Dealing with heterogeneity in the context of distributed feature selection for classification

José Luis Morillo-Salas · Verónica Bolón-Canedo · Amparo Alonso-Betanzos

the date of receipt and acceptance should be inserted later

Abstract Advances in the information technologies have greatly contributed to the advent of larger datasets. These datasets often come from distributed sites, but even so, their large size usually means they cannot be handled in a centralized manner. A possible solution to this problem is to distribute the data over several processors and combine the different results. We propose a methodology to distribute feature selection processes based on selecting relevant and discarding irrelevant features. This preprocessing step is essential for current high-dimensional sets, since it allows the input dimension to be reduced. We pay particular attention to the problem of data imbalance, which occurs because the original dataset is unbalanced or because the dataset becomes unbalanced after data partitioning. Most works approach unbalanced scenarios by oversampling, while our proposal tests both over and undersampling strategies. Experimental results demonstrate that our distributed approach to classification obtains comparable accuracy results to a centralized approach, while reducing computational time and efficiently dealing with data imbalance.

1 Introduction

Feature selection (FS) is a popular machine learning technique, whereby attributes that allow a problem to be clearly defined are selected, while irrelevant or redundant attributes are ignored [1]. Traditionally, an FS algorithm is applied in a centralized manner, i.e., a single selector is used to solve a given problem. However, in a big data scenario, data are often distributed, and a distributed learning approach

This research has been financially supported in part by the Spanish Ministerio de Economía y Competitividad (research projects TIN2015-65069-C2-1-R and PID2019-109238GB-C22), by European Union FEDER funds and by the Consellería de Industria of the Xunta de Galicia (research project GRC2014 /035). Financial support from the Xunta de Galicia (Centro singular de investigación de Galicia accreditation 2016-2019) and the European Union (European Regional Development Fund - ERDF), is gratefully acknowledged (research project ED431G/01).

Universidade da Coruña. CITIC. Grupo LIDIA. Campus de Elviña, 15071 A Coruña, España.
E-mail: ciamparo@udc.es · veronica.bolon@udc.es · jose.luis.morillo@udc.es

allows multiple subsets of data to be processed in sequence or concurrently. While there are several ways to distribute an FS task, the two most common ways are as follows: (i) an identical FS algorithm is run on data stored together in one very large dataset and distributed into several processors and the results are combined, or (ii) an identical FS algorithm is run on data stored in different datasets in different locations and the results are combined. Several works have contributed to the development of different approaches to distributed FS. In [2], the authors proposed a method for distributing the FS process by features, i.e., using a vertical distribution. A merging procedure then updates the feature subset according to the improvements achieved in classification accuracy. In testing over several microarray datasets, a considerable reduction was obtained in the execution time while performance was maintained or even improved with regard to the standard centralized approach. In [3], different ways of distributing the data (by features and by samples) were evaluated to determine to what extent it was possible to obtain similar results as those obtained using the whole dataset. In [4], distance correlation was used to evaluate the dependence between the classes and a given feature subset. A progressively refined feature subset model that uses a probabilistic representation was obtained through repeated extraction and evaluation steps. In [5], a different perspective for distribution was adopted that achieved distributed FS using data complexity instead of classification performance as a measure, obtaining optimal results while reducing the time required.

However, problems may appear when data is distributed into several processors, including a great imbalance between classes or even non-represented classes in some data subsets. The *class imbalance problem* occurs when a dataset is dominated by a class or classes with significantly more instances than the other classes. The learning algorithms would be biased toward the majority classes, since the rules to correctly predict instances are positively weighted in favour of the accuracy metric, whereas specific rules that predict examples from the minority classes are usually ignored. This means that minority class instances are more often misclassified than majority class instances [6]. To compensate for the imbalance problem, different general approaches can be followed [7]:

– **Weight adjustment**. This consists of assigning different weights to the instances of the dataset during the training phase, so as to prioritize less well represented classes in the classification [8].
– **Undersampling**. This consists of eliminating instances of better represented classes. Available methods are used to determine the instances that contribute less as candidates for elimination, such as duplicate instances or instances close to others [9–11].
– **Oversampling**. This consists of generating more instances in the less well represented classes, thus balancing the number of instances in all classes. This is the method we use in this research, as will be detailed below [12–14].

In recent years, several mixed solutions have been used that combine the basic approaches above [15]. Most of them employ the synthetic minority oversampling technique (SMOTE) to generate new samples for minority classes, followed by a cleaning step aimed at improving results. In [16] the authors propose a solution for imbalance consisting of combining the neighbour cleaning rule (NCL) and SMOTE techniques. NCL is used to remove outliers in the majority class while SMOTE is used to increase sample data in minority classes. Some authors have used combinations of over- and undersampling methods, e.g., [17], who describe the application of an editing technique based on the rough set theory (RST) as a new hybrid method with SMOTE aimed at improving results. Other combinations, such as the density-based spatial clustering of applications with noise (DBSCAN) algorithm for undersampling plus SMOTE for oversampling have performed well [18]. Other authors have combined the borderline synthetic minority over-sampling technique (BSMOTE) with data cleaning techniques such as Tomek links or Wilson's edited nearest neighbour rule (ENN) [19] to propose new support vector machine (SVM) classification algorithms for unbalanced datasets. The proposed BSMOTE-Tomek and BSMOTE-ENN hybrid preprocessing algorithms consist of first using SMOTE on minority samples in the neighbourhood of the borderline, and then removing redundant training samples to improve data utilization efficiency. Ensemble-based algorithms have also been popular in solving imbalanced learning [20], with some proposals adding different ways of adaptively selecting informative instances using cost-sensitive learning, among others [21].

Although a plethora of methods have separately tackled classification imbalance and FS, fewer methods have jointly considered FS and imbalance. Yang et al. [22] proposed iterative ensemble FS that combines filters and balanced sampling for multiclass classification of imbalanced microarray data using a one-versus-all (OVA) approach. Their experiments with different state-of-the-art filters – specifically, minimum-redundancy-maximum-relevance (MRMR) and fast correlation-based filter (FCBF) – and under- and oversampling alternatives show that their proposal behaves better than the filters alone for multiclass microarrays. Another work [23] shows that in scenarios such as fraud detection, FS methods are considerably affected by unbalanced datasets, so using undersampling strategies in the preprocessing step can greatly improve the obtained results. Maldonado et al. [24] developed a family of embedded methods for backward FS using SVMs, adapted to select attributes relevant to discrimination between classes under imbalanced data conditions. The model, applicable to high-dimensional datasets and tested on several very imbalanced microarray datasets, achieved better predictions with consistently fewer relevant features.

Scalability is a challenge for both sampling-based and FS techniques and in terms of both the number of samples and the data dimensionality. One way of tackling this problem is to distribute the data into several processors or nodes. This is the approach adopted in our research as described here. Some previous works, including [4, 24] have approached FS for unbalanced high-dimensional datasets, but their approach lacked the ability to tackle datasets with large numbers of samples.

In this work we present a methodology for distributed FS which simultaneously deals with the problem of heterogeneous subsets. The method can be applied to the whole dataset, distributed in order to cope with the high dimensionality, and also when the situation is distributed in origin. In testing our proposal, we used two alternatives: (i) forcing the partitions of the dataset to maintain the balance between the classes, and (ii) applying oversampling techniques when the imbalance is inevitable.

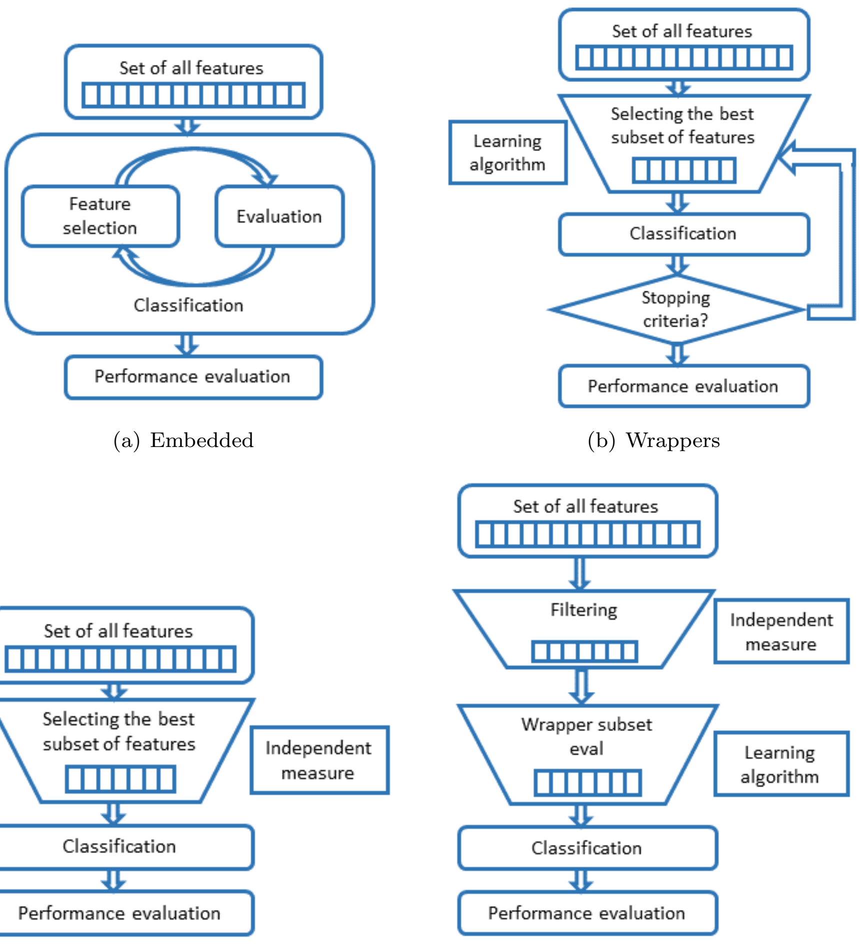
The remainder of the paper is organized as follows. Section 2 provides a short description of the main characteristics of FS methods. Section 3 describes the proposed methodology to improve performance in the context of distributed FS. Section 4 provides a description of the datasets, filters and classification methods used in our study. Section 5 shows **the results of applying the methodology to different datasets, distribution scenarios and resampling approach** and also analyses some research questions in detail. Finally, section 6 includes some final remarks and proposals for future research.

2 Background

Feature selection (FS), which is the process of identifying and eliminating as much irrelevant and redundant information as possible [25], is typically applied as a preprocessing step before classification, although it has also proven to be useful in other learning tasks such as regression, clustering and anomaly detection.

While different procedures are used to implement FS, all of them have a subset evaluation task and a stopping criterion. Focusing on subset evaluation, there exist four main methods, shown in Figure 1:

– **Embedded methods.** With these methods feature subset evaluation is not a separate task prior to classification, but is part of the classification algorithm itself. The application can be as simple as adding or removing features in response to prediction errors regarding new instances [26, 27] or as complex as adding routines to combine features into richer descriptions, as in ID3[28], C4.5[29] CART[30]. The advantages of these methods are that they include interaction with the classification model, are less computationally intensive than wrappers and detect dependencies between features. As a drawback, FS depends on the classifier.
– **Wrapper methods.** These use a previous task to evaluate the feature subset obtained in each step before the induction phase in a classifier. Wrapper methods use a predefined algorithm to evaluate the quality of a feature subset, e.g., SVM[31], employing the estimated accuracy of the resulting classifier as a metric. Their main advantages are that they interact with the classifier and detect features dependencies. However, they are computationally expensive and incur an overfitting risk (when a model is too closely fit to a limited set of data) and the selection depends on the classifier.
– **Filter methods.** Like wrapper methods, these perform FS in a prior step to classification. The main difference is that filters do not use an induction algorithm to evaluate subsets of features but an independent measure such as mutual information or correlation. Since they do not use a learning algorithm, they do not inherit any bias; thus, this model is computationally more efficient than both wrapper and embedded methods. Filters usually operate in two phases: (i) they rank features based on certain criteria, and (ii) they select features with the highest rankings to induce the classification model. The main advantages are their lower cost and their capacity to handle redundant features and to generalize. A drawback is that they do not interact with the classifier.
– **Hybrid methods.** Hybrid models [32] combine the advantages of filters and wrappers while trying to eliminate their respective limitations as much as possible. They usually combine two or more FS algorithms of different conceptual origins in a sequential manner. A typical example consists of first applying a less computationally expensive filter to remove some features and then using a more computationally expensive wrapper for a fine-tuning.



(a) Embedded (b) Wrappers

(c) Filter (d) Hybrid

Fig. 1 Type of FS methods depending on the relation between the evaluation of features and the classification method.

2.1 Related work

There are several ways to distribute an FS task [33]. Here we consider data collected together in one very large dataset and then distributed randomly or homogeneously (the percentages of each class are maintained in the partitions) in different datasets. An identical FS method can be run on each partition and the partial results are combined. The authors of this work have reported previous experience with methodologies to distribute the FS process. In [34, 35] we presented a parallel framework for scaling up FS, by partitioning the data vertically and horizontally, that could be used with different filter methods. The basic idea is to split the data (either by features or by samples) and then apply a filter at each partition, performing several rounds aimed at obtaining a stable set of features.

A merging procedure then combines the results into a single subset of relevant features. More recently, we have included complexity measures to assist the merging procedure [5]. As for dealing with unbalanced data, in a previous work we analysed the usefulness of data complexity measures to evaluate the behaviour of the SMOTE algorithm before and after applying FS to DNA microarray datasets [36].

The approach presented here is different than our previous works since it deals simultaneously with distributed FS and unbalanced data. As mentioned in the Introduction, there are other works in the literature that combine under and over sampling techniques to solve the imbalance problem. However, up to our knowledge, this is the first work that applies oversampling (SMOTE, in our case) also to the majority class, trying to compensate for the noise introduced by SMOTE in the minority class.

3 Proposed method

Before progressing any further, we define some key concepts related to the methodology:

Sample A dataset is defined as a collection of individual data, often called *samples*, *instances* or *patterns*.

Feature The information about each particular sample is given in the form of *features* or *attributes*.

Repetition The datasets need to be divided into training and test sets, for which we follow the standard procedure of randomly using 2/3 of samples for training and 1/3 of samples for testing. To ensure that division is not biased, this process, called *repetition*, is repeated a number of times.

Round Once the dataset is divided into training and test sets, we randomly distribute the training data to different nodes (also called packets). Again, to ensure that distribution is not biased, the process is repeated several times in what we call a *round*.

Packet Each training dataset is divided into a predefined number of nodes or *packets* of data to which oversampling and FS techniques are applied.

Our proposed methodology for dealing with both FS and data imbalance consists of three main steps: (i) partitioning of the data, (ii) application of an FS method to each partition, and (iii) combining the results. For each iteration (i.e., round) of our methodology, the first step is to randomly partition the training dataset into a number of disjoint packets. Repeating the process in several rounds r ensures that we have captured enough information for the combination step. We next apply an oversampling method (if the data are unbalanced) and proceed with the FS process in such a way that the features selected to be removed receive a vote. After the predefined number of rounds, features that have received votes above a certain threshold are removed, and the remaining feature subset is used in the training and test sets.

Note that, before starting, to ensure correct validation it is necessary to obtain training and test sets for each dataset to be tested. Therefore, we randomly divide each dataset into training and test sets and run several *repetitions* of this process (see Figure 2). After division, we start with the methodology as described below.

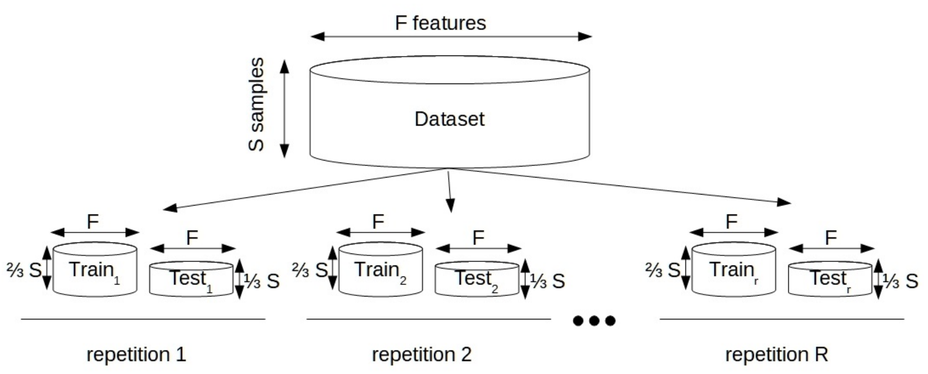


Fig. 2 Example of horizontal division of a dataset into train and test datasets, a number of times called "repetitions".

3.1 Step 1. Data partitioning

The first step (the core of this work) consists of splitting the data without replacement, assigning groups of n samples to each subset of data. This division can be done in one of two ways, horizontally or vertically [5,2]. Horizontal partitioning divides the dataset by instances while keeping all the features, whereas vertical partitioning divides the dataset by features while keeping all the instances. Horizontal partitioning is useful when the number of instances is much larger than the number of features, while vertical partitioning is preferred when the number of features is larger than the number of samples. The number p of packets (or subsets of data) depends on the features/instances ratio (ensuring a minimum of three packets, for our experiments, as we will see in subsequent sections). As mentioned above, this first step is repeated over several rounds (r, in this work set to five). Figure 3 illustrates the procedure of different rounds in which several packets are formed from the training set.

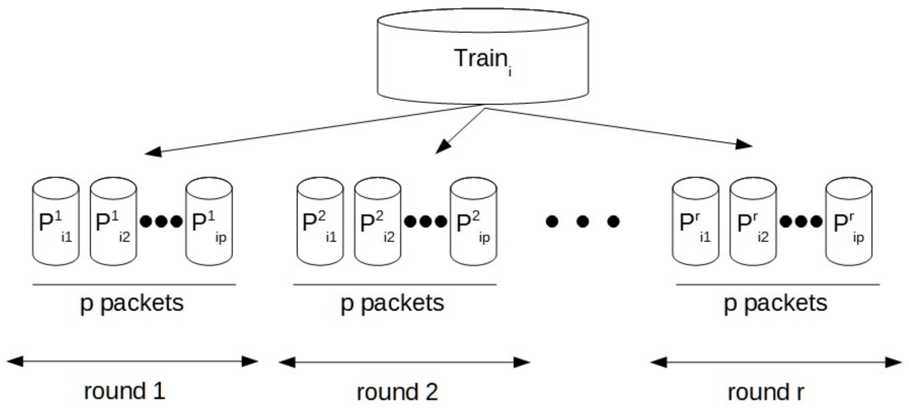


Fig. 3 Train dataset partition in packets (subsets of data).

One of two main approaches may be followed when partitioning the data: *random partitioning*, in data are distributed randomly in the different nodes, or *homogeneous partitioning*, in which the proportions of each class in the original dataset are maintained in each of the newly generated subsets (see an example in Figure 4, including the centralized scenario, in which all the data is together).

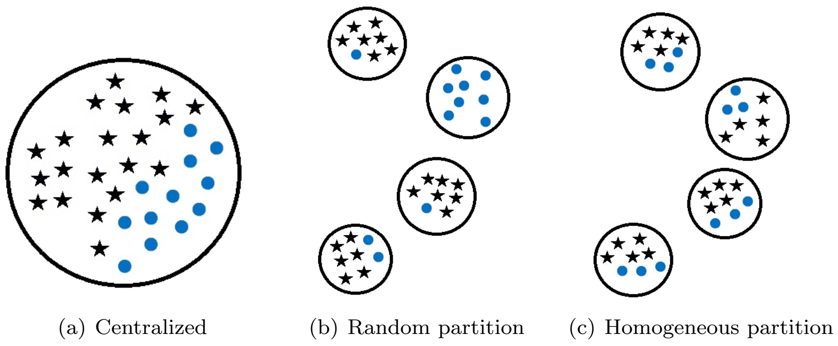

(a) Centralized (b) Random partition (c) Homogeneous partition

Fig. 4 Different approaches for partitioning the data

After partitioning, there are two situations in which new subsets of data may be unbalanced. The first one is when, for a balanced dataset, partitioning is random, leaving the majority of the samples belonging to the same class in some of the nodes by chance and thus producing data imbalance. The second situation is when the dataset is unbalanced before partitioning and the new subsets of data in each node continue to be unbalanced. In these two cases, applying the SMOTE oversampling method [12] adds synthetic minority class examples to the original dataset until the class distribution becomes balanced. The SMOTE algorithm generates the synthetic minority class examples using original minority class examples in the following way: it searches for the k nearest neighbours of the minority class sample to be used as the basis for the new synthetic sample. Next, in the segment that unites the minority class sample with one or all of its neighbours, a synthetic sample is randomly taken and is added to the new oversampled dataset. Fig. 5 shows an example of SMOTE synthetic sample generation in the minority class.

Note that SMOTE is typically used to generate synthetic samples in minority classes only, whereas we propose to also generate synthetic samples in the majority class, as a means to improve the method and its results.

3.2 Step 2. FS application

An FS method is applied to each packet p to reduce the dimensionality of the problem. The features selected to be removed receive votes, a new round is performed leading to a new partitioning of the dataset and other iterations of voting are completed, until the predefined number of rounds is reached (set to five in this work). The set of all rounds can be seen as an *ensemble*, and so a combination of the partial results is needed, as detailed in the next step.

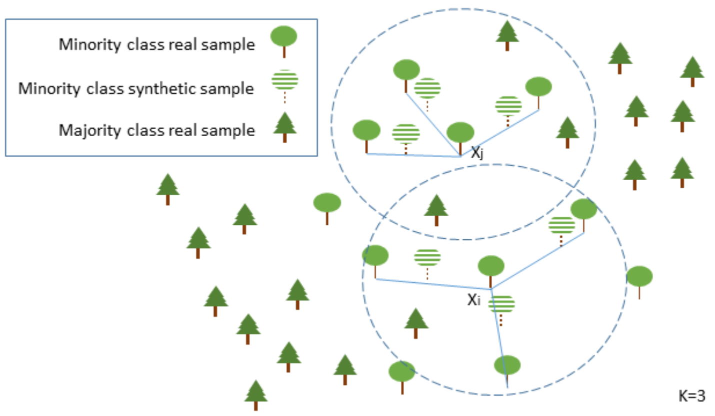

Minority class real sample
Minority class synthetic sample
Majority class real sample
Xj
Xi
K=3

Fig. 5 SMOTE in minority class.

3.3 Step 3: Results combination

The features that receive votes above a predefined threshold have to be removed. Thus, a unique set of features is obtained to train a classifier C and to test its performance on a new set of samples (test dataset). As described in [34,35], choosing the threshold of votes is not an easy matter, since it depends on each dataset. Following the recommendations in [37], selecting the number of votes must take into account two criteria: the training classification error and the percentage of features retained. Both values should be minimized to the extent possible, in accordance with minimization of the following criterion:

$$e[v] \leftarrow \gamma \times error + (1 - \gamma) \times featPercentage$$

where a term γ , introduced to measure the relative importance of both values, is set to $\gamma = 0.75$ as suggested in [37], giving more importance to the classification error. Note that the maximum number of votes is the number of rounds r when dividing by features and $r * p$ when dividing by samples, so all the threshold values from 1 to r or 1 to $r * p$ are evaluated. The features with votes above the obtained threshold are removed from the final subset of features. See an example in Figure 6.

After obtaining the final subset of features to be selected from the whole training set, SMOTE is applied again if the dataset is still unbalanced and a classifier is then applied.

To sum up, a pseudo-code for the proposed methodology is given in Algorithm 1.

4 Experimental setup

The goal of the experimental study is two-fold. First, we aim to check if our distributed FS methodology obtains comparable results to those achieved by a centralized methodology. Second, we want to test if oversampling methods can

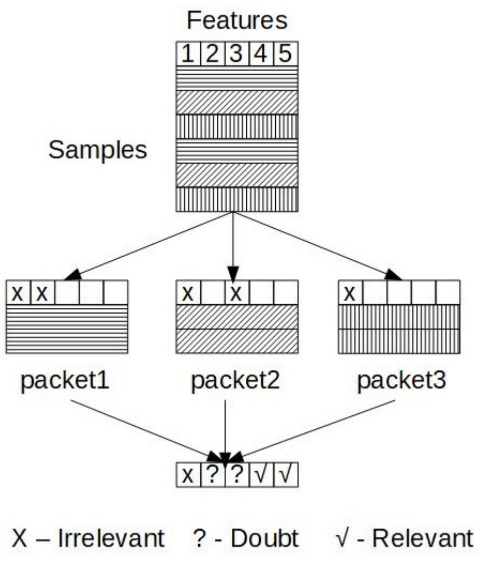


Fig. 6 This is an example in which we have our data divided by samples (in 3 packets), so we have the same set of features in each packet (all the original features, in this case 5 features) and feature 1 was marked as not relevant for packets 1, 2 and 3; feature 2 was marked as not relevant for packet 1, feature 3 was marked as not relevant for packet 2, and features 4 and 5 are not marked as irrelevant for any packet. Our intuition says that, in this case, features 4 and 5 should be relevant and feature 1 should be not relevant, and we will have doubts about features 2 and 3 (this is why threshold of votes is needed).

improve performance results when FS is applied in a distributed environment and to check the effect of creating some synthetic samples from the majority class.

Described below are the datasets chosen for testing the distributed approaches, as well as the filters to perform the FS process. To test the adequacy of our distributed approach, we use four well-known supervised classifiers, of different conceptual origins. All the classifiers and filters are executed using the Weka tool,[1] using the default values for their parameters. Experiments are performed on an Intel®Core^{TM}i5-3470 CPU @ 3.20GHz with RAM 4 GB. The code and the results of the experiments can be found in GitHub.[2].

4.1 Datasets

An important issue to correctly validate the proposed methodology is to choose the appropriate benchmark of datasets. Because we present a distributed methodology which can partition the data both by filters and by samples, we consider it necessary to use datasets with high numbers of samples (standard datasets) and others with high numbers of features (microarray datasets). The standard datasets are seven datasets as described in Table 1, available for download from the UCI

[1] `https://www.cs.waikato.ac.nz/ml/weka/`
[2] `https://github.com/jlmorillo/Heterogeneity_distributed_features`

Algorithm 1: Pseudo-code for proposed methodology

```
initialize the vector of votes to 0
for each repetition $rp$ do
    split dataset into train $d_{tr}$ (2/3 of samples) and test $d_{te}$ (1/3 of samples) sets
    for each round $r$ do
        split dataset $d_{tr}$ randomly or maintaining the class proportions into
          disjoint subsets of data
        for each subset (packet) of data $p$ do
            if data is unbalanced then
                apply SMOTE in minority class
                apply SMOTE in majority class
            end
            apply a feature selection algorithm
            increment one vote for each feature to be removed
        end
    end
    remove features which number of votes is above threshold in $d_{tr}$ and $d_{te}$
    classify with obtained subset of features trained on $d_{tr}$ and applying SMOTE
      if data is unbalanced
end
```

Machine Learning Repository[3]. The microarray datasets, from cancer classification research [2], are five datasets as described in Table 2. Those particular datasets were chosen because they represent different sizes, different numbers and different distribution of classes, and are also widely used in other research, including in our own previous work [5,2].

In the standard datasets, partitioning in packets is done horizontally by samples, while in microarray datasets, partitioning is done vertically by features. In the latter case, homogeneous partitioning is not applied because all the packets have the same class distribution (they have all the samples). In both cases, splitting is made without repetition, so each element is in only one packet.

To determine the number of packets in each dataset (we set three as the minimum because we consider that fewer than three packets/nodes is not worthwhile for a distributed methodology), we calculate proportions, computed as the number of samples divided by the number of features for the horizontal split, and the number of features divided by the number of samples for the vertical split. If the proportion is greater than or equal to three packets, the process stops, otherwise the denominator (number of elements in each packet) is increased by multiplying it successively by a factor from a predetermined array [100, 50, 20, 10, 5 ,2 ,1, 0.5, 0.25] until the condition of three or more packets is met. The pseudo-code of this process is given in Algorithm 2 and an example is shown in Figure 7.

Tables 1 and 2 show, for each dataset, the number of samples, number of features, number of different classes, percentage for the top (one) majority class, percentage for the bottom (one) minority class, imbalance ratio (IR) as the majority class percentage divided by the minority class percentage, and number of packets in the distributed split. We consider that a dataset is unbalanced when IR > 1.

[3] `http://archive.ics.uci.edu/ml/index.php`

Algorithm 2: Pseudo-code for determining the number of packets for each dataset

```
partition=by instances (by default)
proportion = nInstances / nFeatures
if (proportion > 10000) and (nInstances / 10000) ≥ 3 then
    ndatapartition = 10000
else
    if (proportion > 1000) and (nInstances / 1000) ≥ 3 then
        ndatapartition = 1000
    else
        initial packets of 100 times nFeatures. In each iteration, the vector is
          traversed and size of packets decrease until the condition of having at
          least three packages is reached or until the vector is finished
        vector=[100, 50, 20, 10, 5 ,2 ,1, 0.5, 0.25]
        while vector do
            if nInstances/(nFeatures*vector(i)) ≥ 3 then
                ndatapartition=nFeatures*vector(i)
                return
            end
        end
        if condition of 3 packets is not reached, partition by features is made
        partition = by features
        vector=[100, 50, 20, 10, 5 ,2 ,1, 0.5, 0.25]
        while vector do
            if nFeatures/(nInstances*vector(i)) ≥ 3 then
                ndatapartition=nInstances*vector(i)
                return
            end
        end
        if the 3 packet condition is also not reached with the feature partition, it is
          considered the number of instances
        ndatapartition = nInstances;
    end
end
```

Dataset	Samples	Features	Classes	% maj. class	% min. class	IR	Packets
Connect4	67557	42	3	65.83	9.54	6.90	45
Isolet	7797	617	26	3.85	3.50	1.1	3
Musk2	6598	168	2	84.6	15.4	5.49	5
Nomao	34465	120	2	71.4	28.6	2.49	3
Ozone	2536	72	2	97.12	2.88	33.72	4
Spambase	4601	57	2	60.6	39.4	1.53	5
Weight	4024	152	5	34	2.78	12.23	3

Table 1 Standard datasets characteristics.

4.2 Feature Selection filters

From the different categories of FS methods reviewed in section 2, for this work we use filters because they are less computationally expensive and are independent of the induction algorithms. We used several popular filter methods: two ranker

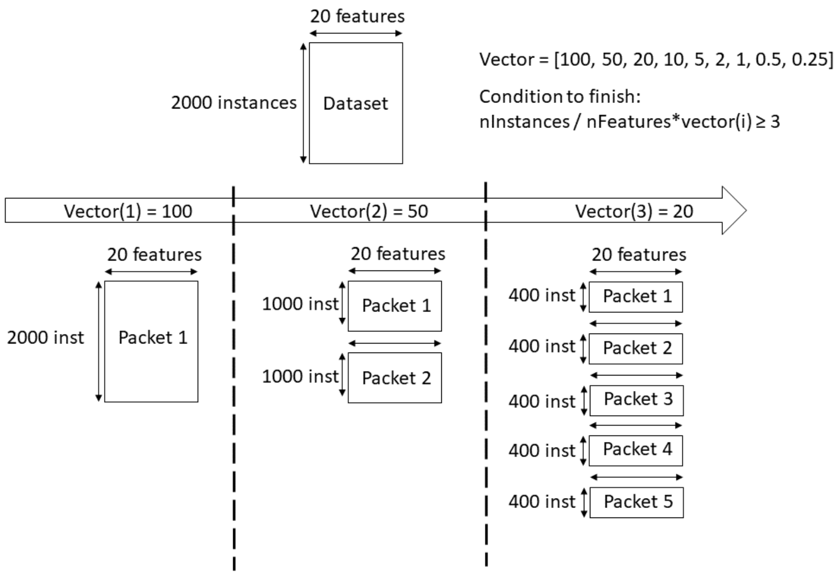


Fig. 7 Algorithm 2 example for determining the number of packets in a partition by instances. The dataset contain 2000 instances and 20 features. The process is iterated until the dataset is divided into three or more packets. In each iteration, the size of packets decreases.

Dataset	Samples	Features	Classes	% maj. class	% min. class	IR	Packets
Brain	21	12625	2	67	33	2.03	9
CNS	60	7129	2	75	25	3	3
Colon	62	2000	2	65	35	1.85	4
Gli85	85	22283	2	69	31	2.22	3
Ovarian	253	15154	2	64	36	1.77	4

Table 2 Microarray datasets characteristics.

algorithms (*InfoGain* and *ReliefF*), a subset method based on correlation between features (*CFS*) and, finally, a subset method based on consistency (*Consistency*). We thus have representative filters based on mutual information, distance, correlation and consistency. All these filters are available in the popular Weka tool and are widely used by FS researchers.

– **InfoGain**: *Information Gain*[38] is one of the best known FS methods. It is a univariate ranker, i.e., it evaluates only one feature at a time. It is based on the information gain equation by Quinlan [29]:

$$IG(A|B) = H(A) - H(A|B) \tag{1}$$

where H is the entropy, an uncertainty measure of a random feature associated with information theory [39]. InfoGain computes the information gain (IG) of each feature with respect to the class. This method obtains an ordered classification of all features and selects the first δ features, where δ is a defined threshold, set in this work, to the number of features selected by CFS.

– **ReliefF**: ReliefF [40] is another very commonly used ranker algorithm in FS. ReliefF derives from Relief [41], which evaluates features by randomly selecting one instance, searching for the nearest neighbour for the same class and the opposite class (Relief is valid only in binary classification) and evaluating the feature values with respect to another closer neighbour's feature values. The idea is that relevant features have similar values when they belong to the same class and different values when they belong to different classes. ReliefF improves Relief by handling multiclass and noisy data. Again we used as a threshold the number of features selected by CFS.
– **CFS**: Correlation-based Feature Selection [25,42] is a FS method that selects the best feature subset instead of returning a ranking of all the features. The goal is to obtain the feature subset that is best correlated with the class containing features that are not correlated among themselves. Irrelevant features are ignored because they have little correlation with the class. Redundant features are also ignored because they have high correlation with one or more features in the candidate subset of features. Each feature is evaluated individually and so CFS cannot identify strong relationships, like those reflected, for example, in parity problems.
– **Consistency**: Like CFS, this FS method [43] selects an optimal feature subset, using consistency interpreted as zero inconsistency. The inconsistency rate compares feature subsets according to the level of consistency with the class. The consistency measure is monotonic, is always equal or greater for each iteration, is fast, is capable of removing redundant and/or irrelevant features and is capable of handling some noise. Different feature search strategies can be followed (exhaustive, complete, heuristic, probabilistic or hybrid), depending on the number of relevant/irrelevant/redundant/total features.

4.3 Classifiers

Classification is an important machine learning task. Essentially it involves dividing up elements so that each one is assigned to one of a number of mutually exhaustive and exclusive categories known as *classes* [44]. Of the several classification algorithms in the literature, for this study we use four classifiers from different families: two linear (Naïve-Bayes and SVM using a linear kernel) and two non-linear (C4.5 and IB1).

– **Naïve-Bayes** uses the Bayes rule defined by Reverend Thomas Bayes (1702–1761) [44]. Bayesian learning can be seen as the process of finding the most probable hypothesis given a set of training examples and *a priori* knowledge about the probability of each hypothesis. The application of Bayes theorem to classification consists of calculating the hypothesis with a higher *a posteriori* probability. It is said to be naïve because it assumes that the attributes are conditionally independent on each other given the class. Advantages are its simplicity, efficiency and robustness to noise and irrelevant features. Besides, it only needs a few instances to estimate the classification parameters. As disadvantages, this method requires *a priori* knowledge and is computationally expensive.
– **SVM** [45] is a type of classifier based on statistical learning techniques. The original idea of the algorithm is to transform a set of input features X into a

set of Y vectors of a higher dimension (even of infinite dimension) in which the problem can be solved linearly, i.e., linear models are used by SVM to implement nonlinear class boundaries. The SVM algorithm performs very well, partly because it allows the construction of flexible decision boundaries and also due to its good generalization capacity. A weak point is its high computational cost, due to the fact that it needs to handle a large number of coefficients which greatly increase when the input set is large.

– **C4.5** was developed by Quinlan [29] as an extension of Iterative Dicotomiser 3 (ID3), based on decision trees. To construct the best tree, C4.5 iterates using the feature with the best gain rate, i.e., the relation between the *information gain* of a feature and class and the *entropy* of the feature. Each feature value makes a new branch where instances are distributed until all the instances have the same class (leaf). To overcome a possible problem of overfitting with this strategy, C4.5 uses *post pruning* after constructing the tree, whereby a *confidence factor* deletes branches with little accuracy gain. In this study the factor is set to 0.5. Afterwards, if so desired, the tree can be converted to easily understandable rules.
– **IB1** is an algorithm part of the Instance Based Learning (IBL) family. In IBL the training examples are stored verbatim while a distance function is used to determine which member of the training set is closest to an unknown test instance. Once the nearest training instance has been located, its class is predicted for the test instance. The only remaining problem is defining the distance function, which is quite easy, particularly if the attributes are numeric [46]. IB1 is limited to looking for the case stored closest to the example to be classified, generally using an Euclidean distance metric. Like Naïve-Bayes, it is simple and works very well, but may be slow because of the need to store all training instances.

4.4 Evaluation metrics

The metrics used to evaluate the performance of the different models are classification accuracy, kappa statistic, packet filter time and classification time.

		Actual class	
		p	n
Predicted class	p	True positive (TP)	False positive (FP)
	n	False negative (FN)	True negative (TN)
Total		P	N

Table 3 Confusion matrix.

From the confusion matrix shown in Table 3, classification accuracy is defined as the percentage of instances that are classified correctly:

$$Accuracy = \frac{TP}{TP + FP} \tag{2}$$

Kappa statistic [47] is an index that compares the agreement between several experts, considering agreement obtained by chance: values close to 1 indicate total agreement (the ideal value) and values close to 0 indicate agreement similar to that obtained by chance:

$$Kappa = \frac{ObservedAgreement - RandomAgreement}{1 - RandomAgreement} \tag{3}$$

where

$$ObservedAgreement = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

and

$$RandomAgreement = \frac{(TN * FP) * (TN + FN) + (FN + TP) * (FP + PN)}{Total * Total} \tag{5}$$

Classification accuracy and the kappa statistic value are obtained from the results of the classification process (line 13 of Algorithm 1), the filtering time by packet, measured in seconds, is obtained from the sum of the oversampling time for FS plus the FS method application time for each partition (lines 6 to 11 of Algorithm 1) and, finally, classification time, measured in seconds, is the summed time to remove features with votes above the threshold, applying SMOTE if data is unbalanced (lines 12 and 13 of Algorithm 1). The reason to include the kappa value is because it assesses the quality of the learning, taking into account situations in which the dataset is unbalanced and the classifier correctly learns majority class instances but systematically misclassifies minority class instances. In the case of accuracy and kappa, the higher the value, the better the results, whereas for filter time by packet and classification time, low values are preferred.

5 Experimental results and Discussion

To evaluate our methodology we compare the results of a centralized scenario with the distributed (random and homogeneous, see Section 3.1) approaches. The results for each scenario are also compared for use of SMOTE in the different approaches. For the standard datasets, the percentage of SMOTE applied in the minority class is [0, 100, 300, 600 and Auto (i.e., the option to balance minority and majority classes)] and in the majority class is [0,20,40,100]. For the microarray datasets with small samples, the percentage in both classes is [0,20,40,100] plus the Auto option for the minority class. Random undersampling (RUS) was tested applying SMOTE on the minority class for percentages of [0, 100, 300, 600] for standard datasets and [0, 20, 40, 100] for microarray datasets with RUS in the majority class to balance both classes. Because of the high number of combinations being tested and the different metrics employed, the results of the experiments are difficult to analyse exhaustively; however, below we analyse results with a view to answer certain research questions.

5.1 Exploring the accuracy of the different approaches

A summary of accuracy results is summarized in Table 4, which shows the number of best cases in each distribution and the SMOTE approach. Overall, we can see that the centralized distribution functions better for the microarrays, while the

Datasets	SMOTE	Centralized	Random	Homogeneous
Standard	Without	27 (37)	35 (53)	**56** (**69**)
	Minority	21 (45)	29 (47)	**67** (**54**)
	Both	29 (**49**)	25 (44)	**62** (48)
	Min.-RUS	24 (30)	34 (44)	**54** (**66**)
Microarray	Without	**71** (55)	12 (55)	
	Minority	**69** (52)	11 (**56**)	
	Both	**71** (50)	9 (**57**)	
	Min.-RUS	**77** (56)	3 (**55**)	

Table 4 From a total of 112 combinations of 4 classifiers, 4 filters and 7 standard datasets and 80 combinations of 4 classifiers, 4 filters and 5 microarray datasets, it is shown for each type of dataset and for each SMOTE scenario, the number of combinations of dataset, filter and classifier with the best value in accuracy average and kappa average (in brackets) for a distribution with respect to the other distributions. Combinations with the same maximum value across multiple distributions are aggregated into the count for each distribution, so the sum of combinations can be greater than the total number of combinations. The highest average values for each type of dataset and SMOTE scenario is in bold font and the highest value for each type of dataset regardless of the SMOTE scenario is also underlined.

distributed approach is a good option for standard datasets, and especially the homogeneous distribution.

Tables A1, A2 and A3 in the Appendix show more specific results in the form of detailed accuracy results for data distributions in a very unbalanced standard dataset (Musk2 - Random), a balanced standard dataset (Isolet - Homogeneous) and a microarray dataset (Brain - Random), respectively. These tables show the prediction accuracy for the four classifiers and four filters, comparing different levels of oversampling (both in minority and majority classes) with the option to apply oversampling in the minority class and random undersampling in the majority class. In the case of Musk2 (Table A1), there are some cases for which the highest accuracy is obtained for the combination of filter, classifier and SMOTE percentage when no oversampling is applied; however, the highest accuracy for all combinations (97.36% with random distribution, Consistency filter and C4.5 classifier) is obtained when SMOTE is applied in both minority and majority classes. As for Isolet (Table A2), there is no combination of filter and classifier for which not applying SMOTE performed better than applying SMOTE; in particular, the highest accuracy for all combinations (96.01%) is obtained for CFS and SVM, again applying SMOTE in both minority and majority classes. Finally, for the Brain dataset (Table A3), the highest accuracy (91.43%, with Consistency filter and C4.5 classifier) is also achieved after applying SMOTE in both minority and majority classes. Moreover, there is no case in which not using SMOTE was better than using SMOTE, for all combinations of FS method and classifier.

In the interest of brevity, the rest of the results for datasets are summarized in Tables A4 to A9 in the Appendix (for complete results see please GitHub repository[4] classification accuracy (Tables A4 and A5), kappa (Tables A6 and A7) and filter time by packet in seconds (Tables A8 and A9).

Recall that for all measures except filter time by packet higher values are better. For each dataset and evaluation metric, the best value for each scenario is

[4] `https://github.com/jlmorillo/Heterogeneity_distributed_features`). In the tables for standard and microarray datasets in the Appendix, the following information is provided: mean and standard deviation (top row), maximum value (second row) and (in the lowest row) the combination that obtained the maximum value for:

indicated in bold font and the best value of all scenarios is indicated by underlining. Remember that, for the microarray datasets, homogeneous partitioning is not applied because the split in data is by features.

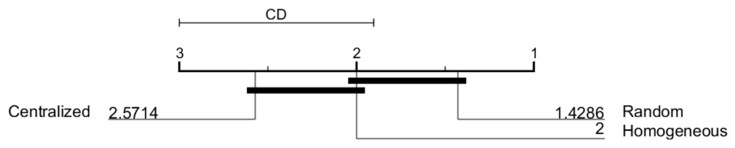
These results are analysed in more detail below, but, in general it can be observed as follows: (i) for accuracy, the distributed scenarios obtain comparable results to the centralized scenario (in some cases even improving on them, probably because of the application of a divide-and-conquer strategy, since, in some cases, the result obtained by the learner is more accurate if focused on a local region of the data), (ii) for kappa, SMOTE is useful in improving performance for unbalanced datasets, and (iii) for packet time, the time used was significantly reduced for the distributed scenarios.

To get more insights on the results, the statistical Friedman test [48] is used to check for significant differences between distributions and between SMOTE scenarios. Used for the Friedman test are mean results for all combinations of dataset, filter, classifier, distribution approach and SMOTE percentages. Executed for the level of significance $\alpha = 0.05$, if $p < 0.05$, the null hypothesis is rejected and at least one of the variables (distribution or scenario) is statistically significant, whereas if $p > 0.05$, the variables are not significantly different. To graphically summarize these statistical accuracy and kappa comparisons for distributions and scenarios in standard and microarray datasets, we use a critical difference (CD) diagram [49]. The axis of a CD diagram represents the average rank of best values of variables in each combination, with the lowest ranks (1 is the best rank) to the right side of the figure. Groups of variables that are not significantly different are connected by a bold line, while the CD value is shown above the graph.

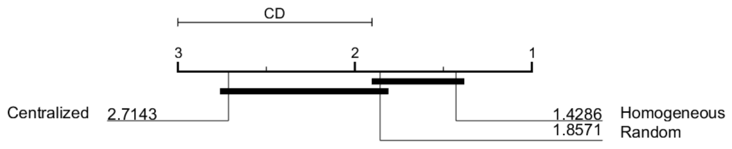
Figures 8 to 14 show statistical comparisons of accuracy and kappa for different distributions and SMOTE applications, while Figure 11 shows a comparison of different SMOTE scenarios. In addition, Figures 12 and 13 show detailed CD diagrams with different combinations of SMOTE percentages in minority and majority classes. Below we explain the unexpected result obtained by considering only the average value for the accuracy ranking.

5.2 Exploring the distributed partitioning consequences for classification

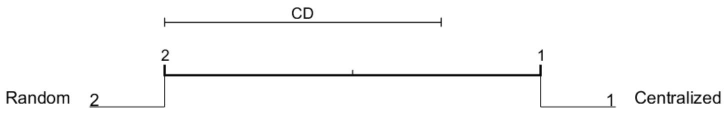
Comparing the two (random and homogeneous) distributed approaches with the centralized scenario, for the standard datasets, in general, the distributed approaches have better accuracy averages, as can be seen in Tables 4 and A4 and Figures 8(a), 9(a) and 14(a). The homogeneous approach seems to achieve higher accuracy and more stable results (better kappa values), as illustrated in Tables 4 and A6 and Figures 9(b), 9(b) and 14(b). As shown in Table A8, the distributed approaches considerably reduce the time required to filter the best features; however, there is an additional cost in calculating a votes threshold to remove irrelevant features and this increases the total classification time comparatively to the centralized approach (as shown in Table 5). The random and homogeneous partitioning approaches are a good solution to reducing computation time (especially if executed concurrently) without degrading classification performance. For the microarray datasets in some cases, classification accuracy slightly worsens with respect to centralized distribution after applying random distribution (Tables 4 and A5 and Figures 8(c), 9(c) and 14(c)), e.g., for Ovarian and Gli 85. This results from the fact that a high accuracy value is obtained in the centralized distribution, probably due to overfitting in datasets with few instances and many features. On the other hand, on applying a vertical split in random distribution, it is possible that average accuracy is reduced when features are marked as relevant/irrelevant in one but not in another packet (because of feature interactions). On the positive side, in random distribution, computational time is considerably reduced with respect to the centralized approach, since for this distribution, the reduction in filtering time (Table A8) is much greater than the increase in classification time (Table 5), while the kappa value is maintained or even increased (Tables 4 and A7 and Figures 8(d), 9(d) and 14(d)), thus making the solution more stable.



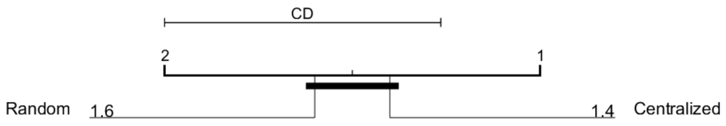
(a) Accuracy - Standard datasets *(p=0.101)*



(b) Kappa - Standard datasets *(p=0.049)*



(c) Accuracy - Microarray datasets *(p=0.025)*

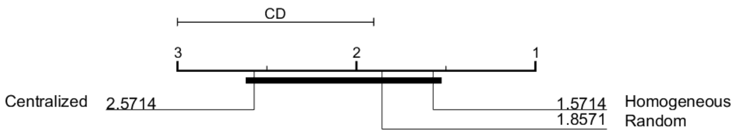


(d) Kappa - Microarray datasets *(p=0.654)*

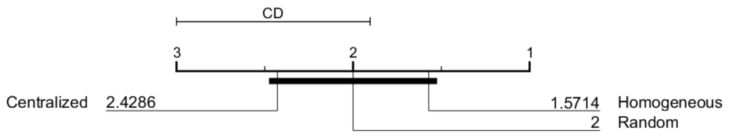
Fig. 8 Statistical comparison of distributions without applying SMOTE. CD diagram represent the average ranks of best values of variables in each combination (1 is the best). The groups of variables that are not significantly different are connected by a bold line.

5.3 SMOTE use: minority class only vs. both minority and majority classes

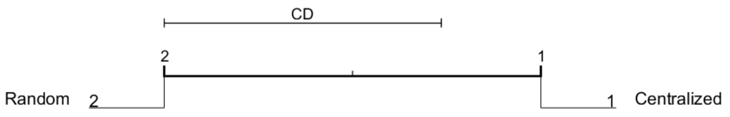
To analyse the advantages of using versus not using SMOTE, we compare the accuracy results in Tables A4 and A5. In standard datasets, except for Spambase and Weight, the best average accuracy is obtained without applying SMOTE. Figure 11 shows the ranking of best-case SMOTE scenarios for the different combinations, while Figure 11(a) shows the ranking for accuracy in standard datasets. Not



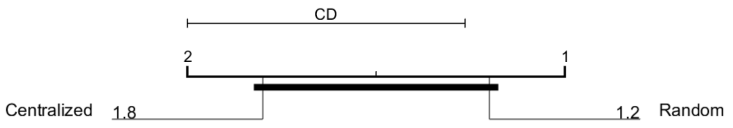
(a) Accuracy - Standard datasets *(p=0.156)*



(b) Kappa - Standard datasets *(p=0.276)*



(c) Accuracy - Microarray datasets *(p=0.025)*



(d) Kappa - Microarray datasets *(p=0.179)*

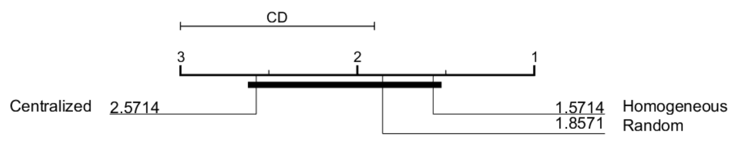
Fig. 9 Statistical comparison of distributions applying SMOTE in minority class.

Scenario	Standard datasets			Microarray datasets	
	Centralized	Random	Homogeneous	Centralized	Random
Without SMOTE	28.65	38.03	46.03	3.70	7.75
SMOTE Minority	69.53	81.04	105.59	4.10	8.71
SMOTE Both	164.45	164.73	193.36	4.43	9.34
SMOTE Min. - RUS Maj.	79.21	80.99	81.57	5.10	9.29

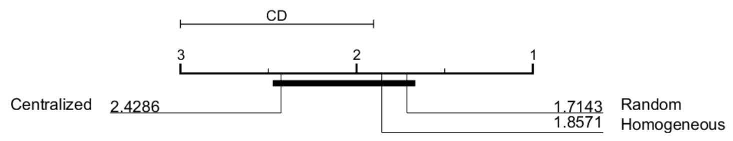
Table 5 Average time in seconds needed to remove features with votes above the threshold and, if data is unbalanced, applying SMOTE and proceeding with classification.

using SMOTE obtains the best results, although those results are not significantly different from those for SMOTE applied in both minority and majority classes. Decomposing the CD diagram with the different combinations of SMOTE in minority class and majority class, as in Figure 12, gives more insights about these results; it is easily seen that distributions using SMOTE in both classes obtain a better ranking than not using SMOTE in any class. Note that combinations with high percentages of SMOTE (e.g., 600) are usually in the last positions of the ranking (left in the figure).

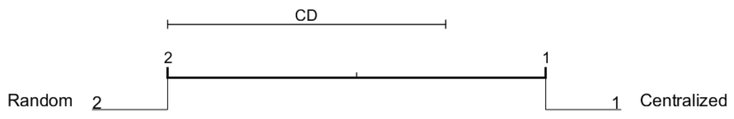
This difference in the ranking when SMOTE is applied in different percentages explains why the average results of applying SMOTE are poorer than those without SMOTE.



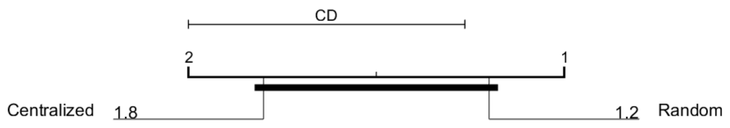
(a) Accuracy - Standard datasets *(p=0.156)*



(b) Kappa - Standard datasets *(p=0.367)*



(c) Accuracy - Microarray datasets *(p=0.025)*



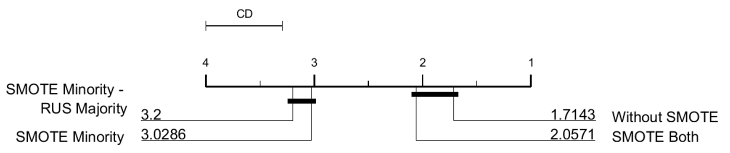
(d) Kappa - Microarray datasets *(p=0.179)*

Fig. 10 Statistical comparison of distributions applying SMOTE in both minority and majority classes.

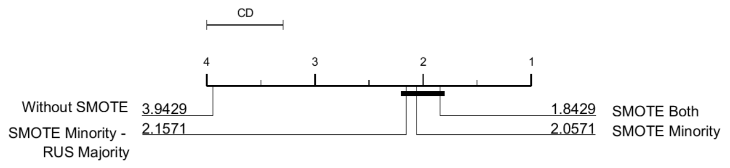
For example, focusing on the results in Table A3, the average value obtained without SMOTE is 78.57, compared to an average value of 77.35 for the rest of combinations of filter-classifier-SMOTE minority class-SMOTE majority class for the Brain dataset and random distribution, which is apparently a poorer result. However, if we consider the maximum value for each option, using SMOTE usually obtains better results than not using SMOTE (even 100 in some cases). For the microarray datasets, the behaviour is similar, as can be seen in Figure 13.

Comparing the accuracy percentages for SMOTE used in the minority class and SMOTE used in both the minority and majority classes (Tables A4 and A5 and Figures 12 and 13), SMOTE for both classes usually obtains good results for both types of datasets. For standard datasets, a percentage of 100 in the minority class generally offers better results, so it is not necessary to increase this percentage to improve the method, as this may even deteriorate performance. This percentage in the minority class combined with different percentages of SMOTE in the majority class yields a good ranking position for all distributions, while a high proportion of SMOTE in the minority class or the balanced Auto option yield poorer rankings. In contrasts, for microarray datasets, the Auto option yields a better ranking that is the same as the higher values for SMOTE, even though the trend is not as clear as for the standard datasets.

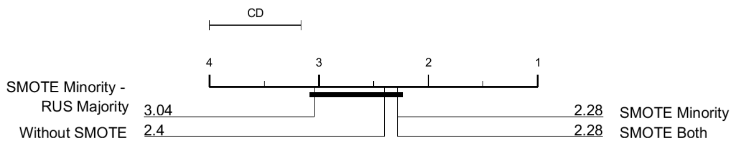
The kappa value (Tables A6 and A7 and Figures 11(b) and 11(d)) is also higher when SMOTE is applied in both classes, and sometimes even in the minority class



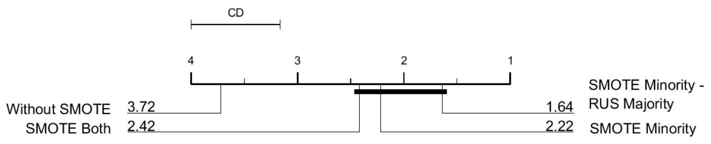
(a) Accuracy - Standard datasets *(p=0.000)*



(b) Kappa - Standard datasets *(p=0.000)*



(c) Accuracy - Microarray datasets *(p=0.140)*

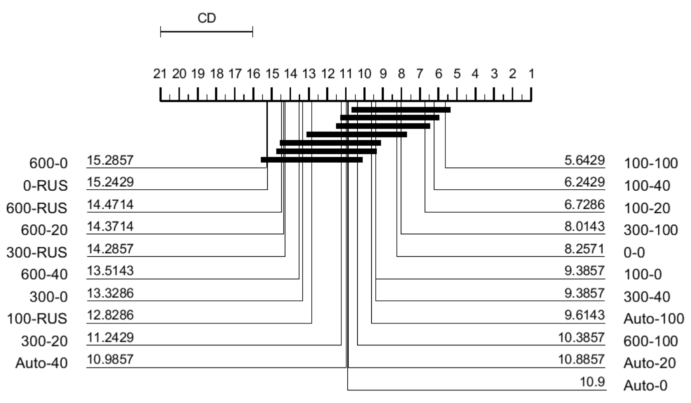


(d) Kappa - Microarray datasets *(p=0.000)*

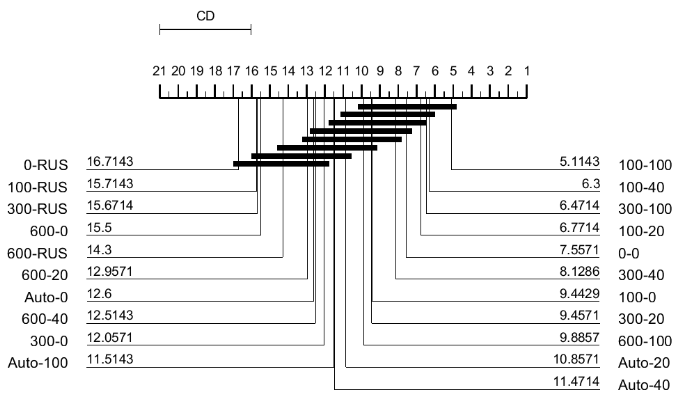
Fig. 11 Statistical comparison on different microarray and standard dataset scenarios.

only, but is never higher when SMOTE is not used; this would confirm that the use of this oversampling method produces more reliable results. This is a significant issue for large unbalanced datasets like Connect-4 and Ozone, where the use of SMOTE versus non-use of SMOTE practically doubles the kappa value. In contrast, for balanced datasets like Isolet, kappa values and accuracy averages hardly vary, as expected. The use of SMOTE implies that more cases have better kappa values (Fig. 11(b) and Fig. 11(d)); both distributed approaches are statistically similar, with very good p statistic values.

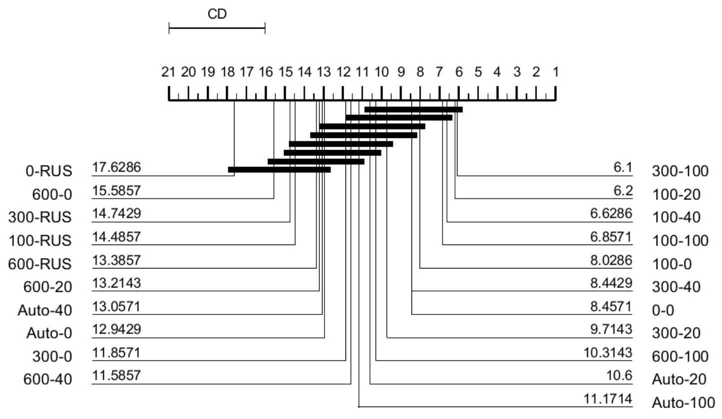
The filtering time (Tables A8 and A9) is always shorter without using SMOTE,, as expected, due to the time consumed by the extra layer of SMOTE processing and especially in large datasets with many features or samples. As expected, and for the same reason, SMOTE applied to both classes results in a higher filtering time than SMOTE applied only in the minority class; furthermore, depending on the dataset and combination, it can be very computationally expensive, e.g., for Connect4 and Gli85. Classification time on average (Table 5) shows similar behaviour as filtering time, with time increasing when distribution is applied, due to the search for a threshold, and when SMOTE is applied, due to the additional time required to apply SMOTE.



(a) Centralized distribution *(p=0.000)*

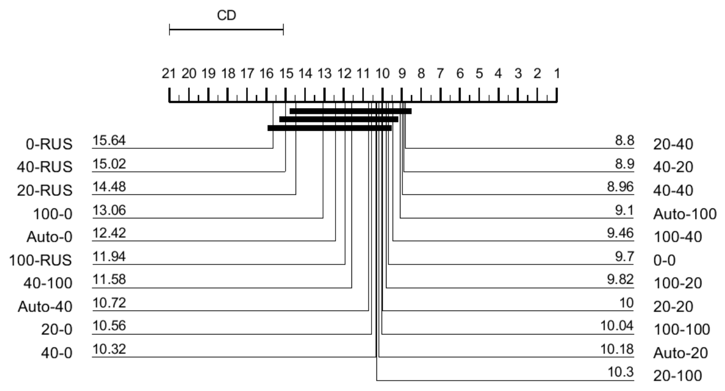


(b) Random distribution *(p=0.000)*

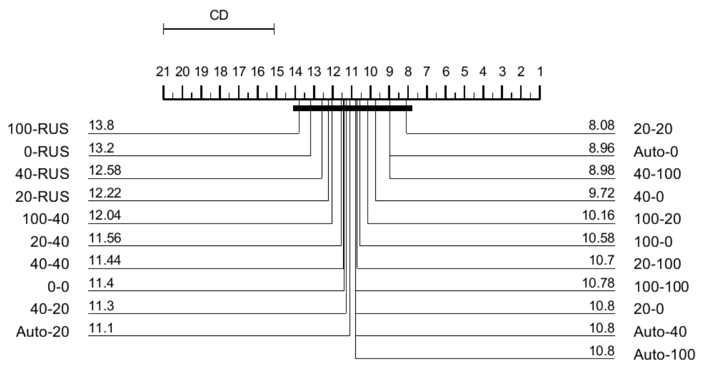


(c) Homogeneous distribution *(p=0.000)*

Fig. 12 Comparison of accuracy ranking for SMOTE different percentage scenarios on standard datasets. Each element in the ranking is labelled as SMOTE percentage in minority class - SMOTE percentage in the majority class or RUS in case of Random Undersampling in majority class.



(a) Centralized distribution *(p=0.000)*



(b) Random distribution *(p=0.571)*

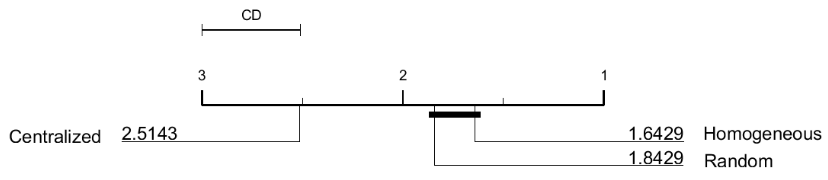
Fig. 13 Comparison of accuracy rankings for SMOTE different percentage scenarios on microarray datasets. Each element in the ranking is labelled as SMOTE percentage in minority class - SMOTE percentage in majority class or RUS in case of Random Undersampling in majority class.

Finally, the best type of distribution for each case does not change whether or not SMOTE is applied.

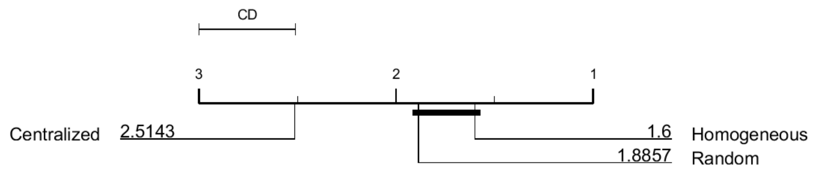
5.4 Split: horizontal vs. vertical

As can be seen in Table 4, with respect to the centralized approach, the distributed approaches with horizontal split (standard datasets) obtain a higher number of better combinations than the random distribution approaches with vertical split (microarray datasets). The reason is that, with a vertical split, redundant features can go in different packages (both would be selected); also it may happen that two features that are relevant together but not separately (if they fall into different packages) will not be selected (such as in the classic XOR problem). One solution is to group the features considering their relevance, ranking the original features before generating the packets and grouping features sequentially over the ranking, in such a way that features with a similar ranking will be in the same packet (as tested by us in a previous work [2]).

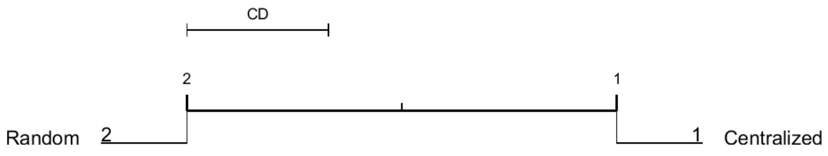
Of the two horizontal split approaches, homogeneous partitioning yielded better results than random partitioning, with more combinations of FS methods obtaining better accuracy results and with less deviation.



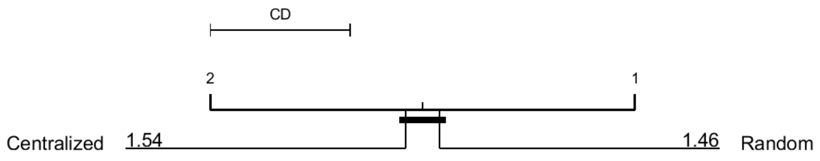
(a) Accuracy - Standard datasets *(p=0.000)*



(b) Kappa - Standard datasets *(p=0.000)*



(c) Accuracy - Microarray datasets *(p=0.000)*



(d) Kappa - Microarray datasets *(p=0.702)*

Fig. 14 Statistical comparison of distributions applying SMOTE in minority class and Random Undersampling in majority class.

5.5 Exploring how filtering and classification methods behave in different scenarios

Regarding the different FS methods tested, for filtering methods, Consistency and CFS generally obtain the best results for the different distributions and SMOTE approaches, while only InfoGain shows similar behaviour when SMOTE is applied in both classes. For classification methods, C4.5 achieves the best average accuracy results, while IB1 is the most stable (with the best kappa value) for almost all distributions and SMOTE approaches. Finally, for the packet filter time, for horizontal division (standard datasets) and for vertical division (microarray datasets), the most computationally expensive methods are ReliefF and CFS, with both filtering methods having quadratic complexity.

5.6 Exploring the benefits to combining undersampling and oversampling approaches

Finally, we compare the application of SMOTE in both the majority and minority classes with the application of SMOTE in the minority class and RUS in the majority class. In this experiment, the samples in the minority class are first increased by applying SMOTE and then RUS is applied to the samples of the majority class to balance both classes. Analysing Figure 11 and Tables A4 to A7, the application of oversampling combined with undersampling yields poorer accuracy results than the other scenarios. This is due to the fact that undersampling removes a large number of samples in order to balance the classes and this causes relevant information to be lost in obtaining the best features, subsequently affecting the classification. The kappa value behaves differently for standard and microarray datasets. For standard datasets, it yields the worst result for all scenarios (the same as accuracy), whereas the opposite happens for the microarray datasets, because the number of samples to be removed is small and the classes are completely balanced, obtaining more robust results. This issue requires further study and undoubtedly constitutes an interesting line of future research.

6 Conclusions and future work

Our methodology for distributed FS tries to solve the problem of the heterogeneity of data in different partitions. We force partitions to maintain the same class distribution as the original dataset and apply the SMOTE oversampling technique to both minority and majority classes. From experimental results for seven standard datasets with a horizontal split and five microarray datasets with a vertical split, we draw the following conclusions:

– The distributed approach with either random or homogeneous partitioning is competitive compared with the standard centralized approach and even improves the classification performance in some cases.
– The packet filter time obtains the lowest value for random distributions running concurrently, thereby considerably reducing processing time, especially for large datasets with many samples or features.
– Although the classification time is greater for the distributed approach due to the search for a threshold to eliminate irrelevant features, this increase is compensated for by the reduction in the filtering time, with the outcome that the total time decreases for the distributed approaches compared to the centralized approach.
– Homogeneous partitioning usually obtains better kappa values (more stable results) than random partitioning and than the centralized distribution.

- Applying SMOTE in the minority class (i.e., the way SMOTE is normally used) can improve the quality of learning in unbalanced datasets, but this depends on the percentage of synthetic samples generated, since in some cases the contrary effect of a slight decrease in general accuracy is obtained.
- Applying SMOTE in both minority and majority classes (i.e., an innovative use of SMOTE) can improve on the standard approach, but, as before, this depends on the percentage applied. Using SMOTE for the majority class as well as the minority class results in better classification accuracy values irrespective of dataset type and partitioning approach. This new approach obtains higher kappa values, which indicates that the method has greater robustness.
- The application of SMOTE in the minority class or SMOTE in both classes obtains better results than the combination of SMOTE in the minority class and RUS in the majority class in terms of balancing classes, especially for unbalanced standard datasets, where the elimination of a large amount of relevant samples prevents best features from being obtained, with the consequent reduction in accuracy.

As future work, we plan to test other methods to deal with heterogeneity, such as further undersampling techniques, weighting, etc. We also plan to explore how to improve vertical divisions and the kind of distribution that simultaneously makes subsets of samples and features, representing a more challenging scenario.

A Appendix

SMOTE Minority Class		0	100				300				600				Auto				0	100	300	600
SMOTE Majority Class		0	0	20	40	100	0	20	40	100	0	20	40	100	0	20	40	100	RUS			
Filter	Classifier																					
CFS	C4.5	96.41	96.04	95.95	95.63	95.45	95.50	95.59	95.32	95.59	95.04	95.50	95.68	95.50	95.36	95.36	95.18	94.72	88,41	91,96	93,89	94,33
	IB1	95.00	94.82	95.09	95.04	95.23	94.86	94.41	94.54	95.13	94.77	94.82	94.50	94.68	94.63	94.22	94.22	94.50	85,78	90,88	92,65	93,36
	Naive-Bayes	81.54	78.72	81.22	79.67	79.63	78.99	77.85	82.13	77.76	77.81	79.40	79.85	76.17	75.53	76.44	77.63	80.63	74,94	73,05	72,32	70,80
	SVM	92.72	91.41	91.77	92.31	93.04	87.13	88.95	90.13	91.50	83.90	85.13	86.72	88.59	86.45	85.36	83.40	86.90	84,48	83,81	82,37	80,51
Consistency	C4.5	96.91	96.59	97.36	96.41	96.50	95.91	95.95	96.50	96.82	96.73	97.27	96.41	96.18	97.14	96.54	96.77	97.14	88,79	92,61	94,41	95,11
	IB1	94.86	95.36	95.63	95.32	95.91	95.50	95.45	95.73	95.73	95.18	95.54	95.50	95.59	95.32	95.36	95.73	95.54	84,53	89,44	93,38	94,13
	Naive-Bayes	82.72	84.22	84.27	83.13	84.58	81.04	81.49	80.90	82.17	80.04	80.67	81.36	81.95	80.17	80.45	79.45	79.26	76,49	73,21	66,71	67,70
	SVM	93.09	91.36	91.50	91.86	92.72	88.31	88.95	89.95	91.18	81.54	83.77	84.95	88.90	84.45	83.08	83.27	83.63	83,43	84,22	82,31	81,32
InfoGain	C4.5	96.00	96.04	96.32	96.82	96.68	95.23	95.68	96.04	95.68	96.36	96.32	95.63	95.59	95.54	95.59	95.95	96.27	89,75	91,99	94,54	94,61
	IB1	95.27	95.73	95.45	95.27	96.00	95.36	95.36	95.45	95.68	95.18	95.23	95.27	95.32	95.23	95.59	95.18	95.41	86,03	90,50	93,27	94,22
	Naive-Bayes	84.58	83.81	83.86	82.90	82.81	82.04	81.95	82.13	82.45	80.54	80.17	82.90	80.99	80.81	80.81	79.54	80.26	75,45	77,66	74,32	69,57
	SVM	92.68	92.68	92.63	92.68	92.91	89.04	90.27	91.13	92.27	81.95	84.13	86.22	90.54	85.58	85.58	85.40	85.04	84,48	85,12	84,37	85,95
ReliefF	C4.5	95.68	95.13	94.72	94.91	94.72	94.50	94.77	94.86	94.91	94.32	94.72	95.00	94.68	94.86	94.36	94.50	94.50	86,44	90,94	93,85	94,04
	IB1	94.95	94.36	94.59	94.50	94.72	94.27	94.22	94.04	94.45	93.77	94.04	94.13	94.77	94.13	94.09	94.00	94.32	86,73	91,29	93,79	94,09
	Naive-Bayes	82.76	79.95	80.90	82.54	81.76	79.49	79.85	80.13	80.58	77.26	78.85	77.99	79.72	79.22	78.76	82.04	79.17	68,87	72,62	73,77	72,63
	SVM	92.77	91.63	92.13	92.36	92.27	89.04	90.77	91.22	92.09	84.72	86.13	86.99	90.31	86.31	86.27	86.22	86.63	83,85	85,03	85,25	84,77

Table A1 Accuracy results for Musk2 unbalanced standard dataset in random distribution.

SMOTE Minority Class		0	100				300				600				Auto				0	100	300	600
SMOTE Majority Class		0	0	20	40	100	0	20	40	100	0	20	40	100	0	20	40	100	RUS			
Filter	Classifier																					
CFS	C4.5	82.25	82.25	82.59	82.92	82.06	82.44	82.35	83.31	82.78	82.54	82.97	82.49	82.25	82.25	82.88	81.91	82.49	81,63	81,47	81,63	81,04
	IB1	89.80	89.13	89.37	89.61	89.23	89.18	89.23	89.18	89.51	89.71	89.08	90.14	90.19	89.80	89.66	89.08	89.13	87,97	88,50	88,59	88,32
	Naive-Bayes	88.55	88.50	88.22	88.22	88.36	88.07	88.31	88.31	88.22	88.07	88.89	88.07	88.89	88.55	88.46	88.22	88.41	86,88	87,30	87,19	87,17
	SVM	95.67	95.77	95.77	96.01	95.67	95.67	95.86	95.82	95.91	95.86	95.72	95.77	95.86	95.67	95.62	95.62	95.96	95,54	95,39	95,10	95,41
Consistency	C4.5	74.36	76.38	73.88	74.17	74.75	73.98	73.74	76.62	74.75	74.89	75.37	74.75	74.56	74.36	73.06	72.49	76.09	69,91	70,67	70,43	72,48
	IB1	81.10	82.25	80.81	82.06	80.66	81.39	80.90	81.05	81.72	81.29	82.64	83.26	81.63	81.10	81.58	83.17	82.11	76,90	77,62	78,12	79,14
	Naive-Bayes	82.64	82.15	82.15	81.72	82.15	82.54	82.83	82.78	80.86	82.54	83.41	83.26	82.78	82.64	82.11	82.25	83.93	78,74	79,09	79,39	80,55
	SVM	88.50	89.27	88.17	89.27	88.41	87.64	88.46	88.65	87.40	88.55	88.84	87.97	88.46	88.55	88.07	88.79	88.74	85,89	86,16	86,54	86,72
InfoGain	C4.5	81.53	81.58	81.10	81.43	81.43	81.82	82.59	81.48	81.72	82.20	81.72	81.77	81.67	81.53	81.53	81.34	81.24	80,54	80,04	79,84	80,11
	IB1	89.80	89.80	90.00	89.66	89.42	90.62	90.24	90.43	90.24	90.67	90.33	90.86	90.43	89.80	89.75	89.51	89.80	88,81	88,95	89,04	89,10
	Naive-Bayes	78.07	77.78	78.21	78.45	77.97	78.69	80.18	79.65	79.46	79.75	79.41	80.52	79.27	78.07	78.40	78.16	79.17	76,93	77,59	76,71	76,78
	SVM	95.43	95.57	95.53	95.57	95.38	95.53	95.82	95.86	95.72	95.48	95.38	95.48	95.67	95.43	95.38	95.48	95.43	95,19	94,80	95,20	95,20
ReliefF	C4.5	80.86	80.95	81.29	81.29	81.53	81.43	82.06	81.48	82.11	81.24	81.29	80.86	81.96	80.86	80.86	81.39	82.01	79,47	79,68	79,37	79,80
	IB1	87.25	87.59	87.35	88.02	87.45	88.07	88.12	88.12	87.88	88.99	88.55	88.55	88.41	87.25	86.82	87.35	87.54	85,76	86,09	86,02	86,12
	Naive-Bayes	84.08	84.94	84.99	84.13	84.22	84.66	84.51	84.99	84.03	85.52	83.98	86.10	85.47	84.08	84.27	85.04	83.98	83,23	83,82	83,36	83,41
	SVM	95.29	95.24	95.14	95.05	94.85	95.33	95.38	95.29	95.09	95.62	95.05	95.14	95.24	95.29	95.09	95.09	94.90	94,03	94,23	94,19	94,45

Table A2 Accuracy results for Isolet balanced standard dataset in homogeneous distribution.

SMOTE Minority Class		0	20				40				100				Auto				0	20	40	100
SMOTE Majority Class		0	0	20	40	100	0	20	40	100	0	20	40	100	0	20	40	100	RUS			
Filter	Classifier																					
CFS	C4.5	80.00	65.72	85.71	71.43	74.28	74.29	68.57	71.43	74.29	60.00	65.71	65.71	60.00	60.00	71.43	71.43	68.57	65,71	80,00	77,14	68,57
	IB1	48.57	48.57	40.00	34.29	34.29	28.57	34.29	37.14	34.29	31.43	25.71	28.57	34.29	28.57	37.14	34.29	40.00	31,43	28,57	28,57	25,71
	Naive-Bayes	77.14	71.43	77.14	71.43	71.43	74.28	77.14	74.28	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	60,00	71,43	68,57	57,14
	SVM	40.00	45.72	57.14	54.29	48.57	57.14	48.57	45.71	57.14	65.71	54.29	42.86	51.43	68.57	54.29	60.00	60.00	65,72	45,72	60,00	60,00
Consistency	C4.5	68.57	74.28	80.00	65.71	91.43	77.14	77.14	74.28	80.00	68.57	74.28	60.00	68.57	74.29	68.57	71.43	57.14	42,86	57,14	65,71	54,29
	IB1	57.14	54.29	57.14	48.57	45.71	48.57	54.28	65.71	57.14	48.57	37.15	51.43	48.57	51.43	57.14	48.57	45.71	34,29	37,14	68,57	37,15
	Naive-Bayes	60.00	71.43	68.57	62.86	51.43	48.57	57.14	62.86	62.86	60.00	54.28	65.71	68.57	77.14	65.71	60.00	65.71	51,43	48,57	51,43	68,57
	SVM	62.86	57.14	51.43	62.86	65.71	57.14	60.00	74.29	65.71	45.71	42.86	51.43	48.57	54.29	40.00	45.72	48.57	37,14	42,86	57,14	34,29
InfoGain	C4.5	80.00	71.43	82.86	71.43	71.43	77.14	62.86	74.29	71.43	80.00	65.71	77.14	77.14	68.57	77.14	74.29	74.28	54,29	68,57	68,57	71,43
	IB1	34.29	31.43	28.57	48.57	42.86	28.57	28.57	37.14	31.43	28.57	28.57	31.43	31.43	28.57	31.43	42.86	28.57	28,57	28,57	28,57	28,57
	Naive-Bayes	71.43	68.57	71.43	71.43	71.43	71.43	71.43	71.43	68.57	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	62,86	60,00	57,14	51,43
	SVM	51.43	60.00	48.57	57.14	57.14	48.57	45.72	60.00	54.29	45.71	60.00	57.14	45.71	57.14	62.86	45.71	57.14	60,00	48,57	54,29	60,00
ReliefF	C4.5	74.29	65.71	65.71	74.28	74.28	77.14	77.14	54.29	65.72	62.86	71.43	68.57	77.14	62.86	77.14	74.28	71.43	62,86	71,43	57,14	68,57
	IB1	40.00	28.57	31.43	42.86	48.57	28.57	37.14	25.72	45.71	28.57	25.71	28.57	51.43	28.57	31.43	31.43	28.57	28,57	28,57	28,57	28,57
	Naive-Bayes	71.43	71.43	71.43	71.43	68.57	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	71.43	57,14	57,14	65,72	65,72
	SVM	60.00	45.71	57.14	68.57	57.14	62.86	57.14	54.29	65.71	48.57	51.43	42.86	57.14	65.71	65.71	54.29	51.43	48,57	54,29	60,00	60,00

Table A3 Accuracy results for Brain microarray dataset in random distribution.

Scenario	Distribution	Connect4	Isolet	Musk2	Nomao	Ozone	Spambase	Weight
Without SMOTE	Centralized	66.30 ± 4.26 **76.88** **Cons+C4.5**	80.70 ± 10.10 95.43 CFS+SVM	88.97 ± 7.62 96.13 IG+C4.5	85.47 ± 9.35 94.75 Cons+C4.5	**91.78 ± 10.35** 97.75 Several	86.18 ± 6.54 **93.61** **CFS+C4.5**	84.94 ± 13.45 **100** **Several**
	Random	**67.55 ± 3.59** 74.85 Rel+C4.5	84.76 ± 6.59 95.62 CFS+SVM	**90.51 ± 6.42** 96.91 Cons+C4.5	**88.57 ± 7.13** 95.12 Cons+C4.5	91.26 ± 11.54 97.4 Several	**87.38 ± 4.68** 93.55 CFS+C4.5	90.19 ± 10.76 **100** **Several**
	Homogeneous	64.85 ± 8.67 76.35 IG+C4.5	**84.95 ± 6.65** **95.67** CFS+SVM	90.45 ± 6.63 **97** **Cons+C4.5**	87.79 ± 8.57 **95.41** **IG+C4.5**	90.83 ± 11.15 **97.99** **All+SVM**	87.37 ± 4.98 93.16 CFS+C4.5	**93.38 ± 8.54** **100** **Several**
SMOTE Min.	Centralized	59.77 ± 11.76 **78.83** **Cons+C4.5+100**	80.22 ± 10.66 **95.86** **CFS+SVM+300**	86.49 ± 9.77 96.82 Cons+C4.5+600	81.96 ± 16.89 94.86 Cons+C4.5+100	**88.61 ± 8.57** 97.75 CFS+SVM+100	85.46 ± 6.93 **93.94** **CFS+C4.5+100**	86.73 ± 12.91 **100** **Several**
	Random	**60.73 ± 11.12** 75.19 Cons+C4.5+300	**84.90 ± 6.58** 95.77 IG+SVM+600	87.88 ± 7.96 **97.14** **Cons+C4.5+Auto**	85.32 ± 12.81 94.93 CFS+C4.5+100	87.97 ± 10.41 97.4 Several	86.15 ± 5.94 93.87 CFS+C4.5+100	90.94 ± 10.65 **100** **Several**
	Homogeneous	57.02 ± 11.14 74.87 Rel+C4.5+100	84.83 ± 6.74 **95.86** **CFS+SVM+600**	**88.62 ± 7.10** 96.73 Cons+C4.5+300	**87.24 ± 10.35** **95.15** **IG+C4.5+Auto**	88.45 ± 9.48 **97.99** **Several**	**86.63 ± 5.27** 93.22 CFS+C4.5+100	**93.63 ± 8.70** **100** **Several**
SMOTE Both	Centralized	62.34 ± 10.60 **78.82** **Cons+C4.5+100+40**	80.11 ± 10.73 95.48 CFS+SVM+Several	87.23 ± 9.45 97.32 Cons+C4.5+600+100	84.01 ± 13.58 95.05 Cons+C4.5+Auto+100	**89.29 ± 8.46** 97.75 Several	87.10 ± 6.03 **93.81** **Cons+C4.5+300+40**	86.13 ± 14.20 **100** **Several**
	Random	**62.40 ± 10.05** 75.19 Rel+C4.5+300+20	**84.93 ± 6.51** 95.96 CFS+SVM+600+20	88.65 ± 7.51 **97.36** **Cons+C4.5+100+40**	86.03 ± 13.01 95.01 IG+C4.5+100+20	88.67 ± 10 97.63 IG+C4.5+100+100	87.82 ± 4.02 93.68 CFS+C4.5+100+All	92.16 ± 9.73 **100** **Several**
	Homogeneous	59.26 ± 10.40 75.52 IG+C4.5+100+100	84.91 ± 6.69 **96.01** **CFS+SVM+100+40**	**89.52 ± 6.48** 96.95 IG+C4.5+100+40	**87.33 ± 10.86** **95.32** **IG+C4.5+300+20**	89.05 ± 9.21 **97.99** **Several**	**88.04 ± 3.91** 93.74 Several	**93.81 ± 9.06** **100** **Several**
SMOTE Min. RUS Maj.	Centralized	**53.14 ± 11.87** **74.27** **Cons+C4.5+600**	80.88 ± 9.56 95.29 CFS+SVM+600	81.78 ± 11.51 95.77 CFS+C4.5+600	84.15 ± 12.34 94.55 Cons+C4.5+100	**78.87 ± 5.44** **89.23** **IG+C4.5+600**	87.06 ± 4.48 92.83 Several+C4.5+100	85.13 ± 16.24 **100** **Several**
	Random	50.07 ± 12.06 70.31 IG+IB1+300	84.15 ± 6.86 95.62 CFS+SVM+100	84.86 ± 8.52 96.00 IG+C4.5+300	**87.28 ± 8.58** **94.97** **Cons+C4.5+300**	78.56 ± 5.00 88.99 Rel+C4.5+600	88.34 ± 3.80 **93.42** **CFS+C4.5+100**	89.43 ± 12.37 **100** **Several**
	Homogeneous	52.24 ± 10.99 71.24 Rel+IB1+600	**84.46 ± 6.90** **96.10** **CFS+SVM+600**	**85.24 ± 8.18** **96.09** **Cons+C4.5+600**	87.18 ± 9.84 94.95 Cons+C4.5+600	78.02 ± 5.62 88.52 Cons+C4.5+600	**88.81 ± 3.21** 92.96 CFS+C4.5+100	**90.84 ± 11.62** **100** **Several**

Table A4 Summary of accuracy results on standard datasets. Values for all approaches and measures are depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the highest values in the distribution for each scenario are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

Scenario	Distribution	Brain	CNS	Colon	Gli85	Ovarian
Without SMOTE	Centralized	**63.75 ± 24.68** **100** **Several**	**59.75 ± 11.38** **80** **Several**	**77.73 ± 9.21** **95.24** **Rel+Naive**	**85.31 ± 5.74** **96.43** **IG+SVM**	**98.76 ± 1.66** **100** **Several**
	Random	61.07 ± 20.95 **100** **Several**	52.62 ± 12.67 70 Several	53.57 ± 13.17 85.71 Cons+C4.5	58.61 ± 14.09 78.57 CFS+IB1	46.81 ± 12.49 89.29 Rel+C4.5
SMOTE Min.	Centralized	**62.72 ± 23.59** **100** **Several**	**57.76 ± 11.19** **85** **Rel+C4.5+100**	**76.62 ± 9.68** **95.24** **Rel+Naive+20**	**84.61 ± 6.09** **100** **Several**	**98.77 ± 1.49** **100** **Several**
	Random	57.67 ± 20.72 **100** **Several**	53.98 ± 13.02 80 Cons+SVM+100	55.69 ± 12.59 80.95 Rel+SVM+100	57.32 ± 14.28 85.71 Rel+IB1+20	48.71 ± 12.40 88.1 Rel+C4.5+20
SMOTE Both	Centralized	**64.28 ± 23.74** **100** **Several**	**59.60 ± 10.21** **90** **IG+IB1+20+100**	**78.17 ± 9.39** **95.24** **Several**	**84.11 ± 6.43** **100** **Several**	**98.76 ± 1.50** **100** **Several**
	Random	58.70 ± 20.92 **100** **Several**	53.68 ± 12.52 85 Cons+C4.5+40+20	56.19 ± 11.50 80.95 Cons+IB1+40+20	55.97 ± 14.53 89.29 CFS+SVM+100+100	47.01 ± 11.72 83.33 Rel+C4.5+20+100
SMOTE Min. RUS Maj.	Centralized	**63.92 ± 20.45** **100** **Several**	**57.96 ± 10.45** **80** **Several**	**75.68 ± 9.89** **95.24** **Rel+SVM+40**	**77.82 ± 8.36** **92.86** **Several**	**98.06 ± 2.12** **100** **Several**
	Random	52.45 ± 23.78 **100** **Several**	51.51 ± 13.38 **80** **Cons+SVM+0**	52.85 ± 12.90 90.48 Cons+SVM+100	53.18 ± 15.95 89.29 IG+Naive+20	52.49 ± 12.83 83.33 Cons+IB1+100

Table A5 Summary of accuracy results on microarray datasets. Values for all approaches and measures are depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the highest values in the distribution for each scenario are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

Scenario	Distribution	Connect4	Isolet	Musk2	Nomao	Ozone	Spambase	Weight
Without SMOTE	Centralized	0.216 ± 0.285 **1** **Cons+IB1**	0.919 ± 0.101 **1** **All+IB1**	0.670 ± 0.319 **1** **All+IB1**	0.662 ± 0.295 **1** **Cons+IB1**	0.342 ± 0.406 **1** **Several+IB1**	0.798 ± 0.167 **0.999** **Several+IB1**	0.809 ± 0.185 **1** **Several**
	Random	0.205 ± 0.180 0.511 Rel+C4.5	**0.943 ± 0.072** **1** **All+IB1**	**0.766 ± 0.233** **1** **All+IB1**	**0.750 ± 0.277** **1** **Several+IB1**	0.313 ± 0.397 **1** **Several+IB1**	**0.825 ± 0.155** **0.999** **Several+IB1**	0.864 ± 0.153 **1** **Several**
	Homogeneous	**0.266 ± 0.255** 0.776 Cons+IB1	**0.943 ± 0.070** **1** underline**All+IB1**	0.742 ± 0.252 **1** **All+IB1**	0.730 ± 0.303 **1** **Several+IB1**	**0.394 ± 0.416** **1** **Several+IB1**	**0.825 ± 0.160** **0.999** **Several+IB1**	**0.907 ± 0.122** **1** **Several**
SMOTE Min.	Centralized	**0.459 ± 0.356** **1** **Cons+IB1+All**	0.919 ± 0.101 **1** **All+IB1+All**	0.800 ± 0.202 **1** **All+IB1+All**	0.679 ± 0.296 **1** **Cons+IB1+All**	**0.698 ± 0.325** **1** **All+IB1+All**	0.835 ± 0.144 **1** **Several**	0.845 ± 0.165 **1** **Several**
	Random	0.432 ± 0.338 0.924 Cons+IB1+600	**0.946 ± 0.068** **1** **All+IB1+All**	**0.818 ± 0.186** **1** **All+IB1+All**	0.758 ± 0.264 **1** **Several+IB1+Several**	0.658 ± 0.358 **1** **Several**	0.843 ± 0.140 **1** **Several**	0.892 ± 0.128 **1** **Several**
	Homogeneous	0.362 ± 0.286 0.925 Cons+IB1+600	**0.946 ± 0.067** **1** **All+IB1+All**	0.810 ± 0.196 **1** **All+IB1+All**	**0.771 ± 0.267** **1** **Several+IB1+Several**	0.667 ± 0.357 **1** **Several**	**0.847 ± 0.138** **1** **Several**	**0.923 ± 0.108** **1** **Several**
SMOTE Both	Centralized	**0.478 ± 0.380** **1** **Several**	0.920 ± 0.101 **1** **Several**	0.789 ± 0.218 **1** **Several**	0.696 ± 0.285 **1** **Cons+IB1+All+All**	**0.667 ± 0.361** **1** **Several**	0.855 ± 0.135 **1** **Several**	0.845 ± 0.177 **1** **Several**
	Random	0.420 ± 0.346 0.965 Rel+IB1+Auto+100	**0.947 ± 0.066** **1** **Several**	**0.811 ± 0.194** **1** **Several**	**0.768 ± 0.272** **1** **Several**	0.632 ± 0.378 **1** **Several**	**0.864 ± 0.122** **1** **Several**	0.908 ± 0.120 **1** **Several**
	Homogeneous	0.351 ± 0.295 0.933 Cons+IB1+600+100	**0.947 ± 0.065** **1** **Several**	0.804 ± 0.201 **1** **Several**	0.765 ± 0.285 **1** **Several**	0.640 ± 0.379 **1** **Several**	0.863 ± 0.127 **1** **Several**	**0.926 ± 0.116** **1** **Several**
SMOTE Min. RUS Maj.	Centralized	**0.414 ± 0.331** **1** **Cons+IB1**	0.921 ± 0.098 **1** **All+IB1+All**	0.766 ± 0.227 **1** **All+IB1+All**	0.703 ± 0.275 **1** **Cons+IB1**	**0.830 ± 0.145** **1** **Several+IB1**	0.843 ± 0.134 **1** **Several+IB1**	0.880 ± 0.147 **1** **Several**
	Random	0.302 ± 0.249 0.871 IG+IB1+600	0.942 ± 0.073 **1** **All+IB1+All**	**0.820 ± 0.183** **1** **All+IB1+All**	**0.774 ± 0.237** **1** **Cons+IB1**	**0.830 ± 0.147** **1** **Several+IB1**	0.859 ± 0.124 **1** **Several+IB1**	0.919 ± 0.102 **1** **Several**
	Homogeneous	0.379 ± 0.253 0.876 Rel+IB1+0	**0.943 ± 0.071** **1** **All+IB1+All**	0.808 ± 0.197 **1** **All+IB1+All**	0.752 ± 0.290 **1** **Several**	0.829 ± 0.149 **1** **Several+IB1**	**0.864 ± 0.122** **1** **Several+IB1**	**0.934 ± 0.093** **1** **Several**

Table A6 Summary of kappa results on standard datasets. Values for all approaches and measures are depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the highest values in the distribution for each scenario are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

Scenario	Distribution	Brain	CNS	Colon	Gli85	Ovarian
Without SMOTE	Centralized	0.876 ± 0.301 **1** **Several**	**0.867 ± 0.259** **1** **Several**	0.826 ± 0.207 **1** **Several**	**0.922 ± 0.163** **1** **Several**	**0.985 ± 0.021** **1** **Several**
	Random	**0.938 ± 0.169** **1** **Several**	0.857 ± 0.254 **1** **Several**	**0.834 ± 0.203** **1** **Several**	**0.922 ± 0.157** **1** **Several**	0.983 ± 0.027 **1** **Several**
SMOTE Min.	Centralized	0.885 ± 0.288 **1** **Several**	0.897 ± 0.214 **1** **Several**	0.864 ± 0.160 **1** **Several**	0.950 ± 0.113 **1** **Several**	**0.988 ± 0.017** **1** **Several**
	Random	**0.945 ± 0.160** **1** **Several**	**0.899 ± 0.170** **1** **Several**	**0.871 ± 0.171** **1** **Several**	**0.955 ± 0.096** **1** **Several**	0.983 ± 0.028 **1** **Several**
SMOTE Both	Centralized	0.881 ± 0.292 **1** **Several**	0.889 ± 0.236 **1** **Several**	0.870 ± 0.167 **1** **Several**	0.94 ± 0.117 **1** **Several**	**0.989 ± 0.016** **1** **Several**
	Random	**0.947 ± 0.166** **1** **Several**	**0.894 ± 0.191** **1** **Several**	**0.875 ± 0.181** **1** **Several**	**0.950 ± 0.089** **1** **Several**	0.986 ± 0.024 **1** **Several**
SMOTE Min. RUS Maj.	Centralized	0.910 ± 0.242 **1** **Several**	0.908 ± 0.190 **1** **Several**	**0.892 ± 0.127** **1** **Several**	**0.964 ± 0.086** **1** **Several**	**0.987 ± 0.017** **1** **Several**
	Random	**0.968 ± 0.103** **1** **Several**	**0.912 ± 0.162** **1** **Several**	0.885 ± 0.150 **1** **Several**	0.960 ± 0.083 **1** **Several**	0.984 ± 0.026 **1** **Several**

Table A7 Summary of kappa results on microarray datasets. Values for all approaches and measures are depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the highest values in the distribution for each scenario are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

Scenario	Distribution	Connect4	Isolet	Musk2	Nomao	Ozone	Spambase	Weight
Without SMOTE	Centralized	305.13 ± 327.41 854.89 Rel+All	102.98 ± 30.88 158.48 Rel+All	42.66 ± 15.58 71.96 Rel+All	292.28 ± 297.29 814.61 Rel+All	7.54 ± 0.98 9.24 Rel+All	11.05 ± 2.27 15.13 Rel+All	21.52 ± 5.23 30.78 Rel+All
	Random	**2.76 ± 0.14** **3.04** **Rel+All**	**25.20 ± 2.98** **27.84** **Rel+All**	**8.04 ± 0.53** **8.86** **Rel+All**	**31.69 ± 15.01** **62.79** **Rel+All**	**2.90 ± 0.06** **3.05** **Rel+All**	**3.35 ± 0.10** **3.48** **Rel+All**	**7.12 ± 0.47** **7.97** **Rel+All**
	Homogeneous	2.91 ± 0.20 3.36 Rel+All	28.35 ± 3.69 31.59 Rel+All	8.24 ± 0.56 9.21 Rel+All	57.48 ± 31.62 112.68 Rel+All	3.10 ± 0.07 3.27 Rel+All	3.43 ± 0.09 3.58 Cons+All	7.83 ± 0.60 8.92 Rel+All
SMOTE Min.	Centralized	541.25 ± 696.34 2240.96 Rel+All+600	111.07 ± 39.42 225.40 Rel+All+600	62.96 ± 49.03 256.02 Rel+All+600	816.27 ± 1351.44 5409.11 Rel+All+600	8.86 ± 2.16 16.37 Rel+All+Auto	19.46 ± 16.58 80.57 Rel+All+600	23.50 ± 7.06 40.74 Rel+All+Auto
	Random	**3.47 ± 0.36** **4.61** **Rel+All+Auto**	**27.36 ± 3.59** **34.19** **CFS+All+100**	**9.40 ± 1.37** **13.15** **Rel+All+600**	**61.10 ± 68.73** **301.28** **Rel+All+600**	**3.33 ± 0.12** **3.71** **Rel+All+Auto**	**4.30 ± 0.62** **6.60** **Rel+All+600**	**7.74 ± 0.64** **9.34** **Rel+All+Auto**
	Homogeneous	3.52 ± 0.35 4.66 Rel+All+600	30.26 ± 4.11 37.08 Rel+All+600	9.72 ± 1.56 14.18 Rel+All+600	119.97 ± 153.36 670.54 Rel+All+600	3.55 ± 0.14 4.09 Rel+All+Auto	4.44 ± 0.67 6.86 Rel+All+600	8.48 ± 0.79 10.48 Rel+All+Auto
SMOTE Both	Centralized	1275.89 ± 1402.88 7797.43 Rel+All+Auto+100	114.58 ± 40.43 246.67 Rel+All+600+100	109.22 ± 104.90 739.93 Rel+All+Auto+100	1469.75 ± 1984.92 9090.18 Rel+All+600+100	13.53 ± 6.07 47.13 Rel+All+Auto+100	25.19 ± 20.85 112.93 Rel+All+600+100	27.60 ± 11.15 71.53 Rel+All+Auto+100
	Random	**4.46 ± 0.64** **7.09** **Rel+All+Auto+100**	**28.58 ± 3.69** **35.47** **Rel+All+600+100**	**12.01 ± 3.01** **25.63** **Rel+All+Auto+100**	**96.57 ± 101.62** **469.61** **Rel+All+600+100**	**4.16 ± 0.32** **5.95** **Rel+All+Auto+100**	**5.38 ± 0.74** **8.32** **Rel+All+600+100**	**8.71 ± 1.00** **12.74** **Rel+All+Auto+100**
	Homogeneous	4.55 ± 0.69 8.13 Rel+All+Auto+100	31.70 ± 4.31 39.21 Rel+All+600+100	12.38 ± 3.36 27.49 Rel+All+Auto+100	191.44 ± 221.54 1033.65 Rel+All+600+100	4.47 ± 0.42 6.79 Rel+All+Auto+100	5.49 ± 0.82 8.74 Rel+All+600+100	9.53 ± 1.23 14.46 Rel+All+Auto+100
SMOTE Min. RUS Maj.	Centralized	238.95 ± 221.80 844.12 Rel+All+300	173.03 ± 47.18 257.96 Rel+All+100	51.60 ± 35.14 174.86 Rel+All+600	464.69 ± 585.26 1948.49 Rel+All+600	9.10 ± 0.64 10.77 Cons+All+600	15.17 ± 4.5 29.60 Rel+All+600	21.82 ± 0.95 24.41 Rel+All+300
	Random	**4.77 ± 0.31** **5.49** **Rel+All+600**	**44.80 ± 4.07** **50.51** **Rel+All+600**	**9.94 ± 1.49** **14.46** **Rel+All+600**	**59.87 ± 39.95** **158.52** **Rel+All+300**	4.07 ± 0.33 5.13 Rel+All+600	4.84 ± 0.49 **5.67** **Rel+All+300**	**8.27 ± 0.29** **8.76** **Rel+All+600**
	Homogeneous	4.89 ± 0.34 5.53 IG+All+100	52.29 ± 6.12 61.94 Rel+All+600	11.23 ± 1.89 17.63 Rel+All+600	84.04 ± 65.56 249.81 Rel+All+600	**3.93 ± 0.21** **4.55** **CFS+All+600**	**4.82 ± 0.47** 6.00 Rel+All+300	8.95 ± 0.31 9.41 Rel+All+600

Table A8 Summary of results of filter time by packet on standard datasets. Values for all approaches and measures are in seconds and depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the lowest values (best values) for each SMOTE scenario and each measure are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

Scenario	Distribution	Brain	CNS	Colon	Gli85	Ovarian
Without SMOTE	Centralized	13.75 ± 2.90 18.73 CFS+All	43.65 ± 53.33 200.59 CFS+All	6.46 ± 1.48 9.60 CFS+All	483.87 ± 1053.18 4684.18 CFS+All	122.98 ± 23.64 178.08 CFS+All
	Random	**3.05 ± 0.18** **3.70** **Rel+All**	**6.45 ± 1.59** **10.59** **CFS+All**	**2.60 ± 0.07** **2.71** **CFS+All**	**73.60 ± 101.24** **312.43** **CFS+All**	**26.84 ± 1.76** **30.47** **CFS+All**
SMOTE Min.	Centralized	14.78 ± 2.94 19.88 CFS+All+Auto	76.79 ± 110.83 393.85 CFS+All+40	7.36 ± 2.11 13.01 CFS+All+100	1627.03 ± 3243.19 12766.70 CFS+All+100	128.36 ± 23.16 190.40 CFS+All+40
	Random	**3.66 ± 0.31** **4.30** **CFS+All+Auto**	**8.18 ± 3.31** **17.67** **CFS+All+100**	**2.96 ± 0.07** **3.12** **CFS+All+Auto**	**84.87 ± 124.16** **548.53** **CFS+All+100**	**28.42 ± 1.83** **34.23** **CFS+All+100**
SMOTE Both	Centralized	16.01 ± 2.95 21.97 CFS+All+Auto+100	89.26 ± 147.32 762.16 CFS+All+40+100	8.12 ± 2.31 16.80 CFS+All+100+100	1159.02 ± 2516.06 11782.35 CFS+All+100+40	137.59 ± 24.36 243.79 CFS+All+Auto+100
	Random	**4.13 ± 0.30** **4.55** **Rel+All+100+100**	**9.53 ± 4.52** **25.27** **CFS+All+100+100**	**3.37 ± 0.08** **3.64** **CFS+All+Auto+100**	**91.90 ± 129.16** **478.15** **CFS+All+40+100**	**30.94 ± 2.12** **37.48** **CFS+All+Auto+100**
SMOTE Min. RUS Maj.	Centralized	16.44 ± 3.55 23.18 CFS+All+100	60.33 ± 95.23 391.34 CFS+All+100	8.45 ± 2.51 19.30 CFS+All+100	1357.32 ± 2860.72 12868.10 CFS+All+100	135.72 ± 32.22 261.19 CFS+All+40
	Random	**4.17 ± 0.19** **4.73** **Rel+All+20**	**8.73 ± 3.21** **19.00** **CFS+All+100**	**3.61 ± 0.20** **3.93** **CFS+All+100**	**55.63 ± 76.76** **339.94** **CFS+All+100**	**29.98 ± 2.04** **35.58** **CFS+All+40**

Table A9 Summary of results of filter time by packet on microarray datasets. Values for all approaches and measures are in seconds and depicted as means+standard deviation (top row), maximum values (second row) and (lowest row) the combination of filter+classifier+SMOTE percentage in minority class+SMOTE percentage in majority class or RUS in majority class that obtain the maximum value. In each dataset, the lowest values (best values) for each SMOTE scenario and each measure are in bold font and the best values by dataset irrespective of the over/under sampling scenario are underlined.

References

1. I. Guyon. *Feature extraction: foundations and applications*, volume 207. Springer, 2006.
2. V. Bolón-Canedo, N. Sánchez-Maroño, and Alonso-Betanzos. Distributed feature selection: An application to microarray data classification. *Applied Soft Computing*, 30:136–150, 2015.
3. V. Bolón-Canedo, K. Sechidis, N. Sánchez-Maroño, A. Alonso-Betanzos, and G. Brown. Insights into distributed feature ranking. *Information Sciences*, 496:378–398, 2019.
4. A. Brankovic, M. Hosseini, and L. Piroddi. A distributed feature selection algorithm based on distance correlation with an application to microarrays. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(6):1802–1815, 2019.
5. L. Morán-Fernández, V. Bolón-Canedo, and A. Alonso-Betanzos. Centralized vs. distributed feature selection methods based on data complexity measures. *Knowledge-Based Systems*, 117:27–45, 2017.
6. H. He and E. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.
7. G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G Bing. Learning from class-imbalanced data: review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.
8. P. Murphy M. Pazzani, C. Merz and C. Brunk. Reducing misclassification costs. In *International Conference of Machine Learning, New Brunswick*, pages 217–225. Morgan Kauffman, 1994.
9. M.A. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan. A multiple expert approach to the class imbalance problem using inverse random under sampling. In *Multiple Classifier Systems*, pages 82–91, 2009.
10. A. H. Solberg and R. Solberg. A large-scale evaluation of features for automatic detection of oil spills in ers sar images. In *International Geoscience and Remote Sensing Symposium*, pages 1484–1486. Lincoln, NE, 1996.
11. N. V. Chawla, F. Herrera, S. Garcia, and A. Fernandez. Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.
12. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *arXiv preprint arXiv:1106.1813*, 2011.
13. H. He, Y. Bai, E.A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE Joint Conference in Neural Networks, IJCNN 2008*, 2008.
14. C. Ling, C.X. amd Li. Data mining for direct marketing: Problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, KDD'98*, volume 98, pages 73–79, 1998.
15.
16. Combining over-sampling and under-sampling techniques for imbalance dataset. In *ICMLC 2017: Proceedings of the 9th International Conference on Machine Learning and Computing (ICMLC)*, pages 243—247, 2017.
17. Smote-rsb *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory., 2012.
18. Dbsm: The combination of dbscan and smote for imbalanced data classification. In *13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–5, 2016.
19. Qingjie Wang, Jingmin Xin, Jiayi Wu, and Nanning Zheng. SVM classification of microaneurysms with imbalanced dataset based on borderline-SMOTE and data cleaning techniques. In Antanas Verikas, Petia Radeva, Dmitry P. Nikolaev, Wei Zhang, and Jianhong Zhou, editors, *Ninth International Conference on Machine Vision (ICMV 2016)*, volume 10341, pages 355 – 361. International Society for Optics and Photonics, SPIE, 2017.
20. An imbalanced data classification algorithm of improved autoencoder neural network. In *Eighth International Conference on Advanced Computational Intelligence (ICACI)*, 2016.
21. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014.
22. Yang J, Zhou J, Zhu Z, Ma X, and Ji Z. Iterative ensemble feature selection for multiclass classification of imbalanced microarray data. *J Biol Res (Thessalon)*, 23(Suppl 1)(13), 2016.

23. A fraud detection model based on feature selection and undersampling applied to web payment systems. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2015.
24. Feature selection for high-dimensional class-imbalanced datasets using support vector machines. *Information Sciences*, 286:228–246, 2014.
25. M. Hall. *Correlation-based Feature Selection for Machine Learning.* PhD thesis, The University of Waikato, 1999.
26. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982. Reprinted in J.W. Shavlik and T.G.Dietterich Eds (1990), Readings in machine learning. San Francisco, CA: Morgan Kaufmann.
27. P.H. Winston. Learning structural description from examples. In *The psychology of computer vision.* P.H. Winston (Ed.), New York, McGraw-Hill, 1975.
28. J.R. Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine learning: An artificial intelligence approach.* R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds), San Francisco, CA: Morgan Kaufmann, 1983.
29. J.R.Quinlan. *C4.5: Programs for machine learning.* San Francisco: Morgan Kaufmann, 1993.
30. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees.* Belmont, CA: Wadsworth, 1984.
31. C. Cortes and V. N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
32. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction. Foundations and Applications.* Springer, 2006.
33. D.J. Hand, H.Mannila, and P. Smyth. *Principles of Data Mining.* MIT press, 2001.
34. V. Bolón-Canedo, N. Sánchez-Maroño, and J. Cerviño-Rabuñal. Scaling up feature selection: a distributed filter approach. In *Conference of the Spanish Association for Artificial Intelligence*, pages 121–130. Springer, 2013.
35. V. Bolón-Canedo, N. Sánchez-Marono, and J. Cervino-Rabunal. Toward parallel feature selection from vertically partitioned data. In *Proceedings of ESANN 2014, pp. 395–400*, 2014.
36. Laura Morán-Fernández, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Data complexity measures for analyzing the effect of smote over microarrays. In *ESANN*, 2016.
37. A. de Haro Garcia. *Scaling data mining algorithms. Application to instance and feature selection.* PhD thesis, Universidad de Granada, 2011.
38. M. A. Hall and L. A. Smith. Practical feature subset selection for machine learning. In *Proceedings of the 21st Australasian Computer Science Conference ACSC 98*, pages 181–191. Springer, 1998.
39. C.E.Shannon. Mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
40. I. Kononenko. Estimating attributes: analysis and extensions of relief. *Machine Learning: ECML-94*, pages 171–182, 1994.
41. K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc, 1992.
42. M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th international Conference on Machine learning*, pages 856–863, 2000.
43. M. Dash, H. Liu, and H. Moto. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1-2):155–176, 2003.
44. M. Bramer. *Principles of Data Mining.* Springer, 2007.
45. V. Vapnik. *The nature of statistical learning theory.* Springer, 1999.
46. I. H. Witten and Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers Inc, 2005.
47. DG. Altman. *Practical Statistics for Medical Research.* Chapman & Hall, 1991.
48. M. Hollander and D.A. Wolfe. *Nonparametric statistical methods.* John Wiley New York, 1973.
49. J. Demšar. Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Author Biographies

José Luis Morillo-Salas received his M.S. (1998) in Computer Science from the Polytechnic University of Madrid (Spain). He has worked in the private sector and since 2003 he is an ICT administrator in the Spanish Government. His interest in computer science encouraged him in 2017 to research on machine learning and feature selection, being currently a PhD student at the University of A Coruña (Spain).

Verónica Bolón-Canedo received her B.S. (2009), M.S. (2010) and Ph.D. (2014) degrees in Computer Science from the University of A Coruña (Spain). After a Postdoctoral fellowship in the University of Manchester, UK (2015) she is currently teaching in the Department of Computer Science of the University of A Coruña. She has extensively published in the area of machine learning and feature selection. On these topics she has co-authored several books and book chapters, and more than 80 research papers in international conferences and journals.

Amparo Alonso-Betanzos received the Ph.D. degree for her work in the area of medical expert systems in 1988 at the University of Santiago de Compostela (Spain). Later, she was a postdoctoral fellow in the Medical College of Georgia, Augusta (USA). She is currently a Full Professor in the Department of Computer Science, University of A Coruña (Spain). Her main current areas are intelligent systems, scalable machine learning, explainable AI and feature selection.