The NoiseFiltersR Package: Label Noise Preprocessing in R

by Pablo Morales, Julián Luengo, Luís P.F. Garcia, Ana C. Lorena, André C.P.L.F. de Carvalho and Francisco Herrera

Abstract In Data Mining, the value of extracted knowledge is directly related to the quality of the used data. This makes data preprocessing one of the most important steps in the knowledge discovery process. A common problem affecting data quality is the presence of noise. A training set with label noise can reduce the predictive performance of classification learning techniques and increase the overfitting of classification models. In this work we present the **NoiseFiltersR** package. It contains the first extensive R implementation of classical and state-of-the-art label noise filters, which are the most common techniques for preprocessing label noise. The algorithms used for the implementation of the label noise filters are appropriately documented and referenced. They can be called in a R-user-friendly manner, and their results are unified by means of the `"filter"` class, which also benefits from adapted `print` and `summary` methods.

Introduction

In the last years, the large quantity of data from many different nature and sources has created several challenges in the Data Mining area. Not only their size, but their imperfections and varied formats are providing the researchers with plenty of new scenarios to be addressed. Consequently, Data Preprocessing (García et al., 2015) has become an important part of the *KDD (Knowledge Discovery from Databases)* process, and related software development is also essential to provide practitioners with the adequate tools.

Data Preprocessing intends to appropriately process the collected data so that subsequent learning algorithms can not only extract meaningful and relevant knowledge from the data, but also induce models with high predictive or descriptive performance. Data preprocessing is known as one of the most time-consuming steps in the whole KDD process. There exist several aspects involved in data preprocessing, like *feature selection*, dealing with *missing values* and detecting *noisy data*. Feature selection aims at extracting the most relevant attributes for the learning step, thus reducing the complexity of models and the computing time taken for their induction. The treatment of missing values is also essential to keep as much information as possible in the preprocessed dataset. Finally, noisy data refers to values that are either incorrect or clearly far from the general underlying data distribution.

All these tasks have associated software available. For instance, the KEEL tool (Alcalá et al., 2010) contains a broad collection of data preprocessing algorithms, which covers all the aforementioned topics. There exist many other general-purpose Data Mining software with data preprocessing functionalities, like WEKA (Witten and Frank, 2005), KNIME, RapidMiner, Python, or R.

Regarding the R statistical software, there are plenty of packages available in the *Comprehensive R Archive Network (CRAN)* repository to address preprocessing tasks. For example, **MICE** (van Buuren and Groothuis-Oudshoorn, 2011) and **Amelia** (Honaker et al., 2011) are very popular packages for handling missing values, whereas **caret** (Kuhn, 2008) or **FSelector** (Romanski and Kotthoff, 2014) provide a wide range of techniques for feature selection. There are also general-purpose packages for decting outliers and anomalies, like **mvoutlier** (Filzmoser and Gschwandtner, 2015). If we examine software in CRAN developed to tackle label noise, there already exist *non-preprocessing* packages that provide label noise robust classifiers. For instance, **robustDA** implements a robust mixture discriminant analysis Bouveyron and Girard (2009), while **probFDA** package provides a probabilistic Fisher discriminant analysis related to the seminal work in Lawrence and Schölkopf (2001).

However, to the best of our knowledge, CRAN lacks an extensive collection of label noise preprocessing algorithms for classification (Sáez et al., 2016; Garcia et al., 2015), some of which are among the most influential preprocessing techniques (García et al., 2016). This is the gap we intend to fill with the release of the **NoiseFiltersR** package, whose taxonomy is inspired on the recent survey on label noise by B. Frénay and M. Verleysen (Frénay and Verleysen, 2014). Yet, it should be noted that there are other packages that include some isolated implementations of label noise filters, since they are sometimes needed as auxiliary functions. This is the case of the **unbalanced** (Pozzolo et al., 2015) package, which deals with imbalanced classification. It contains basic versions of classical filters, such as *Tomek-Links* (Tomek, 1976) or *ENN* (Wilson, 1972), which are tipically applied after oversampling an imbalanced dataset (which is the main purpose of the **unbalanced** package).

In the following section we briefly introduce the problem of classification with label noise, as

well as the most popular techniques to overcome this problem. Then, we show how to use the **NoiseFiltersR** package to apply these techniques in a unified and R-user-friendly manner. Finally, we present a general overview of this work and potential extensions.

Label noise preprocessing

Data collection and preparation processes are usually subject to errors in Data Mining applications Wu and Zhu (2008). Consequently, real-world datasets are commonly affected by imperfections or *noise*. In a classification problem, several effects of this noise can be observed by analyzing its spatial characteristics: noise may create small clusters of instances of a particular class in the instance space corresponding to another class, displace or remove instances located in key areas within a concrete class, or disrupt the boundaries of the classes resulting in an increased boundaries overlap. All these imperfections may harm interpretation of data, the design, size, building time, interpretability and accuracy of models, as well as the making of decisions Zhong et al. (2004); Zhu and Wu (2004).

In order to alleviate the effects of noise, we need first to identify and quantify the components of the data that can be affected. As described by Wang et al. Wang et al. (1995), from the large number of components that comprise a dataset, class labels and attribute values are two essential elements in classification datasets Wu (1996). Thus, two types of noise are commonly differentiated in the literature Zhu and Wu (2004); Wu (1996):

- *Label noise*, also known as *class noise*, takes place when an example is wrongly labeled. Several causes may induce label noise, including subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each instance. Label noise includes contradictory examples Hernández and Stolfo (1998) (examples with identical input attribute values having different class labels) and misclassifications Zhu and Wu (2004) (examples which are incorrectly labeled). Since detecting contradictory examples is easier than identifying misclassifications Zhu and Wu (2004), most of the literature is focused on the study of misclassifications, and the term *label noise* usually refers to this type of noise Teng (1999); Sáez et al. (2014).
- *Attribute noise* refers to corruptions in the values of the input attributes. It includes erroneous attribute values, missing values and incomplete attributes or "do not care" values. Missing values are usually considered independently in the literature, so *attribute noise* is mainly used for erroneous values Zhu and Wu (2004).

The **NoiseFiltersR** package (and the rest of this manuscript) focuses on *label noise*, which is known to be the most disruptive one, since label quality is essential for the classifier training Zhu and Wu (2004). In Frénay and Verleysen (2014) the mechanisms that generate label noise are examined, relating them to the appropriate treatment procedures that can be safely applied. In the specialized literature there exist two main approaches to deal with label noise, and both are surveyed in Frénay and Verleysen (2014):

- On the one hand, *algorithm level* approaches attempt to create robust classification algorithms that are little influenced by the presence of noise. This includes approaches where existing algorithms are modified to cope with label noise by either modeling it in the classifier construction Lawrence and Schölkopf (2001); Li et al. (2007), by applying pruning strategies to avoid overfiting as in Quinlan (2014) or by diminishing the importance of noisy instances with respect to clean ones Miao et al. (2016). There exist recent proposals that combine these two approaches, which model the noise and give less relevance to potentially noisy instances in the classifier building process Bouveyron and Girard (2009).
- On the other hand, *data level* approaches (also called *filters*) try to develop strategies to cleanse the dataset as a previous step to the fit of the classifier, by either creating ensembles of classifiers Brodley and Friedl (1999), iteratively filtering noisy instances Khoshgoftaar and Rebours (2007), computing metrics on the data or even hybrid approaches that combine several of these strategies.

The **NoiseFiltersR** package follows the data level approaches, since this allows carrying out the data preprocessing just once and apply any classifier thereafter, whereas algorithm level approaches are specific for each classification algorithm[1]. Regarding data-level handling of label noise, we take the aforementioned survey by Frénay and Verleysen (2014) as the basis for our **NoiseFiltersR** package. That work provides an overview and references for the most popular classical and state-of-the-art filters, which are organized and classified taking into account several aspects:

[1]Of course, in R there exist implementations of very popular label noise robust classifiers (the aforementioned algorithm-level approach), such as *C4.5* and *RIPPER*, which are called *J48* and *JRip* respectively in **RWeka** package Hornik et al. (2009) (which is a R interface to WEKA software Witten and Frank (2005)), or the method described in Bouveyron and Girard (2009) in its own package.

- Considering how noisy instances are identified, *ensemble* based, *similarity* based and *data complexity* based algorithms are distinguished. The first type makes use of predictions from ensembles of classifiers built over different partitions or resamples of training data. The second is based on label distribution from the nearest neighbors of each instance. And the third attempts to reduce complexity metrics which are related to the presence of noise. As we will explain in the next section (see Table 1), the **NoiseFiltersR** package contains implementations of all these types of algorithms, and the explicit distinction is indicated in the documentation page of each function.
- Regarding how to deal with the identified noise, *noise removal* and *noise reparation* strategies are considered. The first option removes the noisy instances, whereas the second relabels these instances with the most likely label on the basis of the available information. There also exist *hybrid* approaches, which only carry out relabelling when they have enough confidence on the new label. Otherwise, they remove the noisy instance. The discussion between noise removal, noise reparation and their possible sinergies is an active and open field of research (Frénay and Verleysen, 2014, Section VI.H): most works agree on the potential damages of incorrect relabeling (Miranda et al., 2009), although other studies also point out the dangers of removing too many instances and advocate hybrid approaches (Teng, 2005). As we will see in the next section, the **NoiseFiltersR** package includes filters which implement all these possibilities, and the specific behaviour is explicitly indicated in the documentation page of the corresponding function.

The NoiseFiltersR package

The released package implements, documents, explains and provides references for a broad collection of label noise filters surveyed in (Frénay and Verleysen, 2014). To the best of our knowledge, it is the first comprehensive review and implementation of this topic for R, which has become an essential tool in Data Mining in the last years.

Namely, the **NoiseFiltersR** package includes a total of 30 filters, which were published in 24 research papers. Each one of these papers is referenced in the corresponding filter documentation page, as shown in the next *Documentation* section (and particularly in Figure 1). Regarding the noise detection strategy, 13 of them are ensemble based filters, 14 can be cataloged as similarity based, and the other 3 are based on data complexity measures. Taking into account the noise handling approach, 4 of them integrate the possibility of relabelling, whereas the other 26 only allow for removing (which clearly evidences a general preference for data removal in the literature). The full list of implemented filters and its distribution according to the two aforementioned criterions is displayed in Table 1, which provides a general overview of the package.

Table 1: Names and taxonomy of available filters in the NoiseFiltersR package. Every filter is appropriately referenced in its documentation page, where the original paper is provided.

		Noise Identification		
		Ensemble	*Similarity*	*Data Complexity*
Noise Handling	*Remove*	C45robustFilter C45votingFilter C45iteratedVotingFilter CVCF dynamicCF edgeBoostFilter EF HARF INFFC IPF ORBoostFilter PF	AENN BBNR CNN DROP1 DROP2 DROP3 ENG ENN PRISM RNN TomekLinks	saturationFilter consensusSF classifSF
	Repair/ Hybrid	hybridRepairFilter	EWF GE ModeFilter	

The rest of this section is organized as follows. First, a few lines are devoted to the installation process. Then, we present the documentation page for the filters, where further specific details can be looked up. After that, we focus on the two implemented methods to call the filters (*default* and *formula*). Finally, the `"filter"` class, which unifies the return value of the filters in **NoiseFiltersR** package, is presented.

Installation

The **NoiseFiltersR** package is available at CRAN servers, so it can be downloaded and installed directly from the R command line by typing:

```
> install.packages("NoiseFiltersR")
```

This command will also install the eleven dependencies of the package, which mainly provide the classification algorithms needed for some of the implemented filters, and which can be looked up in the "Imports" section of the CRAN website for the package https://cran.r-project.org/web/packages/NoiseFiltersR/index.html.

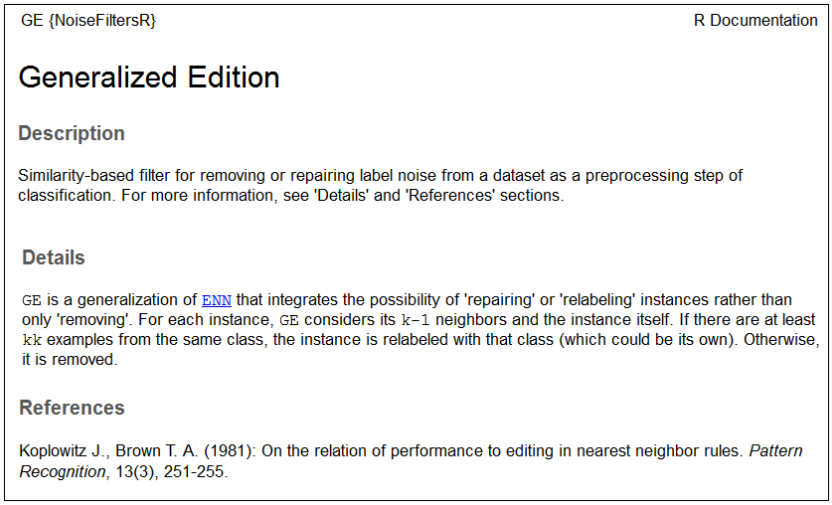
In order to easily access all the package's functions, it must be attached in the usual way:

```
> library(NoiseFiltersR)
```

Documentation

Whereas this paper provides the user with an overview of the **NoiseFiltersR** package, it is also important to have access to specific information for each available filter. This information can be looked up in the corresponding documentation page, that in all cases includes the following essential items (see Figure 1 for an example):

- A *description* section, which indicates the type of filter according to the taxonomy explained in Table 1.
- A *details* section, which provides the user with a general explanation of the filter's behaviour and any other usage particularity or warning.
- A *references* section that points to the original contribution where the filter was proposed, where further details, motivations or contextualization can be found.


GE {NoiseFiltersR} R Documentation

Generalized Edition

Description

Similarity-based filter for removing or repairing label noise from a dataset as a preprocessing step of classification. For more information, see 'Details' and 'References' sections.

Details

GE is a generalization of ENN that integrates the possibility of 'repairing' or 'relabeling' instances rather than only 'removing'. For each instance, GE considers its k−1 neighbors and the instance itself. If there are at least kk examples from the same class, the instance is relabeled with that class (which could be its own). Otherwise, it is removed.

References

Koplowitz J., Brown T. A. (1981): On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition*, 13(3), 251-255.

Figure 1: Extract from GE filter's documentation page, showing the highlighted above aspects.

As usually in R, the function documentation pages can be either checked in the CRAN website for the package or loaded from the command line with the orders ? or help:

```
> ?GE
> help(GE)
```

Calling the filters

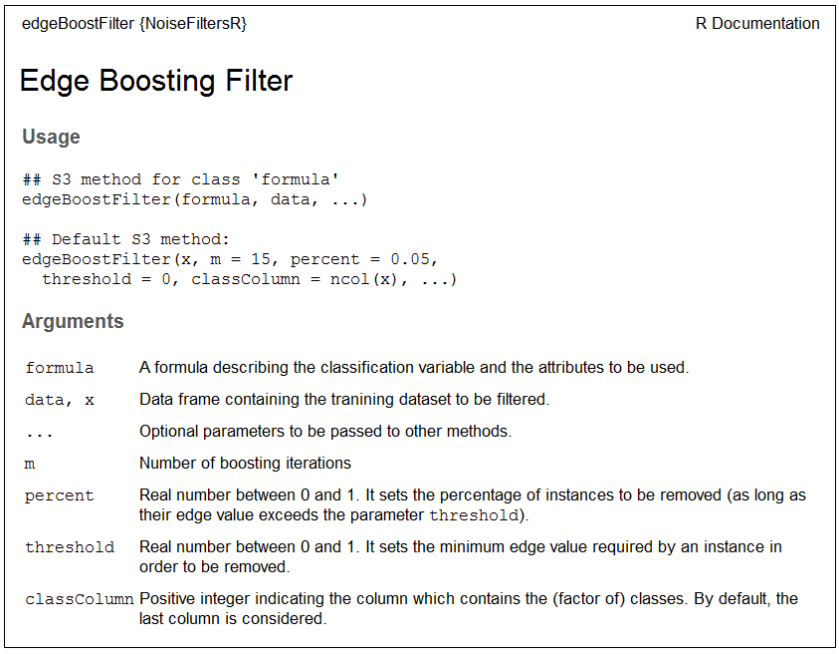
When one wants to use a label noise filter in Data Mining applications, all we need to know is the dataset to be filtered and its *class* variable (i.e. the one that contains the label for each available instance). The **NoiseFiltersR** package provides two standard ways for tagging the class variable when calling the implemented filters (see also Figure 2 and the example below):

- The *default* method receives the dataset to be filtered in the `x` argument, and the number for the class column through the `classColumn` argument. If the latter is not provided, the last column of the dataset is assumed to contain the labels.
- The *formula* method is intended for regular R users, who are used to this approach when fitting regression or classification models. It allows for indicating the class variable (along with the attributes to be used) by means of an expression like `Class~Attr1+...+AttrN` (recall that `Class~.` makes use of all attributes).

Next, we provide an example on how to use these two methods for filtering out the `iris` dataset with `edgeBoostFilter` (we did not change the default parameters of the filter):

```
# Checking the structure of the dataset (last variable is the class one)
> data(iris)
> str(iris)
'data.frame':        150 obs. of  5 variables:
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

# Using the default method:
> out_Def <- edgeBoostFilter(iris, classColumn = 5)
# Using the formula method:
> out_For <- edgeBoostFilter(Species~., iris)
# Checking that the filtered datasets are identical:
> identical(out_Def$cleanData, out_For$cleanData)
[1] TRUE
```



edgeBoostFilter {NoiseFiltersR} R Documentation

Edge Boosting Filter

Usage

```
## S3 method for class 'formula'
edgeBoostFilter(formula, data, ...)

## Default S3 method:
edgeBoostFilter(x, m = 15, percent = 0.05,
  threshold = 0, classColumn = ncol(x), ...)
```

Arguments

`formula`	A formula describing the classification variable and the attributes to be used.
`data, x`	Data frame containing the tranining dataset to be filtered.
`...`	Optional parameters to be passed to other methods.
`m`	Number of boosting iterations
`percent`	Real number between 0 and 1. It sets the percentage of instances to be removed (as long as their edge value exceeds the parameter `threshold`).
`threshold`	Real number between 0 and 1. It sets the minimum edge value required by an instance in order to be removed.
`classColumn`	Positive integer indicating the column which contains the (factor of) classes. By default, the last column is considered.

Figure 2: Extract from `edgeBoostFilter`'s documentation page, which shows the two methods for calling filters in **NoiseFiltersR** package. In both cases, the parameters of the filter can be tunned through additional arguments.

Notice that, in the last command of the example, we used the `$` operator to access the objects returned from the filter. In next section we explore the structure and contents of these objects.

The `"filter"` class

The S3 class `"filter"` is designed to unify the return value of the filters inside the **NoiseFiltersR** package. It is a list that encapsulates seven elements with the most relevant information of the process:

- `cleanData` is a data.frame containing the filtered dataset.
- `remIdx` is a vector of integers indicating the indexes of removed instances (i.e. their row number with respect to the original data.frame).
- `repIdx` is a vector of integers indicating the indexes of repaired/relabelled instances (i.e. their row number with respect to the original data.frame).
- `repLab` is a factor containing the new labels for repaired instances.
- `parameters` is a list that includes the adopted parameters for the filter.
- `call` is an expression that contains the original call to the filter.
- `extraInf` is a character vector including additional information not covered by previous items.

As an example, we can check the structure of the above `out_For` object, which was the return value of `egdeBoostFilter` function:

```
> str(out_For)
List of 7
$ cleanData :'data.frame':        142 obs. of  5 variables:
..$ Sepal.Length: num [1:142] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
..$ Sepal.Width : num [1:142] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
..$ Petal.Length: num [1:142] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
..$ Petal.Width : num [1:142] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
$ remIdx    : int [1:8] 58 78 84 107 120 130 134 139
$ repIdx    : NULL
$ repLab    : NULL
$ parameters:List of 3
..$ m        : num 15
..$ percent  : num 0.05
..$ threshold: num 0
$ call      : language edgeBoostFilter(formula = Species ~ ., data = iris)
$ extraInf  : chr "Highest edge value kept: 0.0669358381115568"
- attr(*, "class")= chr "filter"
```

In order to cleanly display this `"filter"` class in the R console, two specific `print` and `summary` methods were implemented. The appearance of the first one is as follows

```
> print(out_For)

Call:
edgeBoostFilter(formula = Species ~ ., data = iris)

Parameters:
m: 15
percent: 0.05
threshold: 0

Results:
Number of removed instances: 8 (5.333333 %)
Number of repaired instances: 0 (0 %)
```

and contains three main blocks:

- The original *call* to the filter.
- The *parameters* used for the filter.
- An overview of the *results*, with the absolute number (and percentage of the total) of removed and repaired instances.

The `summary` method displays some extra blocks:

- It always adds a title that summarizes the filter and dataset used.
- If there exists additional information in the `extraInf` component of the object, it is displayed under a homonymous block.
- If the argument `explicit` is set to `TRUE` (it defaults to `FALSE`), the explicit results (i.e. the indexes for removed and repaired instances and the new labels for the latters) are displayed.

In the case of the previous `out_For` object, the `summary` command gets the following format:

```
> summary(out_For, explicit = TRUE)

Filter edgeBoostFilter applied to dataset iris

Call:
edgeBoostFilter(formula = Species ~ ., data = iris)

Parameters:
m: 15
percent: 0.05
threshold: 0

Results:
Number of removed instances: 8 (5.333333 %)
Number of repaired instances: 0 (0 %)

Additional information:
Highest edge value kept: 0.0669358381115568

Explicit indexes for removed instances:
58 78 84 107 120 130 134 139
```

Summary

In this paper, we introduced the **NoiseFiltersR** package, which is the first R extensive implementation of classification-oriented label-noise filters. To set a context and motivation for this work, we presented the problem of label noise and the main approaches to deal with it inside data preprocessing, as well as the related software. As previously explained, the released package unifies the return value of the filters by means of the `"filter"` class, which benefits from specific `print` and `summary` methods. Moreover, it provides a R-user-friendly way to call the implemented filters, whose documentation is worth reading and points to the original reference where they were first published.

Regarding the potential extensions of this package, there exist several aspects which can be adressed in future releases. For instance, there exist some other label noise filters reviewed in the main reference (Frénay and Verleysen, 2014) whose noise identification strategy does not belong to the ones covered here: ensemble based, similarity based and data complexity based (as shown in Table 1). Other relevant extension would be the inclusion of some datasets with different levels of artificially introduced label noise, in order to ease the experimentation workflow[2].

Acknowledgements

This work was supported by the Spanish Research Project TIN2014-57251-P, the Andalusian Research Plan P11-TIC-7765, and the Brazilian grants CeMEAI-FAPESP 2013/07375-0 and FAPESP 2012/22608-8. Luís P. F. Garcia was supported by FAPESP 2011/14602-7.

Bibliography

J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2010. [p1]

C. Bouveyron and S. Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009. [p1, 2]

C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999. [p2]

P. Filzmoser and M. Gschwandtner. *mvoutlier: Multivariate outlier detection based on robust methods*, 2015. URL https://CRAN.R-project.org/package=mvoutlier. R package version 2.0.6. [p1]

[2] A wide variety of such datasets can be downloaded from the KEEL dataset repository in the website http://www.keel.es/, and then loaded from R.

B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014. [p1, 2, 3, 7]

L. P. Garcia, J. A. Sáez, J. Luengo, A. C. Lorena, A. C. de Carvalho, and 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. *Knowledge-Based Systems*, 90:153–164, 2015. [p1]

S. García, J. Luengo, and F. Herrera. *Data preprocessing in data mining*. Springer, 2015. [p1]

S. García, J. Luengo, and F. Herrera. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowledge-Based Systems*, 98:1–29, 2016. [p1]

M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery*, 2:9–37, 1998. [p2]

J. Honaker, G. King, and M. Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011. URL http://www.jstatsoft.org/v45/i07/. [p1]

K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets weka. *Computational Statistics*, 24(2):225–232, 2009. [p2]

T. M. Khoshgoftaar and P. Rebours. Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology*, 22(3):387–396, 2007. [p2]

M. Kuhn. Caret package. *Journal of Statistical Software*, 28(5), 2008. [p1]

N. D. Lawrence and B. Schölkopf. Estimating a kernel fisher discriminant in the presence of label noise. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 306–313, 2001. [p1, 2]

Y. Li, L. F. Wessels, D. de Ridder, and M. J. Reinders. Classification in the presence of class noise using a probabilistic kernel fisher method. *Pattern Recognition*, 40(12):3349–3357, 2007. [p2]

Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, and J. Song. Rboost: Label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2216–2228, 2016. [p2]

A. L. Miranda, L. P. F. Garcia, A. C. Carvalho, and A. C. Lorena. Use of classification algorithms in noise detection and elimination. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 417–424. Springer, 2009. [p3]

A. D. Pozzolo, O. Caelen, and G. Bontempi. *unbalanced: Racing for Unbalanced Methods Selection*, 2015. URL https://CRAN.R-project.org/package=unbalanced. R package version 2.0. [p1]

J. R. Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014. [p2]

P. Romanski and L. Kotthoff. *FSelector: Selecting attributes*, 2014. URL https://CRAN.R-project.org/package=FSelector. R package version 0.20. [p1]

J. A. Sáez, M. Galar, J. Luengo, and F. Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems*, 38 (1):179–206, 2014. [p2]

J. A. Sáez, M. Galar, J. Luengo, and F. Herrera. Inffc: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control. *Information Fusion*, 27:19–32, 2016. [p1]

C.-M. Teng. Correcting Noisy Data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 239–248, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers. [p2]

C. M. Teng. Dealing with data corruption in remote sensing. In *International Symposium on Intelligent Data Analysis*, pages 452–463. Springer, 2005. [p3]

I. Tomek. Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, 6:769–772, 1976. [p1]

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011. URL http://www.jstatsoft.org/v45/i03/. [p1]

R. Y. Wang, V. C. Storey, and C. P. Firth. A Framework for Analysis of Data Quality Research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–640, 1995. [p2]

D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3):408–421, 1972. [p1]

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005. [p1, 2]

X. Wu. *Knowledge acquisition from databases*. Ablex Publishing Corp., Norwood, NJ, USA, 1996. [p2]

X. Wu and X. Zhu. Mining with noise knowledge: error-aware data mining. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(4):917–932, 2008. [p2]

S. Zhong, T. M. Khoshgoftaar, and N. Seliya. Analyzing Software Measurement Data with Clustering Techniques. *IEEE Intelligent Systems*, 19(2):20–27, 2004. [p2]

X. Zhu and X. Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22 (3):177–210, 2004. [p2]

Pablo Morales
Department of Computer Science and Artificial Intelligence, University of Granada
18071 Granada, Spain
pablomorales@correo.ugr.es

Julián Luengo
Department of Computer Science and Artificial Intelligence, University of Granada
18071 Granada, Spain
julianlm@decsai.ugr.es

Luís P.F. Garcia
Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
Trabalhador São-carlense Av. 400, São Carlos, São Paulo 13560-970, Brazil
lpgarcia@icmc.usp.br

Ana C. Lorena
Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo
Talim St. 330, São José dos Campos, São Paulo 12231-280, Brazil
aclorena@unifesp.br

André C.P.L.F. de Carvalho
Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
Trabalhador São-carlense Av. 400, São Carlos, São Paulo 13560-970, Brazil
andre@icmc.usp.br

Francisco Herrera
Department of Computer Science and Artificial Intelligence, University of Granada
18071 Granada, Spain
herrera@decsai.ugr.es