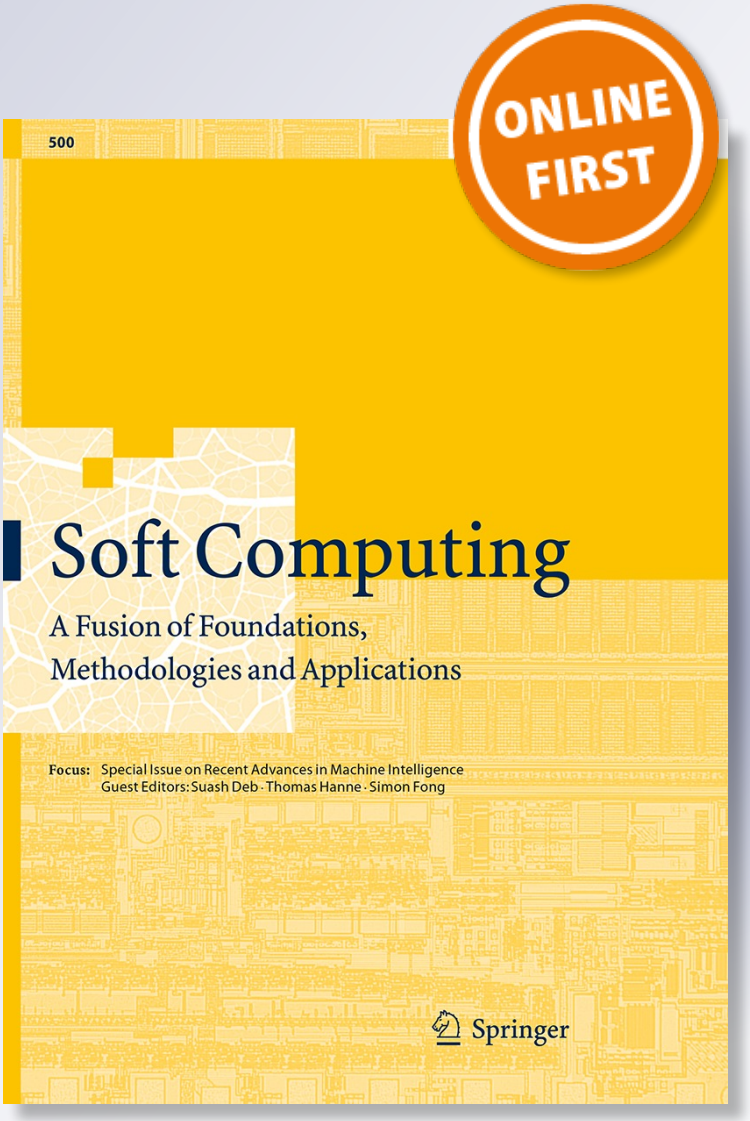
The influence of noise on the evolutionary fuzzy systems for subgroup discovery

J. Luengo, A. M. García-Vico, M. D. Pérez-Godoy & C. J. Carmona

Soft Computing
A Fusion of Foundations, Methodologies and Applications

ISSN 1432-7643

Soft Comput
DOI 10.1007/s00500-016-2300-1



Springer

Soft Comput
DOI 10.1007/s00500-016-2300-1

FOCUS

The influence of noise on the evolutionary fuzzy systems for subgroup discovery

J. Luengo[1] · **A. M. García-Vico**[2] · **M. D. Pérez-Godoy**[2] · **C. J. Carmona**[3]

Abstract External factors such as the presence of noise in data can affect the data mining process. This is a common problem that produces several negative consequences which involves errors in the data collection, preparation and, above all, in the results obtained by the data mining techniques employed. The capabilities of the models built under such circumstances will depend heavily on the quality of the training data. Hence, problems containing noise are complex problems and accurate solutions are often difficult to achieve. A particular supervised learning field like subgroup discovery has overlooked the analysis of noise and its impact on the descriptions obtained. This paper presents an analysis of the impact of noise on the most relevant evolutionary fuzzy systems for subgroup discovery. We also focus on how filtering techniques, devised for predictive tasks, may alleviate the impact of noise on descriptive fields such as subgroup discovery. Specifically, the analysis is carried out using recent filtering techniques for several class noise levels. The results obtained show two different behaviours, on the one hand, the SDIGA and NMEEFSD algorithms present a decrease in the quality of the subgroups when the noise is increased, making necessary the application of noise filtering in order to compensate for this loss of quality. On the other hand, the FuGePSD algorithm demonstrates its great capacity to work in noisy environments without the necessity of using a preliminary filter. The study is completed with an analysis of the interpretability under the influence of noise focused on the number of rules and variables.

Keywords Subgroup discovery · Class noise · Noise filters · Evolutionary fuzzy systems

Communicated by A. Herrero.

✉ C. J. Carmona
cjcarmona@ubu.es

J. Luengo
julianlm@decsai.ugr.es

A. M. García-Vico
agvico@ujaen.es

M. D. Pérez-Godoy
lperez@ujaen.es

[1] Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

[2] Department of Computer Science, University of Jaen, 23071 Jaén, Spain

[3] Department of Civil Engineering, Languages and Systems Area, University of Burgos, 09006 Burgos, Spain

1 Introduction

The main goal of data mining consists of extracting useful and valuable knowledge out of large amounts of raw data (Cherkassky and Mulier 2007). The relevance and interest of this knowledge is strongly influenced by the quality of the used data but, in this stage, a very common problem is the presence of noise in the dataset. Noise is usually found in real-world data, often described as corrupted data items not following the general distribution of the dataset, which can lead to excessively complex models with deteriorated performance (Wu and Zhu 2008) and thus harming the interpretations and knowledge that can be extracted as well as the decisions made based on them. The negative impact of noise has been widely studied (Zhu and Wu 2004) in classification and regression. Studies comparing models obtained with clean and noisy problems show a negative impact on the accuracy, building time, size and interpretability of the system. Hence, the treatment of noise has become one of the main tasks of data preprocessing (García et al. 2015).

Published online: 11 August 2016

Several approaches to dealing with noisy data and to diminishing its negative effects have been studied in the literature, focusing on classification problems. *Robust learners* (Bonissone et al. 2010) are characterized for being less influenced by noise, but designing a robust learner is not a trivial task. C4.5 (Quinlan 1993) is a classical example thanks to its pruning phase. Several studies (Teng 1999) claim that complete or partial noise correction using *data polishing methods* improves test performance results in comparison with no preprocessing, but this is only feasible in small datasets due to its high computational cost. Finally, the most popular choice is *noise filters* (Brodley and Friedl 1999; Khoshgoftaar and Rebours 2007), as they act as a preprocessing step, identifying and eliminating the noisy instances from the training data.

Since class noise corrupts the instances' labelling, any supervised task in data mining will be affected. As we have mentioned, the treatment of noise has focused on classification, by means of proposing advanced filters (Sáez et al. 2016), modification of well-known algorithms (Sun et al. 2016) or comparative analysis (Sluban et al. 2010). However, the effect of noise and the capabilities of descriptive algorithms in the presence of noise have mostly been overlooked, and since this framework of data mining also relies on supervised examples, the negative effects of noise cannot be ignored. Subgroup discovery (SD) (Herrera et al. 2011) is a descriptive data mining technique using supervised learning, i.e. it is a half-way between classification and description, where the knowledge is represented through rules. To the best of our knowledge, no analysis or study has been carried out to analyse the effect of noise and possible strategies to deal with it in SD tasks. The analysis of quality measures in SD is a key factor in order to observe the correct operation of the SD algorithms. The values of these quality measures will be affected by the presence of noise in the data.

In this study, we analyse the effects and treatment of noise in SD learning for evolutionary fuzzy systems (EFSs) and study different approaches to dealing with it. First, we focus on the negative effects of noise in SD and the quality measures commonly used. Next, we examine the possible approaches to dealing with the noise in SD and alleviating the negative effects it produces. Since noise filtering is a popular preprocessing step in supervised learning (Frénay and Verleysen 2014) that does not require any modification of the SD algorithms, we will use three recent filters for class noise: the ensemble filter, the cross-validated committees filter and the iterative-partitioning filter. Please note that most noise filter techniques were devised for predictive instead of descriptive tasks. Therefore, we are also interested in analysing the suitability of such filters in SD. The study is carried out with a wide number of datasets with different amounts of class noise, and an analysis with respect to interpretability, interest and trade-off between generality precision is presented. Moreover, this study is supported with statistical tests.

The rest of this paper is organized as follows. Section 2 introduces the background concepts of SD and EFSs, Sect. 3 presents the noise filters applied in this experimental analysis, Sect. 4 describes the experimental framework and Sects. 5 and 6 include the experimental results and their analysis; and the interpretability of the subgroups obtained, respectively. Finally, Sect. 7 presents some concluding remarks.

2 Preliminaries

This section introduces the main concepts used in this experimental study. First, the definition and main properties of the SD technique are outlined in Sect. 2.1. Next, the EFSs and the most relevant algorithms for SD based on this type of systems are summarized in Sect. 2.2.

2.1 Subgroup discovery

SD is a descriptive data mining technique whose main purpose is to explore relationships between different variables with respect to an interest variable, i.e. within of supervised learning. Introduced by Kloesgen (1996) and Wrobel (1997), it was defined thus (Wrobel 2001):

> In subgroup discovery, we assume we are given a so-called population of individuals (objects, customers, ...) and a property of those individuals we are interested in. The task of subgroup discovery is then to discover the subgroups of the population that are statistically "most interesting", i.e., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest.

The knowledge in this task is usually performed through the use of rules (Gamberger and Lavrac 2002; Lavrac et al. 2004a) where R can be formally defined as:

$$R : Cond \rightarrow TargetValue$$

In this way, $TargetValue$ is a value for the variable of interest (target variable) and $Cond$ is commonly a conjunction of features (attribute-value pairs) which is able to describe an unusual statistical distribution with respect to the $TargetValue$.

Throughout the literature, a wide range of algorithms have been presented which can be grouped with respect to the search strategy employed, e.g. based on beam search like CN2-SD (Lavrac et al. 2004b) and Apriori-SD (Kavsek and Lavrac 2006), exhaustive search like SD-Map (Atzmueller and Puppe 2006) and Merge-SD (Grosskreutz and Rueping 2009) for example, and evolutionary algorithms among others. The proposals based on evolutionary algorithms as a

search strategy are the main focus of this study and are analysed in the following section.

Other important elements for SD such as the target variable, the search strategy, the descriptive language of the subgroups and the study of quality measures used can be found in Carmona et al. (2014), Herrera et al. (2011).

2.2 Evolutionary fuzzy systems for subgroup discovery

A fuzzy system (Zadeh 1975) augmented with a learning process based on evolutionary algorithms (Eiben and Smith 2003) is defined as EFS as can be observed in Herrera (2008). In this definition, two concepts are presented: fuzzy systems and evolutionary algorithms. The former are usually considered in the form of fuzzy rule-based systems (FRBSs), which are composed of "IF-THEN" rules where both the antecedent and consequent can contain fuzzy logic statements. Fuzzy systems are based on fuzzy logic (Zadeh 1975), which already allow us to consider uncertainty, and also to represent the continuous variables in a manner which is close to human reasoning. In this way, interpretable fuzzy rules consider continuous variables as linguistic ones, where values are represented through fuzzy linguistic labels (LLs). The fuzzy set corresponding to each LL can be specified by the user or defined by means of uniform partitions if knowledge is not available. This simple and interpretable representation facilitates their application to real-world problems.

On the other hand, evolutionary algorithms are stochastic algorithms for optimising and searching based on a natural evolution process. These algorithms were introduced by Holland (1975). Different computational models can be found within these types of algorithms such as genetic algorithms (Goldberg 1989; Holland 1975), evolution strategies (Schwefel 1995), evolutionary programming (Fogel 1995), genetic programming (Koza 1992), amongst others.

SD is a rule learning process that can be seen as an approximation problem in which the objective is the learning of the parameters of the model. In this task, the search space can be very complex and the search strategy used becomes a key factor. The use of EFSs is very well suited to this task because these types of algorithms perform a global search in the space in a suitable way, as can be observed in the real-world problems solved in the literature, for example, in Bioinformatics with the description of the Influenza A virus (Carmona et al. 2013a) or a specific problem concerning an acute sore throat (Carmona et al. 2015), in Medicine with the description of patients' patterns in a psychiatric emergency department (Carmona et al. 2011), in E-commerce with the analysis of the usage of customers of a website based on the sale of olive oil (Carmona et al. 2012) or in Industry with the description of concentrating photovoltaic modules (Carmona et al. 2013b), amongst others.

The most relevant EFSs for SD are summarized below, and a complete description can be analysed in Carmona et al. (2014).

2.2.1 SDIGA

SDIGA (del Jesus et al. 2007) is an EFS based on a monoobjective evolutionary algorithm which follows the iterative rule learning (IRL) approach (Venturini 1993) whereby the solution of each iteration is the best individual obtained and the global solution is formed by the best individuals obtained in the different runs. In addition, SDIGA is executed for each value of the target variable because the consequent is not represented in the chromosome and in this way it obtains rules for all values of the target variable.

The individuals represent the antecedent part of the rule through the "$Chromosome = Rule$" approach and the core of SDIGA is an evolutionary algorithm using a postprocessing step based on a local search. This hybrid algorithm extracts one simple and interpretable fuzzy rule with an adequate level of interest for SD.

Fitness is an aggregation function where the selection of quality measures like coverage, significance, unusualness, accuracy, sensitivity, crisp support, fuzzy support, crisp confidence and fuzzy confidence are determined by the expert. The number of objectives within the weighted aggregation function are between 1 and 3.

2.2.2 NMEEFSD

NMEEFSD (Carmona et al. 2010) is an EFS based on a multiobjective evolutionary algorithm called NSGA-II (Deb et al. 2002). This algorithm encodes the solutions according to the "$Chromosome = Rule$" approach, where only the antecedent is represented in the chromosome and the consequent is prefixed to one of the possible values of the target variable in the evolution. In this way, the algorithm is executed as many times as the number of values for the target variable it contains.

The algorithm employs different genetic operators in order to promote generality and diversity within the population and to obtain interesting subgroups for the SD technique. It is very important to highlight the use of the multiobjective approach because it allows the experts the possibility to use different quality measures as objectives. In this way, the final Pareto front obtained by NMEEFSD is the set of non-dominated solutions with respect to the quality measures considered. As can be observed in Carmona et al. (2010), the best results for this algorithm are obtained with the use of the quality measures unusualness and sensitivity. Finally, a screening function is performed at the end of the evolutionary process in order to return only those solutions which reach a predetermined confidence threshold.

2.2.3 FuGePSD

FuGePSD (Carmona et al. 2015) is the most recent EFS presented in the literature. This algorithm is based on a genetic programming algorithm (Koza 1992) with the ability to extract descriptive fuzzy rules for the SD task. Contrary to the previous algorithms, FuGePSD represents the rules through the "$Chromosome = Rule$" approach including both the antecedent and the consequent of the rule. Specifically, FuGePSD employs the genetic cooperative-competition approach where rules of the population cooperate and compete among themselves in order to obtain the optimal solution. The inclusion of the target variable in the representation is often an advantage with respect to alternative evolutionary algorithms available for SD, since FuGePSD is executed only once, obtaining rules for the different values of the target variable.

Due to the nature of the algorithm, it is important to define two fitness criteria: (1) to compete and (2) to cooperate between individuals. The competition is performed through unusualness and the operator token competition which is key for the algorithm, and the cooperation is carried out through the normalized accuracy for all values of the target variable. As we have mentioned before, the token competition operator is a key factor of the algorithm because individuals with good niches will attempt to exploit that one alone and prevent further individuals from sharing its resources, unless a newer one is stronger that the one initially developed.

Finally, only the best rules of the evolutionary process pass through a screening function employed to obtain only rules with values greater than a given threshold of sensitivity and confidence.

3 Noise in data

A large amount of real-world datasets are affected by corruptions that hinder the analysis, interpretations and decisions concerning the models obtained. The data extraction process also has influence on the quality: incorrect input of the data and uncertainty-affected sources are two classic examples. In supervised problems, this becomes especially relevant as it harms the relationship that arises between the input and output attributes. Thus, noise has attracted much attention in classification and regression, where noise spoils the model and the knowledge extracted from data. We can point out two types of noise in the specialized literature (Zhu and Wu 2004):

- Class noise (Cao et al. 2012; Abellán and Masegosa 2012): it occurs when instances are incorrectly labelled. Data entry errors or inadequacy of the information used to label each instance are possible causes.
- Attribute noise (Teng 2004): it refers to alterations in the input attributes' values with respect to their original underlying distribution draws. It includes erroneous attribute values, but also missing values and "do not care" values are included in this category.

Class noise has been considered as the most disruptive type of noise (Sáez et al. 2014). For this reason, we focus on class noise for SD, as it will alter and hinder the description extracted in a greater way than attribute noise: it will cause loss of quality in subgroups because rules cover examples incorrectly labelled with a wrong class label.

Although in the literature there have been many studies dealing with noisy data and model improvement, the most popular choice are noise filters (Brodley and Friedl 1999; Khoshgoftaar and Rebours 2007), as they act as a preprocessing step, identifying the noisy instances and eliminating them from the training data. As we have mentioned, filters are mainly devised for classification tasks and thus they aim to remove examples to optimize the accuracy of predictive models. This objective may be well correlated with the metrics evaluated by SD algorithms, but this question has not been studied yet in the specialized literature. In this paper, we will address this issue by observing the effects of noise and the suitability of noise filters. Specifically, we will use three state-of-the-art class noise filters of different natures which are summarized in the following sections. The ensemble filter is based on an ensemble of three different classifiers, while the cross-validated committees filter generates partitions of the training set to build an ensemble of a single classifier over the different partitions. The iterative-partitioning filter is an iterative filter, where noise is identified in several steps until the stopping criterion is met.

3.1 Ensemble filter

The *ensemble filter* (EF) (Brodley and Friedl 1999) uses a set of learning algorithms to create classifiers in several subsets of the training data that serve as noise filters for the training set. The identification of potentially noisy instances is carried out by performing an Γ-FCV (fold cross-validation) on the training data with μ classification algorithms, called filter algorithms. In the experimentation developed for this contribution, we have utilized the three filter algorithms used by the authors in Brodley and Friedl (1999), which are C4.5, 1-NN and LDA (Mclachlan 2004).

The complete process carried out by EF is described below:

- Split the training dataset D_T into Γ equal sized subsets.
- For each one of the μ filter algorithms.

  - For each of these Γ parts, the filter algorithm is trained on the other $\Gamma - 1$ parts. This results in Γ different classifiers.
  - These Γ resulting classifiers are then used to tag each instance in the excluded part as either correct or mislabelled, by comparing the training label with that assigned by the classifier.
- At the end of the above process, each instance in the training data has been tagged by each filter algorithm.
- Add to D_N the noisy instances identified in D_T using a consensus voting scheme, taking into account the correctness of the labels obtained in the previous step by the μ filter algorithms.
- Remove the noisy instances from the training set: $D_T \leftarrow D_T \setminus D_N$.

3.2 Cross-validated committees filter

The *cross-validated committees filter* (CVCF) (Verbaeten and Assche 2003) uses ensemble methods in order to preprocess the training set to identify and remove mislabelled instances in classification datasets. CVCF is mainly based on performing an Γ-FCV to split the full training data and on building classifiers using decision trees in each training subset. The authors of CVCF place special emphasis on using ensembles of decision trees such as C4.5.

The basic steps of CVCF are the following:

- Split the training dataset D_T into Γ equal sized subsets.
- For each of these Γ parts, C4.5 (as suggested by the authors) is trained on the other $\Gamma - 1$ parts. This results in Γ different classifiers.
- These Γ resulting classifiers are then used to tag each instance in the training set D_T as either correct or mislabelled, by comparing the training label with that assigned by the classifier.
- Add to D_N the noisy instances identified in D_T using a voting scheme (the majority scheme in our experimentation), taking into account the correctness of the labels obtained in the previous step by the Γ classifier built.
- Remove the noisy instances from the training set: $D_T \leftarrow D_T \setminus D_N$.

3.3 Iterative-partitioning filter

The *iterative-partitioning filter* (IPF) (Khoshgoftaar and Rebours 2007) is a preprocessing technique based on the *partitioning filter* (Zhu et al. 2003). It is employed to identify and eliminate mislabelled instances in large datasets. Most noise filters assume that datasets are relatively small and capable of being learned after only one run, but this is not always true and partitioning procedures may be necessary.

IPF removes noisy instances in multiple iterations until a stopping criterion is reached. The iterative process stops if, for a number of consecutive iterations s, the number of identified noisy instances in each of these iterations is less than a percentage p of the size of the original training dataset. Initially, we have a set of noisy instances $D_N = \emptyset$ and a set of good data $D_G = \emptyset$. The basic steps of each iteration are:

- Split the training dataset D_T into Γ equal sized subsets.
- For each of these Γ parts, C4.5 is trained on this part as recommended by the authors. This results in Γ different trees.
- These Γ resulting classifiers are then used to tag each instance in the training set D_T as either correct or mislabelled, by comparing the training label with that assigned by the classifier.
- Add to D_N the noisy instances identified in D_T using majority voting, taking into account the correctness of the labels obtained in the previous step by the Γ classifier built.
- Remove the noisy instances and the good data from the training set: $D_T \leftarrow D_T \setminus \{D_N \cup D_G\}$.

At the end of the iterative process, the filtered data are formed by the remaining instances of D_T and the good data of D_G; that is, $D_T \cup D_G$.

4 Experimental framework

This section outlines the main details of the experimental study performed in order to analyse the suitability of noise filtering for SD. Section 4.1 summarizes the datasets analysed in the study and the processes to induce class noise into the original datasets and the methodology for the analysis of the results. Next, the quality measures used to measure the performance of the SD analysed in this experimental study and the parameters of all algorithms considered are presented in Sect. 4.2.

4.1 Datasets

The experimental study of this paper has been performed with a wide number of datasets. Specifically, we have used 37 datasets from the KEEL Repository[1] (Alcalá-Fdez et al 2011). The main characteristics of these datasets are summarized in Table 1 where the number of attributes and their types together the number of instances and classes for each dataset are shown.

The initial amount of class noise in the datasets is unknown, and thus, no assumptions about the noise level

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

Table 1 Datasets used to introduce noise, including the number of attributes and their type (real, integer or nominal), the number of examples and the number of classes for each one

Name	# Attributes (R/I/N)	# Examples	# Classes
Automobile	25 (15/0/10)	150	6
Balance	4 (4/0/0)	625	3
Banana	2 (2/0/0)	5300	2
Car	6 (0/0/6)	1728	4
Cleveland	13 (13/0/0)	297	5
Contraceptive	9 (0/9/0)	1473	3
Dermatology	34 (0/34/0)	358	6
Ecoli	7 (7/0/0)	336	8
Flare	11 (0/0/11)	1066	6
German	20 (0/7/13)	1000	2
Hayes-roth	4 (0/4/0)	160	3
Heart	13 (1/12/0)	270	2
Ionosphere	33 (32/1/0)	351	2
Iris	4 (4/0/0)	150	3
Led7digit	7 (7/0/0)	500	10
Magic	10 (10/0/0)	19020	2
Monk-2	6 (0/6/0)	432	2
Newthyroid	5 (4/1/0)	215	3
Nursery	8 (0/0/8)	12690	5
Page-blocks	10 (4/6/0)	5472	5
Penbased	16 (0/16/0)	10992	10
Phoneme	5 (5/0/0)	5404	2
Pima	8 (8/0/0)	768	2
Ring	20 (20/0/0)	7400	2
Segment	19 (19/0/0)	2310	7
Shuttle	9 (0/9/0)	58000	7
Sonar	60 (60/0/0)	208	2
Spambase	57 (57/0/0)	4597	2
Splice	60 (0/0/60)	3190	3
Thyroid	21 (6/15/0)	7200	3
Twonorm	20 (20/0/0)	7400	2
Vehicle	18 (0/18/0)	846	4
Vowel	13 (10/3/0)	990	11
Wdbc	30 (30/0/0)	569	2
Wine	13 (13/0/0)	178	3
Yeast	8 (8/0/0)	1484	10
Zoo	16 (0/0/16)	101	7

can be made. In order to control the amount of noise in each dataset, different class noise levels $x\%$ are introduced with the *uniform class noise* scheme (Teng 1999). Following this noise introduction procedure, $x\%$ of the examples are corrupted by randomly replacing their current class label with any other possible one, drawn from a discrete uniform distribution. New class noise datasets are generated from the original ones, considering the noise levels ranging from $x = 0\,\%$ (base datasets) to $x = 20\,\%$, by increments of $5\,\%$.

For each noise level, a noisy dataset from the original one is created following the procedure described below:

1. An amount of $x\%$ of uniform class noise is introduced into a copy of the full original dataset.
2. The original and the noisy dataset are partitioned into five equivalent folds: each fold will contain the same examples. Please note that the noisy copy will have a $x\%$ of labels changed.
3. For the noisy copy, the training partitions are maintained, whereas the test partitions will substitute the noisy examples by the original ones, thus obtaining a test partition that is noise-free.

The measures estimation of the algorithms in datasets is obtained by means of three runs of a stratified fivefold cross-validation. Five partitions are used since, if each partition has a large number of examples, the noise effects will be more notable, facilitating their analysis.

4.2 Quality measures used to evaluate the performance in subgroup discovery

The main property for SD algorithms is the obtaining of interesting, simple and interpretable subgroups, covering the majority of the examples of the interest property (target variable). Considering this definition, we analyse the most relevant quality measures for SD as can be observed in the guidelines presented in Carmona et al. (2014).

It is important to remark that the SD algorithms based on EFSs use fuzzy logic in order to represent continuous variables of the dataset, and it is necessary to present fuzzy subgroups in order to understand their analysis. In Eq. 1, the representation of a canonical fuzzy rule for SD can be observed:

$$R: \; IF \; X_1 = (LL_1^2) \;\; AND \;\; X_3 = (LL_3^1) \\ THEN \;\; TargetValue \qquad (1)$$

where:

- $X = \{X_m / m = 1, \ldots, n_v\}$ is a set of features used to describe the subgroups, and n_v is the number of descriptive features.
- $T = \{TargetValue / j = 1, \ldots, n_{tv}\}$ is a set of values for the target variable, and n_{tv} is the number of values for the target variable.
- $LL_{n_v}^{l_{n_v}}$ is the LL number l_{n_v} of the variable n_v.

The quality measures analysed are:

Table 2 Parameters of the algorithms

Algorithm	Parameters
SDIGA (SDI)	Fitness = (0.7 ∗ Sensitivity + 0.3 ∗ Unusualness); Linguistic labels = 3; Minimum confidence = 0.6; Population size = 100; Maximum evaluations = 10,000; Crossover probability = 0.60; Mutation probability = 0.01
NMEEFSD (NME)	Objective1 = Sensitivity; Objective2 = Unusualness; Linguistic labels = 3; Minimum confidence = 0.6; Population size = 50; Maximum evaluations = 10,000; Crossover probability = 0.60; Mutation probability = 0.10
FuGePSD (FuG)	Fitness = Unusualness; Linguistic labels = 3; Minimum confidence = 0.6; Minimum sensitivity = 0.6; Population size = 100; Maximum generations = 300; Crossover probability = 0.50; Mutation probability = 0.20; Insertion Probability = 0.15; Dropping Probability = 0.15; $w_1 = 0.7$; $w_2 = 0.15$; $w_3 = 0.15$; AllTargetValues = False

– *Unusualness* is the weighted relative accuracy of a rule (Lavrac et al. 1999) which measures interest and a trade-off between generality and precision. It can be computed as:

$$Unus(R_i) = \frac{n(Cond)}{n_s} \left(\frac{n(TargetValue \cdot Cond)}{n(Cond)} - \frac{n(TargetValue)}{n_s} \right) \quad (2)$$

It can be described as the balance between the coverage of the rule and its accuracy gain, where $n(Cond)$ is the number of examples which satisfy the conditions determined by the antecedent part of the rule, n_s is the number of total examples, $n(TargetValue_k \cdot Cond)$ is the number of examples which satisfy the conditions and also belong to the value for the target variable within the rule, and $n(TargetValue_k)$ are all the examples of the target variable. The domain of this quality measure is specified for each problem because there is a direct dependence with respect to the target variable.

– *Sensitivity* is the proportion of actual matches that have been classified correctly (Kloesgen 1996) and it has a component based on generality. It is computed as:

$$Sens(R_i) = \frac{n(TargetValue \cdot Cond)}{n(TargetValue)} \quad (3)$$

This quality measures can be found in the literature as the Support based on the examples of the class, Recall or $TPrate$, and its domain is [0, 1].

– *Fuzzy confidence* is an adaptation of the standard confidence measure for fuzzy rules (del Jesus et al. 2007). This quality measures obtain the precision of one subgroup and it is defined as:

$$FCnf(R_i) = \frac{\sum_{E^k \in E / E^k \in TargetValue} APC(E^k, R_i)}{\sum_{E^k \in E} APC(E^k, R_i)} \quad (4)$$

where APC (Antecedent Part Compatibility) is the degree of compatibility between an example and the antecedent component of a fuzzy rule, i.e. the degree of membership for the example to the fuzzy subspace delimited by the antecedent part of the rule. An example E^k verifies the APC of a rule if

$$APC(E^k, R_i) = T(\mu_{LL_1^1}(e_1^k), \ldots, \mu_{LL_{n_v}^{l_{n_v}}}(e_{n_v}^k)) > 0 \quad (5)$$

This experimental study uses the parameters presented in Table 2 for the algorithms SDIGA, NMEEFSD and FuGePSD following the recommendations of the authors. As we have mentioned previously, in view of the fact that all algorithms are stochastic, three runs are performed, and an average result from 15 values is shown for each dataset. Therefore, values of a set of rules in unusualness ($UNUS$), sensitivity ($SENS$) and fuzzy confidence ($FCNF$) are computed as the average for all rules.

5 The suitability of noise filtering for subgroup discovery algorithms

The main objective of this experimental study is to analyse whether using a filtering approach is beneficial for the performance of different EFSs for SD. For this reason, we compare the capabilities of these algorithms without filtering over the increasingly noisy versions of the base datasets.

Firstly, complete results obtained for the algorithms without filtering are presented in Tables 3, 4 and 5 for the unusualness, sensitivity and fuzzy confidence quality measures, respectively. As can be observed in these results:

Table 3 Results for the unusualness quality measure for the different algorithms analysed

Dataset	0 %			5 %			10 %			15 %			20 %		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Automobile	0.012	0.089	0.081	0.004	0.091	0.078	0.011	0.096	0.084	0.005	0.070	0.083	0.006	0.078	0.076
Balance	0.052	0.089	0.065	0.056	0.086	0.065	0.055	0.080	0.065	0.048	0.073	0.066	0.037	0.075	0.070
Banana	0.025	0.016	0.054	0.024	0.015	0.054	0.023	0.015	0.054	0.021	0.014	0.054	0.018	0.012	0.054
Car	0.021	0.101	0.096	0.013	0.096	0.096	0.009	0.089	0.100	0.004	0.086	0.100	0.005	0.083	0.096
Cleveland	0.018	0.119	0.102	0.014	0.118	0.102	0.014	0.118	0.101	0.008	0.107	0.104	0.011	0.107	0.101
Contraceptive	0.000	0.032	0.027	0.002	0.031	0.027	0.002	0.028	0.021	0.004	0.030	0.019	0.001	0.029	0.026
Dermatology	0.014	0.158	0.122	0.008	0.157	0.121	0.006	0.152	0.134	0.007	0.141	0.139	0.005	0.138	0.131
Ecoli	0.025	0.034	0.079	0.019	0.014	0.079	0.020	0.044	0.098	0.019	0.077	0.120	0.022	0.024	0.145
Flare	0.019	0.157	0.166	0.006	0.142	0.166	0.021	0.154	0.166	0.010	0.139	0.165	0.009	0.130	0.166
German	0.002	0.033	0.025	−0.001	0.034	0.025	0.003	0.027	0.027	0.001	0.032	0.027	0.001	0.025	0.028
Hayes-roth	0.041	0.077	0.027	0.040	0.073	0.027	0.040	0.072	0.013	0.040	0.068	0.063	0.032	0.060	0.025
Heart	0.061	0.116	0.086	0.040	0.107	0.086	0.033	0.110	0.084	0.026	0.098	0.082	0.048	0.090	0.086
Ionosphere	0.026	0.126	0.184	0.025	0.120	0.183	0.019	0.112	0.162	0.021	0.099	0.170	0.022	0.100	0.158
Iris	0.181	0.191	0.124	0.158	0.179	0.124	0.157	0.166	0.127	0.127	0.159	0.142	0.099	0.145	0.128
Led7digit	0.058	0.067	0.061	0.053	0.064	0.061	0.045	0.062	0.068	0.041	0.061	0.070	0.039	0.062	0.068
Magic	0.029	0.059	0.048	0.030	0.055	0.047	0.030	0.054	0.051	0.026	0.048	0.061	0.026	0.049	0.057
Monk-2	0.100	0.123	0.063	0.071	0.116	0.063	0.058	0.111	0.063	0.061	0.107	0.074	0.046	0.094	0.074
Newthyroid	0.057	0.109	0.076	0.063	0.098	0.076	0.063	0.099	0.080	0.061	0.087	0.075	0.073	0.084	0.055
Nursery	0.055	0.200	0.089	0.027	0.213	0.089	0.022	0.202	0.222	0.011	0.184	0.222	0.007	0.175	0.222
Page-blocks	0.001	0.014	0.033	0.001	0.014	0.034	0.001	0.013	0.044	0.001	0.010	0.042	0.001	0.011	0.039
Penbased	0.020	0.061	0.074	0.016	0.058	0.074	0.018	0.054	0.075	0.014	0.042	0.076	0.012	0.041	0.076
Phoneme	0.022	0.026	0.070	0.021	0.024	0.070	0.022	0.023	0.068	0.021	0.022	0.067	0.019	0.021	0.069
Pima	0.045	0.065	0.042	0.039	0.060	0.042	0.039	0.058	0.041	0.026	0.050	0.039	0.034	0.059	0.040
Ring	0.068	0.154	0.040	0.048	0.150	0.035	0.045	0.151	0.041	0.018	0.145	0.055	0.000	0.139	0.043
Segment	0.019	0.116	0.082	0.008	0.111	0.084	0.011	0.104	0.078	0.011	0.098	0.079	0.016	0.095	0.077
Shuttle	0.023	0.066	0.059	0.025	0.058	0.059	0.024	0.057	0.000	0.031	0.050	0.000	0.023	0.049	0.000
Sonar	0.022	0.118	0.059	0.013	0.110	0.063	0.022	0.110	0.058	0.023	0.107	0.054	0.013	0.090	0.036
Spambase	0.000	0.038	0.048	0.000	0.035	0.049	0.000	0.036	0.046	0.000	0.033	0.041	0.000	0.033	0.043

Table 3 continued

Dataset	0 %			5 %			10 %			15 %			20 %		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Splice	0.006	0.083	0.156	0.008	0.081	0.156	0.005	0.079	0.150	0.006	0.074	0.160	0.007	0.067	0.158
Thyroid	0.002	0.006	0.005	0.001	0.007	0.005	0.001	0.006	0.004	0.001	0.006	0.006	0.001	0.006	0.004
Twonorm	0.046	0.068	0.080	0.044	0.066	0.082	0.037	0.061	0.084	0.032	0.048	0.085	0.020	0.056	0.079
Vehicle	0.021	0.113	0.074	0.024	0.088	0.074	0.030	0.098	0.079	0.025	0.093	0.072	0.022	0.081	0.072
Vowel	0.002	0.061	0.049	0.002	0.060	0.049	0.001	0.050	0.044	0.001	0.054	0.043	0.002	0.047	0.042
Wdbc	0.038	0.163	0.131	0.027	0.150	0.133	0.040	0.141	0.104	0.012	0.123	0.043	0.018	0.117	0.108
Wine	0.064	0.181	0.135	0.054	0.173	0.135	0.076	0.158	0.135	0.068	0.153	0.124	0.075	0.144	0.126
Yeast	0.029	0.044	0.006	0.026	0.048	0.007	0.026	0.038	0.011	0.024	0.058	0.007	0.021	0.025	0.013
Zoo	0.044	0.147	0.102	0.037	0.144	0.102	0.032	0.143	0.125	0.036	0.138	0.124	0.026	0.150	0.146

– The algorithm with the best results for the original datasets in all SD quality measures is the NMEEFSD algorithm, which obtains values higher than 76 % in fuzzy confidence with a sensitivity close to 85 %, i.e. subgroups cover the majority of examples for the target variable with a high value of confidence. In addition, the interest of these subgroups is excellent because the values of unusualness are very high with respect to the values obtained by SDIGA and FuGePSD.
– SDIGA and NMEEFSD algorithms have the same behaviour when the noise increases in the experimental study. In this case, the values of unusualness, sensitivity and fuzzy confidence are decreased between 20 and 40 % in usualness and values close to 10 % in sensitivity and fuzzy confidence.
– The FuGePSD algorithm has a good behaviour with the noisy dataset, as can be observed in the experimental study. This algorithm keeps the same values although there is an increasing of noise with respect to the base datasets. It is even able to improve the values of subgroups obtained with noisy datasets. Moreover, FuGePSD is the SD algorithm with the best results obtained in noisy datasets, obtaining more precise and interesting subgroups.

Once the experimental study has been analysed from the point of view of the original datasets, it is necessary to present the behaviour with the application of noise filtering for these EFSs for SD. As we have mentioned in a previous section, the noise filters employed are ensemble filter (EF), cross-validated committees filter (CVCF) and iterative-partitioning filter (IPF). To summarise, average results for each quality measure and algorithm are detailed in Table 6, where the original results without filter (NF) and after preprocessing the noisy datasets with CVCF, EF and IPF are also shown. We also indicate the base case (0 %), in which no class noise is introduced into the dataset. Moreover, in this table are highlighted the best values for each algorithm and quality measure with respect to the different percentages of noise. The complete results for this experimental study can be analysed in the URL http://simidat.ujaen.es/papers/Noise-EFSs-SD.

It is actually complicated to perform a general analysis for all EFSs. In this way, the best results obtained for each algorithm and filter with respect to the different levels of noise are highlighted. With this selection, the understanding of the study is facilitated. In addition, an analysis for each algorithm is performed below:

– SDIGA is the SD algorithm with the lowest values in all the quality measures. However, this assumption is induced because it is the only algorithm which obtains subgroups for all values of the target variable. As can be

Table 4 Results for the sensitivity quality measure for the different algorithms analysed

Dataset	0%			5%			10%			15%			20%		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Automobile	0.980	0.643	0.474	0.935	0.690	0.478	0.897	0.770	0.778	0.936	0.539	0.747	0.956	0.671	0.730
Balance	0.529	0.599	0.524	0.561	0.593	0.524	0.592	0.581	0.556	0.608	0.570	0.524	0.622	0.574	0.541
Banana	0.999	1.000	0.469	0.999	1.000	0.469	0.999	1.000	0.469	0.999	1.000	0.469	0.999	1.000	0.469
Car	0.240	0.481	0.457	0.127	0.478	0.457	0.073	0.470	0.476	0.029	0.469	0.476	0.032	0.472	0.457
Cleveland	0.321	0.857	0.762	0.306	0.847	0.763	0.361	0.812	0.751	0.243	0.786	0.777	0.328	0.788	0.773
Contraceptive	0.219	0.443	0.213	0.202	0.438	0.213	0.291	0.534	0.172	0.261	0.422	0.299	0.209	0.427	0.164
Dermatology	0.827	0.994	0.950	0.791	0.981	0.947	0.776	0.948	0.967	0.816	0.898	0.949	0.778	0.894	0.944
Ecoli	0.903	0.969	0.812	0.855	0.625	0.820	0.879	0.592	0.842	0.871	0.756	0.788	0.927	0.501	0.837
Flare	0.379	0.992	0.993	0.444	0.948	0.993	0.659	0.923	0.992	0.453	0.892	0.991	0.580	0.832	0.993
German	0.901	0.770	0.816	0.943	0.749	0.816	0.900	0.759	0.818	0.865	0.740	0.827	0.846	0.716	0.795
Hayes-roth	0.710	0.501	0.480	0.645	0.516	0.480	0.740	0.483	0.260	0.763	0.498	0.489	0.725	0.453	0.391
Heart	0.921	0.852	0.725	0.841	0.848	0.727	0.904	0.806	0.732	0.886	0.803	0.744	0.920	0.802	0.744
Ionosphere	0.837	0.977	0.951	0.839	0.954	0.950	0.866	0.971	0.961	0.828	0.950	0.956	0.860	0.956	0.965
Iris	1.000	1.000	0.965	0.987	0.993	0.965	0.968	0.941	0.968	0.948	0.915	0.968	0.908	0.839	0.935
Led7digit	0.790	0.815	0.762	0.773	0.782	0.762	0.702	0.732	0.819	0.742	0.661	0.818	0.750	0.680	0.735
Magic	0.893	0.954	0.941	0.941	0.955	0.941	0.938	0.948	0.936	0.965	0.952	0.922	0.937	0.943	0.927
Monk-2	0.639	0.666	0.421	0.492	0.652	0.421	0.398	0.643	0.421	0.450	0.632	0.475	0.339	0.587	0.475
Newthyroid	0.772	1.000	0.910	0.943	0.993	0.910	0.915	0.996	0.930	0.934	1.000	0.909	0.977	0.992	0.903
Nursery	0.445	1.000	0.000	0.204	0.988	0.000	0.160	0.962	1.000	0.090	0.926	1.000	0.054	0.929	1.000
Page-blocks	0.904	0.999	0.898	0.989	0.999	0.896	0.987	0.998	0.944	0.982	0.999	0.943	0.988	0.999	0.947
Penbased	0.514	0.734	0.932	0.401	0.705	0.934	0.392	0.672	0.959	0.298	0.516	0.968	0.257	0.504	0.955
Phoneme	0.906	0.999	0.835	0.904	0.999	0.835	0.833	1.000	0.845	0.899	1.000	0.847	0.829	1.000	0.840
Pima	0.718	0.958	0.815	0.775	0.957	0.809	0.728	0.957	0.819	0.757	0.952	0.810	0.747	0.951	0.812
Ring	0.816	1.000	0.834	0.587	1.000	0.837	0.459	1.000	0.840	0.199	0.999	0.858	0.027	0.999	0.870
Segment	0.936	0.999	0.938	0.806	0.955	0.941	0.841	0.926	0.968	0.808	0.872	0.986	0.830	0.838	0.979
Shuttle	0.738	1.000	0.959	0.978	1.000	0.959	0.968	1.000	0.000	1.000	1.000	0.000	0.994	0.998	0.000
Sonar	0.988	0.953	0.720	0.994	0.944	0.729	0.993	0.955	0.729	0.993	0.929	0.699	0.996	0.957	0.705
Spambase	0.961	0.995	0.948	0.930	0.995	0.945	0.867	0.994	0.952	0.895	0.994	0.958	0.966	0.992	0.956

Table 4 continued

Dataset	0%			5%			10%			15%			20%		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Splice	0.270	0.535	0.993	0.282	0.532	0.993	0.291	0.520	0.950	0.273	0.467	0.978	0.287	0.437	0.953
Thyroid	0.997	0.961	0.888	0.996	0.955	0.895	0.928	0.955	0.947	0.959	0.922	0.925	0.945	0.768	0.942
Twonorm	0.999	0.998	0.747	0.999	0.998	0.739	0.900	0.998	0.738	0.812	0.998	0.757	0.553	0.998	0.781
Vehicle	0.489	0.972	0.759	0.516	0.771	0.759	0.640	0.901	0.762	0.517	0.946	0.730	0.488	0.765	0.816
Vowel	0.206	0.933	0.769	0.168	0.917	0.770	0.162	0.871	0.719	0.171	0.837	0.693	0.144	0.779	0.693
Wdbc	0.944	0.995	0.941	0.990	0.991	0.939	0.966	0.995	0.940	0.942	0.982	0.693	0.996	0.985	0.952
Wine	0.814	1.000	0.922	0.769	0.983	0.920	0.871	0.968	0.932	0.731	0.958	0.927	0.835	0.927	0.941
Yeast	0.994	0.897	0.819	0.981	0.802	0.819	0.983	0.636	0.978	0.975	0.960	0.550	0.998	0.477	0.553
Zoo	0.912	1.000	0.725	0.836	0.958	0.724	0.814	0.956	0.955	0.793	0.913	0.876	0.850	0.918	0.936

observed in Table 6, the algorithm has a good behaviour with noisy datasets when the ensemble filter is applied previously. In this way, the difference of values for quality measures between the original dataset (0 %) and the dataset with the most noise (20 %) when the ensemble filter is applied is null for unusualness and negative for sensitivity and fuzzy confidence, i.e. the ensemble filter is able to detect outliers and problems in subgroups extracted for minority classes which could be penalising results obtained for SDIGA in complex noisy datasets.

- The behaviour of the application of noise filtering with respect to the NMEEFSD algorithm is similar to that commented on previously for SDIGA. NMEEFSD obtains the best results when the ensemble filter is applied in a previous stage to the extraction of subgroups. In this case, the algorithm has a bad behaviour with the presence of noise in the datasets and it is necessary to apply a previous filter such as ensemble. The difference between the original dataset (0 %) and the dataset with the highest percentage of induced noise (20 %), when the ensemble filter is applied, are very close. While in fuzzy confidence there is a slight improvement, in unusualness and sensitivity it is quite the opposite.
- The FuGePSD algorithm has the best behaviour with noisy datasets, as can be observed in Table 6. In fact, this algorithm obtains better results without the application of filters than with the application of some noise filtering. FuGePSD also obtains the best results without filter for all the groups of datasets analysed in this experimental study, i.e. in 5, 10, 15 and 20 % and for all quality measures analysed. In fact, this algorithm keeps the same value in fuzzy confidence between the base datasets and the dataset with 20 % of induced noise and it increases the values of sensitivity and unusualness. In summary, this algorithm is able to improve the interest and generality of the extracted subgroups without losing precision.

It is important to remark upon the large experimental study performed in this contribution, where more than 9300 experiments for each algorithm are carried out in order to obtain these results. However, it is strictly necessary to analyse them from one statistical point of view in order to search for significant differences and confirm the previous assertions. Therefore, a statistical analysis with the Friedman test (Friedman 1937) is used. The main objective is to compare the results obtained and to be able to precisely analyse whether there are significant differences amongst the four algorithms. This test first ranks the jth of k algorithms on the ith of N datasets and then calculates the average rank according to the F-distribution (Distribution value) throughout all the datasets, and calculates the Friedman statistics. If the Friedman test rejects the null

Table 5 Results for the fuzzy confidence quality measure for the different algorithms analysed

Dataset	0%			5%			10%			15%			20%		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Automobile	0.187	0.668	0.676	0.183	0.702	0.676	0.173	0.616	0.606	0.172	0.659	0.702	0.182	0.568	0.643
Balance	0.616	0.682	0.673	0.570	0.666	0.673	0.537	0.647	0.656	0.490	0.624	0.678	0.429	0.622	0.680
Banana	0.537	0.570	0.718	0.535	0.567	0.718	0.534	0.562	0.718	0.531	0.559	0.718	0.526	0.556	0.718
Car	0.867	1.000	1.000	0.937	0.961	1.000	0.968	0.926	1.000	0.994	0.895	1.000	0.991	0.862	1.000
Cleveland	0.381	0.756	0.797	0.322	0.747	0.797	0.331	0.742	0.778	0.348	0.685	0.787	0.344	0.693	0.797
Contraceptive	0.219	0.598	0.695	0.265	0.586	0.695	0.313	0.492	0.626	0.223	0.585	0.647	0.291	0.580	0.679
Dermatology	0.208	0.778	0.811	0.186	0.769	0.798	0.203	0.782	0.783	0.190	0.712	0.803	0.182	0.711	0.801
Ecoli	0.509	0.819	0.683	0.480	0.877	0.679	0.385	0.759	0.757	0.320	0.669	0.788	0.227	0.789	0.830
Flare	0.336	1.000	1.000	0.315	0.937	1.000	0.308	0.906	1.000	0.361	0.877	1.000	0.358	0.849	1.000
German	0.535	0.767	0.762	0.542	0.770	0.761	0.559	0.729	0.762	0.542	0.742	0.782	0.544	0.728	0.759
Hayes-roth	0.691	0.881	0.449	0.727	0.848	0.449	0.624	0.904	0.220	0.631	0.838	0.576	0.549	0.821	0.397
Heart	0.628	0.757	0.731	0.599	0.725	0.730	0.585	0.742	0.714	0.549	0.732	0.728	0.606	0.713	0.745
Ionosphere	0.590	0.834	0.959	0.587	0.830	0.957	0.566	0.801	0.906	0.578	0.776	0.933	0.576	0.773	0.889
Iris	0.936	0.922	0.884	0.904	0.877	0.884	0.863	0.903	0.852	0.779	0.858	0.873	0.790	0.837	0.851
Led7digit	0.619	0.663	0.638	0.573	0.655	0.638	0.565	0.647	0.668	0.468	0.658	0.695	0.408	0.667	0.708
Magic	0.718	0.740	0.742	0.749	0.724	0.741	0.752	0.718	0.756	0.710	0.699	0.767	0.730	0.692	0.778
Monk-2	0.830	0.819	0.742	0.823	0.803	0.742	0.858	0.790	0.742	0.828	0.787	0.765	0.795	0.774	0.765
Newthyroid	0.986	0.850	0.851	0.784	0.835	0.851	0.902	0.802	0.870	0.846	0.757	0.850	0.772	0.753	0.843
Nursery	0.710	1.000	0.400	0.918	0.965	0.400	0.937	0.926	1.000	0.972	0.863	1.000	0.987	0.819	1.000
Page-blocks	0.309	0.917	0.875	0.202	0.883	0.881	0.209	0.849	0.950	0.187	0.812	0.909	0.202	0.772	0.906
Penbased	0.661	0.964	0.845	0.537	0.916	0.846	0.438	0.871	0.841	0.351	0.834	0.856	0.301	0.795	0.876
Phoneme	0.594	0.746	0.775	0.591	0.736	0.775	0.507	0.724	0.791	0.595	0.717	0.793	0.456	0.696	0.783
Pima	0.449	0.749	0.796	0.479	0.729	0.800	0.431	0.740	0.804	0.494	0.703	0.792	0.435	0.708	0.808
Ring	0.531	0.737	0.816	0.380	0.732	0.812	0.308	0.743	0.816	0.130	0.745	0.827	0.011	0.734	0.821
Segment	0.265	1.000	0.807	0.221	0.960	0.813	0.253	0.899	0.818	0.260	0.859	0.817	0.363	0.828	0.827
Shuttle	0.375	0.878	0.908	0.304	0.833	0.908	0.328	0.803	0.000	0.214	0.754	0.000	0.272	0.723	0.000
Sonar	0.543	0.750	0.728	0.525	0.736	0.755	0.539	0.742	0.759	0.547	0.741	0.759	0.529	0.725	0.646
Spambase	0.494	0.648	0.811	0.460	0.639	0.810	0.461	0.634	0.826	0.477	0.631	0.818	0.494	0.614	0.824

Table 5 continued

Dataset	0%			5%			10%			15%			20%		
	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG	NME	SDI	FuG
Splice	0.339	0.740	0.695	0.346	0.720	0.695	0.357	0.716	0.711	0.333	0.723	0.751	0.362	0.695	0.780
Thyroid	0.420	0.933	0.844	0.413	0.906	0.847	0.421	0.872	0.916	0.398	0.844	0.968	0.397	0.827	0.970
Twonorm	0.561	0.618	0.778	0.560	0.623	0.784	0.501	0.604	0.784	0.448	0.577	0.775	0.303	0.607	0.760
Vehicle	0.305	0.516	0.675	0.330	0.696	0.675	0.363	0.566	0.703	0.256	0.471	0.690	0.303	0.596	0.618
Vowel	0.045	0.712	0.623	0.025	0.712	0.627	0.015	0.617	0.711	0.019	0.650	0.691	0.014	0.578	0.675
Wdbc	0.652	0.875	0.985	0.587	0.856	0.985	0.626	0.809	0.955	0.548	0.811	0.691	0.588	0.782	0.960
Wine	0.796	0.828	0.935	0.743	0.811	0.936	0.857	0.764	0.909	0.677	0.760	0.927	0.765	0.756	0.939
Yeast	0.397	0.609	0.813	0.273	0.581	0.837	0.246	0.554	0.824	0.201	0.427	0.877	0.178	0.387	0.817
Zoo	0.262	0.837	0.843	0.217	0.842	0.805	0.219	0.790	0.907	0.287	0.765	0.800	0.214	0.789	0.851

Table 6 Details for the average results obtained for the different algorithms with and without noise filters

	0%	5%				10%				15%				20%			
	NF	NF	CVCF	EF	IPF	NF	CVCF	EF	IPF	NF	CVCF	EF	IPF	NF	CVCF	EF	IPF
SDI																	
UNUS	0.034	0.027	0.033	**0.036**	0.034	0.028	0.033	**0.035**	0.033	0.023	0.031	**0.033**	**0.033**	0.021	0.029	**0.034**	0.031
SENS	0.717	0.697	0.716	0.725	**0.730**	0.695	0.711	**0.729**	0.716	0.668	0.695	**0.730**	0.704	0.663	0.693	**0.729**	0.716
FCNF	0.497	0.471	0.504	**0.523**	0.508	0.467	0.484	**0.524**	0.498	0.439	0.477	**0.489**	0.488	0.426	0.468	**0.505**	0.471
NME																	
UNUS	0.091	0.086	0.084	**0.094**	0.090	0.084	0.089	**0.091**	**0.091**	0.079	0.085	**0.089**	0.085	0.074	0.083	**0.088**	0.085
SENS	0.851	0.825	0.819	**0.858**	0.852	0.817	0.841	**0.859**	0.854	0.803	0.831	**0.856**	0.839	0.768	0.821	**0.846**	0.832
FCNF	0.768	0.758	0.753	**0.791**	0.785	0.729	0.763	**0.787**	0.774	0.705	0.756	**0.770**	0.769	0.695	0.760	**0.785**	0.769
FuG																	
UNUS	0.075	**0.075**	0.074	0.070	0.070	**0.077**	0.072	0.070	0.067	**0.078**	0.075	0.071	0.069	**0.078**	0.075	0.070	0.068
SENS	0.733	**0.734**	0.718	0.684	0.695	**0.754**	0.707	0.686	0.681	**0.741**	0.739	0.688	0.698	**0.743**	0.740	0.687	0.705
FCNF	0.757	**0.758**	0.745	0.682	0.710	**0.748**	0.720	0.692	0.696	**0.759**	0.752	0.714	0.714	**0.757**	0.749	0.693	0.700

Table 7 Results for Friedman's statistical test showing the best (first) ranked approach. If the null hypothesis N_0 is rejected with $\alpha = 0.1$, the filter is highlighted in bold. When the null hypothesis is rejected with the same α, the best option is underlined

	Unusualness				Sensitivity				Fuzzy confidence			
	5 %	10 %	15 %	20 %	5 %	10 %	15 %	20 %	5 %	10 %	15 %	20 %
SDI	**EF**	**EF**	**EF**	**EF**	IPF	IPF	EF	IPF	**EF**	**EF**	**EF**	**EF**
NME	**EF**	**EF**	**EF**	**IPF**	**EF**	**EF**	**EF**	**EF**	**EF**	**EF**	**EF**	**EF**
FuG	EF	NF	NF	NF	NF	**NF**	**NF**	**NF**	NF	NF	**NF**	**NF**

hypothesis, this indicates that there are significant differences.

The results of this statistical analysis can be observed in Table 7 where for each algorithm and quality measure for SD the option with the best ranking in the different levels of noise is presented. In addition, the best option is highlighted in bold in the case of significance differences considering the Friedman test rejects at level of significance $\alpha = 0.10$ and underlined if the null hypothesis is rejected with the same confidence level. A post hoc analysis of the results from Friedman's statistical test showed that no significative differences among the filters were found.

The majority of assumptions performed in the previous analysis are confirmed by the statistical tests. For the NMEEFSD algorithm, the best results in noisy datasets are obtained with significance differences through the previous application of the EF. However, for the SDIGA algorithm the behaviour is more complicated because for unusualness and fuzzy confidence the previous assumptions are confirmed but in sensitivity the best filter is IPF, although without significant differences. In this way, for the SDIGA algorithm it would be advisable to apply EF in noisy environments. Finally, we must point out that the results obtained in the statistical tests for the FuGePSD confirm the excellent behaviour of this algorithm without the necessity of a previous noise filtering.

6 Effects of noise on subgroup discovery models interpretability

In the previous section, we have analysed the effects on the performance of SD methods of the presence of noise with respect to three quality measures. However, this section shows the effect of noise in SD based on the analysis of the interpretability focused on the number of rules and variables. In fact, in the literature in rule learning, and specifically in SD, the number of rules and the average number of labels in the rules are usually considered as a descriptive measure of interpretability (Carmona et al. 2014).

Figure 1 depicts the average number of rules for SDIGA, NMEEFSD and FuGePSD. From this figures, we can point out the following, emphasising that the lower the number of extracted rules is, the more interpretable the model should be:

- As we can observe, the use of noise filters helps SDIGA to maintain the number of rules generated as the noise level increases, while not using a filter almost doubles the number of rules obtained. Thus SDIGA is affected by class noise, generating a fragmented description of the subgroups if not correctly treated.
- NMEEEFSD generates a model comprised of the highest number of rules among all the three SD algorithms considered. However, when noise increases and it is not treated with filtering, the number of rules decrease. On the other hand, by using EF the number of extracted rules is maintained as the highest among all the filters. In this sense, NMEEFSD obtains accurate descriptions but the presence of noise forces NMEEFSD to drop rules according to its fitness function. EF is the most appropriate way to avoid such a problem.
- In the case of FuGePSD, it behaves similarly to SDIGA as the number of rules increases with the noise introduced. However, although the CVC filter drives FuGePSD to extract a similar number of rules to NF, the performance with CVC when compared to NF is significatively lower. Thus, FuGePSD does not benefit from filtering as it is forced to fragment the description it creates by increasing the number of rules.

Since the number of rules are not able to provide a whole picture of the interpretability of a model, we now proceed to analyse the number of labels involved in the rules generated by the three SD techniques considered. In Fig. 2, the average number of labels for SDIGA, NMEEFSD and FuGePSD is shown. Taking into account that the lower the number of labels extracted is, the more legible the model is expected to be, and from these figures we can point out the following:

- In the case of SDIGA, the average number of labels obtained decreases as the noise increases for IPF and EF, while NF and CVC drive SDIGA to use a similar amount of labels in every noise level. It is interesting to remark that IPF and EF are the filters that help SDIGA to behave better in the presence of noise. A possible explanation is that IPF and EF clean the class borders better for SDIGA, helping it to learn better subgroup descriptions, while NF and CVC may cause SDIGA to overfit as the noise increases.

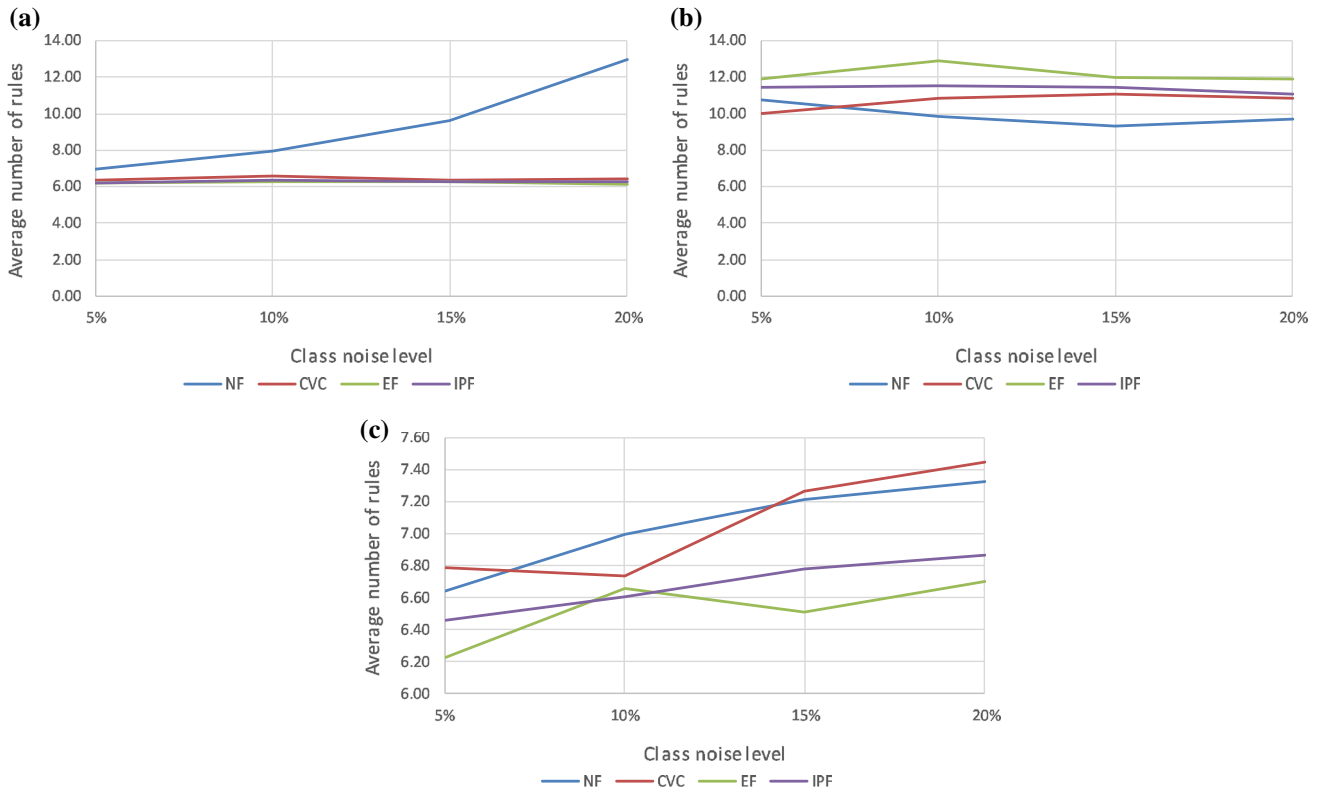


Fig. 1 Average number of rules for each SD algorithm per class noise level. **a** SDIGA, **b** NMEEFSD, **c** FuGePSD

– If we consider NMEEFSD, we can see that NF creates a larger amount of rules, while EF (the best filter for NMEEFSD) helps to keep the number of labels involved the lowest. On the other hand, not filtering (NF) makes NMEEFSD use a larger number of labels in the rules and thus overfit the subgroup descriptions. The other two filters, CVC and IPF, help NMEEFSD to also obtain a lower number of rules than NF, but not as lower as EF.
– FuGePSD shows a completely different behaviour to SDIGA and NMEEFSD. While at low noise levels the number of labels is high, as the noise introduced increases, FuGePSD reduces the number of labels used in the rules in order to avoid overfitting the description (as SDIGA does). The use of filters hinders this behaviour, as FuGePSD maintains a higher number of labels. We can assume that the design of FuGePSD makes it more robust than SDIGA.

As a summary of the depicted results for the average number of rules and labels, we may emphasize on how different the three SD algorithms analysed in this study are. SDIGA is prone to be affected by noise to a greater degree than NMEEFSD and FuGePSD. By using filters, the description that it obtains is somehow improved, but it is still too simple when compared to NMEEFSD's or FuGePSD's. NMEEFSD is an accurate technique at the cost of the largest number of rules but the lowest number of variables. This kind of configuration seems to benefit more from filtering than FuGePSD's. The model generated by FuGePSD contains a lower number of rules than NMEEFSD's but with more labels. In summary, FuGePSD and SDIGA are more affected by class noise than NMEEFSD, mainly because NMEEFSD is able to benefit more from the application of noise filtering: it is the approach that is more stable both in terms of number of rules and labels extracted if noise filters are used.

While we may not decide which model is more interpretable, a model with the lowest rules or a model with the lowest number of labels, it is interesting to stress that noise filtering seems to help more models with many and simple rules than models with few but complex rules. Since filtering usually cleans the class borders, a larger number of rules will probably capture this simplified class description better.

7 Concluding remarks

In this paper, we have analysed the influence of the class noise on the SD problem, that is a descriptive data mining technique using supervised learning. Specifically, the analysis has been performed with the most representative EFSs for SD: SDIGA, NMEEFSD and FuGePSD. To do so, different class noise amounts (5, 10, 15 and 20 %) were introduced

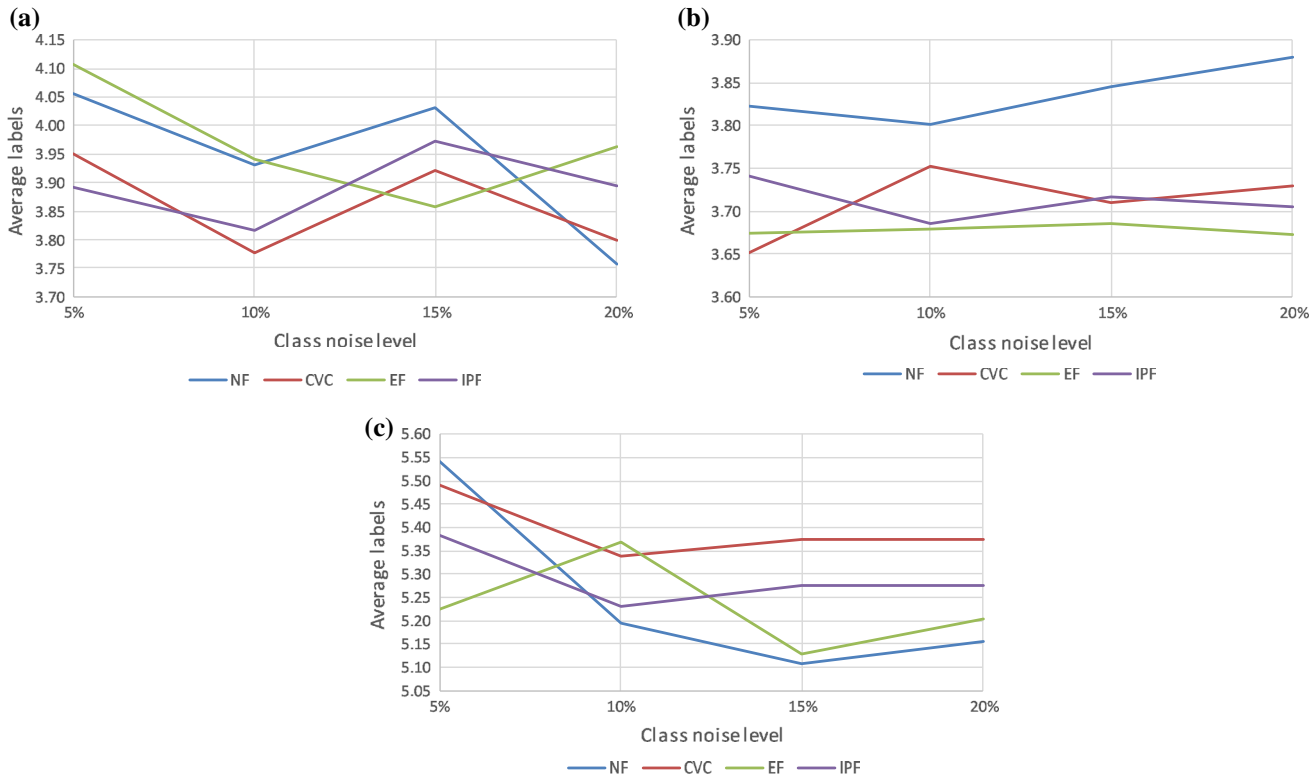


Fig. 2 Average number of labels for each SD algorithm per class noise level. **a** SDIGA, **b** NMEEFSD, **c** FuGePSD

into the original datasets. Since no approaches on how to deal with noise in SD can be found in the literature, we have considered the usage of noise filtering preprocessing techniques commonly used in classification as a possible way to deal with class noise. Since noise filters were designed to be used in classification, one of the goals of this paper is to analyse the suitability of such approaches in a descriptive task as SD.

The study carried out in this manuscript shows different situations with respect to the increment of noise depending on the algorithm analysed. Although all the algorithms employ the fuzzy SD approach that confers robustness against uncertainty and noise to a certain extent, experts can observe the advantages provided for noise filtering in SD. However, it is important to remark on the analysis of the FuGePSD algorithm which is the best EFS for SD in noisy environments. This algorithm is able to detect the noise in the problem and isolate it, and in this way the algorithm does not lose quality in the extracted subgroups with this type of problems.

On the other hand, when the amount of noise increases its treatment cannot be ignored for the SDIGA and NMEEFSD algorithms and the application of specialized techniques as noise filters is beneficial. Specifically, we remark that EF becomes the better option when noise cannot be neglected.

We have also addressed how noise affects the complexity and interpretability of the extracted models. While simple and with lower performance models like those generated by SDIGA marginally benefit from filtering, we have shown that models with few but complex rules will probably benefit much less than models with more but simpler rules. Depending on the type of SD algorithm the practitioner plans to use, experts may decide whether to expend time filtering the dataset by following this assumption.

In summary, this contribution focuses on the usage of noise filters as a viable approach to deal with noisy scenarios with EFSs SD. The noise treatment for SD needs more studies and attention in order to deep its analysis, as this paper covers the first step using well-known preprocessing techniques for a related supervised field. We believe that this paper can serve as a motivation to further analyse other approaches for EFSs SD, develop novel SD algorithms robust to noise and also expand research of class noise in SD. It can also motivate the development of specific preprocessing techniques for SD, as the performance metrics for SD differ from those usually considered in classification (where noise filters are usually born). As a final remark, this study provides the basis for the future use of EFSs of SD in real-world problems when experts address problems with noisy environments, and in this way they could obtain more comprehensible knowledge in this data mining field.

Acknowledgments This work was supported by the Spanish Ministry of Economy and Competitiveness under Project TIN2015-68454-R (FEDER Founds), by the Spanish Ministry of Science and Technol-

ogy under Project TIN2014-57251-P (National Projects) and by the Regional Excellence Projects P11-TIC-7765 and P12-TIC-2958.

Compliance with ethical standards

Conflict of interest The author declares that there is no conflict of interests regarding the publication of this paper.

Ethical approval This article does not contain any studies with human participants and animals performed by any of the authors.

References

Abellán J, Masegosa A (2012) Bagging schemes on the presence of class noise in classification. Expert Syst Appl 39(8):6827–6837

Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult Valued Log Soft Comput 17(2–3):255–287

Atzmueller M, Puppe F (2006) SD-Map—a fast algorithm for exhaustive subgroup discovery. In: Proceedings of the 17th European conference on machine learning and 10th European conference on principles and practice of knowledge discovery in databases. Springer, LNCS, vol 4213, pp 6–17

Bonissone P, Cadenas JM, Carmen Garrido M, Andrés Díaz-Valladares R (2010) A fuzzy random forest. Int J Approx Reason 51(7):729–747

Brodley CE, Friedl MA (1999) Identifying mislabeled training data. J Artif Intell Res 11:131–167

Cao J, Kwong S, Wang R (2012) A noise-detection based AdaBoost algorithm for mislabeled data. Pattern Recognit 45(12):4451–4465

Carmona CJ, González P, del Jesus MJ, Herrera F (2010) NMEEF-SD: non-dominated multi-objective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. IEEE Trans Fuzzy Syst 18(5):958–970

Carmona CJ, González P, del Jesus MJ, Navío M, Jiménez L (2011) Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department. Soft Comput 15(12):2435–2448

Carmona CJ, Ramírez-Gallego S, Torres F, Bernal E, del Jesus MJ, García S (2012) Web usage mining to improve the design of an e-commerce website: OrOliveSur.com. Expert Syst Appl 39:11243–11249

Carmona CJ, Chrysostomou C, Seker H, del Jesus MJ (2013a) Fuzzy rules for describing subgroups from Influenza A virus using a multi-objective evolutionary algorithm. Appl Soft Comput 13(8):3439–3448

Carmona CJ, González P, García-Domingo B, del Jesus MJ, Aguilera J (2013b) MEFES: an evolutionary proposal for the detection of exceptions in subgroup discovery. An application to Concentrating Photovoltaic Technology. Knowl Based Syst 54:73–85

Carmona CJ, González P, del Jesus M, Herrera F (2014) Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms. WIREs Data Min Knowl Discov 4(2):87–103. doi:10.1002/widm.1118

Carmona CJ, Ruiz-Rodado V, del Jesus MJ, Weber A, Grootveld M, González P, Elizondo D (2015) A fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans. Inf Sci 298:180–197

Cherkassky V, Mulier FM (2007) Learning from data: concepts, theory, and methods. Wiley-IEEE Press, New York

Deb K, Pratap A, Agrawal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197

del Jesus MJ, González P, Herrera F, Mesonero M (2007) Evolutionary fuzzy rule induction process for subgroup discovery: a case study in marketing. IEEE Trans Fuzzy Syst 15(4):578–592

Eiben AE, Smith JE (2003) Introduction to evolutionary computation. Springer, Berlin

Fogel DB (1995) Evolutionary computation—toward a new philosophy of machine intelligence. IEEE Press, Piscataway

Frénay B, Verleysen M (2014) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32:675–701

Gamberger D, Lavrac N (2002) Expert-guided subgroup discovery: methodology and application. J Artif Intell Res 17:501–527

García S, Luengo J, Herrera F (2015) Data preprocessing in data mining. Springer, Berlin

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc, Boston

Grosskreutz H, Rueping S (2009) On subgroup discovery in numerical domains. Data Min Knowl Discov 19(2):210–216

Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolut Intell 1:27–46

Herrera F, Carmona CJ, González P, del Jesus MJ (2011) An overview on subgroup discovery: foundations and applications. Knowl Inf Syst 29(3):495–525

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor

Kavsek B, Lavrac N (2006) APRIORI-SD: adapting association rule learning to subgroup discovery. Appl Artif Intell 20:543–583

Khoshgoftaar TM, Rebours P (2007) Improving software quality prediction by noise filtering techniques. J Comput Sci Technol 22:387–396

Kloesgen W (1996) Explora: a multipattern and multistrategy discovery assistant. In: Advances in knowledge discovery and data mining, american association for artificial intelligence, pp 249–271

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge

Lavrac N, Flach PA, Zupan B (1999) Rule evaluation measures: a unifying view. In: Proceedings of the 9th international workshop on inductive logic programming. Springer, LNCS, vol 1634, pp 174–185

Lavrac N, Cestnik B, Gamberger D, Flach PA (2004a) Decision support through subgroup discovery: three case studies and the lessons learned. Mach Learn 57(1–2):115–143

Lavrac N, Kavsek B, Flach PA, Todorovski L (2004b) Subgroup discovery with CN2-SD. J Mach Learn Res 5:153–188

Mclachlan GJ (2004) Discriminant analysis and statistical pattern recognition (Wiley series in probability and statistics). Wiley-Interscience, Hoboken

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Francisco

Sáez JA, Galar M, Luengo J, Herrera F (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. Knowl Inf Syst 38(1):179–206

Schwefel HP (1995) Evolution and optimum seeking. Sixth-generation computer technology series. Wiley, New York

Sluban B, Gamberger D, Lavra N (2010) Performance analysis of class noise detection algorithms. Front Artif Intell Appl 222:303–314

Sun B, Chen S, Wang J, Chen H (2016) A robust multi-class AdaBoost algorithm for mislabeled noisy data. Knowl Based Syst 102:87–102

Sáez JA, Galar M, Luengo J, Herrera F (2016) INFFC: an iterative class noise filter based on the fusion of classifiers with noise sensitivity control. Inf Fusion 27:19–32

Teng C (2004) Polishing blemishes: issues in data correction. IEEE Intell Syst 19(2):34–39

Teng CM (1999) Correcting noisy data. In: Proceedings of the sixteenth international conference on machine learning. Morgan Kaufmann Publishers, San Francisco, pp 239–248

Venturini G (1993) SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In: Proceedings European conference on machine learning. Springer, LNAI vol 667, pp 280–296

Verbaeten S, Assche AV (2003) Ensemble methods for noise elimination in classification problems. In: Fourth international workshop on multiple classifier systems. Springer, pp 317–325

Wrobel S (1997) An algorithm for multi-relational discovery of subgroups. In: Proceedings of the 1st European symposium on principles of data mining and knowledge discovery. Springer, LNAI, vol 1263, pp 78–87

Wrobel S (2001) Inductive logic programming for knowledge discovery in databases. Springer, chap Relational Data Mining, pp 74–101

Wu X, Zhu X (2008) Mining with noise knowledge: error-aware data mining. IEEE Tran Systems Man Cybern Part A Syst Hum 38(4):917–932

Zadeh LA (1975) The concept of a linguistic variable and its applications to approximate reasoning. Parts I, II, III. Inf Sci 8-9:199–249, 301–357, 43–80

Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study. Artif Intell Rev 22:177–210

Zhu X, Wu X, Chen Q (2003) Eliminating class noise in large datasets. In: Proceeding of the twentieth international conference on machine learning, pp 920–927