Knowledge-Based Systems 301 (2024) 112236

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A comparative study on noise filtering of imbalanced data sets

Szilvia Szeghalmy, Attila Fazekas *

Faculty of Informatics, University of Debrecen, H-4002 Debrecen P.O Box 400, Hungary

ARTICLE INFO

Dataset link: Supplementary Materials (Original data)

Keywords:
Noise removal
Oversampling
Imbalanced learning
kNN

ABSTRACT

Class imbalance in data sets used to train classifiers can negatively affect the performance of the resulting models. A commonly used solution to address this issue is to oversample the data sets and supplement them with synthetic samples. However, oversampling can increase the level of noise in the data sets. Several sampling methods attempt to prevent this negative effect in a variety of ways, but there are still many open questions about which solutions might be appropriate in different cases. In our study, we compared the impact of different noise filtering methods on relatively small, imbalanced synthetic data sets and then examined how noise filters perform as a preprocessing step of sampling methods following different sampling strategies. The success of the noise filtering was gauged through the performance of k-Nearest Neighbours models built on the balanced data sets. Our results highlight the importance of cleaning the minority class and provide insight into which noise filtering approaches might be useful as a first step to oversampling on imbalanced noisy data sets. Among the noise filtering methods included in our study, the GMM-based ones perform well on highly noisy imbalanced data sets. It is also worth highlighting some versions of ENN, which are very effective when the noise level is moderate.

1. Introduction

Classification algorithms are extensively utilized to address a variety of problem families [1,2]. However, many otherwise efficient methods are susceptible to noise in the data set. This is a significant concern as data gathered from real-world processes often contain various types of noise. On one hand, the values of the available data (geometrical location in the feature space) can be corrupted. On the other hand, the class labels of the samples in the training set, which form the basis of supervised learning, may be incorrect. This paper focuses on the latter type of noise, commonly known as label noise or class noise. The influence of noise on classification can be alleviated through adjustments at both the algorithmic and data levels. In the former case, classifiers are designed to be more robust to certain types of noise. For instance, special kernels and objective functions have been proposed for SVM [3–5], decision trees can be made more resistant to noise by pruning to prevent overfitting [6], and kNN can also be improved using certain distance metrics [7].

While the goal of data-level solutions is to clean the data set to prepare it for classification [8]. Several methods designed to handle label noise are based on the k-Nearest Neighbours (kNN) algorithm, which determines whether an element is considered noise based on the labels of its neighbours. Several authors have proposed using ensemble classifiers, and the effectiveness of partitioning filters has been demonstrated in various areas. (The methods included in this study are presented in Section 3.2.)

The impact of noise on classification has been extensively investigated, and numerous studies have shown that cleaning a data set from noise can enhance the properties of the model built on it. However, comprehensive studies have primarily focused on balanced cases [9] or specific domains [10], there is no consensus in the literature on how to deal with noise in small, imbalanced data sets.

To grasp the challenges associated with managing noise in an imbalanced data set, consider this straightforward example: Let us say we initially had 500 majority and 50 minority samples. However, due to errors in the labelling process affecting 2% of the samples and assuming the errors were evenly distributed, one minority and 10 majority samples would be mislabelled. Consequently, the noise level within the minority class of the resulting data set would be approximately 17%, as 10 out of (50−1+10) samples are mislabelled. It may seem obvious that we should remove the noise from the minority class, but at this noise level, it is feared that valuable minority samples will be lost during the process (Fig. 1).

Imbalances, especially when classes overlap, can have a negative impact on the performance of models built on a data set even if it does not contain noise [12]. Therefore, the primary goal for such data sets is to balance the size of the classes, for which sampling is a typical

* Corresponding author.
E-mail addresses: szeghalmy.szilvia@inf.unideb.hu (S. Szeghalmy), fazekas.attila@inf.unideb.hu (A. Fazekas).

https://doi.org/10.1016/j.knosys.2024.112236
Received 12 November 2023; Received in revised form 18 June 2024; Accepted 7 July 2024
Available online 24 July 2024

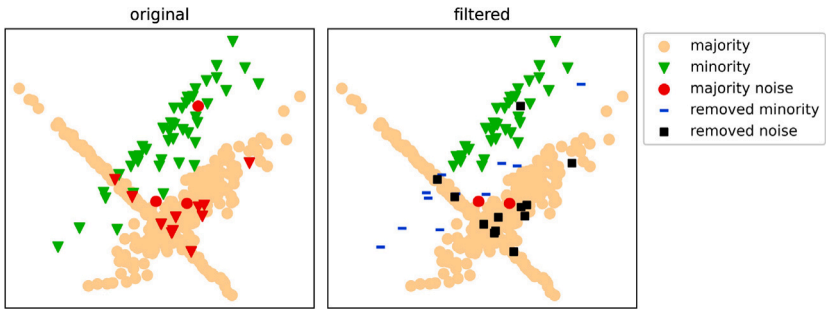

Fig. 1. Illustrating the potential risks of noise filtering on an imbalance data set. The original data set is shown on the left, and the version after applying Iterative-Partitioning Filter with kTLNN classifier [11] is shown on the right. Several correct minority samples have been lost.

solution [13]. Dealing with noise can be part of this process [14–17], and it does not necessarily require a direct noise filtering step. If the correct samples are oversampled, the effect of the incorrect samples on the classification model will be reduced even without removing them from the data set.

In his empirical study, Kovács ranked 86 samplers based on their impact on classification. The top performers included methods with and without a noise filtering step [13]. However, the study was conducted on data sets where the noise level of most data sets is unknown, so the question remains open as to whether it is worth building a noise filter into the sampling process for data preparation.

The aim of our study is to address this issue by investigating the effect of different noise filtering algorithms on small, imbalanced data sets loaded with label noise and by examining how different sampling solutions with and without noise filtering steps affect the kNN classification. Our main motivation was to find out whether it is worth starting the sampling with a noise filtering step, and if so, what methods are worth considering.

Here we would like to draw attention to the fact that large noisy data sets require other solutions [18,19]. Such data sets are typically processed by deep neural networks, at many points of which noise can be counteracted. We can consider the use of various regularization techniques [20], loss functions [21], adaptive layers [22] and even subnetworks [23]. As this topic is not covered in our paper, for interested readers, we recommend Song et al.'s survey [24], which provides a comprehensive review of 62 state-of-the-art deep learning methods.

The main contributions of this work are listed below:

- Our study involved conducting experiments to assess the efficacy of noise filters, both independently and as components of sampling methods, across small imbalanced data sets, noise-free data sets, and noisy data sets. We aimed to understand the influence of noise filters on classification by integrating them into samplers based on different operational principles.
- We explored the correlation between the effectiveness of the noise filtering step and the performance of kNN classification on data sets prepared using a sampler that includes a noise filtering step.
- We delved into how various noise filters could be paired with steps commonly used in oversamplers.
- Our experiments show that Gaussian-based noise filtering combined with a wide range of samplers has a positive effect on classification.

The remainder of the paper is organized as follows. The next section elucidates some fundamental concepts and provides an overview of how various sampling methods attempt to address noise. Section 3 includes descriptions of the noise filters, samplers, and data sets involved in this study.

Section 4 discusses the experimental designs and the analysis of the data from the experiment. The results, along with observations based on these results, are also presented in this section. Finally, we summarize the findings, discuss the limitations of this study, and identify issues for future exploration.

2. Related works

In a classification problem, an imbalance in the data set occurs when the sizes of the different classes are unequal. This imbalance is quantified by a ratio. For instance, in a binary classification problem, an imbalance ratio of 100:1 signifies that for every sample in one class, there are 100 samples in the other. The class with more samples is referred to as the majority or negative class, while the class with fewer samples is known as the minority or positive class.

While a slight imbalance does not pose a significant issue, a severe imbalance can complicate the construction of an effective classification model and the interpretation of its evaluation metrics.

As outlined earlier, one approach to rectify this issue involves adjusting the number of samples to balance the data set. This can be achieved through undersampling the majority class, oversampling the minority class, or employing hybrid methods.

The goal of undersampling is to eliminate samples from the majority class that do not contribute significantly to model construction. This can be achieved, for example, by removing majority samples with similar characteristics or substituting them with a single synthetic sample, such as their centre. Other objectives may include cleaning the decision boundary and eliminating noise [25–28].

The elimination of irrelevant samples from the training set is theoretically advantageous for classification [29], but undersampling carries the risk of information loss because valuable samples may also be deleted during the process [30], which explains why oversampling is more prevalent in literature [31], and why undersampling methods almost exclusively modify the majority class. Despite this, Kang's study suggests that noise filtering of the minority class can positively impact the performance of a classifier built on the training set [32].

The early oversampling methods primarily added copies of selected samples to the data set, but they did not significantly differentiate these samples by weighting them. In contrast, SMOTE [33] employs a completely different approach. Its core concept involves generating new synthetic samples for the minority class along the lines connecting randomly chosen minority samples (seeds) and their randomly selected nearest minority neighbours (co-seeds or pairs). It is clear, that SMOTE is not suitable for noisy data sets because it selects all minority samples, including any potential noise in the data set, as seed points with equal probability. This could lead to noise propagation, as illustrated in Fig. 2.

Numerous authors have suggested hybrid methods to mitigate the noise propagation characteristics of SMOTE. These methods involve oversampling the minority class and then filtering the entire data set. This approach allows noisy samples to contribute to the generation of new samples, but subsequently, samples that are deemed noisy are permanently removed from the oversampled data set. For instance, Batista et al. advocated for the elimination of Tomek Links and the use of ENN. In their experiments, both methods demonstrated effective performance on data sets with a small number (less than 100) of minority samples [34]. The combination of SMOTE and ENN also appears in Hasan et al.'s oversampler [35]. In their work, noise filtering plays

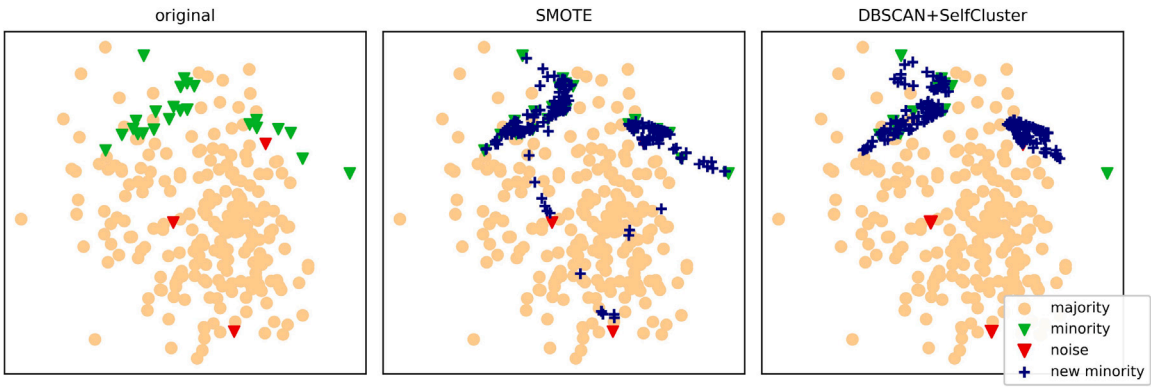

Fig. 2. The effect of SMOTE and the effect of selecting seeds and pairs randomly from the same clusters. In the latter case, the noises forming a single-element clusters could not propagate.

a prominent role, because it prepares the data set for a second round of oversampling, where a synthetic sample can be created between any two samples of the same class.

Gu et al. incorporated the Neighbourhood Cleaning Rule [36] into their hybrid sampler [37], while Sáez et al. and Sun et al. proposed the use of an iterative partition filter [11,38].

There are only a handful of methods that strive to minimize noise in the minority class prior to sampling. One such method is SMOTE-LOF, which eliminates outliers from the minority class based on the Local Outlier Factor [39]. Another example is PDR-SMOTE [40], which uses kNN to determine the noise to be deleted. We can also mention EFN-SMOTE [41], which starts with a clustering step and performs a 1NN-based noise filtering for each cluster separately, deleting not only noisy samples but also borderline ones.

As previously stated, oversampling is the most prevalent approach. These methods do not permanently remove samples from the data set. However, post-filtering may discard new samples that are generated in inappropriate locations, typically surrounded by majority samples [42–44]. A noise filtering step can also occur before the creation of new samples, aiming to limit the selection of minority set for generation to those deemed useful.

Among the methods that focus on the appropriate selection of the samples involved in the generation of the synthetic ones, many use a clustering step which allows isolated samples to be detected and treated differently from other ones. Those samples are usually considered noise, so such samples are less likely to be selected as seeds and co-seeds (pairs) [15,45]. However, we can also find methods that generate more synthetic samples into the small clusters to balance not only the majority and minority classes, but also the clusters within the smaller class [46]. Classification of noisy and non-noisy samples based on kNN is also a common solution [47,48].

If the selection of seeds and pairs is restricted to the same cluster, the spatial expansion of noise can also be mitigated (see Fig. 2) [17,49].

Naturally, numerous samplers integrate the aforementioned techniques. For instance, some separate noise and non-noise samples in the minority set through classification, oversample the non-noise set, and then perform post-filtering to eliminate samples generated in inappropriate locations [50].

This brief overview reveals a multitude of diverse solutions proposed in the literature for addressing this task, and the work of Fernández and his co-authors [16] provides insight into the frequency of different solutions in sampling methods. According to their survey, out of the 90 methods included in the study, only 18 incorporate a permanent noise filtering step, and 11 undersample the majority class, while 35 methods limit the range of samples involved in sample generation (initial selection), often achieved by non-permanent filtering of the minority set [45,51,52].

The authors reported that 25 methods generate samples adaptively, meaning that samples contribute to generating varying numbers of new samples based on their perceived usefulness for classification. For example, some methods prioritize strengthening the position of minority samples surrounded by majority ones, although this may pose risks on noisy data sets [53]. Conversely, other approaches prefer to create new samples further from the decision boundary, in a "safe" space.

We posit that this concise overview underscores the lack of consensus among scholars regarding the appropriate methodologies for handling noisy, imbalanced data sets. This area necessitates substantial further research to elucidate the functions of noise filters and explore alternative solutions. Our study primarily concentrates on evaluating persistent noise filters as preliminary steps in a sampling method.

3. Materials and methods

In this chapter, we describe in detail the data sets and noise filtering methods used in our experiments.

3.1. Data sets

While there are publicly available synthetic and non-synthetic data sets for exploring imbalanced classification problems [54,55], they were not adequate for the experiments we have designed. Consequently, we opted to create synthetic data sets with diverse properties. In these data sets, not only are the labels of the samples known, but it is also clear whether a sample is noise. This approach allows to precisely measure the impact of noise filters. Based on the initial test results, we slightly refined the procedures for the second test, for which we have generated independent data sets.

For the first test, designed to observe the effect of noise filters, we generated a total of 810 imbalanced data sets using the `scikit-learn` package [56]. Half of these data sets contained 1200 samples, while the other half contained 600 samples. Regardless of the total sample count, each data set included either 60, 100, or 140 minority samples before the addition of noise. Thus, the imbalance rate of the data sets – prior to noise addition – ranged from 3.29 to 19.00. The potential number of clusters within the classes was 1, 2, or 3, and the number of clusters in the majority and minority classes could vary independently during generation. The method from `sklearn` placed the clusters on the vertices of a random polytope. Similarly, the number of features that the data set would have, which determines the dimensionality of the space that the data could be represented in, was also a parameter. This parameter could be 4, 8, or 16. The noise amount was 0%, 4%, 8%, 16%, or 32% of the minority class's sample count. We employed symmetric label noise, meaning we altered the labels of randomly selected samples to change their class affiliation, but the samples themselves remained unmodified.

For the second test, aimed to observing the effect of samplers with noise filtering on classification, we generated noise-free data sets in a

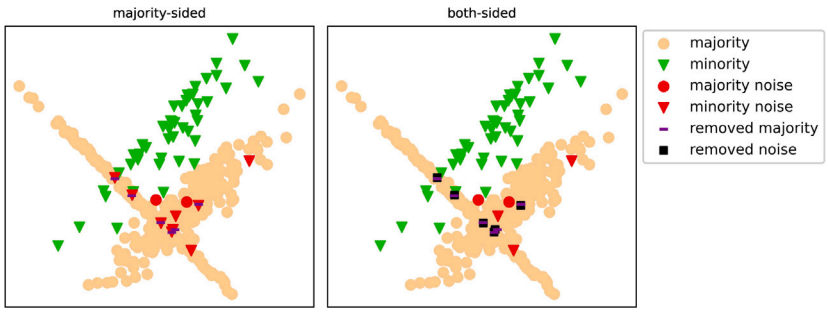

Fig. 3. The result of a TLRemoval that deletes from the majority class and a TLRemoval that deletes from both classes applied to the original data set in Fig. 1.

Table 1
The mean and standard deviation of the AugR of the data sets, and the noise measured in the boundary regions, broken down by the noise level used for generation.

Noise level	AugR	Border noise - Min (%)	Border noise - Maj (%)
0	60.09 ± 18.68	0 ± 0	0 ± 0
4	60.70 ± 18.24	11.67 ± 10.85	0.84 ± 4.07
8	61.31 ± 17.77	20.28 ± 13.29	1.70 ± 3.75
16	62.22 ± 16.91	34.57 ± 15.74	2.54 ± 4.88
32	63.74 ± 15.48	49.54 ± 15.42	4.17 ± 5.64

similar manner but with four times as many samples as before. These sets were divided into four disjunctive parts (folds), maintaining the ratio between the classes. Noise was added to one fold, while the other three folds remained in their original form for testing. This approach also resulted in the production of 810 data sets.

Table 1 provides additional information on the generated data sets, describing the average overlap between classes and the noise level of the class boundaries, because previous studies have shown that in addition to the degree of imbalance, these factors also influence the difficulty of the classification task [54,57,58].

To characterize the overlap between the two classes, we used the Augmented R-value (AugR), which is designed for imbalanced data sets [59]. The level of the noise at the class boundary was defined as the ratio of the number of borderline samples with a false class label to the total number of borderline samples, where the set of borderline samples was defined as members of the Tomek Links [25] following the study of Napierała at al. [54].

The data set collection can be found in the Supplementary Material or [60].

3.2. Filters

In this study, our primary focus is on noise filters that are either an integral part of one or more sampling algorithms or are recommended for use with them. Below, we provide a brief overview of these filters.

`Tomek Link Removal (TLRemoval)`: The fundamental concept of this method is to identify samples that are each other's nearest neighbours based on Euclidean distance but belong to different classes. These majority–minority sample pairs (Tomek Links) are considered noise and are removed. However, in the context of imbalanced data sets, it is common practice to delete only the members of the Tomek Links belonging to the majority class, which aids in the correct classification of borderline minority samples [25,34]. The difference can be observed in Fig. 3.

`One-Sided Selection (OSS)`: The authors suggest that classification success can be enhanced by removing samples suffering from class-label noise and samples near the decision border from the data set. Redundant samples should also be eliminated to reduce computational cost. To achieve these objectives, the Tomek Link removal was combined with a simplified version of a Condensed Nearest Neighbours (CNN) technique [61]. OSS first selects the minority samples of the data set and one randomly chosen majority sample. Then, using the set formed by these samples, it performs a 1-Nearest Neighbours classification on the remaining part of the data set, and expands the previously created set with the incorrectly classified samples. After that, it removes those majority samples from the set that are part of one of the Tomek Links [27].

`CNN + Tomek Link (CNNTL)`: Like OSS, this method also combines the Condensed Nearest Neighbours with Tomek Link Removal, but the order is reversed (Fig. 4). By changing the order of the steps, the authors primarily aimed to reduce the computational cost of the algorithm [34].

`Edited Nearest Neighbour (ENN)`: This method deletes a sample from the majority class if at least 2 of its 3 nearest neighbours belong to the minority class [34]. In the literature, several samplers are found whose noise filtering part is inspired by the ENN [14,35,40,47,48,62,63]. For our experiments, we used a generalized version that allows us to change the number of neighbours, the threshold for deletion, and the scope of filtering (majority class or both classes). The threshold for minor was set in the experiments to delete only isolated samples (Fig. 5).

`GMM-based NoiseFilter (GMMNF)`: Gaussian mixture models (GMM) assume that the data originates from a number of normally distributed subclusters. Thus, the centres and covariance structure of the clusters can be estimated. Based on these, the samples that more likely belong to the minority class, but are in the majority class, can be deleted and vice versa [64,65]. The effect of the GMMNF is illustrated in Fig. 6.

`Overlapping Sample Deletion (OSD)`: This noise filter, which focuses on reducing class overlap under the assumption that this can reduce the noise in the data set, was presented alongside the AKN-Random-SMOTE in [66]. The overlapping set contains those isolated minority samples that are farther from the centre of the minority class than their k nearest neighbours, and those isolated majority samples that are farther from the centre of the majority class than their k nearest neighbours. A sample is considered isolated if all of its k nearest neighbours belong to a different class than the sample itself. The result of the filtering is produced by removing the samples of the overlapping set from the original data set (Fig. 7).

`Relabeller`: This term refers to the optional relabelling step in the SPIDER2 oversampling technique [54]. Similar to ENN, it uses kNN classification to differentiate between unsafe (noise and borderline cases) and safe samples. It then alters the labels of misclassified majority samples to match the minority class label. For our experiment, we expanded the method to allow selection of the relabelling scope (either the majority class or both classes), but only modified the isolated samples of the minority class (Fig. 8).

`Iterative-Partitioning Filter (IPF)`: This iterative procedure splits the database into n equal-sized partitions, constructs a classifier (C4.5) for each, and observes the classification results for the entire data set. Samples misclassified by the majority of classifiers are deemed noise and removed. The process concludes when there are no significant changes in the data set after a few iterations. This method is

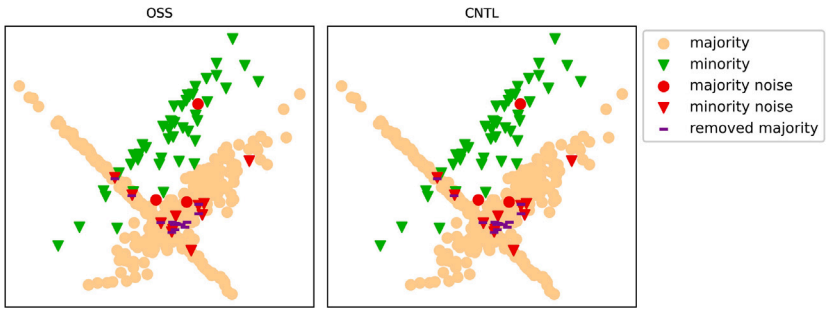

Fig. 4. The result of OSS and CNNTL filters applied to the original data set in Fig. 1.

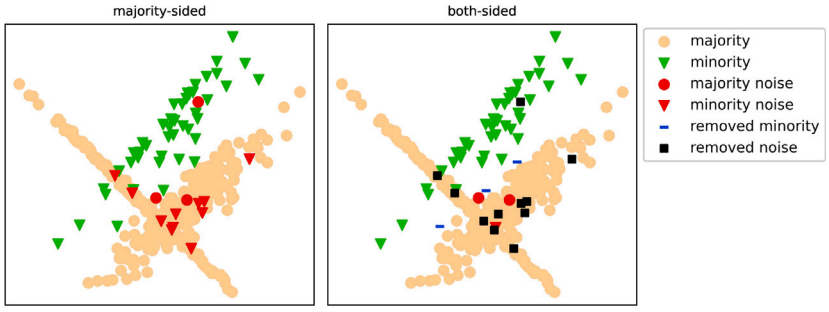

Fig. 5. The result of ENN filters applied to the majority class and on both class of the original data set in Fig. 1. The number of neighbours was set to 5. The threshold for the majority class is 0.5, and 0.9 for the minority, which means only isolated samples remove from the minority class.

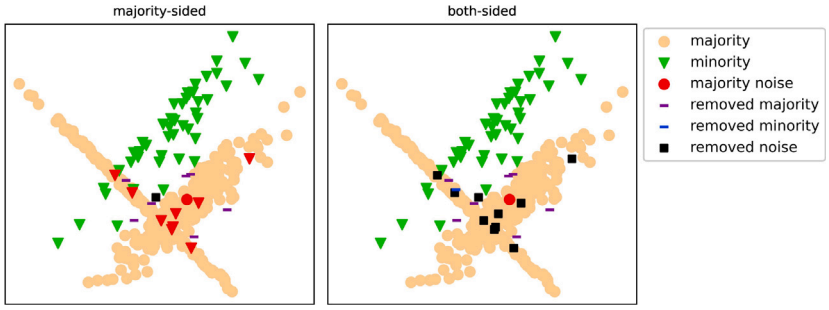

Fig. 6. The result of a GMMNF that deletes from the majority class and a GMMNF that deletes from both classes applied to the original data set in Fig. 1.

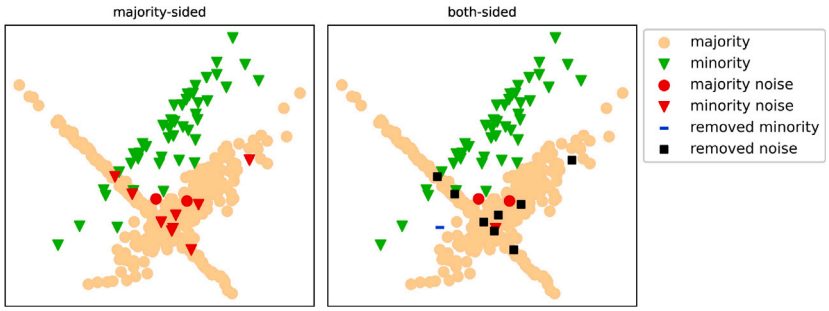

Fig. 7. The result of a OSD that deletes from the majority class and a OSD that deletes from both classes applied to the original data set in Fig. 1.

used, for instance, in SMOTE-IPF [38]. IPF can also be applied with a consensus voting scheme, which implements milder filtering, because only those samples are eliminated in the data set during the iterations, which all classifiers classified into the wrong class (Fig. 9.

`Iterative-Partitioning Filter with kTLNN (IPF_kTLNN)`: Under this name, we refer to the noise filter part of a recently published SMOTE_kTLNN method [11]. As its name suggests, this filter is also based on IPF (with majority voting shame), but the authors replaced the C4.5 classifiers to Two-Layer Nearest Neighbour classifiers (kTLNN) [67], which resulted in improvement in many cases based on their measurements on KEEL's imbalanced data sets [68] (Fig. 1).

`Noise Filter of Combined Cleaning and Resampling Algorithm (CCRNF)`: This cleaning algorithm differs fundamentally from others. Instead of relabelling or deleting the samples from the majority (and minority) class, it displaces majority samples away from the environment of minority ones. To put it vividly, CCRNF clears the environment of minority samples to make a safe place for the synthetic

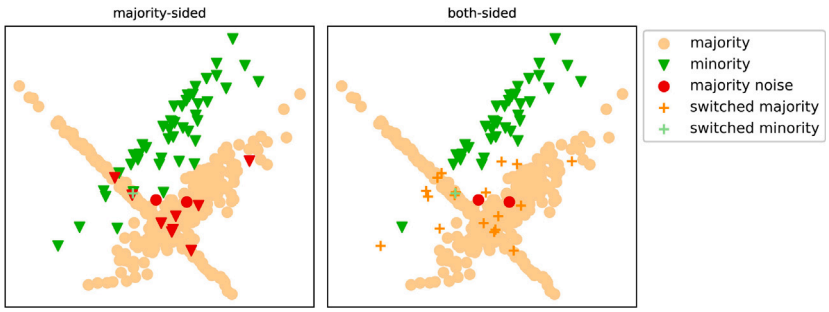

Fig. 8. The result of a Relabeller that deletes from the majority class and a Relabeller that deletes from both classes applied to the original data set in Fig. 1.

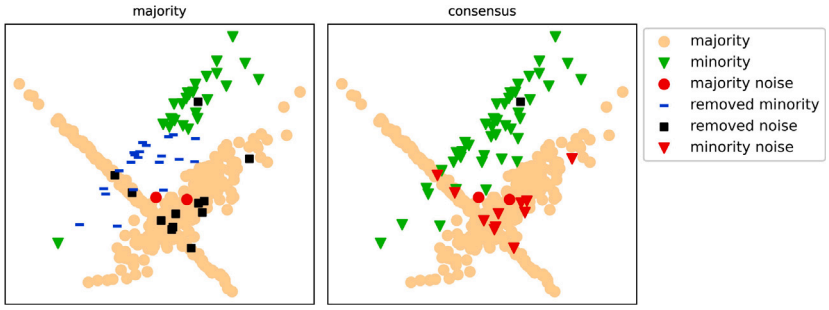

Fig. 9. The result of IPF with majority and consensus voting shame applied to the original data set in Fig. 1.

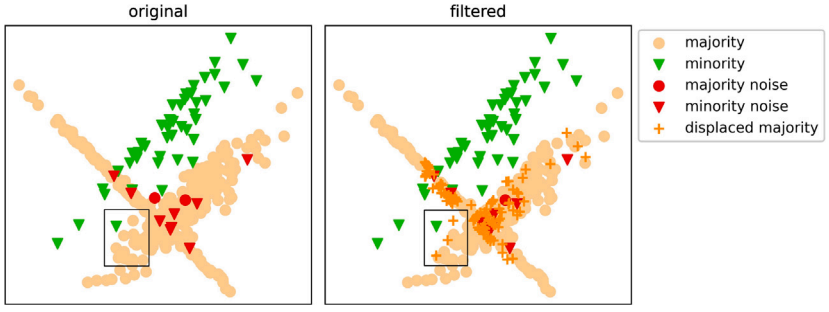

Fig. 10. The principle of the CCRNF can be observed clearly in the part of the image marked with the small rectangle. There are three majority samples around the minority one in the original data set (left). The method moves these samples away from the minority one. Their new location is marked by three crosses in the picture on the right.

samples generated by the oversampler (Fig. 10). More details can be found in [69].

The term `NoFilter` denotes the absence of any filter methods.

The implementation of most of these filters can be found in the smote_variants package [70] as a standalone noise filter or as part of an oversampler, and the algorithm of IPF_kTLNN can be found in [11]. The interested reader can find additional figures in the supplementary material, which show the effect of different noise filtering methods in case of given parameterization.

3.3. Sampling methods

As discussed in Section 2, there are numerous proposals for classifying imbalanced data sets. To gauge the potential usefulness of a specific noise filter as an initial step in an oversampler, we need to examine the effects of the filters in conjunction with various sampling strategies. However, as noted, oversamplers often incorporate some form of solution to eliminate noisy samples or mitigate their impact, so combining filters with them could yield misleading results.

To circumvent this issue, we opted to generate different oversamplers, which comprise the common steps of oversamplers identified in the literature, using the method outlined in the paper [71]. If the synthetic sample generation is based on one point, the steps include: noise filtering (including NoFilter), clustering (including the omission of clustering), steps for weighting the clusters and the samples within them, which govern the probability of selecting the samples as seeds. The final step after choosing the seeds is sample generation with Gaussian jittering [72].

For oversamplers based on two points, the selection of seeds is followed by the creation of a universal set for the co-seeds. In the next step, we form a set of pair candidates for each seed, from which the co-seeds (pairs) needed to generate the new samples are randomly selected. Each synthetic sample is created by interpolating a seed and one of its pairs.

A comprehensive description of the generated samplers can be found in the Supplementary Material.

4. Experiments

In this section, we aim to present the experimental design for studying the impact of noise filters, along with the analysis conducted and the results obtained. As we discuss the results, we will structure

Table 2
The result of methods that filter only the majority class. Only the tuneable parameters are presented in the table.

Noise filter	Parameters	$Spec_{maj}$	$Sens_{maj}$	G_{maj}	$F1_{maj}$	Spec	Sens	G	F1
CNNTL		0.95	0.21	0.28	0.03	0.96	0.03	0.08	0.02
ENN	n_neighbours: 3, enemy_rate: 0.1	0.85	0.71	0.69	0.02	0.87	0.10	0.21	0.02
ENN	n_neighbours: 3, enemy_rate: 0.5	0.98	0.43	0.50	0.09	0.98	0.06	0.15	0.04
ENN	n_neighbours: 5, enemy_rate: 0.1	0.77	0.81	0.71	0.02	0.80	0.11	0.21	0.01
ENN	n_neighbours: 5, enemy_rate: 0.5	0.99	0.32	0.39	0.11	0.99	0.04	0.12	0.04
ENN	n_neighbours: 7, enemy_rate: 0.1	0.69	0.85	0.69	0.01	0.74	0.11	0.21	0.01
ENN	n_neighbours: 7, enemy_rate: 0.5	0.99	0.26	0.33	0.11	0.99	0.04	0.10	0.04
GMMNF	M: 1, lmbd: 0.3	0.97	0.72	0.76	0.20	0.98	0.09	0.22	0.08
GMMNF	M: 1, lmbd: 0.5	0.95	0.77	0.80	0.16	0.95	0.10	0.23	0.07
GMMNF	M: 1, lmbd: 0.7	0.92	0.83	0.83	0.14	0.93	0.11	0.24	0.06
GMMNF	M: 2, lmbd: 0.3	0.98	0.70	0.74	0.22	0.98	0.09	0.21	0.08
GMMNF	M: 2, lmbd: 0.5	0.97	0.73	0.76	0.18	0.97	0.09	0.22	0.08
GMMNF	M: 2, lmbd: 0.7	0.96	0.76	0.77	0.16	0.96	0.10	0.22	0.07
GMMNF	M: 3, lmbd: 0.3	0.99	0.62	0.66	0.22	0.99	0.08	0.19	0.08
GMMNF	M: 3, lmbd: 0.5	0.98	0.66	0.69	0.19	0.98	0.08	0.20	0.07
GMMNF	M: 3, lmbd: 0.7	0.97	0.68	0.71	0.17	0.97	0.09	0.20	0.07
OSS		0.95	0.18	0.25	0.02	0.96	0.02	0.07	0.01
OSD	n_neighbours: 3	1.00	0.18	0.25	0.14	1.00	0.02	0.08	0.04
OSD	n_neighbours: 5	1.00	0.10	0.14	0.11	1.00	0.01	0.04	0.02
OSD	n_neighbours: 7	1.00	0.05	0.07	0.06	1.00	0.01	0.02	0.01
Relabeller	n_neighbours: 3, switch_rate: 0.5	0.98	0.44	0.51	0.09	0.98	0.06	0.15	0.05
Relabeller	n_neighbours: 3, switch_rate: 0.9	1.00	0.19	0.26	0.15	1.00	0.03	0.08	0.04
TLRemoval		0.98	0.18	0.24	0.03	0.98	0.02	0.07	0.02

our presentation according to the experimental steps outlined in our plan, dividing them into distinct sections.

4.1. Experiment 1 – Noise filtering

The first step was to assess the noise detection ability of the filters, which can also be seen as a classification problem between noise and non-noise samples.

Among the methods introduced in Section 3.2, we omitted CCRNF for this test because it is not based on noise detection. The parameterizations of the filters are shown in Table 2. In the rest of the paper, only the non-fixed parameters will be shown in the figures, tables, and text for the sake of clearer markings.

The instances of the noise filters were evaluated on the synthetic data sets presented in Section 3.1 according to sensitivity (Sens), specificity (Spec), G-mean score (G) and F1-score (F1), whose formulas are respectively:

$$\text{Sens} = \frac{TP}{TP+FN}, \quad \text{Spec} = \frac{TN}{TN+FP},$$
$$\text{G} = \sqrt{Sens \times Spec} \quad \text{F1} = \frac{2TP}{2TP+FP+FN},$$

where TP and FP are the number of deleted or relabelled noise and non-noise samples, respectively, and FN and TN are the number of the noise and non-noise samples left unchanged in the database. If a measure refers to only one class instead of the whole data set, it is indicated by a subscript, which is `maj` for the majority class and `min` for the minority one (e.g. $F1_{maj}$).

Before presenting the experimental results, we would like to emphasize that the noise filter instances can be classified into two groups based on whether they delete or relabel samples from the entire data set (i.e., both the majority and minority classes) or only from the majority class. In the rest of the paper, the first group will be referred to as *both-sided filters* and the filters in the second group as *majority-sided filters*.

4.1.1. Results of the majority-sided filters

Table 2 contains the mean results of the filter instances working on the majority class of the data sets. In this class, almost all the filter instances achieved higher than 90% specificity so, we can say that the filters rarely deleted samples that they should not have, except for the ENN instances with an enemy_rate of 0.1, which removed more correct samples than the others. The disadvantages of this parameterization of ENN are also shown by G-mean score.

While the high $Spec_{maj}$ values are associated with moderately good $Sens_{maj}$ for some GMMNF filter instances and the aforementioned ENN versions, the other filters proved to be weak in filtering out noise from the majority class.

The $F1_{maj}$ measure shows even more harshly that the filters had a hard time detecting noise in the majority class.

Since no filtering is performed on the smaller class, we should expect higher specificity and lower sensitivity on the entire data set. The last four columns of the table contain these values. The drastically reduced sensitivity and F1 can be explained by the distribution of noise between classes.

4.1.2. Results of the both-sided filters

Table 3 illustrates how the results vary when the filtering is applied to the entire data set. For most filters, we have already observed their performance in the majority class. In the majority class, the previously unseen IPFs demonstrated very poor noise filtering capabilities ($Sens_{maj} \leq 0.07$) and yielded similarly poor results on the smaller class as well. While the filter instances using consensus-based ensemble classification did not remove the noise ($Sens_{min} = 0$), the others, using majority voting, deleted a significant portion of the correct samples along with the noise, as can be seen from the low value of $Spec_{min}$. The IPF_kTLNN has also had problems distinguishing between noise and correct samples. Even the mildest version of the filter deleted the entire minority class for 272 data sets, while the strongest version did so for 725 data sets out of 810. These performances are likely explained by the relatively small number of samples and the imbalance of the data sets.

Although to a lesser extent, the filter instances using only 3 nearest neighbours (ENNs, Relabellers, OSD) also removed many non-noise samples from the minority classes ($Spec_{min} \leq 0.7$). By increasing the neighbourhood, we obtain milder filters that remove less noise and fewer correct samples.

Overall, GMMNFs provided the most balanced performance, removing 63%–78% of the noise and very few correct elements, which is also reflected in the $F1_{min}$ values. It is also worth looking at how similar G-mean scores can be seen for the minority and majority classes.

The GMMNF instances also achieved the highest F1 values on the entire data set, although these did not reach 50%, which underscores the difficulty of the problem.

While it may not seem fair to compare filters, each designed with a different purpose, across the entire data set, it is important to note that

Table 3
Mean performance of filter instances that operate in both classes. The parameter enemy_rate indicated in the table refers to the majority class, in the case of the minority class, enemy_rate was 0.9.

Noise filter	Parameters	$Spec_{maj}$	$Spec_{min}$	$Sens_{maj}$	$Sens_{min}$	G_{maj}	G_{min}	$F1_{maj}$	$F1_{min}$	Spec	Sens	G	F1
ENN	n_neighbours: 3, enemy_rate: 0.1, enemy_rate_for_minor: 0.9	0.85	0.68	0.71	0.85	0.69	0.75	0.02	0.29	0.84	0.84	0.83	0.11
ENN	n_neighbours: 3, enemy_rate: 0.5, enemy_rate_for_minor: 0.9	0.98	0.68	0.44	0.85	0.51	0.75	0.09	0.29	0.95	0.80	0.87	0.24
ENN	n_neighbours: 5, enemy_rate: 0.1, enemy_rate_for_minor: 0.9	0.77	0.78	0.81	0.76	0.71	0.75	0.02	0.32	0.78	0.77	0.77	0.08
ENN	n_neighbours: 5, enemy_rate: 0.5, enemy_rate_for_minor: 0.9	0.99	0.78	0.36	0.76	0.44	0.75	0.13	0.32	0.97	0.71	0.82	0.28
ENN	n_neighbours: 7, enemy_rate: 0.1, enemy_rate_for_minor: 0.9	0.69	0.83	0.85	0.68	0.69	0.73	0.01	0.33	0.72	0.71	0.70	0.06
ENN	n_neighbours: 7, enemy_rate: 0.5, enemy_rate_for_minor: 0.9	0.99	0.83	0.32	0.68	0.39	0.73	0.14	0.33	0.98	0.64	0.78	0.29
GMMNF	M: 1, lmbd: 0.3	0.97	0.98	0.72	0.63	0.76	0.75	0.20	0.56	0.97	0.64	0.76	0.35
GMMNF	M: 1, lmbd: 0.5	0.95	0.96	0.77	0.71	0.80	0.81	0.16	0.57	0.95	0.72	0.81	0.32
GMMNF	M: 1, lmbd: 0.7	0.92	0.94	0.83	0.77	0.83	0.84	0.14	0.56	0.92	0.78	0.83	0.30
GMMNF	M: 2, lmbd: 0.3	0.98	0.98	0.70	0.66	0.74	0.76	0.22	0.60	0.98	0.67	0.78	0.44
GMMNF	M: 2, lmbd: 0.5	0.97	0.97	0.73	0.72	0.76	0.81	0.18	0.60	0.97	0.72	0.82	0.41
GMMNF	M: 2, lmbd: 0.7	0.96	0.96	0.76	0.76	0.77	0.83	0.16	0.60	0.96	0.76	0.84	0.39
GMMNF	M: 3, lmbd: 0.3	0.99	0.99	0.62	0.64	0.66	0.75	0.22	0.60	0.99	0.63	0.76	0.46
GMMNF	M: 3, lmbd: 0.5	0.98	0.98	0.66	0.70	0.69	0.79	0.19	0.61	0.98	0.69	0.80	0.44
GMMNF	M: 3, lmbd: 0.7	0.97	0.97	0.68	0.74	0.71	0.82	0.17	0.61	0.97	0.73	0.82	0.42
IPF	voting: consensus, max_depth: 3	1.00	1.00	0.02	0.00	0.03	0.00	0.02	0.00	1.00	0.00	0.01	0.00
IPF	voting: consensus, max_depth: 5	1.00	1.00	0.00	0.00	0.01	0.00	0.01	0.00	1.00	0.00	0.00	0.00
IPF	voting: majority, max_depth: 3	0.99	0.14	0.07	0.98	0.11	0.33	0.03	0.17	0.89	0.86	0.87	0.16
IPF	voting: majority, max_depth: 5	0.99	0.24	0.06	0.93	0.09	0.45	0.03	0.18	0.90	0.82	0.85	0.17
IPF_kTLNN	n_neighbours: 3, mul: 1.4	0.99	0.21	0.34	0.99	0.42	0.34	0.14	0.19	0.90	0.91	0.90	0.18
IPF_kTLNN	n_neighbours: 5, mul: 1.4	0.99	0.07	0.23	1.00	0.29	0.12	0.14	0.16	0.88	0.90	0.89	0.16
IPF_kTLNN	n_neighbours: 7, mul: 1.4	1.00	0.03	0.16	1.00	0.22	0.05	0.12	0.15	0.88	0.89	0.88	0.15
OSD	n_neighbours: 3	1.00	0.70	0.18	0.79	0.25	0.73	0.14	0.28	0.97	0.71	0.82	0.27
OSD	n_neighbours: 5	1.00	0.78	0.10	0.74	0.14	0.75	0.11	0.32	0.98	0.66	0.80	0.31
OSD	n_neighbours: 7	1.00	0.83	0.05	0.67	0.07	0.73	0.06	0.33	0.98	0.60	0.75	0.31
Relabeller	n_neighbours: 3, switch_rate: 0.5	0.98	0.39	0.44	0.98	0.51	0.60	0.09	0.22	0.91	0.91	0.91	0.20
Relabeller	n_neighbours: 3, switch_rate: 0.9	1.00	0.68	0.19	0.85	0.26	0.75	0.15	0.29	0.96	0.77	0.85	0.28
TLRemoval		0.98	0.87	0.18	0.42	0.24	0.57	0.03	0.25	0.96	0.39	0.58	0.16

oversamplers, with very few exceptions, rely solely on minority samples to generate new ones. As such, we cannot overlook the significant difference in sensitivity and F1 values between filters that operate on the entire database and those that focus only on the majority class. The subsequent experiment aimed to determine whether samplers, some of which attempt to mitigate the adverse effects of noise by weighting the samples, could offset these disparities.

At the same time, we have also seen the potential drawbacks of both-sided filters. Some of them tend to remove so many correct samples from the minority class that it makes it impossible to perform meaningful analysis later, since cross-validation splits the data sets into even smaller parts.

Therefore further analysis and tests were carried out on a selected subset of the noise filters. To be more specific, we excluded methods with $Spec_{min} < 0.7$ from the previously used filter instances. This included ENNs and Relabellers that use three nearest neighbours and operate on the entire data set, as well as the IPF_kTLNN filters, and IPF filter with majority voting. Other IPF filters were also omitted from this experiment due to their poor performance (F1=0.00).

However, before diving into the impact of sampling strategies, let us first examine the relationship between the properties of the data sets and the outcomes achieved by the noise filters.

4.1.3. Connection between the meta-features and the performance of noise filters

To get an idea of how the properties of the data set are related to the performance of the filters, we calculated the Spearman correlation between several known characteristics of the data sets (such as the number of samples, number of attributes, noise level, IR, and number of clusters, other metafeatures computed by the `PyMFE` package [73], and `AugR_value`, which is a measure used to describe the overlap of imbalanced data sets) and the results (F1, G, Spec, Sens) obtained from the first experiment.

For the both-sided noise filters, the F1 value demonstrated the strongest correlation with the noise level in the data sets. However, in the case of GMMNF, this correlation was only moderate (as shown in Fig. 11). For the majority-sided filters, we did not identify any metafeature that exhibited a strong correlation with F1 or G. The absolute value of the correlation coefficient varied between 0.40 and 0.58. The `nodes_per_inst` shows a strong negative correlation with both specificities for ENNs, TLRemovals, and majority-sided Relabeller. However, no meta-features were found that showed strong correlation with sensitivity.

The detailed results can be found in the Supplementary Materials, and the description of the different meta features is available on the PyMFE's website [74].

4.2. Experiment 2 – Sampling with noise filtering step

In this experiment, we investigated the effect on classification performance of preparing imbalanced data sets with samplers that start with some noise filtering step.

As we anticipated in Section 4.1.2, the filters built into the samplers were selected chosen based on the results of the preceding experiment and the necessity for a certain number of samples to evaluate classifier performance.

On data sets of the size we deal with in the study, the performance of the classifiers is typically estimated by stratified cross-validation [75], during which the data sets are divided into even smaller parts. Consequently, we opted for filters that effectively preserved the correct minority samples.

4.2.1. Process of the experiment

We carried out our experiment on the data sets described in Section 3.1, which were divided into four parts: one noisy and 3 three noise-free. We used the samplers described in Section 3.3 containing

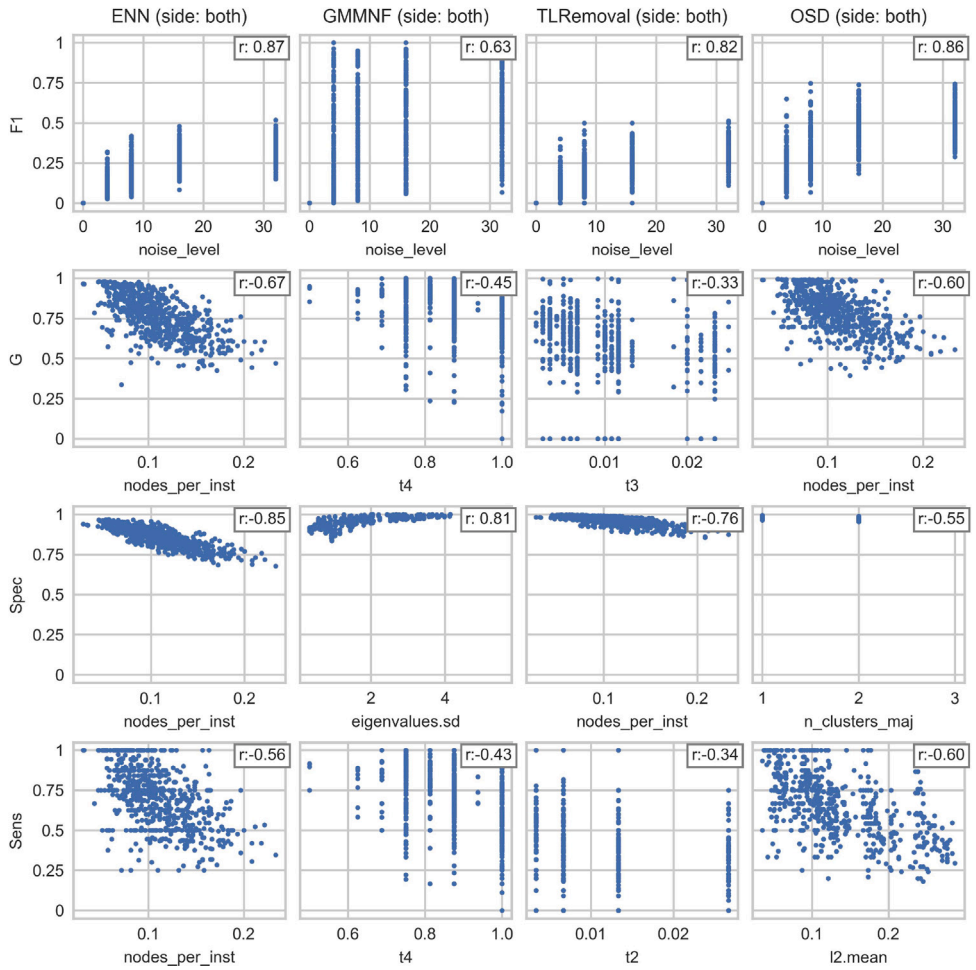

Fig. 11. Strongest correlations between meta-features of the data sets and performance scores of some both-sided noise filters.

selected noise filters and NoFilter and, uniform kNN classifiers with k values of 3, 5 and 7. We did not explore larger k values due to the limited sample size of the minority set.

The training, validation, and testing processes were executed as follows for each sampler and data set. We employed stratified cross-validation (5-fold) on the noisy part of a data set. During this process, we sampled the training part, constructed the kNN classifier, and evaluated the results in the remaining part. It is important to note that the validation part is noisy, as we typically lack knowledge about which samples are noisy and which are not in practical scenarios.

On the other hand, it is crucial to obtain unbiased information about the classification results achieved after filtering and sampling. Therefore, we conducted additional training using the parameters of the classifier that yielded the highest F1 score during cross-validation (which naturally included a filtering step in the sampling process). We then tested this model on the noise-free parts of the data set. The final result of the classification combined with sampling is given by the average of the F1 scores obtained for the three noise-free test sets.

For the sake of simplicity, we will refer to the classification results, which are derived from a data set prepared with a sampling method that includes a specific noise filtering step, to the results of the noise filter, where this does not cause confusion.

The experiment initially aimed to attain complete balance between classes, and the subsequent sections of the paper predominantly delve into this scenario. Subsequently, we assessed the robustness of the results by varying the number of newly generated samples to represent 75%, 50%, or 25% of the difference between the quantities of majority and minority samples after the noise filtering. These variations are referred to as balancing ratios. (It is important to note that not all samplers possess the capability to consistently produce the desired number of samples across all scenarios.)

4.2.2. Statistical tests

To ascertain if there is a statistically significant difference between the mean values of the results of the different filter instances, we used the Friedman and Nemenyi test. This test is a standard approach for comparing classifier performance when normality cannot be assumed (this was visually verified) [76]. During the study, our null hypothesis was that the average performance value for each noise filter was identical to the others. The study revealed that this null hypothesis could be rejected.

According to the Nemenyi test, in most instances, there is a significant difference between the means associated with the different noise filters regardless of the degree of re-balancing. However, this is not necessarily the case between different parameterizations of a filter if they belong to the same group (both-sided vs. majority-sided).

The differences between the individual methods are illustrated in Figs. 12 for the completely balanced data sets. Further results are available in the Supplementary Material. The upper, darker parts of the violin bodies represent the results of the classification results on the noise-free data sets.

It is worth noting that most filters did not cause significant deterioration of the noise-free data sets. The exceptions were ENNs that implement stronger filtering (side: majority, enemy_rate: 0.1).

In addition, some both-sided methods, even caused a slight improvement compared to the results of NoFilter.

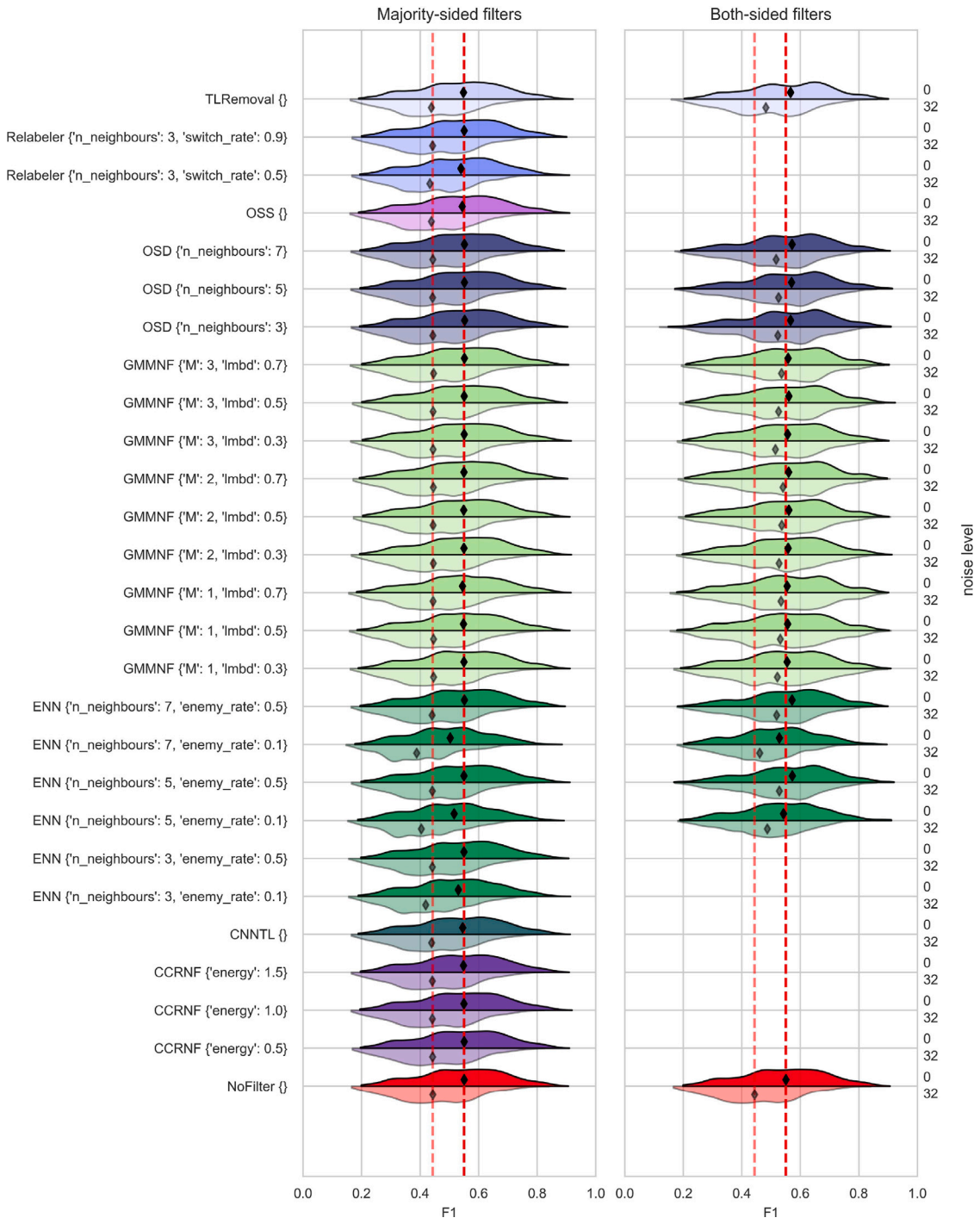

Fig. 12. F1 values of kNN classifiers trained on completely balanced noise-free and noisy data sets, broken down by the noise filter step of the sampling methods. We note that CCRNF only moves the samples of the majority class away from the vicinity of the minority samples, and for both-sided ENN, the enemy_rate parameter refers to the majority class. Only isolated samples are removed from the minority class. The vertical red lines show the results obtained with the samplers with NoFilter, which can be considered as baselines. The diamonds indicate the median values of the F1 scores.

The lower part of the violins display the results measured on the data sets with the highest noise levels. It can be observed that the differences become more pronounced as the noise level increases. The performance of the majority-sided filters significantly declined, while most of the both-sided filters showed only a small decrease in F1 values. The versions of ENN that implement stronger filtering generally underperformed, and TLRemoval also did not produce convincing results when noise levels were high. The both-sided OSDs, GMMNFs, and the milder ENNs performed relatively well, but the elongated violin shapes (especially OSD with n_neighbours: 3) indicates that we can expect both very good and very bad results from its use.

These observations remained consistent across all tested balance ratios. However, the results also indicate that as more new samples are introduced into the data set, the noise filtering step becomes increasingly crucial. Furthermore, without noise filtering, partial balancing of the data set yielded greater classification benefits compared to complete balancing.

The literature generally agrees that deleting from the minority class poses a high risk, as it can lead to the loss of crucial information.

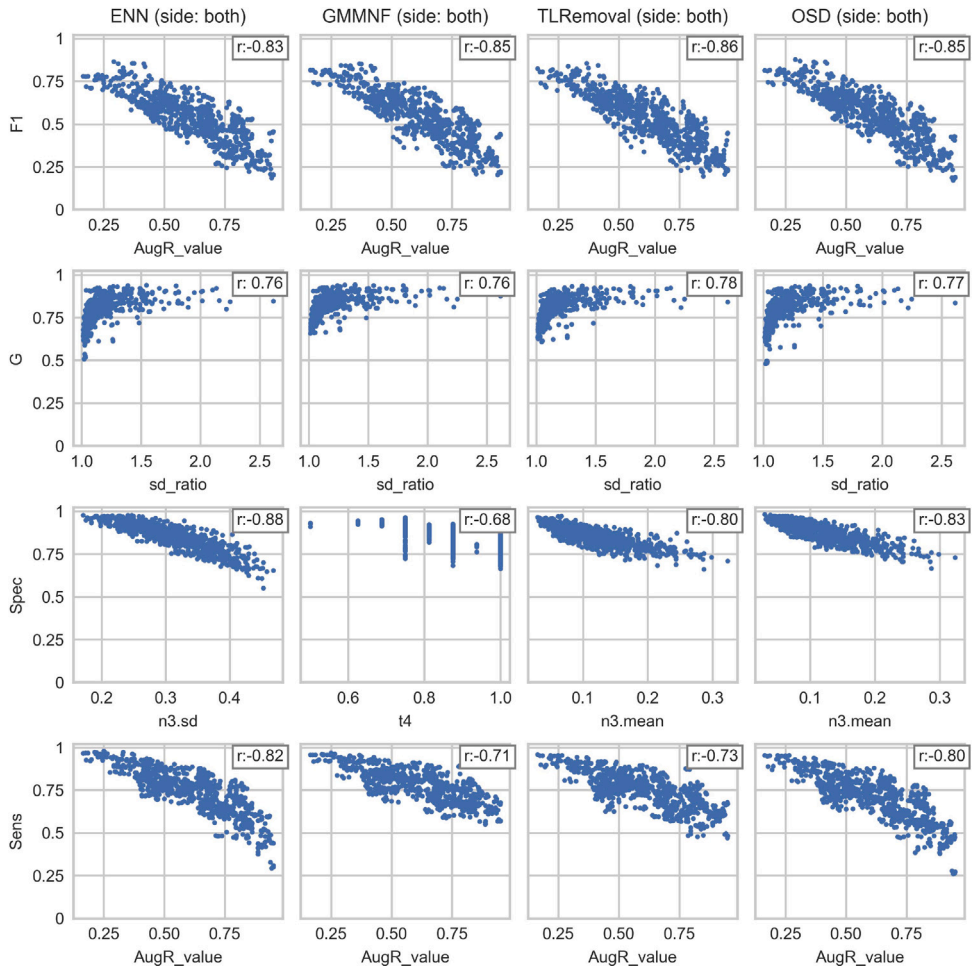

Fig. 13. The strongest correlations between the meta-features of the imbalance data sets and the performance of the kNN classifier performed on the balanced data sets. The columns indicate the noise filters used as the initial step of the sampling methods.

Rather than imposing a categorical ban, a more nuanced approach is necessary. Our measurements indicate that cleaning the minority class is particularly important, especially when dealing with high noise levels.

It is also important to mention to the samplers that most of them do not perform permanent noise filtering. This means that noisy samples will remain part of the data set, even if they are not expanded. In our experiment, out of 810 data sets, there was only one where sampling without noise filtering resulted in the best performing classification model. Based on these findings, we strongly recommend the use of the noise filtering step.

4.2.3. Connection between the meta-features and performance of samplers with noise filter

A similar study was undertaken to map the relationships between the meta-features that characterize the original data sets used in the experiments and the performance score that quantifies the efficiency of the samplers with noise filter. The strongest correlations are depicted in Fig. 13.

In this experiment we no longer measured a strong correlation between noise level and F1. This time, another indicator, `AugR_value`, showed a strong negative correlation with F1 scores across all examined samplers with noise filters. To recall, `AugR_value` is used to describe the overlap between classes in imbalanced data sets. It can be concluded that there is a strong correlation between G-mean score and `sd_ratio`, which expresses the result of statistical homogeneity test of covariances.

Furthermore, a strong correlation was found between `n3` mean and Spec of noise filters based on kNN, which is hardly surprising, because `n3` represents the error rate of the Nearest Neighbour classifier.

4.2.4. Connection between the performance of the noise filters and the performance of the samplers

For real data sets, it is rarely possible to measure the performance of noise filters, since we usually have no information about which samples have incorrect labels in the training set. For this reason, it is also not possible to select the best performing noise filter on the data set before sampling and classification. It is therefore an interesting question how well the general noise removal ability of the filters correlates with the classification results on a balanced data set when the sampling method used for balancing includes the noise filter.

Since Experiment 1, which evaluated the noise detection capabilities of the noise filters, and Experiment 2, which indirectly assessed the performance of the noise filters as an initial step of a sampler, were performed on similar but independent set of data sets, measuring the correlations between the metrics obtained in the two experiments could bring us closer to the answer. For the both-sided filters, the strongest correlations are depicted in Fig. 14. It is important to note that not all measures are interpretable across all data sets, hence the number of values used to measure correlation is not necessarily match the number of data sets.

We can see that there are significant differences among the filters. For example, for ENN and OSD, the correlation between the noise filter performance and classifier performance (F1, Sens) is considerably stronger than for GMMNF and TLRemoval.

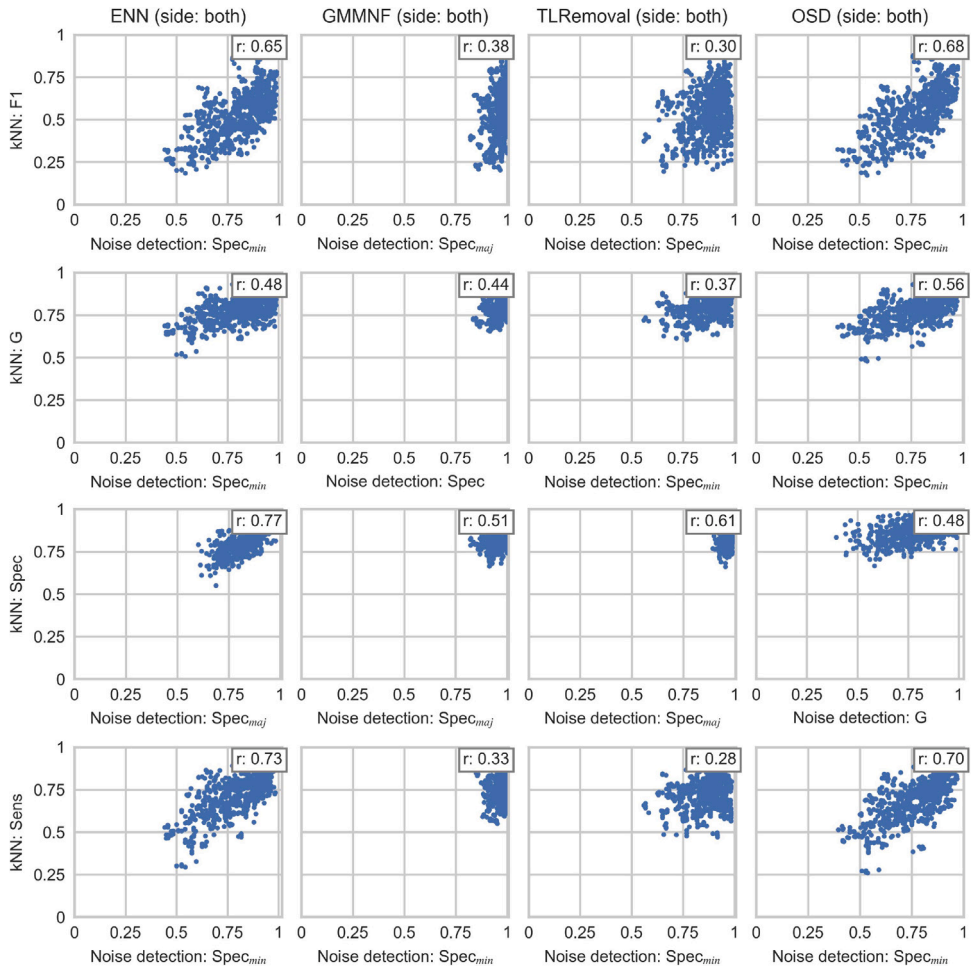

Fig. 14. The strongest correlations are between the performance of the noise filters and the performance of the kNN classifier performed on the balanced data sets. The columns indicate the noise filters used as the initial step of the sampling methods used to balance the data set.

The performance of the majority-sided filters shows only a weak correlation with the classification result in terms of both F1 and sensitivity. Among these filters, ENN demonstrates a stronger relationship (0.74 majority-sided, 0.77 both-sided filtering) between filter's specificity and the specificity of classification.

4.3. Robustness of the methods

From the previous experiment, we have a preliminary understanding of which filters perform better as part of a sampling method. However, it is intriguing to observe the stability of these differences. For instance, does the same filter emerge as the best when we apply different steps after noise filtering?

To answer this question, we established a ranking system among the filter instances. This was based on the F1 values obtained for different databases, as outlined in Experiment 2. A filter was deemed superior to another if the Wilcoxon showed a significant difference in favour of it in terms of the mean F1 values. The rank of a filter is determined by the total number of filters tested and the number of filters it surpasses. (While there may be ties, the ranking is dense.)

Subsequently, we divided the results into increasingly smaller subsets based on the steps of the samplers (refer to Section 3.3). We then determined the ranking of the filters for each subset, yielding an additional 73 ranks per filter.

The *Total_ranks* column in Table 4 contains the ranks of the filter instances based on all their corresponding results. The subsequent four columns present the mean, standard deviation, minimum, and maximum of the 74 ranks.

In Section 4.2.2, we highlighted that the samplers, which started with noise filters operating in both classes, enhanced the classification more effectively than the others. This observation is also reflected in the table that contains the filter rankings (Table 4). In most instances, the standard deviation of the rankings is smaller in this group compared to the other.

In ten instances, the filter, with the same parameterization applied to the majority class, ranked lower than the both-sided version with the worst combination (Max_2), even when it had the most favourable combination of steps (Min_1).

Subsequently, we illustrate the ranks in a tree to demonstrate how the combination of steps influences the order of the filters. (Due to space constraint, we focus here only on the completely balanced case, and within that, only on the best methods, but figures are available for all methods in the Supplementary Material.)

The root contains the Total_rank of the filter instance, and the ranks observed at the other nodes were computed based on the results of the oversamplers that incorporated the methods seen on the paths leading to the nodes. If the synthetic sample generation is based on a single point, the steps for selecting a pair are omitted, which is indicated by 'None' written on the edges. The method applied in the sample generation step is not depicted in the figure, as it is always Gaussian jittering for branches containing 'None', and interpolation otherwise.

Fig. 15 displays the values associated with ENN, which is ranked first. The edges starting from the root denote the clustering method. CLR_MinMaj on the lower branch separates the majority and minority classes. The other two edges represent the clustering performed on the

Table 4
The ranks of the noise filters based on the F1 values of kNN classifiers trained on the balanced data sets. The Total_rank columns display the rankings of the filters, taking into account all the oversamplers in which they are included. Additional columns provide basic statistics for rankings that are based on different subsets of oversamplers. NoFilter is included as a baseline for comparison.

Noise filter	Parameters	Remove from the majority class					Remove from both classes				
		$Total\_rank_1$	$Mean_1$	Std_1	Min_1	Max_1	$Total\_rank_2$	$Mean_2$	Std_2	Min_2	Max_2
CNNTL		26	21.26	2.67	15	28					
ENN	n_neighbours: 3, enemy_rate: 0.1	29	23.49	2.79	17	30					
ENN	n_neighbours: 3, enemy_rate: 0.5	23	19.41	2.32	13	24					
ENN	n_neighbours: 5, enemy_rate: 0.1	30	24.49	2.79	18	31	14	11.43	2.83	7	21
ENN	n_neighbours: 5, enemy_rate: 0.5	21	18.05	2.54	12	23	1	1.68	1.31	1	6
ENN	n_neighbours: 7, enemy_rate: 0.1	31	25.49	2.79	19	32	27	21.12	2.61	15	27
ENN	n_neighbours: 7, enemy_rate: 0.5	20	17.18	2.47	12	22	4	4.84	1.64	2	8
GMMNF	M: 1, lmbd: 0.3	16	13.00	2.03	9	18	10	7.89	1.33	5	11
GMMNF	M: 1, lmbd: 0.5	19	16.09	2.94	9	23	7	5.59	1.37	2	9
GMMNF	M: 1, lmbd: 0.7	24	19.57	2.74	13	25	3	3.91	1.58	1	7
GMMNF	M: 2, lmbd: 0.3	16	12.66	1.75	9	16	11	7.50	1.38	4	11
GMMNF	M: 2, lmbd: 0.5	17	14.00	2.24	9	20	8	5.18	1.47	2	8
GMMNF	M: 2, lmbd: 0.7	19	16.00	2.92	9	22	3	2.88	1.37	1	6
GMMNF	M: 3, lmbd: 0.3	16	12.32	1.95	8	16	12	8.35	1.45	5	12
GMMNF	M: 3, lmbd: 0.5	15	12.28	1.67	8	15	9	6.15	1.54	3	9
GMMNF	M: 3, lmbd: 0.7	16	12.84	1.96	9	19	5	3.76	1.58	1	7
OSD	n_neighbours: 3	20	17.70	3.10	12	24	4	4.76	2.60	1	9
OSD	n_neighbours: 5	19	15.68	2.25	11	21	2	2.69	1.31	1	6
OSD	n_neighbours: 7	18	15.27	2.48	9	22	6	5.46	1.87	2	8
OSS		26	21.27	2.72	15	28					
Relabeller	n_neighbours: 3, switch_rate: 0.5	28	22.46	2.79	16	29					
Relabeller	n_neighbours: 3, switch_rate: 0.9	20	17.05	2.32	12	22					
TLRemoval		25	20.70	2.41	14	26	13	9.38	1.37	6	13
CCRNF	energy: 0.5	19	16.00	2.40	10	22					
CCRNF	energy: 1.0	20	16.99	2.73	10	22					
CCRNF	energy: 1.5	22	19.12	2.35	13	24					
NoFilter		19	15.85	2.36	10	21	19	15.85	2.36	10	21

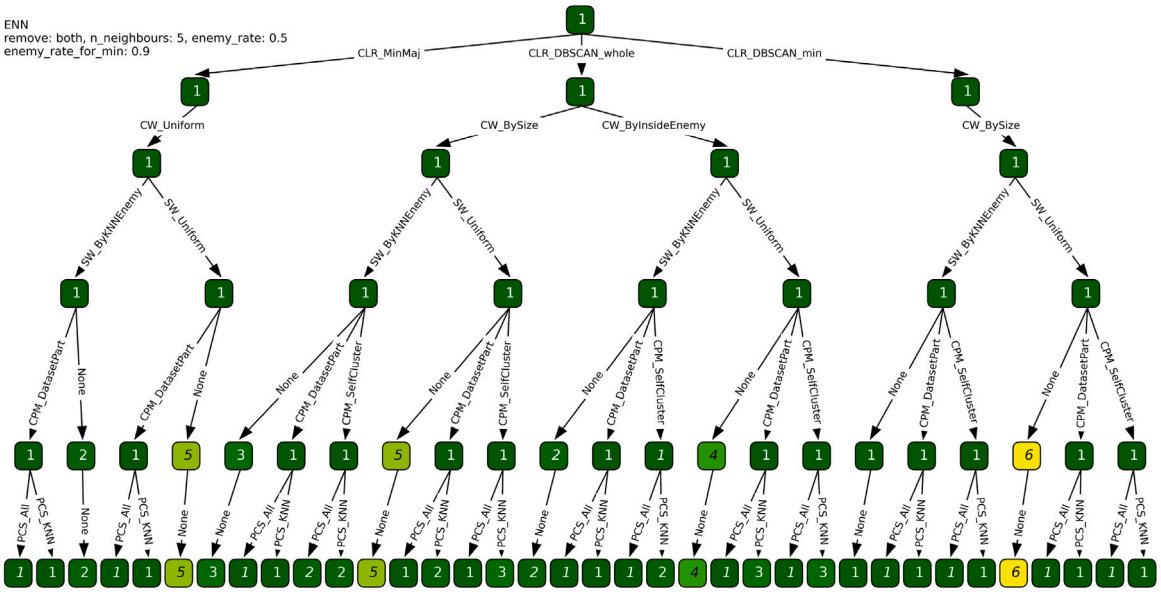

Fig. 15. The figure illustrates how the introduction of a new step influences the global rank of ENN. The colour-coding of the vertices (from green to red) provides a visualization of the global rank, thereby facilitating the comparison of figures representing other methods in the Supplementary Material. The italic style indicates that at least one more filter instance had the same rank as part of a sampling method with the same combination of steps.

entire data set (CLR_DBSCAN_whole) and the clustering applied to the minority class (CLR_DBDCAN_min). In the subsequent step, CW_BySize indicates that more seeds are selected from larger clusters compared to smaller ones, while CW_ByInsideEnemy leads to more seeds being selected from clusters on the decision boundary, thereby strengthening the minority samples at risk of misclassification.

It can be observed that the clustering and cluster weighting steps did not cause a significant difference that would have jeopardized the placement of the filter. However, clear differences emerge between samplers based on one and two points. ENN fell behind in the rankings when we used one-point based sample generation, specifically when seed points were selected from clusters by uniform sampling (SW_Uniform).

The latter two statements also apply to OSDs operating in both classes. As can be seen in Fig. 16, if we restricted the set of co-seeds to the nearest minority neighbours of the seeds, it could negatively impact the classification, despite this being a common practice.

The top-ranked OSD was followed by two GMMNFs, for which we observed contrasting patterns. If oversampling commenced with such

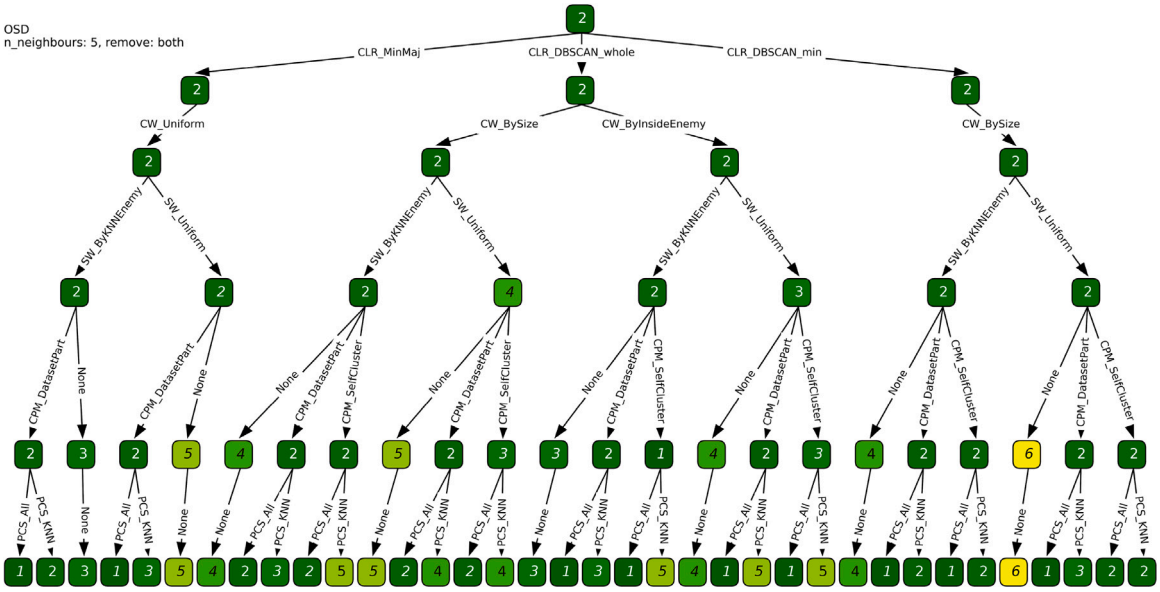

Fig. 16. The figure illustrates how the introduction of a new step influences the global ranking of OSD. The colour gradient of the vertices, ranging from green to red, provides a visual representation of the global rank. This allows for a comparison with figures representing other methods found in the Supplementary Material.

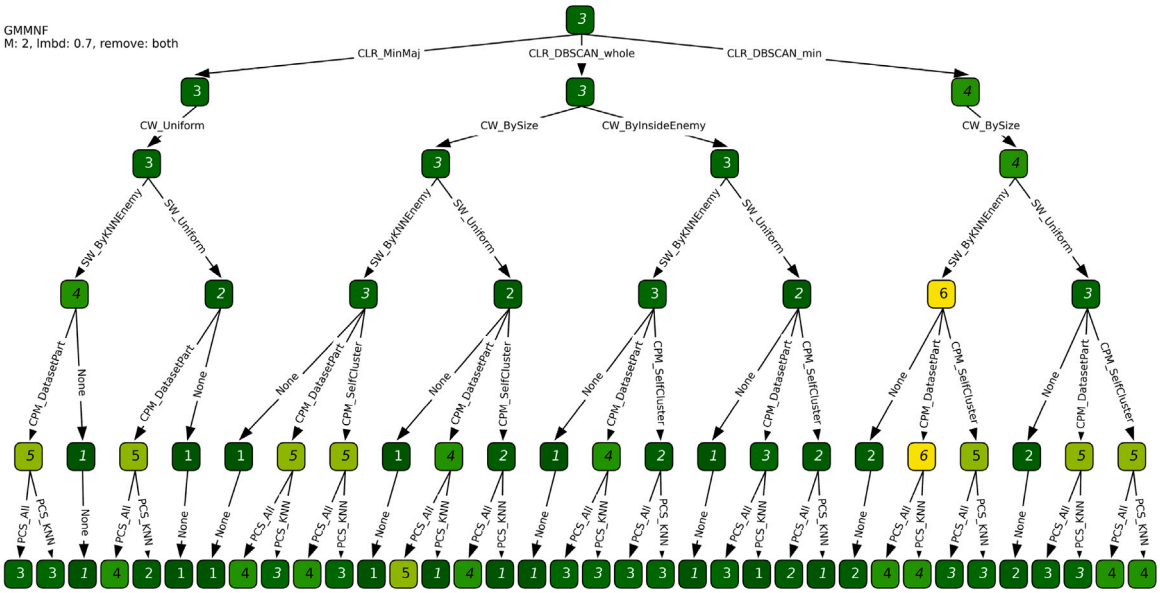

Fig. 17. The figure demonstrates how the introduction of a new step impacts the global ranking of GMMNF. The colour gradient of the vertices, transitioning from green to red, provides a visual representation of the global rank. This visualization facilitates comparison with figures representing other methods in the Supplementary Material.

filtering, then generating samples based on a single point yielded superior results. Conversely, when generating samples based on two points, we typically anticipated better outcomes if the pairs were selected from the nearest neighbours of the seeds. However, this factor became less significant if both the seeds and the pairs were chosen from the same cluster. Please refer to Fig. 17 for more details.

The figures mentioned above take into account the results derived from all data sets. However, Fig. 18 specifically illustrates the variation in the Total_rank of both-sided filters for data sets with differing noise levels.

It is crucial to highlight that an increase in noise level prompts a shift in the ranking order. Notably, GMMNF surpasses both ENN and OSD. We observed similar changes when we did not aim to completely balance the data sets. Some examples of GMMNF have been placed in front of the best performing ENN and OSD. With a balance of 25%, the GMMNF instances took 1st–7th place, and even a TLRemoval was ahead of the best ENN.

4.3.1. Compare to literature methods

In this experiment, we selected a subset of the best-performing filters, namely the both-sided GMMNF instance (M: 2, lambda: 0.7), the both-sided ENN (n_neighbours: 5, enemy_rate: 0.5), and the both-sided OSD (n_neighbours: 5), along with sampler methods documented in the literature.

The selection of the sampler methods was based on Kovács' study [13], which ranked 86 sampler algorithms in combination with different classifiers. We specifically opted for methods that demonstrated strong support for kNN classification, as measured by F1 scores. Additionally, we included samplers with clustering capabilities (DBSMOTE, Kmeans_SMOTE), a sampler designed to preserve borderline samples

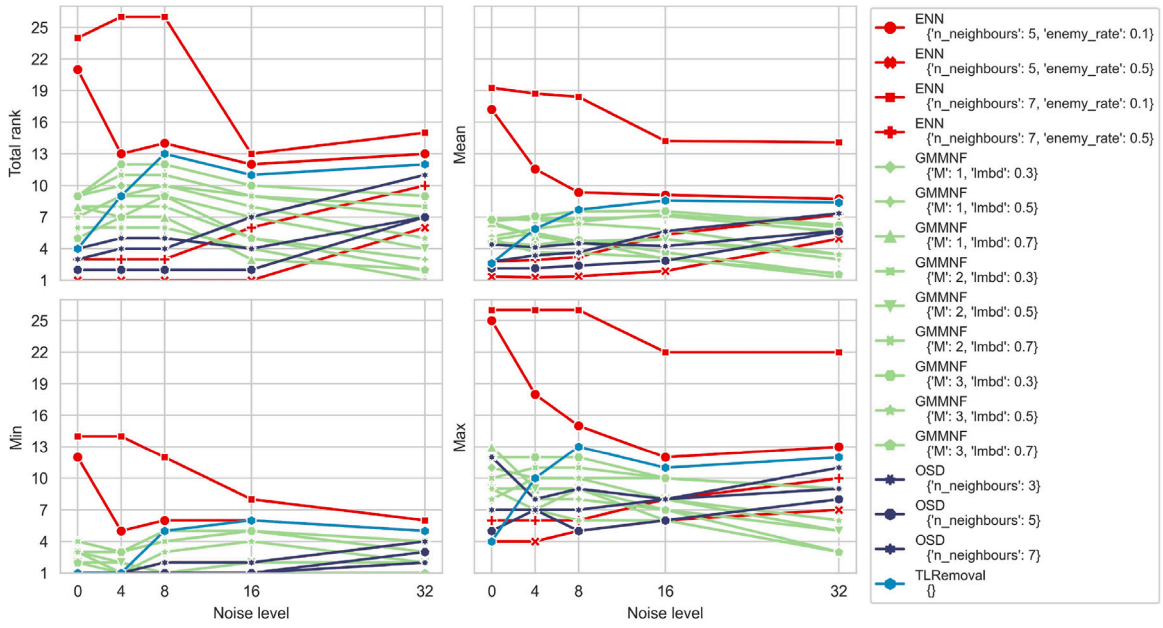

Fig. 18. The figure illustrates the variation in the ranks of both-sided filters in response to increasing noise levels. It is important to note that the noise level is represented as a percentage of the minority samples from the original data set, which is devoid of noise.

Table 5
Median of F1 values of KNN classification on data sets prepared by various sampling methods without (F1) and with extra filtering step (GMMNF, ENN, OSD). Cases are highlighted in bold where the difference is statistically significant in favour of the sampling method combined with GMMNF, ENN, or OSD.

Method	Noise level: 0%				Noise level: 32%			
	F1	GMMNF	ENN	OSD	F1	GMMNF	ENN	OSD
Borderline_SMOTE1 [47]	0.5888	**0.5924**	0.5800	0.5850	0.5318	**0.5716**	**0.5416**	**0.5412**
CCR [69]	0.5481	**0.5577**	0.5556	0.5528	0.4818	**0.5463**	**0.5235**	**0.5183**
CE_SMOTE [78]	0.5654	**0.5782**	**0.5813**	**0.5764**	0.4643	**0.5590**	**0.5313**	**0.5360**
CURE_SMOTE [79]	0.5259	**0.5343**	**0.5414**	**0.5399**	0.4485	**0.5265**	**0.5142**	**0.5079**
DBSMOTE [15]	0.5483	**0.5534**	0.5381	0.5337	0.4785	**0.5418**	**0.5109**	**0.5048**
Gaussian_SMOTE [80]	0.5835	**0.5858**	0.5580	0.5645	0.5345	**0.5818**	**0.5481**	**0.5489**
Kmeans_SMOTE [53]	0.5190	**0.5340**	0.4868	0.4831	0.5095	**0.5411**	0.4942	0.4882
LLE_SMOTE [81]	0.5482	**0.5607**	0.5576	0.5482	0.4842	**0.5486**	**0.5319**	**0.5328**
Lee [43]	0.6044	**0.6031**	0.5934	0.5978	0.5406	**0.5846**	**0.5694**	**0.5611**
Polynom_fit_SMOTE [82]	0.4565	**0.4680**	0.4510	0.4447	0.4360	**0.4539**	**0.4383**	0.4353
ProWSyn [83]	0.5454	**0.5554**	**0.5710**	**0.5682**	0.4518	**0.5362**	**0.5331**	**0.5322**
SMOTE [33]	0.5686	**0.5781**	**0.5943**	**0.5929**	0.4514	**0.5566**	**0.5430**	**0.5443**
SMOTEWB [52]	0.5689	**0.5760**	**0.5790**	**0.5751**	0.4703	**0.5509**	**0.5438**	**0.5439**
Safe_Level_SMOTE [84]	0.5246	**0.5361**	**0.5596**	**0.5522**	0.4326	**0.5188**	**0.5080**	**0.5049**
Supervised_SMOTE [85]	0.6145	0.6125	0.6103	0.6093	0.5510	**0.5979**	**0.5836**	**0.5846**

(as outlined in [77]), and a sampler incorporating post-filtering techniques alongside the original SMOTE.

Following a methodology similar to the one detailed in Section 4.2.2, we conducted the experiment. The results for the completely balanced scenario are presented in Table 5, covering instances with 0% and 32% noise.

Among the filters tested, the GMMNF instance emerged as the most reliable. Notably, ENN and OSD exhibited benefits primarily at higher noise levels.

5. Conclusion

The experiments detailed in the paper lead us to several important conclusions. Our results show that noise filtering, applied as part of sampling and working on the entire data set, can enhance the quality of kNN classification. As the noise level escalates, the advantageous impact of the noise filtering step intensifies. Our study involved various noise filters, and we observed differences in their efficiency. This allowed us to establish a ranking. Based on our experiments, ENN (n_neighbours: 5, enemy_rate: 0.5), OSD (n_neighbours: 5) and GMMNF (M: 2, lmbd: 0.7) working on the entire data set had the best impact on classification as a part of a sampling method. The first two filters owe their ranking to the good results achieved on moderately noisy data sets, while in the case of data sets heavily burdened with noise, different instances of GMMNF performed better. The GMMNF also came out on top when the data sets were not fully balanced.

Interestingly, our experiment also revealed that using a suitable filter can eliminate a substantial number of noise samples without inflicting considerable damage to the smaller class. Specifically, the GMMNF instances managed to eradicate 62%–83% of the minority class noise with minimal false deletions (1%–8%), presenting a viable compromise in numerous scenarios. In high noise level scenarios, GMMNF surpasses all other methods. Consequently, it can be inferred that the rule permitting deletion only from the majority set is not universally applicable. It may be beneficial to consider deletion from the minority set for more effective noise filtering.

The efficiency of noise filters is predominantly influenced by a few highly "expressive" meta-features of the data set. If we focus solely on the noise filters, the noise level is one of these features. However, if we consider a noise filter applied as part of a sampling method, the degree of overlap of the classes also significantly affects efficiency, in addition to the noise level.

Our results also indicate that different sampling strategies should be employed following different noise filters. In the case of ENN and OSD, generating new synthetic samples based on two points was clearly more successful than methods based on one point. However, if the filtering was done with GMMNF, the oversamplers based on one point often better supported the classification.

As mentioned in the Abstract, numerous issues remain to be clarified concerning the balancing of noisy data sets. Although our experiments suggest that better results can be achieved by permanently removing noise before sample generation than by relying on mechanisms to prevent noise propagation, the classifier used in the experiment is known to be sensitive to noise [86]. This issue should also be examined for other classifiers in future studies.

Another intriguing aspect to explore is the potential impact of altering the nature of the noise on the results.

It must also be admitted that the use of synthetic data sets was more of a necessity than an optimal choice for obtaining precise measurements. Although some authors have attempted to add noise to real data sets, since label errors are often the result of expert error, we cannot guarantee that an annotated data set is devoid of noise.

CRediT authorship contribution statement

Szilvia Szeghalmy: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Conceptualization. **Attila Fazekas:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is in supplementary materials. There is a link to download it.

Supplementary Materials (Original data)

Acknowledgment

This research was supported by the University of Debrecen Program for Scientific Publication.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2024.112236.

References

[1] R.K. Mishra, S. Urolagin, J.A.A. Jothi, P. Gaur, Deep hybrid learning for facial expression binary classifications and predictions, Image Vis. Comput. 128 (2022) 104573.
[2] A. Nowakowski, Remote sensing data binary classification using boosting with simple classifiers, Acta Geophys. 63 (2015) 1447–1462.
[3] B. Biggio, B. Nelson, P. Laskov, Support vector machines under adversarial label noise, in: Asian Conference on Machine Learning, PMLR, 2011, pp. 97–112.
[4] A.N. Sadigh, T. Bahraini, H.S. Yazdi, Robust classification via clipping-based kernel recursive least lncosh of error, Expert Syst. Appl. 198 (2022) 116811.
[5] W. Zhu, Y. Song, Y. Xiao, Robust support vector machine classifier with truncated loss function by gradient algorithm, Comput. Ind. Eng. 172 (2022) 108630.
[6] J. Mingers, An empirical comparison of pruning methods for decision tree induction, Mach. Learn. 4 (1989) 227–243.
[7] H.A. Abu Alfeilat, A.B. Hassanat, O. Lasassmeh, A.S. Tarawneh, M.B. Alhasanat, H.S. Eyal Salman, V.S. Prasath, Effects of distance measure choice on k-nearest neighbor classifier performance: a review, Big Data 7 (4) (2019) 221–248.
[8] F. Ridzuan, W.M.N.W. Zainon, A review on data cleansing methods for big data, Procedia Comput. Sci. 161 (2019) 731–738.
[9] J.A. Sáez, J. Luengo, F. Herrera, Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification, Pattern Recognit. (2013) 355–364.
[10] J. Komorniczak, P. Ksieniewicz, M. Woźniak, Data complexity and classification accuracy correlation in oversampling algorithms, in: Machine Learning Research, Morgan Kaufmann, 2022, pp. 175–186.
[11] P. Sun, Z. Wang, L. Jia, Z. Xu, SMOTE-kTLNN: A hybrid re-sampling method based on SMOTE and a two-layer nearest neighbor classifier, Expert Syst. Appl. 238 (2024) 121848.
[12] M. Denil, T. Trappenberg, Overlap versus imbalance, in: Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence, Canadian AI 2010, Ottawa, Canada, May 31–June 2, 2010. Proceedings 23, Springer, 2010, pp. 220–231.
[13] G. Kovács, An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets, Appl. Soft Comput. 83 (2019) 105662.
[14] I. Nekooeimehr, S.K. Lai-Yuen, Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets, Expert Syst. Appl. 46 (2016) 405–416.
[15] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, DBSMOTE: density-based synthetic minority over-sampling technique, Appl. Intell. 36 (2012) 664–684.
[16] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, J. Artificial Intelligence Res. 61 (2018) 863–905.
[17] D.A. Cieslak, N.V. Chawla, A. Striegel, Combating imbalance in network intrusion datasets, in: GrC, 2006, pp. 732–737.
[18] J.M. Johnson, T.M. Khoshgoftaar, A survey on classifying big data with label noise, ACM J. Data Inf. Qual. (2022) 1–43.
[19] Y. Huang, B. Bai, S. Zhao, K. Bai, F. Wang, Uncertainty-aware learning against label noise on imbalanced datasets, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, (6) 2022, pp. 6960–6969.
[20] E. Bisong, E. Bisong, Regularization for deep learning, in: Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, Springer, 2019, pp. 415–421.
[21] Q. Wang, Y. Ma, K. Zhao, Y. Tian, A comprehensive survey of loss functions in machine learning, Ann. Data Sci. (2020) 1–26.
[22] J. Goldberger, E. Ben-Reuven, Training deep neural-networks using a noise adaptation layer, in: International Conference on Learning Representations, 2016, pp. 1–9.
[23] J. Gao, P. Li, Z. Chen, J. Zhang, A survey on deep learning for multimodal data fusion, Neural Comput. 32 (5) (2020) 829–864.
[24] H. Song, M. Kim, D. Park, Y. Shin, J.-G. Lee, Learning from noisy labels with deep neural networks: A survey, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[25] T. Ivan, Two modifications of CNN, IEEE Trans. Syst. Man Commun. (1976) 769–772.
[26] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. SMC-2 (3) (1972) 408–421.
[27] M. Kubat, S. Matwin, et al., Addressing the curse of imbalanced training sets: one-sided selection, in: Icml, Vol. 97, Citeseer, 1997, p. 179.
[28] C. Bunkhumpornpat, K. Sinapiromsaran, DBMUTE: density-based majority under-sampling technique, Knowl. Inf. Syst. 50 (2017) 827–850.
[29] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1) (1997) 245–271, http://dx.doi.org/10.1016/S0004-3702(97)00063-5, Relevance.
[30] H. Shamsudin, U.K. Yusof, A. Jayalakshmi, M.N. Akmal Khalid, Combining oversampling and undersampling techniques for imbalanced classification: A comparative study using credit card fraudulent transaction dataset, in: 2020 IEEE 16th International Conference on Control & Automation, ICCA, 2020, pp. 803–808, http://dx.doi.org/10.1109/ICCA51439.2020.9264517.
[31] H. He, Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications, first ed., Wiley-IEEE Press, 2013.
[32] Q. Kang, X. Chen, S. Li, M. Zhou, A noise-filtered under-sampling scheme for imbalanced classification, IEEE Trans. Cybern. 47 (12) (2017) 4263–4274, http://dx.doi.org/10.1109/TCYB.2016.2606104.
[33] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[34] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 20–29.
[35] Y. Hasan, F. Amerehi, P. Healy, C. Ryan, STEM rebalance: A novel approach for tackling imbalanced datasets using SMOTE, edited nearest neighbour, and mixup, in: 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing, ICCP, IEEE, 2023, pp. 3–9.
[36] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Artificial Intelligence in Medicine: 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001 Cascais, Portugal, July 1–4, 2001, Proceedings 8, Springer, 2001, pp. 63–66.

[37] Q. Gu, Z. Cai, L. Zhu, Classification of imbalanced data sets by using the hybrid re-sampling algorithm based on isomap, in: Z. Cai, Z. Li, Z. Kang, Y. Liu (Eds.), Advances in Computation and Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 287–296.
[38] J.A. Sáez, J. Luengo, J. Stefanowski, F. Herrera, SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, Inform. Sci. 291 (2015) 184–203.
[39] Asniar, N.U. Maulidevi, K. Surendro, SMOTE-LOF for noise identification in imbalanced data classification, J. King Saud Univ. - Comput. Inf. Sci. 34 (6, Part B) (2022) 3413–3423, http://dx.doi.org/10.1016/j.jksuci.2021.01.014.
[40] H. Zhou, Z. Wu, N. Xu, H. Xiao, PDR-SMOTE: an imbalanced data processing method based on data region partition and K nearest neighbors, Int. J. Mach. Learn. Cybern. (2023) 1–16.
[41] H. Ahmad, B. Kasasbeh, B. AL-Dabaybah, E. Rawashdeh, EFN-SMOTE: An effective oversampling technique for credit card fraud detection by utilizing noise filtering and fuzzy c-means clustering, Int. J. Data Netw. Sci. 7 (3) (2023) 1025–1032.
[42] W.A. Rivera, Noise reduction a priori synthetic over-sampling for class imbalanced data sets, Inform. Sci. 408 (2017) 146–161.
[43] J. Lee, N.-r. Kim, J.-H. Lee, An over-sampling technique with rejection for imbalanced class learning, in: Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, 2015, pp. 1–6.
[44] T. Kosolwattana, C. Liu, R. Hu, S. Han, H. Chen, Y. Lin, A self-inspected adaptive SMOTE algorithm (SASMOTE) for highly imbalanced data classification in healthcare, BioData Min. 16 (1) (2023) 15.
[45] S. Wang, Z. Li, W. Chao, Q. Cao, Applying adaptive over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning, in: The 2012 International Joint Conference on Neural Networks, IJCNN, IEEE, 2012, pp. 1–8.
[46] N. Japkowicz, Concept-learning in the presence of between-class and within-class imbalances, in: Advances in Artificial Intelligence: 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI 2001 Ottawa, Canada, June 7–9, 2001 Proceedings 14, Springer, 2001, pp. 67–77.
[47] H. Han, W. Wen-Yuan, M. Bing-Huan, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, Adv. Intell. Comput. (2005) 878–887.
[48] Z. Wei, L. Zhang, L. Zhao, Minority-prediction-probability-based oversampling technique for imbalanced learning, Inform. Sci. 622 (2023) 1273–1295.
[49] T. Xia, Y. Shao, S. Xia, Y. Xiong, X. Lian, W. Ling, GBSMOTE: A robust sampling method based on granular-ball computing and SMOTE for class imbalance, in: Proceedings of the 2023 8th International Conference on Mathematics and Artificial Intelligence, 2023, pp. 19–24.
[50] K. Borowska, J. Stepaniuk, Imbalanced data classification: A novel re-sampling approach combining versatile improved SMOTE and rough sets, in: Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, September 14-16, 2016, Proceedings 15, Springer, 2016, pp. 31–42.
[51] Q. Cao, S. Wang, 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, IEEE, 2011, pp. 543–548.
[52] F. Sağlam, M.A. Cengiz, A novel SMOTE-based resampling technique trough noise detection and the boosting procedure, Expert Syst. Appl. 200 (2022) 117023.
[53] G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, Inform. Sci. 465 (2018) 1–20.
[54] K. Napierała, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Rough Sets and Current Trends in Computing: 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30, 2010. Proceedings 7, Springer, 2010, pp. 158–167.
[55] A. Fernández, S. Garcia, M. Galar, R.C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, vol. 10, Springer, 2018.
[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[57] N. Japkowicz, Class imbalances: are we focusing on the right issue, in: Workshop on Learning from Imbalanced Data Sets II, Vol. 1723, 2003, p. 63.
[58] V. García, J. Sánchez, R. Mollineda, An empirical study of the behavior of classifiers on imbalanced and overlapped data sets, in: Progress in Pattern Recognition, Image Analysis and Applications: 12th Iberoamericann Congress on Pattern Recognition, CIARP 2007, Valparaiso, Chile, November 13-16, 2007. Proceedings 12, Springer, 2007, pp. 397–406.
[59] Z. Borsos, C. Lemnaru, R. Potolea, Dealing with overlap and imbalance: a new metric and approach, Pattern Anal. Appl. 21 (2018) 381–395.
[60] S. Szeghalmy, A. Fazekas, Synthetic imbalanced data sets, 2024, URL https://github.com/szghlm/SyntImbNoisyData/tree/main/Data.
[61] P. Hart, The condensed nearest neighbor rule (corresp.), IEEE Trans. Inf. Theory 14 (3) (1968) 515–516.
[62] J. Stefanowski, S. Wilk, Selective pre-processing of imbalanced data for improving classification performance, in: Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery, 2008, pp. 283–292.
[63] W. Han, Z. Huang, S. Li, Y. Jia, Distribution-sensitive unbalanced data oversampling method for medical diagnosis, J. Med. Syst. 43 (2) (2019) 39.
[64] Z. Xu, D. Shen, Y. Kou, T. Nie, A synthetic minority oversampling technique based on gaussian mixture model filtering for imbalanced data classification, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–14.
[65] K. Cheng, C. Zhang, H. Yu, X. Yang, H. Zou, S. Gao, Grouped SMOTE with noise filtering mechanism for classifying imbalanced data, IEEE Access 7 (2019) 170668–170681.
[66] W. Xie, G. Liang, Z. Dong, B. Tan, B. Zhang, An improved oversampling algorithm based on the samples' selection strategy for classifying imbalanced data, Math. Probl. Eng. (2019) 3526539.
[67] Y. Wang, Z. Pan, J. Dong, A new two-layer nearest neighbor selection method for kNN classifier, Knowl.-Based Syst. 235 (2022) 107604.
[68] A. Fernández, S. García, M.J. del Jesus, F. Herrera, A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets, Fuzzy Sets and Systems 159 (18) (2008) 2378–2398.
[69] M. Koziarski, M. Wozniak, CCR: A combined cleaning and resampling algorithm for imbalanced data classification, Int. J. Appl. Math. Comput. Sci. 27 (4) (2017) 727–736.
[70] G. Kovács, Smote-variants: a python implementation of 85 minority oversampling techniques, Neurocomputing 366 (2019) 352–354.
[71] S. Szeghalmy, A. Fazekas, A highly adaptive oversampling approach to address the issue of data imbalance, Computers 11 (2022) 73, http://dx.doi.org/10.3390/computers11050073.
[72] G. Menardi, N. Torelli, Training and assessing classification rules with imbalanced data, Data Min. Knowl. Discov. 28 (2014) 92–122.
[73] E. Alcobaça, F. Siqueira, A. Rivolli, L.P.F. Garcia, J.T. Oliva, A.C.P.L.F. de Carvalho, MFE: Towards reproducible meta-feature extraction, J. Mach. Learn. Res. 21 (111) (2020) 1–5.
[74] PyMFE, Meta-feature description table, 2021, URL https://pymfe.readthedocs.io/en/latest/auto_pages/meta_features_description.html, Accessed: Febr 15, 2024.
[75] X. Zeng, T.R. Martinez, Distribution-balanced stratified cross-validation for accuracy estimation, J. Exp. Theoret. Artif. Intell. 12 (1) (2000) 1–12.
[76] M. Hollander, D.A. Wolfe, E. Chicken, Nonparametric Statistical Methods, Wiley, 1999.
[77] T.D. Piyadasa, K. Gunawardana, A review on oversampling techniques for solving the data imbalance problem in classification, Int. J. Adv. ICT Emerg. Regions 16 (1) (2023).
[78] S. Chen, G. Guo, L. Chen, A new over-sampling method based on cluster ensembles, in: 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, IEEE, 2010, pp. 599–604.
[79] L. Ma, S. Fan, CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests, BMC Bioinform. 18 (2017) 1–18.
[80] H. Lee, J. Kim, S. Kim, Gaussian-based SMOTE algorithm for solving skewed class distributions, Int. J. Fuzzy Logic Intell. Syst. 17 (4) (2017) 229–234.
[81] J. Wang, M. Xu, H. Wang, J. Zhang, Classification of imbalanced data by using the SMOTE algorithm and locally linear embedding, in: 2006 8th International Conference on Signal Processing, Vol. 3, IEEE, 2006.
[82] S. Gazzah, N.E.B. Amara, New oversampling approaches based on polynomial fitting for imbalanced data sets, in: 2008 the Eighth IAPR International Workshop on Document Analysis Systems, IEEE, 2008, pp. 677–684.
[83] S. Barua, M. Islam, K. Murase, ProWSyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2013, pp. 317–328.
[84] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13, Springer, 2009, pp. 475–482.
[85] J. Hu, X. He, D.-J. Yu, X.-B. Yang, J.-Y. Yang, H.-B. Shen, A new supervised over-sampling algorithm with application to protein-nucleotide binding residue prediction, PLoS One 9 (9) (2014) e107676.
[86] D. Luc, G. László, L. Gábor, A Probabilistic Theory of Pattern Recognition, Springer, 1996.