García *et al. Big Data Analytics* (2016) 1:9
DOI 10.1186/s41044-016-0014-0

Big Data Analytics

REVIEW Open Access

Big data preprocessing: methods and prospects

Salvador García*, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez and Francisco Herrera

*Correspondence:
salvagl@decsai.ugr.es
Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, 18071 Granada, Spain

Abstract

The massive growth in the scale of data has been observed in recent years being a key factor of the Big Data scenario. Big Data can be defined as high volume, velocity and variety of data that require a new high-performance processing. Addressing big data is a challenging and time-demanding task that requires a large computational infrastructure to ensure successful data processing and analysis. The presence of data preprocessing methods for data mining in big data is reviewed in this paper. The definition, characteristics, and categorization of data preprocessing approaches in big data are introduced. The connection between big data and data preprocessing throughout all families of methods and big data technologies are also examined, including a review of the state-of-the-art. In addition, research challenges are discussed, with focus on developments on different big data framework, such as Hadoop, Spark and Flink and the encouragement in devoting substantial research efforts in some families of data preprocessing methods and applications on new big data learning paradigms.

Keywords: Big data, Data mining, Data preprocessing, Hadoop, Spark, Imperfect data, Data transformation, Feature selection, Instance reduction

Background

Vast amounts of raw data is surrounding us in our world, data that cannot be directly treated by humans or manual applications. Technologies as the World Wide Web, engineering and science applications and networks, business services and many more generate data in exponential growth thanks to the development of powerful storage and connection tools. Organized knowledge and information cannot be easily obtained due to this huge data growth and neither it can be easily understood or automatically extracted. These premises have led to the development of data science or data mining [1], a well-known discipline which is more and more present in the current world of the Information Age.

Nowadays, the current volume of data managed by our systems have surpassed the processing capacity of traditional systems [2], and this applies to data mining as well. The arising of new technologies and services (like Cloud computing) as well as the reduction in hardware price are leading to an ever-growing rate of information on the Internet. This phenomenon certainly represents a "Big" challenge for the data analytics community. Big Data can be thus defined as very high volume, velocity and variety of data that require a new high-performance processing [3].

BioMed Central

© 2016 The Author(s). **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.

Distributed computing has been widely used by data scientists before the advent of Big Data phenomenon. Many standard and time-consuming algorithms were replaced by their distributed versions with the aim of agilizing the learning process. However, for most of current massive problems, a distributed approach becomes mandatory nowadays since no batch architecture is able to tackle these huge problems.

Many platforms for large-scale processing have tried to face the problematic of Big Data in last years [4]. These platforms try to bring closer the distributed technologies to the standard user (enginners and data scientists) by hiding the technical nuances derived from distributed environments. Complex designs are required to create and maintain these platforms, which generalizes the use of distributed computing. On the other hand, Big Data platforms also requires additional algorithms that give support to relevant tasks, like big data preprocessing and analytics. Standard algorithms for those tasks must be also re-designed (sometimes, entirely) if we want to learn from large-scale datasets. It is not trivial thing and presents a big challenge for researchers.

The first framework that enabled the processing of large-scale datasets was MapReduce [5] (in 2003). This revolutionary tool was intended to process and generate huge datasets in an automatic and distributed way. By implementing two primitives, Map and Reduce, the user is able to use a scalable and distributed tool without worrying about technical nuances, such as: failure recovery, data partitioning or job communication. Apache Hadoop [6, 7] emerged as the most popular open-source implementation of MapReduce, maintaining the aforementioned features. In spite of its great popularity, MapReduce (and Hadoop) is not designed to scale well when dealing with iterative and online processes, typical in machine learning and stream analytics [8].

Apache Spark [9, 10] was designed as an alternative to Hadoop, capable of performing faster distributed computing by using in-memory primitives. Thanks to its ability of loading data into memory and re-using it repeatedly, this tool overcomes the problem of iterative and online processing presented by MapReduce. Additionally, Spark is a general-purpose framework that thanks to its generality allows to implement several distributed programming models on top of it (like Pregel or HaLoop) [11]. Spark is built on top of a new abstraction model called Resilient Distributed Datasets (RDDs). This versatile model allows controlling the persistence and managing the partitioning of data, among other features.

Some competitors to Apache Spark have emerged lastly, especially from the streaming side [12]. Apache Storm [13] is an open-source distributed real-time processing platform, which is capable of processing millions of tuples per second and node in a fault-tolerant way. Apache Flink [14] is a recent top-level Apache project designed for distributed stream and batch data processing. Both alternatives try to fill the "online" gap left by Spark, which employs a mini-batch streaming processing instead of a pure streaming approach.

The performance and quality of the knowledge extracted by a data mining method in any framework does not only depends on the design and performance of the method but is also very dependent on the quality and suitability of such data. Unfortunately, negative factors as noise, missing values, inconsistent and superfluous data and huge sizes in examples and features highly influence the data used to learn and extract knowledge. It is well-known that low quality data will lead to low quality knowledge [15]. Thus data preprocessing [16] is a major and essential stage whose main goal is to obtain

final data sets which can be considered correct and useful for further data mining algorithms.

Big Data also suffer of the aforementioned negative factors. Big Data preprocessing constitutes a challenging task, as the previous existent approaches cannot be directly applied as the size of the data sets or data streams make them unfeasible. In this overview we gather the most recent proposals in data preprocessing for Big Data, providing a snapshot of the current state-of-the-art. Besides, we discuss the main challenges on developments in data preprocessing for big data frameworks, as well as technologies and new learning paradigms where they could be successfully applied.

Data preprocessing

The set of techniques used prior to the application of a data mining method is named as data preprocessing for data mining [16] and it is known to be one of the most meaningful issues within the famous Knowledge Discovery from Data process [17, 18] as shown in Fig. 1. Since data will likely be imperfect, containing inconsistencies and redundancies is not directly applicable for a starting a data mining process. We must also mention the fast growing of data generation rates and their size in business, industrial, academic and science applications. The bigger amounts of data collected require more sophisticated mechanisms to analyze it. Data preprocessing is able to adapt the data to the requirements posed by each data mining algorithm, enabling to process data that would be unfeasible otherwise.

Albeit data preprocessing is a powerful tool that can enable the user to treat and process complex data, it may consume large amounts of processing time [15]. It includes a wide range of disciplines, as data preparation and data reduction techniques as can be seen in Fig. 2. The former includes data transformation, integration, cleaning and normalization; while the latter aims to reduce the complexity of the data by feature selection, instance

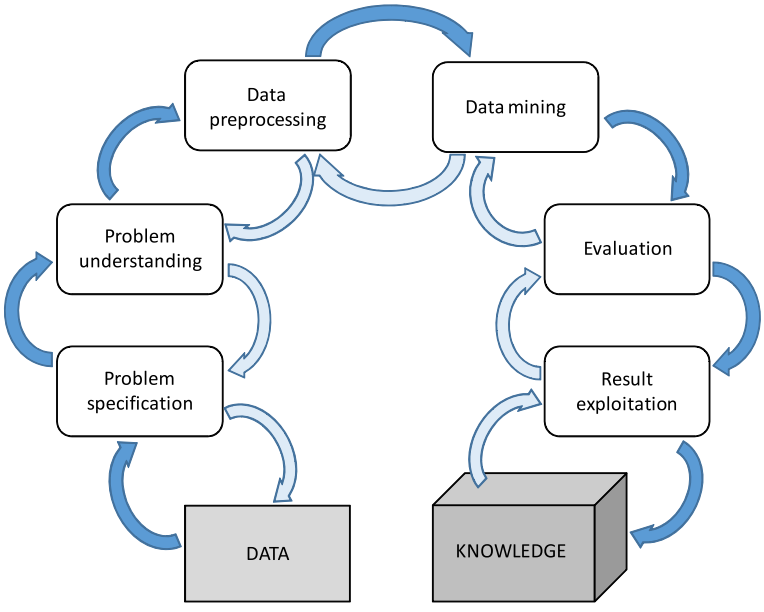

Fig. 1 KDD process

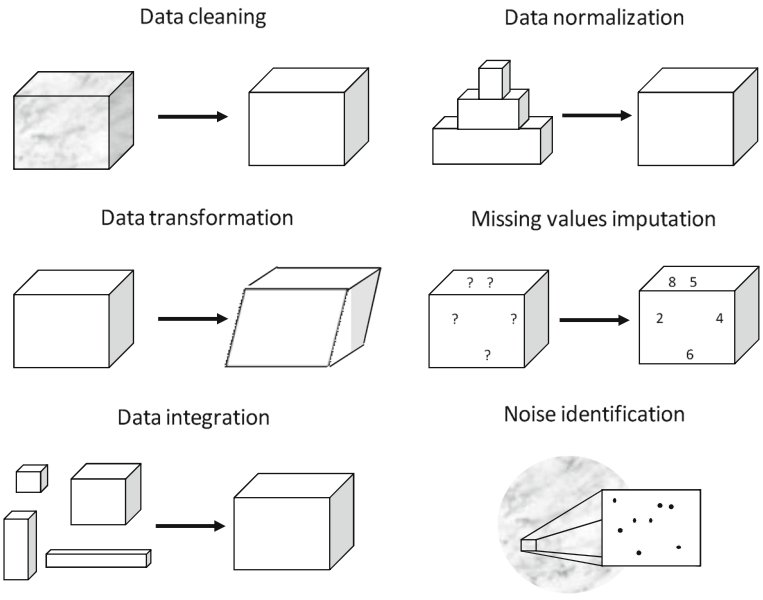

Fig. 2 Data preprocessing tasks

selection or by discretization (see Fig. 3). After the application of a successful data preprocessing stage, the final data set obtained can be regarded as a reliable and suitable source for any data mining algorithm applied afterwards.

Data preprocessing is not only limited to classical data mining tasks, as classification or regression. More and more researchers in novel data mining fields are paying increasingly attention to data data preprocessing as a tool to improve their models. This wider adoption of data preprocessing techniques is resulting in adaptations of known models for related frameworks, or completely novel proposals.

In the following we will present the main fields of data preprocessing, grouping them by their types and showing the current open challenges relative to each one. First, we will tackle the preprocessing techniques to deal with imperfect data, where missing values and noise data are included. Next, data reduction preprocessing approaches will be presented, in which feature selection and space transformation are shown. The following section will deal with instance reduction algorithms, including instance selection and prototype generation. The last three section will be devoted to discretization, resampling for imbalanced problems and data preprocessing in new fields of data mining respectively.

Imperfect data

Most techniques in data mining rely on a data set that is supposedly complete or noise-free. However, real-world data is far from being clean or complete. In data preprocessing it is common to employ techniques to either removing the noisy data or to impute (fill in) the missing data. The following two sections are devoted two missing values imputation and noise filtering.

Missing values imputation

One big assumption made by data mining techniques is that the data set is complete. The presence of missing values is, however, very common in the acquisition processes. A

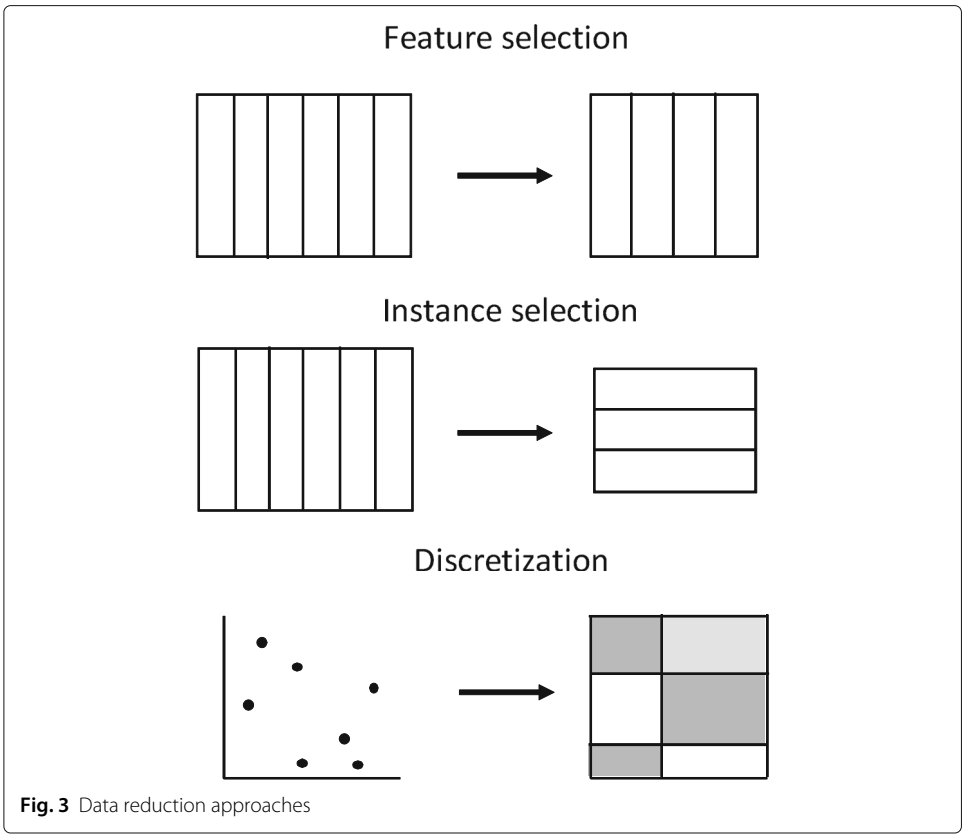

Fig. 3 Data reduction approaches

missing value is a datum that has not been stored or gathered due to a faulty sampling process, cost restrictions or limitations in the acquisition process. Missing values cannot be avoided in data analysis, and they tend to create severe difficulties for practitioners.

Missing values treatment is difficult. Inappropriately handling the missing values will easily lead to poor knowledge extracted and also wrong conclusions [19]. Missing values have been reported to cause loss of efficiency in the knowledge extraction process, strong biases if the missingness introduction mechanism is mishandled and severe complications in data handling.

Many approaches are available to tackle the problematic imposed by the missing values in data preprocessing [20]. The first option is usually to discard those instances that may contain a missing value. However, this approach is rarely beneficial, as eliminating instances may produce a bias in the learning process, and important information can be discarded [21]. The seminal works on data imputation come from statistics. They model the probability functions of the data and take into account the mechanisms that induce missingness. By using maximum likelihood procedures, they sample the approximate probabilistic models to fill the missing values. Since the true probability model for a particular data sets is usually unknown, the usage of machine learning techniques has become very popular nowadays as they can be applied avoiding without providing any prior information.

Noise treatment

Data mining algorithms tend to assume that any data set is a sample of an underlying distribution with no disturbances. As we have seen in the previous section, data gathering

is rarely perfect, and corruptions often appear. Since the quality of the results obtained by a data mining technique is dependent on the quality of the data, tackling the problem of noise data is mandatory [22]. In supervised problems, noise can affect the input features, the output values or both. When noise is present in the input attributes, it is usually referred as *attribute noise*. The worse case is when the noise affects the output attribute, as this means that the bias introduced will be greater. As this kind of noise has been deeply studied in classification, it is usually known as *class noise*.

In order to treat noise in data mining, two main approaches are commonly used in the data preprocessing literature. The first one is to correct the noise by using *data polishing methods*, specially if it affects the labeling of an instance. Even partial noise correction is claimed to be beneficial [23], but it is a difficult task and usually limited to small amounts of noise. The second is to use *noise filters*, which identify and remove the noisy instances in the training data and do no require the data mining technique to be modified.

Dimensionality reduction

When data sets become large in the number of predictor variables or the number of instances, data mining algorithms face the *curse of dimensionality* problem [24]. It is a serious problem as it will impede the operation of most data mining algorithms as the computational cost rise. This section will underline the most influential dimensionality reduction algorithms according to the division established into Feature Selection (FS) and space transformation based methods.

Feature selection

Feature selection (FS) is "the process of identifying and removing as much irrelevant and redundant information as possible" [25]. The goal is to obtain a subset of features from the original problem that still appropriately describe it. This subset is commonly used to train a learner, with added benefits reported in the specialized literature [26, 27]. FS can remove irrelevant and redundant features which may induce accidental correlations in learning algorithms, diminishing their generalization abilities. The use of FS is also known to decrease the risk of over-fitting in the algorithms used later. FS will also reduce the search space determined by the features, thus making the learning process faster and also less memory consuming.

The use FS can also help in task not directly related to the data mining algorithm applied to the data. FS can be used in the data collection stage, saving cost in time, sampling, sensing and personnel used to gather the data. Models and visualizations made from data with fewer features will be easier to understand and to interpret.

Space transformations

FS is not the only way to cope with the curse of dimensionality by reducing the number of dimensions. Instead of selecting the most promising features, space transformation techniques generate a whole new set of features by combining the original ones. Such a combination can be made obeying different criteria. The first approaches were based on linear methods, as factor analysis [28] and PCA [29].

More recent techniques try to exploit nonlinear relations among the variables. Some of the most important, both in relevance and usage, space transformation procedures are LLE [30], ISOMAP [31] and derivatives. They focus on transforming the original

set of variables into a smaller number of projections, sometimes taking into account the geometrical properties of clusters of instances or patches of the underlying manifolds.

Instance reduction

A popular approach to minimize the impact of very large data sets in data mining algorithms is the use of Instance Reduction (IR) techniques. They reduce the size of the data set without decreasing the quality of the knowledge that can be extracted from it. Instance reduction is a complementary task regarding FS. It reduces the quantity of data by removing instances or by generating new ones. In the following we describe the most important instance reduction and generation algorithms.

Instance selection

Nowadays, instance selection is perceived as necessary [32]. The main problem in instance selection is to identify suitable examples from a very large amount of instances and then prepare them as input for a data mining algorithm. Thus, instance selection is comprised by a series of techniques that must be able to choose a subset of data that can replace the original data set and also being able to fulfill the goal of a data mining application [33, 34]. It must be distinguished between instance selection, which implies a smart operation of instance categorization, from data sampling, which constitutes a more randomized approach [16].

A successful application of instance selection will produce a minimum data subset that it is independent from the data mining algorithm used afterwards, without losing performance. Other added benefits of instance selection is to remove noisy and redundant instances (*cleaning*), to allow data mining algorithms to operate with large data sets (*enabling*) and to focus on the important part of the data (*focusing*).

Instance generation

Instance selection methods concern the identification of an optimal subset of representative objects from the original training data by discarding noisy and redundant examples. Instance generation methods, by contrast, besides selecting data, can generate and replace the original data with new artificial data. This process allows it to fill regions in the domain of the problem, which have no representative examples in original data, or to condensate large amounts of instances in less examples. Instance generation methods are often called prototype generation methods, as the artificial examples created tend to act as a representative of a region or a subset of the original instances [35].

The new prototypes may be generated following diverse criteria. The simplest approach is to relabel some examples, for example those that are suspicious of belonging to a wrong class label. Some prototype generation methods create centroids by merging similar examples, or by first merging the feature space in several regions and then creating a set of prototype for each one. Others adjust the position of the prototypes through the space, by adding or substracting values to the prototype's features.

Discretization

Data mining algorithms require to know the domain and type of the data that will be used as input. The type of such data may vary, from categorical where no order among the values can be established, to numerical data where the order among the values there exist.

Decision trees, for instance, make split based on information or separability measures that require categorical values in most cases. If continuous data is present, the discretization of the numerical features is mandatory, either prior to the tree induction or during its building process.

Discretization is gaining more and more consideration in the scientific community [36] and it is one of the most used data preprocessing techniques. It transforms quantitative data into qualitative data by dividing the numerical features into a limited number of non-overlapped intervals. Using the boundaries generated, each numerical value is mapped to each interval, thus becoming discrete. Any data mining algorithm that needs nominal data can benefit from discretization methods, since many real-world applications usually produce real valued outputs. For example, three of the ten methods considered as the top ten in data mining [37] need an external or embedded discretization of data: C4.5 [38], Apriori [39] and Naïve Bayes [40] In these cases, discretization is a crucial previous stage.

Discretization also produce added benefits. The first is data simplification and reduction, helping to produce a faster and more accurate learning. The second is readability, as discrete attributes are usually easier to understand, use and explain [36]. Nevertheless these benefits come at price: any discretization process is expected to generate a loss of information. Minimizing this information loss is the main goal pursued by the discretizer, but an optimal discretization is a NP-complete process. Thus, a wide range of alternatives are available in the literature as we can see in some published reviews on the topic [36, 41, 42].

Imbalanced learning. Undersampling and oversampling methods

In many supervised learning applications, there is a significant difference between the prior probabilities of different classes, i.e., between the probabilities with which an example belongs to the different classes of the classification problem. This situation is known as the class imbalance problem [43]. The hitch with imbalanced datasets is that standard classification learning algorithms are often biased towards the majority class (known as the "negative" class) and therefore there is a higher misclassification rate for the minority class instances (called the "positive" examples).

While algorithmic modifications are available for imbalanced problems, our interest lies in preprocessing techniques to alleviate the bias produced by standard data mining algorithms. These preprocessing techniques proceed by resampling the data to balance the class distribution. The main advantage is that they are independent of the data mining algorithm applied afterwards.

Two main groups can be distinguished within resampling. The first one is *undersampling methods*, which create a subset of the original dataset by eliminating (majority) instances. The second one is *oversampling methods*, which create a superset of the original dataset by replicating some instances or creating new instances from existing ones.

Non-heuristic techniques, as random-oversampling or random-undersampling were initially proposed, but they tend to discard information or induce over-fitting. Among the more sophisticated, heuristic approaches, "Synthetic Minority Oversampling TEchnique" (SMOTE) [44] has become one of the most renowned approaches in this area. It interpolates several minority class examples that lie together. Since SMOTE can still induce over-fitting in the learner, its combination with a plethora of sampling methods can be found in the specialized literature with excellent results. Under-sampling has

the advantage of producing reduced data sets, and thus interesting approaches based on neighborhood methods, clustering and even evolutionary algorithms have been successfully applied to generate quality balanced training sets by discarding majority class examples.

Data preprocessing in new data mining fields

Many data preprocessing methods have been devised to work with supervised data, since the label provides useful information that facilitates data transformation. However, there are also preprocessing approaches for unsupervised problems.

For instance, FS has attracted much attention lately for unsupervised problems [45–47] or missing values imputation [48]. Semisupervised classification, which contains instances both labeled and unlabeled, also shows several works in preprocessing for discretization [49], FS [46], instance selection [50] or missing values imputation [51]. Multi-label classification is a framework prone to gather imbalanced problems. Thus, methods for re-sampling these particular data sets have been proposed [52, 53]. Multi-instance problems are also challenging, and resampling strategies have been also studied for them [54]. Data streams are also a challenging area of data mining, since the information represented may change with time. Nevertheless, data streams are attracting much attention and for instance preprocessing approaches for imputing missing values [55, 56], FS [57] and IR [58] have been recently proposed.

Big data preprocessing

This section aims at detailing a thorough list of contributions on Big Data preprocessing. Table 1 classifies these contributions according to the category of data preprocessing, number of features, number of instances, maximum data size managed by each algorithm and the framework under they have been developed. The size has been computed multiplying the total number features by the number of instances (8 bytes per datum). For sparse methods (like [59] or [60]), only the non-sparse cells have been considered. Figure 4 depicts an histogram of the methods using the size variable. It can be observed as most of methods have only been tested against datasets between zero an five gigabytes, and few approaches have been tested against truly large-scale datasets.

Once seen a snapshot of the current developments in Big Data preprocessing, we will give shorts descriptions of the contributions in the rest of this section. First, we describe one of the most popular machine learning library for Big Data: MLlib; which brings a wide range of data preprocessing techniques to the Spark community. Next the rest of sections will be devoted to enumerate those contributions presented in the literature, and categorized and arranged in the Table 1.

MLlib: a spark machine learning library

MLlib [61] is a powerful machine learning library that enables the use of Spark in the data analytics field. This library is formed by two packages:

- *mllib*: this is the first version of MLlib, which was built on top of RDDs. It contains the majority of the methods proposed up to now.
- *ml*: it comes with the newest features of MLlib for constructing ML pipelines. This higher-level API is built on enhanced DataFrames structures [62].

Table 1 Analyzed methods and the maximum data size managed by each one

Methods	Category	# Features	# Instances	Size (GB)	Framework
[70]	FS	630	65,003,913	305.1196	Hadoop MapReduce
[69]	FS	630	65,003,913	305.1196	Hadoop MapReduce
[74]	FS	1,156	5,670,000	48.8350	MPI
[60]	FS	29,890,095	19,264,097	4.1623	C++/MATLAB
[59]	FS	100,000	10,000,000	1.4901	MapReduce
[76]	FS	100	1,600,000	1.1921	Apache Spark
[80]	FS	127	1,131,571	1.0707	Hadoop MapReduce
[71]	FS	54,675	2,096	0.8538	Hadoop MapReduce
[75]	FS	54	581,012	0.2338	Hadoop MapReduce
[73]	FS	20	1,000,000	0.1490	MapReduce
[77]	FS	–	–	0.0976	Hadoop MapReduce
[79]	FS	256	38,232	0.0729	Hadoop MapReduce
[68]	FS	52	5,253	0.0020	Hadoop MapReduce
[78]	FS	–	–	0.0000	Hadoop MapReduce
[67]	FS	–	–	0.0000	Hadoop MapReduce
[72]	FS	–	–	0.0000	Hadoop MapReduce
[83]	Imbalanced	630	32,000,000	150.2037	Hadoop MapReduce
[84]	Imbalanced	630	32,000,000	150.2037	Hadoop MapReduce
[90]	Imbalanced	630	16,000,000	75.1019	Apache Spark
[89]	Imbalanced	41	4,856,151	1.4834	Hadoop MapReduce
[82]	Imbalanced	41	4,000,000	1.2219	Hadoop MapReduce
[81]	Imbalanced	14	1,432,941	0.1495	Hadoop MapReduce
[86]	Imbalanced	9,731	1,446	0.1048	Hadoop MapReduce
[91]	Imbalanced	14	524,131	0.0547	Hadoop MapReduce
[87]	Imbalanced	36	95,048	0.0255	Hadoop MapReduce
[88]	Imbalanced	8	2,687,280	0.0200	Hadoop MapReduce
[93]	Incomplete	625	4,096,000	19.0735	MapReduce (Twister)
[92]	Incomplete	481	191,779	0.6873	Hadoop MapReduce
[95]	discretization	630	65,003,913	305.1196	Apache Spark
[96]	discretization	630	65,003,913	305.1196	Apache Spark
[94]	discretization	–	–	4.0000	Hadoop MapReduce
[97]	IR	41	4,856,151	1.4834	Hadoop MapReduce

The methods are grouped by preprocessing task, and ordered by maximum data size. Those methods with no information about number of features or instances have been set to zero size

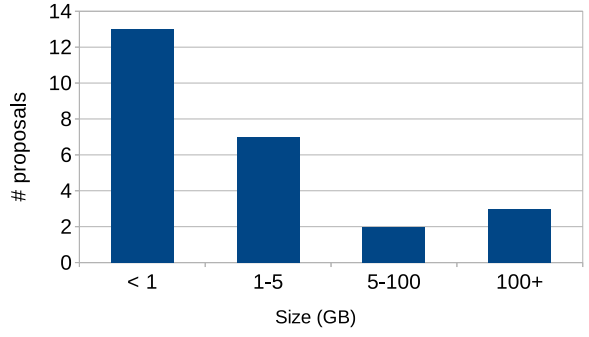

Fig. 4 Maximum data size managed by each preprocessing method (in gigabytes)

Here, we describe and classify all data preprocessing techniques for both versions[1] into five categories: discretization and normalization, feature extraction, feature selection, feature indexers and encoders, and text mining.

Discretization and normalization

Discretization transforms continuous variables using discrete intervals, whereas normalization just performs an adjustment of distributions.

- Binarizer: converts numerical features to binary features. This method makes the assumption that data follows a Bernoulli distribution. If a given feature is greater than a threshold it yields a 1.0, if not, a 0.0.
- Bucketizer: discretizes a set of continuous features by using buckets. The user specifies the number of buckets.
- Discrete Cosine Transform: transforms a real-valued sequence in the time domain into another real-valued sequence (with the same size) in the frequency domain.
- Normalizer: normalizes each row to have unit norm. It uses parameter p, which specifies the p-norm used.
- StandardScaler: normalizes each feature so that it follows a normal distribution.
- MinMaxScaler: normalizes each feature to a specific range, using two parameters: the lower and the upper bound.
- ElementwiseProduct: scales each feature by a scalar multiplier.

Feature extraction

Feature extraction techniques combine the original set of features to obtain a new set of less-redundant variables [63]. For example, by using projections to low-dimensional spaces.

- Polynomial Expansion expands the set of features into a polynomial space. This new space is formed by an n-degree combination of the original dimensions.
- VectorAssambler: combines a set of features into a single vector column.
- Single Value Decomposition (SVD) is matrix factorization method that transform a real/complex matrix M (mxn) into a factorized matrix A.
  The creators expose that for large matrices it is not needed the complete factorization but only to maintain the top-k singular values and vectors. In such way, the dimensions of the implied matrices will be reduced. They also assume that n is much smaller than m (tall-and-skinny matrices) in order to avoid a severe degradation of the algorithm's performance.
- Principal component analysis (PCA) tries to find a rotation such that the set of possibly correlated features transforms into a set of linearly uncorrelated features. The columns used in this orthogonal transformation are called principal components. This method is also designed for matrices with a low number of features.

Feature selection

As explained before, FS tries to select relevant subsets of relevant features without incurring much loss of information [64].

- VectorSlicer: the user selects manually a subset of features.
- RFormula: selects features specified by an R model formula.

- Chi-Squared selector: it orders categorical features using a Chi-Squared test of independence from the class. Then, it selects the most-dependent features. This is a filter method, which needs the number of features to select.

Feature indexers and encoders

These functions convert features from one type to another using indexing or encoding techniques.

- StringIndexer: converts a column of string into a column of numerical indices. The indices are ordered by label frequencies.
- OneHotEncoder: maps a column of strings to a column of unique binary vectors. This encoding allows better representation of categorical features since it removes the numerical order imposed by the previous method.
- VectorIndexer: automatically decides which features are categorical and transform them to category indices.

Other preprocessing methods for text mining

Text mining techniques try to structure the input text, yielding structured patterns of information.

- TF-IDF: this tool is aimed at quantifying how relevant each term is to a document, given a complete set of documents. Term Frequency (TF) measures the number of times that a term appears in a documents, whereas Inverse Document Frequency (IDF) measures how much information is given by a term according to its document frequency. TF is implemented using feature hashing for a better performance, so that each raw feature is mapped into an index. The dimension of the hast table is normally quite high (2^{20}) in order to avoid collisions.
- Word2Vec: it takes as input a text corpus and yields as output the word vectors. It first constructs a vocabulary from the text, and then learns vector representation of words.
- CountVectorizer: transforms a corpus into a set of vectors of token counts. It extracts the vocabulary using an estimator and counts the number of occurrences for each term.
- Tokenizer: breaks some text into individual terms using simple or regular expressions.
- StopWordsRemover: removes irrelevant words from the input text. The list of stop words is specified as parameter.
- n-gram: generates sequences of n-grams terms, where each one is formed by a space-delimited string of n consecutive words.

Feature selection

As mentioned before, FS has a key role to play in dealing with large-scale datasets, especially those that present an ultra-high dimensionality. However, FS methods, like many other learning methods, suffers from the "curse of dimensionality" [65], and consequently, are not expected to scale well. New paradigms and tools have emerged to solve this problematic [66]. Most of them are centered in the use of parallel processing to distribute the massive complexity burden across several nodes. Here, a list of the contributions for FS is presented:

- [59]: Singh et al. proposed a new approximate heuristic, optimized for logistic regression in MapReduce, which employs a greedy search to select features increasingly.
- [67]: Meena et al. designed an evolutionary approach based on Ant Colony Optimization (ACO) with the aim of finding the optimal subset of features. It parallelizes on Hadoop MapReduce some parts of the algorithm, such as: tokenization, the computation of association degrees, and the evaluation of solutions.
- [68]: Tanupabrungsun et al. proposed a Genetic Algorithm (GA) approach with a wrapper fitness function. In this work, the Hadoop master process is in charge of the management of the population whereas the fitness evaluation is parallelized.
- [69]: Triguero et al. proposed an evolutionary feature weighting model to learn the feature weights per map. They introduced a Reduce phase adding the weights and using a threshold to select the most relevant instance. This model was the winner solution in the ECBDL'14 competition.
- [70]: Peralta et al. proposed a different approach based on independent GA processes (executed on each partition). A voting scheme is employed to aggregate the partial solutions.
- [71]: Kumar et al. implemented three feature selectors (ANOVA, Kruskal–Wallis, and Friedman test) based on statistical test. All of them were parallelized on Hadoop MapReduce so as each feature is evaluated independently.
- [72]: Hodge et al. proposed an unified framework which uses binary Correlation Matrix Memories (CMMs) to store and retrieve patterns using matrix calculus. They propose to compute sequentially the CMMs, and them, to distribute them on Hadoop to obtain the final coefficients.
- [73]: A feature selection method based on differential privacy (Laplacian Noise) and a Gini-index measure was designed by Chen et al. This technique was implemented using a general MapReduce model.
- [74]: Zhao et al. proposed a FS framework for both unsupervised and supervised learning, which includes several measures, such as: the Akaike information criterion (AIC), the Bayesian information criterion (BIC), and the corrected Hannan–Quinn information criterion (HQC). This framework has been implemented on MPI.
- [75]: Sun et al. designed a method that computes the total combinatory mutual information, and the contribution degree between all feature variables and class variable. It uses a iterative process (implemented on Hadoop) to select the most relevant features.
- [76]: A filter method based on column subset selection was implemented by Ordozgoiti et al. However, as stated by the authors, this Spark algorithm is not designed to tackle high-dimensional problems.
- [77]: A simple version of TF-IDF (for Hadoop MapReduce) was designed by Chao et al. to deal with text mining problem on Big Data.
- [78]: Dalavi et al. proposed a novel weighting scheme based on supervised learning (using SVMs) for Hadoop MapReduce.
- [79]: He et al. implemented on Hadoop a FS method using positive approximation as an accelerator for traditional rough sets.

- [80]: Wang et al. designed a family of feature selection algorithms for online learning. The algorithm selects those features with bigger weights, according to a linear classifier based on L1-norm.
- [60]: Tan et al. reformulated the FS problem as a convex semi-infinite programming problem. They also proposed to speed up the training phase through several cache techniques and a modified accelerated proximal gradient method. This sequential approach (written in C++ and MatLab) has been included on this list because of its relevance and promising results on Big Data (see Table 1).

Imbalanced data

Classification problems are typically formed by a small set of classes. Some of them come with a tiny percentage of instances compared with the other classes. This highly-imbalanced problems are more noteworthy in Big Data environments where millions of instances are present. Some contributions to this topic have been implemented on Hadoop MapReduce:

- [81]: The first approach in dealing with imbalanced large-scale datasets was proposed by Park et al. In this work, a simple over-sampling technique was employed using Apache Hadoop and Hive on traffic data with a 14 % of positive instances.
- [82]: Hu et al. proposed an enhanced version of Synthetic Minority Over-sampling Technique (SMOTE) algorithm on MapReduce. This method focused on replicating those minority cases that only belong to the boundary region to solve the problem of original SMOTE, which omits the distribution of the original data while yields new samples.
- [83]: Rio et al. adapted some over-sampling, under-sampling and cost-sensitive methods for MapReduce. All these distributed algorithms apply a sampling technique on each data partition, and reduce the partial results by randomly selecting a fixed amount of instances. It was extended for extremely imbalanced data in [84] using a high over-sampling rate to highlight the presence of the minority class. This proposal has been tested against several bio-informatics problems (like contact map prediction and orthogonal detection) with accurate results [69, 85].
- [86]: Wang et al. proposed an algorithm that weighs the penalties associated to each instance in order to reduce the effect of less important points. This weighted boosting method aims at adjusting the weights of each instance in each iteration. The weighted instances are finally classified by a SVM classifier.
- [87]: Bhagat et al. proposed an extension to this work where a combination of SMOTE and a One-vs-All (OVA) approach is tested.
- [88]: Zhai et al. designed a oversampling technique based on nearest neighbors for ensemble learning. This technique yields several over-sampled sets by alternating the process between positive and negative instances. In this work, only the neighbors computation is reported to be distributed using MapReduce.
- [89]: Triguero et al. designed an evolutionary undersampling method for Big Data classification. It is based on two MapReduce stages: the first one builds a decision tree in each map after performing undersampling; and the second one, classifies the test set using the set of trees. The building phase is accelerated by a windowing technique. In [90], an iterative model was designed on Apache Spark aiming at

solving extremely imbalance problems [90]. Through Spark in-memory operations, this model is able to make an efficient use of data.

- [91]: Park et al. developed a distributed version of SMOTE algorithm for traffic detection. This version consists of two MapReduce jobs: the first one calculates distances among the input examples; and the second one sorts the results by distance. The latter step caches the original data into a distributed cache, which can be consider as a serious problem for scalability.

Incomplete data

In most of current real-life problems, there is a potential for incomplete data (also called missing data). Because of either human or machine failure, input data can present some gaps or errors. This problem needs to be faced early with some additional techniques (like imputation methods) that prevent the learning process from its negative effect. Although Big Data systems are more prone to incompleteness, just a couple of contributions have been proposed in the literature to solve this:

- [92]: Chen et al. designed a data cleansing method based on the combination of set-valued decision information system, and a deep analysis of missing information. The algorithm implements on Hadoop MapReduce the computation of the equivalent set-valued decision information system and the boolean equivalence matrix. By using this information, they purge duplicate and inconsistent objects from the input data.
- [93]: Zhang et al. run an investigation about the effect of rough sets over incomplete information systems. Its Twister MapReduce implementation aims at accelerating the computation of the relation matrix, one of the main structures in rough sets theory. One of its main advantages is the Sub-Merge operation implemented, which accelerates the process of joining the relation matrices and saves some space.

Discretization

Discretization task is frequently used to improve the performance and effectiveness of classifiers. It is also used to simplify, and therefore, to reduce continuous-valued datasets. For this reason, data discretization has become one of the most important task in the knowledge discovery process. Nevertheless, standard discretization are not prepared to deal with big datasets. Here, we present the unique contributions in this field:

- [94]: Zhang et al. implemented on Hadoop a parallel version of Chi-Squared discretization method. However, the main drawback of this method is its poor scalability due to the merging process is bounded by the number of input features.
- [95]: Ramírez et al. designed an efficient implementation of Fayyad's discretizer on Apache Spark. This entropy minimization proposal was re-adapted to completely distribute the computation burden associated to the most-consuming operations in this method: feature sorting and boundary points generation. In [96], an updated taxonomy of the most relevant discretization methods is presented along with the Big Data challenge that supposes the application of discretization techniques in large-scale scenarios. A distributed entropy minization discretizer is presented and evaluated on several big datasets.

Instance reduction

Instance selection is a type of preprocessing technique, which aims at reducing the number of samples to be considered in the learning phase. In spite of its promising results with small and medium datasets, this task is normally undermined when coping with large-scale datasets (from tens of thousands of instances onwards).

Just one contribution of Triguero et al. [97] has been able to address this problem from a distributed perspective up to now. In this work, the authors apply an advanced IR technique (called SSMA-SFLSDE) over each data partition (map phase) using Hadoop. The reduce phase offers several ways to aggregate the partial instance sets, either by: concatenating all partial results (baseline), filtering noisy prototypes, or merging redundant samples. An extension to this method was proposed in [98]. A second phase of parallelization based on windowing is included in this extension on the mappers side.

In this section, we have reviewed the most important contributions on large-scale preprocessing. Regarding MLlib, it offers a wide set of preprocessing algorithms, however, almost all these methods looks quite simple. Indeed, focusing on FS, only a simple statistical filter (Chi-squared) has been implemented. On the other hand, a list of more complex and diverse contributions have been presented in the literature. Nevertheless, just a few of these methods have been tested against really huge datasets (greater than 5 GBs). Accordingly, more scalable proposals are required to tackle the actual size of incoming data, and to cover neglected fields of preprocessing (like IR).

Challenges and new possibilities in big data preprocessing

This last section of the paper will be devoted to point out all the existing lines in which the efforts on Big Data preprocessing should be made in the next years. The new possibilities on this topic will be centered onto three main key points: new technologies, to scale the data preprocessing techniques and new learning paradigms on which they can be applied.

New technologies

As we can see in previous content of this paper, new technologies for Big Data are emerging in the last years and few attempts of data preprocessing proposals can be found adapted to take advantage of them. It is clear that Spark [10] is offering better performance results than Hadoop [7] in processing. But also, Spark is a newer technology and there has been little time to develop ideas until now. Thus, the near future will offer new methods developed in Spark under the library MLlib [61] which is growing increasingly.

It is worth mentioning that other emerging platform, such as Flink [14], are bridging the gap of stream and batch processing that Spark currently has. Flink is a streaming engine that can also do batches whereas Spark is a batch engine that emulates streaming by micro-batches. This results in that Flink is more efficient in terms of low latency, especially when dealing with real time analytical processing.

In the particular case of data preprocessing in Spark, excepting basic data preprocessing, we can find some developments in FS and discretization for Big Data. Spark is a more mature technology and implements MLlib with tens of already available learning algorithms. This will make easy and encourage the integration of novel data preprocessing methods in a near future. However, it is desirable to start the development of data preprocessing techniques on Flink, in particular with streaming and real-time applications.

Scaling data preprocessing techniques to deal with big data

Another remarkable outcome derived from the previous analysis of existing Big Data preprocessing techniques is that most of the effort has been devoted to the development of FS methods, and even there are some data preprocessing families in which nothing or almost nothing has been done.

- Instance reduction: these techniques will allow us to arrange a subset of data to carry out the same learning tasks that we could do with original data, but with a low decrease of performance. It is very desirable to have a complete set of instance reduction techniques to obtain subsets of data from big databases for certain purposes and paradigms. The key problem is that these techniques have to be re-adjusted to deal with large scale data, they require high computation capabilities and they are assumed to follow an iterative procedure.
- Missing values imputation: it is a hard problem in which many relationships among data have to be analyzed to estimate the best possible value to replace a missing value.
- Noise treatment: it is again a complex problem in which decisions depend on two perspectives: the computation of similarities among data points and the run and fusion of several decisions coming from ensembles to enable the noise identification approach.

Additionally, there is an open issue related to the arrangement and combination of several data preprocessing techniques to achieve the optimal outcome for a data mining process. This is discussed in [99], where the most influential data preprocessing techniques are presented and some instructive experimental studies emphasize the effects caused by different arrangement of data preprocessing techniques. This is an original complex challenge, but it will be more complex according to the data scales in Big Data scenarios. This complexity may also be influenced by other factors that mainly depends of the data preprocessing technique in question; such as its dependency of intermediate results, its capacity of treating different volumes of data, its possibility of parallelization and iterative processing, or even the input it requires or the output it provides.

New big data learning paradigms

Data mining is not a static field and new problems are continuously arising. In consequence data preprocessing techniques are evolving along with data mining and with the appearance of new challenges and problems that data mining tries to tackle, new proposals of data preprocessing methods have been proposed.

These problems are becoming a part of the Big Data universe and they are being currently addressed by some of the mentioned technologies [100]. In addition, they will require data preprocessing techniques to ensure high quality solutions and good performance in the results obtained. This is another major challenge for Big Data preprocessing and it will concern different learning paradigms, besides classification and regression, such as:

- Unsupervised learning: Clustering [101] and rule association mining [102] have been addressed in Big Data. Developments on real-time applications can be also found in the literature [103]. It is well-known that the success of these problems depends heavily on the quality of data, being the data cleaning, transformation and discretization the techniques with the most important role for this.

- Semi-supervised learning: A significant growth of applications and solutions on this paradigm is expected in the near future. Due to the fact of generating and storing more and more data, the labeling of examples cannot be done for all and the predictive or descriptive task will be supported by a subset of labelled examples [104]. Data preprocessing, especially at the instance level [105], would be useful to improve the quality of this kind of data.
- Data streams and real-time processing: processing large data offering real-time responses is one of the most popular and demanding paradigms in business [106]. Currently, there are some specific approaches in Big Data streams [107–109] and even software development [110]. Data preprocessing techniques, such as noise editing [58], should be able to tackle Big Data scenarios in upcoming applications.
- Non-standard supervised problems: there are some other popular supervised paradigms in which Big Data solutions will be necessary soon. This is the case of ordinal classification/regression [111], multi-label classification [52, 53, 112] or multi-instance learning [54]. All the possible data preprocessing approaches will also be required to enable and improve these solutions.

Conclusions

At the present, the size, variety and velocity of data is huge and continues to increase every day. The use of Big Data frameworks to store, process, and analyze data has changed the context of the knowledge discovery from data, especially the processes of data mining and data preprocessing. In this paper, we presented a review on the rise of data preprocessing in cloud computing. We presented a updated categorization of data preprocessing contributions under the big data framework. The review covered different families of data preprocessing techniques, such as feature selection, imperfect data, imbalanced learning and instance reduction as well as the maximum size supported and the frameworks in which they have been developed. Furthermore, the key issues in big data preprocessing were highlighted.

In the future, significant challenges and topics must be addressed by the industry and academia, especially those related to the use of new platforms such as Apache Spark/Flink, the enhancement of scaling capabilities of existing techniques and the approach to new big data learning paradigms. Researchers, practitioners, and data scientists should collaborate to guarantee the long-term success of big data preprocessing and to collectively explore new domains.

Endnote

[1] *mllib* and *ml* documentation: http://spark.apache.org/docs/latest/mllib-guide.html.

Acknowledgements
This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2014-57251-P and the Andalusian Research Plan P11-TIC-7765.

Authors' contributions
All authors have contributed equally to finish this paper. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 6 January 2016 Accepted: 15 September 2016
Published online: 01 November 2016

References

1. Aggarwal CC. Data Mining: The Textbook. Berlin, Germany: Springer; 2015.
2. Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. IEEE Trans Knowl Data Eng. 2014;26(1):97–107.
3. Laney D. 3D Data Management: Controlling Data Volume, Velocity and Variety. 2001. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf. Accessed July 2015.
4. Fernández A, del Río S, López V, Bawakid A, del Jesús MJ, Benítez JM, et al. Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks. Wiley Interdisc Rew Data Min Knowl Discov. 2014;4(5):380–409.
5. Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. In: OSDI 2004. San Francisco, CA; 2004. p. 137–50.
6. White T. Hadoop, The Definitive Guide. Sebastopol: O'Reilly Media, Inc; 2012.
7. Apache Hadoop Project. Apache Hadoop. 2015. http://hadoop.apache.org/. Accessed December 2015.
8. Lin J. Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail! Big Data. 2012;1(1):28–37.
9. Karau H, Konwinski A, Wendell P, Zaharia M. Learning Spark: Lightning-Fast Big Data Analytics. Sebastopol: O'Reilly Media; 2015.
10. Spark A. Apache Spark: Lightning-fast cluster computing. https://spark.apache.org/. Accessed December 2015.
11. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. *NSDI'12*. San Jose; 2012. p. 15–28.
12. InfoWorld. Apache Flink: New Hadoop contender squares off against Spark. 2015. http://www.infoworld.com/article/2919602/hadoop/flink-hadoops-new-contender-for-mapreduce-spark.html. Accessed December 2015.
13. Storm. Apache Storm. 2015. http://storm-project.net/. Accessed December 2015.
14. Flink. Apache Flink. 2015. https://flink.apache.org/. Accessed December 2015.
15. Pyle D. Data Preparation for Data Mining. San Francisco: Morgan Kaufmann Publishers Inc.; 1999.
16. García S, Luengo J, Herrera F. Data Preprocessing in Data Mining. Berlin: Springer; 2015.
17. Han J, Kamber M, Pei J. Data Mining: Concepts and Techniques, 3rd ed. Burlington: Morgan Kaufmann Publishers Inc; 2011.
18. Zaki MJ, Meira W. Data Mining and Analysis: Fundamental Concepts and Algorithms. New York: Cambridge University Press; 2014.
19. Wang H, Wang S. Mining incomplete survey data through classification. Knowl Inf Syst. 2010;24(2):221–33.
20. Luengo J, García S, Herrera F. On the choice of the best imputation methods for missing values considering three groups of classification methods. Knowl Inf Syst. 2012;32(1):77–108.
21. Little RJA, Rubin DB. Statistical Analysis with Missing Data. Wiley Series in Probability and Statistics, 1st ed. New York: Wiley; 1987.
22. Frénay B, Verleysen M. Classification in the presence of label noise: A survey. IEEE Trans Neural Netw Learn Syst. 2014;25(5):845–69.
23. Zhu X, Wu X. Class Noise vs. Attribute Noise: A Quantitative Study. Artif Intell Rev. 2004;22:177–210.
24. Bellman RE. Adaptive Control Processes - A Guided Tour. Princeton, NJ: Princeton University Press; 1961.
25. Hall MA. Correlation-based feature selection for machine learning. Waikato University, Department of Computer Science. 1999.
26. Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res. 2003;3:1157–82.
27. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng. 2014;40(1):16–28.
28. Kim JO, Mueller CW. Factor Analysis: Statistical Methods and Practical Issues (Quantitative Applications in the Social Sciences). New York: Sage Publications, Inc; 1978.
29. Dunteman GH. Principal Components Analysis. A Sage Publications. Thousand Oaks: SAGE Publications; 1989.
30. Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290(5500):2323–326.
31. Tenenbaum JB, Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290(5500):2319–323.
32. Liu H, Motoda H. On issues of instance selection. Data Min Knowl Disc. 2002;6(2):115–30.
33. Olvera-López JA, Carrasco-Ochoa JA, Martínez-Trinidad JF, Kittler J. A review of instance selection methods. Artif Intell Rev. 2010;34(2):133–43.
34. García S, Derrac J, Cano JR, Herrera F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans Pattern Anal Mach Intell. 2012;34(3):417–35.
35. Triguero I, Derrac J, García S, Herrera F. A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans Syst Man Cybern Part C. 2012;42(1):86–100.
36. Liu H, Hussain F, Tan CL, Dash M. Discretization: An enabling technique. Data Min Knowl Discov. 2002;6(4):393–423.
37. Wu X, Kumar V, (eds). The Top Ten Algorithms in Data Mining. Boca Ratón, Florida: CRC Press; 2009.
38. Quinlan JR. C4.5: Programs for Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers Inc.; 1993.
39. Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proceedings of the 20th Very Large Data Bases Conference (VLDB); 1994. p. 487–99.
40. Yang Y, Webb GI. Discretization for naive-bayes learning: managing discretization bias and variance. Mach Learn. 2009;74(1):39–74.
41. Yang Y, Webb GI, Wu X. Discretization methods. In: Data Mining and Knowledge Discovery Handbook. Germany: Springer; 2010. p. 101–16.
42. García S, Luengo J, Sáez JA, López V, Herrera F. A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. IEEE Trans Knowl Data Eng. 2013;25(4):734–50.
43. López V, Fernández A, García S, Palade V, Herrera F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Inf Sci. 2013;250:113–41.

44. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16(1):321–57.
45. Li Z, Tang J. Unsupervised feature selection via nonnegative spectral analysis and redundancy control. IEEE Trans Image Process. 2015;24(12):5343–355.
46. Han J, Sun Z, Hao H. Selecting feature subset with sparsity and low redundancy for unsupervised learning. Knowl-Based Syst. 2015;86:210–23.
47. Wang S, Pedrycz W, Zhu Q, Zhu W. Unsupervised feature selection via maximum projection and minimum redundancy. Knowl-Based Syst. 2015;75:19–29.
48. Ishioka T. Imputation of missing values for unsupervised data using the proximity in random forests. In: Nternational Conference on Mobile, Hybrid, and On-line Learning. Nice; 2013. p. 30–6.
49. Bondu A, Boullé M, Lemaire V. A non-parametric semi-supervised discretization method. Knowl Inf Syst. 2010;24(1):35–57.
50. Impedovo S, Barbuzzi D. Instance selection for semi-supervised learning in multi-expert systems: A comparative analysis. Neurocomputing. 2015;5:61–70.
51. Williams D, Liao X, Xue Y, Carin L, Krishnapuram B. On classification with incomplete data. IEEE Trans Pattern Anal Mach Intell. 2007;29(3):427–36.
52. Charte F, Rivera AJ, del Jesus MJ, Herrera F. MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowl-Based Syst. 2015;89:385–97.
53. Charte F, Rivera AJ, del Jesús MJ, Herrera F. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing. 2015;163:3–16.
54. Xiaoguang W, Xuan L, Nathalie J, Stan M. Resampling and cost-sensitive methods for imbalanced multi-instance learning. In: 13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013. USA: IEEE; 2013. p. 808–16.
55. Jiang N, Gruenwald L. Estimating missing data in data streams. In: 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007; Bangkok; Thailand; 9 April 2007 Through 12 April 2007. Bangkok; 2007.
p. 981–7.
56. Zhang P, Zhu X, Tan J, Guo L. SKIF: a data imputation framework for concept drifting data streams In: Huang J, Koudas N, Jones GJF, Wu X, Collins-Thompson K, An A, editors. CIKM. Toronto; 2010. p. 1869–1872.
57. Kogan J. Feature selection over distributed data streams through convex optimization. In: Proceedings of the 2012 SIAM International Conference on Data Mining. Anaheim; 2012. p. 475–84.
58. Lu N, Lu J, Zhang G, de Mántaras RL. A concept drift-tolerant case-base editing technique. Artif Intell. 2016;230: 108–33.
59. Singh S, Kubica J, Larsen SE, Sorokina D. Parallel large scale feature selection for logistic regression. In: SIAM International Conference on Data Mining (SDM). Sparks, Nevada; 2009. p. 1172–1183.
60. Tan M, Tsang IW, Wang L. Towards ultrahigh dimensional feature selection for big data. J Mach Learn Res. 2014;15: 1371–1429.
61. Meng X, Bradley JK, Yavuz B, Sparks ER, Venkataraman S, Liu D, Freeman J, Tsai DB, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A. MLlib: Machine learning in apache spark. CoRR. J Machine Learning Res. 2015;17(2016):1–7. abs/1505.06807.
62. Armbrust M, Xin RS, Lian C, Huai Y, Liu D, Bradley JK, Meng X, Kaftan T, Franklin MJ, Ghodsi A, Zaharia M. Spark SQL: Relational data processing in spark. In: ACM SIGMOD International Conference on Management of Data. SIGMOD '15. Melbourne; 2015. p. 1383–1394.
63. Guyon I, Gunn S, Nikravesh M, Zadeh LA. Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing). Germany: Springer; 2006.
64. Blum AL, Langley P. Selection of relevant features and examples in machine learning. Artif Intell. 1997;97(1-2): 245–71.
65. Zhai Y, Ong Y, Tsang IW. The emerging "big dimensionality". IEEE Comput Intell Mag. 2014;9(3):14–26.
66. Bolón-Canedo V, Sánchez-Marono N, Alonso-Betanzos A. Recent advances and emerging challenges of feature selection in the context of big data. Knowl-Based Syst. 2015;86:33–45.
67. Meena MJ, Chandran KR, Karthik A, Samuel AV. An enhanced ACO algorithm to select features for text categorization and its parallelization. Expert Syst Appl. 2012;39(5):5861–871.
68. Tanupabrungsun S, Achalakul T. Feature reduction for anomaly detection in manufacturing with mapreduce GA/kNN. In: 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS). Seoul; 2013. p. 639–44.
69. Triguero I, del Río S, López V, Bacardit J, Benítez JM, Herrera F. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. Knowl-Based Syst. 2015;87:69–79.
70. Peralta D, Río S, Ramírez S, Triguero I, Benítez JM, Herrera F. Evolutionary feature selection for big data classification: A mapreduce approach. Math Probl Eng. 2015. Article ID 246139.
71. Kumar M, Rath SK. Classification of microarray using mapreduce based proximal support vector machine classifier. Knowl-Based Syst. 2015;89:584–602.
72. Hodge VJ, O'Keefe S, Austin J. Hadoop neural network for parallel and distributed feature selection. Neural Netw. 2016. doi:10.1016/j.neunet.2015.08.011.
73. Chen K, Wan W-q, Li Y. Differentially private feature selection under mapreduce framework. J China Univ Posts Telecommun. 2013;20(5):85–103.
74. Zhao Z, Zhang R, Cox J, Duling D, Sarle W. Massively parallel feature selection: an approach based on variance preservation. Mach Learn. 2013;92(1):195–220.
75. Sun Z, Li Z. Data intensive parallel feature selection method study. In: International Joint Conference on Neural Networks (IJCNN). USA: IEEE; 2014. p. 2256–262.
76. Ordozgoiti B, Gómez-Canaval S, Mozo A. Massively parallel unsupervised feature selection on spark. In: New Trends in Databases and Information Systems. Communications in Computer and Information Science. Germany: Springer; 2015. p. 186–96.

77. Chao P, Bin W, Chao D. Design and implementation of parallel term contribution algorithm based on mapreduce model. In: 7th Open Cirrus Summit. USA: IEEE; 2012. p. 43–7.
78. Dalavi M, Cheke S. Hadoop mapreduce implementation of a novel scheme for term weighting in text categorization. In: International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). USA: IEEE; 2014. p. 994–9.
79. He Q, Cheng X, Zhuang F, Shi Z. Parallel feature selection using positive approximation based on mapreduce. In: 11th International Conference on Fuzzy Systems and Knowledge Discovery FSKD. USA: IEEE; 2014. p. 397–402.
80. Wang J, Zhao P, Hoi SCH, Jin R. Online feature selection and its applications. IEEE Trans Knowl Data Eng. 2014;26(3):698–710.
81. Park SH, Ha YG. Large imbalance data classification based on mapreduce for traffic accident prediction. In: 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). Birmingham; 2014. p. 45–9.
82. Hu F, Li H, Lou H, Dai J. A parallel oversampling algorithm based on NRSBoundary-SMOTE. J Inf Comput Sci. 2014;11(13):4655–665.
83. del Río S, López V, Benítez JM, Herrera F. On the use of mapreduce for imbalanced big data using random forest. Inf Sci. 2014;285:112–37.
84. del Río S, Benítez JM, Herrera F. Analysis of data preprocessing increasing the oversampling ratio for extremely imbalanced big data classification. In: IEEE TrustCom/BigDataSE/ISPA, Volume 2. USA: IEEE; 2015. p. 180–5.
85. Galpert D, Del Río S, Herrera F, Ancede-Gallardo E, Antunes A, Aguero-Chapin G. An effective big data supervised imbalanced classification approach for ortholog detection in related yeast species. BioMed Res Int. 2015. article 748681.
86. Wang X, Liu X, Matwin S. A distributed instance-weighted SVM algorithm on large-scale imbalanced datasets. In: IEEE International Conference on Big Data. USA: IEEE; 2014. p. 45–51.
87. Bhagat RC, Patil SS. Enhanced SMOTE algorithm for classification of imbalanced big-data using random forest. In: IEEE International Advance Computing Conference (IACC). USA: IEEE; 2015. p. 403–8.
88. Zhai J, Zhang S, Wang C. The classification of imbalanced large data sets based on mapreduce and ensemble of elm classifiers. Int J Mach Learn Cybern. 2016. doi:10.1007/s13042-015-0478-7.
89. Triguero I, Galar M, Vluymans S, Cornelis C, Bustince H, Herrera F, Saeys Y. Evolutionary undersampling for imbalanced big data classification. In: IEEE Congress on Evolutionary Computation, CEC. USA: IEEE; 2015. p. 715–22.
90. Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F. Evolutionary undersampling for extremely imbalanced big data classification under apache spark. In: IEEE Congress on Evolutionary Computation, CEC, In Press. USA: IEEE; 2016.
91. Park S-h, Kim S-m, Ha Y-g. Highway traffic accident prediction usin vds big data analysis. J Supercomput. 2016. doi:10.1007/s11227-016-1624-z.
92. Chen F, Jiang L. A parallel algorithm for datacleansing in incomplete information systems using mapreduce. In: 10th International Conference on Computational Intelligence and Security (CIS). Kunmina, China; 2014. p. 273–7.
93. Zhang J, Wong JS, Pan Y, Li T. A parallel matrix-based method for computing approximations in incomplete information systems. IEEE Trans Knowl Data Eng. 2015;27(2):326–39.
94. Zhang Y, Yu J, Wang J. Parallel implementation of chi2 algorithm in mapreduce framework. In: Human Centered Computing - First International Conference, HCC. Germany: Springer; 2014. p. 890–9.
95. Ramírez-Gallego S, García S, Mourino-Talin H, Martínez-Rego D, Bolon-Canedo V, Alonso-Betanzos A, Benitez JM, Herrera F. Distributed entropy minimization discretizer for big data analysis under apache spark. In: IEEE TrustCom/BigDataSE/ISPA, Volume 2. USA: IEEE; 2015. p. 33–40.
96. Ramírez-Gallego S, García S, Mouriño-Talín H, Martínez-Rego D, Bolón-Canedo V, Alonso-Betanzos A, Benítez JM, Herrera F. Data discretization: taxonomy and big data challenge. Wiley Interdiscip Rev Data Min Knowl Disc. 2016;6(1):5–21.
97. Triguero I, Peralta D, Bacardit J, García S, Herrera F. MRPR: A mapreduce solution for prototype reduction in big data classification. Neurocomputing. 2015;150 Part A:331–45.
98. Triguero I, Peralta D, Bacardit J, García S, Herrera F. A combined mapreduce-windowing two-level parallel scheme for evolutionary prototype generation. In: IEEE Congress on Evolutionary Computation (CEC); 2014. p. 3036–043.
99. García S, Luengo J, Herrera F. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. Knowl-Based Syst. 2016. doi:10.1016/j.knosys.2015.12.006.
100. Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Khan SU. The rise of "big data" on cloud computing: Review and open research issues. Inf Syst. 2015;47:98–115.
101. Tsapanos N, Tefas A, Nikolaidis N, Pitas I. A distributed framework for trimmed kernel k-means clustering. Pattern Recogn. 2015;48(8):2685–698.
102. Chen Y, Li F, Fan J. Mining association rules in big data with ngep. Clust Comput. 2015;18(2):577–85.
103. Aghabozorgi S, Seyed Shirkhorshidi A, Ying Wah T. Time-series clustering - a decade review. Inf Syst. 2015;53(C): 16–38.
104. Zhu Q, Zhang H, Yang Q. Semi-supervised affinity propagation clustering based on subtractive clustering for large-scale data sets. In: Intelligent Computation in Big Data Era. Germany: Springer; 2015. p. 258–265.
105. Triguero I, García S, Herrera F. SEG-SSC: a framework based on synthetic examples generation for self-labeled semi-supervised classification. IEEE Trans Cybern. 2015;45(4):622–34.
106. Gupta S. Learning Real-time Processing with Spark Streaming. Birmingham: PACKT Publishing; 2015.
107. Works K, Rundensteiner EA. Practical identification of dynamic precedence criteria to produce critical results from big data streams. Big Data Res. 2015;2(4):127–44.
108. Luts J. Real-time semiparametric regression for distributed data sets. IEEE Trans Knowl Data Eng. 2015;27(2):545–57.

109. Sun D, Zhang G, Yang S, Zheng W, Khan SU, Li K. Re-stream: Real-time and energy-efficient resource scheduling in big data stream computing environments. Inf Sci. 2015;319:92–112.
110. De Francisci Morales G, Bifet A. Samoa: Scalable advanced massive online analysis. J Mach Learn Res. 2015;16(1): 149–53.
111. Gutiérrez PA, Pérez-Ortiz M, Sánchez-Monedero J, Fernandez-Navarro F, Hervás-Martínez C. Ordinal regression methods: survey and experimental study. IEEE Trans Knowl Data Eng. 2015;28(1):127–46.
112. Gibaja E, Ventura S. A tutorial on multilabel learning. ACM Comput Surv. 2015;47(3):52–15238.

Submit your next manuscript to BioMed Central and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at
www.biomedcentral.com/submit