Analytical and Experimental Study of Filter Feature Selection Algorithms for High-dimensional Datasets

Adrian Pino[1] Carlos Morell[2]

[1]Universidad de Holguín "Oscar Lucero Moya", Holguín, Cuba
[2] Universidad Central "Marta Abreu de las Villas", Villa Clara, Cuba
apinoa@facinf.uho.edu.cu, cmorellp@uclv.edu.cu

Abstract

In this paper a new taxonomy for feature selection algorithms created for high-dimensional datasets is proposed. Also, several selectors are described, analyzed and evaluated. It was observed that the *Cfs-SFS* algorithm reached the best solutions in most of the cases. Nevertheless, its application in very high-dimensional datasets is not recommended due to its computational cost. *Cfs-BARS*, *Cfs-IRU* and *MRMR* algorithms have similar results to those of *Cfs-SFS*, but in a relatively lesser time. The *INTERACT* algorithm gets good solutions too, but its computational cost is higher if compared to the above mentioned. On the other hand, the *QPFS* and *FSBMC* algorithms reached the worst solutions.

Keywords: feature selection, filter strategy, high-dimensional datasets, supervised classification

1. Introduction

Due to the great amount of information that is present in the current technological processes, many machine learning algorithms have been proposed for extracting the useful knowledge for taking decision in a better way. When the dataset analyzed is high-dimensional (thousands or dozen of thousands features), there are huge possibilities of finding inconsistent, redundant and irrelevant data. In these cases, generally, feature selection algorithms are used before employing the machine learning algorithms [1]. The feature selection algorithms remove the irrelevant and redundant features, in a way that a new dataset with the relevant information implicit is obtained [2]. In this sense, the process of knowledge extraction is carried out with better efficiency and the decision-making process is carried out with a higher level of prediction.

The scientific community has focused specially on the algorithms for high-dimensional dataset because nowadays they are typical in different processes, such as: business administration [3], human-genome projects [4], polymer identification in real time [5], cardiac arrhythmias classification [6], automatic control of prosthesis [7] and others.

According to the evaluation function employed, the feature selection algorithms can be cataloged as filters, wrappers or hybrids [8]. The filters use statistical functions for evaluating the subsets in an independent way of the machine learning algorithm [9], while the wrappers use the machine learning algorithm for determining the prediction power obtained in the subset evaluated. The wrappers are generally costly in terms of computational complexity because the learning and validation process are executed every time a subset is evaluated. Nevertheless, the results obtained by the wrappers are better [10]. On the other hand, the hybrid strategy combines the two mentioned above with the purpose of achieving a balance between the effectiveness and the efficiency of its execution [10].

In this paper, an analytical and experimental study of filter feature selection algorithms for high-dimensional datasets is presented. The algorithms evaluated are those created for supervised learning. Previous feature selection evaluation studies have been developed [11, 12], but the majority of them consider just a few of selectors and/or the datasets used for the comparison are of dissimilar dimensions. With the aim of giving a detailed analysis to the scientific community about the performance of the filter selectors made for high-dimensional datasets, in the next sections a brief survey is presented and to finish, an evaluation of many feature selection algorithms is carried out in different high-dimensional datasets.

2. Feature Selection for High-dimensional Datasets

The feature selection process has been considered as an ever evolving problem because of the growth of the data throughout time [13].

When the dataset contains thousands or dozens of thousands features, to obtain the best subset of all possible (2^n) is unpractical because a lot of time is required. For this reason, the algorithms created for this type of datasets search only heuristically in a reduced search space.

The taxonomy showed in Figure 1 is proposed with the aim of studying and analysing, in a better way, the feature selection algorithms that will be evaluated in this paper. Notice that the algorithms discussed in this paper, are also shown.

According to this taxonomy, the feature selection algorithms can be classified in three groups according to the amount of features evaluated by the evaluation function: univariate, pairwise and multivariate. Theoretically, in this diagram, for all horizontal levels, the

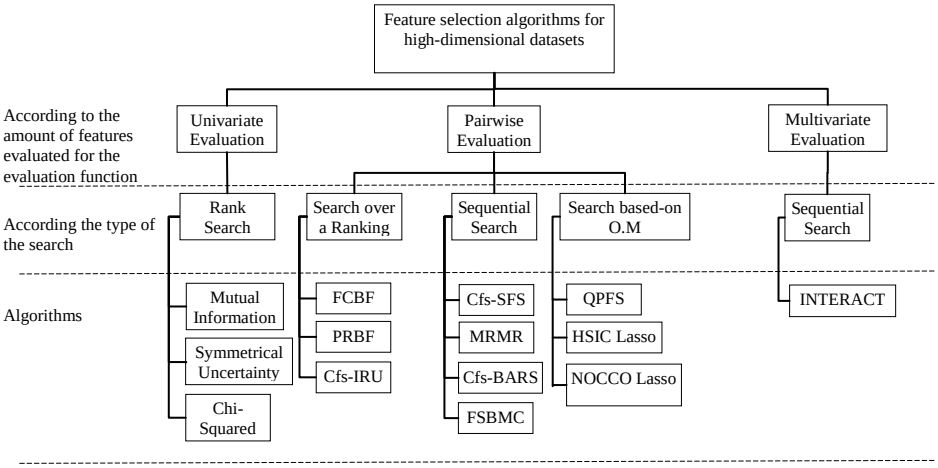

Fig. 1: Taxonomy used in this paper for classifying the feature selection algorithms for high-dimensional datasets.

classifications located to the left are faster the more to the left they are, but less effective. Whereas those located more to the right obtain better results, but they need more time. Hence, a balance for these parameters could be found in the algorithms located in the center of the diagram.

The algorithms located in the first group generate a feature ranking according to the relevance level of each one of them respect to the class feature. Later on, the first features are selected from the ranking.

Within these algorithms there are the *Mutual Information* [14] and *Symmetrical Uncertainty* [15] which are based on the information theory concepts.

The difference between the entropy of feature C and its entropy after observe the feature A_i, represents the *Mutual Information* $I(A_i,C)$ between both features (equation 3).

$$\begin{aligned} I(A_i;C) &= H(C) - H(C \mid A_i) \\ &= H(A_i) - H(A_i \mid C) \\ &= H(A_i) + H(C) - H(C, A_i) \end{aligned} \quad (3)$$

The *Mutual Information* function favors the features that contain many values. For removing this undesirable obstacle in advanced, a normalization method is applied, obtaining thence, the *Symmetrical Uncertainty* function. Notice that both functions are symmetric, nevertheless, only the *Symmetrical Uncertainty* gives values between zero and one.

$$SU(A_i;C) = 2 * \frac{I(A_i;C)}{H(A_i) + H(C)} \quad (4)$$

On the other hand, the *Chi-Squared* function is based on the probability theories and represents the goodness-of-fit test between two distributions.

$$\chi^2(A_i;C) = \sum_{\forall j \in A_i; \forall k \in C} \frac{(P(A_i^j, C_k) - P(A_i^j) * P(C))^2}{P(A_i^j) * P(C)} \quad (5)$$

The algorithms located in the first group do not take into account the redundancy level among the features which implies that the results obtained by the machine learning applied in a posterior stage are not going to be so good [16].

The algorithms of the second group use an evaluation function that only evaluates pairs. They compute the relevance score of a subset through the evaluation of its pairs: feature-feature and feature-class [17]. Due to this way of evaluation, these algorithms are, generally, very fast and also take into account the redundancy among features. Inside this group, there are three other groups according to the search employed: search over a ranking, sequential search and search based on *Optimization Models* (*OM*). The first type of search is based on the creation of a ranking considering the relevance score feature-class and then the most redundant features are removed from the ranking through the pairwise evaluation feature-feature. On the other hand, in the algorithms that use the sequential search, in each iteration, the feature that better improves the current solution is added.

The algorithms that are based on optimization models, determine the weight of each feature which optimizes an objective function. The objective function takes into account the relevance score of each feature (feature-class) and the redundancy score of each pair of features (feature-feature) [17]. Once the model has been solved, the r features with the biggest weight or those that exceed a predefined weight value are selected. Among the models, mostly used in this type of search are the lineal programming [18] and the quadratic programming [19].

Finally, the multivariate functions evaluate the subsets in an integral way, that is to say, they explore the whole data

from the subset for determining its quality. For this reason, they are not so efficient; however, they generally obtain good solutions.

3. Feature Selection Algorithms for High-Dimensional Datasets

In this section the algorithms presented in the low level of Figure 1 are briefly described and analyzed in a chronological order.

3.1. Correlation-based Feature Selection (Cfs)

Cfs function [20] evaluates each candidate subsets taking into account the mean feature-class relevance $\overline{R_{A,C}}$ and the mean feature-feature redundancy $\overline{R_{A,A}}$ of each possible pair of features in the subset. This function is showed in the equation 6.

$$F_{Cfs}(S_c) = \frac{|S_c| * \overline{R_{A,C}}}{\sqrt{|S_c| + |S_c|(|S_c|-1)\overline{R_{A,A}}}} \quad (6)$$

For computing the relevance feature-class and the redundancy feature-feature the *Symmetrical Uncertainty* function is used (see equation 4). *Cfs* is generally employed with the *SFS* strategy [21]-[23] because it is not so complex computationally. In this study, *Cfs* will be also used with *SFS* and it will be called from now on *Cfs-SFS*.
According to experimental studies, the solutions obtained by this algorithm are fairly good [21]-[24], nevertheless, its computational complexity is quite high when it is compared with the algorithms that use a search over a ranking.

3.2. Minimum Redundancy–Maximum Relevance (MRMR)

MRMR function [21] also uses the *SFS* search strategy. For determining if a feature is part of the solution a correlation function is used. This function takes into account the relevance score of the feature for the class and its redundancy score with each one of the already selected. The author of this algorithm proposes two evaluation functions named: *Mutual Information Difference* (equation 7) and *Mutual Information Quotient* (equation 8). It is valid to point out that according to experimental studies of the author with the *Mutual Information Quotient* the solutions obtained are generally less redundant.

$$F_{MRMR}(A_i;S) = I(A_i;C) - \frac{1}{|S|}\sum_{\forall A_j \in S} I^*(A_i;A_j) \quad (7)$$

$$F_{MRMR}(A_i;S) = I(A_i;C) / \frac{1}{|S|}\sum_{\forall A_j \in S} I^*(A_i;A_j) \quad (8)$$

The principal disadvantage of this algorithm is that the number of feature to select *t* must be specified and generally the optimal solution size is not known.
Another disadvantage of this algorithm is the use of the mean redundancy that the candidate feature has with the already selected features, in the evaluation function. Supposing, for example, that the candidate feature is totally correlated to one of the current solution and lightly correlated to the others, it is highly probable that the candidate feature will be added to the solution. Nevertheless the information that this feature contains has been already imported by other feature.
In *MRMR* algorithm, the *Mutual Information* of all possible feature pairs: feature-feature and feature-class, is computed. Therefore, the computational complexity is quadratic $O(n^2)$.
The author of this algorithm also proposes functions for continuous features. However better results are obtained in his experiments when after discretizing the datasets the discrete functions are applied.

3.3. Fast Correlator-Based Filter (FCBF)

In the *FCBF* algorithm [22] a feature ranking is generated according to the feature relevance with the class computed through the *Symmetrical Uncertainty* function.
Once defined the ranking, the first feature is considered as the predominant and its redundancy with the others is computed through the *Symmetrical Uncertainty* function. The features that are more correlated with the predominant feature than with the class will be removed from the ranking. The process is repeated iteratively taking as predominant feature the next feature which remains in the ranking. A threshold λ can be used for determining how much the correlation feature-class value should exceed to the correlation feature-predominant value for avoiding the feature elimination. The final solution will be composed by the features that were not removed in any of the iteration.
The computational complexity of the *FCBF* algorithm is *O(nlogn)* taking into account the operations made in the features of the dataset. In practice, this algorithm is very fast because in an iteration a feature is selected and many others are removed. In datasets with many redundant features, the amount of paired evaluation decrease drastically in each iteration.
According to several experimental studies it has been observed that the *FCBF* algorithm obtains very good results [22]-[25]. One of the principal disadvantages of *FCBF* algorithm is that if two features contribute information to the current predominant feature; but they are redundant inwardly, then the one of bigger relevance will be selected and the other one will be removed. In this case it is not taken into account which of the two features better contributes to the predominant feature.

3.4. Incremental Ranked Usefulness (IRU)

In the *IRU* algorithm [26] a ranking is generated according to the relevance feature-class. Afterwards, beginning with the empty solution, the ranking is explored form the beginning to the end, adding the features that improve the solution to be part of it. In Figure 2 a representative example of the performance of this algorithm is shown. Notice that feature A_i represents the *i-th* feature of the ranking.

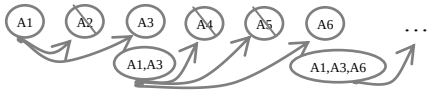

Fig. 2: Example of the *IRU* algorithm

When this search strategy is used in combination with the *Cfs* function, it could be seen as a hybrid between *FCBF* and *Cfs-SFS* algorithms. The similarities that it possesses with respect to the *FCBF* algorithm are: the search over a ranking and the early removing of features that are not good for the current solution. In fact, these two characteristics are the principal pillars of the agility and the efficiency of *FCBF* algorithm. On the other hand, the advantage that *Cfs-IRU* possesses respect to *FCBF* is that in the function *Cfs* the relevance and the redundancy are taken into account at the same time for determining if a feature will form part of the solution.
The computational complexity of this algorithm when it is used in combination with the *Cfs* function is similar to that of the *FCBF*. Therefore, it could be used in very high dimensional datasets.

3.5. Feature Selection Based on Mutual Correlation (FSBMC)

The *Feature Selection based on Mutual Correlation* (*FSBMC*) is proposed in [27]. As it is shown in figure 3, in the first step the *Mutual Correlation* between each feature pair is computed. For this, the *Mutual Correlation* formula, showed in equation 9, is used. Notice that the initial solution *S* will be the whole set of features.
Afterwards, the feature that is mostly correlated with the others is removed from *S* (lines 4-6). This is computed through the mean of the correlation that the feature to evaluate has with the others features in *S* (see equation 10). The algorithm stops when $|A|$-t features have been removed. The *t* value must be specified previously.

Algorithm: FSBMC
Input:
$D(A_1, A_2, ..., C)$ // dataset to be processed
t // size of the final solution
Output: S // selected subset
»Step1: Computing the correlation between pairs
1:For each feature A_i
2: Compute: $r_{A_i,A_j}; \forall A_j \in A; i \neq j$
»Step2: Removing the most redundant features
3:Initially: $S = A$
4:while ($|S| > t$)
5: Being: Aworst the more redundant feature of S:
$A_{worst} = max_{A_i:A_i \in S}\{F_{FSBMC}(A_i;C)\}$
6: $S = S \setminus A_{worst}$
7:return: S

Fig. 3: FSBMC algorithm

The equation used for computing the correlation between feature pairs is only applicable to continuous features. The author of this algorithm does not propose a measure for discrete features because he has only used genomics datasets in his experimental evaluations. Nevertheless, the *Mutual Information* or the *Symmetrical Uncertainty* functions could be used for discrete features.

$$r_{A_i;A_j} = \frac{\sum A_i^v * A_j^v - t * A_i^v * A_i^v}{\sqrt{(\sum (A_i^v)^2 - t * (A_i^v)^2) - (\sum (A_j^v)^2 - t * (A_j^v)^2)}} \quad (9)$$

$$F_{FSBMC}(A_i;S) = \frac{1}{|S|-1} \sum_{\forall A_j \in S; i \neq j} |r_{A_i;A_j}| \quad (10)$$

The principal difference of this algorithm respect to the others is that it does not use the correlation feature-class in his search strategy, being applicable to unsupervised classification problems [28]. Unfortunately, if a feature is lightly correlated with the class and it is not redundant with the others, then it will have high probabilities of being selected.
The computational complexity of *FSBMC* is *O(n(n+1)/2)* in the first step. On the other hand, for the second step $\sum_{k=t}^{n} |S|(|S|+1)$ simple mathematic operations are required, where $|S|$ represents the number of features remaining in the current iteration. For very high dimensional datasets this amount of operations is remarkable.

3.6. INTERACT

In the *INTERACT* algorithm [24] the principal goal is to get in the final solution all the features that interact inwardly. When a feature by itself is not highly correlated with the class, but combined with others features it reaches a high correlation for predicting the class values, then it is said that this feature interacts with others.
The evaluation function proposed by the authors of this algorithm for obtaining subsets composed by features that interact inwardly is the *Consistency Count* showed in the equation 11.

$$cc(S;A_i) = ICR(S \setminus A_i) - ICR(S) \quad (11)$$

$$ICR(S) = \frac{\sum_{1 \leq i \leq p} IC(E_i)}{m} \quad (12)$$

$$IC(E_i) = |E_i| - \max_{1 \le k \le t}\{E_i^k\} \quad (13)$$

For a feature subset *S*, the set of all instances with equal value in all its features is denoted E_i. From each set E_i, $|C|$ instance subsets can be obtained if their instances are separated by class.

The *Inconsistency Count* $IC(E_i)$ for an instances subset E_i is computed through the difference of the amount the instances in E_i and the amount of instances that the biggest instances subset E_i^k has (see equation 13). This measure takes value zero when all the instances in E_i belongs to the same class and takes value $|E_i|(|C|-1)/|C|$ when the class of all the instances in E_i are equally distributed.

On the other hand, the *Inconsistency Count Rate* is computed through the mean of all the *Inconsistency Count* of each instances subset E_i. Notice that *m* is the number of instances in the dataset.

As it can be observed the *Consistency Count* $cc(S, A_i)$ is basically a subtraction between the *Inconsistency Count* of a subset *S* without a feature A_i and the *Inconsistency Count* of the entire subset *S*. Hence, the *Consistency Count* represents the inconsistency that A_i contributes to the subset *S*. This can also be expressed as the consistency that *S* obtains after removing feature A_i from it.

The *INTERACT* algorithm is presented in figure 4.

Algorithm: INTERACT

Input:
$D(A_1, A_2, \ldots, C)$ // dataset to be processed
λ // inconsistency threshold
Output: S // subset selected

»Step1: Ranking generation

1: Create a descendent ranking *R* through $SU(A_i;C)$

»Step2: Removing inconsistent features

2: For each feature A_i in *R*, from the end to the beginning :
3: if $cc(S;Ai) > \lambda$
4: $S = S \setminus \{A_i\}$
5: return: *S*

Fig. 4: INTERACT algorithm

The authors of this algorithm use a hashing mechanism quite efficient for computing the *Consistency Count*. The whole process is made through two principal operations: removing keys and updating values in a hash table.

According to experimental studies of the author, this algorithm obtains good results considering the quality of the solutions and the running time. Despite this, the algorithms: *CCC* [29] and *SDC* [30] have been created for eliminating several intrinsic deficiencies of *INTERACT*. Nevertheless, improvements were made, oriented to obtain better solutions, but increasing the computational complexity. For this reason, they are not recommendable for high-dimensional datasets.

3.7. Best Agglomerative Ranked Subset (BARS)

In the *BARS* algorithm [25], a ranking is generated taking into account the relevance of each feature with the class. Later, many feature pairs are obtained through the union of the first feature in the ranking and each one of the followings, the second one and each one of the followings, and so on until the *q-th* feature and the followings in the ranking. In this way *q(n-1)/2* features pairs are obtained, where *q* is a predefined parameter. Then all these feature pairs are evaluated and a new ranking of pairs is made taking into account their relevance. The pairs that are not more relevant than the first/best feature in the previous ranking are removed.

In a second iteration, given the ranking, the process is repeated again, but now the subsets generated will have four features due to the union of feature pairs. It is valid remark that some subsets will have just three features because could be exists two feature pairs with common features. The stopping criterion is reached when in an iteration the new ranking obtained is empty, that is to say that none one of the subsets generate is more relevant than that the best obtained in the previous iteration. Finally, the returned solution will be the best subset of the previous iteration.

When the dataset analyzed is very consistent and does not have many redundant features, many subsets must be evaluated for reaching the stopping criterion. Consequently, the *BARS* algorithm is an efficient strategy when it is used in datasets with much irrelevant information.

3.8. Pearson Redundancy Based Filter (PRBF)

In the *PRBF* algorithm [23] a search strategy over a ranking is used. A very similar process to the *FCBF* algorithm is employed. The only difference is that after generating the ranking the *Chi-Squared* function (see equation 5) is used for the redundancy analysis. In this step, a feature A_j will be removed by a predominant feature A_i, if: $p\text{-}value(X^2(A_i,A_j)) > \alpha$. Where α is a predefined parameter. According to experimental results, the author proposes $\alpha=0,001$.

3.9. Quadratic Programing Feature Selection (QPFS)

In [31] the feature selection problem is modeled as a non-lineal quadratic problem [19]. Be *Q* a square matrix that represents the correlation between each feature pairs and be *F* a vector that stores the correlation of each feature with the class. It could be obtained a new vector *W* that represents the weight of each feature that optimizes a quadratic model. A basic quadratic model that contains the variables mentioned before is represented in the equation 14.

$$min_w\left\{\frac{1}{2}W^T QW - F^T W\right\}$$
$$with: W_i \geq 0 \; and \sum_{i=1}^{n} W_i = 1 \quad (14)$$

As it can be observed, for minimizing this objective function it is necessary to obtain the W_i values that minimize the redundancy pairs contained in Q and that maximize the relevance feature-class contained in F.

The computational complexity of the algorithms that solve this kind of model is very high. Hereby, a new algorithm called *QPFS*, based on a quadratic model, was proposed in [31] for high-dimensional datasets. The innovation is based on the application of the diagonalization method to the redundancy matrix Q.

$$min_w\left\{\frac{1}{2}W^T U\Lambda U^T - F^T W\right\} \quad (15)$$

In the equation 15 the new objective function is shown. The matrix Q has been transformed into $U\Lambda U^T$. U is an invertible matrix whose column vectors are the eigenvectors of Q and Λ is a diagonal matrix composed by the eigenvalues of Q.

The author proposes the use of the *Nyström* method [32] for obtaining an approximated diagonal matrix without evaluating all the feature pairs when the dataset is very high-dimensional. This reduces, in a great manner, the computational cost of *QPFS*. Nevertheless, it has negative repercussions over the obtained results [33].

3.10. HSIC Lasso and NOCCO Lasso

The optimization model *Least Absolute Shrinkage and Selection Operator* (*Lasso*) [34] has been used for feature selection [33, 35, 36].

In the *HSIC Lasso* algorithm [33] the *Independence Criterion of Hilbert-Schmidt* [37] has been used for determining the regression coefficient W_i of each feature A_i. In the solution of this model high weights are obtained for the features that have high correlation with the class and low correlation with the other features. The model is shown in the following equation.

$$\min\{\frac{1}{2}HSIC(C;C) - \sum_{i=1}^{n} W_i\, HSIC(A_i;C) + \frac{1}{2}\sum_{i,j=1}^{n} W_i W_j\, HSIC(A_i;A_j) + \lambda\|W\|_1\}; \quad (16)$$
$$with: W_1,..,W_n \geq 0$$

Here, the function $HSIC(A_i;A_j) = tr(\overline{K}^{(i)}\overline{L})$ evaluates the independence score between the feature A_i and the class feature C. $\overline{K}^{(i)}$ is a squared centered gram matrix which represents the dispersion score existing in the values of A_i. This matrix can be obtained through a Gaussian function.

Another algorithm based on the Lasso operator is *NOCCO Lasso* [33]. In this algorithm $\widetilde{K}^{(i)} = \overline{K}^{(i)}(\overline{K}^{(i)} + \varepsilon n I_n)^{-1}$ and $\widetilde{L} = \overline{L}(\overline{L} + \varepsilon n I_n)^{-1}$ are used instead of $\overline{K}^{(i)}$ and $\overline{L}$. The model regularization parameter is ε, and must be bigger than zero.

$$\min\{\frac{1}{2}\sum_{i,j=1}^{n} W_i W_j\, D^{NOCCO}(A_i;A_j) - \sum_{i=1}^{n} W_i\, D^{NOCCO}(A_i;C) + \lambda\|W\|_1\}; \quad (19)$$
$$with: W_1,..,W_n \geq 0$$

In the equation 19 the *NOCCO Lasso* model is shown. The *Normalized-Cross Covariance Operator* $D^{NOCCO}(A_i;A_j) = tr(A_i;A_j)$ represents a dependency measure based on kernel [38].

Both algorithms have low computational complexity in datasets with few instances, but when there are many instances ($m^2 > n$), its application is very expensive [33].

Finally, the nature of these operators does not allow their use in datasets with discrete features; therefore the insertion of a discrete operator in the Lasso model is even a non-studied problem.

4. Experimental Evaluation

In this section the results of the experimentation. Unfortunately, the evaluation of *HSIC Lasso* and *NOCCO Lasso* algorithms was not possible because they can only be used in continuous datasets. Furthermore, it was confirmed that *FCBF* and *PRBF* algorithms obtained very similar results in all the datasets; hereby the *PRBF* results are not shown.

On the other and, the algorithms *PRBF*, *MRMR* and *FSBMC* were implemented in this research using the Weka framework [39]. Moreover, the *QPFS* algorithm was taken from its author's implementation[1]. The other algorithms used in this evaluation are available in the Weka environment.

4.1. Methodology of Evaluation

With the aim of observing the practical behavior of each algorithm, the parameters considered in the evaluation were: the running time, the number of features removed and the accuracy reached by three machine learning algorithms in the reduced datasets.

[1] http://arantxa.ii.uam.es/~gaa/software_files/QPFS-1.0.zip

Table 1: Datasets used in this experimental evaluation

Datasets	Acronym	Features	Instances	Classes
ARRHYTHMIA	ARR	280	452	13
MADELON	MAD	501	2000	2
SECOM	SEC	591	1567	2
MULTIPLE FEAT.	MFE	650	2000	10
INTERNET ADV.	ADS	1559	3279	2
ARCENE	ARC	10001	100	2
ISOLET	ISO	618	6238	26
DEXTER	DEX	20001	300	2
GISETTE	GIS	5001	6000	2
P53-MUTANTS	MUT	5409	16772	2
PEMS-SF	PEM	138673	267	7
DOROTHEA	DOR	100001	800	2

The twelve datasets used in the experiment are shown in table 1. They were acquired from the UCI machine learning repository [40].

The machine learning algorithms used were *Naïve Bayes*, *C4.5* and *K-NN* with *K=3* [41]. They are very representatives and therefore, they are very used in this type of experimentation. For computing the accuracy of the machine learning algorithms in the reduced datasets, a 10-fold cross validation process with only one run was used as it was suggested in [42].

The *Friedman's* non-parametric test was used for detecting significant differences among all the results obtained by the selectors. The *Bergmann-Hommel's post-hoc* test was used for detecting significant differences between the results of each pair of algorithms as it was proposed in [43]. The non-parametric tests were used because it was checked that at least the results obtained by a feature selection algorithm in all the datasets do not have a normal distribution.

When there was required, numeric attributes in the training data were discretized using MDL-based discretization [44] with intervals learned from the training data. Notice that in evaluation functions such as: F_{MRMR} the *Mutual Information* coefficient was normalized.

Finally, the number of features to select by the *MRMR* and *FSBMC* algorithms was adjusted to *t= 100* because with this value good results are obtained [21, 27]. On the other hand, this parameter was adjusted to *t= 50* for *QPFS* algorithm for the same reason [31, 33].

4.2. Discussion of the Results

In Table 2 the running time of each algorithm in each dataset is shown.

Observing the results for this parameter, the algorithms could be divided into two groups: the fastest and the slowest. In the fastest group are: *FCBF*, *Cfs-BARS*, *Cfs-IRU* and *MRMR*, as long as with a notable running time differences are *FSBMC*, *QPFS*, *INTERACT* and *Cfs-SFS* algorithms.

As can be observed, the *FCBF* algorithm is the fastest in most of the cases. In fact, as it was pointed out, this takes place because in the redundancy analysis step, in an iteration only, a feature is selected and many others are removed. Similar strategies are used in *Cfs-BARS* and *Cfs-IRU* algorithms.

On the other hand, the results of *FSBMC* could not be available (NA) in the highest dimensional datasets because of its excessive time consuming (more than 72 hours). The only algorithm capable of obtaining the final solution in the datasets PEM and DOR before 72 hours was *FCBF*. In Table 3 the number of features selected by each algorithm in each dataset is shown. Again the *FCBF* algorithm points out positively, although it can be observed that all the algorithms achieve a splendid reduction in all the datasets.

With the aim of evaluating the quality of the solutions obtained by the algorithms in each dataset, three supervised classification algorithms were applied to the reduced subsets. In Table 6 the accuracy reached by the *Naïve Bayes*, *C4.5* and *K-NN* machine learning algorithms in the reduced datasets is presented. As can be observed for the *Naïve Bayes* classifier the algorithms with the best behavior are Cfs-SFS, *Cfs-BARS* and *Cfs-IRU*. On the other hand, the worst results were obtained by *FSBMC* and *QPFS* algorithms. It seems to be that the weak point of *FSBMC* is the fact of not taking into account the correlation feature-class. Otherwise, *QPFS* algorithm shows an unstable behavior.

Table 2: Running time in seconds of each algorithm in the different datasets

Datasets	Cfs-SFS	FCBF	MRMR	Cfs-IRU	FSBMC	INTERACT	Cfs-BARS	QPFS
ARR	0,857	**0,164**	0,423	0,355	1,079	1,200	0,201	1,900
MAD	1,587	**1,125**	4,117	1,623	17,854	3,354	1,449	16,370
SEC	2,615	**1,280**	3,437	2,195	18,383	8,643	1,64	17,680
MFE	35,615	20,626	**7,500**	14,111	30,523	24,099	10,099	32,890
ADS	23,053	44,872	29,452	**14,140**	456,024	82,231	17,709	65,180
ARC	2286,460	**1,533**	37,710	41,018	23488,931	131,499	17,496	37,240
ISO	239,135	**45,374**	71,881	189,501	NA	98,148	66,505	198,970
DEX	4474,246	**22,197**	206,897	96,014	NA	1288,706	83,12	1020,780
GIS	1350,364	**109,647**	450,545	639,760	NA	2551,882	147,368	5702,460
MUT	2972,056	**476,160**	1481,350	1048,921	NA	7642,748	1613,519	5997,090
PEM	NA	**111,064**	NA	NA	NA	NA	NA	NA
DOR	NA	**526,762**	NA	NA	NA	NA	NA	NA
Mean	1138,599	**72,298**	229,331	204,764	4002,132	1183,251	195,911	1309,056

Table 3: Number of features selected by each algorithm in each dataset

Datasets	Cfs-SFS	FCBF	MRMR	Cfs-IRU	FCBMC	INTERACT	Cfs-BARS	QPFS
ARR	26	**12**	50	24	50	23	16	100
MAD	7	**4**	50	7	50	15	6	100
SEC	17	**9**	50	12	50	23	11	100
MFE	150	136	**50**	146	**50**	137	137	100
ADS	72	75	50	71	50	**50**	72	100
ARC	53	39	50	56	50	**38**	39	100
ISO	184	**40**	50	244	NA	57	64	100
DEX	48	**35**	50	**35**	NA	40	36	100
GIS	77	**28**	50	73	NA	51	42	100
MUT	27	**10**	50	21	NA	20	14	100
PEM	NA	**130**	NA	NA	NA	NA	NA	NA
DOR	NA	**104**	NA	NA	NA	NA	NA	NA
Mean	66	**39**	50	69	50	45	44	100

For corroborating the observations previously exposed the mean ranking offered by the Friedman's test is shown on Table 4 for each machine learning algorithm. The *QPFS*, *FCBF* and *MRMR* algorithms always occupied the three last places.

Table 4: Mean ranking coefficient of the Friedman's test for each machine learning algorithm.

Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking
Cfs-SFS	2.40	Cfs-SFS	2.95	Cfs-SFS	2.75
Cfs-BARS	3.05	Cfs-BARS	3.00	Cfs-IRU	3.25
Cfs-IRU	3.05	Cfs-IRU	3.50	INTERACT	3.70
INTERACT	4.50	INTERACT	4.10	Cfs-BARS	3.80
FCBF	4.75	MRMR	4.15	MRMR	4.30
MRMR	4.90	FCBF	4.60	FCBF	4.95
QPFS	5.35	QPFS	5.70	QPFS	5.25
(a) Naïve Bayes		(b) C4.5		(c) KNN	

On the other hand, *Cfs-SFS* obtains the first place in the ranking made for each machine learning algorithm. The adjusted p-value obtained through the Friedman's test for the *Naïve Bayes*, *C4.5* and *K-NN* algorithms was of *0.01*, *0.06* y *0.112* respectively. With this datum it can be concluded that there are significant differences in the accuracy obtained by the *Naïve Bayes* and *C4.5* algorithms for $\alpha=0.1$.

A post-hoc test is executed for detecting significant differences among pair of algorithms. The Bergmann-Hommel'stest is a good choice because it takes into account the family-wise error [50].

As it can be observed in Table 5, there are significant differences between *Cfs-SFS* and *QPFS* algorithms according to the accuracy obtained by the *Naïve Bayes* classifier. When the *C4.5* classifier is used the *Cfs-SFS* and *Cfs-BARS* algorithms have significant differences with the *QPFS*.

Lastly, it can be concluded that for the datasets and classifiers used in this study, the *Cfs-SFS* algorithm is the one which provides the best solutions. Nevertheless, it is not recommended its use in the very high-dimensional datasets. Conversely, the *FCBF* algorithm is the fastest and the most reducing, and its efficiency is not significantly worse than the *Cfs-SFS*. A balanced behavior between the efficient of FCBF and the effectiveness of *Cfs-SFS* could be found in the performance of *Cfs-BARS*, *Cfs-IRU* and *MRMR* algorithms.

Table 5: Adjusted *p-values* of the post-hoc *Bergmann-Hommel's* test for each machine learning algorithm in the hypothesis of comparison. There are indicated in blond letter the *p-values* that allows rejecting its correspondent hypothesis with a confidence level of *0.1*.

Bergmann-Hommel (adjusted p-value)		
Hypothesis	Naïve Bayes	C4.5
Cfs-SFS vs QPFS	**0,047**	**0,093**
Cfs-SFS vs MRMR	0,145	2,356
Cfs-SFS vs FCBF	0,165	1,315
Cfs-BARS vs QPFS	0,259	**0,093**
Cfs-IRU vs QPFS	0,259	0,251
Cfs-SFS vs INTERACT	0,268	2,356
MRMR vs Cfs-BARS	0,555	2,356
MRMR vs Cfs-IRU	0,555	2,356
FCBF vs Cfs-BARS	0,555	1,315
FCBF vs Cfs-IRU	0,555	2,356
INTERACT vs Cfs-BARS	0,800	2,356
Cfs-IRU vs INTERACT	0,800	2,356
INTERACT vs QPFS	3,411	1,315
Cfs-SFS vs Cfs-IRU	3,411	3,415
Cfs-SFS vs Cfs-BARS	3,411	3,415
FCBF vs QPFS	3,411	2,356
MRMR vs QPFS	3,411	1,315
MRMR vs INTERACT	3,411	3,415
FCBF vs INTERACT	3,411	3,415
FCBF vs MRMR	3,411	3,415
Cfs-IRU vs Cfs-BARS	3,411	3,415

Through *INTERACT* algorithm good solutions are generally obtained, but it is very slow for very high dimensional datasets. Unexpectedly, the *QPFS* algorithm, with the configuration parameter used in this study, obtains poor results when is compared with the rest of the algorithms. Finally, in this study the *FSBMC* gets the worst results and its application in very high dimensional datasets was not possible for its high computational complexity.

Table 6: 10-fold cross validation accuracy reached by each machine learning algorithm in each reduced dataset

	Cfs-SFS	FCBF	MRMR	Cfs-IRU	FSBMC	INTERACT	Cfs-BARS	QPFS
Naïve Bayes								
ARR	69,912	65,929	**70,133**	68,584	44,912	68,363	68,805	63,938
MAD	**60,650**	57,700	60,300	60,650	52,950	59,150	60,650	**63,050**
SEC	83,089	**89,087**	55,712	82,451	14,869	60,115	88,003	25,527
MFE	**97,650**	96,750	97,200	97,350	79,700	96,800	97,300	**97,650**
ADS	96,218	96,127	**96,584**	96,340	87,771	95,669	96,310	92,132
ARC	93,000	93,000	**94,000**	**94,000**	63,000	**94,000**	**94,000**	62,000
ISO	**88,602**	81,420	84,306	88,169	NA	85,332	86,502	83,264
DEX	92,333	89,667	**93,333**	91,667	NA	90,667	91,667	88,333
GIS	**93,333**	89,383	88,867	92,750	NA	88,517	92,383	78,283
MUT	95,099	**96,476**	95,850	94,610	NA	96,315	94,616	95,451
PEM	NA	**83,521**	NA	NA	NA	NA	NA	NA
DOR	NA	**94,000**	NA	NA	NA	NA	NA	NA
Mean	86,989	85,554	83,628	86,657	57,200	83,493	**87,024**	75,163
C4.5								
ARR	68,363	**69,469**	68,584	68,363	41,814	65,266	69,248	67,257
MAD	73,400	61,600	72,700	73,400	51,400	**78,200**	72,800	68,950
SEC	92,470	92,597	**93,363**	92,597	92,853	92,597	92,534	91,512
MFE	**95,300**	94,050	92,500	94,000	51,250	93,900	94,450	94,500
ADS	96,615	96,584	**97,072**	96,493	87,832	96,889	96,950	92,223
ARC	83,000	78,000	81,000	**84,000**	58,000	**84,000**	**84,000**	58,000
ISO	**83,200**	77,942	73,726	82,703	NA	80,042	80,699	80,955
DEX	**85,333**	81,667	83,667	83,000	NA	85,000	85,333	80,000
GIS	**93,950**	91,100	92,517	93,917	NA	93,688	93,733	92,483
MUT	99,177	**99,249**	99,159	99,213	NA	99,136	99,219	99,034
PEM	NA	**88,390**	NA	NA	NA	NA	NA	NA
DOR	NA	**89,875**	NA	NA	NA	NA	NA	NA
Mean	**87,081**	84,226	85,429	86,769	63,858	86,872	86,897	82,792
KNN (K=3)								
ARR	**65,708**	65,487	63,053	**65,708**	52,876	61,504	62,168	62,168
MAD	76,300	57,700	65,300	76,300	52,050	**85,250**	73,400	58,150
SEC	92,597	91,704	91,385	91,832	91,895	**93,299**	91,832	92,725
MFE	97,200	97,250	94,000	97,450	80,550	97,400	97,550	96,150
ADS	96,462	96,432	**97,438**	96,401	87,832	96,310	96,554	92,040
ARC	**86,000**	78,000	81,000	84,000	71,000	82,000	**86,000**	65,000
ISO	**90,769**	82,959	80,186	89,516	NA	85,845	87,785	87,464
DEX	88,333	88,000	**90,333**	86,333	NA	86,000	86,000	84,333
GIS	95,750	91,333	93,267	95,517	NA	**96,150**	94,533	93,883
MUT	99,088	99,118	**99,165**	99,094	NA	99,118	98,909	99,130
PEM	NA	**74,532**	NA	NA	NA	NA	NA	NA
DOR	NA	**92,750**	NA	NA	NA	NA	NA	NA
Mean	**88,821**	84,798	85,513	88,215	72,701	88,288	87,473	83,304

5. Conclusions

Feature selection, as a preprocessing technique, constitutes a fundamental step for improving the results of the machine learning algorithms. In this paper, a new taxonomy is proposed for the feature selection algorithms created for high dimensional datasets. Furthermore, several algorithms of this type are described and analyzed. An experimental evaluation of those algorithms that have obtained good results on the reviewed literature was carried out in high dimensional datasets. For this evaluation it is observed that through *Cfs-SFS* the best solutions are generally obtained, but due to the elevated computational complexity of the *SFS* strategy, its application in high dimensional datasets are not so practical, mainly in real time problems. On the other hand, *Cfs-BARS* and *Cfs-IRU* algorithms obtain similar results to the *Cfs-SFS*, but they do it in a lesser time. *INTERACT* algorithm obtains good results too, but its running time is even larger that the used by *Cfs-SFS*. This is due to the backward search made on the altars of discovering interacting features.

With the aim of evaluating the *QPFS* algorithm, its parameter of subsample was adjusted to $p=0.005$ and the results obtained were not good if they are compared to the results of the algorithms mentioned previously. Moreover, in spite of decreasing its number of evaluations with $p=0.005$, the running time was considerably bigger.

Finally, the *FSBMC* algorithm obtains discouraging results and it was not possible its application in the majority of the datasets because of its high computational cost.

6. References

[1] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning. Artificial Intelligence", 97: pp. 245–271, 1997.

[2] R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin based feature selection - theory and algorithms", *ICML*, 2004.

[3] K. Ng and H. Liu, "Customer retention via data mining", *AI Review*, pp. 569-590, 2000.
[4] E. Xing, M. Jordan, and R. Karp, "Feature selection for high-dimensional genomic microarray data", *Proc. Of the Eighteenth International Conference on Machine Learning*, pp. 601-608, 2001.
[5] R. Leitner, H. Mairer, and A. Kercek, "Real-time classification of polymers with NIR spectral imaging and blob analysis", *Real-Time Imaging*, 9: pp. 245 – 251, 2003.
[6] J. Rodriguez, A. Goni, and A. Illarramendi, "Real-time classification of ECGs on a PDA". *IEEE Transactions on Information Technology in Biomedicine*, 9(1): pp. 23–34, 2005.
[7] P. Shenoy, K. Miller, B. Crawford, and R. Rao, "Online electro-myographic control of a robotic prosthesis, *IEEE Trans Biomed Eng*, 55(3): pp. 1128–1135, 2008.
[8] R. Kohavi and G. John, "Wrappers for feature subset selection", *Artificial Intelligence Journal, Special issue on relevance*, 97(1-2), pp. 273–324, 1997.
[9] H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery and Data Mining". *Boston: Kluwer Academic Publishers, ISBN 0-7923-8198-X*, 1998.
[10] S. Das, "Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection", *Proc. Of 18th Intl Conf. Machine Learning*, pp. 74-81, 2001.
[11] M. Hall and G. Holmes, "Benchmarking Attribute Selection Techniques for Discrete Class Data Mining", *IEEE Transactions On Knowledge and Data Engineering*, Vol. 15, No. 3, May/June, 2003.
[12] N. Sánchez, A. Alonso and M. Tombilla, "Filter Methods for Feature Selection – A Comparative Study", *Springer-Verlag Berlin Heidelberg 2007*, LNCS 4881, pp. 178–187, 2007.
[13] H. Liu, H. Motoda, R. Setiono and Z. Zhao, "Feature Selection: An Ever Evolving Frontier in Data Mining", *Workshop and Conference Proceedings. The Fourth Workshop on Feature Selection in Data Mining*, 10: pp. 4-13, 2010.
[14] H. Almuallim and T. Dietterich, "Learning boolean concepts in the presence of many irrelevant features". *Artificial Intelligence*, 69(1-2): pp. 279–305, 1994.
[15] R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, 1993.
[16] W. Press, B. Flannery, S. Teukolski and W. Vetterling, "Numerical Recipes", *C. Cambridge University Press*, 1988.
[17] A. Harol, C. Lai, E. Pezkalska and R. Duin, "Pairwise feature evaluation for constructing reduced representations", *Pattern Anal Applic* vol. 10 pp. 55–68, 2007.
[18] J. Neter and W. Wasserman. "Applied Linear Statistical Models", *R. Irwin, INC.*, 1974.
[19] D. Bertsekas, "Nonlinear Programming", *Athena Scientific*, 1999.
[20] M. Hall, "Correlation-based feature selection for discrete and numeric class machine learning", *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 359-366, 2000.
[21] C. Ding and H. Peng, "Minimum Redundancy Feature Selection from Microarray Gene Expression Data," *Proc. Second IEEE Computational Systems Bioinformatics Conf.*, pp. 523-528, 2003.
[22] L.Yu, H. Liu, "Efficient feature selection via analysis of relevance and redundancy". *J. Mach. Learn. Res.* 5, pp. 1205–1224, 2004.
[23] J. Biesiada and W. Duch, "Feature Selection for High-Dimensional Data: A Pearson Redundancy Based Filter," *in Advances in Soft Computing*, vol. 45, pp. 242–249, Springer, 2008.
[24] Z. Zhao and H. Liu. "Searching for interacting features", *In Proceedings of International Joint Conference on Artificial Intelligence*, pp. 1156-1161, 2007.
[25] R. Ruiz, J. Riquelme and J. Aguilar-Ruiz, "Best Agglomerative Ranked Subset for Feature Selection". *JMLR: Workshop and Conference Proceedings*, vol. 4, pp. 148-162, 2008.
[26] R. Ruiz, J. Riquelme and J. Aguilar-Ruiz, "Búsqueda secuencial de subconjuntos de atributos sobre un ranking". *Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005*, ISBN: 84-9732-449-8, pp.251-260, 2005.
[27] M. Haindl, P. Somol, D. Ververidis and C. Kotropoulos, "Feature Selection Based on Mutual Correlation", *Lecture Notes in Computer Science, Progress in Pattern Recognition, Image Analysis and Applications*, v:4225, ISSN 0302-9743, pp. 569-577, 2006.
[28] R. Duda, P. Hart, and D. Stork. "Pattern Classification", *John Wiley & Sons*, 2001.
[29] K. Shin, X. Xu, "Consistency-based feature selection", *In: 13th International Conferecne on Knowledge-Based and Intelligent Information & Engineering Systems*, 2009.
[30] K. Shin and X. Ming, "A Consistency-Constrained Feature Selection Algorithm with the Steepest Descent Method", *MDAI 2009, LNAI 5861*, pp. 338–350, 2009.
[31] I. Rodriguez-Lujan, R. Huerta, C. Elkan and C. Cruz, "Quadratic programming feature selection", *Journal of Machine Learning Research*, 11, pp. 1491–1516, 2010.
[32] C. Fowlkes, S. Belongie, and J. Malik, "Efficient spatiotemporal grouping using the Nystrom method". *Proc. Of IEEE Conf. Comput. Vision and Pattern Recognition*, pp. 231–238, 2001.
[33] Y. Makoto, S. Leonid and S. Masashi, "High-Dimensional Feature Selection by Feature-Wise Non-Linear Lasso", 2012.
[34] R. Tibshirani, "Regression shrinkage and selection via the lasso". *Journal of the Royal Statistical Society*, Series B, 58(1), pp. 267–288, 1996.

[35] V. Roth, "The generalized Lasso", *IEEE Transactions on Neural Networks*, 15(1), pp. 16–28, 2004.
[36] F. Li, Y. Yang and E. Xing, "From lasso regression to feature vector machine". *In Y. Weiss, B. Scholkopf, and J. Platt, editors, Advances in Neural Information Processing Systems 18 (NIPS2005)*, pp. 779–786. MIT Press, Cambridge, MA, 2006.
[37] A. Gretton, O. Bousquet, A. Smola, and B. Scholkopf, "Measuring statistical dependence with Hilbert-Schmidt norms", *In 16th International Conference on Algorithmic Learning Theory (ALT2005)*, pp. 63–78, 2005.
[38] K. Fukumizu, A. Gretton, X. Sun, and B. Scholkopf, "Kernel measures of conditional dependence", *Advances in Neural Information Processing Systems 21 (NIPS2008)*, pp. 489–496, 2009.
[39] I. Witten and E. Frank, "Data Mining Practical Machine Learning Tools and Techniques with JAVA Implementations". Morgan Kaufmann Publishers, 2 Edition, 2005.
[40] C. Blake and C. Merz, "UCI repository of machine learning databases", 1998.
[41] T. Mitchell, "Machine Learning", McGraw-Hill, 1997.
[42] J. Demšar, "Statistical comparisons of classifiers over multiple data sets". *Journal of Machine Learning Research*, 7: pp. 1–30, 2006.
[43] S. García, and F. Herrera, "An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons", *Journal of Machine Learning Research* pp. 2677-2694, 2008.
[44] U. Fayyad, and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning". *In Int. Joint Conf. on AI*, pp. 1022–1027. Morgan Kaufmann, 1993.