Neurocomputing 353 (2019) 83–95

Contents lists available at ScienceDirect

Neurocomputing

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

Label noise filtering techniques to improve monotonic classification

José-Ramón Cano[a], Julián Luengo[b], Salvador García[b,*]

[a] Department of Computer Science, EPS of Linares, University of Jaén, Campus Científico Tecnológico de Linares, Cinturón Sur S/N, Linares 23700, Jaén, Spain
[b] Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain

ARTICLE INFO

Article history:
Received 13 December 2017
Revised 8 April 2018
Accepted 26 May 2018
Available online 13 March 2019

Keywords:
Ordinal classification
Monotonic classification
Preprocessing
Noise filtering

ABSTRACT

The monotonic ordinal classification has increased the interest of researchers and practitioners within machine learning community in the last years. In real applications, the problems with monotonicity constraints are very frequent. To construct predictive monotone models from those problems, many classifiers require as input a data set satisfying the monotonicity relationships among all samples. Changing the class labels of the data set (relabeling) is useful for this. Relabeling is assumed to be an important building block for the construction of monotone classifiers and it is proved that it can improve the predictive performance.

In this paper, we will address the construction of monotone datasets considering as noise the cases that do not meet the monotonicity restrictions. For the first time in the specialized literature, we propose the use of noise filtering algorithms in a preprocessing stage with a double goal: to increase both the monotonicity index of the models and the accuracy of the predictions for different monotonic classifiers. The experiments are performed over 12 datasets coming from classification and regression problems and show that our scheme improves the prediction capabilities of the monotonic classifiers instead of being applied to original and relabeled datasets. In addition, we have included the analysis of noise filtering process in the particular case of wine quality classification to understand its effect in the predictive models generated.

1. Introduction

In the last years learning with ordinal data has increased the attention of machine learning research community. Ordinal datasets are characterized by the presence of an ordinal output variable. We find many examples of ordinal data in real life [1–3].

The classification with monotonicity constraints, also known as monotonic classification [4] or isotonic classification [5], is an ordinal classification problem [6] where a monotonic restriction is present. In monotonic classification, a higher value of an attribute in an example, fixing other values, should not decrease its class assignment. The monotonicity of relations between the dependent and explanatory variables is very usual as a prior knowledge form in data classification [7]. To illustrate, while considering a credit card application [8], a \$1000 to \$2000 income may be considered a medium value of income in a data set. If a customer A has a medium income, a customer B has a low income (i.e. less than \$1000) and the rest of input attributes remain the same, there is a relationship of partial order between A and B: $B < A$. Considering that the application estimates lending quantities as output class, it is quite obvious that the loan that the system should give to customer B cannot be greater than the given to customer A. If so, a monotonicity constraint is violated in the decision. Monotonicity is modelling or representing the business logic of the bank's process, which depends on this order.

Several monotonic classification approaches have been proposed in the specialized literature. They include classification trees and rule induction [9–14], neural networks [15,16], instance-based learning [4,17,18] and hybridizations [19,20]. Some of them require the training set to be purely monotone to work correctly, such as the MKNN classifier [18]. MKNN needs as input a completely monotonic data set and if not, it must be converted in monotonic using relabeling preprocessing methods [18]. Other classifiers are more permissive allowing as input non-monotonic datasets, but they may not guarantee to make monotonic predictions. An example of this could be the MID algorithm [9]. The natural approach is to deal with non-monotonic datasets, as we will justify next. The proposals we find in the literature are designed to work in one way or another or even both at the same time. But sometimes you have to delve into the papers to find out since there is not a catalogue of methods categorized by each type of problem.

* Corresponding author.
E-mail addresses: jrcano@ujaen.es (J.-R. Cano), julianlm@decsai.ugr.es (J. Luengo), salvagl@decsai.ugr.es (S. García).

https://doi.org/10.1016/j.neucom.2018.05.131
0925-2312/© 2019 Elsevier B.V. All rights reserved.

Nevertheless, real-world datasets may surely contain noise, which is defined as anything that obscures the relationship between the features of an instance and its class [21,22]. Among other consequences, noise can adversely impact the classification performances of induced classifiers. By extrapolating to monotonic classification, noise also alters the monotonicity constraints present in the data.

In order to test the performance of monotonic classifiers, the usual trend is to generate datasets that fulfil the monotonicity constraint. The main argument is that models trained on monotonic datasets often have better predictive performance than models trained on the original data [23]. Monotonic datasets can be created by generating artificial data [24] or by relabeling of real data [18,25,26].

This paper proposes a different general approach to deal with the construction of monotonic models by any classifier. As an alternative, we will consider the examples that do not fulfil the monotonic constraints as noisy examples. For the first time in the literature, we propose the application of noise filtering algorithms in a preprocessing stage for monotonic ordinal classification.

In this study, four classical noise filtering algorithms have been readjusted to this domain. The algorithms considered are the Edited Nearest neighbor (ENN [27]), the Relative neighborhood Graph Editing (RNGE [28]), the Iterative Partition Filtering (IPF [29]) and the Iterative Noise Filter based on the Fusion of Classifiers (MINFFC [30]). The monotonic version of them is noted as MENN, MRNGE, MIPF and MINFFC, respectively. They will be reformulated to detect the non-monotonic samples, which are considered as noisy samples, following different heuristics and strategies. The remaining samples will be those used as input in well-known monotonic classifiers. Our objective is to analyze their performances in comparison with an optimal relabeling scheme of the original datasets in a large collection of datasets.

The paper is organized as follows. In Section 2 we present the monotonic classification problem, its formal definition and the monotonic classifiers used in the study. Section 3 is devoted to describing the noise filtering algorithms and their adaptation to satisfy the monotonicity constraints. Section 4 describes the experimental framework. Section 5 analyzes the results obtained in the empirical study and presents a discussion and analysis of them. In Section 6 we apply the filter process analyzed in the Winequality-red data set to study its behavior. Finally, Section 7 concludes the paper.

2. Monotonic classification

In this section, we introduce the monotonic classification problem and the monotonic classifiers considered in this paper.

2.1. Problem definition

The property of monotonicity commonly appears in domains of our lives such as natural sciences, natural language, game theory or economics [7,31]. For instance, the case of bankruptcy prediction in companies, where appropriate actions can be taken in time considering the information based on financial indicators taken from their annual reports. The monotonicity is clearly present in the comparison of two companies where one dominates the other on all financial indicators, which supposes that the overall evaluation of the second cannot be higher than the evaluation of the first. This rule could be applied to the credit rating strategy used by banks [8] as well as for the bankruptcy prediction strategy.

The monotonic ordinal data can be defined as following. Let D be a data set with f ordinal attributes $A_1, \ldots, A_f$ and one output class attribute Y having c possible ordinal values. The data set consists of n examples x_i. A partial ordering $\preceq$ on D is defined as

$$x \preceq x' \Leftrightarrow A_j(x) \leq A_j(x'), \forall j = 1, \ldots, f \tag{1}$$

Two examples x and x' in space D are *comparable* if either $x \preceq x'$ or $x' \preceq x$, otherwise x and x' are *incomparable*. Two examples x and x' are *identical* if $x = x$' and *non-identical* if $x \neq x$';

Considering this notation, we denote a pair of comparable examples (x,x') *monotone* if

$$x \preceq x' \wedge x \neq x' \wedge Y(x) \leq Y(x') \tag{2}$$

or

$$x = x' \wedge Y(x) = Y(x') \tag{3}$$

A data set D with n examples is monotone if all possible pairs of examples are either monotone or incomparable.

2.2. Monotonic classifiers

This section reviews the monotone classifiers considered in this paper. They are well-known in the monotonic classification topic.

- Monotonic k-Nearest neighbor Classifier (MkNN), proposed in [18], can be described as follows.
  The first step in this proposal is to relabel the training data aiming to remove all monotonicity violations [32], using as few label changes as possible.
  Starting from the original nearest neighbor rule, the class label assigned to a new data point x_0 must lie in the interval [y_{min}, y_{max}] in order to preserve the monotonicity conditions, where

  $$y_{min} = max\{y | (x, y) \in D \wedge x \preceq x_0\} \tag{4}$$

  and

  $$y_{max} = min\{y | (x, y) \in D \wedge x_0 \preceq x\} \tag{5}$$

  y_{max} is the minimum class value of all those instances (x) in the data set (D) whose attribute values are all bigger than, or equal to, those of x_0. And y_{min} is determined as the maximum class value of all those instances (x) in the data set (D) whose attributes values are all smaller than, or equal to, those of x_0. The nearest neighbor rule considered takes the k nearest neighbors of x_0 from D whose labels are included in [y_{min}, y_{max}] and makes predictions according to majority voting. If there is not any neighbor which has its class in the range [y_{min}, y_{max}] means that any class assigned will result in a violation of monotonicity and a random class is assigned as prediction [18].
- Ordinal Learning Model (OLM). It was the first method proposed for ordinal classification with monotonicity constraints [4]. Its goal is to choose a subset $D' \subseteq D$ of training samples in which all meet the monotonicity conditions. The classification of new objects is done by the following function:

  $$f_{OLM}(x) = max\{y_i : x_i \in D', x_i \preceq x\} \tag{6}$$

  If there is no object from D' which is dominated by x, then a class label is assigned by the nearest neighbor rule. D' is chosen to be consistent and not to contain redundant examples. An object x_i is redundant in D' if there is another object x_j such that $x_i \succeq x_j$ and $y_i = y_j$.
- Ordinal Stochastic Dominance Learner (OSDL). It was presented in [17] and [33] as an instance-based method for ordinal classification with monotonicity constraints based on the concept of ordinal stochastic dominance. The rationale behind it can be given through an example: In life insurance, one may expect a stochastically greater risk to the insurer from older and sicker applicants than from younger and healthier ones. Higher premiums should reflect greater risks and vice versa. There are several definitions of stochastic ordering. The most commonly used

is the following: For each example x_i, the OSDL computes two mapping functions: one that is based on the examples that are stochastically dominated by x_i with the maximum label (of that subset), and the second is based on the examples that cover (i.e., dominate) x_i, with the smallest label. Later, an interpolation between the two class values (based on their position) is returned as a class.

- Monotone Induction of Decision trees (MID) [9]. Its main idea was to modify the conditional entropy in the ID3 algorithm ([34]), by adding a term called *order-ambiguity-score*, which adapts the splitting strategy towards the building of monotonic trees. However, this strategy did not guarantee the tree to be a completely monotone function.

3. Label noise filtering in monotonic classification

As in standard classification, the correct labeling of the training set is crucial to obtain accurate models that will correctly predict new examples. Most classification algorithms assume that the labeling in the data is correct and follows the underlying distribution without any disturbances. However, in the real world, this assumption is naive. Real-world data is far from being perfect and corruptions usually affect the dependence between the input and output attributes [21]. These disturbances will alter or bias the models, hindering their quality.

In classification, the noise may affect the data registration of the input attributes or the labeling process made automatically or by an expert [35]. If the noise has affected the input attributes, it is usually named *attribute noise*. On the other hand, *class noise* or *label noise* means that the noise has corrupted the correct label of some instances. Some studies have analyzed the impact of these types of noise, indicating that class noise is more harmful than attribute noise, as the bias introduced is greater. For this reason, in this work, we will focus on label noise and the different ways to tackle its greater impact.

To address this, a conventional strategy consists of *relabeling* the input data to fulfil the monotonicity assumptions [18,23,25]. In summary, this process alters the class of those pairs of instances which violate the monotonic restrictions, trying to minimize the total number of changes.

As an alternative strategy, we focus our attention on the approaches to deal with class noise in the specialized literature for classification, which is often grouped into three families.

1. We can adapt the classifiers to take into account noise and become *robust learners*, less influenced by noise. Some classifiers were designed as robust against noise from their very conception, as Kernel Logistic Regression [36], a robust SVM [37] or robust Fisher discriminant analysis learners [38,39] just to mention a few. We can find adaptations of noise sensitive learners as AdaBoost [40] to become robust learners in recent proposals [41].
2. While robust learners can deal with noisy data directly, not all classifiers have been modified to be robust. The solutions designed to make a classifier robust cannot be easily extrapolated to other learners. Thus, a popular option is to apply *noise filtering* methods [42–44], which work at data level before the classifier is applied. Noise filters aim to detect and eliminate the corrupted instances that would hinder the model built by the classifier, enabling any classifier model to work with noisy datasets.
3. Noise filters are a popular option due to their easy application and independence with the classifier. However, the cost of eliminating instances cannot be disregarded, especially in highly noisy problems. In these cases, the number of instances eliminated would be high enough to produce a data shift in the class borders. An optimal preprocessing technique would recover the noisy instances, relabeling them with their true label. This family of techniques are known as *data correcting methods* [45,46].

Frénay and Verleysen [21] point out that filtering noisy instances is more efficient than correcting them [47,48]. Since correction is never perfect (as a perfect classifier is rarely held), errors will be further added and can accumulate with the noise we intend to remove. Obtaining correctors with low wrong correction rates are computationally expensive ("less efficient" as Frénay and Verleysen meant). Therefore, correcting methods have drawn less attention in the literature than filtering approaches. Thus, we will focus on filtering approaches for noise. Among filters, those based on similarity measures and multi-classifiers of ensembles are very popular.

We propose the application of data preprocessing techniques to the original data, which have been successfully applied in the past in similar domains [49–53]. In particular, we consider readjusted noise filtering algorithms to tackle the monotonic domain [21]. These methods identify and remove some of the examples belonging to the data set presenting a negative effect to the fulfilment of the monotonicity constraints, keeping unaltered the rest of the information held in the original data set. Our selection of popular noise filters is justified based on those that obtained the best performances in other learning domains, such as standard classification [30], imbalanced classification [54] or semi-supervised classification [55]. Next, we describe the filters used in this paper.

3.1. Monotonic Edited Nearest neighbor (MENN)

The Monotonic Edited Nearest neighbor (MENN) evolves from the classical Edited Nearest neighbor algorithm [27]. It iterates over each instance x in the dataset, and finds the k monotonic nearest neighbors for x. Once such neighbors are found, the k labels are counted. The label with the highest frequency among the k neighbors' labels is compared to the actual label of x. If the label of x is different than the most frequent label of its k monotonic nearest neighbors, x is removed from the training set.

The use of the monotonic k-nearest neighbors (see Section 2.2 for further details on how to obtain the k monotonic neighbors) instead of the classical k-nearest neighbors rule constitutes the adaptation of this algorithm to the monotonic scope. The pseudo-code of MENN is presented in Algorithm 1.

Algorithm 1 MENN algorithm.

```
function MENN($T$ - training data, $k$ - number of nearest neighbor)
    initialize: $S = T$
    for all $x \in S$ do
        $X' = \emptyset$
        $y_{min} = max\{class(x') | x' \in T \wedge x' \leq x\}$
        $y_{max} = min\{class(x') | x' \in T \wedge x \leq x'\}$
        for $i = 1$ to $k$ do
            Find $x'_i \in T$ s.t. $x \neq x'_i$ and $||x - x'_i|| = \min_{x^j \in (T \setminus X')} ||x - x^j||$ and $class(x'_i) \in [y_{min}, y_{max}]$
            $X' = X' \cup \{x'_i\}$
        end for
        if $class(x) \neq majorityClass(X')$ then
            $S = S \setminus \{x\}$
        end if
    end for
    return $S$
end function
```

3.2. Monotonic Relative neighborhood Graph Editing (MRNGE)

The Monotonic Relative neighborhood Graph Editing (MRNGE) works in a decremental fashion like MENN and it is based on proximity graphs [28]. In a first phase, MRNGE creates a proximity graph, where the closest instances are linked together in such a way that no other closest example is found between them. This is the goal of the Proximity Graph function. Once the proximity graph has been built, we can rapidly access to the nearest neighbors of any instance.

In the second phase, MRNGE utilizes the graph to remove the examples by looking at is neighbors. However, substantial differences can be found with respect to MENN: instead of only counting the direct neighbors of the reference example x, MRNGE will access to the neighbors' neighbor. Without the proximity graph, this operation would be computationally expensive.

This "second order neighborhood" aims to diminish the influence of small clusters of noisy examples by extending the examined neighborhood. Thus, MRNGE first examines the k monotonic neighbors for the given example x by accessing the proximity graph. Only in the case where the most frequent class label among the k neighbors differs from the label of x, we go further and examine the labels of x neighbors' neighbors (please note that MENN would have stopped here, removing x). MRNGE will count the labels of this "second order neighborhood" and only in the case of the most frequent label of the neighbor's neighbors is different to the label of x will mean that x is removed from the dataset. Again, the precomputed proximity graph enables the fast gathering of the neighbors of any given instance.

To work with monotonic classification problems, the graph will be created considering the monotonicity constraints (see Section 2.1). Algorithm 2 depicts in detail the procedure MRNGE.

Algorithm 2 MRNGE algorithm.

```
function MRNGE($T$ - training data, $k$ - number of nearest neighbor)
    initialize: $PG = Proximity\_Graph(T)$, $S = T$
    for all $x \in S$ where $x$ is misclassified by its $k$ neighbors in $PG$ do
        Consider the subgraph, $R$, given by $x$ and all its graph neighbors from its same class
        $S = S \setminus \{x\}$ if the graph neighborhood of $R$ has a majority of neighbors from different class than $x$.
    end for
    return $S$
end function
function PROXIMITY_GRAPH($T$ - training data)
    initialize: $PG = (V, E)$, $V = T$, $E = \emptyset$
    for all $x_i \in E$ do
        for all $x_j \in E$ do
            for all $x_k \in E$ do
                if (k $\neq$ i and k $\neq$ j) then
                    $d_{ij} = EuclideanDistance(x_i, x_j)$
                    $d_{ik} = EuclideanDistance(x_i, x_k)$
                    $d_{jk} = EuclideanDistance(x_j, x_k)$
                    if $d_{ij}^2 \leq d_{ik}^2 + d_{jk}^2$ then
                        $E = E \bigcup (x_i, x_j)$
                    end if
                end if
            end for
        end for
    end for
    return $PG$
end function
```

3.3. Monotonic Iterative Partition Filtering Editing (MIPF)

The Monotonic Iterative Partition Filtering (MIPF) is a global noise filter which applies a classifier to several subsets of the training data set to detect possible noisy examples. It removes noisy instances in multiple iterations until the number of identified noisy examples, for a number of consecutive iterations, is less than a percentage of the size of the original training data set [29]. The classifier embedded in the classic Iterative Partition Filtering algorithm is the C4.5 [56]. Since C4.5 does not takes into account the monotonic restrictions of examples, for our MIPF proposal, we consider the ordinal interpretation of C4.5 [57] as the base classifier. It is worth mentioning that the ordinal C4.5 does not produce monotonic models but ensures ordinal classification. MIPF is described in Algorithm 3.

Algorithm 3 MIPF algorithm.

```
function MIPF($T$ - dataset with Monotonic Violations, $\Gamma$ - number of subsets, $y$ - amount of good data to be eliminated in each step, $p$ - minimum percentage of noisy instances to continue)
    initialize: $T_G = \{\}$, $F$ = Ordinal C4.5
    repeat
        Split the training data set $T$ into $T_i, i = 1 \ldots \Gamma$ equal sized subsets
        for each subset $T_i$ do
            Use $\{T_j, j \neq i\}$ to train $F$ resulting in $F^i$ different classifiers
        end for
        $D_N = \{\}, D_G = \{\}$
        for each instance $t$ in $T$ do
            Classify $t$ with every $F^i$
            if $t$ is voted as noisy then
                $D_N = D_N \cup t$
            end if
        end for
        $D_G = \{t_l \in T | t_l \notin D_N; l = 1, \ldots, y\}$
        $T_G = T_G \cup D_G$
        $T = T - \{D_N \cup D_G\}$
    until $|D_N| < p \cdot |T|$
    return $T \cup T_G$
end function
```

3.4. Monotonic Iterative class Noise Filter based on the Fusion of Classifiers (MINFFC)

Finally, we include the Monotonic Iterative Noise Filter based on the Fusion of Classifiers (MINFFC), based on the proposal for standard classification made in [30]. It is an advanced filter that gathers strategies and techniques from different kinds of noise filters: ensembles of filters, noise scoring and an iterative processing. It is based on an ensemble of classifiers to select a candidate set of noisy instances. It first performs a preliminary filtering, generating a temporarily reduced set, which is then used to generate the final filtering ensemble. The final filtering ensemble only points out the set of noisy candidate instances. Such a set is the foundation to compute a noise score for all the instances, thus removing those that are detected as noise with a score greater than 0. The process is applied iteratively until the number of removed examples is below a percentage for a given number of iterations.

For monotonic classification, the computation of the *noiseScore* is adapted from that described in [30]. We multiply the *clean* value of each instance by the NMI1 index (see Section 4.2) previously computed. MINFFC is described in Algorithm 4.

Algorithm 4 MINFFC algorithm.

```
function MINFFC($T$ - dataset with Monotonic Violations, $g$ - number of iterations to stop, $p$ - minimum percentage of noisy instances to stop, $k$ - number of neighbors to compute the noise score, $NMI1$ - Non-monotonic index for each instance)
    initialize: $T_G = \{\}$, $F_1$ = Ordinal C4.5, $F_2$ = 3-NN, $F_3$ = Logistic Regression, $it = 0$
    repeat
        Split the training data set $T$ into $T_i, i = 1, \ldots, \Gamma$ equal sized subsets
        for each subset $T_i$ do
            Use $\{T_j, j \neq i\}$ to train $F_j^i, j = 1, \ldots, 3$ different classifiers
        end for
        $D_N = \{\}, D_G = \{\}$
        for each instance $t$ in $T$ do
            Classify $t$ with every $F_j^i$ where $t \notin T_i$
            if $t$ is voted as noisy then
                $D_N = D_N \cup t$
            end if
        end for
        $D_G = \{t_l \in T | t_l \notin D_N; l = 1, \ldots, y\}$
        Split the training data set $D_G$ into $D_i, i = 1, \ldots, \Gamma$ equal sized subsets
        for each subset $D_i$ do
            Use $\{D_j, j \neq i\}$ to train $F_j^i, j = 1, \ldots, 3$ different classifiers
        end for
        $D'_N = \{\}$
        for each instance $t$ in $T$ do
            Classify $t$ with every $F_j^i$ where $t \notin T_i$
            if $t$ is voted as noisy then
                $D'_N = D'_N \cup t$
            end if
        end for
        for each instance $t$ in $T$ do
            score = $noiseScore(t) \cdot NMI1(t)$
            if score $\geq 0$ then
                $T = T - t$
            end if
        end for
        if $|D_N| < p \cdot |T|$ then
            $it = it - 1$
        else
            $it = g$
        end if
    until it = 0
    return $T$
end function
```

4. Experimental framework

In this section, we present the experimental framework developed to analyze the proposal of application of four well-known noise filtering algorithms readjusted to work in this domain. Section 4.1 introduces the datasets used in the comparison. Section 4.2 describes the metrics used to evaluate the compared methods. Section 4.3 lists all the parameters used for each method in the experimental comparison. Finally, Section 4.4 describes the statistical procedures employed to support the analysis carried out.

Table 1
Description of the 12 datasets used in the study.

Data set	Ins.	At.	Cl.	#Mon. features
Balance	625	4	3	4
BostonHousing	506	12	4	10
Car	1728	6	4	6
Era	1000	4	9	4
Esl	488	4	9	4
Lev	1000	4	5	4
CPU	209	6	4	6
QualitativeBankruptcy	250	6	2	6
Swd	1000	10	4	7
WindsorHousing	545	11	2	2
Winequality-red	1599	11	11	8
Wisconsin	683	9	2	9

4.1. Datasets

The study includes 12 datasets whose class attribute can be expressed as ordinal and presents a monotonic relationship with the features. Four datasets are actual classical ordinal classification datasets commonly used in this field (Era, Esl, Lev and Swd [4]). The other 7 are regression datasets whose class attribute was discretized into 4 categorical values, maintaining the class distribution balanced. The last one is Winequality-red, a well-known data set in classification which is extensively analyzed in Section 6. All of the 12 datasets are classical problems used in the classification scope and extracted from the UCI [58] and KEEL[1] repositories [59,60].

In order to evaluate the performance of the different approaches with different amounts of monotonic violations, we have generated three corrupted versions of each dataset. These altered versions are created by changing a *noise%* of instances by relabeling them with a new class label. The new label can only be the precedent or the following one, thus generating realistic disorders in terms of monotonic violations:

- When corrupting to the next label, the modified instance is not the corrupted datum, but the instances that have been surpassed by such noisy instance.
- When applying this corruption scheme, the class order is considered as cyclical. If the last label is to be corrupted to the next class label, we will select the first label as the new output value. On the other hand, if the first label is to be corrupted to the previous label, we will select the last label instead. These exceptional cases will induce even more noise than the intermediate labels.

Such a noise introduction scheme follows the NAR mechanism as described in [21], in which the true label has influence in the observed (and possibly corrupted) label. We have applied $noise\% = 10\%, 20\%$ and 30% levels only in the training partitions to simulate from low to high noisy scenarios. Since the true labels are known, we can later examine the performance of the preprocessing approaches in term of the well and wrongly filtered instances.

All the algorithms are run using run a 10-fold cross validation scheme (10-fcv). For all the training partitions, three different noisy versions are generated (with different seeds) for each noise level. Therefore, we obtain 30 executions per dataset and noise level.

Table 1 shows the names of datasets, their number of instances, attributes, and classes. In the last column, we present the number of features which present monotonic relation with the class, using for this the RMI measure [61]. This metric is calculated using each feature and the class and takes values in the range $[-1; 1]$, where a -1 means that the relationship is totally inverse (if the feature

[1] http://www.keel.es/datasets.php.

increases, the class decreases), and a 1 represents a completely direct relation (if the feature increases, the class increases). When the RMI value is in the range [−0.1; 0.1], we consider that there is no relation between the feature and the class. Counting features whose RMI value is out of that range, we calculate the number of features with order respect to the class. If all the features present order, we refer that the data set presents total ordering. In case of that number is lower than the total number of features, the data set presents partial order. In this paper, instances having missing values have been ignored.

4.2. Evaluation metrics

In order to compare the four monotonic filters, we will use five metrics commonly employed in the monotonic classification field. They are listed as follows:

- Mean Absolute Error (MAE), is calculated as the sum of the absolute values of the errors and then dividing it by the number of classifications. Various studies conclude that MAE is one of the best performance metrics in ordered classification [62,63].
- Accuracy (ACC) is computed as the percentage of correctly classified instances. Is a traditional measure in the classification topic that we include as a reference metric.
- Non-Monotonicity Index 1 (NMI1) [64], is defined as the number of clash-pairs divided by the total number of pairs of examples in the data set:

$$\text{NMI1} = \frac{1}{n(n-1)} \sum_{x \in D} \text{NClash}(x) \tag{7}$$

  where x is an example from the data set D. NClash(x) is the number of examples from D that do not meet the monotonicity restrictions (or clash) with x and n is the number of instances in D.
- Non-Monotonicity Index 2 (NMI2) [26], is defined as the number of non-monotone examples divided by the total number of examples:

$$\text{NMI2} = \frac{1}{n} \sum_{x \in D} \text{Clash}(x) \tag{8}$$

  where Clash(x) = 1 if x clashes with some examples in D, and 0 otherwise. If Clash(x) = 1, x is called a non-monotone example.
- Non-Comparable. This is a metric related to the number of pairs of non-comparable instances in the data set. Two instances x and x' are non-comparable if they do no satisfy $x \preceq x' \wedge x \neq x'$. This measure is also considered due to the fact that for some monotonic classifiers, it is harder to construct accurate models agreeing the number of non-comparable pairs raises.
- *Size* of the subset selected using the noise filtering algorithms. We include it to analyze the noise removing capabilities of each method.

4.3. Parameters configuration

See Table 2. The parameters of the four techniques that are the same as the original algorithms have been fixed following the suggestions of their authors. These original parameters are related to the neighborhood size or the number of partitions (i.e. base classifiers used), which have a great impact in the computational cost of the filtering methods. We understand that the authors of the standard classification versions obtained a compromise between the noise detection accuracy and the computational complexity that can be translated to our monotonic problems.

Table 2
Parameters considered for the algorithms compared. Underlined parameters has been optimized by grid search. The others have been fixed following the suggestions of the original authors.

Algorithm	Parameters
MENN	$k = 3$
MRNGE	firstOrderEdition = true
MIPF	numberPartitions = 5, consensus filter
	<u>confidence = 0.25, 2 items per leaf</u>
MINFFC	numberPartitions = 3, majority filter
	$k = 3$, <u>threshold= 0</u>
	<u>confidence = 0.25, 2 items per leaf</u>
M*k*NN	$k = 3$, distance = euclidean
OLM	modeResolution = conservative
	modeClassification = conservative
OSDL	classificationType = media, balanced = No
	weighted = No, tuneInterpolationParameter = No,
	lowerBound = 0, upperBound = 1
	interpolationParameter = 0.5, interpolationStepSize = 10
MID	confidence = 0.25, 2 items per leaf, R = 1

4.4. Statistical procedures

Several hypothesis testing procedures are considered to determine the most relevant differences found among the methods [65]. The use of nonparametric tests is preferred over parametric ones, since the initial conditions that guarantee the reliability of the later may not be satisfied. Friedman statistical test, a multiple comparison tests, is used to contrast the behavior of the algorithms [66] by ranking them and showing which are significantly different than the best thanks to Holm's posthoc test.

5. Analysis on the usage of monotonic filters to remove non-monotonic instances

This section is devoted to analyzing the results obtained, providing a summary of results including graphics and statistical outcome. We present the results considering two perspectives:

1. We compare the behavior of the algorithms using prediction quality measures MAE and Accuracy. In addition to the noise removal algorithms, we include the results obtained using the original datasets as input and datasets after relabeling the training partitions (keeping the tests partitions as they are). As we mentioned before, the relabeling is introduced to study its behavior in the real data when new unseen examples have to be classified using as input the relabeled training sets. The relabeling used is the optimal proposal described in [18].
   From the original data, one can create a graph representing the monotonicity violations between the instances. The instances correspond to vertex, and the violations are edges. A subset of the vertexes of a graph is an independent set if there are not two vertices in the subset that are adjacent [25]. In a monotonic violation graph, a maximum weights independent set corresponds to a monotone subset of the maximum size.
   Re-labeling the complement of the maximum independent set produces a completely monotonic set, with the fewest number of label changes in the instances. Although finding the maximum independent set is an NP-Hard problem, this is not the case for partial order graphs (comparability graphs or networks). In these graphs, the maximum independent set corresponds to a maximum antichain in the corresponding partial order and can be calculated in polynomial time by resolution of a minimum flow problem on a transportation network that is simply created from the comparability

Table 3
Average accuracy results for the filters and relabel with all the classifiers and each noise level. Best values are stressed in bold with respect to full precision results. Friedman rankings are provided indicating the best rank (control algorithm) in bold. When Holm's posthoc indicates a p-value < 0.05 the cell of the compared algorithm is grayed, whereas a p-value < 0.1 is indicated by underling the rank of the compared algorithm.

		Accuracy averages				Friedman's rankings			
	Preprocessing	0% (Original)	10%	20%	30%	0% (Original)	10%	20%	30%
MKNN	No preprocessing	0.69	0.65	0.60	0.55	3.00	3.18	3.55	3.64
	Relabelling	0.52	0.43	0.40	0.37	5.00	5.45	5.36	5.27
	MENN	0.61	0.58	0.54	0.51	3.36	3.50	3.50	3.86
	MRNGE	0.51	0.47	0.44	0.43	4.55	4.68	4.59	4.14
	MIPF	**0.71**	**0.70**	**0.69**	**0.65**	**1.91**	**1.59**	**1.55**	**1.64**
	MINFFC	0.69	0.68	0.66	0.63	3.18	2.59	2.45	2.45
MID	No preprocessing	0.72	0.69	0.65	0.61	2.45	2.68	2.82	2.73
	Relabelling	0.53	0.44	0.40	0.38	4.73	5.18	5.27	5.27
	MENN	0.60	0.58	0.55	0.51	3.95	3.77	3.77	4.14
	MRNGE	0.51	0.47	0.44	0.42	4.68	4.68	4.77	4.41
	MIPF	**0.73**	**0.71**	**0.69**	**0.65**	**1.91**	**1.68**	**1.82**	**1.91**
	MINFFC	0.69	0.67	0.66	0.63	3.27	3.00	2.55	2.55
OLM	No preprocessing	0.56	0.50	0.49	0.47	3.68	3.91	3.82	3.59
	Relabelling	0.49	0.41	0.37	0.35	4.45	5.00	4.91	4.82
	MENN	0.54	0.50	0.47	0.46	3.09	3.32	3.59	3.77
	MRNGE	0.45	0.40	0.38	0.37	4.14	4.68	4.41	4.68
	MIPF	**0.60**	**0.59**	**0.57**	**0.55**	**2.50**	**1.68**	**1.73**	**1.68**
	MINFFC	0.58	0.58	0.56	0.55	3.14	2.41	2.55	2.45
OSDL	No preprocessing	0.59	0.46	0.44	0.40	**2.64**	3.77	3.55	3.91
	Relabelling	0.49	0.39	0.37	0.35	4.45	4.64	4.73	4.36
	MENN	0.58	0.52	0.49	0.47	3.41	3.55	3.41	3.77
	MRNGE	0.51	0.45	0.42	0.42	3.68	4.05	4.32	4.05
	MIPF	**0.59**	**0.58**	**0.56**	0.54	3.00	**1.86**	**2.00**	**2.09**
	MINFFC	0.57	0.57	0.56	**0.54**	3.82	3.14	3.00	2.82

graph [67]. As can be noted, the monotone violation network is a comparability network.
This analysis is carried out in Section 5.1.

2. We study the algorithms using different metrics to study how the filtering and relabeling process affects the monotonic properties of the datasets: NMI1, NMI2, *Non-Comparable* and *Size*. Section 5.2 is devoted to study such measures.

5.1. Performance measures

In this section, we provide the accuracy and MAE results for all the classifiers and preprocessing strategies described above. These measures are computed over the test partitions after applying Relabeling, MENN, MRNGE, MIPF and MINFFC. We also include the absence of preprocessing, named as *No preprocessing*, to show the consequences of leaving an increasing amount of monotonic violations in the training set.

Table 3 shows the averaged accuracy values for all the datasets. The algorithm with the best value is stressed in bold. As we can appreciate, MIPF is the best performing technique on average, except for OSDL at 30% noise, where MINFFC is the best choice. It is interesting to note that Relabeling is not performing as well as expected, obtaining poor accuracy values when we introduce higher amounts of noise by forcing violations in the monotonic constraints.

On the right side of Table 3 the rankings of Friedman's test are shown. Friedman's test rejects the null-hypothesis in all cases. MIPF is again the best ranked algorithm, except for OSDL when we consider the original datasets. Relabeling is the worst ranked algorithm when we add any quantity of noise, suggesting that it is not a suitable technique in these environments. Grey cells depict when Holm's posthoc test indicates that there is a p-value below 0.05 rejecting the null hypothesis in favor of the control algorithm. The large amounts of shaded cells support the choice of MIPF as the best performing algorithm, while MINFFC is the only alternative that it is not statistically different.

Next, we include the averaged MAE results in Table 4 with the same format as that described for Table 3. In this case, MIPF is always the algorithm with the best average MAE, as well as the most robust choice as its MAE increases less than the other techniques as noise increases. MRNGE and Relabeling are the algorithms with the worst average MAE, being greatly affected by noise.

If we attend to the Friedman's test rankings and p-values, we may indicate that MIPF is the best choice overall. Again, Friedman's test rejects the null-hypothesis in all cases. Only not preprocessing for MID and OSDL and MINFFC in some cases are comparable to MIPF, as no statistical differences are found. However, the better performance of MIPF for all the four classifiers promotes it as the most recommendable strategy to apply in noisy environments for monotonic classification.

In summary, the application of a noise filtering stage based on MIPF is beneficial for all the classifiers considered. In particular, the combination of MID and MIPF seems to be the most robust combination, showing the best accuracy values across all noise levels. While some classifiers, as OSDL, are less affected by a previous preprocessing stage based on noise filtering, sensitive classifiers as MKNN take more advantage from noise filtering comparing No preprocessing against any other filtering technique. Nevertheless, MENN and MRNGE filters are not the best choices in noisy monotonic classification. In the next section, we will try to get some insights on why MIPF is able to attain better performance than the compared algorithms and why MENN and MRNGE perform poorly.

5.2. Monotonicity metrics

Table 5 is dedicated to the monotonic metrics considered. The table is structured into five columns, first for the name of the algorithm and others for the noise levels studied. The results, grouped by metric, are the average metric values of the 12 datasets for the

Table 4
Average MAE results for the filters and relabeling with all the classifiers and each noise level. Best values are stressed in bold with respect to full precision results. Friedman rankings are provided indicating the best rank (control algorithm) in bold. When Holm's posthoc indicates a p-value < 0.05 the cell of the compared algorithm is greyed, whereas a p-value < 0.1 is indicated by underling the rank of the compared algorithm.

		MAE averages				Friedman's rankings			
	Preprocessing	0% (Original)	10%	20%	30%	0% (Original)	10%	20%	30%
MKNN	No preprocessing	0.44	0.48	0.54	0.60	3.27	3.36	3.55	3.91
	Relabelling	0.75	0.86	0.93	0.97	4.73	5.36	5.27	5.09
	MENN	0.52	0.55	0.59	0.62	3.09	3.23	2.95	3.32
	MRNGE	0.89	0.97	1.00	1.03	4.73	4.86	4.86	4.59
	MIPF	**0.38**	**0.40**	**0.41**	**0.44**	**1.91**	**1.41**	**1.64**	**1.64**
	MINFFC	0.45	0.47	0.50	0.52	3.27	2.77	2.73	2.45
MID	No preprocessing	0.36	0.39	0.44	0.48	2.09	2.64	2.73	2.55
	Relabelling	0.74	0.84	0.92	0.95	4.86	5.36	5.27	5.18
	MENN	0.52	0.54	0.57	0.61	3.82	3.41	3.41	3.86
	MRNGE	0.89	0.97	1.03	1.06	5.05	5.05	5.05	4.95
	MIPF	**0.35**	**0.37**	**0.40**	**0.43**	**1.91**	**1.36**	**1.55**	**1.82**
	MINFFC	0.46	0.49	0.52	0.54	3.27	3.18	3.00	2.64
OLM	No preprocessing	0.68	0.76	0.77	0.80	3.68	3.91	3.36	3.32
	Relabelling	0.83	0.93	1.00	1.03	4.18	4.64	4.64	4.91
	MENN	0.69	0.75	0.79	0.80	3.00	3.14	3.50	3.41
	MRNGE	1.03	1.11	1.20	1.23	4.59	5.05	4.95	5.05
	MIPF	**0.57**	**0.57**	**0.62**	**0.62**	**2.41**	**1.59**	**1.82**	**1.77**
	MINFFC	0.65	0.67	0.68	0.67	3.14	2.68	2.73	2.55
OSDL	No preprocessing	0.54	0.66	0.68	0.72	**2.45**	3.14	3.09	3.27
	Relabelling	0.80	0.91	0.98	1.00	4.64	5.09	4.73	4.45
	MENN	0.57	0.63	0.65	0.67	3.14	3.45	3.41	3.77
	MRNGE	0.86	0.96	1.00	1.03	3.77	4.05	4.41	4.32
	MIPF	**0.54**	**0.54**	**0.57**	**0.59**	3.00	**1.86**	**2.09**	**2.18**
	MINFFC	0.58	0.59	0.60	0.62	4.00	3.41	3.27	3.00

Table 5
Average of the monotonicity metrics with respect to monotonic noise filtering algorithms.

	Preprocessing	0% (Original)	10%	20%	30%
NMI1	No preprocessing	0.021	0.024	0.026	0.028
	Relabeling	0.002	0.001	0.001	0.002
	MENN	0.005	0.007	0.008	0.009
	MRNGE	**0.001**	**0.001**	**0.000**	**0.000**
	MIPF	0.017	0.018	0.019	0.021
	MINFFC	0.015	0.015	0.014	0.015
NMI2	No preprocessing	0.649	0.822	0.858	0.876
	Relabeling	0.123	0.111	0.107	0.114
	MENN	0.385	0.467	0.495	0.524
	MRNGE	**0.045**	**0.026**	**0.019**	**0.016**
	MIPF	0.368	0.405	0.455	0.487
	MINFFC	0.263	0.257	0.245	0.263
Non-Comparable	No preprocessing	63265.24	67441.60	70829.14	74165.93
	Relabeling	60444.83	71906.22	75223.07	69595.63
	MENN	25806.34	10088.76	8691.59	8729.69
	MRNGE	**23132.81**	**8202.12**	**3830.44**	**3129.12**
	MIPF	38383.60	33834.46	30491.37	28302.81
	MINFFC	29075.15	22584.23	17492.62	13228.82
Size	No preprocessing	657.41	657.41	657.41	657.41
	Relabeling	657.41	657.41	657.41	657.41
	MENN	372.39	**277.89**	257.80	249.65
	MRNGE	**335.69**	278.98	**245.15**	**229.25**
	MIPF	537.28	503.53	477.15	453.03
	MINFFC	447.65	401.61	360.39	317.65

noise filtering algorithms, Relabeling and No preprocessing. The best result is stressed in bold.

All the metrics results in Table 5 are intrinsically related, but NMI1, NMI2 and *Non-Comparable* are specific for the monotonic classification problem. Observing *Size*, the highest reduction rates corresponds to MRNGE. Average NMI1 and NMI2 indicate the grade of monotonicity in a data set: we must take as reference value NMI1 and NMI2 values for *No preprocessing* at 0% noise level. It is clear that in original data set the values are higher, while the monotonic noise removal techniques introduced in this paper reduce them.

In Fig. 1 we present the boxplots for NMI2, as NMI1 shows very low variance and is much less descriptive. As can be seen, *No preprocessing* decreases its variance as noise increases, while its median raises the noise introduced. Relabeling achieves a stable behavior, obtaining the same median and variance for all noise levels. MENN work reasonably well without noise, but adding more violations makes MENN perform much worse in terms of the variance shown. Please note that MRNGE is also a similarity-based filter but, while MENN performs badly with noise, MRNGE achieves a low median and variance, which seems to be the best outcome so far. However, MRNGE is not the best

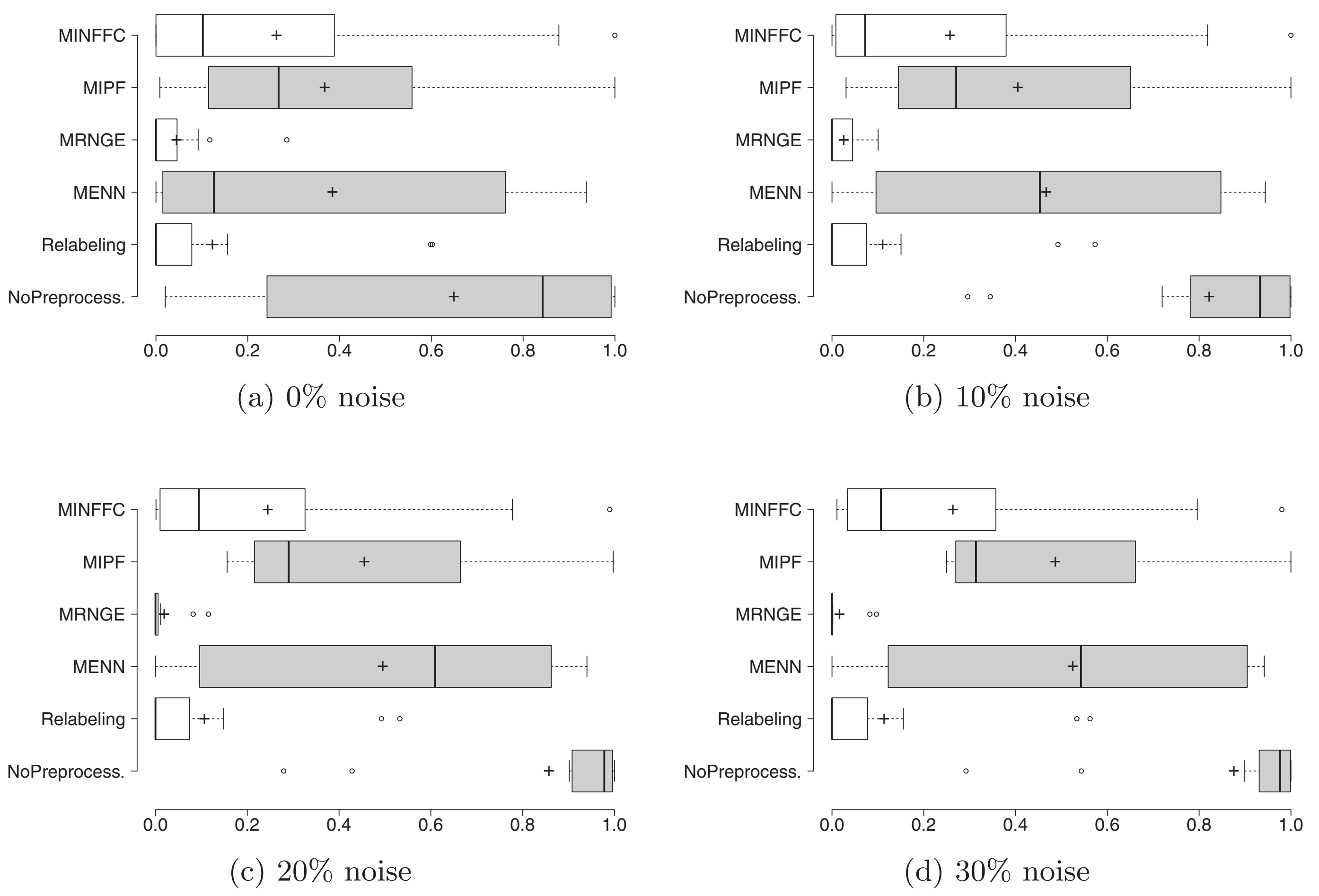


Fig. 1. NMI2 boxplots for each preprocessing technique in each noise level. Crosses indicate sample mean.

algorithm in terms of accuracy or MAE. If we pay attention to MIPF and MINFFC, which obtain better performance than MRNGE with worse NMI2 values, their boxplots show an almost constant variance while the median slightly increases as noise does. While the modification of the original data set by means of Relabeling aims to produce completely monotonic datasets with those rates equal to zero, it is interesting to note that MRNGE is able to reduce NMI1 and NMI2 even further than Relabeling.

We also want to pay attention to *Non-comparable* values, as they indicate the number of monotonic violations that remain in the dataset after applying the different preprocessing techniques. Fig. 2 depicts the boxplots associated to the *Non-comparable* values for each preprocessing approach. We can observe that *No preprocessing* and Relabeling show a similar variance but, while *No preprocessing* has a higher median, Relabeling achieves a low median value. The case of MENN and MRNGE is interesting, as they aim to reduce the number of violations as much as possible, and thus they achieve the best results for *Non-comparable*. MIPF and MINFCC reduce their variance as the noise increases, but MINFFC is more exaggerated in this behavior. Since MIPF and MINFFC are the best performing algorithms, we may conclude that extreme behaviors as those shown by Relabeling or MRNGE are not desirable: while the former does not solve most of the violations, the latter tends to remove too many instances to eliminate the violations and altering the information contained in the dataset.

At this point, MRNGE is the preprocessing technique that is able to obtain the lowest amounts of non-comparable instances. However, we observed in Section 5.1 that MRNGE is not the best performing algorithm. Since MRNGE also creates the most reduced datasets in terms of size, we may conclude that MRNGE is removing too many instances, which would lead to fewer violations of monotonic restrictions as shown by NMI1, NMI2 and *Non-comparable* values. This excessive removal will create an information loss in the dataset that penalizes the model obtained and thus showing poor performance in Accuracy and MAE values.

An alternative way to analyze the behavior of the different filters would be to examine how accurate is their noisy instances identification. Fig. 3 shows the percentages of good and bad decisions of each noise filter, both in removing and keeping the instances in the dataset. Since we need to know the corrupted instances to examine whether they were removed or not, we can only create these graphics for 10, 20 and 30% noise levels. As can be seen, MRNGE and MENN can eliminate all the noisy instances at 10% noise level, but they also eliminate a large portion of the correct instances. On the other hand, MIPF keeps some noisy instances in the dataset at 10% noise level, but it is able to keep more correct instances than the others.

As the noise increases, MENN and MRNGE lose the ability to identify the correct examples, while they keep removing instances to nearly clear all the violations induced in the dataset. At 30% noise level, both MENN and MRNGE cannot keep enough correct instances in the dataset as MIPF does. MINFFC is able to maintain a larger proportion of correct instances than MENN and MRNGE, but not as many as MIPF does. The ability of MIPF of maintaining not noisy instances causes that it enables the classifiers to obtain the best results in accuracy and MAE, even when it has the

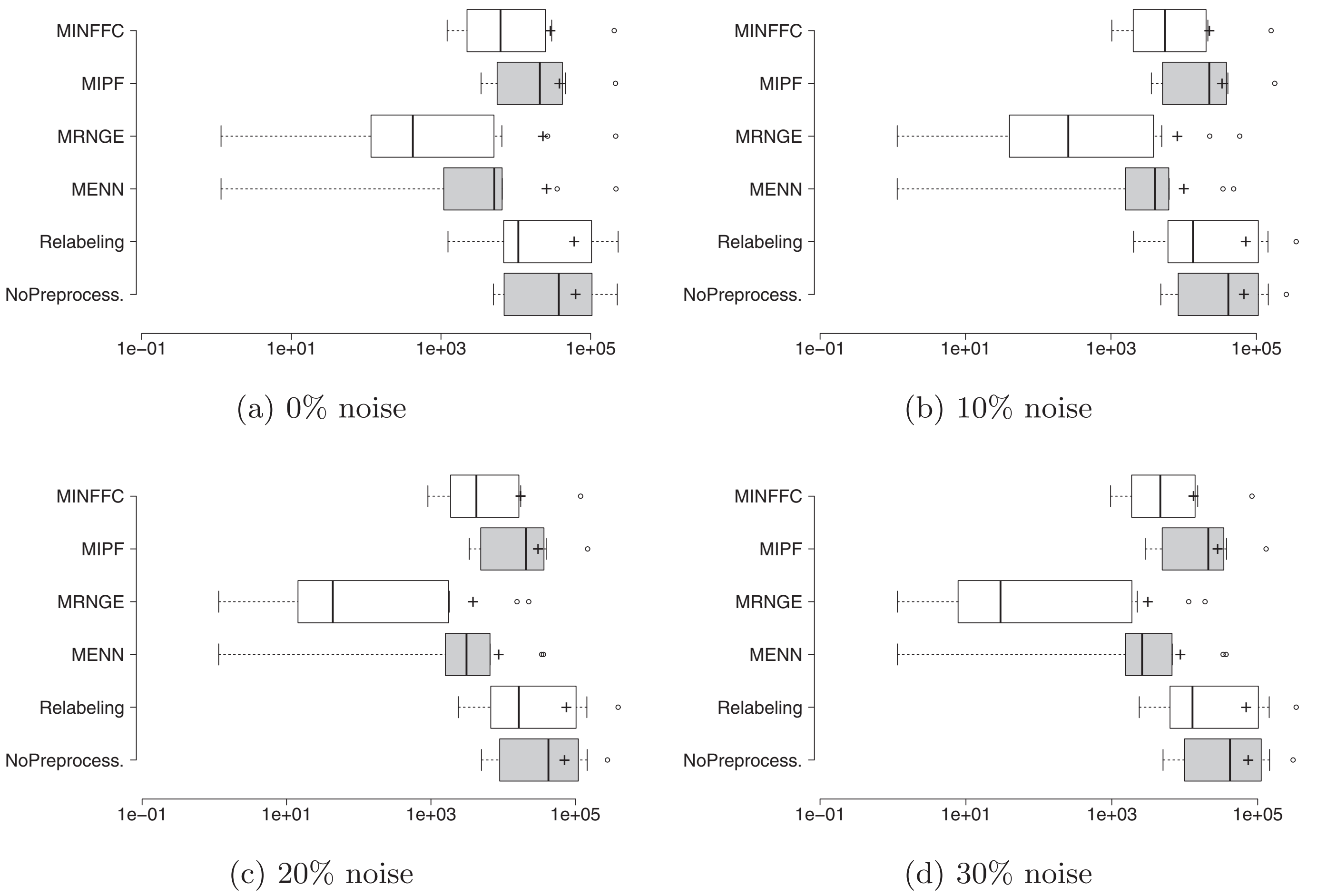


Fig. 2. *Non-comparable* boxplots for each preprocessing technique in each noise level. *X* axis is in log scale. Crosses indicate sample mean.

largest amount of noisy instances compared to the other filtering algorithms.

We can observe that one big problem of some filters, MENN and MRNGE, is their over-filtering behavior. This problem was already detected in standard classification and motivated the proposal of advanced noise filters instead of just applying instance selection methods. The leading solutions to this problem were the iterative removal of noise since cleaner datasets will help to posterior accurate filtering steps, and to establish a minimum amount of noise to be removed since no classifier can perfectly detect noisy instances and some false positives will always appear. In this work we show that accurate monotonic classification filters also need to apply these mechanisms, as MIPF and MINFFC does, to avoid over-filtering in the dataset.

Finally, we must point out that the usage of monotonicity metrics alone cannot describe the ability of noise preprocessing algorithms in monotonic classification, as they can be largely minimized by removing too many instances as MRNGE does. Maintaining a good proportion of clean instances is crucial to enable the classifiers to obtain generalizable models. MIPF is the best approach analyzed in this respect.

6. Experimental results and analysis on the benchmark data set: Winequality-red

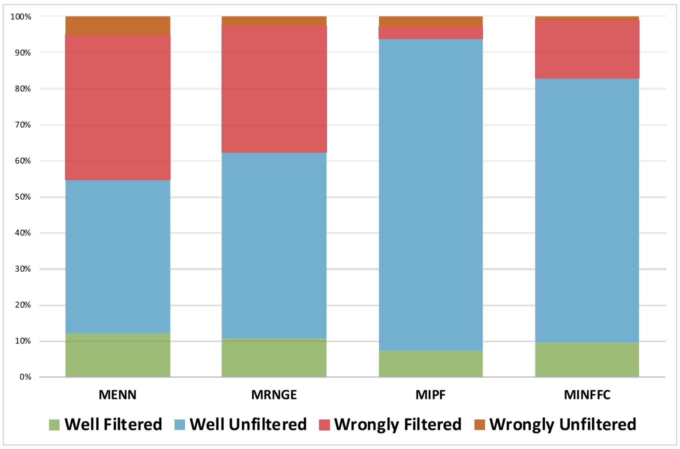
In this section, we apply the best combination filter-classifier analyzed (MIPF + MID) in Winequality-red, one of the benchmark datasets considered in the study in Section 4. The goal is to analyze the effect of the filter process in the predictive models generated by the monotonic classifiers.

The Winequality-red data set was introduced in [68] and can be found in the KEEL data set repository [59]. It is related to a red variant of the Portuguese Vinho Verde wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

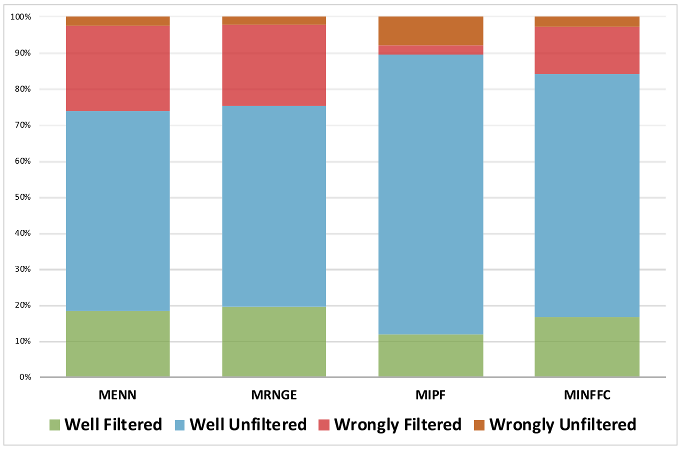
The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones). The data set consists of a sample of 1599 wines, described by 11 attributes and classified as 11 levels of quality. The attributes are:

- Fixed Acidity (Fix), with values in the range [4.6,15.9].
- Volatile Acidity (Vol), with values in the range [0.12,1.58].
- Citric Acid (Cit), with values in the range [0.0,1.0].
- Residual Sugar (Res), with values in the range [0.9,15.5].
- Chlorides (Chl), with values in the range [0.012,0.611].
- Free Sulfur Dioxide (Fre), with values in the range [1.0,72.0].
- Total Sulfur Dioxide (Tot), with values in the range [6.0,289.0]
- Density (Den), with values in the range [0.990,1.003].
- PH (Ph), with values in the range [2.74,4.01].
- Sulphates (Sul), with values in the range [0.33,2.0].
- Alcohol (Alc), with values in the range [8.4,14.9].

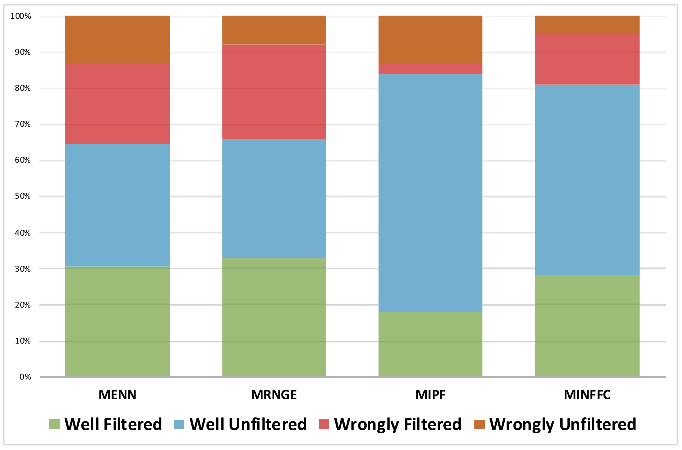
Real life datasets usually contain instances (wines in this case) which do not satisfy the monotonicity constraints. Table 6 presents two instances of the Winequality-red data set which produce a



(a) 10% noise



(b) 20% noise



(c) 30% noise

Fig. 3. Percentages of the noise filters regarding to the successful/wrongly filtered instances for each noise level. Blue and green indicate correct decisions, while orange and red are related to wrong actions. The higher the sum of blue and green areas, the better.

Table 6
Two instances which break the monotonicity constraint in Winequality-red data set.

Fix	Vol	Cit	Res	Chl	Fre	Tot	Den	Ph	Sul	Alc	Class
13.50	0.53	0.79	4.80	0.12	23.0	77.0	1.00	3.18	0.77	13.00	5
11.20	0.28	0.56	1.90	0.07	17.0	60.0	0.99	3.16	0.58	9.80	6

Table 7
Results in Winequality-red data set for MID and MIPF+MID.

	NMI1	NMI2	Non-comp	Size	MAE	#Branches
MID	0.00032	0.31345	331872	1439.1	0.46216	344
MIPF+MID	0.00030	0.29717	311433.75	1404.4	0.47275	321

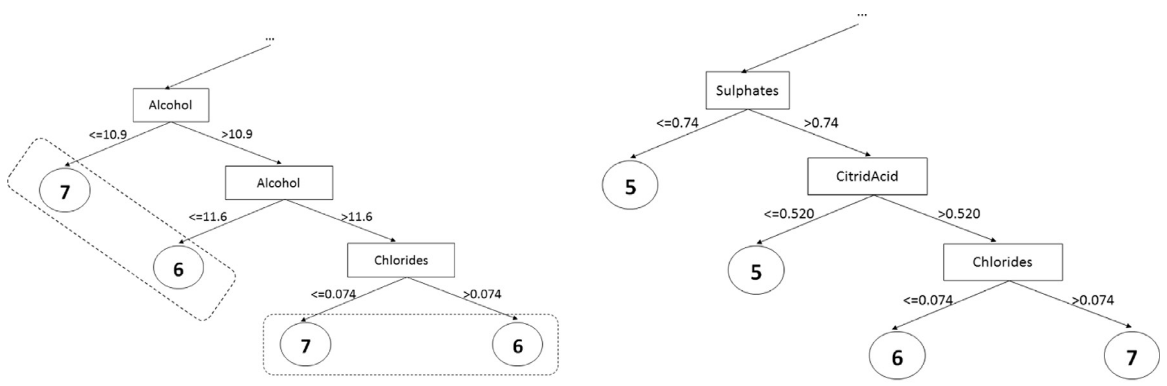
monotonicity collision between them. Lower values in features conclude with better qualification.

In this dataset, there are 471 instances with one or more monotonic collisions among them, which significantly affects to the monotonicity of the prediction models.

The filter methods are necessary to reduce these number of collisions, thus the generated model is able to keep its prediction capabilities while the monotonicity restrictions are taken into account.

To analyze this situation, the data set has been evaluated following a 10-fcv using MID, and MIPF+MID. The average performance they offered appears in Table 7. As the analysis in the previous section reflects, the best prediction models are produced by the MID algorithm, while its combination with MIPF keeps similar prediction capabilities improving all the monotonicity metrics, and reducing the size of the model (number of rules).

The removed instances produce many differences in the decision trees generated by the MID algorithm, as we can see in Fig. 4 where we plot a portion of the tree extracted from the same partition of Winequality-red data set by MID, without and with MIPF filter. The dotted boxes in Fig. 4a reflect violations of the monotonicity constraints. As it can be seen in Table 7 and Fig. 4, the structure of the decision tree changes significantly in the number of branches (last column in Table 7), antecedent order and



(a) MID without MIPF (b) MID with MIPF

Fig. 4. Decision trees extracted from Winequality-Red using MID with and without MIPF (dotted boxes reflect violations of the monotonicity constraints).

partition range, which transforms the meaning of the rules. In addition, the filter method is able to resolve many of the monotonicity conflicts which already appear using the MID classifier isolate.

7. Conclusions

In this paper, we have proposed the use of noise filtering algorithms as a preprocessing stage to decrease the monotonicity violations present in the original data. We have analyzed four noise removal algorithms, adapted to the monotonic domain, using different prediction rates and metrics over a high number of datasets, coming from standard classification and regression problems. The main conclusions related to the analyzed algorithms are:

- Monotonic noise removal algorithms are able to remove instances which negatively affect to the monotonicity of real data, altering the lowest possible the concepts represented in the original data and improving the efficiency and efficacy of the monotone classifiers.
- Relabeling is not able to deal with noisy environments, as its premises are skewed by the corrupted instances, wrongly relabeling instances and creating data shifting in the training set with respect to the test partition.
- Monotonic Iterative Partitioning Filtering (MIPF) is able to preserve and even to improve the prediction performances offered by classical monotonic classifiers such as M*k*NN, OLM, OSDL and MID.
- In the particular case of Winequality-red analyzed as an example, MIPF affects positively to the monotonicity constraints associated with the models extracted by MID.

We have also shown that monotonicity metrics cannot describe what constitutes a good filtering, as they can be biased by removing too many instances. While we have shown that filtering can greatly help to diminish the impact of noisy instances in monotonic classification, there is still promising options to explore: although Relabeling is not designed to work with noisy instances, correct reparation of an instance can greatly help to improve even further the results of this work. Since the monotonicity metrics can deceive the noise filters, other measures can be designed to avoid the greedy removal of preprocessing techniques.

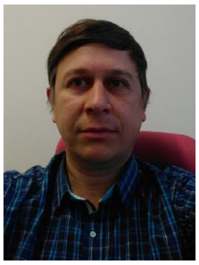
Acknowledgments

This work was supported by TIN2014-57251-P and TIN2017-89517-P, by the Spanish "Ministerio de Economía y Competitividad" and by "Fondo Europeo de Desarrollo Regional" (FEDER) under Project TEC2015-69496-R and the Project BigDaP-TOOLS – Ayudas Fundación BBVA a Equipos de Investigación Científica 2016.

References

[1] J. Pinto da Costa, H. Alonso, J. Cardoso, The unimodal model for the classification of ordinal data, Neural Netw. 21 (1) (2008) 78–91.
[2] M. Cruz-Ramírez, C. Hervás-Martínez, J. Sánchez-Monedero, P.A. Gutiérrez, Metrics to guide a multi-objective evolutionary algorithm for ordinal classification, Neurocomputing 135 (2014) 21–31.
[3] P.A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernandez-Navarro, C. Hervás-Martínez, Ordinal regression methods: survey and experimental study, IEEE Trans. Knowl. Data Eng. 28 (1) (2016) 127–146.
[4] A. Ben-David, L. Serling, Y. Pao, Learning and classification of monotonic ordinal concepts, Comput. Intell. 5 (1989) 45–49.
[5] B. Malar, R. Nadarajan, Evolutionary isotonic separation for classification: theory and experiments, Knowl. Inf. Syst. 37 (3) (2013) 531–553.
[6] K. Antoniuk, V. Franc, V. Hlaváč, V-shaped interval insensitive loss for ordinal classification, Mach. Learn. 103 (2) (2016) 261–283.
[7] W. Kotlowski, R. Slowiński, On nonparametric ordinal classification with monotonicity constraints, IEEE Trans. Knowl. Data Eng. 25 (11) (2013) 2576–2589.
[8] C.-C. Chen, S.-T. Li, Credit rating with a monotonicity-constrained support vector machine model, Expert Syst. Appl. 41 (16) (2014) 7235–7247.
[9] A. Ben-David, Monotonicity maintenance in information theoretic machine learning algorithms, Mach. Learn. 19 (1995) 29–43.
[10] R. Potharst, J. Bioch, Decision trees for ordinal classification, Intell. Data Anal. 4 (2000) 97–111.
[11] K. Cao-Van, B. De Baets, Growing decision trees in an ordinal setting, Int. J. Intell. Syst. 18 (2003) 733–750.
[12] W. Kotłowski, R. Słowiński, Rule learning with monotonicity constraints, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 537–544.
[13] C. Marsala, D. Petturiti, Rank discrimination measures for enforcing monotonicity in decision tree induction, Inf. Sci. 291 (2015) 143–171.
[14] J. Alcalá-Fdez, R. Alcalá, S. González, Y. Nojima, S. García, Evolutionary fuzzy rule-based methods for monotonic classification, IEEE Trans. Fuzzy Syst. 25 (6) (2017) 1376–1390.
[15] H. Daniels, M. Velikova, Monotone and partially monotone neural networks, IEEE Trans. Neural Netw. 21 (6) (2010) 906–917.
[16] H. Zhu, E.C. Tsang, X.-Z. Wang, R.A.R. Ashfaq, Monotonic classification extreme learning machine, Neurocomputing 225 (2017) 205–213.
[17] S. Lievens, B. De Baets, K. Cao-Van, A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting, Ann. Oper. Res. 163 (2008) 115–142.
[18] W. Duivesteijn, A. Feelders, Nearest neighbour classification with monotonicity constraints, in: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD, in: Lecture Notes in Computer Science, 1, Springer, 2008, pp. 301–316. 5211
[19] J. García, A.M. AlBar, N.R. Aljohani, J.-R. Cano, S. García, Hyperrectangles selection for monotonic classification by using evolutionary algorithms, Int. J. Comput. Intell. Syst. 9 (1) (2016) 184–201.
[20] J. García, H.M. Fardoun, D.M. Alghazzawi, J.-R. Cano, S. García, Mongel: monotonic nested generalized exemplar learning, Pattern Anal. Appl. 20 (2) (2017) 441–452.
[21] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn. Syst. 25 (5) (2014) 845–869.
[22] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowl. Inf. Syst. 38 (1) (2014) 179–206.

[23] A. Feelders, Monotone relabeling in ordinal classification, in: Proceeedings of IEEE International Conference on Data Mining, ICDM, 2010, pp. 803–808.
[24] R. Potharst, A. Ben-David, M.C. van Wezel, Two algorithms for generating structured and unstructured monotone ordinal data sets, Eng. Appl. Artif. Intell. 22 (4–5) (2009) 491–496.
[25] M. Rademaker, B. De Baets, H. De Meyer, Optimal monotone relabelling of partially non-monotone ordinal data, Optim. Methods Softw. 27 (1) (2012) 17–31.
[26] I. Milstein, A. Ben-David, R. Potharst, Generating noisy monotone ordinal datasets, Artif. Intell. Res. 3 (1) (2014) 30–37.
[27] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[28] J. Sánchez, F. Pla, F. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recognit. Lett. 18 (1997) 507–513.
[29] T. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, J. Comput. Sci. Technol. 22 (2007) 387–396.
[30] 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, Inf. Fusion 27 (2016) 19–32.
[31] T. Tran, D. Phung, W. Luo, S. Venkatesh, Stabilized sparse ordinal regression for medical risk stratification, Knowl. Inf. Syst. 43 (3) (2015) 555–582.
[32] A. Feelders, M. Velikova, H. Daniels, in: Two Polynomial Algorithms for Relabeling Non-monotone Data, 2006.
[33] S. Lievens, B. De Baets, Supervised ranking in the weka environment, Inf. Sci. 180 (24) (2010) 4763–4771.
[34] J. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[35] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artif. Intell. Rev. 22 (2004) 177–210.
[36] J. Bootkrajang, A. Kabán, Learning kernel logistic regression in the presence of class label noise, Pattern Recognit. 47 (11) (2014) 3641–3655.
[37] A. Ghosh, N. Manwani, P. Sastry, Making risk minimization tolerant to label noise, Neurocomputing 160 (2015) 93–107.
[38] N.D. Lawrence, B. Schölkopf, Estimating a kernel fisher discriminant in the presence of label noise, in: Proceedings of International Conference on Machine Learning, ICML, 1, 2001, pp. 306–313.
[39] C. Bouveyron, S. Girard, Robust supervised classification with mixture models: learning from data with uncertain labels, Pattern Recognit. 42 (11) (2009) 2649–2658.
[40] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Mach. Learn. 40 (2) (2000) 139–157.
[41] Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, J. Song, Rboost: label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners, IEEE Trans. Neural Netw. Learn. Syst. 27 (11) (2016) 2216–2228.
[42] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, J. Artif. Intell. Res. 11 (1999) 131–167.
[43] T.M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, J. Comput. Sci. Technol. 22 (2007) 387–396.
[44] S. Verbaeten, A.V. Assche, Ensemble methods for noise elimination in classification problems, in: Proceedings of the Fourth International Workshop on Multiple Classifier Systems, Springer, 2003, pp. 317–325.
[45] C.-M. Teng, Correcting noisy data, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, pp. 239–248.
[46] B. Nicholson, V.S. Sheng, J. Zhang, Label noise correction and application in crowdsourcing, Expert Syst. Appl. 66 (2016) 149–162.
[47] S. Cuendet, D.Z. Hakkani-Tr, E. Shriberg, Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech, in: A. Popescu-Belis, S. Renals, H. Bourlard (Eds.), Proceedings of International Workshop on Machine Learning for Multimodal Interaction, MLMI, Lecture Notes in Computer Science, 4892, Springer, 2007, pp. 144–155.
[48] A.L.B. Miranda, L.P.F. Garcia, A.C.P.L.F. Carvalho, A.C. Lorena, Use of classification algorithms in noise detection and elimination, in: E. Corchado, X. Wu, E. Oja, Ã. Herrero, B. Baruque (Eds.), Proceedings of International Conference on Hybrid Artificial Intelligence Systems, HAIS, Lecture Notes in Computer Science, 5572, Springer, 2009, pp. 417–424.
[49] S. García, J. Luengo, F. Herrera, Tutorial on practical tips of the most influential data preprocessing algorithms in data mining, Knowl.-Based Syst. 98 (2016) 1–29.
[50] S. García, J. Derrac, J.-R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 417–435.
[51] J.-R. Cano, S. García, F. Herrera, Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes, Pattern Recognit. Lett. 29 (2008a) 2156–2164.
[52] J.-R. Cano, F. Herrera, M. Lozano, S. García, Making CN2-SD subgroup discovery algorithm scalable to large size data sets using instance selection, Expert Syst. Appl. 35 (4) (2008b) 1949–1965.
[53] D. Han, Y. Hu, G. Wang, Uncertain graph classification based on extreme learning machine, Cogn. Comput. 7 (3) (2015) 346–358.
[54] 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, Inf. Sci. 291 (2015) 184–203.
[55] I. Triguero, J.A. Sáez, J. Luengo, S. García, F. Herrera, On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification, Neurocomputing 132 (2014) 30–41.
[56] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.
[57] E. Frank, M. Hall, A Simple Approach to Ordinal Classification 2167 (2001) 145–156. Lecture Notes in Computer Science
[58] K. Bache, M. Lichman, in: UCI Machine Learning Repository, 2013. http://archive.ics.uci.edu/ml
[59] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult-Valued Logic Soft Comput. 17 (255–287) (2010) 11.
[60] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesus, L. Sánchez, F. Herrera, KEEL 3.0: an open source software for multi-stage analysis in data mining, Int. J. Comput. Intell. Syst. 10 (2017) 1238–1249.
[61] Q. Hu, X. Che, L. Zhang, D. Zhang, M. Guo, D. Yu, Rank entropy-based decision trees for monotonic classification, IEEE Trans. Knowl. Data Eng. 24 (11) (2012) 2052–2064.
[62] L. Gaudette, N. Japkowicz, Evaluation Methods for Ordinal Classification 5549 (2009) 207–210. Lecture Notes in Computer Science
[63] N. Japkowicz, M. Shah, Evaluating Learning Algorithms. A Classification Perspective, Cambridge University Press, 2014.
[64] H. Daniels, M. Velikova, Derivation of monotone decision models from noisy data, IEEE Trans. Syst. Man Cybern. – Part C 36 (2006) 705–710.
[65] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC, 2011.
[66] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (2010) 2044–2064.
[67] R.H. Möhring, Algorithmic aspects of comparability graphs and interval graphs, in: Graphs and Order, Springer, 1985, pp. 41–101.
[68] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553.



José Ramón Cano received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 1999 and 2004, respectively.

He is currently a Professor in the Department of Computer Science, University of Jaén, Jaén, Spain.

His research interests include data mining, data reduction, data complexity, interpretability-accuracytrade-off, motonic classification and evolutionary algorithms.

Julián Luengo received the M.S. degree in computer science and the Ph.D. from the University of Granada, Granada, Spain, in 2006 and 2011, respectively.

He currently acts as an Assistant Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada, Spain.

His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, noisy data, data complexity and fuzzy systems.

Salvador García received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Salvador García has published more than 70 papers in international journals (more than 50 in Q1), with more than 4500 citations, h-index 26, over 45 papers in international conference proceedings (data from Web of Science). He is a member of the editorial board of the "Information Fusion" (Elsevier), "Swarm and Evolutionay Computation" (Elsevier) and "AI Communications" (IOS Press) journals, and he is co-Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer). He is a co-author of the book entitled "Data Preprocessing in Data Mining" published by Springer. His research interests include data science, data preprocessing, Big Data, evolutionary learning, Deep Learning, metaheuristics and biometrics.

Dr. García has been given some awards and honors for his personal work or for his publications in and conferences, such as IFSA-EUSFLAT 2015 Best Application Paper Award and IDEAL 2015 Best Paper Award. He belongs to the list of the Highly Cited Researchers in the area of Computer Sciences (2014–2017): http://highlycited.com/ (Clarivate Analytics).