Knowledge-Based Systems 163 (2019) 693–704

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

New label noise injection methods for the evaluation of noise filters

Luís P.F. Garcia [a,b,*], Jens Lehmann [a,c], André C.P.L.F. de Carvalho [b], Ana C. Lorena [d,e]

[a] *Institute for Applied Informatics, Leipzig University, Hainstraße, 11, Leipzig, Saxony, Germany*
[b] *Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Trabalhador São-carlense Av. 400, São Carlos, São Paulo 13560-970, Brazil*
[c] *Computer Science Institute, University of Bonn, Römerstraße 164, Bonn, North Rhine-Westphalia, Germany*
[d] *Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo, Talim St. 330, São José dos Campos, São Paulo 12231-280, Brazil*
[e] *Divisão de Ciência da Computação, Instituto Tecnológico de Aeronáutica, Praça Marechal Eduardo Gomes, 50, São José dos Campos, São Paulo 12228-900, Brazil*

A R T I C L E   I N F O

Article history:
Received 5 February 2018
Received in revised form 17 September 2018
Accepted 20 September 2018
Available online 25 September 2018

Keywords:
Label noise
Noise injection
Borderline noise
Noise filters

A B S T R A C T

Noise is often present in real datasets used for training Machine Learning classifiers. Their disruptive effects in the learning process may include: increasing the complexity of the induced models, a higher processing time and a reduced predictive power in the classification of new examples. Therefore, treating noisy data in a preprocessing step is crucial for improving data quality and to reduce their harmful effects in the learning process. There are various filters using different concepts for identifying noisy examples in a dataset. Their ability in noise preprocessing is usually assessed in the identification of artificial noise injected into one or more datasets. This is performed to overcome the limitation that only a domain expert can guarantee whether a real example is indeed noisy. The most frequently used label noise injection method is the noise at random method, in which a percentage of the training examples have their labels randomly exchanged. This is carried out regardless of the characteristics and example space positions of the selected examples. This paper proposes two novel methods to inject label noise in classification datasets. These methods, based on complexity measures, can produce more challenging and realistic noisy datasets by the disturbance of the labels of critical examples situated close to the decision borders and can improve the noise filtering evaluation. An extensive experimental evaluation of different noise filters is performed using public datasets with imputed label noise and the influence of the noise injection methods are compared in both data preprocessing and classification steps.

1. Introduction

Data from real world applications frequently present inconsistencies that affect data quality, such as missing data or unknown values, noise and faults in the data acquisition process [1,2]. Data acquisition is inherently leaned to errors, even though growing efforts are made to avoid them. It is also a resource-consuming step, since at least 60% of the efforts in Data Mining are spent on data preparation, which includes data preprocessing and data transformation [3]. Some studies estimate that, even in controlled environments, at least 5% of the examples in a dataset have problems [4,5].

Although many Machine Learning (ML) techniques have internal mechanisms to deal with noisy data, such as the pruning process in Decision Trees Induction Algorithms (DTIA) [6] and the use of slack variables in Support Vector Machines (SVM) [7], the presence of noise in the training dataset can harm the induction of accurate ML models. These impairments include an increase in processing time, a higher complexity, overfitting of the induced model and a possible deterioration of its predictive performance for new data [8]. When these models are used in critical environments, they may also have security and reliability impacts [9].

There are usually two approaches to deal with noisy data that may reduce the dependency of these standard ML internal mechanisms [10]: to employ a noise-tolerant classification algorithm [11]; or to adopt a preprocessing step, also known as data cleansing algorithm [10], to identify and remove noisy examples. The use of noise-tolerant classification algorithms aims to induce robust classification models by using some information from the noisy data. The preprocessing step, on the other hand, normally involves the application of one or more Noise Filters (NF) to identify noisy data. The identified inconsistencies can be corrected or, more frequently, removed from the dataset [12]. The research carried out in this paper follows the second approach, which is to deal with noise in a preprocessing step using NF and focusing on label noise, which

[*] Corresponding author at: Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Trabalhador São-carlense Av. 400, São Carlos, São Paulo 13560-970, Brazil.
E-mail addresses: lpgarcia@icmc.usp.br (L.P.F. Garcia), jens.lehmann@cs.uni-bonn.de (J. Lehmann), andre@icmc.usp.br (A.C.P.L.F. de Carvalho), aclorena@ita.br (A.C. Lorena).

https://doi.org/10.1016/j.knosys.2018.09.031
0950-7051/© 2018 Elsevier B.V. All rights reserved.

can be regarded as the most disruptive type of noise in supervised learning.

An evident evaluation of the NFs concerns their ability to identify noisy examples. Nonetheless, it is usually impossible to guarantee whether a given example is indeed noisy without the support of a domain expert [13,14]. The evaluation by an expert tends to increase the cost and duration of the preprocessing step [15]. This limitation is reduced when artificial datasets are used or when simulated noise is systematically added into a real dataset for NF evaluation [16–18]. In most noise identification studies, the injection of simulated noise is preferred to the use of artificial datasets [18–22].

The most frequently used noise injection method randomly exchanges the labels of a percentage of the training examples. However, this method may produce unrealistic noisy cases [13,14]. Furthermore, since there is no criterion for choosing the examples whose label will be disturbed, noisy cases easy to detect can be produced. This may occur when examples located in the center of a cluster from a given class or far from the decision boundary are selected. This paper proposes two new methods for the artificial addition of label noise into datasets. In these methods, the examples whose labels will be disturbed are selected among those next to the decision border (borderline noise). The difference between the two methods is the criterion and bias adopted to estimate which are the borderline examples to be disturbed. The first method is based on the *ratio of intra/inter class Nearest Neighbor distance*, which is used for assessing the apparent complexity of a classification task in Ho and Basu [23]. The second method is based on the distance between the examples and the decision border induced by a radial kernel SVM classifier. It is based on the *minimized sum of error distance by linear programming* measure from Ho and Basu [23], but using a non-linear classifier instead. While in the original work of Ho and Basu [23] these measures are summed over all examples in the dataset, here the individual complexity of each example is assessed. Thus, the two methods evaluate the individual examples and choose those that may better represent challenging noisy examples found in real data.

This study also compares the robustness of several NFs regarding noise detection when different noise injections methods are used, for a large number of datasets and different noise levels. This allows the analysis of NF predictive performance for different artificial label noise injection methods and how they are affected by using the two proposed methods. An additional contribution is the definition of a performance threshold that can be employed to recommend when to use NFs. This threshold takes into account whether the classification performance can be improved after the dataset is preprocessed. Finally, all noise injection methods detailed in this paper were assembled into an R package named `born` (*Borderline Noise*). The `born` package is publicly available at https://github.com/lpfgarcia/born.

The rest of this paper is organized as follows. Section 2 presents an overview of NFs, whereas Section 3 describes some existent artificial label noise injection methods and details the two new proposed noise injection methods. Section 4 presents the datasets adopted and the methodology followed in the experiments, while Section 5 presents and discusses the main experimental results. Finally, Section 6 provides the main conclusions from this study and points out directions for future work.

2. Noise filters

NFs [13,20–22,24,25] are preprocessing techniques that can be applied to any given dataset to identify potentially noisy examples [10]. According to previous studies, the use of filtered data can improve the predictive performance and reduce the complexity of classification models [13,19,20,22]. NFs can use different information to detect noise, such as: neighborhood or density information [24–26]; descriptors extracted from the data [12,13]; and predictions from classification models [13,20–22,27]. Next, the NF techniques considered in this study are presented.

2.1. Ensemble based noise filters

Previous studies have shown the benefits of using ensemble-based NFs [20–22], which employ a set of classifiers for noise detection. They are based on the premise that distinct classifiers, when combined, generate a model with a lower bias and the disagreement on their predictions can be an indicative of noise presence. There are various aggregation strategies to combine the predictions from multiple classifiers for noise identification [22]. The most common are the consensus and majority voting strategies. In the first strategy, an example is considered noisy only if it is misclassified by all classifiers in the ensemble. In the second, an example is assumed noisy if it is misclassified by the majority of the classifiers in the ensemble.

In Brodley and Friedl [22], for instance, the authors combine the noise identification predictions from classifiers induced by different classification algorithms. According to the authors, the cross-validation predictions made by k-Nearest Neighbor (k-NN) [28], Decision Trees (DT) [6] and linear SVM [7] classifiers, combined through majority voting, presented the best predictive performance in noise identification. This NF will be referred here as *Static Ensemble Filter* (SEF), because the set of classifiers that compose the ensemble is previously fixed.

The robustness of SEF is increased by the *Dynamic Ensemble Filter* (DEF) technique [20]. This increase is obtained by adapting, for each dataset, the set of noise identification classifiers to be combined. The classifiers are chosen according to a criterion that considers their predictive classification performance on the training data. Afterwards, a majority voting (or consensus) aggregation strategy is used to combine the predictions from the chosen classifiers and to assess whether an example is noisy. For choosing the set of classifiers composing the ensemble, their individual cross-validation predictive performance on the training data is evaluated. The three classifiers with best predictive performance are selected.

Another recent ensemble-based NF is *High Agreement Random Forest* (HARF) [13,21], which uses RF classifiers for noise identification. HARF considers the rate of disagreement in the predictions from the individual trees in the forest to detect the noisy examples: if this rate is relatively high (70% up to 90%), the example is considered noisy; otherwise, it is labeled as clean.

The *Cross-validated Committees Filter* (CVCF) technique, proposed in Verbaeten and Assche [27], induces multiple classifiers using a single classification technique in a cross-validation strategy. Misclassified training examples are regarded as potentially noisy. The number of times an example is marked as noisy reflects its reliability. If the example is marked as noisy in most of the cross-validation rounds, CVCF classifies the example as noisy. In Verbaeten [29], a DTIA is used in a similar scheme to propose the *Decision Tree Filter* (DTF) filter. In this NF, a DT is induced from the training datasets and the instances misclassified in the corresponding test sets are removed. In Sáez et al. [19] a framework for noise detection named *Iterative Noise Filter based on the Fusion of Classifiers* (INFFC) is used to detect noisy examples. It is very similar to what is presented in Sluban et al. [13], where the information gathered from different classifiers are combined. The main difference between the papers is that the first uses an iterative process with multiple classifiers. For both of them, first, a preliminary filtering, based on predictive performance of multiple classifiers, is performed. Afterwards, the clean examples are used to detect the noisy examples in the full set of examples. Finally, the noisy examples are only removed if they exceed a noise score metric. The iterative process stops when the number of consecutive iterations achieves a maximum number of executions or the number of identified noisy examples are less than a given a fraction of the size of the original training dataset. The classifiers used in the iterative filtering process were induced by C4.5, a DT induction algorithm, k-NN and Multilayer Perceptron (MLP).

2.2. Noise filters based on data descriptors

The *Saturation Filter* (SF) was initially proposed by Gamberger and Lavrač [30] to explore the notion of training data saturation and the Occam's Razor theory. A saturated set is the smallest dataset that allows the induction of a correct and simple hypothesis able to capture all relevant information required for data representation. Thereby, the algorithm searches for examples that can be removed in order to transform an unsaturated dataset into a saturated dataset. A measure named *Complexity of the Least Correct Hypothesis* (CLCH) is used to identify these examples. SF removes α examples per iteration (α is a parameter of the technique), generating all possible combinations of saturated data. If the CLCH value decreases when a subset of examples is removed, this subset is considered noisy. In Sluban et al. [13], some efforts were made to reduce the computational burden of SF. The proposed modifications included the use of a DTIA to prune the examples that are most probably noisy before applying the SF iterations. Finally, the difference between the sizes of a pruned and an unpruned DT gives the CLCH estimate [13].

2.3. Distance based noise filters

Other popular NFs are based on the distance between examples and employ the k-NN algorithm [25,26,31]. They consider an example to be safe if it is next to other examples from its class, and noisy otherwise. They also consider that small perturbations in a borderline example can move it to the wrong side of the decision border, making it unreliable. Therefore, the distance-based NFs usually remove both noisy and borderline examples. This tends to smooth the margin of separation between different classes. The Edited Nearest Neighbor (ENN) [26] technique removes an example if the label of its k nearest neighbors is different from its own label. The *All-k-ENN* (AENN) technique applies the k-NN classifier with several increasing values of k [25]. At each iteration, examples that have the majority of their neighbors from other classes are marked as noisy.

2.4. Other noise filters

There are many other NFs in the literature [19,24,32,33]. Khoshgoftaar and Rebours [32] proposed the *Iterative Partitioning Filter* (IPF) technique, which induces DT models in an iterative process using k-fold cross-validation. The iterative process finishes when less than 1% of the data is not misclassified by the DT after the third iteration. Sáez et al. [33] combined the *Synthetic Minority Over-sampling Technique* (SMOTE) and the IPF in what was named SMOTE-IPF searches for noisy examples in imbalanced datasets. This study adopted the techniques ENN, AENN, PruneSF, CVCF, HARF, DEF and SEF, which are well-known NFs that have distinct characteristics and biases for noise identification. The R code of these NFs can be found in Morales et al. [34].

3. Artificial noise

Ideally, noise identification should involve a validation step, where the objects labeled as noisy are confirmed as such, before they can be further processed [13]. Since the most common approach is to eliminate noisy data, it is important to properly distinguish these examples from the safe data, which should be preserved.

In a real application, evaluating whether a given example is safe usually has to rely on the judgment of a domain specialist, who is not always available. Furthermore, the need to consult a specialist tends to increase the cost and duration of the preprocessing step. This problem is reduced when artificial datasets are used, or when simulated noise is added to a dataset in a controlled way [18]. The systematic addition of noise simplifies the validation of the NFs and the study of the noise influence on the learning process.

Next, the main methods for label noise injection considered in this study are presented according to the taxonomy suggested in Frenay and Verleysen [10].

3.1. Noisy at random model

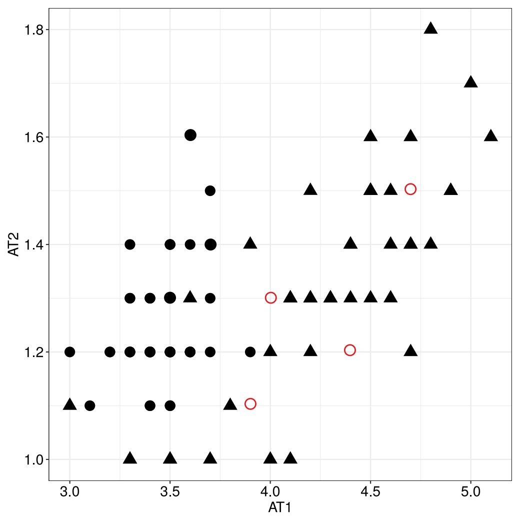
There are two main methods to inject noise in the class labels: (i) Random, when each example has the same probability of having its label exchanged by another label [16]; and (ii) Pairwise, when a percentage x% of the majority class examples have their labels flipped to the second majority class label [17]. Whatever the strategy adopted to add noise to a dataset, it is necessary to corrupt the examples within a given rate. In most of the related studies, noise is added according to rates that range from 5% to 40%, with intervals of 5% [18]. There are also studies that choose the fixed rates as 2%, 5% and 10% [13]. Besides, due to its stochastic nature, this addition is normally repeated a number of times for each noise level.

The artificial binary dataset shown in Fig. 1 illustrates the random and pairwise methods. The original dataset has 2 classes (● and ▲), which are non linearly separable, with 21 and 35 examples, respectively. Fig. 1(a) shows the same artificial dataset after the injection of 10% of label noise by the Pairwise method (from the ○ class), while Fig. 1(b) has 10% of noisy examples produced by the Random method (from both ○ and △ classes). The noisy examples are highlighted in red.

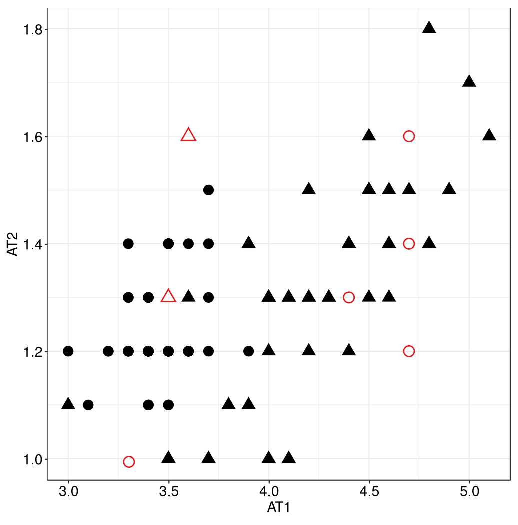
It must be observed that these methods have some limitations. Even adopting the same percentage of artificial noise, the number of examples that have their label exchanged is different in both methods. While the number of noisy cases in the Random method is defined by the number of examples in the dataset, the Pairwise method uses the number of examples in the majority class. Therefore, since they add different numbers of noisy examples into the dataset, the methods cannot be directly compared. This is illustrated by Fig. 1, in which 10% of noise is added for both methods: while for the Pairwise method (Fig. 1(a)) four examples are corrupted, the Random method changes the labels of seven examples (Fig. 1(b)). To overcome this limitation, one has to enforce that the same number of examples are corrupted by both methods, leading to the use of different noise percentage rates for the different methods. For instance, in the example from Fig. 1, the noise rate for the Pairwise method would have to be increased to 20%.

Another limitation is that, by being highly random, these methods arbitrarily select the corrupted examples and their new labels. Thus, a large number of executions is needed to reliably estimate the predictive performance of NFs. Furthermore, case studies based on real datasets analyzed by domain experts [13,14] and previous experimental studies for understanding the effects of label noise in classification tasks [24,33,35] indicate that Random noise may be easily detected. In the examples from Fig. 1, for instance, all NFs from the previous section are able to recover the noisy examples with a precision close to 90%.

A more realistic and challenging scenario should consider adding label noise to critical examples, like those located in the



(a) Pairwise method.



(b) Random method.

Fig. 1. Examples of the outputs of two artificial class label noise injection methods from the literature: (a) Pairwise; and (b) Random. A rate of 10% of noise is used in both cases . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

decision boundaries. For instance, Sluban et al. [13] performed a study on noise identification in a coronary heart disease (CHD) classification dataset in which an expert analysis of the noisy examples revealed that most of them were due to a poorly calibrated equipment, generating values slightly away from the boundary limits and leading to an incorrect labeling of the examples. In Garcia et al. [14], a similar study using a real-world ecology-related dataset revealed that most of the noisy examples detected were related with stochastic events or disorders that introduced small variations in the environmental characteristics, which in turn enabled or disabled the species presence. These were also borderline examples very difficult to detect without the analysis of a domain expert. In the same line, Sáez et al. [36] studied the effect of label noise in the domain of medical data classification. They pointed out label noise introduced by humans are more frequent in problems for which the labeling process is complex or for which there is not a consensus among multiple labelers.

In Garcia et al. [24], an experimental study to understand the effects of label noise in classification tasks also reinforce the importance of borderline examples. In this study, a set of classification complexity measures from Ho and Basu [23] and graph-based measures from Kolaczyk [37] are monitored after label noise injection. The main results indicate that measures that inspect examples close to decision boundaries are the best to explain the complexity induced by label noise imputation, reinforcing the importance of the class label reliability of borderline examples.

3.2. Noisy not at random model

There are also other methods that use extra information to add artificial label noise to the examples. In Niaf et al. [38], a label noise injection method based on the distribution of the classes adds uniform noise to the predictive attributes and changes the label based on the highest class probability. In Chhikara and McKeon [39], an exponential label noise injection method uses the distance from the examples to the classification boundary to calculate the probability of mislabeling in binary classification tasks.

Two methods for borderline noise injection are proposed in this paper to add more challenging and yet realistic artificial noise to classification datasets and to improve NF evaluation. Based on a successful use of data complexity measures in the analysis of label noisy data [24,40], the two label noise injection methods proposed are based on concepts of two of the complexity measures originally proposed by Ho and Basu [23]. The first method, named *Neighborwise*, is based on the ratio of intra/inter class Nearest Neighbor (NN) distance of each example. The second method measures the distance to a non-linear boundary induced by a SVM classifier and is named *Non Linearwise*.

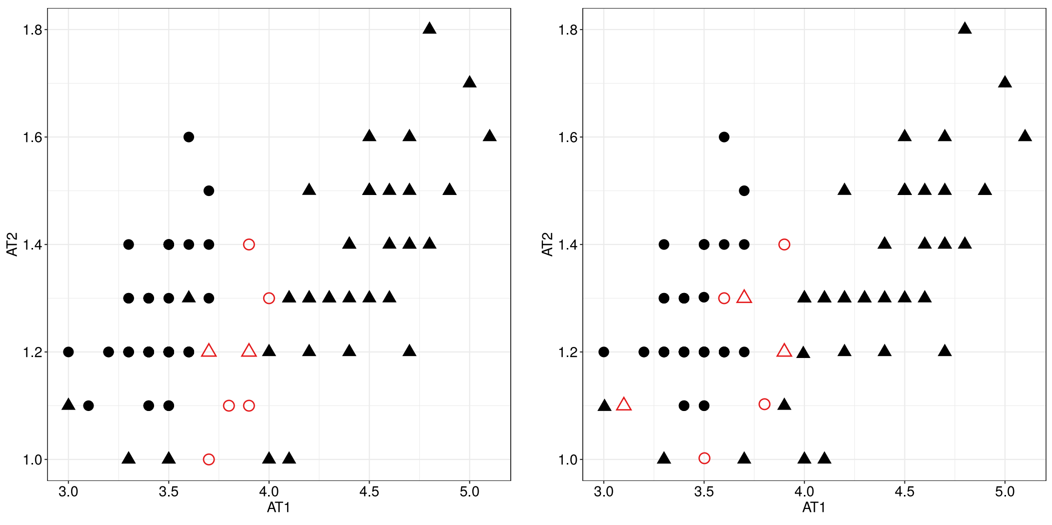
The Neighborwise noise injection method uses the ratio of the intra and inter class distances to analyze the spread of the examples from distinct classes and to determine the borderline examples. The intra-class distance is the distance between examples from same class, while the inter-class distance is the distance between examples from distinct classes. In this method, Eq. (1) is applied to each example from the training dataset T composed of n tuples $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i = (x_{i1}, \dots, x_{im})$ is an example described by m features and y_i corresponds to its class label. In this equation, d denotes the euclidean distance between an example and its NN.

$$dNN(\mathbf{x}_i) = \frac{d(\mathbf{x}_i, NN(\mathbf{x}_i) \in y_i)}{d(\mathbf{x}_i, NN(\mathbf{x}_i) \notin y_i)} \tag{1}$$

Eq. (1) gives the ratio of two distances: the euclidean distance between an example and its NN from the same class and the euclidean distance between an example and its nearest enemy, which corresponds to its NN from another class. A high dNN value is obtained for an example $\mathbf{x}_i$ far from the other examples from its class, whilst it is close to examples from another class. These are probable borderline examples and are candidate to be corrupted by the Neighborwise method. Algorithm 1 describes how this method works.

```
Input: $T$ (training data) with $n$ labeled examples in the form
  $(\mathbf{x}_i, y_i)$, and the percentage of noisy examples $p$
Output: $N$ (set of examples to be corrupted)
  $N \leftarrow \emptyset$
  for $i \leftarrow 1, \dots, n$ do
    $dNN(\mathbf{x}_i) \leftarrow \frac{d(\mathbf{x}_i, NN(\mathbf{x}_i) \in y_i)}{d(\mathbf{x}_i, NN(\mathbf{x}_i) \notin y_i)}$
  end for
  Order $T$ in descending order according to the $dNN(\mathbf{x}_i)$ values computed
  $N \leftarrow p$% first examples from $T$
  return $N$
```

Algorithm 1: Neighborwise noise injection method.



(a) Non Linearwise method.   (b) Neighborwise method.

Fig. 2. Examples of the outputs of two artificial class noise injection methods proposed: (a) Non Linearwise; and (b) Neighborwise. A rate of 10% of noise is used in both cases.

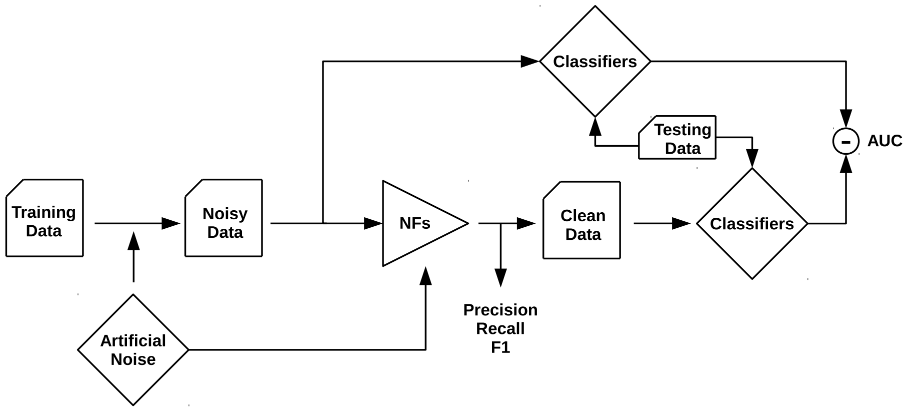


Fig. 3. Evaluation methodology followed in the experiments.

The Non Linearwise noise injection method quantifies the non linearity of a dataset by computing the distance between each example and a radial decision border separating the classes. An SVM with a radial basis (RBF) kernel function [7] is used to induce the decision border. Therefore, first a SVM classifier is induced used the training data T. For each example $\mathbf{x}_i$, its distance dNL to the decision border, as expressed by Eq. (2), is computed. In this equation, f denotes the SVM decision function obtained, and $f(\mathbf{x}_i)$ corresponds to the SVM output value (before the application of a *signal* function) for the example $\mathbf{x}_i$.

$$dNL(\mathbf{x}_i) = |f(\mathbf{x}_i)| \tag{2}$$

Low dNL values indicate that the example $\mathbf{x}_i$ is near the decision border and that the model is highly uncertain about its label. On the other hand, examples that can be easily classified and are far from the decision border will show a high dNL_i value. The examples with the lowest dNL values are those considered to be corrupted, as they are nearest to the decision border. Algorithm 2 shows the pseudocode of this method.

```
Input: $T$ (training data) with $n$ labeled examples in the form
  $(\mathbf{x}_i, y_i)$, and the percentage of noisy examples $p$
Output: $N$ (set of examples to be corrupted)
  $N \leftarrow \emptyset$
  $f \leftarrow$ train SVM classifier with RBF Kernel function on $T$
  for $i \leftarrow 1, \ldots, n$ do
    $dNL(\mathbf{x}_i) \leftarrow |f(\mathbf{x}_i)|$
  end for
  Order $T$ in ascending order according to the $dNL_i$ values
  computed
  $N \leftarrow p\%$ first examples from $T$
  return $N$
```

Algorithm 2: Non Linearwise label noise injection method.

The Non Linearwise method is limited to binary-classification datasets. Multiclass classification tasks must first be decomposed into a set of binary classification subtasks. For such, the *one-vs-one* approach is used, where one binary subtask is derived for each pair of classes [41]. To combine the results of the subtasks, the distance between the example and the nearest of the pairwise decision borders is used.

Considering the same dataset used in Figs. 1, 2(a) shows the artificial dataset corrupted with 10% of artificial Non Linearwise label noise, while Fig. 2(b) has 10% of artificial Neighborwise label noise.

The noisy examples, highlighted in red, are, for both methods, close to the decision boundary.

Comparing Figs. 1 and 2, it is possible to assert that the borderline methods produce more challenging datasets. While for the first methods a very high NF performance is expected, because some of the noisy examples are close to safe areas, for the second methods, a decrease of performance is expected, once the noisy examples are close to the decision border. In the examples from Fig. 2, for instance, the average precision of the NFs from Section 2 is lower than 35%.

It must be observed that, for both methods, if T already contains noisy examples, these examples can be selected and have their labels eventually corrected. But this is expected to be a minor problem that will not happen frequently. Other criteria could also be used to evaluate the examples and for choosing those examples to be corrupted. For instance, other complexity measures from Ho and Basu [23] which also capture the separability of the classes (N1, N3 and L2) could be employed instead. The *fraction of points on class boundary* (N1), which builds a minimum spanning tree from the data and counts the number of examples from opposing classes in the tree, is one of them. However, the measures considered here are simpler to compute and easier to adapt to allow an individual evaluation of the examples, which can be used for ranking the examples to have their labels modified. Previous studies also employed complexity measures to design more challenging artificial datasets, as Maciá and Bernadó-Mansilla [42]. Although related to this work, they were not concerned with neither the identification of examples to be disturbed by label noise injection nor the evaluation of the predictive performance of NFs.

It is worth noting that the Non Linearwise and Neighborwise methods are different from the other label noise injection methods found in the related literature. While in Teng [16] and Zhu et al. [17] the methods inject label noise randomly, the method in Niaf et al. [38] combines noise in both the predictive and the label attributes, which can be more intricate to apply to a real dataset. The exponential label noise method in Chhikara and McKeon [39] uses a Linear Discriminant Analysis (LDA) classifier to obtain the decision boundary from which the probabilities of mislabeling are calculated. Although the concept of classifier boundary used is similar to that of the methods proposed in this paper, in LDA the probabilities of misclassification have a small overall effect; moreover, the error rate of the LDA classifier is highly influenced by the original noise level of the dataset.

4. Experimental framework

This section describes the experiments carried out to evaluate the label noise imputation methods proposed in this study, for different NFs. Section 4.1 presents the datasets employed in these experiments and Section 4.2 describes the methodology followed in the experiments.

4.1. Datasets

The experiments were carried out using 142 benchmark classification datasets selected from the UCI and OpenML repositories [43,44]. Because they are benchmark datasets extracted from different application domains, it is not possible to assert that they are noiseless. Nonetheless, as pointed out in Maciá and Bernadó-Mansilla [42], most of the existent benchmark datasets used by the ML community are very simple and many ML techniques are able to attain a high predictive performance when applied to these datasets. Table 1 summarizes the main characteristics extracted from the datasets used, including dimensionality ratio [45,46], imbalance ratio (IR) [47] and the *error ratio of a Linear Classifier* (L_2) from Ho and Basu [23]. The dimensionality ratio first applies Principal Component Analysis (PCA) to a dataset and extracts the number of components d' necessary to represent 95% of the data variability. Next, it takes the ratio $\frac{d'}{n}$, where n is the number of examples in the dataset. It reflects data sparsity by considering a minimized set of uncorrelated features. The IR measure divides the number of examples in the majority class by the number of examples in the minority class, providing an overview on the imbalance present in the dataset. The third characteristic is the error rate obtained when a linear classifier (a SVM classifier using a linear Kernel) is applied to the dataset. It will be zero for linearly separable datasets.

The 142 datasets were divided into groups according to their values for the previous characteristics: from low ($d_r < 0.5$) to a medium/high dimensionality ratio ($d_r \geq 1$); from balanced ($IR = 1$), to imbalanced ($IR \geq 2$); and finally from linear ($L_2 < 5\%$) to probably non-linear classification datasets ($L_2 \geq 20\%$).

4.2. Methodology adopted

Fig. 3 summarizes the methodology employed in the experiments. Each dataset was first partitioned using 10-fold cross-validation. Afterwards, label noise was injected into examples from the training folds obtained, using the following label noise imputation methods: Random, Non Linearwise and Neighborwise, with the 5%, 10%, 20% and 40% rates. As 10 different noisy versions were generated for each method, each dataset and noise level, 17040 datasets with label noise were produced.

Noise injection was controlled to allow the recognition of the noisy examples and to evaluate the predictive performance of the NFs described in Section 2, namely HARF, SEF, DEF, CVCF, DTF, PruneSF, ENN and AENN, to identify the artificial noise injected in the noisy versions of the training datasets. Next, a filtered version of the dataset was extracted (clean data in Fig. 3). Both the noisy and filtered versions of the training datasets are used to induce classification models. Their Area Under the ROC curve (AUC) performance on the corresponding test data was then assessed.

The classifiers combined by SEF are 3-NN, DT and SVM with linear kernel function, as suggested by Brodley and Friedl [22], with outputs combined through majority voting. DEF used a majority voting strategy to combine three classifiers. The classification algorithms that can compose the DEF ensemble come from different learning paradigms: SVM [7] with linear and RBF kernel functions, Random Forest (RF) [48], k-NN [28], DT induced by the C4.5 algorithm [6] and Naïve Bayes (NB) [49]. The HARF filter considers an example as noisy if it is incorrectly classified by at least 70% of the RF, which is composed by 500 DTs. CVCF and DTF use the C4.5 algorithm. PruneSF also uses the C4.5 [6] algorithm for estimating the CLCH values. Finally, ENN uses k-NN with $k = 5$ and AENN uses k-NN with k ranging from $k = 1$ to $k = 9$. For all NFs that employ cross-validation in their internal mechanisms, 10 folds were generated.

To evaluate the noise identification performance of the NFs, the well-known precision, recall and F_β-score measures are used. Precision is the percentage of noisy cases correctly identified among those examples identified as noisy by the filter. Recall is the percentage of noisy cases correctly identified among the noisy cases injected into the dataset ($TP + FN$). The F_β-score metric combines both precision and recall values. Considering $\beta = 1$, this measure is an harmonic average, giving equal importance to both precision and recall. All measure values range from 0 to 1 and the higher the value, the better is the noise identification performance.

Filtered datasets are generated in a preprocessing step, for each systematic method of noise imputation, noise level and NF. Classifiers are induced using the preprocessed datasets in order to verify if the preprocessing step can increase the predictive performance achieved in comparison to the use of the original

Table 1
Characteristics of the benchmark datasets used in the experiments.

d_r	**IR**	L_2		
		$L_2 < 5\%$	$5\% \leq L_2 < 20\%$	$L_2 \geq 20\%$
$d < 0.5$	$IR = 1$	movement-libras, mu284, seeds, segmentation		aids, meta-data
	$1 < IR < 2$	auto_price, badges2, cardiotocography, collins, cpu, creditscore, dbworld-subjects, flags, wdbc, wine	boston, engine1, machine_cpu, wine-quality-red, australian, heart-hungarian, horse-colic-surgical, parkinsons, pollution, statlog-australian-credit, statlog-heart	asbestos, blood-transfusion-service, cloud, cpu_small, ilpd, no2, plasma_retinol, pm10, schlvote, space_ga, statlog-german-credit, strikes, wildcat, witmer_census_1980
	$IR \geq 2$		climate-simulation	
$0.5 \leq d < 1$	$IR = 1$	crabs, prnn_crabs, vowel-reduced, vowel		newton_hema, pollen
	$1 < IR < 2$	acute-nephritis, acute-urinary, breast-cancer-wisconsin, ecoli, kr-vs-kp, leukemia-haslinger, seropositive, thyroid-newthyroid, transplant, voting, vsoil, zoo	bank8FM, chscase_vine1, glass, ionosphere, vehicle, bodyfat, hutsof99_logis, lowbwt, mammographic-mass, mines-vs-rocks, puma8NH, rmftsa_ladata, spectf-heart, stock, vertebra-column-2c, wholesale-customers	baskball, bupa, delta_ailerons, diabetes, fruitfly, kidney, lupus, mbagrade, pc1_req, planning-relax, prnn_fglass, saheart, spect-heart, tae, veteran, vhamster, wholesale-channel
	$IR \geq 2$		appendicitis, fertility-diagnosis, thoracic-surgery	
$d \geq 1$	$IR = 1$	iris, molecular-promoters, molecular-promotor	prnn_synth	monks1
	$1 < IR < 2$	banknote-authentication, dermatology, japansolvent, led7digit, qualitative-bankruptcy, rmftsa_ctoarrivals, tic-tac-toe, vgalaxy	balance, boxing, elusage, flare, geyser1, pwLinear, user-knowledge, grub-damage, michiganacc, monks3, vineyard, vinnie, yeast	banana, blogger, chscase_vine2, cmc, diabetes_numeric, disclosure_z, dresses-sales, habermans-survival, hayes-roth, monks2, phoneme, popularKids, quake, rmftsa_sleepdata, sensory, titanic, venvironmental, vethanol
	$IR \geq 2$	car	backache	

data. Here the following classification algorithms with different bias are used: 3-NN [28], RF [48] and SVM [7] with a RBF kernel function. The multiclass AUC measure was used to evaluate the predictive performance of the classifiers. The difference between the AUC values obtained in the filtered datasets and in their noisy counterparts was also measured [50].

To evaluate the statistical significance of the experimental results, the Friedman and the Wilcoxon signed-rank statistical tests [51] with 95% of confidence level were applied to compare the predictive performances of the different NFs and classifiers. The datasets used, together with the experimental results, are available at https://lpfgarcia.github.io/borderline/.

5. Experimental results

This section presents the results obtained in the experiments carried out in this study. Section 5.1 reports the ranking, the precision–recall space and the average F_1 values for each NF. Section 5.2 reports the predictive performance of the classifiers in the preprocessed and original datasets.

5.1. NF performance per noise level

Fig. 4 summarizes the F_1-based predictive performance of the NFs for the three different methods of artificial noise imputation. It shows the average ranking of each filter per noise level, regarding its F_1-based predictive performance for all datasets. The x-axis represents the noise levels. The y-axis shows the ranking value. The NF with the best F_1 value in noise identification will have the lowest average ranking value. HARF performance is shown by a line with black circles, CVCF performance is represented by red triangles, SEF by blue crosses, DEF by green multiplication signs, DTF by purple lozenges, AENN by orange upside down triangles, ENN by yellow hollow squares and PruneSF by brown asterisks.

As can be seen in Fig. 4, the ranking of the NFs regarding their predictive performance changes according to the noise injection method used. For the Random method, HARF was the NF with the best performance for 5% of noise level, with statistical significance according to the Friedman test and Nemenyi post-test at 95% of confidence level. DEF was the best filter for 10% and 20% of noise levels, except when statistically compared with HARF. SEF was superior to all others for 40% of noise level, although its F_1 values were statistically similar to those obtained by DEF. The Non Linearwise method had SEF as the best NF for all noise levels, except when statistically compared with DTF for all noise levels and to DEF for low noise levels (5% and 10%). The NFs with the best performance for the Neighborwise method were: (*i*) DEF, for 5% and 10% of noise levels, except when statistically compared with SEF for both noise levels and HARF for the lowest noise level; and (*ii*) SEF for 20% and 40% of noise levels, except when compared with DEF at 20% of noise level. The NFs with worst ranking positions were: DTF and ENN, for the random method; CVCF and ENN for the borderline methods. It is worth noting that, despite also being used for filtering borderline cases, ENN had a low performance in the identification of the borderline noisy examples.

Therefore, DEF and SEF obtained the best performance in borderline noise identification. It must be observed that the ranking position of these NFs is stable for the different noise levels introduced, once they are always among the top-three best performing NFs. DTF also performed well in the borderline noise identification. On the other hand, ENN, CVCF, AENN and PruneSF presented a poor performance in the identification of borderline noise. The results for the Random method presented a high variation. Whilst HARF is very efficient for low Random noise levels, the same was not true for the highest noise levels. DEF maintained a consistent performance for Random noise. The NFs showed a mixed behavior, sometimes increasing, other times decreasing their noise detection ability as higher levels of Random noise are injected. This may be a consequence of the mixed presence of easy and difficult noisy examples in the Random datasets.

Based on the previous observations, it is possible to state that DEF was among the best performing NFs in different scenarios. DEF is an heterogeneous ensemble that adapts the classifiers to be combined through a simple strategy.

Fig. 5 plots the results of the best NF for all datasets in the precision–recall space. Each plot compares two noise injection methods. Each row of Fig. 5 corresponds to a particular noise level. The results obtained in the datasets corrupted by the Random method are shown by black circles, the Non Linearwise method results are represented by red triangles and the Neighborwise method results are represented by blue crosses. In each plot, the x-axis represents the precision and the y-axis represents the recall. The dashed lines show F_1 isolines from 0.1 to 0.9 with intervals of 0.1, i.e., points for which the corresponding F_1 values are attained. The use of isolines in NF evaluation was introduced in Sluban et al. [13]. The best results are situated near to the top-right of the plots.

It is possible to notice in Fig. 5 different patterns of spreading of points in the precision–recall space. The Random method presented the lowest spreading. The results for the Random method also tend to be closer to the top-right of the plots, evidencing that the introduced noise may be easier to identify in this case. Besides, when noise is added through the Random method, the recall of the NFs tends to be higher than the precision (most of the noisy examples introduced are correctly identified, but safe cases can also be identified as noisy). Another interesting result about the Random method is that, for all datasets, as the noise level increases, the spreading of points decreases and concentrates when the F_1 performance is higher than 0.4. This can be due to an increase of NF evaluation performance in these cases. Both borderline methods are able to obtain a more diverse spreading of precision–recall values than the Random method.

Regarding the Neighborwise method, the precision and recall had the same importance for low noise levels. For high noise levels, precision was superior to recall. Thus, most of the correctly identified noisy examples are indeed noisy, but the predictive performance of the NFs in the identification of all noisy examples is reduced. The spreading of the points also changed, becoming more concentrated for high noise levels.

The application of the Non Linearwise method also resulted in a large spreading of the precision–recall values. However, the NFs usually gave the same importance to precision and recall. For high noise levels, the spreading of points is still large, but precision becomes more important than recall, as in the Neighborwise method.

In a comparison of the results from borderline methods, the Non Linearwise method results in a larger spreading of the points than the Neighborwise method. The increase of noise level also presented a lower impact in the Non Linearwise method. As consequence, the Non Linearwise method was able to produce more challenging noisy datasets.

5.2. Performance in the classification step

Fig. 6 summarizes the average differences of AUC performance in the classification step using the preprocessed datasets in relation with the AUC obtained for the noisy datasets. It presents the AUC performance for the k-NN, RF and SVM algorithms for each dataset. The Random method is shown in black, the Non Linearwise method in red and the Neighborwise method in blue. The x-axis shows the accumulated AUC values (for 40 executions, that is, four noise levels x 10 cross-validation runs) and the y-axis shows the datasets. The datasets are ordered according to the average F_1 values obtained by the NFs. The horizontal dotted lines represent

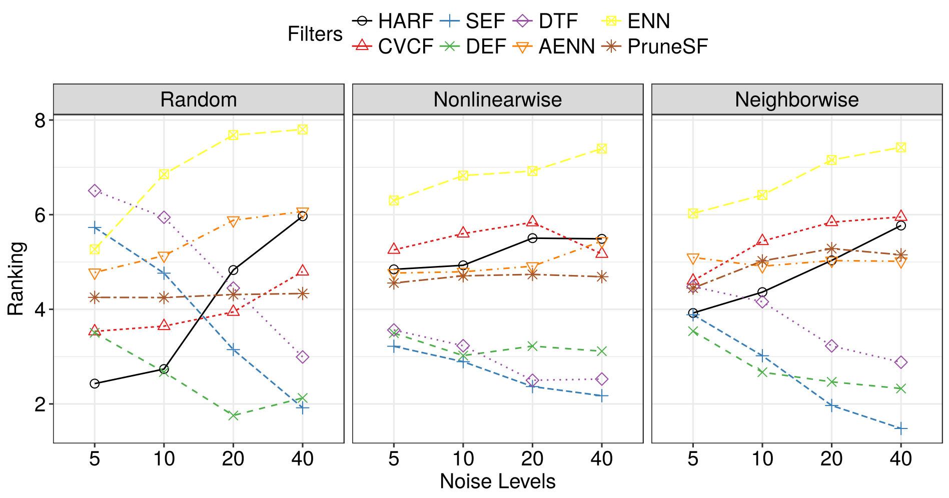


Fig. 4. Ranking of the NFs according to the F_1-based predictive performance in noise identification. The position of each NF in the ranking corresponds to the average obtained for all datasets in the corresponding noise level.

Table 2
Average accumulated AUC differences between preprocessed and noisy datasets.

F_1	Classifier	Random	Nonlinearwise	Neighborwise
$0.9 < F_1 \leq 1.0$	k-NN	15.25	**15.68**	6.08
	SVM	**5.70**	1.90	5.17
	RF	6.44	**9.49**	3.91
$0.8 < F_1 \leq 0.9$	k-NN	2.29*	**6.40***	0.64
	SVM	4.98*	**6.19***	1.58*
	RF	4.03*	**4.10***	3.27*
$0.7 < F_1 \leq 0.8$	k-NN	**16.58***	−2.21	−0.71
	SVM	**5.95***	−1.18	−0.68
	RF	**7.95***	1.01	0.77
$0.6 < F_1 \leq 0.7$	k-NN	**19.43***	2.71	−0.83
	SVM	**2.51***	1.74	0.62
	RF	**7.52***	6.67*	1.88^{+}
$0.5 < F_1 \leq 0.6$	k-NN	**15.25***	−1.90	−6.60*
	SVM	**4.39**$^{+}$	0.64	0.90
	RF	**6.40***	4.59	−1.40*

limits on the F_1 performance. Only datasets for which the average F_1 performance in noise identification was higher than 0.5 are shown. In this figure, positive values (right handed bars) indicate an increase of predictive performance when the datasets are preprocessed compared to the use of the original datasets.

According to these results, k-NN and RF benefited more from the use of NFs. The SVM classifier showed a small increase of performance when NFs were used. This suggests that while k-NN and RF may be favored by the filtering process, SVM has robust mechanisms that are already able to deal with Random, Non Linearwise or Neighborwise noise. For specific datasets, like *creditscore*, *cpu*, *seroposive*, *breast-cancer-wisconsin*, *auto_price*, *cpu_small* and *machine_cpu*, the predictive performance of the classifiers was decreased for the Neighborwise method. For the *cpu*, *seroposive*, *breast-cancer-wisconsin*, *machine_cpu* and *movement-libras* datasets, the classification performance was decreased when using the Non Linearwise method. Most of these datasets are linearly separable, according to Table 1, and have an intermediate IR. Therefore, the Non Linearwise noise addition may have a low impact or has even been inadequate for these datasets.

Comparing the performance between the noise injection methods, the Random method presented the highest increase of performance, mainly for the k-NN classifier. This was expected, once its noise addition mechanism is naive and unable to evaluate the NFs properly. Another explanation for these results is that k-NN is a local classifier and the Random method spreads noisy examples more uniformly in different neighborhoods. For the same reason, the k-NN classifier presented a low performance for the Neighborwise noise addition method. Most of the noise added in this case directly impairs the k-NN decisions, since both are based on neighborhood information. The use of the Non Linearwise method decreased the predictive performance for few datasets. Most of the decreases were observed for datasets with low F_1 performance and for the SVM classifier. It is important to note that this method has a strong bias with the SVM classifier, once it uses the same classifier to identify the borderline examples to be disturbed.

Table 2 summarizes Fig. 6, presenting the average difference of AUC values for each F_1 interval, classification algorithm and noise injection method. Positive values represent a performance increase in the classification step after a NF is applied. Negative values indicate a decrease of classification performance after noise filtering. The highest values are highlighted in boldface. Asterisks and the plus signs represent statistically significance differences, according to the Wilcoxon signed-rank statistical test at 95% and 90% of confidence levels, respectively.

When noise was added using the Random method, all classification algorithms benefited from the use of NFs. For the Non Linearwise and Neighborwise methods, an increase of classification performance is more evident when the NF shows a high noise identification performance ($F_1 \geq 0.8$). Although the classification performance increase was smaller for borderline methods, the Non Linearwise method obtained the best results for almost all classifiers. These results reinforce the findings that Random noise may be simpler to identify and that NFs can be better evaluated by

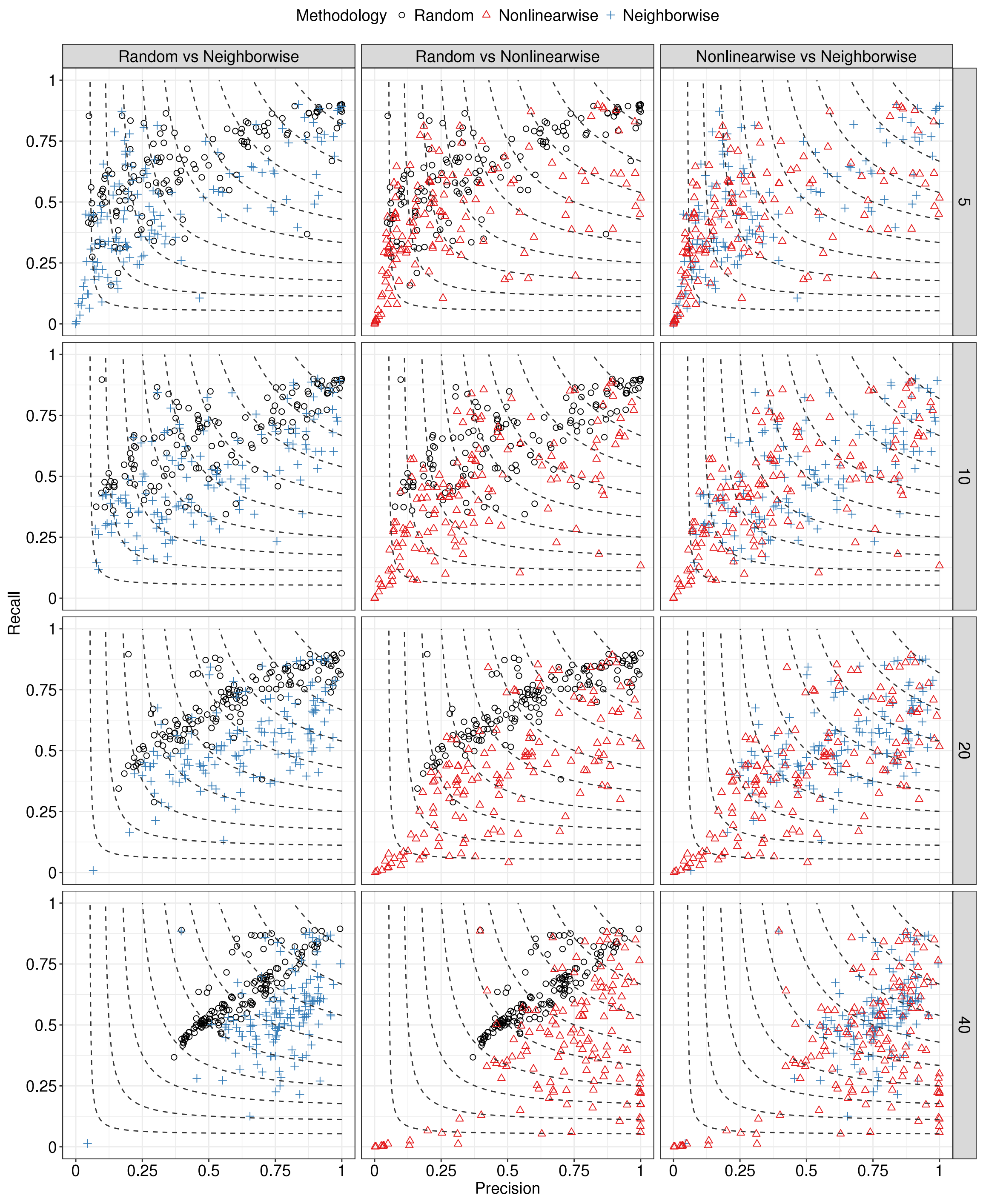


Fig. 5. Performance of the best NF for all datasets in the precision–recall space. The F_1 predictive performance is represented by isoline curves in the precision–recall space.

using borderline noise. In fact, the low performance presented by the NFs in these situations reinforce the need of new approaches in the design of NFs.

6. Conclusion

This paper proposed borderline methods to produce more challenging and realistic noisy datasets, aiming to improve NF evaluation. Several benchmark public datasets, different NFs and varied levels of noise were employed in experiments that evaluated the influence of the noise injection methods in the preprocessing and classification steps.

The experimental results show that the Random method was not able to properly evaluate the NFs and that the borderline methods produce more challenging and realistic noisy cases for the investigated datasets. The borderline methods were able to produce results spanning a larger area of the precision–recall space when compared to the Random method. Another interesting result was the change in the NF ranking. While HARF and DEF were the best NFs for the Random method, SEF and DEF were the best for the borderline methods.

Particularly in the classification step, although the performance increased after the removal of the noise injected by the Random method, using the Non Linearwise method, the increase of classification performance is higher when the NFs obtain high F_1 scores in

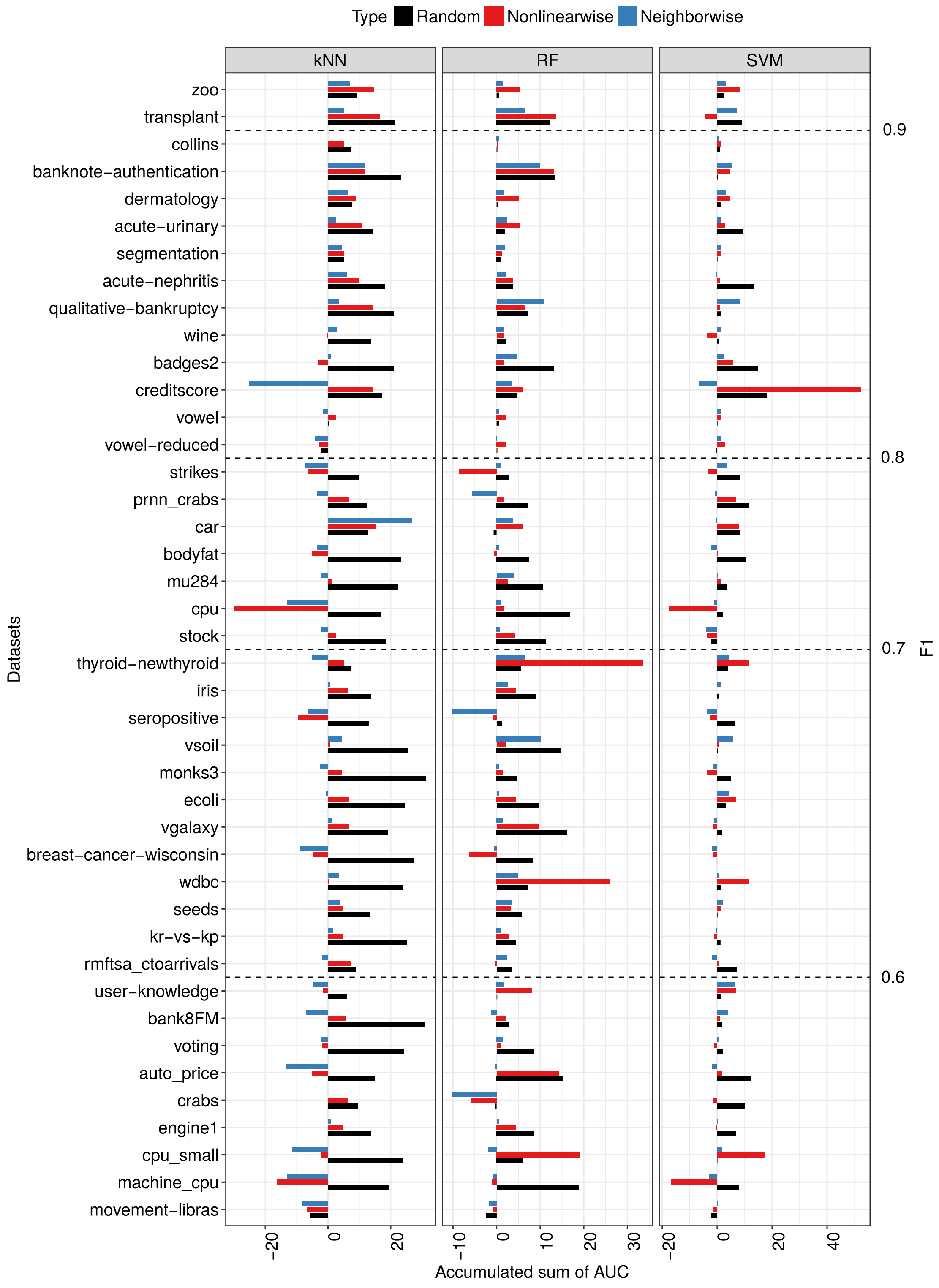


Fig. 6. Difference of AUC values obtained by the classifiers on the preprocessed datasets in relation to the AUC values obtained for the noisy datasets. Only datasets for which the NF F_1 performance is higher than 0.5 are shown.

noise identification. Therefore, even though the noise injected by borderline methods can be more complex, the preprocessing step can still increase the predictive performance of the classification models after removing this type of noise, mainly for the Non Linearwise method. Good results were also obtained in the identification of the Neighborwise noise, but with a lower predictive performance. This reinforces that the RBF function draws accurate decision boundaries or gives more importance to specific areas of the dataset, while the intra/inter distance function may result in a poor predictive performance in the identification of the intrinsic noise already present in a dataset.

As future work, the authors would like to evaluate other noise injection methods based on different measures, such as the hardness measures proposed by Smith et al. [11]. These measures quantify instance hardness through the misclassification obtained by a set of distinct classification algorithms. The authors would also like to improve the predictive performance of the NFs with a proper hyper-parameter tuning and to find out the influence of intrinsic dataset noise level in the predictive performance of NFs.

Acknowledgments

The authors would like to thank CNPq, Brazil (processes 305291/2017-3 and 152098/2016-0), FAPESP, Brazil (processes 2016/18615-0, 2013/07375-0 and 2012/22608-8) and CAPES, Brazil for their financial support. The first and third authors would like to thank CeMEAI-FAPESP for the computational resources and Intel for the hardware and software server used in part of the experiments.

References

[1] R.Y. Wang, V.C. Storey, C.P. Firth, A framework for analysis of data quality research, IEEE Trans. Knowl. Data Eng. 7 (4) (1995) 623–640.
[2] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Knowledge discovery and data mining: towards a unifying framework, in: 2nd International Conference on Knowledge Discovery and Data Mining, SIGKDD, 1996, pp. 82–88.
[3] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann, 1999.
[4] X. Wu, Knowledge Acquisition from Databases, in: Tutorial Monographs in Artificial Intelligence, Greenwood, 1995.
[5] J.I. Maletic, A. Marcus, Data cleansing: beyond integrity analysis, in: Information Quality, IQ, 2000, pp. 200–209.
[6] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[7] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.
[8] A.C. Lorena, A.C.P.L.F. de Carvalho, Evaluation of noise reduction techniques in the splice junction recognition problem, Genet. Mol. Biol. 27 (4) (2004) 665–672.
[9] D.M. Strong, Y.W. Lee, R.Y. Wang, Data quality in context, Commun. ACM 40 (5) (1997) 103–110.
[10] B. Frenay, M. Verleysen, Classification in the presence of label noise: a survey, Neural Netw. Learn. Syst. IEEE Trans. 25 (5) (2014) 845–869.
[11] M.R. Smith, T. Martinez, C. Giraud-Carrier, An instance level analysis of data complexity, Mach. Learn. 95 (2) (2014) 225–256.
[12] D. Gamberger, N. Lavrač, C. Groselj, Experiments with noise filtering in a medical domain, in: 16th International Conference on Machine Learning, ICML, 1999, pp. 143–151.
[13] B. Sluban, D. Gamberger, N. Lavrač, Ensemble-based noise detection: noise ranking and visual performance evaluation, Data Min. Knowl. Discov. 28 (2) (2014) 265–303.
[14] L.P.F. Garcia, A.C. Lorena, A.C.P.L.F. de Carvalho, Ensembles of label noise filters: a ranking approach, Data Min. Knowl. Discov. 30 (5) (2016) 1192–1216.
[15] X. Wu, X. Zhu, Mining with noise knowledge: error-aware data mining, IEEE Trans. Syst. Man Cybern. A 38 (4) (2008) 917–932.
[16] C.M. Teng, Correcting noisy data, in: 16th International Conference on Machine Learning, ICML, 1999, pp. 239–248.
[17] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, in: 20th International Conference on Machine Learning, ICML, 2003, pp. 920–927.
[18] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artif. Intell. Rev. 22 (3) (2004) 177–210.
[19] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, INFFC: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control, Inform. Fusion 27 (2016) 19–32.
[20] L.P.F. Garcia, A.C. Lorena, A.C.P.L.F. de Carvalho, A study on class noise detection and elimination, in: Brazilian Symposium on Neural Networks, SBRN, 2012, pp. 13–18.
[21] B. Sluban, D. Gamberger, N. Lavrač, Advances in class noise detection, in: 19th European Conference on Artificial Intelligence, ECAI, 2010, pp. 1105–1106.
[22] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, J. Artificial Intelligence Res. 11 (1999) 131–167.
[23] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 289–300.
[24] L.P.F. Garcia, A.C.P.L.F. de Carvalho, A.C. Lorena, Effect of label noise in the complexity of classification problems, Neurocomputing 160 (2015) 108–119.
[25] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Sci. Cybern. 6 (6) (1976) 448–452.
[26] D.L. Wilson, Asymtoptic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Sci. Cybern. 2 (3) (1972) 408–421.
[27] S. Verbaeten, A.V. Assche, Ensemble methods for noise elimination in classification problems, in: 4th International Workshop Multiple Classifier Systems, 2003, pp. 317–325.
[28] T.M. Mitchell, Machine Learning, in: McGraw Hill Series in Computer Science, McGraw Hill, 1997.
[29] S. Verbaeten, Identifying mislabeled training examples in ILP classification problems, in: 12th Dutch-Belgian Conference on Machine Learning, 2002, pp. 1–8.
[30] D. Gamberger, N. Lavrač, Conditions for Occam's razor applicability and noise elimination, in: 9th European Conference on Machine Learning, ECML, 1997, pp. 108–123.
[31] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Mach. Learn. 38 (3) (2000) 257–286.
[32] T. Khoshgoftaar, P. Rebours, Generating multiple noise elimination filters with the ensemble-partitioning filter, in: IEEE International Conference on Information Reuse and Integration, 2004, pp. 369–375.
[33] 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.
[34] P. Morales, J. Luengo, L.P.F. Garcia, A.C. Lorena, A.C.P.L.F. de Carvalho, F. Herrera, NoiseFiltersR: Label Noise Filters for Data Preprocessing in Classification, 2016. URL https://CRAN.R-project.org/package=NoiseFiltersR.
[35] L.P.F. Garcia, J.A. Sáez, J. Luengo, A.C. Lorena, A.C.P.L.F. de Carvalho, F. Herrera, Using the one-vs-one decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems, Knowl.-Based Syst. 90 (2015) 153–164.
[36] J.A. Sáez, B. Krawczyk, M. Woźniak, On the influence of class noise in medical data classification: treatment using noise filtering methods, Appl. Artif. Intell. 30 (6) (2016) 590–609.
[37] E.D. Kolaczyk, Statistical Analysis of Network Data: Methods and Models, in: Springer Series in Statistics, Springer, 2009.
[38] E. Niaf, R. Flamary, C. Lartizien, S. Canu, Handling uncertainties in svm classification, in: IEEE Statistical Signal Processing Workshop, SSP, 2011, pp. 757–760.
[39] R.S. Chhikara, J. McKeon, Linear discriminant analysis with misallocation in training samples, J. Amer. Statist. Assoc. 79 (388) (1984) 899–906.
[40] L.P.F. Garcia, A.C.P.L.F. de Carvalho, A.C. Lorena, Noise detection in the meta-learning level, Neurocomputing 176 (2016) 14–25.
[41] A.C. Lorena, A.C. De Carvalho, J.M. Gama, A review on the combination of binary classifiers in multiclass problems, Artif. Intell. Rev. 30 (1–4) (2008) 19.
[42] N. Maciá, E. Bernadó-Mansilla, Towards uci+: A Mindful Repository Design, Inform. Sci. 261 (2014) 237–262.
[43] M. Lichman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2013. http://archive.ics.uci.edu/ml.
[44] J. Vanschoren, J.N.V. Rijn, B. Bischl, L. Torgo, OpenML: Networked Science in Machine Learning, ACM SIGKDD Explor. 15 (2) (2013) 49–60.
[45] C. Walt, E. Barnard, Measures for the characterisation of pattern-recognition data sets, in: 18th Annual Symposium of the Pattern Recognition Association of South Africa, PRASA, 2007, pp. 1–6.
[46] A.C. Lorena, I.G. Costa, N. Spolaôr, M.C.P. de Souto, Analysis of complexity indices for classification problems: cancer gene expression data, Neurocomputing 75 (1) (2012) 33–42.
[47] A. Tanwani, M. Farooq, Classification potential vs. classification accuracy: a comprehensive study of evolutionary algorithms with biomedical datasets, in: Learning Classifier Systems, 2010, pp. 127–144.
[48] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[49] D.D. Lewis, Naive (bayes) at forty: the independence assumption in information retrieval, in: 10th European Conference on Machine Learning, ECML, 1998, pp. 4–15.
[50] D.J. Hand, R.J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Mach. Learn. 45 (2) (2001) 171–186.
[51] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.