On developing an automatic threshold applied to feature selection ensembles

B. Seijo-Pardo, V. Bolón-Canedo and A. Alonso-Betanzos[1]

Department of Computer Science - University of A Coruña
Campus de Elviña s/n 15071 - A Coruña, Spain

Abstract

Feature selection ensemble methods are a recent approach aiming at adding diversity in sets of selected features, improving performance and obtaining more robust and stable results. However, using an ensemble introduces the need for an aggregation step to combine all the output methods that conform the ensemble. Besides, when trying to improve computational efficiency, ranking methods that order all initial features are preferred, and so an additional thresholding step is also mandatory. In this work two different ensemble designs based on ranking methods are described. The main difference between them is the order in which the combination and thresholding steps are performed. In addition, a new automatic threshold based on the combination of three data complexity measures is proposed and compared with traditional thresholding approaches based on retaining a fixed percentage of features. The behavior of these methods was tested, according to the *SVM* classification accuracy, with satisfactory results, for three different scenarios: synthetic datasets and two types of real datasets (where sample size is much higher than feature size, and where feature size is much higher than sample size).

Keywords: Ensemble Learning, Feature Selection, Automatic Thresholding

1. Introduction

In recent years the size of the datasets used for machine learning has increased considerably, with the result that feature selection (FS) has become an essential preprocessing step for many data mining applications. Since FS reduces storage needs and removes irrelevant and redundant information, it improves the computational time needed for the machine learning algorithms. Several studies have demonstrated that FS can greatly improve the performance of subsequent classification [1, 2, 3]. Many approaches and algorithms [4, 1, 5] have been employed for this task, in the quest for more robust, compact and high-quality feature subsets.

To evaluate the features of a dataset, two different general approaches may be used: (i) individual evaluation and (ii) subset evaluation [6]. Individual eval-

uation methods, also known as rankers, assign a level of relevance to each feature and return an ordered ranking of all the features. Although this approach is not capable of eliminating redundant features, it notably improves computational performance over the subset evaluation approach. Subset evaluation generates successive subsets of features that are iteratively evaluated, using an optimality criterion, until the final subset of selected features is obtained. Although this approach has the advantage of detecting feature redundancy, it is computationally less efficient.

Although machine learning methods traditionally have used a single learning model to solve a particular problem, recently it has been shown that combining multiple different models can improve results. This approach, called ensemble learning, is based on the supposition that combining the output of multiple experts is better than using the output of any single expert [7, 8, 9]. Analogously, while FS is more frequently based on using a single algorithm, lately a few works have adopted the idea of ensemble learning for this task [10, 11, 12]. An ensemble for FS works by combining the outputs of several FS methods, aggregating partial results to obtain more robust and stable features for subsequent learning tasks. Two general strategies can be used to introduce the key concept of diversity in the ensembles. In the heterogeneous approach several different FS algorithms are used, whereas the traditional homogeneous approach uses different partitions of the training dataset fed to the same algorithm and producing different results that are also combined. This second strategy is the one exploited by the well-known bagging and boosting algorithms [13, 14]. Diversity and robustness are thus achieved through the use of multiple feature evaluation criteria [15]. Although both approaches—in which diversity is the key concept—are of interest, the heterogeneous strategy is of most interest when the user does not have the technical knowledge necessary to select the most suitable algorithm for their problem. Ensembles of filters have previously been used for different scenarios and also for different classifiers, with outputs combined by means of common simple voting [16, 17]. Ensembles of feature rankers have also been used for different applications [18, 19], with the single ranked features combined in a global ranking using different approaches. Other works propose a feature ranking scheme for an ensemble of multilayer perceptrons (MLPs) [20], applied with a stopping criterion based on the Out-of-Bootstrap (OOB) estimate [21].

In this study, the ensemble learning idea was applied to the FS process and different ensemble configurations and designs were executed and compared. An heterogeneous ensemble approach was implemented, aimed at reducing the variability induced by using individual FS methods and taking advantage of the strengths and overcoming the weaknesses of the individual methods. In addition, ranker methods were used to configure the FS ensemble, since rankers can reduce the size of data without compromising the time and memory requirements of machine learning algorithms.

Since we were working with rankings, at some point we needed to establish a threshold to retain only the relevant features and to combine the rankings obtained by the different methods configuring the ensemble. In this respect, the main novelty of our proposal herein is the use of two different models, depending

on whether thresholding was performed before or after combination (*Design TC* and *Design CT*). The performance of each model is analyzed and compared to the other according to the *SVM* classification accuracy. Since establishing an adequate threshold is not trivial, we also propose a methodology for establishing automatic thresholds based on measurements of data complexity [22] for feature rankings, both in *Design TC* and *Design CT*.

To sum up, the main contributions of our proposal are: (i) to free the user from having to select a specific FS method that works well with their dataset, given that most methods produce variable results depending on application characteristics; and (ii) to free the user from having to select a specific threshold and having to experiment with different percentages of retained features. The outcome is completely automatic FS methods that are independent of the nature of the dataset in that they obtain a generic threshold that runs smoothly in different scenarios and extracts the best subset of features from each dataset without having to pre-set threshold in feature percentages.

We experimented with a large and assorted suite of datasets, including artificial datasets, classical real datasets and microarray datasets. Based on our results, we state conclusions and propose guidelines of possible interest for future applications of ensembles for FS purposes.

The remainder of this paper is organized as follows. Section 2 describes the rationale under the design of the two ensemble approaches proposed; Section 3 is an introduction to the proposed method and its different components: ranker methods, combination (also called aggregation) methods, threshold values and classifier method used; Section 4 describes the proposed scenarios, experimental design and experimental results; and finally, Section 5 summarizes our conclusions and recommendations and proposes new lines of future work.

2. Information Fusion design

In this study an ensemble of FS methods was used with the aim of obtaining more consistent, efficient and robust solutions than those yielded by individual methods. Using an ensemble means that the performance variance of obtaining a single result is reduced; in addition, the combination of multiple subsets might help to remove less relevant features [10, 11, 12]. The approach also has the advantage of not requiring the user to understand the technical details of individual algorithms and their suitability for certain datasets. We tested different ensemble methods and different numbers of ranking techniques to configure an ensemble (described in [23, 11]), formed of six different FS methods—the combination that produced the best results.

There are several ways to design an ensemble [24] and the first decision is to select the FS methods. In our proposal, rankers were used since computational efficiency was our priority. The different FS rankers were individually applied to a particular dataset and the single final subset was obtained by combining the obtained outputs, for which reason a combination method was chosen. The use of rankers made it mandatory to apply a threshold to limit the number of selected features and so ensure efficiency in the subsequent learning methods.

Different designs were obtained depending on the order of the combination and thresholding operations. Finally, of other possibilities for the ensemble [24, 11], we opted for an ensemble of n different ranker methods applied to the same training data, with two different designs: (i) rankings combined before thresholding; and (ii) a threshold cutoff applied before combining rankings.

2.1. Design CT: combination followed by thresholding

The generic design of an ensemble of feature rankers is based on obtaining the result of each ranker method —an ordered ranking— using an aggregator to fuse the rankings into a single final ranking and subsequently applying a threshold cutoff to obtain a final practical subset of features [7]. The pseudo-code for this approach is given in Algorithm 1.

Algorithm 1: Pseudo-code for Design CT: combination followed by thresholding

Data: N — number of ranker methods
Data: T — number of features to be selected

Result: P — classification prediction

for *each n from 1 to N* **do**
  Obtain ranking R_n using ranker method r_n
end
R = Obtain the final ranking by joining all R_n rankings using the *Min* combination method.
T = Select a threshold value cutoff t from those available and apply.
S = Select the T top attributes from R.
Build the classifier with the selected attributes S.
Obtain prediction P.

2.2. Design TC: thresholding followed by combination

We redesigned the generic ensemble (i.e. *Design TC*) by reversing the order of the combination and thresholding steps. Therefore, the result of each ranker method was obtained as a first step, as in the generic design. A threshold cutoff was selected and applied to each single output to obtain individual partial subsets of features. Finally, these subsets were joined to achieve a single final subset of features. The pseudo-code for this approach is given in Algorithm 2.

3. Proposed methodology

As with most learning algorithms, each FS method has its strengths and weaknesses and performance depends on the characteristics of the datasets to which the method is applied. To optimize performance, some knowledge of existing algorithms is required to be able to select an appropriate method for a

Algorithm 2: Pseudo-code for Design TC: thresholding followed by combination

Data: N — number of ranker methods
Data: T_n — number of features to be selected

Result: P — classification prediction

```
for each n from 1 to N do
    Obtain ranking R_n using ranker method r_n
    T_n = Select a threshold cutoff t from those available and apply to each R_n.
    S_n = Select the T_n top attributes from each R_n.
end
C = Select a combination method.
S = Obtain the final subset by combining all S_n subsets using C.
Build the classifier with the selected attributes S.
Obtain prediction P.
```

particular dataset. One possible solution to this problem is to use an ensemble of FS methods that performs acceptably well, regardless of the nature of the problem. Several FS methods can be used, with their partial results combined as a single result. Several decisions need to be made, however, regarding the specific FS and combination methods to be used. Below we first describe rankers, combination methods, thresholds and classification methods and then we describe the ensemble designs derived from the two different combination and thresholding sequences.

3.1. Feature selection methods

Of the many FS methods described in the literature, four filter methods and two embedded methods were chosen. This set of ranker methods was selected because (i) they are amply used by researchers for FS purposes; and (ii) the fact that they are based on different metrics ensures diversity in the final ensemble:

- *Chi-Square* [25] (filter). This univariate filter, based on the χ^2 statistic, independently evaluates each feature with respect to the classes. The higher the chi square value, the more relevant the feature with respect to the class.
- *Information Gain (InfoGain)* [26] (filter). One of the most common univariate methods for attribute evaluation, this filter assesses features according to their information gain considering a single feature at a time.
- *Minimum Redundancy Maximum Relevance (mRMR)* [27] (filter). This filter uses mutual information to select the most relevant features for the target class that are also minimally redundant, i.e., it selects features that are maximally dissimilar.
- *ReliefF* [28] (filter). The original *Relief* filter [29] works by randomly sampling an instance from the dataset and then locating its nearest neighbor

from the same and opposite class. The values of the nearest neighbor attributes are compared to the sampled instance so as to update relevance scores for each attribute. The rationale is that a useful attribute should differentiate between instances from different classes and should have the same value for instances from the same class. *ReliefF* has the added ability of dealing with multiclass problems and is also more robust in dealing with incomplete and noisy data. This method can be applied in all situations, has low bias, includes interaction between features, and may capture local dependencies that other methods miss.

- *Recursive Feature Elimination for Support Vector Machines (SVM-RFE)* [30] (embedded). This embedded method trains an *SVM* classifier iteratively with the current set of features. The least important features are then removed by an *RFE* process using weight as the ranking criterion.

- *Feature Selection-Perceptron (FS-P)* [31] (embedded). This embedded method is based on a perceptron, a type of artificial neural network that can be viewed as a linear classifier, i.e., as the simplest kind of feedforward neural network. It consists of training a perceptron in a supervised learning context. Interconnection weights are used as indicators of the most relevant features for ranking.

In a previous publication [11, 12], a preliminary diversity study was carried out on two classical datasets called *SpamBase* and *Isolet* (described in Section 4.1 below). This study, aimed at checking whether the rankings obtained by the different methods were substantially different, consisted of comparing the rankings obtained by the FS methods forming the ensemble using Spearman's rank correlation coefficient [32].

The results obtained (with Spearman ρ values far from 1 reflecting equality in the rankings) pointed to a considerable difference between the partial rankings, indicating that the FS rankers chosen for the ensemble would be sufficiently diverse in their behavior.

3.2. Threshold values

As previously mentioned, the FS methods used in this study are all rankers (they sort all features) and so it was necessary to establish a threshold cutoff to obtain a practical subset of features. Most studies in the literature use thresholds that retain different percentages of features [33, 1]. Since threshold values are dependent on the particular dataset being studied, several attempts have been made to develop a general automatic threshold [34, 35, 12]. The idea of establishing an automatic threshold [23, 12] is based on using dataset complexity measures to obtain an optimal number of features to be used for subsequent classification purposes. That idea is taken up and expanded in this study for new scenarios and new designs. Note that the time required to calculate automatic thresholds is almost negligible, especially for datasets with few classes (see Section 4.2).

We carried out an exhaustive study that compared traditional fixed thresholds—taken as our baseline—with our proposed automatic approaches, both for *Design CT* and *Design TC*. The automatic approach for the thresholding step (which was already described for *Design CT* elsewhere [23]) can be applied similarly on *Design TC*. The approach individually calculates the complexity measure for each feature of the dataset and, finally, it establishes the final subset of features according to the following formula:

$$e = \alpha \times CM + (1 - \alpha) \times \rho \tag{1}$$

where α is a parameter with a value in the interval $[0, 1]$ that balances the importance of both the error obtained and the number of features retained, ($\alpha = 0.75$ empirically for this work), CM is one of the complexity measures described below (*F1*, *F2*, *F3* or *CF*) and ρ is the percentage of features retained with a value in the interval $[0, 1]$. In this work, the feature percentages were calculated for batches of $log_2(n)$ features. That is, we calculated e using only the first batch of $log_2(n)$ features, and then calculated e for the batch of $2 \times log_2(n)$, selecting the best result for both. A smaller complexity value e represents an easier problem.

Tested to delimit data dimensionality were seven different threshold values, four of which are automatic thresholds based on complexity measures:

- *Fixed thresholds. 50%*, *25%* and *10%* fixed thresholds were used to select the top 50%, 25% and 10% of the features of the ordered ranking respectively.

- *Maximum Fisher's discriminant ratio (F1).* This measure is defined for a multidimensional problem as:

$$F1 = \frac{\sum_{i=1,j=1,i\neq j}^{c} p_i p_j (\mu_i - \mu_j)^2}{\sum_{i=1}^{c} p_i \sigma_i^2}, \tag{2}$$

  where μ_i, σ_i^2, and p_i are the mean, variance and proportion of the ith class c, respectively. In this work the inverse of the Fisher ratio ($1/F1$) has been used, such that a small complexity value represents an easy problem.

- *Volume of overlap region (F2).* Let the maximum and minimum values of each feature f_i in class c_j be $max(f_i, c_j)$ and $min(f_i, c_j)$. The overlap measure *F2* is thus defined as:

$$F2 = \prod_i \frac{max(0, MINMAX_i - MAXMIN_i)}{MAXMAX_i - MINMIN_i}, \tag{3}$$

  where $i = 1, \ldots, d$ for a d-dimensional problem, and

$$MINMAX_i = MIN(max(f_i, c_1), max(f_i, c_2))$$
$$MAXMIN_i = MAX(min(f_i, c_1), min(f_i, c_2))$$
$$MAXMAX_i = MAX(max(f_i, c_1), max(f_i, c_2))$$
$$MINMIN_i = MIN(min(f_i, c_1), min(f_i, c_2))$$

For multiclass problems, *F2* is computed for each pair of classes, the absolute value is obtained for all of the classes and, finally, the product of all these values is returned as output. A low value for this measure means that the features can discriminate between the instances of different classes.

- *Maximum (individual) feature efficiency (F3).* In a procedure that removes unambiguous points falling outside the overlapping region in each dimension, the efficiency of each feature is defined as the fraction of all remaining points separable by that feature. The maximum feature efficiency *F3* is defined for a d-dimensional problem as:

$$F3 = \frac{|\bigcup_{i=1}^{d}\{x_{ji} \in [MINMAX_i, MAXMIN_i] : x_j \in D\}|}{s} \tag{4}$$

where x_j is each of the examples in the training set D, x_{ji} the value of example x_j for feature i, and s the total number of examples in the training set D.

For multiclass problems, *F3* is computed for each pair of classes, the absolute value for each is obtained and, finally, the maximum of all these values is returned. In this work the inverse of this measure ($1/F3$) has been used, such that a smaller complexity value represents an easier problem.

- *Complexity fusion (CF).* This automatic threshold selects an optimal number of features according to a combination of $1/F1$, *F2* and $1/F3$ complexity measures, using an average of the three as the complexity value:

$$CF = \frac{\frac{1}{F1} + F2 + \frac{1}{F3}}{3} \tag{5}$$

According to definition of complexity measures used in this calculation, a small complexity *CF* value represents an easy problem. In addition, due to *F1* measure can achieve any positive real value and *F2* and *F3* take values in the range $[0, 1]$, *CF* measure can obtain any real positive value, where *F1* measure value may dominate over the other two measures in the final result.

3.3. Combination methods

The different ranker methods had to be combined in order to produce a single final output. Therefore, according to when the union occurs—before or after thresholding—two different kind of combination stages were implemented:

- *Aggregation or ranking combination:* This approach was used on the *Design CT* ensemble, i.e. when the FS method outputs are combined before applying thresholding. This kind of combination method performs a fusion of several rankings using some reduction function. A single final ranking is thus the output of the union method that combines all the input rankings.

For this study, we used the *Minimum union method (Min)* to combine the different input rankings. This method, based on simple arithmetic operations, selects the minimum of the relevance values yielded by each ranking in the FS ensemble [36]. Despite its simplicity, this approach achieved the best results for DNA microarray datasets in a previous work [37].

The behavior of this method can be illustrated with a simple example. Imagine that we apply an ensemble of four different ranker methods to a dataset with five features $\{a, b, c, d, e\}$. As can be seen in Table 1, we obtain four different rankings of features $\{R_1, R_2, R_3, R_4\}$, one for each ranker method in the ensemble. The last column in the table shows the calculations made by the *Min* method, which computes the best value achieved by each ranking along the different rankings (where best means the highest position). Note that using this method can result in ties between features, so elements that are tied are returned to their original position. Thus, in this example, the *Min* method returns the ranking $\{a, b, e, c, d\}$.

Element	R_1	R_2	R_3	R_4	R_{Min}
a	1	3	1	1	1
b	2	1	2	5	1
c	3	5	3	4	3
d	4	4	5	3	3
e	5	2	4	2	2

Table 1: Example of how the *Min* 'aggregator' works with multiple rankings.

- *Subset combination:* This was used for the *Design TC* ensemble, i.e. when FS method outputs are thresholded before combination. In this case, subsets are fused and the ranking order is therefore not taken into account. Seven different combination methods were used, grouped into two categories: (i) methods that fuse all subsets (*U1-U6*); and (ii) methods that fuse the least complex subsets (*L3*). These methods are described as follows:

    1. *Fusion of six subsets (U1-U6).* The fusion of six subsets (a subset for each FS method) merges the features of as many subsets as the number of fusion methods (1 to 6). The fusion method *U1* obtains a final subset by fusing all the subsets, while *U6* obtains a final subset through the intersection of six subsets. Fusion methods *U2*, *U3*, *U4* and *U5* select the features that appear simultaneously in at least 2, 3, 4 and 5 subsets, respectively. So, $U_i \subseteq U_j, \forall i \geq j$ where $i, j = 1 \ldots 6$ and $U_i \neq \{\emptyset\}, i = 1 \ldots 6$ provided that the subset was selected at least by *ReliefF*. The rationale for this decision to use the *ReliefF* method as a base for the ensemble is that its use is recommended when the nature of the dataset is unknown [1].

The behavior of this method is illustrated with a simple example. Imagine that we apply an ensemble of six different ranker methods and an automatic threshold to a dataset with five features $\{a, b, c, d, e\}$. As can be seen in Table 2, we obtain six different subsets of features $\{S_1, S_2, S_3, S_4, S_5, S_6\}$, one for each method in the ensemble. Notice that S_4 corresponds to results obtained by *ReliefF* method and therefore they will be used as a base in the final results with the aim of avoiding empty subsets and improving the final results. The six columns in the right of the table illustrate the calculations made by the different *U1-U6* methods.

S_1	S_2	S_3	S_4	S_5	S_6	U1	U2	U3	U4	U5	U6
a	a	a	a	a	a	a	a	a	a	a	a
c	c	b	d	d	d	b	c	d	d	d	d
e	d	d			e	c	d	e	e		
	e	e				d	e				
						e					

Table 2: Example of how *Fusion of subsets (U1-U6)* works with multiple subsets.

2. *Fusion of the three least complex subsets (L3).* This combination method is based on selecting the three least complex subsets—according to *CF* complexity measure (see Section 3.2)—and joining them to select all their features.
   A simple example explains this method. Imagine that we apply an ensemble of six different ranker methods and a automatic threshold to a dataset with five features $\{a, b, c, d, e\}$. Each feature has a complexity value assigned according to *CF* complexity measure, and this value is used to calculate subset complexity. For this example the following values were set: $a = 0.5$, $b = 0.3$, $c = 0.7$, $d = 0.2$ and $e = 0.9$. Final subset complexity is obtained by adding the individual complexity value for each feature in the subset and dividing this value by the number of features in the subset (last row in Table 3). As can be seen in Table 3, we obtain six different subsets of features $\{S_1, S_2, S_3, S_4, S_5, S_6\}$ with six different subset complexity values, one for each method in the ensemble. Finally, the subsets with the three lowest complexity values (the shaded columns in the table) are joined to obtain a unique single subset, as illustrated in the last column of Table 3.

3.4. Final ensemble configuration

As indicated in Section 2, we developed two different ensemble designs, depending on the order of the combination and thresholding steps. *Design CT*, indicating the specific ranker methods, "aggregators" or combination methods and threshold values used in this study is depicted in Figure 1, while *Design TC* is depicted in Figure 2.

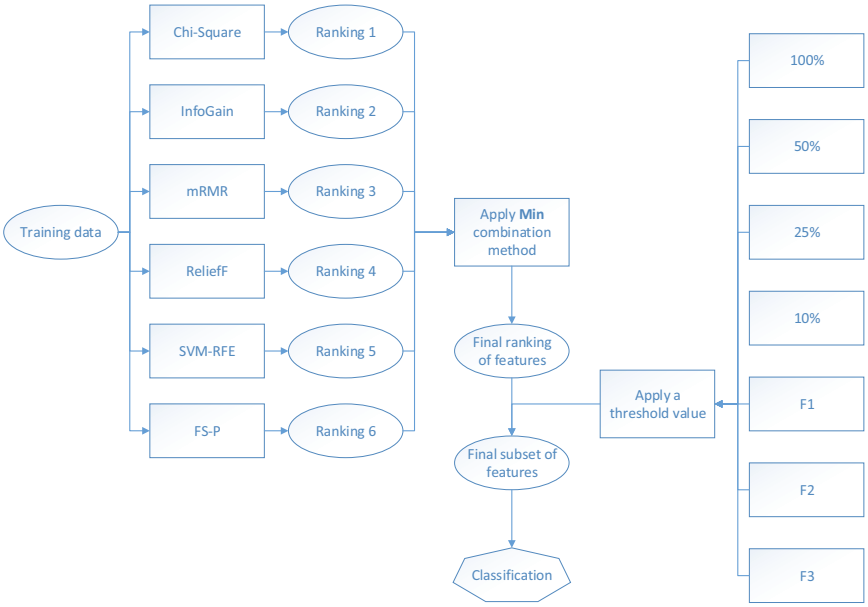

Figure 1: *Design CT*: combination followed by thresholding

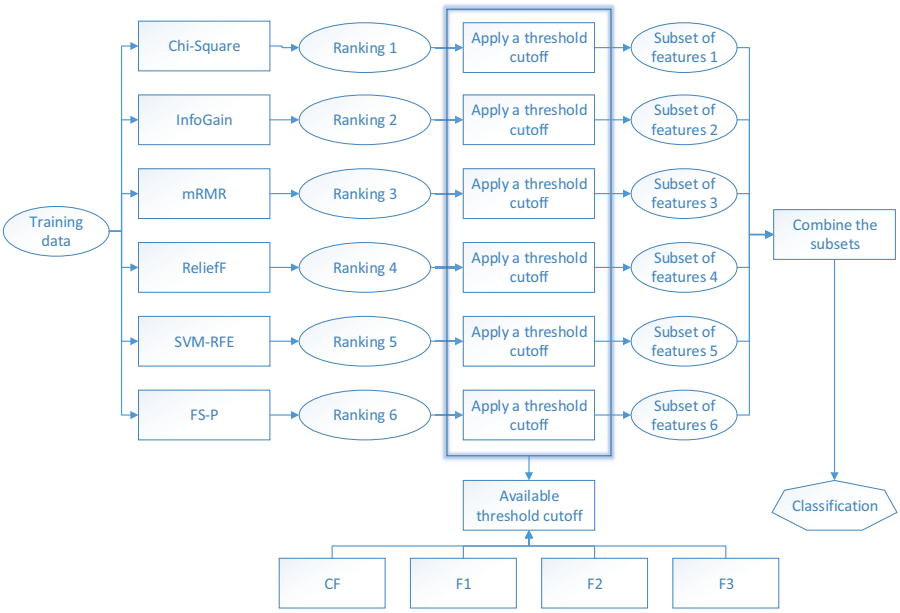

Figure 2: *Design TC*: thresholding followed by combination

S_1	S_2	S_3	S_4	S_5	S_6	L3
a	a	a	a	a	a	a
c	c	b	d	d	d	b
d	d	d		e	e	c
	e					d
0.47	0.58	0.33	0.35	0.53	0.53	0.43

Table 3: Example of how *Fusion of the three least complex subsets (L3)* works with multiple subsets.

3.5. Classification method

To compare the accuracy results obtained by our proposed methods, we used a *Support Vector Machine (SVM)* [4] algorithm. The *SVM* classifier is based on the idea of Structural Risk Minimization (SRM) [38], which has received much attention in recent years [39, 40], given that, in a large number of applications, it performs better than traditional learning techniques such as neural networks [39]. In a comparison of the performance of different classifiers for ensemble results [12], *SVM* showed the best outcomes, which is why it was used in this study. Note that the goal of this study was not to test the influence of the classifier, but to determine a suitable automatic threshold for rankings of features obtained by the ensemble.

3.6. Statistical tests

Statistical tests were conducted with the aim of comparing the results obtained by the different methods. We first generated the fitness measure F [41] that combines both classification test error (ϵ) percentages and proportions of selected features (ρ). The fitness of an individual method m, F_m, is given by:

$$F_m = \alpha(1-\epsilon) + (1-\alpha) \times (1-\rho), \tag{6}$$

where α ($0 \leq \alpha \leq 1$) is a parameter that balances the influence of both the above factors. In this study α was set to $\alpha = 5/6$, so as to give greater importance to classification error results than to the number of features.

The fitness measure F was used to obtain average ranking tables (Table 13 and Table 14) for the FS methods [42], based on statistical results from Friedman [43, 44], Bonferroni-Dunn [45], Holm [46], Hochberg [47] and Hommel [48].

We also used graphical representations of the Nemenyi test for post-hoc testing (critical difference diagrams [49]) so as to obtain a visual representation of the results (Figure 8 and Figure 16). In these diagrams, the top line represents the axis on which the average rankings of methods are drawn, with those appearing on the right hand side (lowest rankings) performing better. On comparing all the FS methods, the groups of methods that were not significantly different were connected. We also show the critical difference (CD) above the graph.

4. Experimental study

An exhaustive experimental study was carried out to test the threshold approaches in different scenarios.

4.1. Scenarios

Three different types of datasets were tested (synthetic, classical and microarray), so as to cover different scenarios in terms of, for instance, the presence of noise, correlation between features and ratios between sample and feature sizes. All these datasets have numerical attributes:

1. **Synthetic datasets (Type 1)**, for which feature relevance was known in advance. The correct output was used as a reference baseline and all samples were used as training data. Three different feature relevance categories were considered:
   - *Relevant:* A feature is relevant if its value varies systematically with category membership [50].
   - *Redundant:* A feature is redundant if it is highly correlated with one or more of the other features [51].
   - *Irrelevant:* A feature is irrelevant if it is not correlated with or predictive of the class; otherwise it is useful [51].

   There are several reasons for initially testing new FS methods on synthetic datasets [52]:
   - Controlled studies can be developed by systematically varying chosen experimental conditions, e.g. adding more redundant features in the input, as this tests the strengths and weaknesses of the algorithms.
   - Previous knowledge of optimal dataset features allows full control of the experimental conditions, meaning that the closeness to any solution can be assessed automatically and with confidence.

   In this study, eleven different synthetic datasets were analyzed, covering a large suite of problems (data nonlinearity, noise in the inputs and in the target, increasing number of irrelevant and redundant features, etc). Additionally, the fact that some of the datasets had a significantly higher number of features than samples implied an added difficulty for the FS methods. Parts of these datasets have been used previously [1, 53] to test methods and algorithms, which has the added advantage that evaluation can be performed regardless of the classifier used. Table 4 shows the different problems covered by each dataset, as well as the number of features and samples and the relevant attributes which should be selected by the FS methods (for further details see [1]). These datasets were used first in our experimental study to identify similar performances between methods and so reduce the number of subsequent experiments with real datasets.

Table 4: Synthetic datasets used in the experimental study.

Dataset	Samples	Features	Relevant features	Microarray
Corral-100	500	100	1-4	
Led-100[1]	500	100	1-7	
Monk1	500	100	1,2,5	
Monk2	500	100	1-6	
Monk3	500	100	2,4,5	
XOR-100	500	100	1,2	
Parity3+3	500	100	1-3	
SD1[2]	300	4000	G_1, G_2	✓
SD2[2]	300	4000	$G_1 - G_4$	✓
SD3[2]	300	4000	$G_1 - G_6$	✓

[1] Led-100 was used without noise in the inputs ($N = 0$) and with 15% noise ($N = 15$).

[2] G_i means that the FS method must select only one feature within the i-th group of features.

2. **Classical datasets (Type 2)**, for which the number of samples is higher than the number of features. Five popular datasets were chosen (Table 5). The numbers of samples and features range from 1 484 to 67 557 and from 8 to 617, respectively, and the datasets represent both binary and multiclass problems. These datasets also reflect issues that may arise in real problems, such as missing values or nonlinearity (as happens in *Spambase* and in *Madelon*, respectively). Ten-fold cross validation was applied to these datasets and the average error across all ten trials was computed.

Table 5: Classical datasets used in the experimental study.

Dataset	Samples	Features	Classes	Download
Spambase	4 601	57	2	UCI repository [54]
Madelon	2 400	500	2	UCI repository [54]
Connect4	67 557	42	3	UCI repository [54]
Isolet	7 797	617	26	UCI repository [54]
USPS	9 298	256	10	FS repository [55]

3. **Microarray datasets (Type 3)**, for which the number of features is much higher than the number of samples. Seven different DNA microarray datasets (Table 6) were tested. This kind of dataset is based on classifications of healthy and unhealthy patients when different tumor types are annotated as the output class. They represent a particular challenge for FS researchers due to the small sample size and the large number of gene expressions.

The datasets are available at `http://datam.i2r.a-star.edu.sg/datasets/krbd/`. K-fold was not performed on this dataset type due to the small number of samples. Datasets in the repository that were originally divided into training and test sets were maintained as such, while datasets with just a single training set were randomly divided for comparative purposes (using the common rule of thumb of 2/3 training and 1/3 testing data) and holdout validation was applied. Both binary and multiclass DNA microarray datasets were selected. Table 6 shows the number of features, samples and classes.

Table 6: DNA binary microarray datasets used in the experimental study.

Dataset	Samples		Features	Classes
	Train	Test		
Colon	42	20	2 000	2
DLBCL	32	15	4 026	2
Leukemia	38	34	7 129	2
Lung	32	149	12 533	2
Ovarian	169	84	15 154	2
11 Tumors	116	58	12 533	11
Leukemia 2	48	24	11 225	3

4.2. Experimental procedure

The experimental procedure was divided into two main steps according to the ensemble design employed (*Design CT* or *Design TC*). In both designs default parameters ($C = 1$ and $gamma = 0.01$) were used for the *SVM* classifier with a radial basis function kernel (RBFK), given that the goal was not to obtain the lowest possible error, but to determine the combinations that behaved best for each dataset type and to compare different ensemble approaches. Using automatic thresholds (for which computation time is negligible) both frees the user from the highly time-consuming operations of selecting and calculating percentages for different fixed thresholds (see Table 7 for an example) and also eliminates classifier dependency.

4.2.1. Design CT

The experimental procedure in this case was as follows:

1. Individually implement the six FS methods (see Section 3.1).
2. Apply the six FS methods to obtain the six different rankings.
3. Merge the rankings using the *Min* combination method (see Section 3.3) to obtain a final practical ranking.
4. Obtain a final ranking of the practical features subset according to a threshold cutoff (see Section 3.2).
5. Use the *SVM* classifier (see Section 3.5) and estimate the test error to check the suitability of the approach.

Table 7: Example computation times for automatic thresholds. Times are displayed in seconds (s.). T_f indicates the time required to calculate the corresponding F measure, T_{th} represents the time required to calculate the threshold cutoff and T_{tot} shows the total time required (sum of the two previous times).

Dataset	Method	T_f (s.)	T_{th} (s.)	T_{tot} (s.)
Connect4	F1	0.077	0.023	0.100
	F2	0.061	0.018	0.079
	F3	0.067	0.015	0.082
Isolet	F1	8.116	0.023	8.139
	F2	2.216	0.018	2.234
	F3	3.238	0.015	3.253

6. Perform statistical tests to better understand the classification results.

For the analysis of the synthetic (Type 1) datasets, only steps $1-4$ of the experimental procedure were implemented, with the aim of comparing the different ordered rankings of features. The procedure was fully implemented for Type 2 (classical) and Type 3 (microarray) datasets, using the corresponding validation scheme (i.e., ten-fold for Type 2 and train/test for Type 3).

Table 8 summarizes the ranking methods, combination methods and threshold cutoffs used in this study for each step of the *Design CT* ensemble.

Table 8: Summary of FS methods for *Design CT*

FS methods	Combination methods	Threshold cutoffs
Chi-Square	Min	50 %
InfoGain		25 %
mRMR		10 %
ReliefF		F1
SVM-RFE		F2
FS-P		F3
		CF

4.2.2. Design TC

The experimental procedure used was as follows:

1. Individually implement the six FS methods (see Section 3.1).
2. Apply the six FS methods to obtain the six different rankings.
3. Obtain a practical subset of features for each ranking according to a threshold cutoff (see Section 3.2).
4. Merge the subsets using the different combination methods (see Section 3.3) to obtain a final practical subset.

5. Apply the *SVM* classifier (see Section 3.5) and estimate the test error to check the suitability of the approach.
6. Perform statistical tests to better understand the classification results.

As in Subsection 4.2.1, the procedure was applied to Type 2 and 3 datasets using the corresponding validation scheme (i.e., ten-fold for Type 2 and train/test for Type 3). To analyze Type 1 datasets, only steps $1-4$ of the experimental procedure were implemented, with the aim of comparing the different subsets of features.

Table 9 summarizes the methods used in this study for each step of the *Design TC* ensemble.

Table 9: Summary of the FS methods available for *Design TC*

FS methods	Threshold cutoffs	Combination methods
Chi-Square	F1	U1
InfoGain	F2	U2
mRMR	F3	U3
ReliefF	CF	U4
SVM-RFE		U5
FS-P		U6
		L3

4.3. Synthetic dataset results

The results obtained for synthetic datasets are shown in Tables 10 and 11. The columns under each dataset show the number of relevant features (R) and irrelevant features (I) selected. Also included for each method as an evaluator of FS effectiveness is an index of success ([1]), labeled *Suc*:

$$suc. = \left[\frac{R_s}{R_t} - \alpha\frac{I_s}{I_t}\right] \times 100, \tag{7}$$

where R_s is the number of relevant features selected, R_t is the total number of relevant features, I_s is the number of irrelevant features selected and I_t is the total number of irrelevant features. The term α, introduced to weight the choice of an irrelevant feature over the exclusion of a relevant feature, is defined as $\alpha = avg\{\frac{1}{2}, \frac{R_t}{I_t}\}$. The index of success (maximum 100, with higher values indicating better methods) attempts to reward the selection of relevant features and to penalize the selection of irrelevant ones, penalizing two situations in particular:

- An incomplete solution: relevant features are excluded.
- An erroneous solution: irrelevant features are included.

The results in each table are divided into rows according to the thresholds and combination methods used. It can be observed that the *F1*, *F2* and *F3* blocks contain the same seven methods, with the first method corresponding to *Design CT* and the last six methods corresponding to *Design TC*. Finally, the last block, called *CFL3*, is the new approach applied to *Design TC*, based on combining the least complex subsets.

One of the first conclusions that can be drawn is that *F2* and *F3* results are the same for all the synthetic datasets studied, indicating that both complexity measures yield the same type of information. This simplifies subsequent analysis of Type 2 and Type 3 datasets, for which only the results for one of the complexity measures, *F2* or *F3*, will be shown.

Regarding the combination methods, it can be observed that the *Min* method applied to *Design CT* obtained the best results for different automatic thresholds. Also, when the more restrictive thresholds (*U4*, *U5* or *U6*) were applied to *Design TC*, better results were obtained for the synthetic datasets. When a threshold such as *U1* was used, it included all the irrelevant features that the different methods selected (usually not the same features), resulting in a relatively large number of irrelevant features. However, when more restrictive thresholds were used, features common to all the methods were usually selected as relevant, while irrelevant features were not selected. This would suggest the advisability of using a threshold such as *U3* or *U4* to avoid extreme cases.

The situation with the SD datasets is rather special. With the least complex dataset, SD1, the trend was the same as with the remaining synthetic datasets, so a good threshold option might be *U3* or *U4*, as these would reduce the number of irrelevant features without excluding relevant features. However, as the complexity of the classification problem increases (as happened with SD2 and SD3), using a restrictive threshold would seem to lead to some relevant features being excluded. In these cases, it would seem advisable to use a threshold such as *U1* to avoid excluding any relevant features.

As can be seen, the issue of selecting an appropriate threshold for all the datasets is far from trivial. Focusing on *Design CT*, the *Min* combination method worked reasonably well for the different thresholds (it always got the best result), but for *Design TC*, it was necessary to develop new subset combination methods further. The new *CFL3* measure was investigated in terms of finding a threshold independent of any particular dataset, so as to improve the stability and robustness of the other combination approaches. This measure used the three complexity measures (*F1*, *F2* and *F3*) to implement the threshold cutoff and then combined the three least complex subsets to achieve a final practical subset. As can be seen in Table 10, the *CFL3* method obtained a slightly poorer index of success (*Suc*) than the best methods (*U4*, *U5* and *U6*). For complex datasets, such as SD1, SD2 and SD3 (see Table 11), the *CFL3* method achieved better results than *U4*, *U5* and *U6*. The *CFL3* method was studied for real datasets since it obtained stable results for the different synthetic datasets, independently of their nature.

Table 10: Results for synthetic datasets. C indicates whether the correlated feature was selected (✓) or not selected (x), R is the relevant features selected, I is the number of irrelevant features selected and *Suc* is the index of success.

Method	Corral100				Led100 - N0%			Led100 - N20%			Monk1			Monk2			Monk3			Xor100			Parity3+3		
	R	C	I	Suc	R	I	Suc	R	I	Suc	R	I	Suc	R	I	Suc	R	I	Suc	R	I	Suc	R	I	Suc
MINF1	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
F1U1	1-4	✓	9	97.18	1-7	0	100	1-7	1	99.69	1,2,5	33	90.97	3,6	25	25.84	2,5	24	60.10	1,2	36	90.44	1,2	24	60.10
F1U2	1-4	✓	4	98.59	1-7	0	100	1-7	0	100	1,2,5	29	92.06	3,6	18	27.94	2,5	16	62.29	1,2	35	90.71	1,2	22	60.65
F1U3	1-4	✓	3	98.87	1-7	0	100	1-7	0	100	1,2,5	25	93.16	3,6	9	30.63	2,5	12	63.38	1,2	29	92.30	1,2	5	64.48
F1U4	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	25	93.16	3,6	5	31.83	2,5	12	63.38	1,2	27	92.83	1,2	6	65.02
F1U5	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	25	93.16	3,6	5	31.83	2,5	12	63.38	1,2	27	92.83	1,2	6	65.02
F1U6	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	25	93.16	3,6	5	31.83	2,5	12	63.38	1,2	26	93.10	1,2	5	65.30
MINF2	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
F2U1	1-4	✓	9	97.18	1-7	0	100	1-7	1	99.69	1,2,5	19	94.80	3,6	22	26.74	2,5	21	60.92	1,2	18	95.22	1,2	19	61.47
F2U2	1-4	✓	4	98.59	1-7	0	100	1-7	0	100	1,2,5	12	96.72	3,6	16	28.53	2,5	12	63.38	1,2	17	95.49	1,2	16	62.29
F2U3	1-4	✓	3	98.87	1-7	0	100	1-7	0	100	1,2,5	8	97.81	3,6	7	31.23	2,5	5	65.30	1,2	10	97.34	1,2	7	64.75
F2U4	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	5	98.63	3,6	5	31.83	2,5	5	65.30	1,2	6	98.41	1,2	6	65.02
F2U5	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	6	98.41	1,2	6	65.02
F2U6	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
MINF3	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
F3U1	1-4	✓	9	97.18	1-7	0	100	1-7	1	99.69	1,2,5	19	94.80	3,6	22	26.74	2,5	21	60.92	1,2	18	95.22	1,2	19	61.47
F3U2	1-4	✓	4	98.59	1-7	0	100	1-7	0	100	1,2,5	12	96.72	3,6	16	28.53	2,5	12	63.38	1,2	17	95.49	1,2	16	62.29
F3U3	1-4	✓	3	98.87	1-7	0	100	1-7	0	100	1,2,5	8	97.81	3,6	7	31.23	2,5	5	65.30	1,2	10	97.34	1,2	7	64.75
F3U4	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.63	3,6	5	31.83	2,5	5	65.30	1,2	6	98.41	1,2	6	65.02
F3U5	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	6	98.41	1,2	6	65.02
F3U6	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
MINCF	1-4	✓	2	99.15	1-7	0	100	1-7	0	100	1,2,5	4	98.91	3,6	5	31.83	2,5	5	65.30	1,2	5	98.67	1,2	5	65.30
CFL3	1-4	✓	4	98.59	1-7	0	100	1-7	0	100	1,2,5	10	97.26	3,6	12	29.73	2,5	10	63.76	1,2	12	96.81	1,2	11	63.66

Table 11: Results for synthetic datasets. (#) is the number of features selected, OPT(x) is the number of selected features within the optimal subset, where x indicates the optimal number of features, Rd is the number of redundant features, I is the number of irrelevant features and *Suc* is the index of success.

Method	SD1					SD2					SD3				
	(#)	OPT(2)	Rd	I	Suc	(#)	OPT(4)	Rd	I	Suc	(#)	OPT(6)	Rd	I	Suc
MINF1	12	1	5	6	49.93	12	3	7	2	74.94	12	4	7	1	66.62
F1U1	32	1	9	22	49.81	40	3	18	19	74.77	61	4	35	22	66.31
F1U2	15	1	9	5	49.91	27	2	18	7	49.84	34	4	26	4	66.48
F1U3	12	1	9	2	49.93	24	2	18	4	49.86	18	4	13	1	66.58
F1U4	12	1	9	2	49.93	24	2	18	4	49.86	14	3	10	1	49.93
F1U5	12	1	9	2	49.93	24	2	18	4	49.86	12	2	9	1	33.27
F1U6	12	1	9	2	49.93	24	2	18	4	49.86	12	2	9	1	33.27
MINF2	12	1	5	6	49.93	12	3	7	2	74.94	12	4	7	1	66.62
F2U1	32	1	9	22	49.81	31	3	12	16	74.82	44	4	27	13	66.42
F2U2	15	1	9	5	49.91	16	2	10	4	49.91	29	4	21	4	66.51
F2U3	12	1	9	2	49.93	13	2	10	1	49.93	15	3	11	1	49.92
F2U4	12	1	9	2	49.93	13	2	10	1	49.93	13	2	10	1	33.26
F2U5	12	1	9	2	49.93	13	2	10	1	49.93	12	2	9	1	33.27
F2U6	12	1	9	2	49.93	13	2	10	1	49.93	12	2	9	1	33.27
MINF3	12	1	5	6	49.93	12	3	7	2	74.94	12	4	7	1	66.62
F3U1	32	1	9	22	49.81	31	3	12	16	74.82	44	4	27	13	66.42
F3U2	15	1	9	5	49.91	16	2	10	4	49.91	29	4	21	4	66.51
F3U3	12	1	9	2	49.93	13	2	10	1	49.93	15	3	11	1	49.92
F3U4	12	1	9	2	49.93	13	2	10	1	49.93	13	2	10	1	33.26
F3U5	12	1	9	2	49.93	13	2	10	1	49.93	12	2	9	1	33.27
F3U6	12	1	9	2	49.93	13	2	10	1	49.93	12	2	9	1	33.27
MINCF	12	1	5	6	49.93	12	3	7	2	74.94	12	4	7	1	66.62
CFL3	14	1	9	2	49.93	13	2	10	1	49.93	23	3	19	1	49.87

4.3.1. Redundant features

We also studied how redundant features were handled by the ensemble approach compared to the individual ranking methods. The behavior of each approach was illustrated using a small synthetic dataset with a total of 100 samples and 10 features: relevant, features 1-3; redundant in relation to the previous features, features 4-6; and irrelevant, features 7-10. Applied to this dataset, with the aim of obtaining a final ranking or a final subset of features, was each ranking method (Section 3.1) and each ensemble design (Section 2).

Table 12 shows the results achieved by each method, with redundant features highlighted in boldface. Note that, since feature 1 was redundant with feature 4, 2 with 5 and 3 with 6, it was irrelevant which of the two features was selected. Redundancy only exists if two related features are selected. For example, the *F1U2* method selected features 2, 3, 4, 5, 6, 9 and 10. In this case feature 4 was relevant (since the method did not select feature 1) while 5 and 6 were redundant with features 2 and 3. All the features were ordered according to their relevance in ranking-type methods, while the final set of selected features were indicated in subset-type methods. As can be seen in Table 12, although some individual ranking methods can deal with redundancy (*mRMR* and *SVM-RFE*), this capacity is lost once these methods are combined to configure the final ensemble. Thus, ensemble approaches are better suited to capturing relevance rather than redundancy.

4.4. Classical dataset results

For experiments with classical datasets (datasets where sample size is larger than feature size), average test error percentages and the number of selected features were compared for *Design CT* and *Design TC*. Results are shown in Figures 3, 4, 5, 6 and 7. Since the number of experimental results is very high, we only report graphical results (see `http://www.lidiagroup.org/index.php/en/materials-en.html` for detailed tables).

Each figure shows the results achieved for a specific dataset; shown on the left side is the average test error percentage obtained by the *SVM* classifier and on the right side, the number of features selected by each ensemble configuration. Two different textures are reflected in each figure: striped bars to represent *Design CT* methods and solid bars to represent *Design TC* methods. Note that the $log_2(n)$ scale is used on the right side to improve visualization of the figures. As *F2* and *F3* threshold cutoffs obtained the same results as for synthetic datasets (see Section 4.3), for the sake of brevity only the results achieved by *F2* are shown in this section. The best results are highlighted in orange in each figure.

Figure 3 shows the results obtained for the *Spambase* dataset. The best result was achieved by the *Design TC* ensemble with the *F1* threshold cutoff, whereas the poorest results were achieved by *F2U5* and *F2U6*. This dataset did not show any substantial classification improvement when FS was applied, although the dimensionality was reduced for slightly better accuracy. Finally, for the *CFL3* cutoff method, the relationship between the number of features and

Table 12: Results for dealing with redundancy in a small synthetic dataset. Features 1-3 are relevant, 4-6 are redundant and 7-10 are irrelevant. Features are ordered according to their relevance in ranking-type methods and the subset of final selected features is indicated for subset-type methods. Redundant features are highlighted in boldface.

Method	Type	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Chi-Square	Ranking	10	3	2	9	**4**	**5**	**6**	7	8	1
InfoGain	Ranking	10	3	2	9	**4**	**5**	**6**	7	8	1
mRMR	Ranking	2	8	7	1	9	10	3	**5**	**4**	**6**
ReliefF	Ranking	**5**	2	**6**	3	**4**	1	8	7	9	10
SVM-RFE	Ranking	**5**	**6**	**4**	2	3	1	7	9	8	10
FS-P	Ranking	2	**5**	1	**4**	3	**6**	7	8	9	10
MINF1	Ranking	2	**5**	10	3	**6**	8	1			
F1U1	Subset	1	2	3	**4**	**5**	**6**	7	8	9	10
F1U2	Subset	2	3	4	**5**	**6**	9	10			
F1U3	Subset	2	4	**5**	6						
F1U4	Subset	2	**5**	6							
F1U5	Subset	2	**5**	6							
F1U6	Subset	2	**5**	6							
MINF2	Ranking	2	**5**	10							
F2U1	Subset	1	2	3	**4**	**5**	**6**	7	8	10	
F2U2	Subset	2	3	**5**	**6**	10					
F2U3	Subset	2	**5**	6							
F2U4	Subset	2	**5**	6							
F2U5	Subset	2	**5**	6							
F2U6	Subset	2	**5**	6							
MINCF	ranking	2	5	10							
CFL3	Subset	2	3	**5**	8	10					

percentage test error obtained was better than when compared to not using FS; however, when compared to the *F1* threshold, and even though fewer features were used, the error increased slightly.

Figure 4 shows the results obtained for the *Madelon* dataset, where the best result was achieved, again, by the *Design TC* ensemble, whereas the poorest result was achieved by the 100% threshold, i.e. when there was no previous FS. In contrast, both the *F1* and *F2* threshold cutoffs produced similar results, with the *U2*, *U3* and *U4* combination methods obtaining the best results. Finally, the *CFL3* method obtained the best test error results too, although with slightly higher dimensionality.

Figure 5 shows the results obtained for the *Connect4* dataset. Accuracy results improved as dimensionality was reduced, and *Design TC* improved on the dimension-error relationship with respect to *Design CT*. The best results were obtained by the *U2* combination method, for both the *F1* and *F2* thresholds.

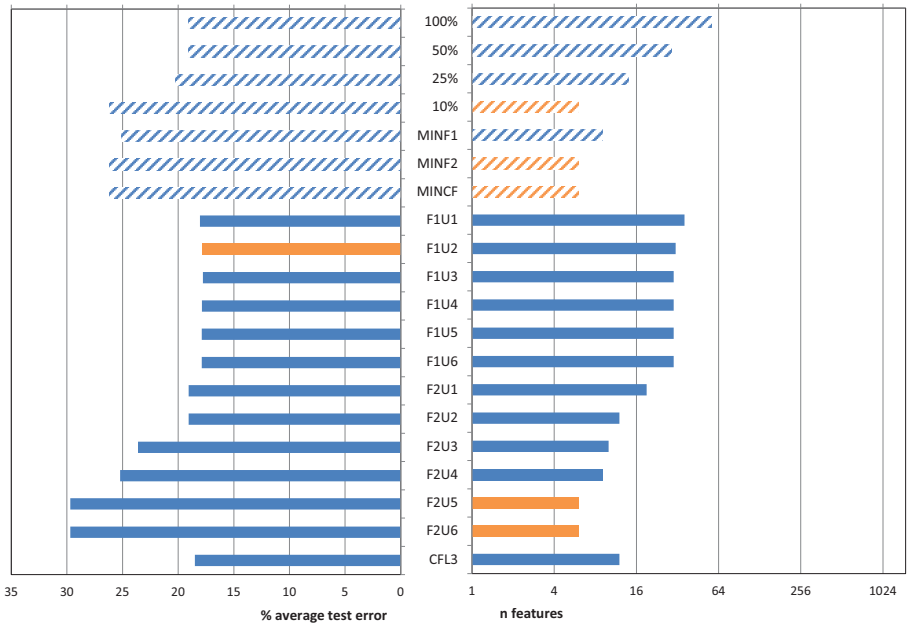

Figure 3: Results for the *Spambase* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

The *U3* and *U4* combination methods and the *CFL3* ensemble achieved very similar percentage test errors while slightly reducing dimensionality. It can be observed, at this point, that real datasets with a small number of classes appear to take advantage of the automatic threshold for all designs, as they not only improve accuracy but also significantly reduce the dimensionality of the problem.

Finally, two datasets with a large number of classes were included in the study: *Isolet* and *USPS*. For *Isolet* (Figure 6), with 26 different classes, only *Design CT* using a fixed 50% threshold obtained better results than if the complete dataset (without FS) were used.

Finally, the results for the *USPS* dataset, with 10 different classes, are shown in Figure 7. Clearly, FS worsens the results of the classification for all the combinations tested. As in the preceding dataset, automatic thresholds results were poorer than fixed threshold results in most cases. This would suggest that, for datasets with a large number of classes, the use of automatic thresholds based on complexity measures is not recommended.

To better understand the aforementioned results, the fitness measure F (explained in Section 3.6) was calculated for different FS methods and datasets. This measure was used for two different post-hoc tests:

- Table 13 shows a ranking of the FS methods, where a higher rank reflects

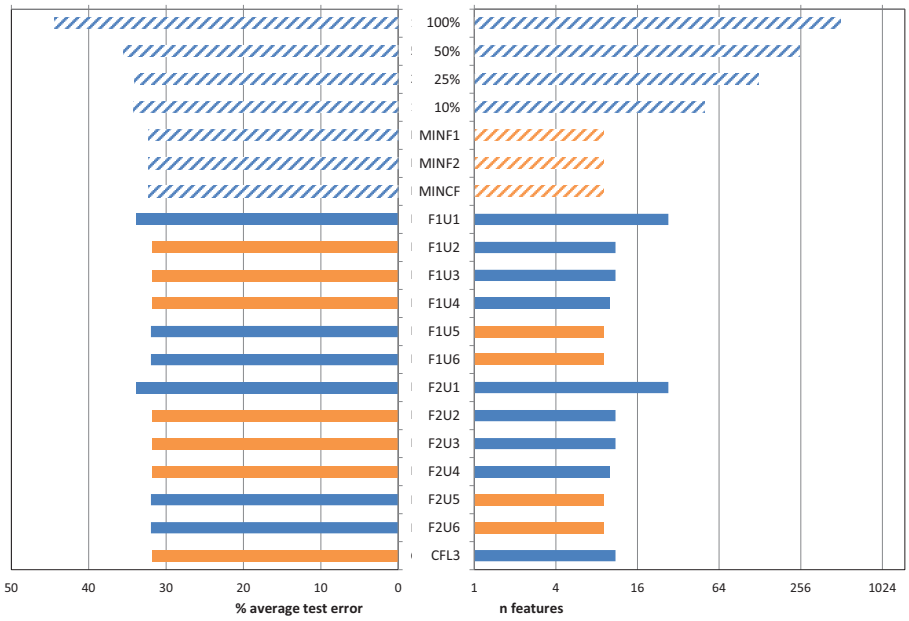

Figure 4: Results for the *Madelon* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

a better method.

- Figure 8, constructed with the aim of obtaining a visual representation of the results, shows the CD between FS methods (see explanation in Section 3.6).

Table 13 and Figure 8 illustrate the performance of different FS methods and ensemble configurations for a classical dataset context. If we focus only on *Design CT* methods, *25%* and *10%* thresholds obtain higher ranking positions and better CD values than the automatic thresholds, but always worse than *Design TC* performance. This is due to the good error results obtained by the fixed thresholds for the *Isolet* and *USPS* datasets, which have a large number of classes (26 and 10, respectively). It should be noted that *CFL3* method achieves the best ranking value according to Table 13.

We draw a number of general conclusions as follows:

1. *Design TC* appears to be a better ensemble option than *Design CT*, since the results obtained not only reduce the test error but also result in a smaller number of features.
2. Automatic thresholds are a good option when the number of different classes in the dataset is small. In this study, the automatic approach

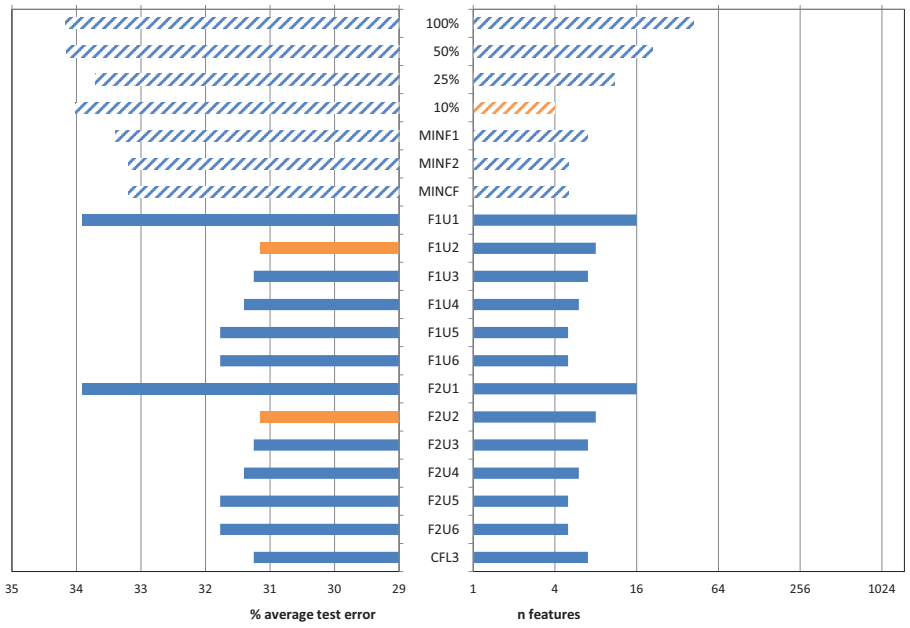

Figure 5: Results for the *Connect4* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

worked well for datasets with 2 or 3 classes, with performance deteriorating as the number of classes increased.

3. The experiments would suggest the use of *U2* or *U3* as combination methods when applying the *Design TC* ensemble.
4. Although the *CFL3* threshold method obtained a poor result for the *USPS* dataset, for the other four cases performance was good, robust and stable and always improved on or matched several configurations of the *Design TC* ensemble with a remarkable dimensionality reduction. As a result, *CFL3* reaches the best fitness measure F value and it ranks first in Table 13.

4.5. Microarray dataset results

The main characteristic of this type of dataset is that feature size is greater than sample size. The different datasets selected range from binary (*Colon* and *Ovarian*) to multiclass (*Leukemia 2*). As in the previous section, we compared average test error percentages and the number of selected features for the different ensemble configurations and designs (some of the results for *Design CT* ensemble have previously been published in [23]). Since the number of experimental results is very high, we only report graphical results (see `http://www.lidiagroup.org/index.php/en/materials-en.html` for detailed tables).

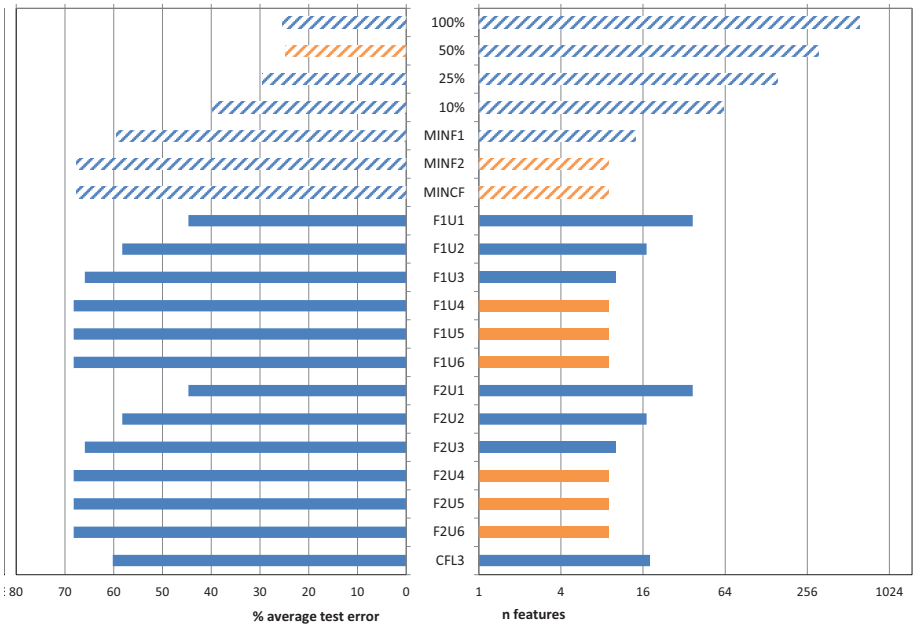

Figure 6: Results for the *Isolet* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

Figure 9 shows the results obtained for the *Colon* dataset. The best results (both in terms of accuracy and dimensionality reduction and using only 11 features out of the original 2000) were achieved by the *Design TC* ensemble with the *F1* and *F2* threshold cutoffs and the *U4*, *U5* and *U6* combination methods. The poorest results were achieved when we only used the classifier (no FS) and the fixed thresholding approaches.*MINCF* and *CFL3* methods obtained better results than the latter two approaches, but it was clearly outperformed by some automatic thresholds for *Design TC*.

Figure 10 shows the results obtained for the *DLBCL* dataset. Automatic thresholding with the *Design CT* ensemble obtained lower errors and a greater reduction in dimensionality than other thresholds. All *Design TC* ensemble methods achieved same test error results, poorest than *Design CT*, even though *F1* and *F2* automatic thresholds with *U4*, *U5* and *U6* combination methods perform a great dimensionality reduction, like *Design CT* methods.

Figures 11, 12 and 13, for the *Leukemia*, *Lung* and *Ovarian* datasets, respectively, confirm the trend evident in the two previously analyzed datasets.

Regarding the nature of the threshold cutoff, automatic thresholding provided better results than fixed thresholding for those three datasets. Of the automatic thresholds, the *Design TC* ensemble improved or equaled the *Design CT* ensemble for the three datasets. If only a single *Design TC* configuration

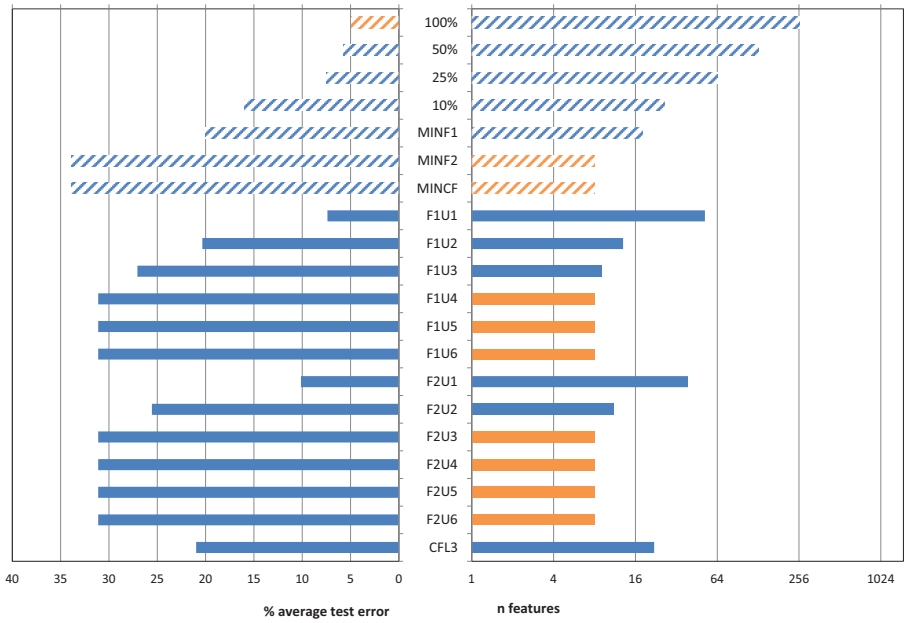

Figure 7: Results for the *USPS* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

that worked well on these datasets was to be highlighted, *CFL3* would be our choice, as it achieved the lowest percentage test error for the *Leukemia* dataset, and a percentage test error very close to the best for the *Lung* and *Ovarian* datasets. In terms of dimensionality reduction it should be noted that automatic thresholds with *Design CT* and *Design TC* with *U5* and *U6* combination methods achieve the best results in the five aforementioned datasets. Overall, automatic thresholds proved the best choice for binary microarray datasets since they not only improved the percentage error but also significantly reduced the dimensionality of the problem.

As in the previous classical scenario, an analysis of two multiclass microarray datasets was included in order to complete the study. Figure 14 and Figure 15 show results for the *11 Tumors* and *Leukemia 2* datasets, respectively. In both cases, one or more of the automatic thresholds outperformed the fixed thresholds. For *Leukemia 2*, there was a remarkable improvement in both accuracy and dimensionality reduction, especially for the *Design TC* ensemble in *Leukemia 2* dataset, which returned an error of only 4% (contrasting with the 54% error returned by the classifier without FS) and also considerably reduced the number of features used (only 13, versus the original 11 225). The problem was thus greatly simplified, was easier to interpret and visualize and was computationally far less burdensome.

FS method	Ranking value
CFL3	14.6
F2U2	14.0
25%	13.0
F1U2	12.2
F2U3	12.1
F1U3	11.9
F2U1	11.9
10%	11.6
MINF1	11.0
F2U4	10.9
F1U4	10.7
F1U1	9.7
F1U5	9.4
F1U6	9.4
50%	9.0
F2U5	8.4
F2U6	8.4
MINF2	7.7
MINCF	7.7
100%	6.4

Table 13: Average rankings for FS methods applied to classical datasets. A higher rank reflects a better method.

To summarize the aforementioned results, the fitness measure F (explained in Section 3.6) was calculated for different FS methods and datasets. This measure was used for two post-hoc tests:

- Table 14 shows a ranking of the FS methods, where a higher rank represents a better method.
- Figure 16, elaborated with the aim of obtaining a visual representation of the results, shows the CD between FS methods (see Section 3.6).

Referring to Table 14 and Figure 16, it would appear that *MINF1* is the best FS method for dealing with microarray datasets. According to Table 14, *F1U4* and *F2U4* are the best *Design TC* methods, but according to Figure 16, *F1U3* and *F2U3* seem to work slightly better than the previous ones. In any case, the most notable conclusion is that the fixed thresholds substantially underperformed the automatic thresholds.

We draw a number of general conclusions as follows:

1. Automatic thresholds would appear to be the best option for dealing with microarray datasets, where feature size is much greater than sample size. Compared to fixed thresholds, automatic thresholds are not only more accurate but are also capable of reducing dimensionality.

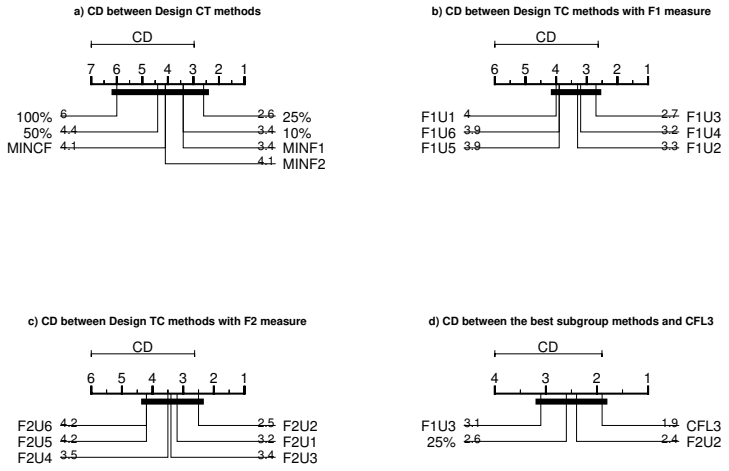

Figure 8: Critical differences between different FS methods applied to classical datasets. A lower value reflects a better method.

2. The *Design CT* ensemble would seem to be a better option than the *Design TC* ensemble, in view of its higher ranked results.
3. There are no significant differences in the results for the different *Design TC* methods.

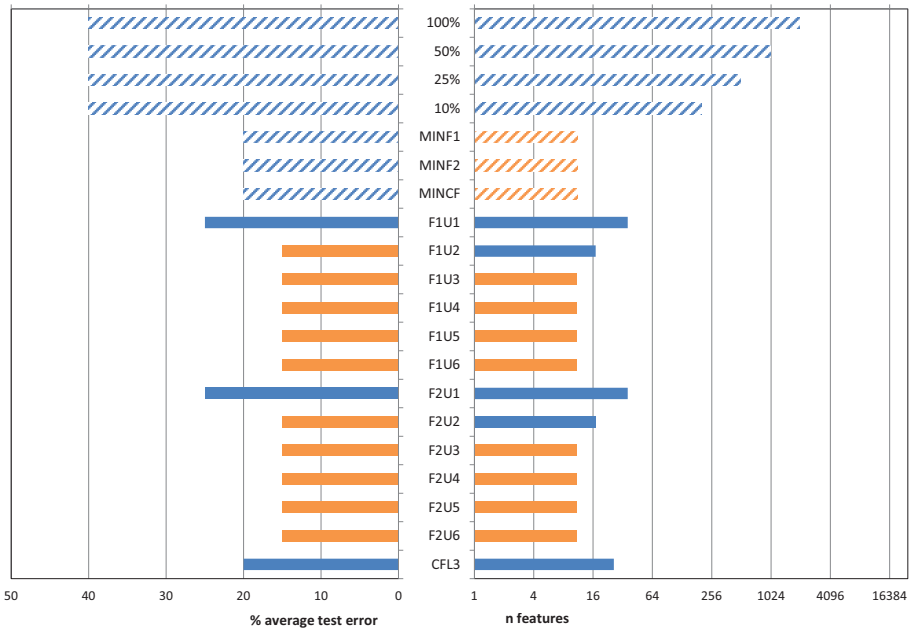

Figure 9: Results for the *Colon* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

5. Conclusions and future work

5.1. Lessons learned

In this study of ranker ensemble methods, two steps were needed to obtain a final result, namely, to combine partial results and to establish a threshold that retained only relevant features. Regarding the order of operations, we tested two designs: combination followed by thresholding (*Design CT*), and thresholding followed by combination (*Design TC*). Using different real datasets, we also tested fixed thresholds (established percentages of retained features) and a novel automatic threshold based on data complexity measures. Table 15 summarizes our recommendations regarding designs and methods suitable for different types of datasets, discussed as follows:

- On the basis of the study of a synthetic dataset (Section 4.3.1), even though an individual FS method might detect redundancy, the final aggregation step in the ensemble approach was not capable of reflecting this, indicating that ensemble approaches are better suited for capturing relevance than redundancy.

- Focusing on synthetic datasets, *Design CT* and the *Min* combination method achieved the best performance, both in the index of success (*suc.*)

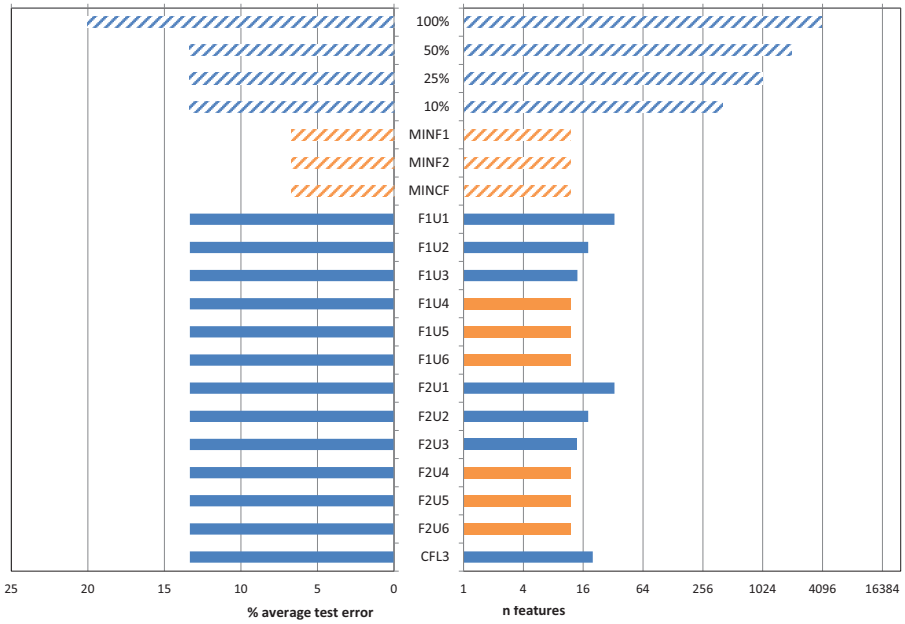

Figure 10: Results for the *DLBCL* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

measure and in the number of selected features. Moreover, results were the same using the *F1*, *F2* and *F3* automatic thresholds.

- Regarding real datasets, the best option would appear to be the novel *Design TC* ensemble, which first establishes the thresholds and then combines subsets of features. In general, this design obtained the lowest error rates and even used fewer features.

- The experimental results demonstrate that, in general, automatic thresholds perform better than fixed thresholds. Automatic thresholds also have the advantage of freeing the user from having to pre-select and test fixed percentages for a given classifier. In the case of datasets with a sample size greater than feature size (classical), this conclusion only seems to hold for a small number of classes (2-3), as performance seems to deteriorate as the number of classes increases. This may be due to the fact that the complexity measures used to establish automatic thresholds were developed specifically for binary and not multiclass problems. Therefore, for classical datasets with a large number of classes we would recommend using fixed thresholds. Finally, *CFL3* method is, generally speaking, robust and stable and does not perform significantly worse than the other methods.

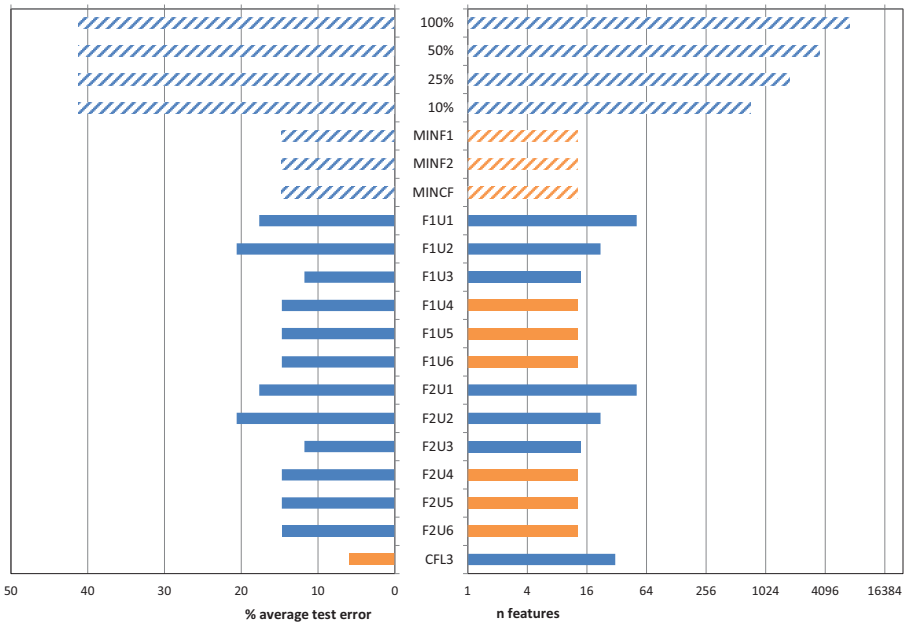

Figure 11: Results for the *Leukemia* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

5.2. Concluding remarks and future work

FS ensembles emerged with the aim of freeing the user from having to decide the best method for each particular problem, while also adding diversity, robustness and stability to the process. Several studies in the literature have already demonstrated the adequacy of using an FS ensemble instead of a single FS method [16, 56, 17, 37, 11]. In this study we propose an ensemble designed on ranker FS methods that requires combination and thresholding to be performed in one or the other order. We therefore tested two designs: combination before and after thresholding. As for decisions regarding the threshold, this is a problem that still has not been resolved by the research community. The typical approach is to choose a fixed threshold percentage (e.g. selecting the top 10% of ranked features), but the right percentage is very dependent on the nature of data. A possible solution is to use classification accuracy to evaluate the quality of subsets of features obtained after experimenting with different thresholds; however, this approach implies a significant computational burden, besides being highly dependent on the learning algorithm used. We used automatic thresholds (these can be adapted to the nature of the dataset without compromising the computational cost) and complexity measures (instead of classification accuracy) to evaluate the quality of the possible subsets of features resulting from establishing the threshold. We tested our approach

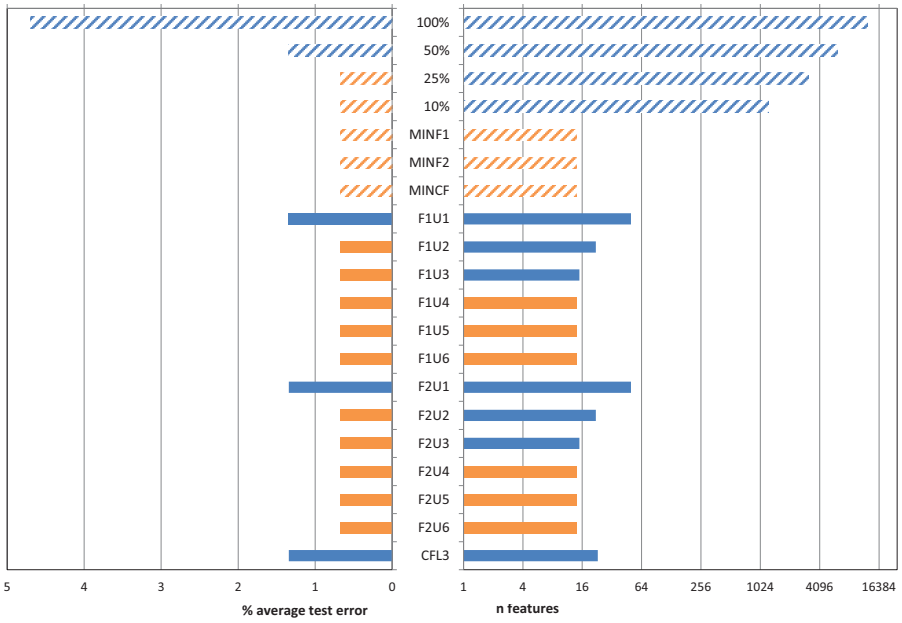

Figure 12: Results for the *Lung* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

on both classical datasets and microarray datasets, after first performing an exhaustive analysis of synthetic datasets to identify similar performances between methods and so reduce the number of subsequent experiments.

As future work, we plan to develop new methods for application to the combination phase of ensembles and also plan to apply more informed criteria to solving the ties between some combiners. Another potentially interesting new line of research would be to develop ensemble methods that could ensure the elimination of redundancy achieved by individual FS methods.

Acknowledgments

This research has been financially supported in part by the Spanish Ministerio de Economa y Competitividad (research project TIN 2015-65069-C2-1-R), by the the Xunta de Galicia (research projects GRC2014/035 and the Centro Singular de Investigación de Galicia, accreditation 2016-2019) and by the European Union (FEDER/ERDF).

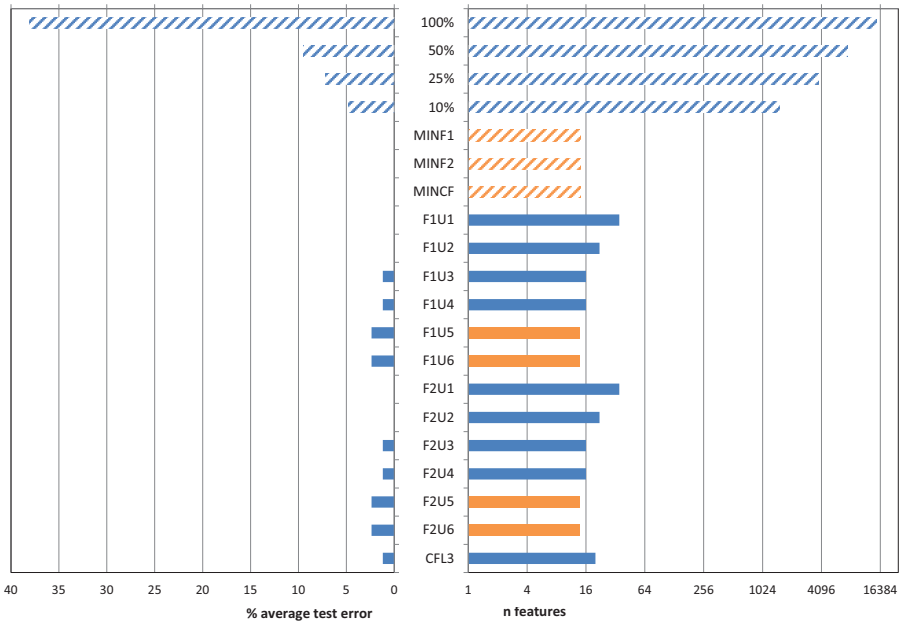

Figure 13: Results for the *Ovarian* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

References

[1] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, A review of feature selection methods on synthetic data, Knowledge and information systems 34 (3) (2013) 483–519.

[2] K. Gao, T. M. Khoshgoftaar, H. Wang, An empirical investigation of filter attribute selection techniques for software quality classification, in: Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on, IEEE, 2009, pp. 272–277.

[3] D. Rodríguez, R. Ruiz, J. Cuadrado-Gallego, J. Aguilar-Ruiz, Detecting fault modules applying feature selection to classifiers, in: Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on, IEEE, 2007, pp. 667–672.

[4] I. Guyon, Feature extraction: foundations and applications, Vol. 207, Springer Science & Business Media, 2006.

[5] V. Bolon-Canedo, N. Sanchez-Maroo, A. Alonso-Betanzos, Feature selection for high-dimensional data, Springer, 2015.

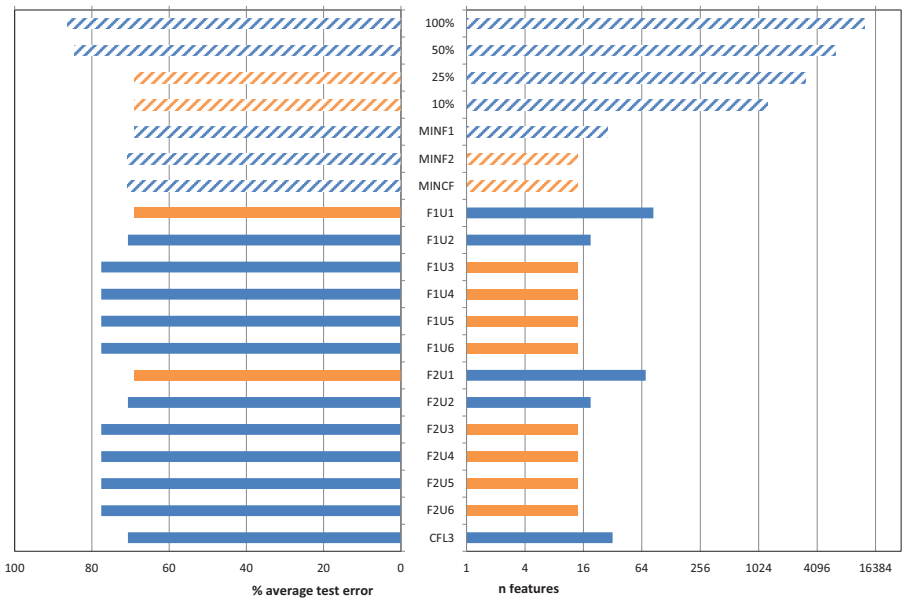

Figure 14: Results for the *11 Tumors* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

[6] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, The Journal of Machine Learning Research 5 (2004) 1205–1224.

[7] L. I. Kuncheva, Combining pattern classifiers: Methods and algorithms (kuncheva, li; 2004)[book review], Neural Networks, IEEE Transactions on 18 (3) (2007) 964–964.

[8] L. I. Kuncheva, C. J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine learning 51 (2) (2003) 181–207.

[9] Y. Ren, L. Zhang, P. N. Suganthan, Ensemble classification and regression-recent developments, applications and future directions [review article], IEEE Computational Intelligence Magazine 11 (1) (2016) 41–53.

[10] Q. Shen, R. Diao, P. Su, Feature selection ensemble., Turing-100 10 (2012) 289–306.

[11] B. Seijo-Pardo, I. Porto-Díaz, V. Bolón-Canedo, A. Alonso-Betanzos, Ensemble feature selection: Homogeneous and heterogeneous approaches, Knowledge-Based Systems 118 (2017) 124–139.

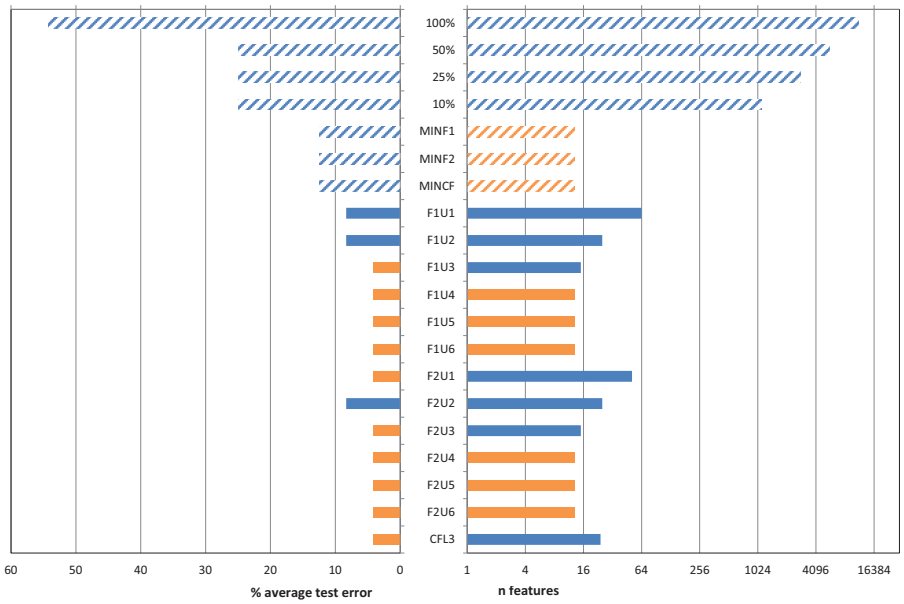

Figure 15: Results for the *Leukemia 2* dataset. Left side: average test error percentage obtained by the *SVM* classifier. Right side: number of features selected by each ensemble configuration. Striped bars represent *Design CT* methods and solid bars represent *Design TC* methods. The best results on each side are highlighted in orange.

[12] B. Seijo-Pardo, V. Bolón-Canedo, A. Alonso-Betanzos, Testing different ensemble configurations for feature selection, Neural Processing Letters (2017) 1–24.

[13] R. Schapire, The strength of weak learnability, Machine learning 5 (2) (1990) 197–227.

[14] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.

[15] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Diversity in search strategies for ensemble feature selection, Information fusion 6 (1) (2005) 83–98.

[16] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, An ensemble of filters and classifiers for microarray data classification, Pattern Recognition 45 (1) (2012) 531–539.

[17] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, Data classification using an ensemble of filters, Neurocomputing 135 (2014) 13–20.

[18] H. Wang, T. M. Khoshgoftaar, A. Napolitano, A comparative study of ensemble feature selection techniques for software defect prediction, in: Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on, IEEE, 2010, pp. 135–140.

FS method	Ranking
MINF1	14.6
MINF2	14.1
MINCF	14.1
F1U4	13.6
F2U4	13.6
F1U5	12.9
F1U6	12.9
F2U5	12.9
F2U6	12.9
F1U3	12.5
F2U3	12.5
CFL3	10.7
F1U2	10.6
F2U2	10.6
F2U1	9.8
F1U1	9.2
10%	5.1
25%	4.1
50%	2.0
100%	1.0

Table 14: Average rankings of FS methods applied to microarray datasets. Higher ranks represents better methods.

[19] J. Olsson, D. W. Oard, Combining feature selectors for text classification, in: Proceedings of the 15th ACM international conference on Information and knowledge management, ACM, 2006, pp. 798–799.

[20] T. Windeatt, R. Duangsoithong, R. Smith, Embedded feature ranking for ensemble mlp classifiers, IEEE Transactions on Neural Networks 22 (6) (2011) 988–994.

[21] T. Windeatt, M. Prior, Stopping criteria for ensemble-based feature selection, in: Multiple Classifier Systems, Springer, 2007, pp. 271–281.

[22] T. K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE transactions on pattern analysis and machine intelligence 24 (3) (2002) 289–300.

[23] B. Seijo-Pardo, V. Bolón-Canedo, A. Alonso-Betanzos, Using data complexity measures for thresholding in feature selection rankers, in: Conference of the Spanish Association for Artificial Intelligence, Springer, 2016, pp. 121–131.

[24] M. Bramer, Data for data mining, in: Principles of data mining, Springer, 2013, pp. 9–19.

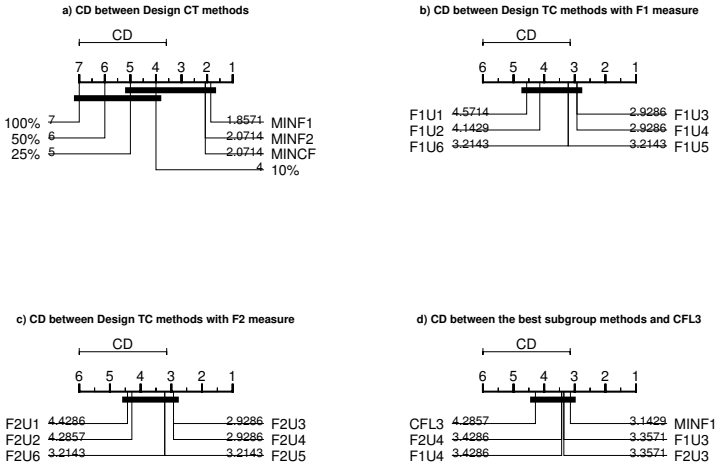

Figure 16: Critical differences between different FS methods applied to microarray datasets. Lower values represent better methods.

[25] H. Liu, R. Setiono, Chi2: Feature selection and discretization of numeric attributes, in: 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, IEEE Computer Society, 1995, pp. 388–388.

[26] J. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.

[27] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, Pattern Analysis and Machine Intelligence, IEEE Transactions on 27 (8) (2005) 1226–1238.

[28] I. Kononenko, Estimating attributes: analysis and extensions of relief, in: Machine Learning: ECML-94, Springer, 1994, pp. 171–182.

[29] K. Kira, L. A. Rendell, The feature selection problem: Traditional methods and a new algorithm, in: AAAI, 1992, pp. 129–134.

[30] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine learning 46 (1-3) (2002) 389–422.

Table 15: Summary recommendations for best designs and methods for different dataset types.

Scenario	Number of classes (C)	Design	Threshold method	Combination method
Classical	$C <= 3$	TC	CF	L3
	$C > 3$	CT	25%	MIN
Microarray	$C = 2$	CT	CF	MIN
	$C > 2$	TC	F2	U1

[31] M. Mejía-Lavalle, E. Sucar, G. Arroyo, Feature selection with a perceptron neural net, in: Proceedings of the international workshop on feature selection for data mining, 2006, pp. 131–135.

[32] S. B. Lyerly, The average spearman rank correlation coefficient, Psychometrika 17 (4) (1952) 421–428.

[33] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, Recent advances and emerging challenges of feature selection in the context of big data, Knowledge-Based Systems 86 (2015) 33–45.

[34] T. M. Khoshgoftaar, M. Golawala, J. Van Hulse, An empirical study of learning from imbalanced data using random forest, in: Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on, Vol. 2, IEEE, 2007, pp. 310–317.

[35] M. Mejía-Lavalle, E. Sucar, G. Arroyo, Feature selection with a perceptron neural net, in: Proceedings of the international workshop on feature selection for data mining, 2006, pp. 131–135.

[36] P. Willett, Combination of similarity rankings using data fusion, Journal of chemical information and modeling 53 (1) (2013) 1–10.

[37] B. Seijo-Pardo, V. Bolón-Canedo, A. Alonso-Betanzos, Using a feature selection ensemble on dna microarray datasets, in: Proceeding of 24th European Symposium on Artificial Neural Networks, 2016, pp. 277–282.

[38] V. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, 2000.

[39] C. J. Burges, A tutorial on support vector machines for pattern recognition, Data mining and knowledge discovery 2 (2) (1998) 121–167.

[40] B. Schölkopf, C. J. Burges, A. J. Smola, Advances in kernel methods: support vector learning, MIT press, 1999.

[41] A. d. Haro García, Scaling data mining algorithms. Application to instance and feature selection, Granada: Universidad de Granada, 2012.

[42] S. Garcia, F. Herrera, An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons, Journal of Machine Learning Research 9 (Dec) (2008) 2677–2694.

[43] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the american statistical association 32 (200) (1937) 675–701.

[44] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[45] O. J. Dunn, Multiple comparisons among means, Journal of the American Statistical Association 56 (293) (1961) 52–64.

[46] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics (1979) 65–70.

[47] Y. Hochberg, A sharper bonferroni procedure for multiple tests of significance, Biometrika 75 (4) (1988) 800–802.

[48] G. Hommel, A stagewise rejective multiple test procedure based on a modified bonferroni test, Biometrika 75 (2) (1988) 383–386.

[49] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.

[50] J. H. Gennari, P. Langley, D. Fisher, Models of incremental concept formation, Artificial intelligence 40 (1-3) (1989) 11–61.

[51] M. A. Hall, Correlation-based feature selection for machine learning, Ph.D. thesis, The University of Waikato (1999).

[52] L. A. Belanche, F. F. González, Review and evaluation of feature selection algorithms in synthetic problems, arXiv preprint arXiv:1101.2320.

[53] L. C. Molina, L. Belanche, À. Nebot, Feature selection algorithms: A survey and experimental evaluation, in: Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, IEEE, 2002, pp. 306–313.

[54] M. Lichman, UCI machine learning repository, `http://archive.ics.uci.edu/ml` (2013).

[55] J. Li, K. Cheng, S. Wang, F. Morstatter, R. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, arXiv preprint arXiv:1601.07996.

[56] B. Seijo-Pardo, V. Bolón-Canedo, I. Porto-Díaz, A. Alonso-Betanzos, Ensemble feature selection for rankings of features, in: Advances in Computational Intelligence, Springer, 2015, pp. 29–42.