with Pearson correlation coefficients and *p*-values presented in Table 6, and scatterplots of the two variables in Fig. 7), as well as

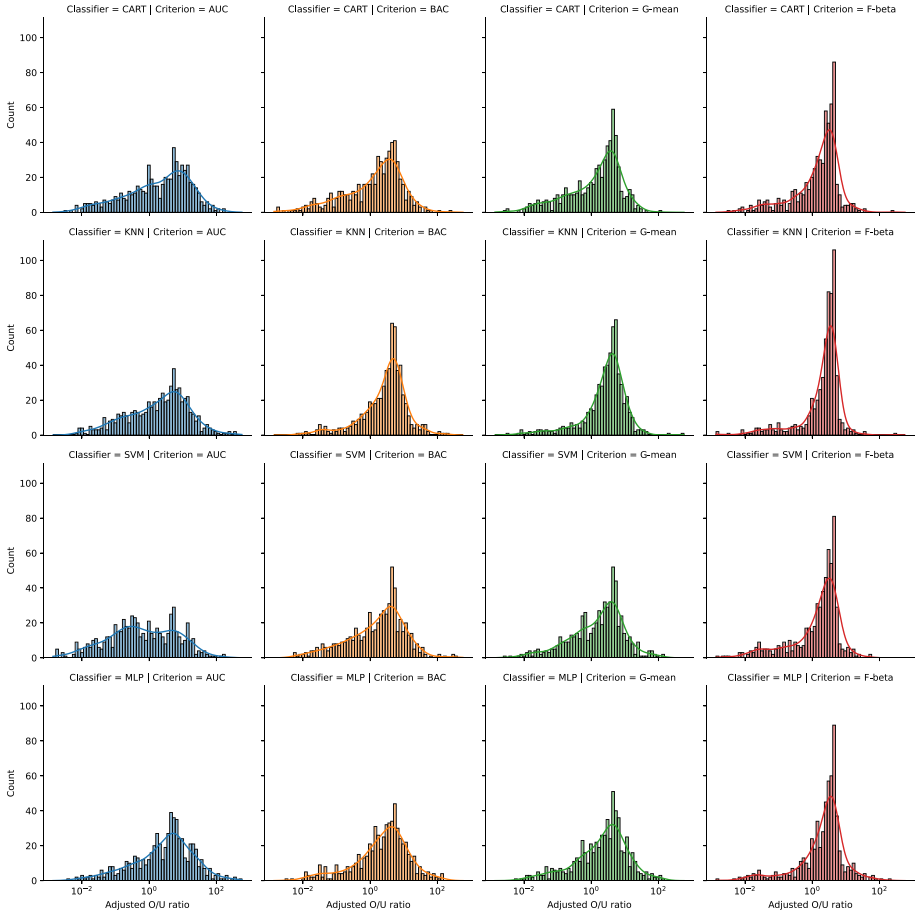

Fig. 6 Distributions of adjusted O/U ratios. Note log scale

the number of observations and the IR after resampling (with correlation coefficients in Table 7, and scatterplots in Fig. 8).

As can be seen, for either BAC, G-mean or F-beta used as the optimization criteria, we observed weak-to-medium level correlations, statistically significant in every case; less clear trends were observed in the case of AUC. We can conclude that while some dataset characteristics can be used as a predictor of the resampling parameters, they are not correlated strongly enough to be used instead of a traditional parameter selection. Still, some trends are visible: for instance, both the dataset size and the original IR tend to be negatively correlated with the resulting IR, meaning that larger and/or more imbalanced datasets tend to be resampled with lower strength. Note that we considered additional input (such as the number of features) and output (such as the resulting O/U ratio) variables, but observed either relations weaker than in the case of IR before and after resampling, or none at all. We did not include them all in the paper for brevity, but they were provided together with the algorithm's implementation.

Finally, in the last stage of the conducted analysis, we proceeded with the question of what observation types tend to be favored during the resampling. We began by evaluating

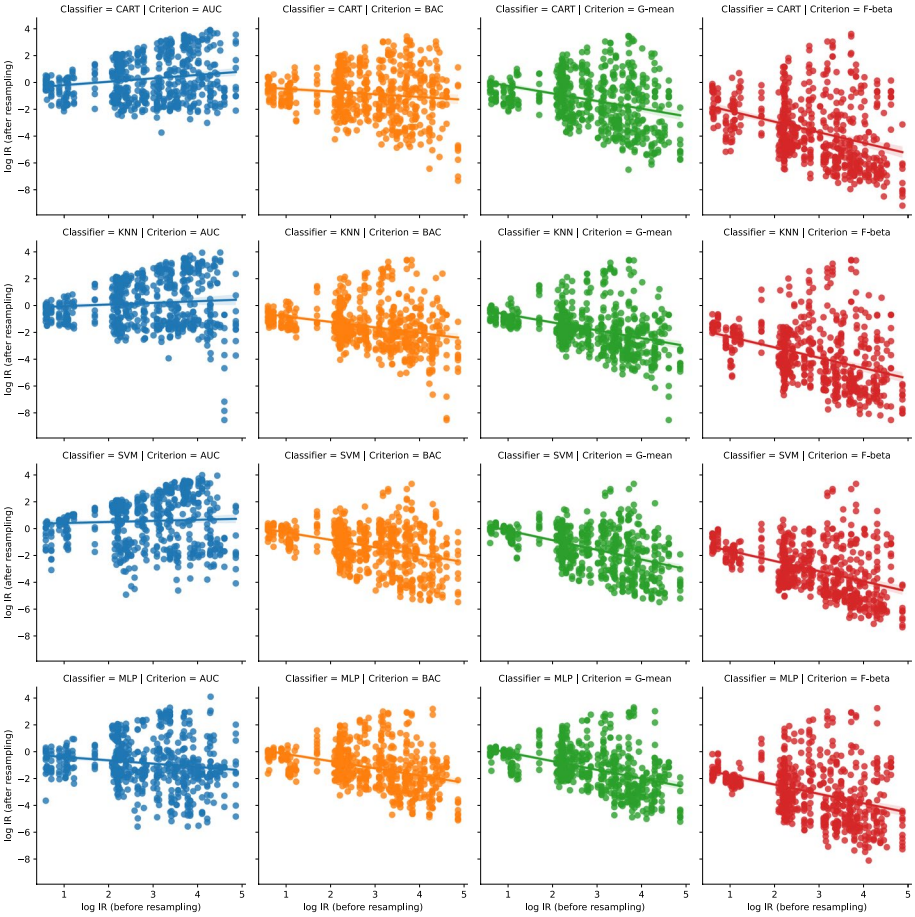

Fig. 7 Scatterplots of log IR values before and after resampling, with regression lines fitted

Table 6 Pearson correlation coefficients and *p*-values between log IR before and after resampling

	AUC	BAC	G-mean	F-beta
CART	0.17 ($p = 0.0000$)	−0.12 ($p = 0.0026$)	−0.32 ($p = 0.0000$)	−0.35 ($p = 0.0000$)
KNN	0.07 ($p = 0.0729$)	−0.27 ($p = 0.0000$)	−0.38 ($p = 0.0000$)	−0.37 ($p = 0.0000$)
SVM	0.05 ($p = 0.2372$)	−0.37 ($p = 0.0000$)	−0.47 ($p = 0.0000$)	−0.43 ($p = 0.0000$)
MLP	−0.15 ($p = 0.0003$)	−0.37 ($p = 0.0000$)	−0.43 ($p = 0.0000$)	−0.35 ($p = 0.0000$)

the average proportion of observations belonging to different observation types across all datasets and cross-validation folds (with the proportion calculated on the training partition). The results were presented in Table 8. As can be seen, in the case of the minority class all observation types were represented, on average, in a roughly similar proportion (with the highest proportion assigned to observations with no same class neighbors, i.e., the more difficult examples). This was not the case for the majority class, for which

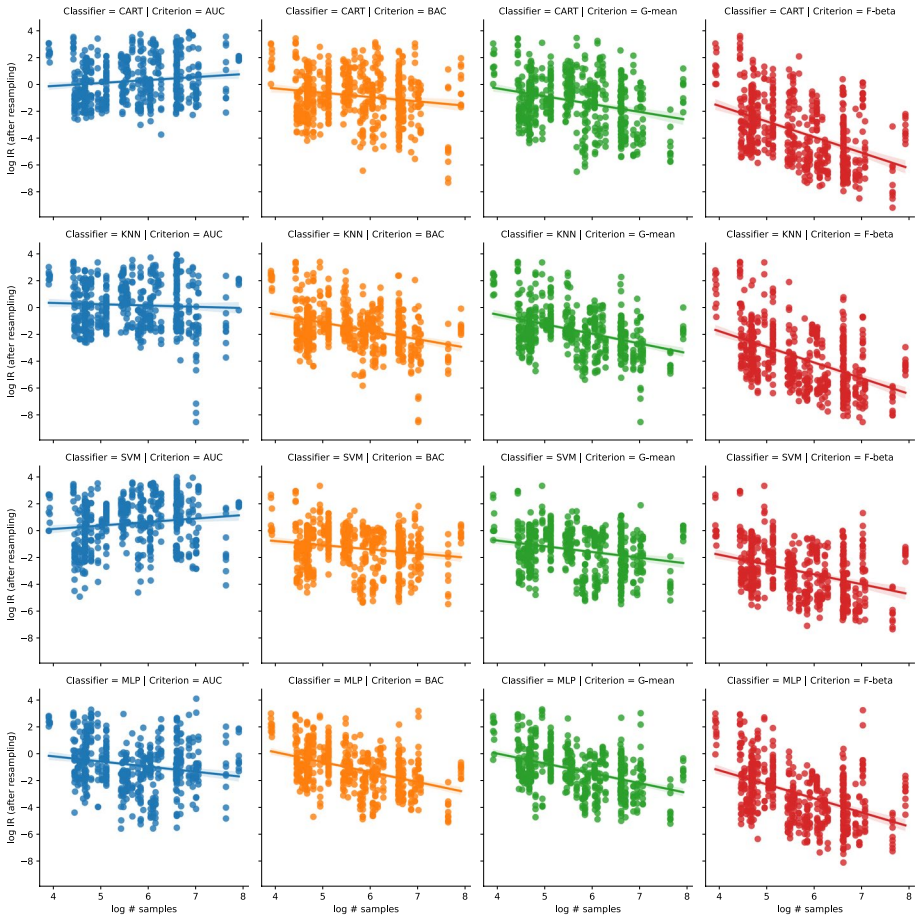

Fig. 8 Scatterplots of log # of samples and log IR after resampling, with regression lines fitted

Table 7 Pearson correlation coefficients and p-values between log # of samples and log IR after resampling

	AUC	BAC	G-mean	F-beta
CART	0.13 ($p = 0.0017$)	−0.17 ($p = 0.0000$)	−0.28 ($p = 0.0000$)	−0.44 ($p = 0.0000$)
KNN	−0.05 ($p = 0.2310$)	−0.35 ($p = 0.0000$)	−0.40 ($p = 0.0000$)	−0.48 ($p = 0.0000$)
SVM	0.13 ($p = 0.0010$)	−0.18 ($p = 0.0000$)	−0.24 ($p = 0.0000$)	−0.36 ($p = 0.0000$)
MLP	−0.20 ($p = 0.0000$)	−0.44 ($p = 0.0000$)	−0.43 ($p = 0.0000$)	−0.46 ($p = 0.0000$)

instances with all neighbors belonging to the same class heavily dominated. We expanded on this by calculating the percentage of datasets for which at least a single observation from a given type was present, with the results presented in Table 9. As can be seen, some observation types were sparsely represented in the considered datasets: the main takeaway of this observation is that this might affect the results of the further analysis and need to be taken into account.

Table 8 Mean and standard deviations of the proportion of observations with k nearest neighbors belonging to the same class

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
Minority	0.370 ± 0.289	0.234 ± 0.190	0.156 ± 0.163	0.122 ± 0.144	0.118 ± 0.185
Majority	0.003 ± 0.008	0.011 ± 0.025	0.028 ± 0.047	0.090 ± 0.085	0.868 ± 0.148

Table 9 Percentage of datasets with at least a single observation for which k nearest neighbors belong to the same class

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
Minority	94.7%	86.3%	69.8%	53.5%	43.3%
Majority	25.2%	41.5%	74.8%	95.7%	100.0%

Next, we calculated the average proportion of observations of different types created or discarded due to either oversampling or undersampling via encodings generated by LNE. To calculate this average proportion, we only considered the datasets for which at least a single observation from said type was present (so that the datasets without specific observation types do not bias the results). Additionally, as previously mentioned, some observation types were underrepresented, so we normalized the datasets for each dataset in one of two ways. First of all, by the number of observations from a given (minority or majority) class: in the case of oversampling this value was equal to $\frac{\#oversampled}{\#minority}$, whereas for the undersampling, it was $\frac{\#undersampled}{\#majority}$. This normalization was introduced to standardize the results across the datasets with different numbers of minority and majority class observations. Secondly, by the number of observations from the specific type present in the dataset. This type of normalization took into account the fact that some observation types were underrepresented.

The results were presented in Table 10. First of all it should be noted that the variance of the results was fairly high in every considered case, making the observed results reliable only to an extent. However, some trends can be observed: in the case of oversampling, when normalized by the number of minority observations, there was a monotonic trend, with the proportion of oversampling around the observations with no same class neighbors being the highest. However, this is partially due to the fact that they were most represented in the original data: when normalized by the number of observations of individual types, the most preferred type was that with a single same class neighbor in the 4-NN neighborhood, closely followed by two same class neighbors, which roughly corresponds with the standard taxonomy used by various SMOTE variants of rare and borderline instances. Analogically, when normalized by the number of majority instances, undersampling seems to be heavily focused on the observations with all same class neighbors. However, when normalized by the number of observations from specific types, this trend reverses, and observations with all the same class neighbors become least preferred for undersampling. The overall trend is that while the variance is very high, indicating high per-dataset variability, an equivalent of rare and borderline minority observations tends to be favored for oversampling and safe majority observations for undersampling.

We also tried to answer the question of whether the composition of the original dataset, that is the number of observations of a given type present in the data before resampling, affects the preference towards particular types during oversampling. We focused

Table 10 Mean and standard deviations of the proportion of observations of different types (with k denoting the number of nearest neighbors from the same class) created or discarded due to either oversampling (O) or undersampling (U)

	Normalization	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
O	# of minority	25.35 ± 56.48	18.76 ± 43.72	11.69 ± 25.79	8.91 ± 19.82	8.99 ± 23.32
O	# of individual types	66.56 ± 163.78	109.39 ± 376.31	88.20 ± 258.46	54.09 ± 129.93	44.75 ± 124.06
U	# of majority	0.01 ± 0.01	0.02 ± 0.03	0.03 ± 0.04	0.06 ± 0.06	0.48 ± 0.35
U	# of individual types	0.71 ± 0.38	0.67 ± 0.39	0.68 ± 0.36	0.69 ± 0.34	0.53 ± 0.37

specifically on the oversampling because of a greater diversity of observation types in the data. We computed the average proportions, like before, normalized by the number of observations from individual types but computed separately only on datasets, for which a given observation type was present in a large proportion (≥ 0.3) in the original data. The results were presented in Table 11. Lower values of K indicate the datasets with a large proportion of less certain minority class observations, i.e., difficult datasets. While, once again the variance of the results was high, we can observe a general trend of the more difficult datasets ($K \in \{0, 1\}$) producing encodings focusing resampling on observations other than outliers ($k \neq 0$), and less difficult datasets ($K \in \{2, 3, 4\}$) focusing on outliers ($k = 0$). This seems to indicate that with high baseline certainty resampling tends to focus on unsafe observations (with the hypothesis being that certain regions are represented well enough, and the borderline regions can be the focus of boosting), and with low baseline certainty on safe observations (since the high confidence regions of predictions have yet to be established).

Finally, having introduced the categorization into datasets consisting of a large proportion of original observations from a given type, we also examined if there are any visible differences between the different types of datasets concerning the global properties (O/U ratio and IR after resampling). The table containing this comparison was presented in Table 12. Similar to the results of the previous analysis, the split between more ($K \in \{0, 1\}$) and less ($K \in \{2, 3, 4\}$) difficult datasets was visible here as well: specifically, more difficult datasets tended to favor using more undersampling than oversampling, and resample to a lesser degree than the less difficult datasets.

5 Conclusions

This paper proposed Local Neighborhood Encoding, a novel technique for resampling imbalanced data, combining oversampling and undersampling in an evolutionary algorithm-based procedure that optimizes the proportion of resampling performed around different types of observations. The conducted experimental study showed that LNE significantly outperforms standard resampling algorithms. In addition, the conducted ablation study showed that dynamic selection of resampling strength is the main factor in good LNE performance. Conducted experiments using proxy estimators, a strategy that involves using a less computationally intensive classifier in the coding optimization process, demonstrated

Table 11 Mean and standard deviations of the proportion of observations of different types (with k denoting the number of nearest neighbors from the same class) created due to oversampling, normalized by the number of observations from individual types, computed only on the datasets for which the proportion of minority observations from type $K \geq 0.3$

	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
$K = 0$	57.40 ± 99.07	179.13 ± 519.19	183.53 ± 398.34	98.50 ± 173.08	112.96 ± 235.01
$K = 1$	55.84 ± 123.12	41.42 ± 83.70	123.15 ± 291.31	68.79 ± 154.39	75.62 ± 149.20
$K = 2$	98.20 ± 170.28	60.88 ± 145.44	32.87 ± 68.73	62.23 ± 124.01	68.36 ± 127.79
$K = 3$	94.51 ± 262.36	45.41 ± 74.86	37.51 ± 77.41	27.82 ± 65.25	34.36 ± 70.35
$K = 4$	97.41 ± 272.07	78.15 ± 192.49	75.20 ± 199.22	63.31 ± 160.22	29.69 ± 62.61

Table 12 Mean and standard deviations of the adjusted O/U ratio and IR after resampling, computed only on the datasets for which the proportion of minority observations from type $K \geq 0.3$

	$K = 0$	$K = 1$	$K = 2$	$K = 3$	$K = 4$
Adjusted O/U ratio	5.59 ± 21.19	5.97 ± 22.64	7.88 ± 23.11	8.21 ± 18.92	7.55 ± 15.74
IR (after resampling)	1.97 ± 5.31	2.33 ± 5.70	1.26 ± 3.90	1.45 ± 2.93	1.36 ± 3.04

that in some cases, especially when using performance metrics such as F-beta, it is possible to preserve the original performance of the LNE while reducing training time.

Finally, utilizing the interpretability of the encodings, we conducted a meta-analysis of the solutions produced by LNE on a large set of benchmark datasets. While there was a significant variance in the obtained results, suggesting that dataset-specific tuning is still required, some common trends have been observed:

- A combination of oversampling and undersampling was the preferred strategy, with strong oversampling combined with reasonable weak undersampling.
- The optimal strength of resampling was strongly dependent on the performance chosen metric, BAC and G-mean prefer approximately balanced distributions, while metrics such as F-beta favored overbalancing.
- The general characteristics of the datasets before resampling can be, to some extent, used to predict the properties of the resampling (such as the oversampling-to-undersampling ratio or resampling strength); however, specific tuning of these parameters is still required to achieve the optimal performance.
- Produced solutions, on average, tended to prioritize rare and borderline observations, during oversampling, and unsafe examples during undersampling. Similar observations were made in the work mentioned earlier on selective oversampling, where the most common object fractions for oversampling were borderline and rare (Sáez et al., 2016).
- However, when taking into the account the original distribution of the observation types, for more difficult datasets there was a tendency to produce encodings focusing oversampling on observations other than outliers, and less difficult datasets focusing on outliers.

LNE is an efficient oversampling strategy that can be used when the dataset size is relatively small and/or computational resources are not limited. Its interpretability can also be

used to gain insight into existing resampling strategies, as it allows the removal of errors introduced by methods such as Borderline-SMOTE and Safe-Level-SMOTE, the search for which becomes part of the optimization process. Possible future research directions include scaling the approach to larger dataset sizes and exploring the idea of using proxy estimators in more depth. The use of proxy classifiers in place of dataset difficulty scores should also be considered. However, this would require confirmation of the hypothesis that simplifying data distributions in preprocessing positively affects the quality of final classification models.

Acknowledgements This work was supported by the Polish National Science Centre under the grant No. 2019/35/B/ST6/04442 as well as the PLGrid Infrastructure.

Author contribution M.K. conceived of the presented idea, implemented the algorithm, planned and carried out the experiments, and analysed the results. M.W. contributed to the analysis of the results. Both authors discussed the results, drew the conclusions, and contributed to the final manuscript, helping with writing, reviewing and editing.

Funding This work was supported by the Polish National Science Centre under the grant No. 2019/35/B/ST6/04442.

Data availability All data used in this work is publicly available as reported in Sect. 4.1.

Code availability All code used in this work is publicly available as reported in Sect. 4.1.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

References

Alcalá-Fdez, J., Fernández, 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 & Soft Computing, 17*, 255–287.

Alpaydin, E. (1999). Combined 5 × 2 cv F test for comparing supervised classification learning algorithms. *Neural Computation, 11*(8), 1885–1892.

Barandela, R., Hernández, J. K., Sánchez, J. S., & Ferri, F. J. (2005). Imbalanced training set reduction and feature selection through genetic optimization. In *CCIA* (pp. 215–222).

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. *ACM SIGKDD Explorations Newsletter, 6*(1), 20–29.

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys, 49*(2), 1–50.

Brzezinski, D., Stefanowski, J., Susmaga, R., & Szczęch, I. (2018). Visual-based analysis of classification measures and their properties for class imbalanced problems. *Information Sciences, 462*, 242–261.

Brzeziński, D., Stefanowski, J., Susmaga, R., & Szczęch, I. (2020). On the dynamics of classification measures for imbalanced and streaming data. *IEEE Transactions on Neural Networks and Learning Systems, 31*(8), 2868–2878.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 475–482). Springer.

Cao, Q., Wang, S. Z. (2011). Applying over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning. In *2011 International conference on information management, innovation management and industrial engineering* (vol. 2, pp. 543–548). IEEE.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Philip Kegelmeyer, W. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets*. Springer.

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research, 61*(1), 863–905.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(4), 463–484.

García, S., & Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation, 17*(3), 275–306.

Gazzah, S., Amara, N.E.B. (2008). New oversampling approaches based on polynomial fitting for imbalanced data sets. In *2008 The 8th IAPR international workshop on document analysis systems* (pp. 677–684). IEEE.

Han, H., Wang, W. -Y., & Mao, B. -H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878–887). Springer

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the international joint conference on neural networks, 2008, part of the IEEE world congress on computational intelligence, 2008, Hong Kong, China, June 1-6, 2008* (pp. 1322–1328).

Hualong, Yu., Ni, J., & Zhao, J. (2013). ACOSampling: An ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data. *Neurocomputing, 101*, 309–318.

Johnson, J., & Khoshgoftaar, T. (2019). Survey on deep learning with class imbalance. *Journal of Big Data, 6*, 27.

Karia, V., Zhang, W., Naeim, A., & Ramezani, R. (2019). GenSample: A genetic algorithm for oversampling in imbalanced datasets.

Khoshgoftaar, T. M., Seiffert, C., Hulse, J. V., Napolitano, A., & Folleco, A. (2007). Learning with limited minority class data. In *6th International conference on machine learning and applications (ICMLA 2007)* (pp. 348–353).

Khoshgoftaar, T. M., Seliya, N., & Drown, D. J. (2010). Evolutionary data analysis for the class imbalance problem. *Intelligent Data Analysis, 14*(1), 69–88.

Kim, H.-J., Jo, N.-O., & Shin, K.-S. (2016). Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction. *Expert Systems with Applications, 59*, 226–234.

Kim, Y., Lee, Y., & Jeon, M. (2021). Imbalanced image classification with complement cross entropy. *Pattern Recognition Letters, 151*, 33–40.

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing, 83*, 105662.

Kovács, G. (2019). smote-variants: A Python implementation of 85 minority oversampling techniques. *Neurocomputing, 366*, 352–354.

Koziarski, M. (2021). CSMOUTE: Combined synthetic oversampling and undersampling technique for imbalanced data classification. In *2021 International joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Koziarski, M., Krawczyk, B., & Woźniak, M. (2017). Radial-based approach to imbalanced data oversampling. In *International conference on hybrid artificial intelligence systems* (pp. 318–327). Springer.

Koziarski, M. (2020). Radial-based undersampling for imbalanced data classification. *Pattern Recognition, 102*, 107262.

Koziarski, M. (2021). Potential Anchoring for imbalanced data classification. *Pattern Recognition, 120*, 108114.

Koziarski, M., & Woźniak, M. (2017). CCR: Combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer Science, 27*(4), 727–736.

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence, 5*, 04.

Lee, J., Kim, N., Lee, J. -H. (2015). An over-sampling technique with rejection for imbalanced class learning. In *Proceedings of the 9th international conference on ubiquitous information management and communication* (pp. 1–6).

Lee, S. S. (2000). Noisy replication in skewed binary classification. *Computational Statistics & Data Analysis, 34*(2), 165–191.

Li, M., Xiong, A., Wang, L., Deng, S., & Ye, J. (2020). ACO resampling: Enhancing the performance of oversampling methods for class imbalance classification. *Knowledge-Based Systems, 196*, 105818.

Maciejewski, T., & Stefanowski, J. (2011). 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, April 11-15, 2011, Paris, France* (pp. 104–111).

Nakamura, M., Kajiwara, Y., Otsuka, A., & Kimura, H. (2013). LVQ-SMOTE-learning vector quantization based synthetic minority over-sampling technique for biomedical data. *Biodata Mining, 6*(1), 16.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems, 46*(3), 563–597.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(Oct), 2825–2830.

Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution: A practical approach to global optimization*. Springer Science & Business Media.

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint*arXiv:1811.12808

Sáez, J. A., Krawczyk, B., & Woźniak, M. (2016). Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition, 57*, 164–178.

Sandhan, T., Choi, J. Y. (2014). Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition. In *2014 22nd international conference on pattern recognition* (pp. 1449–1453). IEEE.

Stapor, K., Ksieniewicz, P., García, S., & Woźniak, M. (2021). How to design the fair experimental classifier evaluation. *Applied Soft Computing, 104*, 107219.

Węgier, W., Koziarski, M., & Woźniak, M. (2023). Optimized hybrid imbalanced data sampling for decision tree training. In *Proceedings of the companion conference on genetic and evolutionary computation* (pp. 339–342).

Węgier, W., Koziarski, M., & Woźniak, M. (2022). Multicriteria classifier ensemble learning for imbalanced data. *IEEE Access, 10*, 16807–16818.

Weiss, G. M., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research, 19*(1), 315–354.

Wojciechowski, S. (2021). Multi-objective evolutionary undersampling algorithm for imbalanced data classification. In *Computational science–ICCS 2021: 21st international conference, Krakow, Poland, June 16-18, proceedings, part III* (pp. 118–127). Berlin, Heidelberg: Springer-Verlag.

Zhou, B., Yang, C., Guo, H., Hu, J. (2013). A quasi-linear SVM combined with assembled SMOTE for imbalanced data classification. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1–7). IEEE.

Authors and Affiliations

Michał Koziarski[1] · **Michał Woźniak**[1]

✉ Michał Koziarski
michal.koziarski@pwr.edu.pl

Michał Woźniak
michal.wozniak@pwr.edu.pl

[1] Department of Systems and Computer Networks, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50–370 Wrocław, Poland