electronics

MDPI

Article

FDR^2-BD: A Fast Data Reduction Recommendation Tool for Tabular Big Data Classification Problems

María José Basgall [1,2,3], Marcelo Naiouf [2] and Alberto Fernández [3,*]

[1] National Scientific and Technical Research Council, CONICET, La Plata 1900, Argentina; mjbasgall@lidi.info.unlp.edu.ar
[2] Institute of Research in Computer Science LIDI, III-LIDI, Scientific Research Commission, Province of Buenos Aires, CIC-PBA, School of Computer Science, National University of La Plata, UNLP, La Plata 1900, Argentina; mnaiouf@lidi.info.unlp.edu.ar
[3] Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence, DaSCI, University of Granada, 18071 Granada, Spain
* Correspondence: alberto@decsai.ugr.es

Abstract: In this paper, a methodological data condensation approach for reducing tabular big datasets in classification problems is presented, named FDR^2-BD. The key of our proposal is to analyze data in a dual way (vertical and horizontal), so as to provide a smart combination between feature selection to generate dense clusters of data and uniform sampling reduction to keep only a few representative samples from each problem area. Its main advantage is allowing the model's predictive quality to be kept in a range determined by a user's threshold. Its robustness is built on a hyper-parametrization process, in which all data are taken into consideration by following a *k*-fold procedure. Another significant capability is being fast and scalable by using fully optimized parallel operations provided by Apache Spark. An extensive experimental study is performed over 25 big datasets with different characteristics. In most cases, the obtained reduction percentages are above 95%, thus outperforming state-of-the-art solutions such as FCNN_MR that barely reach 70%. The most promising outcome is maintaining the representativeness of the original data information, with quality prediction values around 1% of the baseline.

Keywords: big data; data reduction; classification; preprocessing techniques; Apache Spark

Citation: Basgall, M.J.; Naiouf, M.; Fernández, A. FDR^2-BD: A Fast Data Reduction Recommendation Tool for Tabular Big Data Classification Problems. *Electronics* **2021**, *10*, 1757. https://doi.org/10.3390/electronics10151757

Academic Editor: Rashid Mehmood

Received: 23 June 2021
Accepted: 18 July 2021
Published: 22 July 2021

1. Introduction

The term big data mainly refers to the huge amount of data being continuously generated [1]. The technological developments society is facing, such as IoT, cloud computing, social networks, and mobile devices, among others, are the main reasons for this growth, leading to Industry 4.0 [2,3]. However, big data includes not only the volume but also the variety and velocity, among other aspects, involved in concepts that are still being defined [4–6].

In the machine learning field, data classification is a widespread task that learns from targeted data aiming to predict the class label of unseen data. For this purpose, different paradigms of classification algorithms exist, and they are used in multiple application fields [7,8]. When analyzing the publicly available tabular big data problems for binary classification, the presence of a notable conceptual redundancy of information in the data might be observed [9] (redundant instances and/or features) that leads to an unnecessary computational cost. Furthermore, it is known that the classifiers only need a set of instances that correctly represent a problem in order to generate an adequate model [10]. In this sense, maintaining the original raw data, including these redundant features and instances, has a strong negative impact related to scalability and storage.

In addition to the former, not all users have direct access to the computational infrastructure suitable for big data or the ability to work with dedicated big data frameworks

such as Apache Spark [11]. It is possible to rent an external cloud computing service, but this may result in an excessive cost when doing the whole data science cycle. In addition, it should be noted that there are state-of-the-art machine learning algorithms that have not yet been developed to be scalable. In contrast, several implementations can be found for smaller size datasets in standard packages for Python, R, among other programming languages.

Applying preprocessing techniques focusing on data condensation/reduction, such as feature and instance selection [12–15], helps to cope with the undesired conceptual redundancy. Having a reduced version of the original data while keeping as much information as possible is always a desirable situation [16,17], but even more so in the big data problems where time-consuming processing tasks are implicit. For the standard size datasets, i.e., "non-big datasets", a myriad of data reduction proposals have been designed [18,19]. However, most of them are focused either on the selection of the most representative features or on the instances. Few proposals are based on a combination or synergy between both filtering methods [20].

Despite the large number of data reduction techniques available in small data scenarios, we were not able to find the same situation for big data analytics. Most of the current data reduction standard approaches for big data are highly costly in terms of execution time and resources, mainly owing to the use of solutions based on clustering and distances ($O(dn^2)$ complexity), which are not easy to calculate in a distributed way without being approximate.

Recent studies have shown that publicly available big datasets used as benchmarks might have questionable quality regarding some unwanted aspects [9,21], such as redundant examples, disproportion of classes, and noise, among others. The presence of some of these issues could be due to the nature of the data or to the artificial generation of the data by replication of instances in order to generate big datasets. Another recurrent characteristic in these large sets of data is that they usually have several features where not all of them have the same discrimination power (if any). In view of the above, designing new big data reduction techniques becomes necessary.

Our hypothesis is that these big tabular datasets may have a large conceptual redundancy that allows us to obtain, from a reduced version of them, similar predictions to those achieved with the full training dataset. Hence, it is not only possible but also necessary and advantageous to perform a data condensation process for big tabular datasets, with the premise of maintaining the predictive quality. For this purpose, it is essential to maintain representative data points for all the initial clusters originally present in the raw dataset. To do so, it is mandatory to reduce the input dimensionality so as to "compress" the information into more dense groups of instances. Then, the most straightforward way to proceed is by applying a uniform sampling based on a class-stratification procedure. This way, the original density function that represents the data should be maintained, keeping the lesser number of samples from each low-dimensional hyper-cube space.

In order to work with the data in a reduced space, the Fast Data Reduction Recommendation tool for tabular big data (FDR^2-BD), an effective and fully scalable methodology, is presented in this paper. The aims of this proposal are the following:

(a) To determine whether a dataset is reducible or not by means of a uniform stratified sub-sampling.
(b) To estimate the maximum possible reduction percentage and the most important features.

Both answers will be based on maintaining a predictive quality range on the original raw data, determined by a threshold. The rationale for this requirement is that simply decreasing data is useless if the resulting data are not representative of the information contained in the original dataset. Thus, a very important capability of our proposal is to include the aforementioned threshold of acceptable loss in the predictive quality that acts as a "controller" between the reduction rate and the prediction values obtained. In some case studies, the need to reduce the data is such that users are willing to accept a slightly

lower predictive power than that achieved with original data in exchange for a sub-sample of the data that allows them to deal with it more quickly and efficiently.

The proposed methodology focuses on the information condensation by reducing data in a dual way (vertical and horizontal). Furthermore, it is based on a hyper-parametrization procedure that is fully transparent to the user. This iterative procedure performs an internal cross validation which gives more support to the individually obtained reduction values, which are aggregated to obtain the final reduction value.

The FDR^2-BD proposal has been developed as a tool using Scala language programming under the Apache Spark framework. It uses several Spark primitives and utilities to achieve good scalability and maximum efficiency, and it follows a global approach design in order to process all data at once to obtain exact results (as if it was sequential processing). Its implementation is available in a repository at https://github.com/majobasgall/big_data_reduction_recommender, accessed on 14 June 2021.

For the sake of contrasting the appropriateness of our proposed methodology for big data reduction, we conducted a thorough analysis over 25 different datasets obtained from different well-known public data repositories. The experimental results show high reduction values for most of the datasets studied, regarding the dimensionality as well as the suggested reduction percentages (around 70% reduction of the features and 98% reduction of the instances) for a predictive loss threshold of 1%. In absolute terms, there are datasets that achieve a horizontal reduction of four, eight, and even nine million instances, while maintaining their predictive capabilities for the original problem. The maximum possible reduction percentages obtained outperform the standard techniques selected for comparison purposes. As our proposal is based on a hyper-parametrization process, this fact gives more robustness to the recommendation values that are yielded.

The remainder of this paper is organized as follows. In Section 2, the big data design approaches as well as a data reduction introduction are briefly discussed. In addition, the current and ongoing data reduction solutions in big data are mentioned. Then, in Section 3, the FDR^2-BD tool's methodology is detailed, and the most relevant technical implementation is highlighted. Then, in Section 4, the experimental environment of this work is described. Next, the results are presented in Section 5. Finally, Section 6 contains the future work observed from the experimental study and summarizes and concludes the manuscript.

2. Data Reduction in Big Data

In this section, some brief comments on the two design schemes for solutions in big data are first presented in order to establish the basis for the different types of solutions that may be encountered (Section 2.1). Furthermore, a concise introduction to data reduction (Section 2.2) is provided. Finally, a quick review of its solutions in big data (Section 2.3) is given.

2.1. Big Data Design Approaches

To take advantage of this enormous amount of data that is implicit to big data, traditional machine learning techniques have to be adapted to be fully scalable, or new ones should be developed [22]. In order to process all this data in the necessary scalable way, a "divide-and-conquer" programming model known as MapReduce [23] is the basis of most widespread open-source frameworks for big data. To design an application for big data using the MapReduce scheme, two approaches can be found regarding data and model distribution [24]. On the one hand, the local approach results in an approximated model because each partition of data is processed separately without any knowledge of the data belonging to the other partitions. Hence, in this approach, each data partition is isolated. On the other hand, the global design generates an exact model because data and models are considered to be shared across all cluster nodes. In other words, data are split into a number of partitions, but, at the same time, each node is able to know the data from other partitions by using distributed data structures. As the reader will be able to interpret

from [24], there is a clear preference for models based on a global design because, as they work with all the data at once, the solutions they generate are identical to those that would be obtained sequentially in terms of exactness.

2.2. Data Reduction

Traditionally, data reduction methods refer to the preprocessing tasks that are commonly applied in the pipeline of the knowledge extraction process for downsizing a dataset with respect to the number of examples (also known as *horizontal or row-wise reduction*) without losing the information that it represents. For instance, in a classification task, training a model from the reduced data would perform similarly in comparison to training with the whole dataset. In the taxonomy, two widely used groups of data reduction techniques can be found: instance selection (IS) [13] and instance generation (IG) [18]. On the one hand, the IS task's goal is to select a subset of an original dataset; on the other hand, IG methods provide a reduced set of data that can be built from a complete set of synthetic instances or in combination with a sub-sample of the original dataset.

In the specialized literature, it can be found that the basis of most data reduction methods usually involves applying an instance-based classifier algorithm. A widespread IS method is the fast condensed nearest neighbor (FCNN) rule [25], which uses the NN rule to find a sub-sample of instances from the initial dataset. The idea behind FCNN is that it starts with an initial subset consisting of the centroids of each class, and, in the subsequent iterations, the subset is augmented until the stop condition is reached. The way to add instances into the subset is by incorporating the nearest enemy that is inside the Voronoi region of each instance that belongs to the subset. Despite having relatively good reduction results, regarding the complexity, the FCNN has a quadratic time complexity in terms of the number of instances in the worst case.

More instance-based methods constitute the state-of-the-art for data reduction, such as steady-state memetic algorithm (SSMA) [26], random mutation hill climbing (RMHC) [27], and democratic instance selection (DIS) [28]. The significance of these approaches is mainly based on their extension to the big data scenario, as will be introduced in what follows.

However, data can also be reduced regarding their variables (also known as *vertical, dimensionality or column-wise reduction*), not only because data are highly dimensional, but also, when they are not, there is a need for working with a subset of features that are the most representative, as this boosts the robustness of the final learned model. The most popular variable-based reduction techniques are feature selection (FS) and feature extraction [12,29]. The former, as its name suggests, selects the most important features that can affect the behavior of a machine learning algorithm. The latter builds a whole set of new features from combining the original ones. Among the methods in the theoretical framework of FS [30] for dimensionality reduction, those belonging to the group of embedded solutions are the most prominent. The idea is based on obtaining the contribution of each variable after training a classifier model. To mention the most relevant features of these methods, they are the most accurate and much less prone to overfitting. Among the different embedded solutions, one of the most widely used is random forest (RF) due to its good results in general-purpose applications [31,32].

Regardless of which data reduction method is applied, the actual benefits of its application in the data science life-cycle are alleviating storage and memory requirements, the complexity of algorithmic computation, and runtimes, in addition to the fact that they may obtain better quality data by removing conceptual redundancy [9].

2.3. Data Reduction Meets Big Data

Dealing with big data does not mean all of the data is going to be relevant for the knowledge discovery process. Usually there is a subset of data that contains useful information, and from it, there is a smaller piece of data that retains the real knowledge. Therefore, it is necessary to eliminate these layers of data that do not involve more than noise to the modeling [17].

When it comes to the big data scenario, few distributed solutions to traditional instance-based reduction approaches can be found, apart from the traditional FCNN method. Both the FCNN and SSMA techniques have been made scalable (FCNN_MR [33] and SSMA-SFLSDE [34], respectively) using the MRPR framework [35]. This framework divides the input data, and following a local MapReduce approach, the corresponding IS task is applied to each split. Then, different strategies are available in order to merge the reduced set of data of each split.

The parallel DIS implementation (MR-DIS [36]) also follows a local approach while that of the RMHC (RHMC_MR [33]) applies the kNN algorithm repeatedly following a global approach. In [37], authors have compared the IS methods for big data, and they have concluded that there is not a clear outperforming method overall. However, regarding their runtime, FCNN_MR is the only approach whose computational cost is within an acceptable range.

With respect to the feature-based methods available for big data, in [38], several state-of-the-art information theory-based methods of distributed implementations can be found. In addition, the fully parallel implementation of RF is included as part of the available machine learning algorithms provided by the MLlib library [22] of Apache Spark.

3. FDR^2-BD: A Fast Data Reduction Recommendation Tool for Tabular Big Data Classification Problems

In this section, we present and describe FDR^2-BD, a novel methodology for information condensation in tabular big data classification problems. The outcome is a decision support tool that provides a set of recommended parameters, in terms of feature and instance reduction values, by means of a hyper-parametrization process. As such, the aforementioned output values are well-supported, as they are aggregated from different subsets from the training data. The global design of this solution was developed to be fast and scalable even when addressing very large data files. This is further achieved by including native core functions from Apache Spark in the implementation.

The remainder of this section is organized as follows. First, the complete design and characteristics that comprise FDR^2-BD are detailed in Section 3.1. Then, the technical highlights that allow for boosting the scalability of our tool implementation are commented on in Section 3.2.

3.1. FDR^2-BD Description and Workflow

The complete design of the FDR^2-BD methodology addresses three requirements that we consider relevant for data reduction solutions.

1. First, the ability to operate automatically by means of a hyper-parametrization process, which is based on an ordered descendent list of suggested reduction percentages. To do so, it works over a dataset following a *k*-fold cross validation scheme in order to ensure all data are analyzed.
2. Second, the possibility to achieve the maximum level of data reduction by using thresholds of acceptable loss of predictive performance. Both the list of the percentages (`redPerc list`) and the predictive loss threshold (PLT) influence the ending conditions of the hyper-parametrization process, and they can be set by the data expert user.
3. Finally, the capability of being a scalable methodology able to deal with big data in reasonable times by taking advantage of parallel computing. To achieve this, the technical implementation details are described in Section 3.2.

The focus of FDR^2-BD is to yield parameters for a recommended horizontal and vertical tabular data condensation towards classification problems. Specifically, the user can apply a more or less granular reduction percentage list (as pointed out in requirement #1) and/or set how much predictive quality they are willing to lose for gaining data reduction (as referred to in requirement #2). This comprises the scenario from which the dataset is found to be non-reducible due to a large loss in the model performance. Among

the different predictive performance measures, this proposal uses geometric mean (GM) (see Equation (2)) because it consolidates the quality of all classes represented, regardless of their a priori distribution.

The complete workflow of FDR^2-BD, as depicted in Figure 1, enables data condensation in three different stages that are enumerated below:

1. First, an initial reduction regarding the dimensionality (vertical or column-wise reduction) can be achieved through the selection of the most important features.
2. Second, if the FS generates redundant instances, i.e., exact duplicated "rows", they are removed (horizontal or row-wise reduction).
3. Third, the main process is carried out by means of a hyper-parametrization with the aim to decide what percentage of instances could be removed while maintaining a desired quality threshold. From the resultant decision, a final and intensive vertical reduction can be carried out.

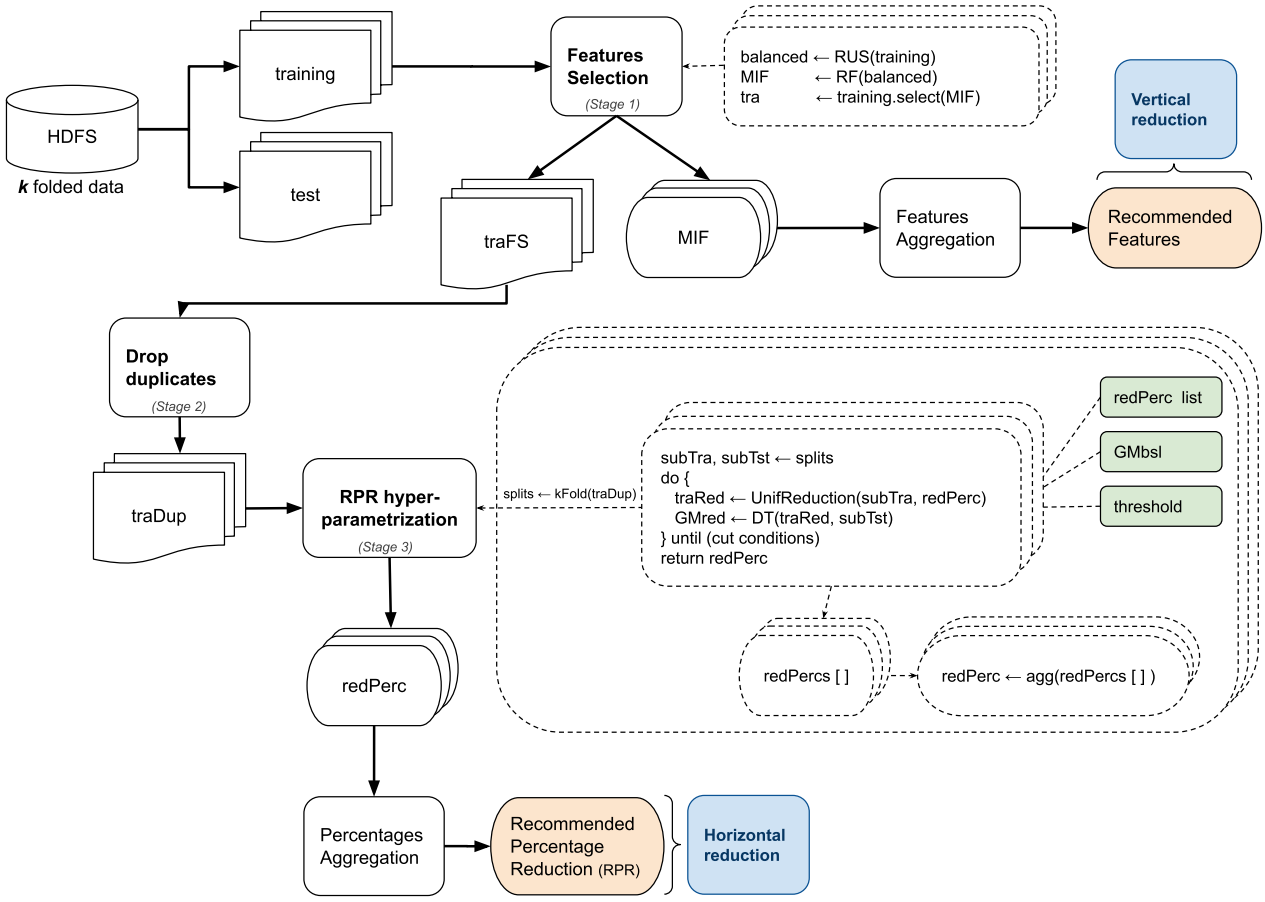


Figure 1. The FDR^2-BD's workflow showing the three main stages of the procedure.

In what follows, each single step of the procedure is detailed.

First Stage: Feature selection.

Some of the attributes in these kinds of datasets may not be representative and necessary for example classification and thus can be omitted. With this vertical reduction, conceptual redundancy can be easily found as duplicated instances.

The initial input data follow the *k*-fold cross validation method, with k equal to 5 (5fcv), so for each fold or training subset, the FS step is performed. Our design must take into account the possible event of an uneven class distribution. Indeed, this may cause a bias of the feature importance toward majority class examples. To deal with this issue, the FS stage includes an instance preprocessing task for those uneven class sets,

aiming to transform them into balanced ones. One standard approach is random under sampling (RUS), which, as its name suggests, performs random reduction of the majority class instances until achieving the desired proportion between the number of examples of each class [39]. In our methodology, RUS is configured to achieve a perfect class balance.

An embedded model-based feature importance identification procedure is used, due to its greater robustness compared to the use of statistical values. Among the different models, an RF is chosen based on its greater diversity as it is an ensemble model, and because it is fully scalable and native to Spark, according to the capabilities mentioned in Section 2.2. Finally, in order to define the most important features (also referred to `MIFs`), the cumulative explained variance is calculated; then, the chosen features are those which can cover up to 90% of the variance.

The outputs of the FS stage previously described are the k vertical reduced training sets and their corresponding k `MIFs`. Moreover, from the latter, an aggregation is performed with the aim of obtaining the recommended set of features across all folds. To do so, the used criterion is to find the features that were selected as important in more than the 50% of the k folds.

Second Stage: Drop duplicates.

Reducing the dimensionality of the problem causes clusters of data to increase their density, and even duplicated instances can now be found in the dataset. Therefore, it is highly necessary to delete redundant examples (keeping only one occurrence of each) in order to avoid their negative influence, which may affect or distort the computation during the following stage.

Third Stage: Recommended percentage reduction (RPR) hyper-parametrization.

The aim of this stage is to obtain the maximum possible percentage reduction of the instances according to a qualitative threshold of predictive quality. In order to do so, an internal k' fold (Internal $\boldsymbol{k'}$ Fold) process is performed by sub-splitting the k non-redundant sets (generated by the previous stage) into k' folds. Each sub-split is represented as k' pairs of training and test sets (referred to as `subTra` and `subTst`), and they are analyzed by means of a hyper-parametrization process of uniform data sampling to find k reduction percentage values.

The hyper-parametrization process considers each `subTra` and applies an iterative procedure by carrying out a stratified data reduction based on a uniform sub-sampling (`UnifReduction`). Each step of this procedure considers a reduction level according to the current element of the `redPerc list`. A default definition of this list is suggested in Section 4.2 along with additional algorithm parameters.

Two conditions are considered to determine whether the iterative procedure is finished: on the one hand, the performance evaluation of the resultant reduced set of data is within the desired threshold and, on the other hand, there are no more values in `redPerc list`, and thus, the problem cannot be condensed to any of these percentages without affecting the predictive performance.

In order to check the quality of the reduced set (`traRed`), a fast decision tree (DT) model is learned using `traRed`, and then classifies the `subTst`. The performance is evaluated through the GM measure (named `GMred`), and it is compared against the range determined by the average baseline GM (`GMbsl`) and the PLT.

Thus, the process iterates generating different sizes of reduced data until one of the stop conditions is achieved. If the current `GMred` is within the desired range of predictive performance, its corresponding reduction percentage will be the output, and since each of them were obtained from the k' splits procedure, it is necessary to aggregate them in order to have a single output value.

As the FDR^2-BD tool works following a hyper-parametrization scheme, some aggregation criteria must be provided for merging all individual output parameters obtained during the process. Specifically, two complementary approaches are followed: (1) the mini-

mum reduction percentage is selected, being a more conservative approach, and (2) the median reduction percentage is chosen. Whatever criterion is set, a reduction percentage per k' is the output of this step. Following this, one more aggregation task is performed over the previous outputs in order to obtain the FDR^2-BD's recommended percentage reduction (RPR) for the input dataset with the desired predictive loss threshold.

It is worth mentioning that, as FDR^2-BD acts as a parameter recommender tool, it does not generate a reduced dataset but returns a pair of recommendation parameters to modify the dataset in order to reduce it in a dual way. In other words, until the user applies the steps as shown in the pseudocode of Algorithm 1, the data reduction will not be performed. After condensing data by applying the recommended feature and instance reduction, a single model is learnt from it with the aim of obtaining the results. Note that from the step indicated in line 3 onwards, it may not be necessary to use a big data framework to carry out these actions, as long as the data should fit in the main memory.

Algorithm 1 Pseudo-code showing the use of FDR^2-BD for data reduction.

Require: data, redPercList, GMbsl, PLT, k, aggCriterion
1: (recommFeats, RPR) ← run FDR^2-BD using input parameters
2: (training, test) ← split `data`
3: tra ← select the `recommFeats` features from `training`
4: condensedTra ← apply UniformReduction over `tra` with `RPR`
5: model ← train the classifier using `RPR`
6: results ← classify `test` from model

3.2. Technical Implementation Highlights

The FDR^2-BD tool was developed using Scala programming language under the Apache Spark framework. As Spark writes intensively in memory, it is one of the fastest big data frameworks. Our proposal design follows a global approach that considers the whole dataset at once, generating exact results. The Spark version chosen is 3.0.1 due it outperforming the previous versions in scalability, not just in its core functionalities but also in its libraries. Regarding data structures, the dataframes and datasets were selected over the resilient distributed datasets (RDD) base data structure, due they provide the benefits of RDDs with an extra optimization, meaning that they are the fastest ones..

It is important to emphasize that the stages of our tool's workflow were implemented using both Spark primitives and some algorithms and utilities to build distributed machine learning pipelines provided by the MLlib [22] library of Spark. All of them are fully efficient and robust, and they ease the purpose of making our proposal fully scalable. The most relevant primitives and functionalities used in our code are detailed in Table 1.

The FDR^2-BD implementation is available in a repository at https://github.com/majobasgall/big_data_reduction_recommender, accessed on 14 June 2021.

Table 1. Spark primitives and utilities used in FDR^2-BD.

Spark Operation	Description
map	Performs a transformation to each element of a data structure
filter	Selects all data structure elements that satisfy a predicate
sample	Takes a sample of a set of data
union	Combines two Spark data structure row-wise
kFold	Splits data into k pairs of training and validation sets, following a Bernoulli sampling
dropDuplicates	Eliminates duplicated rows of a dataframe
RandomForestClassifier *DecisionTreeClassifier*	Are distributed learning algorithms for data classification

4. Experimental Environment

In this section, the experimental framework is described. To do so, the selected datasets of the study are first shown in Section 4.1, including a brief description and their main characteristics such as the number of examples, variables, and classes. Then, in Section 4.2 the state-of-the-art methods for fair comparison, the classification algorithm for the modeling stage, and their parameters' configuration are presented. Next, the evaluation metrics regarding reduction and performance are described in Section 4.3. Subsequently, in Section 4.4, the scalability evaluation measures are discussed. Lastly, in Section 4.5, the characteristics of the infrastructure used for the experiments and analysis are introduced.

4.1. Datasets' Description

In order to analyze most of the widespread used tabular data in big data classification problems, 25 datasets from different repositories (UCI machine learning [40], OpenML [41], Kaggle [42]) were selected. A summarized description of each set of data alphabetically sorted is shown in Table 2, where the number of examples (#Ex.), number of attributes (#Atts.), number of instances for each class (#(class0; class1)), class distribution percentage (%(class0; class1)), number of each type of feature (Co/Di/Ca, where Co = continuous, Di = discrete, Ca = categorical), and the file size (in MB) are included.

The datasets used for this analysis were split following the *k*-fold cross validation method, with k equal to 5 (5fcv).

Table 2. Datasets summary.

Dataset	#Ex.	#Atts.	#(Class0; Class1)	%(Class0; Class1)	Co/Di/Ca	Size (MB)
Agrawal	1,000,000	9	(672,044; 327,955)	(67.2; 32.8)	4/2/3	71.6
airlines	539,383	7	(299,119; 240,264)	(55.46; 44.54)	4/0/3	18.3
BNG_Australian	1,000,000	14	(573,051; 426,949)	(57.31; 42.69)	14/0/0	80.8
BNG_heart	1,000,000	13	(555,946; 444,054)	(55.59; 44.41)	6/0/7	65.7
census	140,246	41	(132,085; 8162)	(94.18; 5.82)	1/12/28	68.7
click_prediction	1,963,972	11	(1,636,593; 327,379)	(83.33; 16.67)	0/11/0	136.3
covtype1	581,012	54	(369,307; 211,705)	(63.56; 36.44)	10/0/44	71.7
covtype1_vs_2	495,173	54	(283,468; 211,705)	(57.25; 42.75)	10/0/44	61.1
covtype2	581,012	54	(297,544; 283,468)	(51.21; 48.79)	10/0/44	71.7
creditCard	284,015	30	(283,540; 480)	(99.83; 0.17)	28/1/0	145.2
ECBDL14-10mill-90	12,000,000	90	(11,760,000; 240,000)	(98; 2)	60/0/30	3481.6
ethylene_ECO_E_LH	4,208,261	16	(3,895,861; 312,400)	(92.58; 7.42)	16/0/0	517
ethylene_EM_E_LH	4,178,500	16	(3,840,313; 338,191)	(91.91; 8.09)	16/0/0	518.7
fars_fatal	100,968	29	(58,852; 42,116)	(58.29; 41.71)	0/1/28	59
HEPMASS_IR_16	5,578,255	28	(5,250,122; 328,133)	(94.12; 5.88)	28/0/0	1331.2
higgs	11,000,000	28	(5,827,686; 5,172,314)	(52.98; 47.02)	28/0/0	7680
HIGGS_IR_16	6,193,440	28	(5,829,120; 364,320)	(94.12; 5.88)	28/0/0	3378.7
homeCredit	307,511	171	(282,686; 24,825)	(91.93; 8.07)	1/9/161	113.1
hyperplane	1,000,000	10	(500,007; 499,993)	(50; 50)	10/0/0	91.4
klaverjas	981,541	34	(528,339; 453,202)	(53.83; 46.17)	2/32/0	79.2
MiniBooNE	129,590	49	(93,101; 36,489)	(71.84; 28.16)	49/0/0	67.7
rlcp	5,749,132	4	(5,728,201; 20,931)	(99.64; 0.36)	1/3/0	180.1
skin	245,057	3	(194,198; 50,858)	(79.25; 20.75)	0/4/0	3
susy	5,000,000	18	(2,712,173; 2,287,827)	(54.24; 45.76)	18/0/0	1740.8
SUSY_IR_16	2,881,684	18	(2,712,173; 169,511)	(94.12; 5.88)	18/0/0	1022.5

4.2. Comparison Methods, Classifier, and Parameters

As a base classifier for the experimental study, the DTs [43] provided by the Spark's machine learning library (MLlib) [22] are used. Two significant and related reasons support the decision to select this classifier. On the one hand, the implementation of this algorithm in Spark is based on a global design that makes the output more robust by using all training

data at once. On the other hand, it has the advantage of being a very efficient approach for the learning and prediction tasks with respect to other classification paradigms.

In order to confirm our proposal's goodness not only in the context of data reduction but also on baseline classification results, a fair comparison is required. Therefore, two approaches will be used as state of the art: on the one hand, a DT as base model and, on the other hand, the FCNN_MR as a representative of the data reduction techniques available for big data.

It has to be considered that using the raw dataset for model training may not be appropriate or comparable with respect to our technique for two main reasons. First, FS significantly influences the construction and learning of the DT model. Second, there may be a clear bias towards majority concepts for those datasets with a non-uniform distribution of examples in classes. Thus, the baseline method consists of a simple but necessary two-stage preprocessing process: first, an FS and drop duplicates stage following the same procedure as FDR^2-BD (please refer to Section 3.1) and second, an undersampling of instances using the RUS technique to balance the class percentages in a fast and effective way [39]. RUS is a state-of-the-art algorithm for balancing class by random deletion of the majority class instances until the desired ratio between classes is achieved. The choice is based on the fact that both FDR^2-BD and RUS perform uniform sampling in their workflows, and they generate a balanced dataset (in RUS, when setting a ratio between classes of 1:1).

The most relevant parameter configurations used for each method are shown in Table 3. With respect to the FDR^2-BD parameters, they are explained in detail in Section 3. In this study, the redPerc list is arbitrarily set with a 1% granularity between 99% and 90%, and a 10% granularity for the rest, in order to obtain a broad overview of the large list of datasets. The versatility of this proposed tool allows any interested user to provide a list with a finer granularity with the aim of exhaustively analyzing the reduction percentage of each dataset. The FDR^2-BD is run by setting the PLT with a stricter value (1%) and with a more relaxed one (5%). The cumulative explained variance represents the desired value to be covered by the features with the highest values of representativeness. The 1:1 final ratio represents the same quantity of instances for each class in the resultant dataset. The parameters of FCNN_MR are those recommended by the authors (interested readers can find additional details in [35]). Finally, the degree of parallelism is represented by the number of data partitions used by the Spark session, set as 128. The number of partitions refers to the number in which the Spark data structure is divided and distributed across the cluster nodes. For Spark, partitions are the basic units of parallelism, and a data structure is a collection of them.

Table 3. Algorithms and parameters.

Algorithm	Parameter	Value
FDR^2-BD	PLT	1%, 5%
	redPerc list (%)	{99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 80, 70, 60, 50, 40, 30, 20, 10}
	aggregation criteria	minimum
	internal *k*-fCV	5
FS	covered variance	up to 90%
DT	impurity	Entropy
	maxDepth	5
RUS	final ratio	1:1
FCNN_MR	k	1
	ReducerType	join

4.3. Evaluation Metrics: Condensation and Predictive Performance

In this section of the paper, the measurements to asses our proposal from two different approaches are presented: data condensation and predictive capabilities.

To evaluate data condensation, the following dimensionality and number of instances measurements are used:

- Dimensionality reduction percentage (`DimRed`): number of features selected with respect to the original amount in the training data.
- Size reduction percentage (`SizeRed`): number of remaining examples with respect to the original data volume.

Regarding predictive performance evaluation metrics, Table 4 shows a confusion matrix for a binary problem from which the metrics to evaluate the classification task are obtained. This matrix organizes the instances of each class according to their correct or incorrect identification. Therefore, the true positive (TP) and true negative (TN) values represent the prediction quality for each individual class. These measures indicate if the preprocessing is favoring a single class or concept.

Table 4. Confusion matrix for performance evaluation of a binary classification problem.

Actual	**Predicted**	
	Positive	**Negative**
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

One typical metric for assessing classification models is the Accuracy (ACC), defined by Equation (1) when the problem is a binary one.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

However, ACC it is not suitable when data are skewed. Another metric more appropriate for all types of problems is known as the GM [44], and it can be calculated from the following rates obtained from the confusion matrix:

True Positive Rate (TPR) is the percentage of positive instances correctly classified, and it is defined as

$$TPR = \frac{TP}{TP + FN}$$

True Negative Rate (TNR) is the percentage of negative instances correctly classified and it is defined as

$$TNR = \frac{TN}{FP + TN}$$

The GM metric attempts to maximize the accuracy of each one of the two classes at the same time and is noted by Equation (2).

$$GM = \sqrt{TPR * TNR} \tag{2}$$

4.4. Scalability Evaluation

The FDR^2-BD's scalability assessment aims to understand its behavior in a parallel environment, and it is based on the following two types of scaling: sizeup and speedup. The former consists of changing the size of the data while keeping the degree of parallelism (number of partitions) fixed, and the latter evaluates the tool behavior when the parallelism degree changes while keeping constant the data size. The sizeup is defined by Equation (3), where T_{full} is the total runtime obtained by using the full original dataset, whereas T_s is the total runtime when using a version of the original dataset determined by the s size (s times the original dataset). The speedup is defined by Equation (4), where T_p is the total

runtime achieved for a number of partitions set p, and T_1 is the time when no parallelism is applied.

In the context of this work, the total runtime is considered as the period of time since the Spark job is launched until it is finished. It thus includes the typical overhead generated by the Spark environment, loading and distributing data across the computing nodes, processing of the algorithm in itself, and learning and classifying data.

$$sizeup(s) = \frac{T_s}{T_{full}} \tag{3}$$

$$speedup(p) = \frac{T_1}{T_p} \tag{4}$$

4.5. Infrastructure

Regarding the infrastructure used to perform the experiments using Apache Spark, the Atlas cluster at University of Granada was used. The cluster consists of fourteen nodes connected via a Gigabit Ethernet network. Each node has an Intel Core i7-4930K microprocessor at 3.40 GHz, 6 cores (12 threads), and 64 GB of main memory working under Linux CentOS 6.9. The infrastructure works with Hadoop 2.6.0 (Cloudera CDH5.8.0), where the head node is configured as NameNode and ResourceManager, and the rest are DataNodes and NodeManagers.

5. Experimental Results

This section presents the results of applying the FDR^2-BD tool. First, in Section 5.1, the study regarding the data volume reduction generated by the use of FDR^2-BD is presented. Then, the influence of FS with respect to horizontal reduction is shown in Section 5.2. Next, the comparison between the different algorithms mentioned in this work is carried out in Section 5.3. This comparative study focuses on the condensation details and the predictive performance achieved. Finally, the conducted scalability study of our proposal focused on sizeup and speedup is presented in Section 5.4.

5.1. Data Volume Reduction Study

The goal of any reduction method is to find if a dataset is well represented by a condensed subset of it. Of course, there is a clear premise of maintaining the original predictive performance of any model learned from this reduced set, or at least with only a small or admissible impact on it.

In the following, the results obtained with our tool for all the studied big datasets are presented, focusing on the reduction level achieved for two benchmark loss predictive quality thresholds. In Section 1, it is mentioned that the publicly available big datasets, by the way they are generated or obtained, can have redundancy, as partially seen in [9]. The aim of this experimental study over several big datasets is to determine that there not only exists redundancy in the data but also that our tool should be able to find meaningful reduction percentages without loss of predictive capacity.

By applying our proposal on a tabular dataset, one of the outputs is the recommended percentage reduction for the selected prediction loss threshold. The RPR can be "None" if the tool concludes no uniform reduction can be done within the prediction range given by the PLT or a value in the redPerc list (described in Section 4.2).

Table 5 shows the RPR provided by the FDR^2-BD tool for the PLTs chosen for the experimentation, for each dataset. It can be observed that the large majority of the datasets (more than 80%) achieve a significant reduction between 93 and 99%, ensuring a predictive behavior within the desired range. For example, in some datasets, the order of magnitude of the reduction is 6. More specifically, in the case of `ECBDL14-10mill-90`, the dataset is reduced from 9.6 million instances to 96 thousand, the `higgs` dataset goes from 8.8 million to 109 thousand, and the `susy` dataset is reduced from 4 million instances to 40 thousand.

The results obtained clearly answer our hypothesis formulated at the beginning of this section, reinforcing that there is indeed a high level of conceptual redundancy in most of the datasets. For `census` and `airlines` datasets, no suggested reduction percentages are obtained. The predictive quality achieved by the hyper-parametrization process is not within the accepted range considering the given threshold. Regarding the `skin` dataset, it can be reduced by this methodology if the PLT is set to 5%, obtaining a reduction of 97%. It is worth mentioning that the GM baseline result for this dataset is around 0.965, so reducing it 97% by accepting a 5% loss of predictive performance is still a good compromise. In a similar way, `fars_fatal` dataset achieves a significantly larger RPR when shifting the PLT from 1 to 5%.

Table 5. Recommended percentage reduction (RPR) per each prediction loss threshold (PLT).

	PLT [%]			**PLT [%]**	
	1	**5**		**1**	**5**
Dataset	**RPR [%]**		**Dataset**	**RPR [%]**	
Agrawal	99	99	fars_fatal	30	94
airlines	–	–	HEPMASS_IR_16	98	99
BNG_Australian	99	99	higgs	98	99
BNG_heart	99	99	HIGGS_IR_16	99	99
census	–	–	homeCredit	93	98
click_prediction	99	99	hyperplane	99	99
covtype1	97	99	klaverjas	99	99
covtype1_vs_2	96	98	MiniBooNE	95	99
covtype2	94	99	rlcp	99	99
creditCard	98	99	skin	–	97
ECBDL14-10mill-90	99	99	susy	99	99
ethylene_ECO_E_LH	97	99	SUSY_IR_16	98	99
ethylene_EM_E_LH	99	99			

5.2. The Influence of FS in the Data Volume Downsizing

In order to show the behavior or characterization of each dataset from FDR^2-BD, Figure 2 presents the overall and disaggregated horizontal reductions for the PLT equal to 1%. There is no doubt that performing the FS stage as the initial step of the whole process has, in many cases, a significant implication in that the level of reduction is considerably extended, in particular for those datasets with nominal variables. In this type of dataset, a reduction of features makes instance replication much more likely, meaning that the information represented by the dataset is too exhaustive and that the same knowledge can be extracted with a minority subset of it. An extreme example of this is the predominantly nominal `fars_fatal` set of data that has the largest reduction given by the duplicates removal step (almost 99%). At the same time, there are cases where reduction is generated in both stages, although the greatest reduction occurs in the uniform reduction stage. Despite the large reduction achieved with stages 1 and 2, in most of the case studies, it is still necessary to apply the data reduction process by means of a uniformly distributed sampling over the problem space.

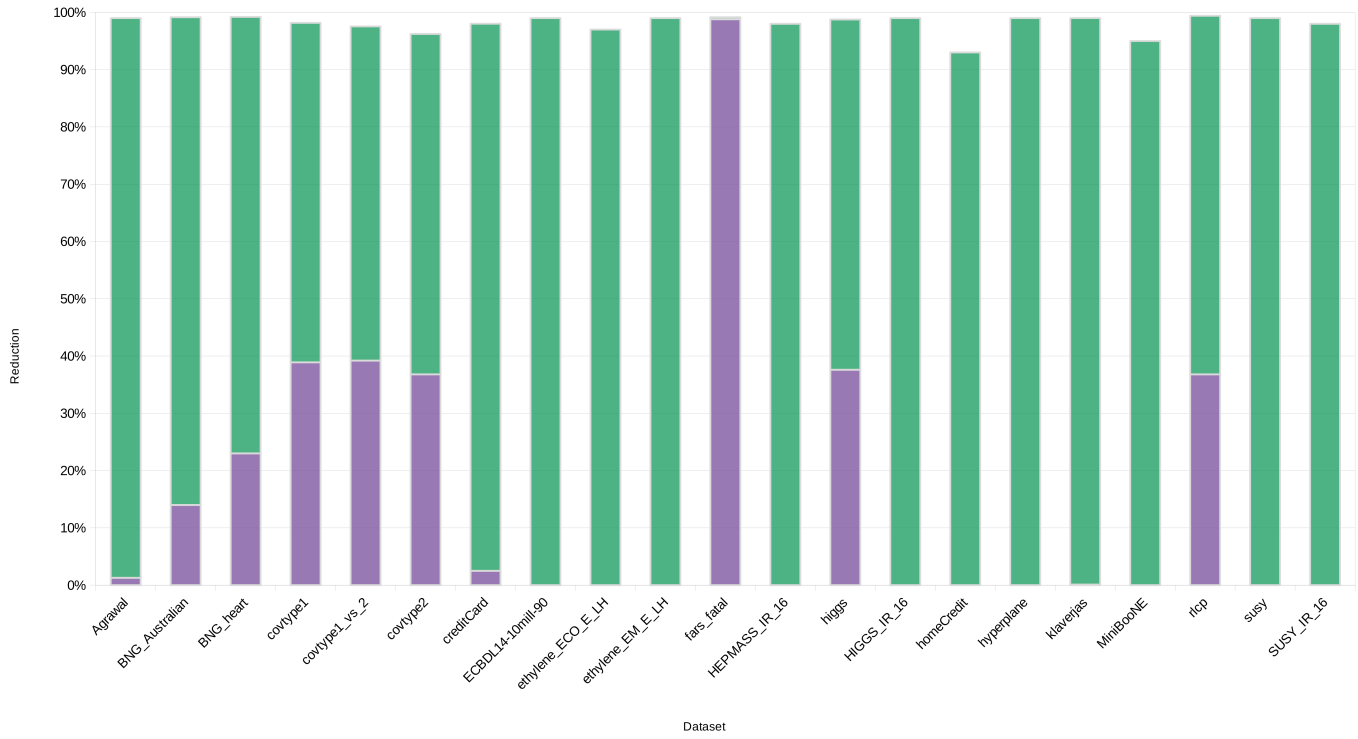


Figure 2. Volume reduction by drop repeated instances (violet) and uniform sampling (green).

5.3. Data Condensation Details and Performance Evaluation

Table 6 shows the data reduction measurements (`SizeRed` and `DimRed`) obtained by each of the comparative techniques and the FDR^2-BD, as well as their resultant datasets' predictive capabilities (measured through the GM metric). In this and the following results, the baseline method is denoted as `BSL`.

It can be observed that FDR^2-BD far outperforms the other algorithms in terms of data condensation for most datasets, achieving similar predictive performance as the baseline. In general terms, our methodology applies a horizontal reduction of 93% to 99%, as it was stressed in the previous section.

There are two datasets for which our tool obtains equivalent results to the other two comparative methods in terms of `SizeRed`. These are the `rlcp` and `creditCard` datasets, and the reasons for this behavior are that `rlcp` is a very simple dataset with low complexity, so there is no difficulty for any of the three methods to achieve the same performance. Regarding `creditCard`, it is a dataset with a high disproportion in its classes, but fortunately, the class clusters are well separated and, therefore, applying the baseline technique also achieves a good modeling.

The results show that our tool achieves a remarkable difference in data reduction: a reduction more than 45% greater than the rest of the experimented techniques, keeping or even improving the predictive quality, of course. The excellent performance obtained using FDR^2-BD over the `Agrawal`, `higgs`, `hyperplane`, `klaverjas` and `susy` datasets excels.

Table 6. Data condensation details and performance evaluation.

	Agrawal			**BNG_Australian**			**BNG_heart**			**click_prediction**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.9475	0.9506	0.9476	0.8606	0.8332	0.8561	0.8541	0.8549	0.8534	0.6112	0.2254	0.6171
SizeRed	42	38	99	14	66	99	23	55	99	68	61	99
DimRed	67	0	67	71	0	71	62	0	62	64	0	64
	covtype1			**covtype1_vs_2**			**covtype2**			**creditCard**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.7534	0.7505	0.7585	0.7557	0.7487	0.7492	0.7411	0.7496	0.7351	0.9269	0.8662	0.9239
SizeRed	56	76	98	38	76	98	37	73	96	99	99	98
DimRed	87	0	89	85	0	91	83	0	85	67	0	73
	ECBDL14-10mill-90			**ethylene_ECO_E_LH**			**ethylene_EM_E_LH**			**HEPMASS_IR_16**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.6969	0.1514	0.6959	0.8814	0.605	0.8777	0.8692	0.7295	0.8693	0.8284	0.7347	0.8252
SizeRed	96	88	99	85	98	97	84	97	99	88	85	98
DimRed	79	0	82	50	0	50	44	0	44	86	0	86
	higgs			**HIGGS_IR_16**			**homeCredit**			**hyperplane**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.6509	0.6634	0.6488	0.6557	0.2771	0.6576	0.5953	0.1388	0.5954	0.3494	0.3432	0.5263
SizeRed	38	39	99	88	75	99	84	74	93	0	42	99
DimRed	82	0	82	79	0	79	90	0	91	60	0	60
	klaverjas			**MiniBooNE**			**rlcp**			**susy**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.8006	0.807	0.8072	0.8743	0.8458	0.8689	0.9309	0.9308	0.9306	0.7612	0.7542	0.762
SizeRed	0	41	99	49	72	95	99	99	99	0	52	99
DimRed	79	0	79	75	0	78	25	0	25	78	0	78
	SUSY_IR_16											
	BSL	**FCNN_MR**	**FDR^2-BD**									
GM	0.7613	0.5604	0.7614									
SizeRed	88	82	98									
DimRed	78	0	78									

It is important to notice that both the FDR^2-BD and baseline apply a feature reduction stage based on the feature importance values. However, the DimRed values could be different for the same dataset as a result of FDR^2-BD determining the recommended features from the aggregation of the obtained features from each fold of a dataset and the baseline selecting the features in only each fold, so they might change from fold to fold. It is therefore shown that the use of feature aggregation of all data folds equals or even improves the percentage of feature reduction.

With respect to the instance selection method behavior, our method is superior against FCNN_MR regarding the predictive quality in more than 66% of the datasets. Furthermore, in those cases when FCNN_MR achieves a GM similar to FDR^2-BD, the SizeRed goes between 38% up to 76%, which is a highly inferior range of values in comparison with the ones obtained by our methodology. Furthermore, due the nature of the FCNN_MR method (as well as the available IS methods for big data), no vertical reduction is generated at all.

In order to summarize the results, Table 7 presents the average GMs and reduction metrics values and a comparison between our methodology with respect to the other ones. The differences between the measures (column `Diff`) and the number of times our tool outperforms, equals, or underperforms the tool being compared (column `W/T/L`, meaning Win/Tie/Loses) are shown.

Table 7. Performance evaluation and data condensation (average, differences, and W/T/L counting).

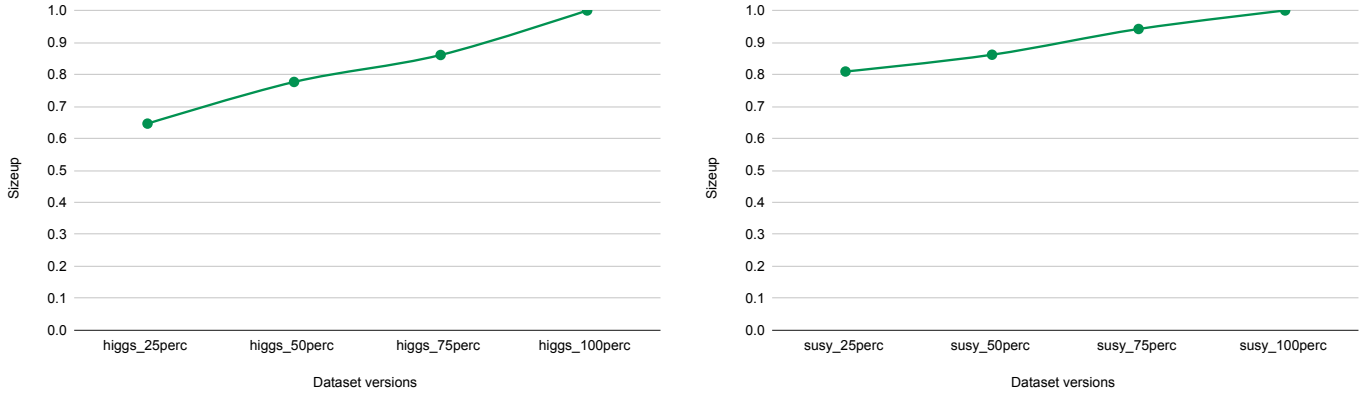
	GM			**SizeRed**			**DimRed**		
	Avg	**Diff**	**W/T/L**	**Avg**	**Diff**	**W/T/L**	**Avg**	**Diff**	**W/T/L**
BSL	0.7669	0.0077	4/15/2	56	42	19/1/1	71	1	7/14/0
FCNN_MR	0.6438	0.1308	14/5/2	71	27	18/1/2	0	72	21/0/0
FDR^2-BD	0.7746	–	–	98	–	–	72	–	–

It can be seen that our proposal is able to maintain the predictive quality even above the percentage stipulated by the PLT parameter with respect to baseline. In addition, the obtained reduction values regarding horizontality and verticality by applying FDR^2-BD are the greatest (98% and 72%, respectively), with significant differences with respect to BSL and FCNN_MR.

5.4. Scalability Evaluation

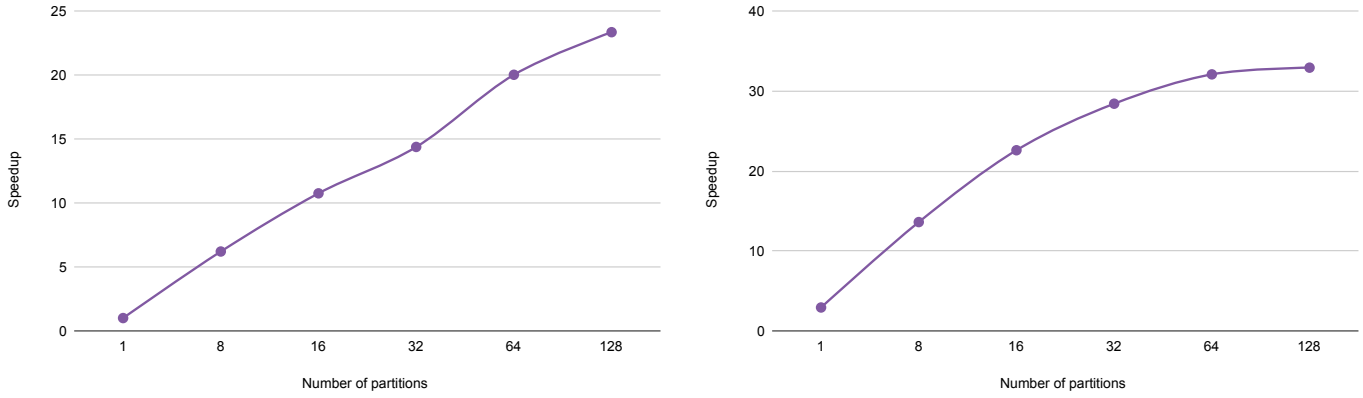
In Figures 3 and 4, the sizeup and speedup achieved are shown for two (`higgs` and `susy`) of the 25 experimental datasets in order to show the general behavior obtained. The selected datasets are chosen because they are widely used in big data classification articles. With the aim of assessing the sizeup, we have used our tool by varying the input size of the original dataset (higgs_100perc), generating three different versions of it according to their proportion of the original data size (higgs_75perc, higgs_50perc, higgs_25perc). Results show that when a dataset increases in size, the scaleup value grows, as is expected.

With respect to the speedup evaluation, our tool uses the original dataset, and the parallelism degree base is 8, doubling the previous parallelism degree value in each run. It is important to remark that a linear speedup is difficult to achieve due to the fact that increasing the number of partitions will introduce additional communication overhead. However, a linear trend in an increasing direction can be noticed as the number of partitions increases, meaning that this growing actually augments the parallelism degree of FDR^2-BD.



(**a**) Higgs dataset. (**b**) Susy dataset.

Figure 3. Sizeup.



(**a**) Higgs dataset. (**b**) Susy dataset.

Figure 4. Speedup.

6. Concluding Remarks and Future Work

In this paper, the benefits of data condensation by reducing tabular datasets in classification, particularly in the context of big data, are analyzed. A methodological proposal that conforms to a condensation parameter (reduction percentage and most important features, if any) recommendation tool is depicted. The design is simple and scalable based on a hyper-parametrization process fully transparent to the user, in which all data are taken into consideration by following a *k*-fold procedure. FDR^2-BD is able to analyze data condensation in a dual way (vertical and horizontal) by receiving the insight of the data expert by setting two simple conditions: a predictive quality threshold associated with the model obtained from the condensed set and a range of reduction percentages to check. Furthermore, its implementation uses fully optimized parallel operations and utilities provided by Apache Spark.

Throughout an extensive experimental study over 25 big datasets with different characteristics, it has been observed that the FDR^2-BD tool obtains reduction percentages above 95% in most cases. These values are far above state-of-the-art solutions such as FCNN_MR that barely reach 70%. The key of our proposal lies in a smart combination of FS to generate dense clusters of data and a uniform sampling reduction to keep only a few representative samples from each problem area. Therefore, the results show how the modeling process remains almost unchanged, with quality values around 1% of the baseline.

As future work, our intention is to test the quality of the proposal on multi-class problems, where the boundaries between concepts may be more difficult, as well as on

problems with imbalance. To this end, data complexity metrics can be included to guide the process in those areas of the space where a higher degree of redundancy is observed.

Author Contributions: Conceptualization, M.J.B. and A.F.; Data curation, M.J.B.; Formal analysis, M.J.B., M.N., and A.F.; Investigation, M.J.B. and A.F.; Methodology, M.J.B. and A.F.; Project administration, A.F.; Resources, A.F.; Software, M.J.B.; Supervision, M.N. and A.F.; Validation, M.J.B.; Visualization, M.J.B.; Writing—original draft, M.J.B. and A.F.; Writing—review and editing, M.N. and A.F. All authors will be informed about each step of manuscript processing including submission, revision, revision reminder, etc., via emails from our system or the assigned Assistant Editor. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found in the following repositories: (UCI Machine Learning [40], OpenML [41], Kaggle [42]).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MIFs	Most important features
RPR	Recommended percentage reduction
PLT	Prediction loss threshold

References

1. Wu, X.; Zhu, X.; Wu, G.; Ding, W. Data mining with big data. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 97–107. [CrossRef]
2. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. *J. Ind. Inf. Integr.* **2017**, *6*, 1–10. [CrossRef]
3. Bousdekis, A.; Lepenioti, K.; Apostolou, D.; Mentzas, G. A Review of Data-Driven Decision-Making Methods for Industry 4.0 Maintenance Applications. *Electronics* **2021**, *10*, 828. [CrossRef]
4. Gandomi, A.; Haider, M. Beyond the hype: Big data concepts, methods, and analytics. *Int. J. Inf. Manag.* **2015**, *35*, 137–144. [CrossRef]
5. Rashid, A.N.M.B. Access methods for Big Data: Current status and future directions. *EAI Endorsed Trans. Scalable Inf. Syst.* **2018**, *4*. [CrossRef]
6. Osman, A. A novel big data analytics framework for smart cities. *Future Gener. Comput. Syst.* **2019**, *91*, 620–633. [CrossRef]
7. Bruni, R.; Bianchi, G. Effective Classification Using a Small Training Set Based on Discretization and Statistical Analysis. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2349–2361. [CrossRef]
8. Bruni, R.; Bianchi, G. Website categorization: A formal approach and robustness analysis in the case of e-commerce detection. *Expert Syst. Appl.* **2020**, *142*, 113001. [CrossRef]
9. Maillo, J.; Triguero, I.; Herrera, F. Redundancy and Complexity Metrics for Big Data Classification: Towards Smart Data. *IEEE Access* **2020**, *8*, 87918–87928. [CrossRef]
10. Acampora, G.; Herrera, F.; Tortora, G.; Vitiello, A. A multi-objective evolutionary approach to training set selection for support vector machine. *Knowl. Based Syst.* **2018**, *147*, 94–108. [CrossRef]
11. Zaharia, M.; Xin, R.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.; et al. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
12. Liu, H.; Motoda, H. (Eds.) *Computational Methods of Feature Selection*; Chapman & Hall/CRC: Boca Raton, FL, USA, 2007; p. 419.
13. Liu, H.; Motoda, H. (Eds.) *Instance Selection and Construction for Data Mining*; Springer: Boston, MA, USA, 2001; [CrossRef]
14. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [CrossRef]
15. Rostami, M.; Berahmand, K.; Nasiri, E.; Forouzandeh, S. Review of swarm intelligence-based feature selection methods. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104210. [CrossRef]
16. Hariri, R.H.; Fredericks, E.M.; Bowers, K.M. Uncertainty in big data analytics: Survey, opportunities, and challenges. *J. Big Data* **2019**, *6*, 44. [CrossRef]
17. Luengo, J.; García-Gil, D.; Ramírez-Gallego, S.; López, S.G.; Herrera, F. *Big Data Preprocessing: Enabling Smart Data*; Springer International Publishing: Cham, Switzerland, 2020. [CrossRef]

18. Triguero, I.; Derrac, J.; Garcia, S.; Herrera, F. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 86–100. [CrossRef]
19. Garcia, S.; Derrac, J.; Cano, J.R.; Herrera, F. Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [CrossRef] [PubMed]
20. Pérez-Rodríguez, J.; Arroyo-Peña, A.G.; García-Pedrajas, N. Simultaneous instance and feature selection and weighting using evolutionary computation: Proposal and study. *Appl. Soft Comput.* **2015**, *37*, 416–443. [CrossRef]
21. Das, S.; Datta, S.; Chaudhuri, B.B. Handling data irregularities in classification: Foundations, trends, and future challenges. *Pattern Recognit.* **2018**, *81*, 674–693. [CrossRef]
22. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.; Amde, M.; Owen, S.; et al. MLlib: Machine Learning in Apache Spark. *J. Mach. Learn. Res.* **2016**, *17*, 1–7.
23. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
24. Ramírez-Gallego, S.; Fernández, A.; García, S.; Chen, M.; Herrera, F. Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. *Inf. Fusion* **2018**, *42*, 51–61. [CrossRef]
25. Angiulli, F. Fast Nearest Neighbor Condensation for Large Data Sets Classification. *IEEE Trans. Knowl. Data Eng.* **2007**, *19*, 1450–1464. [CrossRef]
26. García, S.; Cano, J.R.; Herrera, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognit.* **2008**, *41*, 2693–2709. [CrossRef]
27. Skalak, D.B. Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In Proceedings of the Eleventh International Conference, New Brunswick, NJ, USA, 10–13 July 1994; pp. 293–301.
28. García-Osorio, C.; de Haro-García, A.; García-Pedrajas, N. Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts. *Artif. Intell.* **2010**, *174*, 410–441. [CrossRef]
29. Liu, H.; Motoda, H. *Feature Selection for Knowledge Discovery and Data Mining*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1998; Volume 454, p. 214.
30. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
31. Saeys, Y.; Abeel, T.; Van de Peer, Y. Robust Feature Selection Using Ensemble Feature Selection Techniques. In *Machine Learning and Knowledge Discovery in Databases*; Lecture Notes in Computer Science; Daelemans, W., Goethals, B., Morik, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5212, pp. 313–325. [CrossRef]
32. Resende, P.A.A.; Drummond, A.C. A Survey of Random Forest Based Methods for Intrusion Detection Systems. *ACM Comput. Surv.* **2018**, *51*, 1–36. [CrossRef]
33. García-Gil, D.; Alcalde-Barros, A.; Luengo, J.; García, S.; Herrera, F. Big Data Preprocessing as the Bridge between Big Data and Smart Data: BigDaPSpark and BigDaPFlink Libraries. In Proceedings of the 4th International Conference on Internet of Things, Big Data and Security, Heraklion, Greece, 2–4 May 2019; pp. 324–331. [CrossRef]
34. Triguero, I.; García, S.; Herrera, F. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognit.* **2011**, *44*, 901–916. [CrossRef]
35. Triguero, I.; Peralta, D.; Bacardit, J.; García, S.; Herrera, F. MRPR: A MapReduce solution for prototype reduction in big data classification. *Neurocomputing* **2015**, *150*, 331–345. [CrossRef]
36. Arnaiz-González, A.; González-Rogel, A.; Díez-Pastor, J.F.; López-Nozal, C. MR-DIS: Democratic instance selection for big data by MapReduce. *Prog. Artif. Intell.* **2017**, *6*, 211–219. [CrossRef]
37. Triguero, I.; García-Gil, D.; Maillo, J.; Luengo, J.; García, S.; Herrera, F. Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1289. [CrossRef]
38. Ramirez-Gallego, S.; Mourino-Talin, H.; Martinez-Rego, D.; Bolon-Canedo, V.; Benitez, J.M.; Alonso-Betanzos, A.; Herrera, F. An Information Theory-Based Feature Selection Framework for Big Data Under Apache Spark. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 1441–1453. [CrossRef]
39. Fernandez, A.; del Rio, S.; Chawla, N.V.; Herrera, F. An Insight into Imbalanced Big Data Classification: Outcomes and Challenges. *Complex Intell. Syst.* **2017**, *3*, 105–120. [CrossRef]
40. Lichman, M. UCI Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 10 March 2021).
41. Vanschoren, J.; van Rijn, J.N.; Bischl, B.; Torgo, L. OpenML: Networked Science in Machine Learning. *SIGKDD Explor.* **2013**, *15*, 49–60. [CrossRef]
42. Kaggle Team. Kaggle—Datasets. 2021. Available online: https://www.kaggle.com/datasets (accessed on 10 March 2021).
43. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993.
44. Barandela, R.; Sánchez, J.S.; García, V.; Rangel, E. Strategies for learning in class imbalance problems. *Pattern Recognit.* **2003**, *36*, 849–851. [CrossRef]