The Knowledge Engineering Review, Vol. 34, e1, 1–33.   © Cambridge University Press, 2019
doi:10.1017/S026988891800036X

Data preprocessing in predictive data mining

STAMATIOS-AGGELOS N. ALEXANDROPOULOS, SOTIRIS B. KOTSIANTIS and MICHAEL N. VRAHATIS

Computational Intelligence Laboratory (CILab), Department of Mathematics, University of Patras, GR-26110 Patras, Greece; e-mail: alekst@math.upatras.gr, sotos@math.upatras.gr, vrahatis@math.upatras.gr

Abstract

A large variety of issues influence the success of data mining on a given problem. Two primary and important issues are the representation and the quality of the dataset. Specifically, if much redundant and unrelated or noisy and unreliable information is presented, then knowledge discovery becomes a very difficult problem. It is well-known that data preparation steps require significant processing time in machine learning tasks. It would be very helpful and quite useful if there were various preprocessing algorithms with the same reliable and effective performance across all datasets, but this is impossible. To this end, we present the most well-known and widely used up-to-date algorithms for each step of data preprocessing in the framework of predictive data mining.

1 Introduction

The data preprocessing always has an important effect on the generalization performance of a supervised machine learning (ML) algorithm. By taking into consideration that well-known and widely used methods of ML often involved in data mining (DM), the importance of the data preprocessing in DM can be easily recognized. In general, ML is concerned with predicting an outcome given some data (Witten *et al*., 2016). Many ML methods can be formulated as formal probabilistic models. Thus, in this sense, ML is almost the same as statistics, but it differs in that it generally does not take care of parameter estimates (just for predictions) and it is focused on computational efficiency (Witten *et al*., 2016). DM is a field that is based on various approaches of ML and statistics and it is accomplished by an expert on a specific dataset with a desired result in mind (Witten *et al*., 2016). In many cases, the dataset is massive, complicated or it is possible to have particular problems, for example, when there are more features than instances. In general, typically, the aim is either to generate some preliminary insights related to an application field where there is little a priori knowledge, or to be able to predict accurately future observations.

In many problems, the dataset that it has to be managed contains noisy data which makes the elimination of noisy instances necessary and one of the hardest processes in ML (Zhu & Wu, 2004; van Hulse & Khoshgoftaar, 2006; van Hulse *et al*., 2007). Another difficult issue is the distinguish among inliers and true data values. These values (inliers) are error data values that can be found in the interior of a statistical distribution and their localization and correction constitutes a very difficult task. Thus, although the multivariate data cleaning constitutes a complicate procedure, it is necessary, efficient and effective process (Escalante, 2005).

A common problem that has to be tackled by utilizing data preparation is the handling of the missing data (Pearson, 2005; Farhangfar *et al*., 2007). In many cases, various datasets that contain both numerical and categorical data have to be handled. On the other hand, it is well-known that many algorithms, such as the logic learning algorithms, can handle better or exhibit a better performance only with categorical instances. In the case that it occurs, the discretization of numerical data constitutes a very important issue

(Elomaa & Rousu, 2004; Flores *et al.*, 2011). A very useful approach for handling the above problem is the grouping of the categorical data. As a consequence, the original dataset is transformed from categorical to numerical. From another point of view, a dataset with many different types of data or many values is difficult to be handled. This raises questions such as, which set of data gives the most useful information and therefore which amount should be chosen for inducing decision trees or rule learners? As a natural consequence, there may be a serious risk of overestimating much of the data, which makes the selection process of the most informative features a very difficult task. A well-known and widely used technique through which this normal learning difficulty is covered from very large datasets is the selection of a sample data from the initially large dataset (Wilson & Martinez, 2000). On the other hand, the question remains and is pertinent, which sample is appropriate to choose or how large does this subset of data need to be? Various techniques have been proposed to solve the issue of imbalanced data-classes (Batista *et al.*, 2004; Estabrooks *et al.*, 2004), but it still remains a problem that requires further study.

The data preprocessing steps with which we will deal in the paper at hand is exhibited in Figure 1. In particular, the flow that is followed before applying the learning algorithms to a particular DM task is shown. Initially, we collect the information or we receive datasets from a source. Next, the preprocessing steps are properly applied to clean the sample to make it useful. Then, the data are given as an input to the learning algorithms, and finally, they are applied to solve a specific problem (e.g., classification, regression, pattern recognition, clustering etc.). All or some of the data preprocessing steps may be useful in all of the algorithms shown in Figure 1. It is worth to emphasize a detail that is presented and concerns the step of discretization or normalization of the data. The only, perhaps, diversifying the reader may encounter has to do with algorithms that manage only datasets consisting of numbers, such as support vector machines (SVM) or neural networks. In this case, we have to proceed with normalization, rather than discretization of data. In any other case, any preprocessing step for any algorithm can be used. Regarding the field of applications, namely the problems that can be applied to these steps include, but not limited to, the classification and regression problem. However, since we are dealing with predictive DM tasks, this review focuses on methods that are applied for dealing with classification and regression problems.

One of the key challenges in the DM approach that is related to the performance of the learning algorithms, is the result to be dense and easily clarified. The representation of the dataset plays an important role in accomplishing this task. A dataset with too many features or features with correlations should not be included in the learning process, since these kinds of data does not offer useful information (Guyon & Elisseeff, 2003). Thus, to use only the most informative data it is necessary to select features

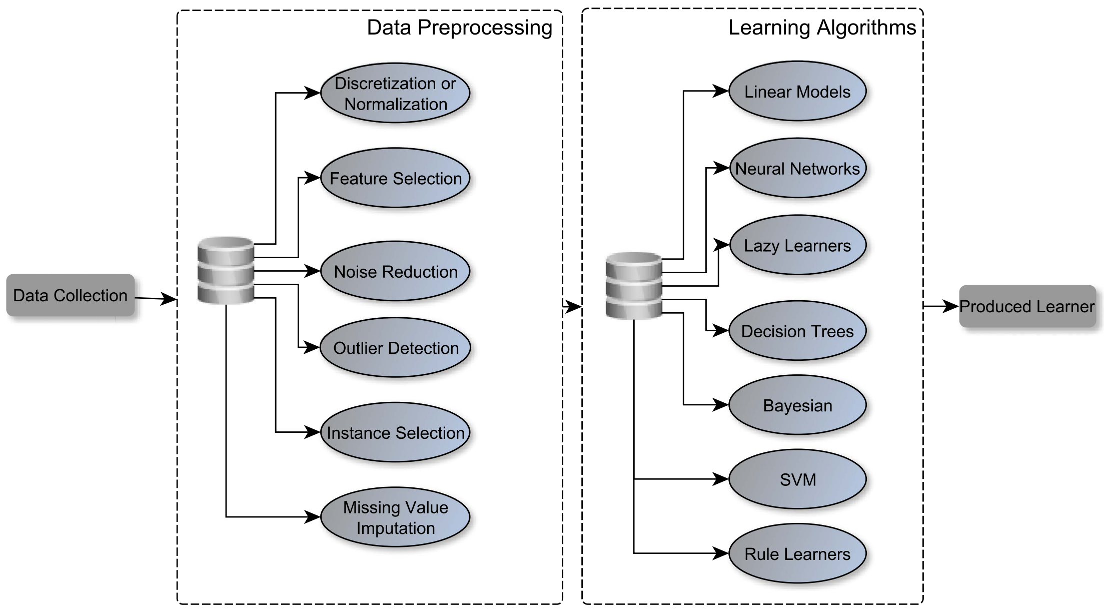


Figure 1 Predictive modeling process

that will reduce information which is unnecessary and unrelated to the objective of our study. Furthermore, instance and feature selection (FS) is also useful in distributed DM and knowledge discovery from distributed databases (Skillicorn & McConnell, 2008; Czarnowski, 2010).

An alternative approach towards the usage of informative data is the feature weighting approach based on some conditions (e.g., distance metrics) (Panday *et al.*, 2018; Zhou *et al.*, 2018). To this end, the original dataset can be separated into weighted features and the features with the highest weight are involved in the learning process. In addition, in many cases it is more useful to construct features by transforming the initial dataset instead of finding a good subset of features (Mahanipour *et al.*, 2018; Virgolin *et al.*, 2018).

As we have already mentioned, the paper at hand addresses issues of data preprocessing. For more details about this subject, the reader is referred to Pyle (1999) and García *et al.* (2015). In this article, we focus on the most well-known and widely used up-to-date algorithms for each step of data preprocessing in predictive DM.

The next section covers noise and outlier detection. The topic of processing unknown feature values is described in Section 3. Next, in Section 4 the normalization and discretization processes are presented. Sections 5 and 6 cover, respectively, instance and FS. The paper ends in Section 7 with a discussion and some concluding remarks. In addition, at the end of each section, we provide a table of citations[1] that the most known and widely used methods of each data preprocessing step have received. In our point of view, this reveals the importance and the contribution of each method as well as its impact on the scientific community. Moreover, the reader can study a brief description of each method which is included in every table. Finally, the reader can reach in the Appendix a simple code of basic preprocessing steps, that gives simple tutorial examples in R programming language related to the usage of some known preprocessing algorithms.

2 Noise and outlier detection

One of the main problems to be tackled is that a variety of algorithms, such as instance-based learners, are sensitive to the presence of noise. This has the effect of misclassification which is based, for example, on the wrong nearest neighbor. This is so, because similarity measures can easily be falsified by data with noise in their values. A well-known preprocessing technique for finding and removing the noisy instances in the dataset is the noise filters. As a result, a new improved set of data is generated without noise and the resulting dataset can be used as input to a DM algorithm.

A simple filter approach is the variable-by-variable data cleaning. In this approach, the values that are considered as 'suspicious' values according to specific criteria are discarded or corrected. Thus, they can be included in the dataset that will be given as input to a DM algorithm. In this approach, the criteria include, among others, the following: an expert evaluates the suspicious data as errors or as false labeled or in other cases a classifier predicts those values as 'unclear' data.

As we have already mentioned in the introduction, besides the effect of noisy values, it is possible the appearance of dataset values called outliers. In Aggarwal (2013) the author has described a method for creating a sorted list of values, with those values that are higher to be identified as outliers. Below in this section we describe methods used to identify outliers, as they are categorized in the following way (Chen *et al.*, 2010a):

(a) statistics-based methods,
(b) distance-based methods,
(c) density-based methods.

A basic statistics-based method has been presented in Huang *et al.* (2006). It assumes a statistical model or a distribution (e.g., Gaussian or normal) for the original dataset and the outliers can be detected by using some statistical tests. In addition, many methods, like the technique that has been proposed in

[1] Source: Google Scholar.

Buzzi-Ferraris and Manenti (2011) identify the outliers and at the same time they evaluate the mean, the variance and those values that are outliers.

In the case of large datasets, in Angiulli and Pizzuti (2005) the authors have proposed a distance-based outlier detection algorithm which is able to identify the top *m* outliers of the dataset, where *m* is a given integer. Their algorithm, called HilOut, computes the weight of a point as the sum of the distance separating it from its *k*-nearest neighbors (*k*-NN). Hence, the points with the highest amount of weight are identified as the outliers. A similar approach for the detection of mining distance-based outliers is related to the fast algorithm, called RBRP, that has been presented in Ghoting *et al.* (2008). Specifically, the authors faced the problem of outlier detection in high-dimensional datasets. The experiments that they conducted shown that their method is more efficient in terms of computational time, compared to other approaches. An outlier mining algorithm, named OMABD, which is based on dissimilarity has been proposed in Zhou and Chen (2012). According to this algorithm, the detection of outliers is based on a comparison of two values: (i) the first value determines the dissimilarity degree that is constructed by the dissimilarity of each object of the dataset and (ii) the second one determines a dissimilarity threshold. In Filzmoser *et al.* (2008) the authors have proposed a more computationally efficient algorithm for detecting the outliers of a dataset. This algorithm is able to handle efficiently high-dimensional datasets using principal component analysis (PCA). Last but not least we refer to the method that has been proposed in Chen *et al.* (2010b) which implements the neighborhood outlier identification. So, inspired from the well-known and widely used *k*-NN algorithm, the authors created a new algorithm called neighborhood based outlier detection algorithm. This algorithm compared with classic distance metric methods, such as *k*-NN, distance-based and replicator neural network based outlier detection method, outperformed the above methods in the case of the mixed datasets.

The density-based outlier detection identifies an outlying instance regarding the density of the surrounding space. Despite the fact that density-based outlier detection methods have an amount of advantages, the computational complexity issue remains a strong and of great concern problem for its application. An approach for the reduction of the computational complexity of density-based outlier detection has been proposed in Kim *et al.* (2011). Particularly, the authors combine two known techniques: the KD-tree structure and the *k*-NN algorithm, to improve the performance of the existing local outlier factor (LOF) algorithm. The experimental results that they have noted over three known datasets of UCI (University of California at Irvine Machine Learning Repository), shown that their approximation is better in terms of computational time than the original LOF algorithm.

On the other hand, in Liu *et al.* (2012) the authors proposed an approach without using a density parameter. Their method, called iForest, was compared with several known methods, such as random forest, LOF, one class of SVM and ORCA. The experimental results showed that iForest overcomes all the above methods in terms of computational time and it needs more less memory-requirement than the other methods. Specifically, this method focuses randomly on a specific feature. It selects randomly a value between the maximum and minimum values of the specific feature to 'isolate' instances. This process can be iteratively repeated with the usage of a tree structure. The required number of partitions to apply the above isolation procedure to a dataset is the same as the length of the path from the top of the tree to the terminating node on its leaves. As they noticed, in such an itree, if a path length for specific instances is shorter than the expected, then they are highly likely to be anomalies. The editing techniques focus specifically on the removal of noise or anomalous instances in the training set (Segata *et al.*, 2010). Therefore, the assessment of each individual instance of the dataset whether or not to be removed is necessary and very important. Approaches for the outlier detection and noise filtering can be categorized as (i) supervised, (ii) semi-supervised and (iii) unsupervised (Angiulli & Fassetti, 2014). The supervised and semi-supervised methods, label data to create a model that unravels outliers from inliers. On the other hand, unsupervised approaches do not require any label object and detect outliers as points that are quite dissimilar from the remaining ones.

Another important issue which is related to noise is the case where the noise affects the output attribute. It is widely known that noise affects both input and output attributes. In Brodley & Friedl (1999) an ensemble filter (EF) which improves the filtering by the usage of a set of learners has been introduced. So, the authors conducted experiments included single classifiers and a set of classifiers, testing the EF over several datasets. The experimental results showed that the classification accuracy was better when the filtering approach of the multiple learners was used. In Delany *et al.* (2012) the authors have presented a very useful study on the evaluation of different noise reduction techniques regarding the form of the examples on which the techniques are focused. The results that have been obtained in Sáez *et al.* (2013) have shown that there is a notable relation between the complexity metrics of a dataset and the efficiency and effectiveness of several noise filters which are related to the nearest-neighbor classifiers.

In Sáez *et al.* (2016) the authors have proposed a method that combines the decision of many classifiers to detect noise by unifying many different learners through an iterative scheme. Specifically, the filtering scheme that they have introduced identifies the noise by avoiding the detection of noisy instances at every new iteration of the process. Also, in Garcia *et al.* (2016a) the authors have examined how label noise detection can be improved by using an ensemble of noise filtering schemes. In this approach, the authors tried to limit the noisy data to the final dataset, as well as to remove the useless data. So, by creating meta-features from corrupted datasets, they have provided a meta-learning model that predicts the noisy data over a new dataset.

Furthermore, in Ekambaram *et al.* (2016) it has been shown that some basic principles used in SVMs have the particularity of capturing the mislabelled examples as support vectors. Due to the fact that there cannot be a single method that overcomes all the others in a specific noise identification task, the research that has been presented in Garcia *et al.* (2016b) is of great importance, since it presents a recommendation for the expected performance of specific filters that are used for particular tasks.

Table 1 exhibits a brief description of the most known and widely used methods, related to the noise and outlier detection issues and presents their applicability. Furthermore, it presents information about the citations that they have received.

3 Missing feature values

The problem of missing feature values related to (not necessarily) large sets of data is one of the main problems that has to be tackled during data preprocessing. Most of the datasets in real-world applications are datasets that contain incomplete information, that is, missing a small or a large part of the dataset. There are classifiers that exhibit robust behavior in datasets with missing values, such as naive Bayes, and therefore the final result of the decision is not affected by the missing values. On the other hand, there are classifiers, such as neural networks and *k*-NN, that require a careful handling of the incomplete information.

The above-described problem has been a major concern for the scientific community and significant progress has been made by many researchers in this field over the last years (Zhang *et al.*, 2008). However, there are important issues that require attention and raise questions that the experts are called upon to answer. The most important question is derived from the source of this 'incompleteness'. More specifically, is the dataset incomplete because someone forgot these values or for some other reason they were lost? as well as, does a particular feature considered for some reason as unnecessary or unenforceable? whereas in fact these data are useful and necessary for a better prediction by the learner. In general, missing at random means that the tendency for a data point to be missing is not related to the missing data, but it is related to some of the observed data. In this case, it is safe to remove the data with missing values depending upon their occurrences. On the other hand, in the other case (missing not at random) by removing instances with missing values could produce a bias in the learning model. Thus, we could use imputation in this case, but this does not necessarily give better results (Zhang *et al.*, 2008).

There are several methods and techniques for handling missing data which can be chosen as follows (Jerez *et al.*, 2010; Cismondi *et al.*, 2013):

(a) Most common feature value of a categorical attribute: Fill in missing values with the value that appears most often in the given dataset.
(b) Concept most common feature value of a categorical attribute: Similarly to the above case, except that the missing values are filled by the values of the same class.

Table 1 Brief description, applicability in noise and outlier detection and number of citations of the references related to the noise and outlier detection issues presented in Section 2

Title	Noise detection	Outlier detection	TC	CpY	Brief description
'Identifying mislabeled training data'	Yes	No	578	30.42	The goal of this paper is the identification and the elimination of mislabeled training examples. To achieve this, the authors proposed an ensemble of classifiers to filter the noisy instance of the training data and to improve the classification process (Brodley & Friedl, 1999)
'Outlier mining in large high-dimensional data sets'	No	Yes	368	28.31	The proposed method is based on distances to compute and separate the outlier data from the rest of the dataset. The authors provided an algorithm that computes the highest *n* outliers, where *n* is a given integer number (Angiulli & Pizzuti, 2005)
'Review of outlier detection'	No	Yes	31	2.58	Assuming that the dataset obeys a particular distribution, the authors compute the outlier data according to the corresponding statistical criteria (Huang *et al*., 2006)
'Fast mining of distance-based outliers in high-dimensional datasets'	No	Yes	215	21.50	This approach is similar to Angiulli and Pizzuti (2005) method for the detection of the outliers based on distances. The proposed algorithm targets at high-dimensional datasets and provides a fast determination of the nearest neighbors (Ghoting *et al*., 2008)
'Outlier identification in high dimensions'	No	Yes	315	31.50	The proposed method uses basic aspects of PCA for the identification of outliers. The proposed scheme is appropriate for the use of high-dimensional datasets and requires less computational time regarding other similar approaches (Filzmoser *et al*., 2008)
'Neighborhood outlier detection'	No	Yes	60	7.50	An algorithm is proposed for outlier detection based on neighborhood. In particular, the authors proposed a new metric that uses the neighborhood of the instance to identify if the instance is outlier (Chen *et al*., 2010b)
'A comparison of outlier detection algorithms for ITS data'	No	Yes	65	8.125	The authors provided three main types of outlier detection algorithms: the statistics-based, the distance-based and the density-based one. Also, they presented a comparison of these methods on intelligent transportation systems (Chen *et al*., 2010a)
'Noise reduction for instance-based learning with a local maximal margin approach'	Yes	No	48	6.00	An approach for noise reduction which is competitive regarding the well-known *k*-NN classifier is proposed. This new classifier, called local support vector machine, is a modification of the widely used SVM classifier and performs really-well in large and noisy datasets (Segata *et al*., 2010)
'Outlier detection in large data sets'	No	Yes	42	6.00	In this approach, the authors provided a model that except of the outlier detection, computes the mean, the variance and the outliers of the datasets (Buzzi-Ferraris & Manenti, 2011)
'Fast outlier detection for very large log data'	No	Yes	37	5.29	The KD-tree structure and the well-known *k*-NN classifier are combined, to reduce the computational cost of density-based outlier detection (Kim *et al*., 2011)

'Profiling instances in noise reduction'	Yes	No	14	2.33	A comparison between noise reduction methods, regarding instance-based learners, is presented. In addition, the authors study the removal instances, and in particular, the type of these instances (Delany *et al.*, 2012)
'An introduction to outlier analysis'	No	Yes	42	8.40	The basic outlier models, as well as widely used meta-learning models for outlier analysis are presented (Aggarwal, 2013)
'Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification'	Yes	No	59	11.80	A method that selects the appropriate noise filter for a given noisy dataset is presented. The method is based on data complexity measures and improves the results of the well-known *k*-NN classifier (Sáez *et al.*, 2013)
'Exploiting domain knowledge to detect outliers'	No	Yes	9	2.25	The main subject is the comparison of the concept-based outlier algorithm with other similar modifications. The main difference is the required executing time (Angiulli & Fassetti, 2014)
'Active cleaning of label noise'	Yes	No	13	6.50	A method based on SVM approach that locates data which may de outliers is proposed. The data that identified as 'suspicious' are removed from the dataset (Ekambaram *et al.*, 2016)
'Noise detection in the meta-learning level'	Yes	No	12	6.00	The authors through the creation of meta-features provided a method that predicts the noisy data (Garcia *et al.*, 2016a)
'Ensembles of label noise filters: a ranking approach'	Yes	No	10	5.00	This study examines the possibility of improving the noise detection process through an ensemble of noise filtering methods. To this end, the authors conducted extensive experiments (Garcia *et al.*, 2016b)
'INFFC: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control'	Yes	No	28	14.00	An iterative scheme that combines several classifiers for the detection of noise is proposed. The amount of examples that can be considered as noisy is not stable and it depends on the user (Sáez *et al.*, 2016)

TC = number of the total citations; CpY = number of citations per year; *k*-NN = *k*-nearest neighbor.

(c) Mean substitution of a numerical attribute: Fill in missing feature values with the feature's mean value that is computed using the available values. Alternatively, instead of using the 'general' feature mean, it can be used the feature mean of the samples that belong to the same class.
(d) Regression method for a numerical attribute or classification methods for a categorical attribute: Develop a regression model for the numerical attribute case, fill in the blanks of the dataset by taking the decision-outcome of the model that is builded in using all the other known features as predictors. Similarly, develop the classification model for the categorical case (Honghai *et al.*, 2005; Silva-Ramírez *et al.*, 2011).
(e) Method of treating missing feature values as special values: In this case, the unknown elements of the dataset are considered as complete new values for the features that contain missing values.

Furthermore, we point out that in Farhangfar *et al.* (2008) an inclusive experimental study has been performed which is based on imputation methods to tackle the missing value problem in 15 discrete datasets. Their experiments have shown that on average the imputation improves the subsequent classification. Various other algorithms have been designed to impute target missing values such as the *k*-NN variant algorithm, the shell neighbors imputation (SNI) algorithm (Zhang, 2011) and the random method for single imputation (Qin *et al.*, 2009). Regarding the approach of Zhang (2011) the author proposed a variation of the *k*-NNI algorithm. In this case only two neighbors take part in the selection of the most neighboring from the missing data value. Specifically, the left and the right one. These points are calculated using the shell neighbor definition. This constitutes the main difference with the *k*-NNI algorithm which takes the *k* neighbors, where *k* is fixed. The experimental results over mixed datasets shown that the SNI algorithm overcomes the *k*-NNI algorithm with respect to classification and imputation accuracy. Concerning the approach of Qin *et al.* (2009), the authors through experiments that they conducted, compared the kernel-based method that they proposed with several non-parametric methods. The results showed that their non-parametrical optimization (POP) algorithm outperforms the other methods regarding statistical parameters, such as mean, distribution function and quantile after missing data and imputation efficiency.

An imputation approach named NIIA has been proposed in Zhang *et al.* (2011). This method imputes missing data by using information within incomplete instances. It is an iterative imputation scheme that has been designed for imputing iteratively the missing target values. It imputes each missing value successively until its convergence.

In Luengo *et al.* (2012), the authors have tried to solve the missing value problem through the usage of different imputation methods. Specifically, they have focused on a classification task and their experimental results have shown that specific missing values imputation methods provide better results regarding the accuracy.

In Lobato *et al.* (2015), the authors have presented for the first time a solution to the problem of missing values by combining evolutionary computation techniques, such as genetic algorithms (GA), for data imputation. In particular, the authors have proposed a multi-objective GA, named MOGAImp, which is suitable for mixed-attribute datasets.

A very difficult and time-consuming problem constitutes the selection of the optimal combination between classification and imputation methods. A new, alternative scheme that proposes such a combination adaptively has been introduced in Sim *et al.* (2016). To build this new scheme and to find the proper pair among the imputation method and the classifier, the authors conducted a set of testing experiments. An important difference concerning the past attempts was that the authors used meta-data to cover missing data. Thus, they avoid the extremely high amount of executing time. In addition, another approach that covers the incomplete data through adaptive imputation based on belief function theory has been presented in Liu *et al.* (2016). Specifically, the authors handle the problem of classification of incomplete patterns covering the missing values through the *k*-NN and self-organizing map methods. This occurs only in the cases where the available information are not enough for the classification of a given object.

Table 2 exhibits a brief description and information about the citations that the most known and widely used methods, related to the missing feature values issues, have received.

Table 2 Brief description and number of citations of the references related to the missing feature values issues presented in Section 3

Title	TC	CpY	Brief description
'A SVM regression based approach to filling in missing values'	60	4.61	An SVM regression model is used to predict the missing values of a given dataset (Honghai *et al*., 2005)
'Impact of imputation of missing values on classification error for discrete data'	214	21.4	The authors focused their study on discrete data, to estimate the impact of several imputation approaches in the procedure of classification. Their study consist of extensive experiment including the comparison between well-known imputation methods (Farhangfar *et al*., 2008)
'Missing value imputation based on data clustering'	69	6.9	A clustering-based method, regarding the filling of missing, data is provided. To achieve this, the authors used the most similar attribute with no missing values to those with missing values (Zhang *et al*., 2008)
'POP algorithm: kernel-based imputation to heat missing values in knowledge discovery from datasets'	41	4.55	A kernel-based method is proposed and compared with different non-parametric approaches (Qin *et al*., 2009)
'Missing data imputation using statistical and machine learning methods in a real breast cancer problem'	175	21.875	A comparison between two categories is made: the statistical and machine learning methods. The methods were tested on a real-world problem, that of breast cancer (Jerez *et al*., 2010)
'Missing value imputation on missing completely at random data using multilayer perceptron'	84	12	A comparison between three well-known imputation methods and an automated model based on ANNs is made. The experiments that carried out, showed that MP model gives better results (Silva-Ramírez *et al*., 2011)
'Missing data imputation by utilizing information within incomplete instances'	75	10.71	The proposed algorithm is an iterative scheme that imputes the missing target values iteratively. To make this happen, the algorithm exploits the useful information inside the incomplete data (Zhang *et al*., 2011)
'Shell-neighbor method and its application in missing data imputation'	103	14.71	A *k*-NN variant algorithm for filling the missing data is proposed. To achieve this, the author tried to improve previous attempts taking the most neighboring data (only the left and the right), regarding the missing target values (Zhang, 2011)
'On the choice of the best imputation methods for missing values considering three groups of classification methods'	129	21.50	The data imputation task using several imputation approaches is tackled. This study is focused on the classification problem (Luengo *et al*., 2012)
'Missing data in medical datasets: impute, delete or classify?'	49	9.80	The proposed scheme is appropriate to select which of the missing data will be completed and which ones are not. To manage this goal, the authors used a statistical classifier and fuzzy modeling to be more accurate (Cismondi *et al*., 2013)
'Multi-objective genetic algorithm for missing data imputation'	23	7.67	In this study, for the first time the data imputation problem using an evolutionary algorithm is addressed (Lobato *et al*., 2015)
'Adaptive imputation of missing values for incomplete pattern classification'	84	42	The problem of classification of incomplete patterns covering the missing values through the *k*-NN and SOM methods is handled (Liu *et al*., 2016)
'Adaptive pairing of classifier and imputation methods based on the characteristics of missing values in data sets'	5	2.5	In this approach, meta-data have been used to fill in missing data. An adaptive model selects the appropriate pair of classifiers for this purpose (Sim *et al*., 2016)

TC = number of the total citations; CpY = number of citations per year; SVM = support vector machine; ANN = artificial neural network; *k*-NN = *k*-nearest neighbor.

4 Normalization and discretization

In various datasets that we have to manage, there are very often large differences between the feature values, such as the maximum and minimum value, for example, 0.001 and 10 000. In general, this issue is not desirable and requires careful intervention to make a scaling down transformation in such a way that all attribute values to be appropriate and acceptable. This process is known as feature scaling or data normalization (in the case of data preprocessing) and it is necessary and very important for various classifiers, such as neural networks, SVMs, *k*-NN algorithms, as well as fuzzy classifiers which cannot perform well if large differences between the feature values occur in the dataset.

The most common methods for this scope are the following:

(a) min–max normalization or feature scaling in [0, 1]. The formula that gives the new feature value is the following:

$$v' = \frac{v - \min\_\text{value}}{\max\_\text{value} - \min\_\text{value}}$$

(b) min–max normalization or feature scaling in [*a*, *b*]. The formula that gives the new feature value is the following:

$$v' = \frac{v - \min\_\text{value}}{\max\_\text{value} - \min\_\text{value}}(\text{new}\max\_\text{value} - \text{new}\min\_\text{value}) + \text{new}\min\_\text{value}$$

(c) *z*-score normalization or standardization. The formula that gives the new standardization feature value is the following:

$$v' = \frac{v - \text{mean}}{\text{stand\_dev}}$$

(d) unit length scaling:

$$v' = \frac{v}{\text{P}\,v\,\text{P}}$$

where '*v*' is the original feature value while '*v*′' is the new, normalized data value, 'min _value' and 'max _value' are, respectively, the minimum and maximum feature values before the normalization process. In the second method, the interval [*a*, *b*] indicates a specific range that we want to transform into the original dataset. So, 'new max _value' and 'new min _value' point out the desirable new maximum and new minimum value of the normalized dataset (the *a* and *b* value, respectively). Moreover, in method (c) 'mean' denotes the mean value of the feature vector and 'stand_dev' represents the standard deviation. Essentially, this method constitutes a measurement of how many standard deviations the value is from its mean value. Finally, the 'P · P' indicates a norm, for example, the Euclidian length of the feature vector or the Manhattan distance. In conclusion, selecting one of the above normalization methods is clearly dependent on the dataset we want to normalize (or transform).

Another important issue to address is that of values of the continuous features of a dataset. Usually, the features of the sample can get continuous values or values from a wide range. This may require a time-consuming procedure to handle or, in many cases, may be inefficient for the processes used in the field of predictive DM. Thus, the main outcome of these and a solution to the above problem is the discretization process.

The discretization process can actually be very useful to a variety of classifiers, such as decision trees (DTs) or Bayesian classifiers. On the other hand, the scientific community needs to resolve several other issues about this process, as identifying the most appropriate interval borders or the right arity for the discretization of a numerical value range. These issues are open to resolution and extensive study. For more details about the discretization processes, the reader is referred to Liu *et al.* (2002), Kurgan and Cios (2004) and Liu and Wang (2005).

The main difference between supervised and unsupervised discretization methods is that the methods of the first category discretize the features of the dataset regarding the class in which they belong, while the other ones do not. On the other hand, the distinction of methods in top-down and bottom-up methods concerns the initialization of the starting interval. In the top-down methods, the initialization takes place to the starting interval and retrospectively cleaved into smaller ones. On the other hand, in the bottom-up methods the initialization of single value intervals takes place followed by merging of adjacent intervals. Moreover, the quality of the discretization processes is measured using two basic criteria: (i) the classification performance and (ii) the number of discretization intervals.

Two of the most well-known and widely used discretization methods are (Dougherty *et al.*, 1995): (i) the equal size and (ii) the equal frequency. Both of them are unsupervised methods. The first method, as its name implies, finds the maximum and the minimum value of the attributes, and then divides the found range into *k* equal-sized intervals. The second one finds the number of values of the attribute and then it separates them into intervals which contain the same number of instances.

A very important issue regarding the top-down and the bottom-up methods is that they require the involvement of the user with respect to a set of parameters, such as the modification of the criteria for stopping the discretization processes. A well-known bottom-up discretization algorithm, named Khiops has been presented in Boulle (2004). The advantage of this method with respect to the other methods of the same category, and especially the ChiSplit and the ChiMerge, is that it uses as stopping criterion the χ^2 statistical test. As a result, this method optimizes a global criterion and not a local one. Moreover, it does not require any parameters to be specified by the user. Furthermore, it is not lagging in terms of time complexity (as it is demonstrated by the experimental results) in accordance with other methods that optimize a local criterion. This makes Khiops a very useful and easy to be handled. Another well-known method of the same family, called Ameva, manages to potentially produce the minimum number of intervals, has been presented in Gonzalez-Abril *et al.* (2009). Specifically, this method has been compared with the known algorithm, called CAIM and also with other genetic approaches. The experimental results have shown that the Ameva always produces the smallest number of discrete intervals and even, in cases with a large number of classes is not lagging in terms of complexity. Regarding the performance of Ameva in relation with the genetic approaches, the performance of these algorithms is very similar.

In Tsai *et al.* (2008), the authors have proposed an incremental, supervised and top-down discretization algorithm through which better classification accuracy has been achieved. This algorithm is based on the class-attribute contingency coefficient and it is combined with a greedy method that has also exhibited good execution time results. As we have already mentioned, the measurement of the quality of the discretization algorithms is one of the issues that deserves a thorough attention. The experiments that have been carried out in Jin *et al.* (2009) have shown that more discretized intervals usually equate to fewer classification errors and as a result lower the cost of the data discretization process.

In Janssens *et al.* (2006) the concepts of (i) error-based, (ii) entropy-based and (iii) cost-based discretization methods have been analyzed. The methods of the first category, group the values into one interval aiming through the minimization of the errors in the training set to achieve the optimal discretization. On the other hand, entropy-based methods, use the entropy value of a point to find the min–max value of an interval. Like top-down methods, the entropy discritization methods, divide the starting interval into smaller ones by choosing the values which minimize the entropy until the intervals become optimal or some specific criteria to be fulfilled (Kumar & Zhang, 2007). This category is widely used and the notion of entropy has been used also in Gupta *et al.* (2010) and de Sá *et al.* (2016) to exploit the inter-attribute dependencies of the data as well as to discretize the rank of data. On the other hand, with the error-based methods of the third category, the error is taken into account as a cost of misclassification.

The Bayesian classifiers are among the most known classifiers in DM, and therefore the researchers have been highly involved in this category of algorithms. Specifically, to improve the rate of accuracy in the prediction of naive Bayes classifier, in particular in reducing the variance and the bias, in Yang *et al.* (2009) it has been proposed a scheme that combines proportional discretization and fixed frequency discretization. Another attempt to improve the accuracy of naive Bayes classifier has been done in Wong (2012). The author, in his attempt to present a relationship that would explain the level of dependence between the class and the continuous feature, has proposed a method which uses a non-parametric measure. With this measure the author eventually has achieved the improvement of the classifier. Furthermore, through the extensive experimental research that it has been carried out in Mizianty *et al.* (2010), an important observation regarding the discretization process and its effect on both naive and semi-naive Bayesian classifiers has been obtained. The authors have shown that despite the time-consuming process of discretization relative to training time of classifier, the whole process often is completed faster than naive Bayes classification process.

A method that performs the discretization process automatically, based on two factors: (i) the dispersion as well as (ii) the uncertainty of the range of the data has been presented in Augasta & Kathirvalavakumar (2012). The experiments conducted by the authors that are based on real-world data have shown that the proposed algorithm approaches to give the smallest number of discrete intervals, and the discretization time is less than other algorithms. In addition, in Li *et al.* (2011) the authors have proposed the class-attribute coherence maximization (CACM) criterion and the corresponding discretization algorithms: (i) the CACM algorithm and (ii) the efficient-CACM algorithm. The extensive experiments they have done show that their method surpasses the performance of other well-known classifiers such as RBF-SVM and C4.5. In addition, the variation that they gave was faster compared with the fast-CAIM method.

Furthermore, in Cano *et al.* (2016) an improved version of the CAIM algorithm has been proposed. The new algorithm, named ur-CAIM achieves more flexible discretization, it presents more qualitative sub-intervals and it has better classification results for unbalanced data. In addition, a more complete and more detailed survey about discretization methods has been presented in Yang *et al.* (2009), as well as a taxonomy and empirical analysis of these methods has been presented in Garcia *et al.* (2013). Also, an updated overview of discretization techniques in conjunction with taxonomy has been presented in Ramírez-Gallego *et al.* (2016).

Table 3 exhibits a brief description and information about the citations that the most known and widely used methods, related to the normalization and discretization issues, have received.

5 Instance selection

The instance selection approach is not only used to handle noise (Aridas *et al.*, 2016; Aridas *et al.*, 2017) but also to deal with the infeasibility of learning from huge datasets (Wu & Zhu, 2008). The training time required by the neural networks and SVMs as well as the classification time of instance-based learners is clearly affected by the size of the dataset. In this case, instance selection can be considered as an optimization problem that attempts to maintain the quality along with minimizing the size of the dataset (Liu & Motoda, 2002). Moreover, by increasing the number of instances the complexity of the induced model is risen resulting to the decrease of the interpretability of the results. Thus, instance selection is highly recommended in the case of big datasets as it is noted in García-Pedrajas and PéRez-RodríGuez (2012).

The sampling is a widely used process since it constitutes a powerful computationally intense procedure operating on a sub-sample of the dataset and is able to maintain or even to increase the accuracy (Klinkenberg, 2004). There is a variety of procedures for sampling instances from a large dataset (over a Gb). The most well-known are the following (Cano *et al.*, 2005): (i) the random sampling that selects a subset of instances randomly and (ii) the stratified sampling in the case where the class values are not uniformly distributed in the dataset.

In Reinartz (2002), the author has presented a unifying framework, which covers approaches related to instance selection. Furthermore, in Kim and Oommen (2003) and in García *et al.* (2012a) a taxonomy and a ranking of prototype reduction schemes have been also provided. In general, instance selection methods have been grouped into different categories according to Olvera-López *et al.* (2010):

(a) Type of selection: (i) The condensation methods that try to compute a consistent subset by removing instances that cannot increase the classification accuracy, (ii) the edition methods that try to remove noisy instances and (iii) the hybrid methods that search for a subset in which both noisy and useless instances occur.
(b) Direction of search: (i) The incremental methods that start with an empty set and add instances according to some criterion, on the contrary (ii) the decremental methods that start with the whole set

and remove instances according to some criterion and (iii) the mixed methods that start with a subset and add or remove instances that fulfill a certain criterion.

(c) Evaluation of search: (i) The wrapper methods that consider a selection criterion based on the accuracy obtained by a classifier and (ii) the filter methods that use a selection function that is not based on a specific classifier.

In Bezdek and Kuncheva (2001), the authors have compared 11 methods for finding prototypes upon which the nearest-neighbor classifier is based. In this comparison, several methods have exhibited a good performance and none of them is superior to all the others. In the case of binary problems, an important observation has been presented in Pkekalska *et al.* (2006). Specifically, according to their observation the dissimilarity-based discrimination functions relying on reduced prototype sets (3–10% of the training instances) offer a similar or much better classification accuracy than the *k*-NN applied on the entire training set. Also, the selection rule is related to the *k*-NN classifier in most wrapper algorithms (Derrac *et al.*, 2010b).

A well-known algorithm for instance selection is the HitMiss network (HMN) algorithm that it has been proposed in Marchiori (2008). This algorithm focuses on the redundancy removal rather than the noise reduction. Specifically, the proposed algorithm was used to improve the performance of the 1-NN algorithm. The author proposed three variants of the original algorithm. Each of them provided separate advantages and improvements with respect to the 1-NN algorithm. For example, the variant called HMN-C deleted instances from the dataset but nevertheless it did not affect the accuracy of the 1-NN algorithm. Or the HMN-EI version that it repeatedly ran the HMN-E version, which was the second one that made storage reduction. All the three variants were tested extensively on multiple, diverse datasets. The results showed that the proposed algorithm has a better generalization ability than other known instance selection algorithms. In addition, it greatly improves the accuracy of the 1-NN classifier. Furthermore, in Nikolaidis *et al.* (2011), the authors have proposed the class boundary preserving algorithm. This approach uses the concept of reachable set and proposes an extension involving more than just the nearest enemy. Then, by using the multiple reachable sets and the nearest enemies establishes geometric structure patterns to remove redundant instances.

In de Haro-García & García-Pedrajas (2009) an instance selection filtering strategy has been proposed. In this strategy, the main idea is to divide the dataset into small mutually exclusive blocks and then individually to apply an instance selection algorithm to each block. Afterward, the instances that are selected from each block are merged in subsets of about the same size and the instance selection algorithm is applied again. The process is repeated until a validation error starts to increase.

In Czarnowski (2012) a clustering approach for instance selection has been presented. Initially, this approach makes groups of instances of each class into clusters and then, in the second step a wrapper process is used. Furthermore, an evolutionary instance selection algorithm has been proposed in García *et al.* (2008). Specifically, this algorithm uses a memetic algorithm that combines evolutionary algorithms and local search strategies. Due to this combination, the inherent convergence problem is avoided that plagues evolutionary algorithms. In fact, very good results are achieved as the number of data increases. An additional evolutionary instance selection algorithm is the cooperative coevolutionary instance selection algorithm that has been proposed in García-Pedrajas *et al.* (2010). In this approach, the technique of divide and conquer is utilized. Thus, the original dataset is subdivided into smaller ones, and a global optimization technique, based on populations, is looking for the best combination. The advantage of this method is that the user can control the objective functions (such as accuracy or storage requirements) through the fitness function. In addition, this approach does not require the calculation of a distance metric or the usage of a particular classification algorithm. The experimental results have shown that the proposed algorithm is not inferior to any other of the compared algorithms. In addition, it is superior in requirement terms and it is promising in the case of large datasets regarding to the terms of computational cost and efficiency.

For the characterization of datasets various measures have been proposed in Caises *et al.* (2011). These measures are used to select from some pre-selected methods, the method (or combination of methods) that is expected to produce the best results. Furthermore, an instance ranking method per class using borders

Table 3 Brief description and number of citations of the references related to the normalization and discretization issues presented in Section 4

Title	TC	CpY	Brief description
'Discretization: an embeding technique'	922	57.625	In this study widely used discretization methods are provided, as well as their application mainly to the classification task. Moreover, the authors make a categorization of these techniques and study aspects, such as accuracy and speed (Liu *et al.*, 2002)
'KHIOPS: a statistical discretization method of continues attributes'	142	10.14	The author proposed a new algorithm that can handle continues attributes. This method uses the χ^2 statistical test and optimizes a global criterion (Boulle, 2004)
'CAIM discretization algorithm'	471	33.64	The proposed method has been designed for supervised data. The authors created an algorithm that separates the dataset using intervals. The dataset is discretized without the assistance of a user (Kurgan & Cios, 2004)
'A discretization algorithm based on a heterogeneity criterion'	52	4.00	A new heuristic approach to find the optimal discretization scheme is proposed. In addition, the authors proposed a new criterion based on the heterogeneity definition (Liu & Wang, 2005)
'Evaluating the performance of cost-based discretization versus entropy-and error-based discretization'	56	4.67	The authors compared two well-known discretization methods: the cost-based method and the entropy- and pure error-based one (Janssens *et al.*, 2006)
'Hand-geometry recognition using entropy-based discretization'	78	7.10	In this method the original dataset is divided into smaller intervals based on the entropy (Kumar & Zhang, 2007)
'A discretization algorithm based on class-attribute contingency coefficient'	192	19.2	The authors proposed an algorithm which is based on class-attribute contingency coefficient, to achieve better classification accuracy (Tsai *et al.*, 2008)
'AMEVA: an autonomous discretization algorithm'	82	9.11	This new approach has been compared with the well-known CAIM algorithm (Kurgan & Cios, 2004) to obtain which one provide the minimum interval division (Gonzalez-Abril *et al.*, 2009)
'Data discretization unification'	79	8.78	The authors provided a dynamic algorithm which manages the best discretization based on entropy. They concluded that more discretized intervals occur in most of the times to fewer classification errors (Jin *et al.*, 2009)
'Discretization for naive-bayes learning: managing discretization bias and variance'	174	19.33	A combination of proportional discretization and fixes frequency discretization is provided, to improve the classification of naive Bayes (Yang *et al.*, 2009)
'Discretization methods'	80	8.88	A survey paper that outlining basic method and aspects of discretization process are presented (Yang *et al.*, 2009)
'A clustering-based discretization for supervised learning'	41	5.125	The authors, through the usage of *k*-means clustering algorithm provided a clustering based discretization scheme which uses the entropy and description length criterion (Gupta *et al.*, 2010)
'A survey of discretization techniques: taxonomy and empirical analysis in supervised learning'	275	54.6	In this survey work, the most representative and up-to-date discretization methods are provided (Garcia *et al.*, 2013)

‘ur-CAIM: improved CAIM discretization for unbalanced and balanced data’	27	13.5	A variant of the well-known CAIM algorithm is proposed. The experiments conducted by the authors showed that this new version achieves better classification results than the basic CAIM approach (Cano *et al*., 2016)
‘Entropy-based discretization methods for ranking data’	25	12.5	The authors proposed a new supervised discretization scheme based on the minimum description length principle (de Sá *et al*., 2016)
‘Data discretization: taxonomy and big data challenge’	49	24.5	The aims of this work are: firstly to illustrate the most used discretization methods and secondly to highlight the possibility of parallelization to be applied in the Big Data case (Ramírez-Gallego *et al*., 2016)

TC = number of the total citations; CpY = number of citations per year.

has been introduced in Hernandez-Leal *et al.* (2013). The border instances are those nearby to the decision boundaries between the classes and are the most useful for the learning algorithms.

In Lin *et al.* (2015), the authors have introduced the representative data detection approach, which is based on outlier pattern analysis and prediction. A learning model is initially trained to learn the patterns of (un)representative data that are selected by a specific instance selection method from a small amount of training data. Afterwards, the leaner can be used to select the rest useful instances of the large amount of training data.

In Arnaiz-González *et al.* (2016), two new algorithms with linear complexity for instance selection purposes have been presented. Both algorithms use locality-sensitive hashing to find similarities between instances. Also, in Tsai and Chang (2016), the authors have investigated the effect of performing instance selection to filter out some noisy data regarding the imputation task. In particular, the authors have examined whether the instance selection process should precede or follow the imputation process. For this purpose, they have provided four ways/combinations to find which one is the best. This has the effect of identifying/creating the most appropriate training set for the classifier that used. The authors have performed extensive experiments to find the best combination on a variety of datasets. The four combinations have been tested for the *k*-NN and SVM classifier.

In Liu *et al.* (2017), the authors have studied the support vector recognition problem mainly in the context of the reduction methods to reconstruct the training set. The authors have focused on the fact of uneven distribution of instances in the vector space to propose an efficient self-adaption instance selection algorithm from the viewpoint of geometry-based method.

Furthermore, recently, in Fernández *et al.* (2017) the authors have used a multi-objective evolutionary algorithm to obtain an optimal joint set of both features and instances. Specifically, the proposed methodology has been applied to imbalanced datasets. Using the well-known C4.5 classifier combined with the well-known NSGA-II evolutionary method, they were able to expand the search space, which enables to build an ensemble of classifiers. Thus, through FS overcomes the overlapping problem that is inherent in multi-class imbalanced datasets as well as through the instance selection two problems have been solved, namely the elimination of noise and the imbalance issue. The authors have conducted experiments including comparisons with widely used and well-known classifiers on binary-class and multi-class imbalanced datasets. The experimental results have shown that the proposed method named EFIS-MOEA exhibits a better performance than the compared ones and is promising in terms of its applicability to big datasets.

In general, the prototype generation (PG) approach builds new artificial prototypes to increase the accuracy. In Triguero *et al.* (2012b), the authors have provided a survey of PG methods that are specifically designed for the nearest-neighbor algorithm. Furthermore, they have conducted an experimental study to measure the performance in terms of accuracy and reduction capabilities.

In Nanni and Lumini (2011), instance generation for creating an ensemble of the classifiers has been used. The training phase consists in repeating *N* times the PG, and then the scores resulting from classifying a test instance using each set of prototypes are combined by voting.

In general, the classifiers are expected to be able to generalize over unseen instances of any class with equal accuracy, which constitutes an ideal situation. Also, in many applications learners are faced with imbalanced datasets, which can cause the learner to be biased towards one class. This bias is the result of one class being greatly under-represented in the training data compared to the other classes. It is related to the way in which learners are designed. Inductive learners are typically designed to minimize errors over the training examples. Classes that contain few examples can be mostly ignored by learning algorithms since the cost of performing well on the over-represented class outweighs the cost of doing poorly on the smaller class (He & Garcia, 2009).

We point out that the imbalanced datasets have recently received attention in various ML tasks (Lemaître *et al.*, 2017). There are various procedures to cope with the imbalanced problem in datasets include among others the following (Sun *et al.*, 2009): (i) duplicate training examples of the under-represented class which is actually re-sampling the examples (over-sampling), (ii) remove training examples of the over-represented class which is referred to as downsizing (under-sampling), (iii) combine of both over- and under-sampling and (iv) ensemble learning approaches.

In Chawla *et al.* (2002), the authors have proposed the synthetic minority over-sampling technique (SMOTE) where synthetic (artificial) samples are generated rather than over-sampling. SMOTE generates the same number of synthetic data samples for each original minority instance. This happens without consideration to neighboring examples, which increases the occurrence of overlapping between classes (Wang & Japkowicz, 2004).

In Farquad and Bose (2012), the authors have employed SVM as a preprocessor. Their approach replaces the actual target values of the training data by the predictions of the trained SVM. Next, the modified training data are used to train other learners.

In van Hulse and Khoshgoftaar (2009), a comprehensive experimental investigation using noisy and imbalanced data has been presented. The results of the experiments demonstrate the impacts of the noise on learning from imbalanced data. Specifically, the noise resulting from the corruption of instances from the true minority class is seems to be the most severe. In Cano *et al.* (2008), the authors have proposed the combination of the stratification and the instance selection algorithms for scaling down the dataset. Two stratification models have used to increase the presence of minority classes in datasets before the instance selection.

As we have already mentioned in the introduction and as it is noted in the literature (López *et al.*, 2012), the problem of managing imbalanced datasets is one of the most important and most difficult problems in DM. The classification problem is a widely encountered problem with so many aspects. The most common approaches include: data sampling or appropriate modification of the algorithm to be able to handle imbalanced datasets. According to the authors of López *et al.* (2012), the solution seeks to the suitable combination of the above approaches. In particular, they have observed that learning solutions which are cost-sensitive if they combined with data sampling and algorithmic modification, can reach better mis-classification costs in the minority class. In addition, they have minimized the high-cost errors.

Experiments over 17 real datasets using eight different classifiers have shown that over-sampling the minority class outperforms under-sampling the majority class when the datasets are strongly imbalanced. On the other hand, there are not significant differences for the databases with a low imbalance (García *et al.* 2012a).

Table 4 exhibits a brief description and information about the citations that the most known and widely used methods, related to the instance selection issues, have received.

6 Feature selection

It is well-known and widely recognized that several learners, such as the *k*-NN procedure, is very sensitive relative to irrelevant features. In addition, the presence of irrelevant features can make SVMs and neural network training very inefficient and in many cases impractical. Furthermore, the Bayesian classifiers do not perform well in the case of redundant variables. To tackle these issues, the FS approach is proposed to be used. The FS is the procedure of identifying and removing as many irrelevant and redundant features as possible. This reduces the dimensionality of the data and facilitates learning algorithms to operate faster and more effectively. In general, the features can be distinguished as follows (Hua *et al.*, 2005):

(a) Relevant: The features that have an influence on the class and their role cannot be assumed by the rest.
(b) Irrelevant: Irrelevant features that do not have any influence on the class.
(c) Redundant: A redundancy occurs whenever a feature can take the role of another (perhaps the simplest way to model redundancy). In practice, it is not so straightforward to determine feature redundancy when a feature is correlated (perhaps partially) with a set of features.

In general, the FS algorithms consist of two approaches (Hua *et al.*, 2005): (i) a selection algorithm that generates the proposed subsets of features to find an optimal subset and (ii) an evolutionary algorithm that decides about the quality of the proposed feature subset by returning some 'measure of goodness' to the selection algorithm. On the other hand, without the application of a proper stopping criterion, it is possible the FS process to run in an exhaustively way or without termination through the space of subsets. Usually, the stopping criteria are: (i) either the addition (or the deletion) of any feature that does not offer a better subset and/or (ii) whether an optimal subset has been found according to some evaluation function. In

Table 4 Brief description and number of citations of the references related to the instance selection issues presented in Section 5

Title	TC	CpY	Brief description
'Nearest prototype classifier designs: an experimental study'	188	11.06	This paper gives a detailed comparison between several approaches for finding prototypes (Bezdek & Kuncheva, 2001)
'SMOTE: synthetic minority over-sampling technique'	6422	401.375	The well-known SMOTE algorithm where artificial samples are generated rather than over-sampling is proposed (Chawla *et al*., 2002)
'On issues of instance selection'	196	12.25	In this paper main issues and developments regarding the instance selection techniques are addressed (Liu & Motoda, 2002)
'A unifying view on instance selection'	119	7.44	A framework which includes instance selection approaches is provided, as well as by generating sampling and specific examples it points out their advantages and disadvantages (Reinartz, 2002)
'Learning drifting concepts: example selection vs example weighting'	454	32.49	An intelligent model based on SVM classifier is proposed, to filter and adapt changeable situations (Klinkenberg, 2004)
'Prototype selection for dissimilarity-based classifiers'	326	27.17	Extensive experiments on several metric and non-metric dissimilarity representation are conducted, as well as prototype selection schemes are provided (Pkekalska *et al*., 2006)
'A memetic algorithm for evolutionary prototype selection: a scaling up approach'	161	16.10	An efficient combination of evolutionary algorithms and local search techniques to tackling the prototype selection task is proposed (García *et al*., 2008)
'A survey on evolutionary instance selection and generation'	135	16.875	This work presented the useful role of evolutionary algorithms regarding data reduction task. The authors provided a number of applications of data mining where the evolutionary algorithms have applied (Derrac *et al*., 2010b)
'A review of instance selection methods'	235	29.375	This is a survey work where the authors summarize the most known methods of instance selection (Olvera-López *et al*., 2010)
'Prototype selection for nearest neighbor classification: Taxonomy and empirical study'	447	74.50	The authors presented an extensive review paper by considering the most used prototype selection methods for helping nearest-neighbor learner to surpass an amount of inherent disadvantages (Garcia *et al*., 2012a)
'Analysis of preprocessing vs cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics'.	157	26.17	In this approach the data sampling property with well-known classifiers seems to resolve the handling of imbalanced data (López *et al*., 2012)
'A taxonomy and experimental study on prototype generation for nearest neighbor classification'	195	32.50	The authors presented a survey of prototype generation methods regarding the nearest-neighbor classifier (Triguero *et al*., 2012b)
'Instance selection of linear complexity for big data'	22	11.00	In this work, two new algorithms which examine the characteristics of the instances to find similarities, are proposed. Both algorithms use locality-sensitive hashing to achieve the above purpose (Arnaiz-González *et al*., 2016)
'Combining instance selection for better missing value imputation'	14	7.00	The authors examined whether the instance selection process should proceed or follow the imputation process (Tsai & Chang, 2016)
'An efficient instance selection algorithm to reconstruct training set for support vector machine'	15	15.00	In this work the support vector recognition problem regarding the reduction methods, to reconstruct the training dataset has been studied (Liu *et al*., 2017)

TC = number of the total citations; CpY = number of citations per year.

Piramuthu (2004), various FS techniques have been compared concluding that none of them is superior to all the others.

Regarding the selection strategy, the filtering methods rank features independently of the classifier. In the filtering methods, a feature can be selected by taking into consideration some predefined criteria such as: (i) mutual information (Chow & Huang, 2005), (ii) class separability measure (Mao, 2004) or (iii) variable ranking (Caruana & Sa, 2003).

On the other hand, the wrapper methods use a classifier to assess feature subsets. These methods are computationally consuming in practice and they are not independent of the type of classifier used (Liu & Motoda, 2007). The wrapper methods use the cross-validation approach to predict the benefits of adding or removing a feature from the feature subset used. In the case of the forward stepwise selection, a feature subset is iteratively build up. Each of the unused features is added to the model in turn, and the feature that most improves the accuracy of the model is chosen. In the case of the backward stepwise selection, the algorithm starts by building a model that includes all the available input features. In each iteration, the algorithm locates the variable whose its absence causes the most improvement of the performance (or causes least deterioration). A problem with the forward selection is that it may fail to include features that are interdependent since it adds one variable at a time. On the other hand, it is able to locate small effective subsets quite soon, since the early evaluations that involve relatively few features are fast. On the contrary, in the backward selection interdependencies are well handled, but at the beginning the evaluations are computationally consuming.

Various FS methods treat the multi-class case directly rather than decomposing it into several two-class problems. The sequential forward floating selection and the sequential backward floating selection are characterized by the changing number of features that are included or are eliminated at different stages of the procedure (Somol & Pudil, 2002). In Maldonado and Weber (2009), the authors have introduced a wrapper algorithm for FS by using SVMs with kernel functions. Their method is based on a sequential backward selection by using the number of misclassifications in a validation subset as the measure to decide which feature has to be eliminated in each step.

Furthermore, the GA is an additional well-known approach to tackle FS issues (Smith & Bull, 2005). In this approach, for each iteration a feature is chosen and it is decided whether to be included in the subset or to be excluded from it. All the combinations of unknown features are used with an equal probability. Due to the probabilistic nature of the search, a feature that should be in the subset will be superior of the others, even whether it is dependent on another feature. An important characteristic of the GA is that it is designed to exploit the epistasis (that is the interdependency between bits in the string), and thus is well-suited for the FS approach. On the other hand, GAs typically require a large number of evaluations to reach a minimizer.

To combine the advantages of the filter and the wrapper models, various hybrid models have been proposed to deal with high-dimensional data. In Unler *et al.* (2011), the authors have presented a hybrid filter-wrapper feature subset selection algorithm which is based on the particle swarm optimization (PSO) (Kennedy & Eberhart, 1995; Parsopoulos & Vrahatis, 2010) for the SVM classification. The filter model is based on the mutual information and is a combined measure of feature relevance and redundancy with respect to the selected feature subset. Also, the wrapper model can be considered as a modified discrete PSO algorithm.

In Hu *et al.* (2010), the authors have proposed a concept of neighborhood margin and neighborhood soft margin to calculate the minimal distance between different classes. They have used the criterion of the neighborhood soft margin to estimate the quality of the candidate features and to build a forward greedy algorithm for FS.

Instead of approaching instance selection or FS problems separately, various research efforts have been made for the study of the simultaneous instance selection and FS (Derrac *et al.*, 2010a). To make this happen, the authors provided an evolutionary scheme that applied to the *k*-NN classification problem. The experimental results showed that the proposed method outperforms other well-known evolutionary approaches, such as FS-GGA, FS-SSGA, FS-CHC and 1-NN, regarding the data reduction process.

Furthermore, in Shu and Shen (2016), the authors, based on the rough set theory, have addressed the problem of the FS for cost-sensitive data with missing values. They have proposed a multicriteria evaluation function to characterize the significance of the candidate features. In particular, this occurs by

Table 5 Brief description and number of citations of the references related to the feature selection issues presented in Section 6

Title	TC	CpY	Brief description
‘A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms’	844	40.19	The evaluations made by the authors showed that the methods which use feature weighting perform better, have lower preprocessing requirements and need fewer training data (Wettschereck *et al*., 1997)
‘Orthogonal forward selection and backward elimination algorithms for feature subset selection’	134	9.57	Two orthogonal feature subset selection algorithms are proposed. The first one is based on orthogonal forward selection, while the other is based on backward elimination (Mao, 2004)
‘Evaluating feature selection methods for learning in data mining applications’	216	15.43	A number of feature selection methods are compared, concluding that no one outperforms all the others (Piramuthu, 2004)
‘Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information’	158	12.15	The authors provided an effective feature selection approach to compute efficiently high order statistic information (Chow & Huang, 2005)
‘Optimal number of features as a function of sample size for various classification rules’	186	14.31	In this study, the parallel computation of the useful feature subset take place. To achieve this, the authors applied several classifiers (Hua *et al*., 2005)
‘Genetic programming with a genetic algorithm for feature construction and selection’	101	7.77	The authors combined genetic programming with genetic algorithms as preprocessing step of DTs. In particular, they used genetic programming as feature constructor and a genetic algorithm as feature selector (Smith & Bull, 2005)
‘Reducing the dimensionality of data with neural networks’	7835	652.92	The authors ‘drop’ the dimensionality of the given dataset using MLP networks (Hinton & Salakhutdinov, 2006)
‘Kernel PCA for novelty detection’	424	38.54	The author through kernel PCA approach achieved to reduce a large feature space to a low-dimensional one (Hoffmann, 2007)
‘A wrapper method for feature selection using support vector machines’	223	24.78	In this study, the sequential backward selection process is used to eliminate step by step useless features (Maldonado & Weber, 2009)
‘mr^{2}PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification’	168	24.00	In this approach the well-known PSO algorithm is used as a preprocessing step of SVM classifier (Unler *et al*., 2011)
‘Multi-criteria feature selection on cost-sensitive data with missing values’	8	4.00	The authors provided a scheme to characterize the significance of the candidate features by talking into account their associated costs (Shu & Shen, 2016)

TC = number of the total citations; CpY = number of citations per year; PCA = principal component analysis; PSO = particle swarm optimization; SVM = support vector machine.

Table 6 Applicability/usability of preprocessing procedures regarding well-known learners

	LMs	DTs	ANN	RLs	Bayesian	SVM	LLs
Discretization	✗	✓	✗	✓	✓	✗	✗
Normalization	✓	✗	✓	✗	✗	✓	✓
Feature selection	✗	✗	*	✗	✗	*	*
Noise detection	✗	*	*	*	✗	✗	✗
Outlier detection	*	✗	✗	✗	✗	*	✗
Instance selection	✗	✗	✓	✗	✗	✓	✓
Missing data imputation	✓	✓	✓	✓	✗	✓	✓

LMs = linear models; DTs = decision trees; ANN = artificial neural networks; RLs = rule learners; SVM = support vector machines; LLs = lazy learners.

taking into consideration not only the power in the positive region and boundary region but also their associated costs.

In general, in the FS approaches, a feature takes a binary weight, where ‘1’ indicates that the feature has been selected and ‘0’ otherwise. However, feature weighting assigns a value, usually in the interval [0, 1] to each feature and the greater this value is, the more salient the feature will be. The usage of the weight-based models is widely used in various classification issues (Wettschereck *et al.*, 1997). In general, for this usage, the weighted neural networks can be considered as the most used example (Park, 2009), while, SVMs (Shen *et al.*, 2012) and nearest-neighbor methods (Mateos-García *et al.*, 2012) can also use weights for a better performance. Also, it is well known that the proper adjustment of the weights during the training process improves the model classification accuracy. In Triguero *et al.* (2012a), the authors have incorporated a new feature weighting scheme using two different prototype generation methodologies. In essence, the authors with the proposed hybrid evolutionary scheme improved the performance of other evolutionary approaches that have been used to optimizing the positions of the prototypes.

It is widely known that the DTs use the divide and conquer procedure, thus, they tend to perform well whether a few very relevant attributes exist, but less in the case where many complicate interactions exist. In general, the problem of the feature interaction can be tackled by constructing new features from the basic feature set. This procedure is called feature construction/transformation. In this case, the new generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to the improved comprehensibility of the produced classifier and the enhanced understanding of the learned concept.

In Smith and Bull (2005), the authors have used the approaches of the genetic programming and the GA as preprocessors of the dataset in the DTs. Specifically, for this case they have considered the C4.5 algorithm which is used to generate a DT. They have used the genetic programming as feature constructor and the GA as feature selector. Moreover, they have shown, by using 10 datasets that their framework performs better than the DTs alone. In addition, they have shown that effective preprocessing of the input data to DTs can significantly improve classification performance.

In general, PCA is a kind of analytical procedure based on a subspace and is able to estimate an original sample by using low-dimensional characteristic vectors (Hoffmann, 2007). Also, a large feature space can be converted to a low-dimensional one by training a multilayer neural network with small hidden layers. Thus, the well-known and widely used gradient descent optimization methods can then be used for fine-tuning the weights in such autoencoder networks (Hinton & Salakhutdinov, 2006). Hence, while PCA is restricted to a linear map, the autoencoders do not.

It has been shown that feature construction reduces the complexity of the space spanned by the input data. In Piramuthu and Sikora (2009), the authors have presented an iterative algorithm for improving the performance of any inductive learning process by using the feature construction as a preprocessing step. The choice between FS and feature construction depends on the application domain and the specific available training set. In general, FS leads to savings in measurements cost since some of the features are discarded and the selected features retain their original physical interpretation. In addition, the retained features may be important for understanding the physical process that produces the patterns. On the contrast, transformed features generated by feature construction are able to provide a better discriminative ability than the best subset of the given features. On the other hand, these new features may not have a clear physical meaning.

Table 5 exhibits a brief description and information about the citations that the most known and widely used methods, related to the FS issues, have received.

7 Discussion and concluding remarks

Predictive DM algorithms enable us to discover knowledge from large datasets subjected to appropriate conditions. The most important issue is the original dataset to be reliable. If the data that are received as input by the DM algorithms are unreliable or are contaminated by noise, then they cannot provide good and competitive results. In other words, the datasets should be of a high quality to obtain reliable models that will provide accurate results. For these reasons, data preprocessing is necessary to exploit predictive DM algorithms in knowledge discovery processes.

Since the quality of the dataset and the performance of the DM algorithms are directly related, considerable efforts have been conducted by several researchers to record some seeking points regarding the influence of each other. Therefore, through experimental research that have been conducted by Crone *et al.* (2006), it has been shown that the representation of the dataset affects quite a lot the learning methods and that the algorithm performance is better if efficient data preprocessing has been performed. In addition, the influence of preprocessing varies according to the algorithm. Thus, in the paper at hand the most well-known and widely used up-to-date algorithms for each step of the data preprocessing have been presented.

In the case where a dataset is huge, it may not be feasible to run a DM algorithm. To this end, it is necessary to reduce the amount of data, which can be done through the proper selection of the really useful data. This task can be accomplished by performing instance selection. Thus, through the appropriate selection of instances, irrelevant data or noise-containing data are avoided, and therefore the sample becomes of a higher quality and utilizable by a DM algorithm.

In most of the cases, missing data should be preprocessed to allow the whole dataset to be processed by a DM algorithm. In the case where the features of the dataset are continuous, the symbolic algorithms such as DTs or rule learners can be integrated with a discretization algorithm that transforms numerical attributes into discrete ones.

A very important result that came out by the comparison between several FS methods that has been presented in Hua *et al.* (2009), shows that none of the methods that have been considered for this comparison perform best against all the frameworks. Despite that fact, some correlations have been observed between specific FS methods and the size of the dataset. This element verifies the conclusion, that the quality of the sample, whether it is too large or if it contains noisy values, plays a key role for the tasks that have to be handled in DM. Moreover, the placement and FS can lead to a new model construction using the existing set of features. In many cases, feature construction is able to offer a better discrimination ability than a good FS.

There are various classification algorithms that, although respond well to a specific class of problems, they cannot be applied to tackle other similar problems. These kinds of algorithms can be applicable also to additional problems through data transformation (Galar *et al.*, 2011).

Another interesting point is the issue of the dataset shift. The dataset shift occurs when the testing data experience an incident that leads to (i) a change in the distribution of a single feature, (ii) a combination of features or (iii) the class boundaries. As a result, the common assumption that the training and testing data follow the same distributions is frequently abused in real-world applications (Quionero-Candela *et al.*, 2009). For more details about the dataset shift and the related work in this field, the reader is referred to Moreno-Torres *et al.* (2012).

Despite the progress that has been made by the research community in the last few years, there are still a number of issues to be resolved. For example, the issue of managing data over time in non-stationary environments (Quionero-Candela *et al.*, 2009). In such a problem, an adaptive model is necessary (Losing *et al.*, 2018). Furthermore, data preprocessing methods for handling data streams (Ramírez-Gallego *et al.*,

2017) will be well studied in the future, since they are able to tackle issues of Internet and technologies for massive online data collection.

Moreover, the question of what is the most suitable or appropriate series of data preprocessing steps to make a data mining algorithm more efficient? remains unanswered. A possible sequence of steps could be the steps that have been used in the division of the modules of the paper at hand. Nevertheless, in order this claim to be meaningful, extensive experiments and further study are required. We intend to address these critical issues and elaborate on the details in a future publication. Finally, in order an inexperienced reader to be familiar with this subject, we provide simple code of the most commonly used algorithms for preprocessing data in the Appendix.

Table 6 illustrates the applicability of the data preprocessing techniques presented in this paper with respect to the most well-known algorithms. The reader can distinguish with the mark '✓' the algorithms that require a corresponding preprocessing step. On the other hand, with the '✗' mark indicates the need for data preprocessing, for example in the case of the DTs and instance selection process. In addition, the mark '*' indicate in which method one of the above preprocessing steps could be applied, but in certain cases.

In conclusion, by taking into consideration the information that we have analyzed in each section, the reader is able to realize the importance of the data preparation processes in predictive DM. It is evident that the application of the preprocessing (or not), may affect the performance of a classifier. The order in which the steps should be executed is not obvious. Several studies, as it is noted in the paper at hand have been performed to determine the efficiency of such classification algorithms, under certain preprocessing data steps. Unfortunately, this is not feasible for all the existing problems. Nevertheless, we believe that through the citation analysis that we have provided for each preprocessing technique, we are in a position to obtain information for the impact of each method. For example, it cannot be taken as a coincidence that the SMOTE algorithm regarding the FS process, has highlighted and gathered most of the references in comparison to the other methods. Consequently, the reader by taking it into account is able to recognize which method exhibits a significant impact, and hence to select it for preprocessing.

Acknowledgment

The authors thank the anonymous reviewers for their constructive comments, which helped them to improve the manuscript. This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Life-long Learning» in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

Appendix

Simple code of basic preprocessing steps, that gives simple tutorial examples in R programming language related to the usage of some known preprocessing algorithms.

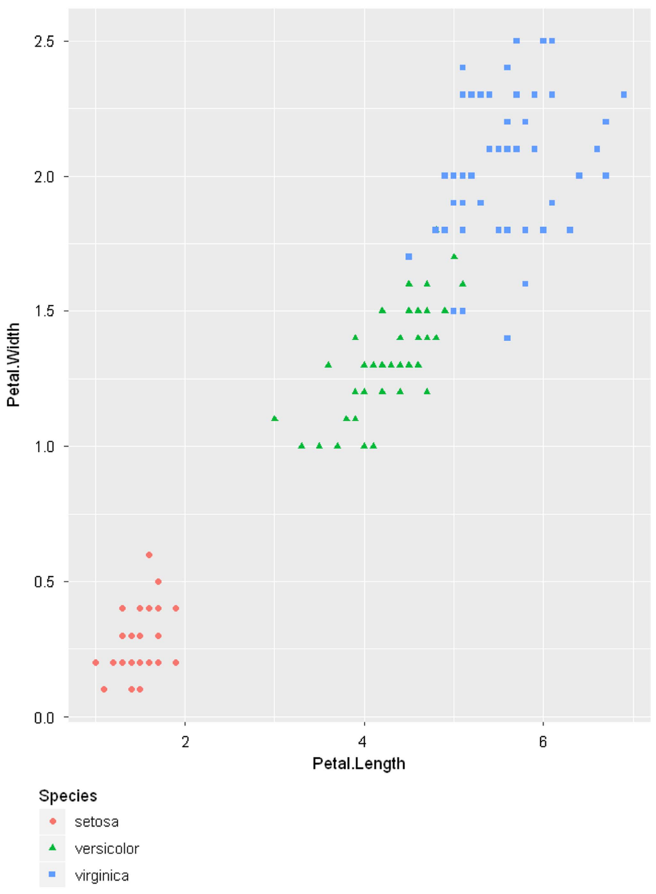
A Classification Dataset Example

```
In [1]: data(iris)
        head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

B Visualization Example

```
In [2]: library(ggplot2)
        ggplot(iris, aes(Petal.Length,Petal.Width)) +
        geom_point(aes(color =
        Species, shape = Species))
```



C Discretization Example

```
In [3]: #install.packages("discretization")
        library(discretization)
        cm=disc.Topdown(iris, method=3)# Ameva discretization
        cm$cutp #cutpoints
        head(cm$Disc.data) #show first lines of dataset
```

1. (a) 4.3 (b) 5.45 (c) 6.15 (d) 7.9

2. (a) 2 (b) 2.95 (c) 3.35 (d) 4.4

3. (a) 1 (b) 2.45 (c) 4.75 (d) 6.9

4. (a) 0.1 (b) 0.8 (c) 1.75 (d) 2.5

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	3	1	1	setosa
1	2	1	1	setosa
1	2	1	1	setosa
1	2	1	1	setosa
1	3	1	1	setosa
1	3	1	1	setosa

D Feature Selection Filtering Example

```
In [4]: library(mlr)
        #install.packages("FSelector")
        fv = generateFilterValuesData(iris.task, method = "information.gain")
        fv
```

```
Loading required package: ParamHelpers


FilterValues:
Task: iris-example
          name    type information.gain
1 Sepal.Length numeric        0.4521286
2  Sepal.Width numeric        0.2672750
3 Petal.Length numeric        0.9402853
4  Petal.Width numeric        0.9554360
```

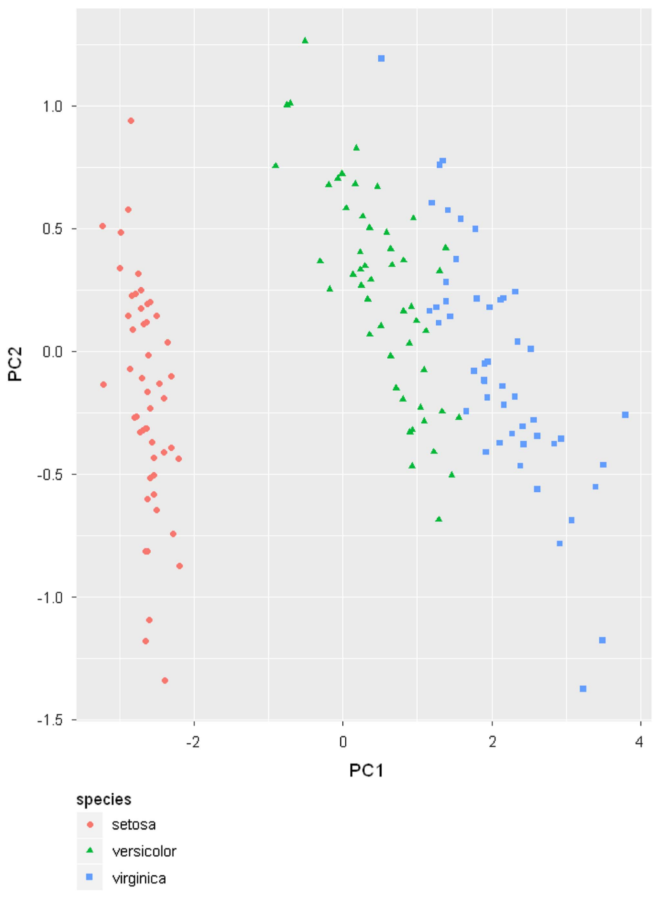
E Outlier and noise detection example

```
In [5]: #install.packages("NoiseFiltersR")
        library(NoiseFiltersR)
        out_For <- INFFC(Species~., iris)
        summary(out_For, explicit = TRUE)
```

```
Iteration 1: 4 noisy instances removed

Iteration 2: 0 noisy instances removed

Iteration 3: 0 noisy instances removed

Iteration 4: 0 noisy instances removed



Filter INFFC applied to dataset iris

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

Parameters:
consensus: FALSE
p: 0.01
s: 3
k: 5
threshold: 0

Results:
Number of removed instances: 4 (2.666667 %)
Number of repaired instances: 0 (0 %)

Explicit indexes for removed instances:
71 84 107 134
```

F Principal Component Analysis Example

```
In [6]: library(ggplot2)
        pca = prcomp(iris[,-5])
        ggplot(cbind(as.data.frame(pca$x), species = iris$Species), aes(PC1, PC2)) +
        geom_point(aes(colour = species, shape = species))
```



G Normalization example

```
In [7]: task = makeClassifTask(data = iris, target = "Species")
        task.norm = normalizeFeatures(task, method = "standardize") #Center and scale
        summary(getTaskData(task.norm))
```

```
  Sepal.Length        Sepal.Width        Petal.Length       Petal.Width
 Min.   :-1.86378   Min.   :-2.4258   Min.   :-1.5623   Min.   :-1.4422
 1st Qu.:-0.89767   1st Qu.:-0.5904   1st Qu.:-1.2225   1st Qu.:-1.1799
 Median :-0.05233   Median :-0.1315   Median : 0.3354   Median : 0.1321
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.67225   3rd Qu.: 0.5567   3rd Qu.: 0.7602   3rd Qu.: 0.7880

 Max.   : 2.48370   Max.   : 3.0805   Max.   : 1.7799   Max.   : 1.7064
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

H Regression Dataset Example

```
In [8]: data(airquality)
        summary(airquality)
```

```
     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   :37       NA's   :7
     Month            Day
 Min.   :5.000   Min.   : 1.0
 1st Qu.:6.000   1st Qu.: 8.0
 Median :7.000   Median :16.0
 Mean   :6.993   Mean   :15.8
 3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :9.000   Max.   :31.0
```

I Imputation Example

```
In [9]: imp = impute(airquality, classes = list(integer= imputeLearner("regr.cforest")))
        summary(imp$data)
```

```
     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 20.00   1st Qu.:120.0   1st Qu.: 7.400   1st Qu.:72.00
 Median : 32.26   Median :201.0   Median : 9.700   Median :79.00
 Mean   : 41.35   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 61.00   3rd Qu.:256.0   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
     Month            Day
 Min.   :5.000   Min.   : 1.0
 1st Qu.:6.000   1st Qu.: 8.0
 Median :7.000   Median :16.0
 Mean   :6.993   Mean   :15.8
 3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :9.000   Max.   :31.0
```

J Imbalanced Dataset and Instance Selection Example

```
In [10]: data.imbal.train = rbind(
           data.frame(x = rnorm(100, mean = 1), class = "A"),
           data.frame(x = rnorm(2000, mean = 2), class = "B")
         )
         task = makeClassifTask(data = data.imbal.train, target = "class")
         task.under = undersample(task, rate = 1/20)
         table(getTaskTargets(task.under))
```

```
  A   B
100 100
```

References

Aggarwal, C. C. 2013. An introduction to outlier analysis. In *Outlier Analysis.*, Springer, 1–40.

Angiulli, F. & Fassetti, F. 2014. Exploiting domain knowledge to detect outliers. *Data Mining and Knowledge Discovery* **28**(2), 519–568.

Angiulli, F. & Pizzuti, C. 2005. Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering* **17**(2), 203–215.

Aridas, C. K., Kotsiantis, S. B. & Vrahatis, M. N. 2016. Combining prototype selection with local boosting. In *Artificial Intelligence Applications and Innovations (AIAI) 2016. IFIP Advances in Information and Communication Technology*, Iliadis, L. & Maglogiannis, I. (eds). Springer, 475.

Aridas, C. K., Kotsiantis, S. B. & Vrahatis, M. N. 2017. Hybrid local boosting utilizing unlabeled data in classification tasks. *Evolving Systems* 1–11.

Arnaiz-González, Á., Dez-Pastor, J.-F., RodríGuez, J. J. & Garca-Osorio, C. 2016. Instance selection of linear complexity for big data. *Knowledge-Based Systems* **107**, 83–95.

Augasta, M. G. & Kathirvalavakumar, T. 2012. A new discretization algorithm based on range coefficient of dispersion and skewness for neural networks classifier. *Applied Soft Computing* **12**(2), 619–625.

Batista, G. E., Prati, R. C. & Monard, M. C. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter* **6**(1), 20–29.

Bezdek, J. C. & Kuncheva, L. I. 2001. Nearest prototype classifier designs: an experimental study. *International Journal of Intelligent Systems* **16**(12), 1445–1473.

Boulle, M. 2004. KHIOPS: a statistical discretization method of continuous attributes. *Machine Learning* **55**(1), 53–69.

Brodley, C. E. & Friedl, M. A. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research* **11**, 131–167.

Buzzi-Ferraris, G. & Manenti, F. 2011. Outlier detection in large data sets. *Computers & Chemical Engineering* **35**(2), 388–390.

Caises, Y., González, A., Leyva, E. & Pérez, R. 2011. Combining instance selection methods based on data characterization: an approach to increase their effectiveness. *Information Sciences* **181**(20), 4780–4798.

Cano, A., Nguyen, D. T., Ventura, S. & Cios, K. J. 2016. ur-CAIM: improved CAIM discretization for unbalanced and balanced data. *Soft Computing* **20**(1), 173–188.

Cano, J.-R., Garca, S. & Herrera, F. 2008. Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes. *Pattern Recognition Letters* **29**(16), 2156–2164.

Cano, J. R., Herrera, F. & Lozano, M. 2005. Strategies for scaling up evolutionary instance reduction algorithms for data mining. In *Evolutionary Computation in Data Mining*, Springer, 21–39.

Caruana, R. & de Sa, V. R. 2003. Benefitting from the variables that variable selection discards. *Journal of Machine Learning Research* **3**(3), 1245–1264.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**, 321–357.

Chen, S., Wang, W. & van Zuylen, H. 2010a. A comparison of outlier detection algorithms for ITS data. *Expert Systems with Applications* **37**(2), 1169–1178.

Chen, Y., Miao, D. & Zhang, H. 2010b. Neighborhood outlier detection. *Expert Systems with Applications* **37**(12), 8745–8749.

Chow, T. W. & Huang, D. 2005. Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. *IEEE Transactions on Neural Networks* **16**(1), 213–224.

Cismondi, F., Fialho, A. S., Vieira, S. M., Reti, S. R., Sousa, J. M. & Finkelstein, S. N. 2013. Missing data in medical databases: impute, delete or classify? *Artificial Intelligence in Medicine* **58**(1), 63–72.

Crone, S. F., Lessmann, S. & Stahlbock, R. 2006. The impact of preprocessing on data mining: an evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research* **173**(3), 781–800.

Czarnowski, I. 2010. Prototype selection algorithms for distributed learning. *Pattern Recognition* **43**(6), 2292–2300.

Czarnowski, I. 2012. Cluster-based instance selection for machine classification. *Knowledge and Information Systems* **30**(1), 113–133.

de Haro-García, A. & García-Pedrajas, N. 2009. A divide-and-conquer recursive approach for scaling up instance selection algorithms. *Data Mining and Knowledge Discovery* **18**(3), 392–418.

de Sá, C. R., Soares, C. & Knobbe, A. 2016. Entropy-based discretization methods for ranking data. *Information Sciences* **329**, 921–936.

Delany, S. J., Segata, N. & Mac Namee, B. 2012. Profiling instances in noise reduction. *Knowledge-Based Systems* **31**, 28–40.

Derrac, J., Garca, S. & Herrera, F. 2010a. IFS-CoCo: instance and feature selection based on cooperative coevolution with nearest neighbor rule. *Pattern Recognition* **43**(6), 2082–2105.

Derrac, J., Garca, S. & Herrera, F. 2010b. A survey on evolutionary instance selection and generation. *International Journal of Applied Metaheuristic Computing* **1**, 60–92.

Dougherty, J., Kohavi, R. & Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, 194–202. Elsevier.

Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L. O., Goldgof, D. B. & Kasturi, R. 2016. Active cleaning of label noise. *Pattern Recognition* **51**, 463–480.

Elomaa, T. & Rousu, J. 2004. Efficient multisplitting revisited: optima-preserving elimination of partition candidates. *Data Mining and Knowledge Discovery* **8**(2), 97–126.

Escalante, H. J. 2005. A comparison of outlier detection algorithms for machine learning. In *Proceedings of the International Conference on Communications in Computing*, 228–237.

Estabrooks, A., Jo, T. & Japkowicz, N. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence* **20**(1), 18–36.

Farhangfar, A., Kurgan, L. & Dy, J. 2008. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition* **41**(12), 3692–3705.

Farhangfar, A., Kurgan, L. & Pedrycz, W. 2007. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* **37**(5), 692–709.

Farquad, M. & Bose, I. 2012. Preprocessing unbalanced data using support vector machine. *Decision Support Systems* **53**(1), 226–233.

Fernández, A., Carmona, C. J., del Jesus, M. & Herrera, F. 2017. A pareto based ensemble with feature and instance selection for learning from multi-class imbalanced datasets. *International Journal of Neural Systems* **27**, 1–17.

Filzmoser, P., Maronna, R. & Werner, M. 2008. Outlier identification in high dimensions. *Computational Statistics & Data Analysis* **52**(3), 1694–1711.

Flores, M. J., Gámez, J. A., Martnez, A. M. & Puerta, J. M. 2011. Handling numeric attributes when comparing bayesian network classifiers: does the discretization method matter? *Applied Intelligence* **34**(3), 372–385.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H. & Herrera, F. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* **44**(8), 1761–1776.

Garcia, L. P., de Carvalho, A. C. & Lorena, A. C. 2016a. Noise detection in the meta-learning level. *Neurocomputing* **176**, 14–25.

Garcia, L. P., Lorena, A. C., Matwin, S. & de Carvalho, A. C. 2016b. Ensembles of label noise filters: a ranking approach. *Data Mining and Knowledge Discovery* **30**(5), 1192–1216.

García, S., Cano, J. R. & Herrera, F. 2008. A memetic algorithm for evolutionary prototype selection: a scaling up approach. *Pattern Recognition* **41**(8), 2693–2709.

García, S., Derrac, J., Cano, J. & Herrera, F. 2012a. Prototype selection for nearest neighbor classification: taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34**(3), 417–435.

García, S., Luengo, J. & Herrera, F. 2015. *Data Preprocessing in Data Mining*. Springer.

García, S., Luengo, J., Sáez, J. A., Lopez, V. & Herrera, F. 2013. A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering* **25**(4), 734–750.

García, V., Sánchez, J. S. & Mollineda, R. A. 2012b. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems* **25**(1), 13–21.

García-Pedrajas, N., Del Castillo, J. A. R. & Ortiz-Boyer, D. 2010. A cooperative coevolutionary algorithm for instance selection for instance-based learning. *Machine Learning* **78**(3), 381–420.

GarcíA-Pedrajas, N. & PéRez-RodríGuez, J. 2012. Multi-selection of instances: a straightforward way to improve evolutionary instance selection. *Applied Soft Computing* **12**(11), 3590–3602.

Ghoting, A., Parthasarathy, S. & Otey, M. E. 2008. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery* **16**(3), 349–364.

Gonzalez-Abril, L., Cuberos, F. J., Velasco, F. & Ortega, J. A. 2009. AMEVA: an autonomous discretization algorithm. *Expert Systems with Applications* **36**(3), 5327–5332.

Gupta, A., Mehrotra, K. G. & Mohan, C. 2010. A clustering-based discretization for supervised learning. *Statistics & Probability Letters* **80**(9), 816–824.

Guyon, I. & Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* **3**(3), 1157–1182.

He, H. & Garcia, E. A. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* **21**(9), 1263–1284.

Hernandez-Leal, P., Carrasco-Ochoa, J. A., Martnez-Trinidad, J. F. & Olvera-Lopez, J. A. 2013. Instancerank based on borders for instance selection. *Pattern Recognition* **46**(1), 365–375.

Hinton, G. E. & Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* **313** (5786), 504–507.

Hoffmann, H. 2007. Kernel PCA for novelty detection. *Pattern Recognition* **40**(3), 863–874.

Honghai, F., Guoshun, C., Cheng, Y., Bingru, Y. & Yumei, C. 2005. A SVM regression based approach to filling in missing values. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 581–587. Springer.

Hu, Q., Che, X., Zhang, L. & Yu, D. 2010. Feature evaluation and selection based on neighborhood soft margin. *Neurocomputing* **73**(10), 2114–2124.

Hua, J., Tembe, W. D. & Dougherty, E. R. 2009. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition* **42**(3), 409–424.

Hua, J., Xiong, Z., Lowey, J., Suh, E. & Dougherty, E. R. 2005. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics* **21**(8), 1509–1515.

Huang, H., Lin, J., Chen, C. & Fan, M. 2006. Review of outlier detection. *Application Research of Computers* **8**, 2006–2008.

Janssens, D., Brijs, T., Vanhoof, K. & Wets, G. 2006. Evaluating the performance of cost-based discretization versus entropy-and error-based discretization. *Computers & Operations Research* **33**(11), 3107–3123.

Jerez, J. M., Molina, I., Garca-Laencina, P. J., Alba, E., Ribelles, N., Martn, M. & Franco, L. 2010. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine* **50**(2), 105–115.

Jin, R., Breitbart, Y. & Muoh, C. 2009. Data discretization unification. *Knowledge and Information Systems* **19**(1), 1–29.

Kennedy, J. & Eberhart, R. C. 1995. Particle swarm optimization. In *IEEE International Conference on Neural Networks Proceedings 1995*, **4**, 1942–1948. IEEE.

Kim, S., Cho, N. W., Kang, B. & Kang, S.-H. 2011. Fast outlier detection for very large log data. *Expert Systems with Applications* **38**(8), 9587–9596.

Kim, S.-W. & Oommen, B. J. 2003. A brief taxonomy and ranking of creative prototype reduction schemes. *Pattern Analysis & Applications* **6**(3), 232–244.

Klinkenberg, R. 2004. Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis* **8**(3), 281–300.

Kumar, A. & Zhang, D. 2007. Hand-geometry recognition using entropy-based discretization. *IEEE Transactions on Information Forensics and Security* **2**(2), 181–187.

Kurgan, L. A. & Cios, K. J. 2004. CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering* **16**(2), 145–153.

Lemaître, G., Nogueira, F. & Aridas, C. K. 2017. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* **18**(17), 1–5.

Li, M., Deng, S., Feng, S. & Fan, J. 2011. An effective discretization based on class-attribute coherence maximization. *Pattern Recognition Letters* **32**(15), 1962–1973.

Lin, W.-C., Tsai, C.-F., Ke, S.-W., Hung, C.-W. & Eberle, W. 2015. Learning to detect representative data for large scale instance selection. *Journal of Systems and Software* **106**, 1–8.

Liu, C., Wang, W., Wang, M., Lv, F. & Konan, M. 2017. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowledge-Based Systems* **116**, 58–73.

Liu, F. T., Ting, K. M. & Zhou, Z.-H. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data* **6**(1), 1–39.

Liu, H., Hussain, F., Tan, C. L. & Dash, M. 2002. Discretization: an enabling technique. *Data Mining and Knowledge Discovery* **6**(4), 393–423.

Liu, H. & Motoda, H. 2002. On issues of instance selection. *Data Mining and Knowledge Discovery* **6**(2), 115–130.

Liu, H. & Motoda, H. 2007. *Computational Methods of Feature Selection*. CRC Press.

Liu, X. & Wang, H. 2005. A discretization algorithm based on a heterogeneity criterion. *IEEE Transactions on Knowledge and Data Engineering* **17**(9), 1166–1173.

Liu, Z.-G., Pan, Q., Dezert, J. & Martin, A. 2016. Adaptive imputation of missing values for incomplete pattern classification. *Pattern Recognition* **52**, 85–95.

Lobato, F., Sales, C., Araujo, I., Tadaiesky, V., Dias, L., Ramos, L. & Santana, A. 2015. Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters* **68**, 126–131.

López, V., Fernández, A., Moreno-Torres, J. G. & Herrera, F. 2012. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications* **39**(7), 6585–6608.

Losing, V., Hammer, B. & Wersing, H. 2018. Incremental on-line learning: a review and comparison of state of the art algorithms. *Neurocomputing* **275**, 1261–1274.

Luengo, J., Garca, S. & Herrera, F. 2012. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and Information Systems* **32**(1), 77–108.

Mahanipour, A., Nezamabadi-pour, H. & Nikpour, B. 2018. Using fuzzy-rough set feature selection for feature construction based on genetic programming. In *2018 3rd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, 1–6. IEEE.

Maldonado, S. & Weber, R. 2009. A wrapper method for feature selection using support vector machines. *Information Sciences* **179**(13), 2208–2217.

Mao, K. 2004. Orthogonal forward selection and backward elimination algorithms for feature subset selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **34**(1), 629–634.

Marchiori, E. 2008. Hit miss networks with applications to instance selection. *Journal of Machine Learning Research* **9**(6), 997–1017.

Mateos-García, D., García-Gutiérrez, J. & Riquelme-Santos, J. C. 2012. On the evolutionary optimization of k-NN by label-dependent feature weighting. *Pattern Recognition Letters* **33**(16), 2232–2238.

Mizianty, M. J., Kurgan, L. A. & Ogiela, M. R. 2010. Discretization as the enabling technique for the naive Bayes and semi-naive Bayes-based classification. *The Knowledge Engineering Review* **25**(04), 421–449.

Moreno-Torres, J. G., Raeder, T., Alaiz-RodríGuez, R., Chawla, N. V. & Herrera, F. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* **45**(1), 521–530.

Nanni, L. & Lumini, A. 2011. Prototype reduction techniques: a comparison among different approaches. *Expert Systems with Applications* **38**(9), 11820–11828.

Nikolaidis, K., Goulermas, J. Y. & Wu, Q. 2011. A class boundary preserving algorithm for data condensation. *Pattern Recognition* **44**(3), 704–715.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martnez-Trinidad, J. F. & Kittler, J. 2010. A review of instance selection methods. *Artificial Intelligence Review* **34**(2), 133–143.

Panday, D., de Amorim, R. C. & Lane, P. 2018. Feature weighting as a tool for unsupervised feature selection. *Information Processing Letters* **129**, 44–52.

Park, D.-C. 2009. Centroid neural network with weighted features. *Journal of Circuits, Systems, and Computers* **18** (08), 1353–1367.

Parsopoulos, K. E. & Vrahatis, M. N. 2010. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global).

Pearson, R. K. 2005. *Mining Imperfect Data: Dealing with Contamination and Incomplete Records*. SIAM.

Piramuthu, S. 2004. Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research* **156**(2), 483–494.

Piramuthu, S. & Sikora, R. T. 2009. Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications* **36**(2), 3401–3406.

Pkekalska, E., Duin, R. P. & Paclk, P. 2006. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition* **39**(2), 189–208.

Pyle, D. 1999. *Data Preparation for Data Mining*, 1. Morgan Kaufmann.

Qin, Y., Zhang, S., Zhu, X., Zhang, J. & Zhang, C. 2009. POP algorithm: kernel-based imputation to treat missing values in knowledge discovery from databases. *Expert Systems with Applications* **36**(2), 2794–2804.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A. & Lawrence, N. D. 2009. *Dataset Shift in Machine Learning*. MIT Press.

Ramírez-Gallego, S., Garca, S., Mouriño-Taln, H., Martnez-Rego, D., Bolón-Canedo, V., Alonso-Betanzos, A., Bentez, J. M. & Herrera, F. 2016. Data discretization: taxonomy and big data challenge. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **6**(1), 5–21.

Ramírez-Gallego, S., Krawczyk, B., Garca, S., Woźniak, M. & Herrera, F. 2017. A survey on data preprocessing for data stream mining: current status and future directions. *Neurocomputing* **239**, 39–56.

Reinartz, T. 2002. A unifying view on instance selection. *Data Mining and Knowledge Discovery* **6**(2), 191–210.

Sáez, J. A., Galar, M., Luengo, J. & Herrera, F. 2016. INFFC: an iterative class noise filter based on the fusion of classifiers with noise sensitivity control. *Information Fusion* **27**, 19–32.

Sáez, J. A., Luengo, J. & Herrera, F. 2013. Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* **46**(1), 355–364.

Segata, N., Blanzieri, E., Delany, S. J. & Cunningham, P. 2010. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems* **35**(2), 301–331.

Shen, C., Wang, X. & Yu, D. 2012. Feature weighting of support vector machines based on derivative saliency analysis and its application to financial data mining. *International Journal of Advancements in Computing Technology* **4**(1), 199–206.

Shu, W. & Shen, H. 2016. Multi-criteria feature selection on cost-sensitive data with missing values. *Pattern Recognition* **51**, 268–280.

Silva-Ramírez, E.-L., Pino-Mejas, R., López-Coello, M. & Cubiles-de-la Vega, M.-D. 2011. Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks* **24**(1), 121–129.

Sim, J., Kwon, O. & Lee, K. C. 2016. Adaptive pairing of classifier and imputation methods based on the characteristics of missing values in data sets. *Expert Systems with Applications* **46**, 485–493.

Skillicorn, D. B. & McConnell, S. M. 2008. Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed Computing* **68**(1), 16–36.

Smith, M. G. & Bull, L. 2005. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines* **6**(3), 265–281.

Somol, P. & Pudil, P. 2002. Feature selection toolbox. *Pattern Recognition* **35**(12), 2749–2759.

Sun, Y., Wong, A. K. & Kamel, M. S. 2009. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence* **23**(04), 687–719.

Triguero, I., Derrac, J., Garcia, S. & Herrera, F. 2012a. Integrating a differential evolution feature weighting scheme into prototype generation. *Neurocomputing* **97**, 332–343.

Triguero, I., Derrac, J., Garcia, S. & Herrera, F. 2012b. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42**(1), 86–100.

Tsai, C.-F. & Chang, F.-Y. 2016. Combining instance selection for better missing value imputation. *Journal of Systems and Software* **122**, 63–71.

Tsai, C.-J., Lee, C.-I. & Yang, W.-P. 2008. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences* **178**(3), 714–731.

Unler, A., Murat, A. & Chinnam, R. B. 2011. mr 2 PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. *Information Sciences* **181**(20), 4625–4641.

van Hulse, J. & Khoshgoftaar, T. 2009. Knowledge discovery from imbalanced and noisy data. *Data & Knowledge Engineering* **68**(12), 1513–1542.

van Hulse, J. & Khoshgoftaar, T. M. 2006. Class noise detection using frequent itemsets. *Intelligent Data Analysis* **10** (6), 487–507.

van Hulse, J. D., Khoshgoftaar, T. M. & Huang, H. 2007. The pairwise attribute noise detection algorithm. *Knowledge and Information Systems* **11**(2), 171–190.

Virgolin, M., Alderliesten, T., Bel, A., Witteveen, C. & Bosman, P. A. 2018. Symbolic regression and feature construction with gp-gomea applied to radiotherapy dose reconstruction of childhood cancer survivors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 1395–1402. ACM.

Wang, B. & Japkowicz, N. 2004. Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, 19.

Wettschereck, D., Aha, D. W. & Mohri, T. 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* **11**(1–5), 273–314.

Wilson, D. R. & Martinez, T. R. 2000. Reduction techniques for instance-based learning algorithms. *Machine Learning* **38**(3), 257–286.

Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Wong, T.-T. 2012. A hybrid discretization method for nave bayesian classifiers. *Pattern Recognition* **45**(6), 2321–2325.

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

Yang, Y., Webb, G. I. & Wu, X. 2009. Discretization methods. In *Data Mining and Knowledge Discovery Handbook*. Springer, 101–116.

Zhang, S. 2011. Shell-neighbor method and its application in missing data imputation. *Applied Intelligence* **35**(1), 123–133.

Zhang, S., Jin, Z. & Zhu, X. 2011. Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software* **84**(3), 452–459.

Zhang, S., Zhang, J., Zhu, X., Qin, Y. & Zhang, C. 2008. Missing value imputation based on data clustering. In *Transactions on Computational Science I*, Springer, 128–138.

Zhou, M.-J. & Chen, X.-J. 2012. An outlier mining algorithm based on dissimilarity. *Procedia Environmental Sciences* **12**, 810–814.

Zhou, Y., Chen, Y., Feng, L., Zhang, X., Shen, Z. & Zhou, X. 2018. Supervised and adaptive feature weighting for object-based classification on satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **11**(9), 3224–3234.

Zhu, X. & Wu, X. 2004. Class noise vs. attribute noise: a quantitative study. *Artificial Intelligence Review* **22**(3), 177–210.