# VARIABLE SELECTION FOR LINEAR REGRESSION IN LARGE DATABASES: EXACT METHODS

Joaquín Pacheco, jpacheco@ubu.es

Silvia Casado, scasado@ubu.es

Department of Applied Economics

University of Burgos

**Abstract.** This paper analyzes the variable selection problem in the context of Linear Regression for large databases. The problem consists in selecting a small subset of independent variables that can perform the prediction task optimally. This problem has a wide range of applications. One important type of application is the design of composite indicators in various areas (sociology and economics, for example). Other important applications of variable selection in linear regression can be found in fields such as chemometrics, genetics, and climate prediction, among many others. For this problem, we propose a Branch & Bound method. This is an exact method and therefore guarantees optimal solutions. We also provide strategies that enable this method to be applied in very large databases (with hundreds of thousands of cases) in a moderate computation time. A series of computational experiments shows that our method performs well compared with well-known methods in the literature and with commercial software.

**Keywords:** variable selection, linear regression, Branch & Bound methods, heuristics

July 21, 2020

## 1. Introduction

### 1.1. Motivation

Research very often involves analyze datasets with one dependent variable and multiple independent variables ("response" variable and "predictor" variables), giving a dataset that is multivariate and multidimensional. Frequently these analyses have been based on traditional models such as Multiple Linear Regression. Other more recent methods are based on neural networks, support vector machines, nearest neighbor, etc. In the simplest case, multiple linear regression involves a regression of the dependent variable with respect to the set of predictor variables. Although this full model regression approach might seem logical, there are several key problems. One of the most important is the following: having multiple predictors in a model adds noise to the analysis, with the effect that non-significant results may be returned, even when the model contains significant predictors (Mundry and Nunn 2009). Moreover, it is commonly assumed that only a small proportion of the predictor variables are truly influential to the response (Wang and Feng, 2019).

Recent improvements to data-collection technologies have resulted in complex regression problems in which the number of candidate predictor variables explaining the response variable may be very large. However, not all of the variables are equally relevant to this task and many of them repeat the information that they contribute. So, in regression, when the predictor vector contains many variables, variable selection becomes necessary, to improve the precision of a model fit. The variable selection process attempts to identify the ''best'' subset of predictors and simultaneously remove those variables that are redundant. The major benefits of variable selection are as follows (Sayed et al. (2018)): (i) improving the predictive performance of a statistical model by preventing overfitting; (ii) identifying a model that captures the essence of a system; and (iii) providing a computationally efficient set of explanatory variables. The problem of variable selection in linear regression is also known as Sparse Linear Regression.

In addition to these advantages, variable selection in linear regression has interesting applications. One of its main applications is updating composite indicators. Suppose a composite indicator consists of a large set of variables. If the set of variables that forms the composite indicator is too large it could be advisable (both from the economic point of view and from the point of view of understanding) to reduce the number of variables that explain the indicator, while maximizing the approximation (correlation) to the indicator initially obtained. In other words, the objective is to select a smaller subset of variables that is able to explain most of the information from the initial composite indicator (that is, the one obtained with all the original variables).

Composite indicators are also used in several areas (economy, society, quality of life, nature, technology, etc.) as measures of the evolution of regions or countries in such areas. The importance of composite indicators is explained in Nardo et al. (2005a and 2005b) and Bandura (2008), among other studies. More recent references concerning the importance of composite indicators can be found in Blancas et al. (2010) and Parada et al. (2015).

Other interesting fields of application of this problem are, for example, musical audio denoising (Févotte et al., 2008 and Févotte et al., 2006 ), wireless communications (Mateos et al., 2010), spectral analysis of images (Iordache et al., 2014, Bioucas-Dias et al., 2012, and Bioucas-Dias and Plaza, 2010), chemometrics (Filzmoser et al., 2012 ), genetics (Vounou et al., 2010 and Li et al., 2015), climate prediction (Chatterjee et al., 2012) and computer network diagnosis, neuroimaging analysis, and compressed sensing (Rish and Grabarnik, 2014), among others.

As explained in the foregoing paragraphs, variable selection in linear regression is an interesting process that could provide important benefits and also has interesting applications. In related literature, various methods have been proposed for this task. In this study we propose an exact method which guarantees the optimal solution. In addition, this method can find this optimal solution in very large databases (with hundreds of thousands of cases and moderate numbers of variables) in a moderate computation time. There are some interesting exact methods in the literature; however, to the best of our knowledge there have so far been no previous references that propose exact methods in large databases.

## 1.2 Related Literature

Variable selection procedures are important in applied data analysis (since in many cases a large number of variables are measured) in order to detect all the variables that have no influence on the response to be predicted and to eliminate them from the prediction algorithm (Aneiros et al., 2015). Models that include all the covariates are difficult to interpret and irrelevant variables increase the variance (Gijbels and Vrinssen, 2015). In the literature several variable selection methods are proposed for multiple linear regression models. Conventional variable selection strategies involving sequential searches (forward selection, backward elimination, or stepwise selection) use a range of goodness-of-fit measures, such as adjusted R2, Akaike Information Criterion (AIC), Bayesian Information Criteria (BIC), and Mallows Cp. These methods have various shortcomings, as is explained in Fan and Li (2001): they will not always provide the best subset, they become increasingly ineffective in higher dimensions, and they show high sensitivity to small changes in the data. Despite their weaknesses, they are still the first choice in routine data analysis and are applied in large databases because of their simplicity (Luo and Ghosal, 2016). The Nonnegative Garrote (Breiman, 1995) uses a penalty on shrinkage factors of the regression coefficients. In Tibshirani (1996), the Least Absolute Shrinkage and Selection Operator (LASSO) method is proposed. This method is a version of ordinary least squares (OLS) that constrains the sum of the absolute regression coefficients. Finally, Least Angle Regression, LARS (Efron et al., 2004), is a refinement of the LASSO algorithm that is easy to implement. Specifically, LARS sequences the candidate predictors in order of importance.

In general, these previous methods tend to get trapped in locally optimal models and face design problems with complex patterns of multicollinearity, specifically in large datasets (Hans and Dobra, 2007). To avoid these disadvantages several metaheuristic techniques have been introduced for solving large problems, such as Simulated Annealing (Meiri and Zahavi, 2006) and Genetic algorithms (Kilinc et al., 2016; Sayed et al., 2019). Metaheuristic

techniques are appropriate for solving the problem of variable selection in regression because from a computational point of view it is an NP-Hard problem. Examples of exact variable selection methods in other prediction and/or classification models may be found in Brusco and Steinley (2011) and Brusco et al. (2009). In this study we propose an exact method which guarantees the optimal solution. It can find this optimal solution in very large databases (with hundreds of thousands of cases).

### 1.3. Contribution

This paper proposes a Branch & Bound method for the variable selection problem in regression models. The method has been designed to solve this problem in databases with a very large number of cases. In addition, various tools and strategies are proposed for improving this method. These tools and strategies consist in using the information from a previously executed heuristic method. Incorporating these modifications produces a variant of the original method that is more efficient and effective

Both the original method and the variant obtain the optimal solution in acceptable computation times in databases with a moderate number of variables. When databases with a larger number of variables are involved, the computation time can be excessive. The methods therefore have to be interrupted after a certain period of time and it is not guaranteed that the optimal solution will be obtained. Nevertheless, in these cases the solution these methods reach (especially the variant) is of high quality.

Moreover, it must be pointed out that in our methods the computational complexity does not depend on the number of cases, making it possible to work with databases with a very large number of cases. This represents an additional advantage over other known methods. Details are given in Appendix 2.

We have conducted a series of computational experiments, using several artificial databases (matrices) and databases from a well-known repository. The experiments include comparisons with other well-known variable selection methods from the literature, as well as with commercial software. The results include statistical tests and demonstrate the high performance of both the original method and its variant. Specifically, the variant clearly outperformed the other methods analyzed.

In summary the main contributions are the following: a) The development of exact methods capable of finding better solutions than traditional methods in feature selection for linear regression and that can be applied in large databases; b) The design of tools to improve the performance of these methods. These tools are based on the use of information provided by some fast heuristic methods. This strategy could be used in similar problems.

The remainder of this paper is organized as follows. Section 2 outlines the definition of the problem. In section 3, the basic Branch & Bound method is explained, and the various tools for accelerating the Branch & Bound method are analyzed in Section 4. Section 5 contains a description of the simple and fast heuristic method. The computational experiments are discussed in section 6. The last section presents the final conclusions of the study and related future lines of research.

**2. Definition and formulation**

Consider a data matrix, $X$, corresponding to *m* cases and characterized by $n$ variables. We shall label the set of variables $V = \{1, 2, \dots, n\}$ (the variables are identified by their indices for the sake of simplicity).

Let $x_{ij}$ be the value of variable $j$ in the case $i$, $i = 1, \dots, m$; $j = 1, \dots, n$; Let $\boldsymbol{x_j}$ be the column vector with the values of variable $j$; in other words

$$\boldsymbol{x_j} = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{mj} \end{pmatrix} \quad j = 1, \dots, n.$$

It is known that $= \left(x_{ij}\right)_{i=1,\dots,m;j=1,\dots,n} = (\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_n})$. Let $y_i$ be the value of a variable $Y$ in the case $i$, $i = 1, \dots, m$.

For any subset of variables $S \subset V$, let us define:

$f(S)$ = R-squared ($R^2$) value of the linear regression model with $Y$ as the dependent variable and $S$ as the set of independent variables.

Let $p \in N$, verifying that $1 \leq p \leq n$, so that the problem may be defined as:

$$\text{Maximize } f(S) \tag{1}$$

subject to:

$$|S| = p \tag{2}$$

$$S \subset V \tag{3}$$

The optimal solution and the value corresponding to the problem defined by (1)–(3) are respectively denoted by $S_p^*$ and $g(p)$, that is, $g(p) = f(S_p^*)$.

**3. Description of the basic Branch & Bound method**

Let $p_0$ ($p_0 \leq n$) a fixed value, the initially proposed exact method finds the optimal solution $S_p^*$ and $g(p)$, $\forall\, p, 1 \leq p \leq p_0$. This method (denoted by *BnB*) is an algorithm based on a Branch & Bound strategy. Therefore, it involves the recursive exploration in the set of solutions. This set is represented by a search tree. Each node of the tree corresponds to a specific set of solutions. In the exploration of each node, it is checked if any of the solutions of the corresponding set could improve the best solution found for some $p$ value. If not, the exploration of that node ends. Otherwise, the exploration of that node continues. In this case, the set of solutions of this node is divided into two subsets corresponding to two new nodes, and these new nodes are explored.

More specifically, each node $C$ is defined by two subsets of variables $A$ and $B$, $A, B \subset V$, so that the solutions of node $C$ are all the solutions $S$ that contain all the variables of $A$ ("fixed variables"), and do not contain any variable of $B$ ("forbidden variables "). Obviously, $A \cap B =$

$\emptyset$. To divide the set of solutions of $C$, an element $a \in V - A - B$ is selected. The first of the subsets of solutions ("left branch") adds $a$ as a fixed variable, and the second of the subsets ("right branch") adds $a$ as a forbidden variable. Figure 1 illustrates this division process ("Branch Process"). This strategy is very similar to those used for other variable selection problems (Brusco and Steinley, 2011; Pacheco et al., 2013).

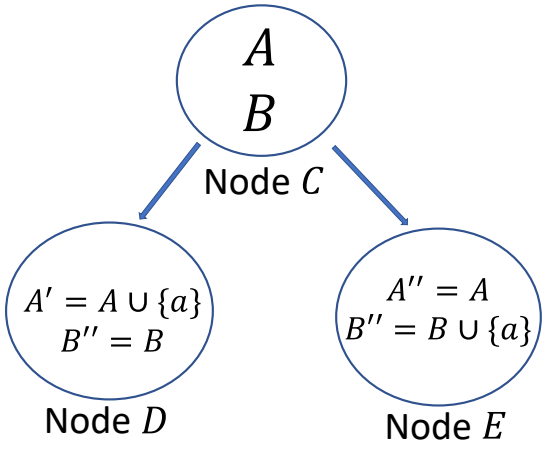


**Figure 1.** Branch process

The procedure for exploring the node corresponding to two subsets $A$ and $B$ such us $A, B \subset V$, and $A \cap B = \emptyset$ is denoted by *ExplorationNode* and is described in Pseudocode 1. The *BnB* method is described in Pseudocode 2. Note that *BnB* starts with the exploration of the initial node that corresponds to the whole set of solutions, ($A = B = \emptyset$, i.e., there are no fixed or forbidden variable).

---

Procedure *ExplorationNode*($A$, $B$)

If $f(A) > g(|A|)$ then make $S^*_{|A|} = A$ and $g(|A|) = f(A)$

If $f(V-B) > g(|V-B|)$ then make $S^*_{|V-B|} = V-B$ and $g(|V-B|) = f(V-B)$

If $(|A| = p_0)$ or $(A \cup B = V)$ then Exit (end Exploration of the node)

If $g(|A|) \geq f(V-B)$ then Exit (end Exploration of the node) (4)

Determine $a = argmax\ \{ f(A \cup \{v\}) / v \in V-A-B \}$ (5)

Make $A' = A \cup \{a\}$ and $B' = B$

Execute *ExplorationNode* ($A'$, $B'$)

Make $A'' = A$ and $B'' = B \cup \{a\}$

Execute *ExplorationNode* ($A''$, $B''$)

---

**Pseudocode 1. *ExplorationNode*** Procedure

---

Method *BnB*

Make $g(p) = 0, \forall\, p \leq p_0$ (6)

Make $A = \emptyset, B = \emptyset$

Execute *ExplorationNode*($A$, $B$)

---

**Pseudocode 2.** *BnB* Method

Some points are explained below:

- It has to be said that although $S_p^*$ and $g(p)$ are respectively defined as the optimum solution to problem (1) – (3) (section 2) and its corresponding value, in the description of the algorithm they are the corresponding approaches found during the search. Obviously, at the end of the execution of the *BnB* method $S_p^*$ and $g(p)$ correspond with this optimum and its objective function value.

- Note that if the condition of line (4) is verified, that is, $g(|A|) \geq f(V-B)$, then it can be concluded that no solution of that node can improve the best solution found for any $p$ value. Indeed, every solution $S$ in that node verifies $A \subset S \subset V-B$, therefore $f(S) \leq f(V-B) \leq g(|A|)$. On the other hand, by Appendix 1, $g(|A|) \leq g(|S|)$ then $f(S) \leq g(|S|)$; so, $S$ cannot improve $g(|S|)$.

- Note that line (5) establishes the criterion for choosing $a$, in this case the element that most improves the objective function $f$.

- As we can see in pseudocodes 1 and 2, the algorithm solves problem (1) – (3) for all the values of $p$ such that $p \leq p_0$. It must be pointed out that all the values $g(p)$, $p \leq p_0$ are important for the algorithm to function properly. In fact, high values of $g(p)$ enable the restriction in line (4) of the *ExplorationNode* procedure to be met, and unnecessary explorations are therefore avoided. If only the value of $g(p_0)$ is updated, but the values $g(p)$ for $p < p_0$ are not updated, then $g(p) = 0$ will remain true, for $p < p_0$. So, this restriction may never be satisfied and therefore the exploration of the corresponding node will have to continue, even though it contains no reliable solutions. Therefore, to avoid high computation time, it is important that the algorithm updates all the values of $g(p)$, for $p \leq p_0$, even though we are ultimately interested only in determining $S_{p_0}^*$ and $g(p_0)$.

In this sense, a strategy to reduce the computation time (to be explained in more detail in section 4) is not to start the algorithm with $g(p) = 0$ in line (6), but with good approximations to $g(p)$ and $S_p^*$. Specifically, values obtained by a rapid heuristic method will be proposed as initial values of $g(p)$ and $S_p^*$. In this way, fulfilment of the restriction in line (4) is favored from the start and unnecessary explorations are therefore avoided. In section 6, the effect of using this strategy is analyzed through the computational experiments with the variant of the Branch & Bound method described in section 4

**4. Description of various tools and a new variant**

In order to reduce the computation time of the basic Branch & Bound method, we propose some modifications. These consist in adding certain tools (use of information from heuristics) and

result in a variant of the original method. The following describes both the modifications and the corresponding variant:

- As explained above, the expression of line (4) can help to identify and avoid unnecessary explorations. However, the values of $g(p)$ are initially set a 0, as indicated in the expression of line (6). This value means that there is no compliance with the condition of line (4) in the first iterations, and the corresponding explorations are therefore not interrupted. Subsequently, compliance with this condition is increasingly forthcoming as the values of $g(p)$ are updated and increased.

Therefore, one idea that may help to increase the proportion of times that compliance with the condition of line (4) is forthcoming, and thereby to reduce the explorations, is to find initial values of $g(p)$ that are as high as possible as quickly as possible. In this respect, various heuristic algorithms have demonstrated that they can find good solutions to variable selection problems. Among the most recent references, the works of Pacheco et al. (2009), Brusco et al. (2009), and Brusco (2014) may be mentioned. In addition, these heuristic methods required a much shorter computing time than the exact methods. Heuristic strategies can therefore be good options for obtaining good (high) initial values of $g(p)$ (and the corresponding approximations to $S_p^*$).

- Moreover, the execution of a heuristic method can provide further useful information to be used efficiently in executing the Branch & Bound method. Specifically, this information may be used to select element *a* in line (5) for ramification. In particular, the object is to determine $\forall a \in V$

$$maxv(a) = max\{f(S): a \in S, |S| = p_0,\ \ S \text{ solution visited in the execution of the heuristic}\}$$

These values were found during the execution of the heuristic. Subsequently, in the execution of the Branch & Bound method, the element $a \in V - A - B$ with the highest $maxv(a)$ in each exploration is chosen in line (5). In doing so, it is not necessary to calculate the $f$ function to determine this element, at the same time as a logical rather than an arbitrary criterion is employed. In fact, the elements that belong to the solution $S_{p_0}^*$ obtained by the heuristic will be selected in the first explorations.

We propose a variant of the original *BnB* method that consists in adding the following two modifications:

a) In line (6) of the *BnB* method, replace:

Make $g(p) = 0$, for $p \leq p_0$

with

Read the values of $g(p)$ and the corresponding $S_p^*$ values obtained by the heuristic method

b) replace line (5) of the *ExplorationNode* procedure with the following line (5a)

Determine $a = argmax\ \{\ maxv(v)\ /\ v \in V - A - B\ \}$ (5a).

The effect of these modifications (using information contributed by a heuristic method) will be examined in section 6.

**5. A simple and fast heuristic algorithm**

As explained in section 4, a heuristic method should be run before executing *VariantBnB*. We have designed a fast heuristic method to solve the problem defined by (1) – (3), for different values of $p$ ($p \leq p_0$). The Heuristic algorithm that we propose in this section obtains the approximations to the values of $g(p)$ (and those corresponding to $S_p^*$) gradually; that is, beginning with $p = 1$ and ending with $p = p_0$. Moreover, the solution obtained for $p - 1$ is used as prior information to find the initial solution for $p$. The set of solutions for the different values of $p$ are stored in the vector $\boldsymbol{S}$, $\boldsymbol{S} = (S_1^*, S_2^*, \ldots, S_{p_0}^*)$ and the corresponding values of $g$ are stored in $\boldsymbol{G}$, $\boldsymbol{G} = (g(1), g(2), \ldots, g(p_0))$. The Heuristic algorithm is described in pseudocode 3.

---

***Heuristic*** **Algorithm** (input: $p_0$; var: $\boldsymbol{S}$, $\boldsymbol{G}$)

1. Determine $i^* = argmax\{ f(\{i\}) : i \in V \}$
2. Do $S_1^* = \{i^*\}$, $g(1) = f(S_1^*)$
3. **For** $p$ = 2 to $p_0$ **do**

    **Begin**

    4.Do $S_{ant} = S_{p-1}^*$

    5.Determine $i^* = argmax\{ f(S_{ant} \cup \{i\}) : i \in V - S_{ant}, S_{ant} \cup \{i\}\ is\ a\ feasible\ set \}$

    6.Make $S = S_{ant} \cup \{i^*\}$

    7.Execute ***LocalSearch***($p$, $S$)

    8.Do $S_p^* = S$ and $g(p) = f(S_p^*)$

    **end**

---

**Pseudocode 3.** ***Heuristic*** Algorithm

As we can see, the heuristic algorithm obtains the initial solution for $p$ = 1, which is trivial. Subsequently, it uses the solution obtained for $p$ – 1 ($S_{ant}$) in each iteration to complete a rapid initial solution $S$ for $p$. This initial solution is improved by a local search procedure (***LocalSearch***) and the approximation to $S_p^*$ and $g(p)$ is thereby obtained.

The ***LocalSearch*** procedure is an iterative method. It works as follows: in each iteration the set of the "neighborhood solutions" of the current solution $S$ is explored; if the current solution $S$ is improved by its best neighborhood solution, $S$', then the current solution moves to $S$'. The process ends if none of the neighborhood solutions improves the current solution. The set of the "neighborhood solutions" of the current solution $S$ is denoted as $N(S)$. The ***LocalSearch*** procedure is described in Pseudocode 4.

**Procedure *LocalSearch*** (input: $p_0$, var $S$)

**Repeat**

1. $f_{old} = f(S)$
2. Determine $S' = argmax\{ f(S'') : S'' \in N(S) \}$
3. If $f(S') > f(S)$ then do $S = S'$

**until** $f(S') \leq f_{old}$

**Pseudocode 4. *LocalSearch*** Procedure

$N(S)$ is the set of *feasible* solutions that can be reached from $S$ by neighborhood moves (neighboring moves are thus identified with the solutions they generate). In this case, each move is defined by exchanging an element of $S$ for an element outside it.

## 6. Computational experiments

To analyze the performance of the basic Branch & Bound method and its variant, we performed a set of computational experiments. For this purpose we designed set of X and Y matrices . The process of designing these matrices is described in subsection 6.1. In addition, a set of databases from the well-known UCI (University of California, Irvine) repository is presented in subsection 6.2.

Two sets of computational experiments were performed. In sub-sections 6.3 and 6.4 we describe these experiments and their corresponding results. The first set analyzed the efficiency of the tools proposed in section 4 for reducing the computation time of the Branch & Bound method. The second set compared the performance of our Branch & Bound method (specifically the variant proposed in section 4) with some well-known methods for feature selection in linear regression.

It should be noted that all the algorithms, methods, and procedures described in this study were implemented in Object Pascal using the Delphi compiler and the Rad Studio (10.3 – Rio) development environment. All the experiments were performed on an i7 7700 CPU 4.20 GHz PC using the same compiler.

### 6.1 Design of data matrices

A set of data matrices were generated for the various computational tests. These matrices are composed of the $X$ matrix of the independent variables, and (column) $Y$ for the dependent variable. The process of generating these matrices (similar to those used in Brusco et al. 2009 and Pacheco et al. 2013) consists of designing population correlation matrices $L$ with size $n$; a set of $m$ vectors following the normal distribution with the $L$ correlation matrix is generated from each population correlation matrix $L$; these *m* vectors make up the $X$ matrix (each vector is a row), and finally the $Y$ column is obtained from $X$. In Pacheco et al. (2013) it is explained how to generate vectors following the distribution $N(\mathbf{0}, L)$.

The population correlation matrices $L$ follow a simple pattern: the correlations between the different variables can have two values: a high value (0.7) and a low value (0.2). Specifically, variables 1, 2, 3 have high correlation values with each other and a low correlation value with the rest, variables 4, 5, 6 have high correlation values with each other and a low correlation value with the rest, and so on. As an example, a correlation population matrix $L$ is shown below with $n = 12$.

$$\begin{pmatrix} 1 & 0.7 & 07 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.7 & 1 & 0.7 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.7 & 0.7 & 1 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 1 & 0.7 & 0.7 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.7 & 1 & 0.7 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.7 & 0.7 & 1 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 1 & 0.7 & 0.7 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.7 & 1 & 0.7 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.7 & 0.7 & 1 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 1 & 0.7 & 0.7 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.7 & 1 & 0.7 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.7 & 0.7 & 1 \end{pmatrix}$$

As explained above, for each correlation population matrix $L$ a set of $m$ vectors (cases) is generated following the distribution $N(\mathbf{0}, L)$. A value of $m$ = 100000 was used. These $m$ vectors (cases) make up the matrix $X$.

Finally, the values of $y_i$ , $i = 1, \dots, m$, are obtained in the following way:

$$y_i = \beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2} + \cdots + \beta_n \cdot x_{in} + 0.5 \cdot \varepsilon$$

where $\varepsilon$ is a vector generated from the normal distribution $N(0,1)$. The values of $\beta_i$ are assigned as follows:

$$\beta_1 = 0, \beta_2 = 0.3, \beta_3 = 1, \beta_4 = 0, \beta_5 = 0.3, \beta_6 = 1, \beta_7 = 0, \beta_8 = 0.3, \beta_9 = 1, \dots$$

and so on.

Following this process, 19 matrices $(X \mid Y)$, with different values of $n$, were generated (one for each value). For each matrix a range of values of $p$ was considered. The lowest value of $p$ was defined as $\lfloor n \cdot 0.1 \rfloor + 1$, and the highest as $\lfloor n \cdot 0.2 \rfloor + 1$. The problem was to be solved for each of the values of $p$ in this range, in each matrix. These values are shown in Table 1.

| **Matrix #** | $n$ | $p$ range | **Matrix #** | $n$ | $p$ range |
|---|---|---|---|---|---|
| **1** | 18 | 2 – 4 | **11** | 48 | 5 – 10 |
| **2** | 21 | 3 – 5 | **12** | 51 | 6 – 11 |
| **3** | 24 | 3 – 5 | **13** | 54 | 6 – 11 |
| **4** | 27 | 3 – 6 | **14** | 57 | 6 – 12 |
| **5** | 30 | 4 – 7 | **15** | 60 | 7 – 13 |
| **6** | 33 | 4 – 7 | **16** | 63 | 7 – 13 |

| Matrix # | $n$ | $p$ range | Matrix # | $n$ | $p$ range |
|---|---|---|---|---|---|
| **7** | 36 | 4 – 8 | **17** | 66 | 7 – 14 |
| **8** | 39 | 4 – 8 | **18** | 69 | 7 – 14 |
| **9** | 42 | 5 – 9 | **19** | 72 | 8 – 15 |
| **10** | 45 | 5 – 10 | | | |

**Table 1.** Matrices and values of $p$ considered

The parameters related to the design of the population matrix, $X$, follow the same structure and values as in Pacheco et al. (2013). The parameters of the linear model used to obtain the vector $y$ follow similar patterns as in other recent works (for example Gijbels and Vrinssen, 2015).

### 6.2. Databases from the literature

As well as the matrices described in subsection 6.1, 10 databases from the well-known UCI repository (Dua and Graff, 2019) at https://archive.ics.uci.edu/ml/index.php were used for the computational experiments. Table 2 shows these databases and their features: number of cases ($m$) and number of variables ($n$). It also indicates the range of values of $p$ to be used (the same criterion was followed as in subsection 6.1)

| # | ***Database*** | | $\boldsymbol{m}$ | $\boldsymbol{n}$ | $\boldsymbol{p}$ **range** |
|---|---|---|---|---|---|
| 1 | *YearPredictionMsdn* | | 515345 | 90 | 10 – 19 |
| 2 | *Buzz in social media* | *TomsHardware* | 28179 | 96 | 10 – 20 |
| 3 | | *Twitter* | 583250 | 77 | 8 – 16 |
| 4 | *Parkinson Speech Dataset with Multiple Types of Sound Recordings* | | 1040 | 27 | 3 – 6 |
| 5 | *Geographical Original of Music* | | 1059 | 68 | 7 – 14 |
| 6 | *Facebook Comment Volume Dataset* | *Feature_Variant_1* | 40949 | 53 | 6 – 11 |
| 7 | | *Feature_Variant_2* | 81312 | | |
| 8 | | *Feature_Variant_3* | 121098 | | |
| 9 | *Superconductivity Data* | | 21263 | 81 | 9 – 17 |
| 10 | *Parkinson Telemonitoring* | | 5875 | 20 | 3 – 5 |

**Table 2.** Databases used

The data used were the same as those that appear in the UCI repository; in other words, they were not transformed or manipulated in any way. Only in the *Geographical Original of Music* database were some changes made. This database uses two dependent variables, latitude and longitude (geodesic coordinates), indicating the location of each case. We found the middle value of these variables, that is, the location of the "midpoint" of all cases. We then calculated the Euclidean distance from each case to this midpoint. This distance to the midpoint was taken as a new dependent variable and the original dependent variables (latitude and longitude) were ignored. Nevertheless, the data used are available to interested readers.

### 6.3 Computational results: comparison of the Branch & Bound methods

This subsection presents computational tests comparing the method described in section 3 (which we shall denote as *BnB Original*) and the variant described in section 4 (which we shall call *BnB Variant*). For both methods a maximum computation time of 1800 seconds was set for each database. The object was to avoid excessive computation times. Both methods are exact, and if they end without reaching this maximum time they guarantee that the optimal solution is obtained. The design of both methods ensures that they find the solution for all values of $p \leq$

$p_0$. Therefore, when they are executed taking $p_0$ as the maximum value in the range considered for each matrix (database), they find the solution for all the values of $p$ in this range. Table 3 shows the results for the matrices generated, described in subsection 6.1 (hereinafter "fictitious matrices"). For each of these matrices, the computation time in seconds (*time*) of both methods and the value of the objective function ($\boldsymbol{f}$) for each value of $p$ in the range considered are shown. In the case of the variant, the computation time includes the time used by the heuristic method executed beforehand. For the matrices in which the maximum computation time was reached, the results achieved up to that point are shown. In these cases it is not guaranteed that the solutions are optimal and the best value is indicated in bold type.

| $n$ | $p$ | BnB Original | | BnB Variant | |
|---|---|---|---|---|---|
| | | *time* | $f$ | *time* | $f$ |
| 18 | 2 | *0.211* | 0.54267 | *0.114* | 0.54267 |
| | 3 | | 0.69606 | | 0.69606 |
| | 4 | | 0.81110 | | 0.81110 |
| 21 | 3 | *1.256* | 0.65949 | *0.766* | 0.65949 |
| | 4 | | 0.76851 | | 0.76851 |
| | 5 | | 0.85269 | | 0.85269 |
| 24 | 3 | *3.307* | 0.63320 | *1.841* | 0.63320 |
| | 4 | | 0.73722 | | 0.73722 |
| | 5 | | 0.81792 | | 0.81792 |
| 27 | 3 | *19.851* | 0.61118 | *12.410* | 0.61118 |
| | 4 | | 0.71220 | | 0.71220 |
| | 5 | | 0.79005 | | 0.79005 |
| | 6 | | 0.85219 | | 0.85219 |
| 30 | 4 | *107.749* | 0.69208 | *77.329* | 0.69208 |
| | 5 | | 0.76798 | | 0.76798 |
| | 6 | | 0.82844 | | 0.82844 |
| | 7 | | 0.87801 | | 0.87801 |
| 33 | 4 | *294.822* | 0.67477 | *191.636* | 0.67477 |
| | 5 | | 0.74914 | | 0.74914 |
| | 6 | | 0.80806 | | 0.80806 |
| | 7 | | 0.85630 | | 0.85630 |
| 36 | 4 | *1617.791* | 0.66261 | *1220.798* | 0.66261 |
| | 5 | | 0.73451 | | 0.73451 |
| | 6 | | 0.79233 | | 0.79233 |
| | 7 | | 0.83947 | | 0.83947 |
| | 8 | | 0.87850 | | 0.87850 |
| 39 | 4 | *1800** | **0.64955** | *1800** | **0.64955** |
| | 5 | | **0.72105** | | **0.72105** |
| | 6 | | **0.77763** | | **0.77763** |
| | 7 | | **0.82393** | | **0.82393** |
| | 8 | | **0.86273** | | **0.86273** |
| 42 | 5 | *1800** | **0.70964** | *1800** | **0.70964** |
| | 6 | | **0.76520** | | **0.76520** |
| | 7 | | **0.81105** | | **0.81105** |
| | 8 | | **0.84907** | | **0.84907** |
| | 9 | | **0.88116** | | **0.88116** |
| 45 | 5 | *1800** | **0.70046** | *1800** | **0.70046** |
| | 6 | | **0.75609** | | **0.75609** |
| | 7 | | **0.80077** | | **0.80077** |
| | 8 | | **0.83806** | | **0.83806** |
| | 9 | | **0.86954** | | **0.86954** |

| $n$ | $p$ | BnB Original | | BnB Variant | |
|---|---|---|---|---|---|
| | | *time* | $f$ | *time* | $f$ |
| | 10 | | **0.89654** | | **0.89654** |
| 48 | 5 | *1800** | **0.69201** | *1800** | **0.69201** |
| | 6 | | **0.74630** | | **0.74630** |
| | 7 | | **0.79068** | | **0.79068** |
| | 8 | | **0.82773** | | **0.82773** |
| | 9 | | **0.85916** | | **0.85916** |
| | 10 | | **0.88585** | | **0.88585** |
| 51 | 6 | *1800** | **0.73570** | *1800** | **0.73570** |
| | 7 | | **0.77994** | | **0.77994** |
| | 8 | | **0.81687** | | **0.81687** |
| | 9 | | **0.84808** | | **0.84808** |
| | 10 | | **0.87477** | | **0.87477** |
| | 11 | | 0.89778 | | **0.89780** |
| 54 | 6 | *1800** | **0.73073** | *1800** | **0.73073** |
| | 7 | | **0.77405** | | **0.77405** |
| | 8 | | **0.81019** | | **0.81019** |
| | 9 | | **0.84091** | | **0.84091** |
| | 10 | | **0.86721** | | **0.86721** |
| | 11 | | **0.88999** | | **0.88999** |
| 57 | 6 | *1800** | **0.72443** | *1800** | **0.72443** |
| | 7 | | **0.76742** | | **0.76742** |
| | 8 | | **0.80340** | | **0.80340** |
| | 9 | | **0.83374** | | **0.83374** |
| | 10 | | **0.85963** | | **0.85973** |
| | 11 | | 0.88201 | | **0.88236** |
| | 12 | | 0.90162 | | **0.90196** |
| 60 | 7 | *1800** | **0.76180** | *1800** | **0.76180** |
| | 8 | | 0.79709 | | **0.79717** |
| | 9 | | 0.82704 | | **0.82706** |
| | 10 | | **0.85275** | | **0.85275** |
| | 11 | | 0.87503 | | **0.87506** |
| | 12 | | **0.89449** | | **0.89449** |
| | 13 | | 0.91166 | | **0.91176** |
| 63 | 7 | *1800** | **0.75669** | *1800** | **0.75669** |
| | 8 | | **0.79182** | | **0.79182** |
| | 9 | | **0.82176** | | **0.82176** |
| | 10 | | **0.84731** | | **0.84731** |
| | 11 | | **0.86910** | | **0.86910** |
| | 12 | | **0.88833** | | **0.88833** |
| | 13 | | 0.90528 | | **0.90529** |
| 66 | 7 | *1800** | **0.75217** | *1800** | **0.75217** |
| | 8 | | **0.78713** | | **0.78713** |
| | 9 | | 0.81647 | | **0.81650** |
| | 10 | | 0.84143 | | **0.84186** |
| | 11 | | 0.86329 | | **0.86367** |
| | 12 | | 0.88242 | | **0.88258** |
| | 13 | | 0.89927 | | **0.89946** |
| | 14 | | 0.91417 | | **0.91434** |
| 69 | 7 | *1800** | 0.74599 | *1800** | **0.74644** |
| | 8 | | 0.78074 | | **0.78119** |
| | 9 | | 0.81008 | | **0.81053** |
| | 10 | | 0.83536 | | **0.83562** |
| | 11 | | 0.85726 | | **0.85749** |

| $n$ | $p$ | BnB Original | | BnB Variant | |
|---|---|---|---|---|---|
| | | ***time*** | $f$ | ***time*** | $f$ |
| | 12 | | 0.87623 | | **0.87640** |
| | 13 | | 0.89297 | | **0.89326** |
| | 14 | | 0.90791 | | **0.90820** |
| 72 | 8 | *1800** | **0.77640** | *1800** | **0.77640** |
| | 9 | | **0.80562** | | **0.80562** |
| | 10 | | **0.83068** | | **0.83068** |
| | 11 | | **0.85237** | | **0.85237** |
| | 12 | | 0.87131 | | **0.87134** |
| | 13 | | 0.88810 | | **0.88811** |
| | 14 | | **0.90302** | | **0.90302** |
| | 15 | | **0.91626** | | **0.91626** |

**Table 3.** Comparison of the two Branch & Bound methods in the fictitious matrices

The following conclusions can be drawn from Table 3:

- The two Branch & Bound methods managed to reach a conclusion in data matrices of up to 36 variables inclusive. In these matrices the optimal solutions were found for each of the values of $p$ considered. However, we can see that in all these matrices the computation time taken by the variant was clearly shorter. Table 4 shows the percentage reduction in computation time for the databases in which both methods reached a conclusion ($n \leq 36$). As we can see, this percentage reduction ranged from 24% to nearly 46%.

- For matrices with more variables ($n \geq 39$), the execution of both methods was interrupted when they reached the set time of 1800 seconds. In these cases it is not guaranteed that the solutions obtained are the optimal ones. Nevertheless, in all these matrices we can see that the results obtained by the variant are always better than or as good as those obtained by the *BnB Original* method. Specifically, of the 79 instances (combinations of matrices and values of $p$) they reached the same value in 54 and the value obtained by the variant was strictly better in 25.

| $n$ | % reduction | $n$ | % reduction |
|---|---|---|---|
| 18 | 45.97 | 30 | 28.23 |
| 21 | 39.01 | 33 | 35.00 |
| 24 | 44.33 | 36 | 24.54 |
| 27 | 37.48 | | |

**Table 4.** Time reductions achieved by *BnB Variant*

Analogous tests were performed with the databases from the literature. Table 5 shows the results obtained.

| ***Base*** | $p$ | BnB Original | | BnB Variant | |
|---|---|---|---|---|---|
| | | ***time*** | $f$ | ***time*** | $f$ |
| *YearPredictionMsdn* | 10 | *1800** | 0.19744 | *1800** | **0.20038** |
| | 11 | | 0.20173 | | **0.20321** |
| | 12 | | 0.20469 | | **0.20609** |
| | 13 | | 0.20689 | | **0.20842** |
| | 14 | | 0.20889 | | **0.21104** |
| | 15 | | 0.21078 | | **0.21271** |
| | 16 | | 0.21257 | | **0.21428** |

| ***Base*** | ***p*** | **BnB Original** | | **BnB Variant** | |
|---|---|---|---|---|---|
| | | ***time*** | ***f*** | ***time*** | ***f*** |
| | 17 | | 0.21448 | | **0.21594** |
| | 18 | | 0.21616 | | **0.21743** |
| | 19 | | 0.21758 | | **0.21885** |
| *Buzz in social media- TomsHardware* | 10 | *1800** | **0.96050** | *1800** | **0.96050** |
| | 11 | | 0.96072 | | **0.96078** |
| | 12 | | 0.96109 | | **0.96110** |
| | 13 | | 0.96142 | | **0.96144** |
| | 14 | | **0.96178** | | **0.96178** |
| | 15 | | **0.96195** | | **0.96195** |
| | 16 | | 0.96214 | | **0.96220** |
| | 17 | | 0.96246 | | **0.96264** |
| | 18 | | **0.96266** | | **0.96266** |
| | 19 | | 0.96277 | | **0.96292** |
| | 20 | | 0.96291 | | **0.96306** |
| *Buzz in social media- Twitter* | 8 | *1800** | **0.93380** | *1800** | **0.93380** |
| | 9 | | **0.93423** | | **0.93423** |
| | 10 | | **0.93454** | | **0.93454** |
| | 11 | | 0.93456 | | **0.93471** |
| | 12 | | 0.93456 | | **0.93489** |
| | 13 | | 0.93476 | | **0.93497** |
| | 14 | | 0.93491 | | **0.93505** |
| | 15 | | 0.93500 | | **0.93508** |
| | 16 | | 0.93508 | | **0.93524** |
| *Parkinson Speech Dataset* | 3 | *0.629* | 0.38551 | *0.314* | 0.38551 |
| | 4 | | 0.40038 | | 0.40038 |
| | 5 | | 0.40650 | | 0.40650 |
| | 6 | | 0.41360 | | 0.41360 |
| *Geographical Original of Music* | 7 | *1800** | **0.16314** | *1800** | **0.16314** |
| | 8 | | 0.17114 | | **0.17202** |
| | 9 | | 0.18107 | | **0.18165** |
| | 10 | | **0.19030** | | **0.19030** |
| | 11 | | 0.19657 | | **0.19710** |
| | 12 | | 0.20444 | | **0.20471** |
| | 13 | | 0.20970 | | **0.21178** |
| | 14 | | 0.21428 | | **0.21606** |
| *Facebook Comment Volume Dataset - Feature_Variant_1* | 6 | *38.084* | 0.31653 | *33.738* | 0.31653 |
| | 7 | | 0.31813 | | 0.31813 |
| | 8 | | 0.31980 | | 0.31980 |
| | 9 | | 0.32139 | | 0.32139 |
| | 10 | | 0.32199 | | 0.32199 |
| | 11 | | 0.32260 | | 0.32260 |
| *Facebook Comment Volume Dataset - Feature_Variant_2* | 6 | *63.297* | 0.31946 | *19.313* | 0.31946 |
| | 7 | | 0.32249 | | 0.32249 |
| | 8 | | 0.32357 | | 0.32357 |
| | 9 | | 0.32439 | | 0.32439 |
| | 10 | | 0.32519 | | 0.32519 |
| | 11 | | 0.32587 | | 0.32587 |
| *Facebook Comment Volume Dataset - Feature_Variant_3* | 6 | *122.466* | 0.34604 | *16.477* | 0.34604 |
| | 7 | | 0.34720 | | 0.34720 |
| | 8 | | 0.34822 | | 0.34822 |
| | 9 | | 0.34902 | | 0.34902 |
| | 10 | | 0.34956 | | 0.34956 |
| | 11 | | 0.35002 | | 0.35002 |

| *Base* | $p$ | BnB Original | | BnB Variant | |
|---|---|---|---|---|---|
| | | *time* | $f$ | *time* | $f$ |
| *Superconductivity Data* | 9 | *1800** | 0.66388 | *1800** | **0.67094** |
| | 10 | | 0.67094 | | **0.67866** |
| | 11 | | 0.67477 | | **0.68126** |
| | 12 | | 0.67840 | | **0.68469** |
| | 13 | | 0.68133 | | **0.68860** |
| | 14 | | 0.68678 | | **0.69165** |
| | 15 | | 0.68839 | | **0.69450** |
| | 16 | | 0.69374 | | **0.69901** |
| | 17 | | 0.69682 | | **0.69939** |
| *Parkinson Telemonitoring* | 3 | *0.019* | 0.19317 | *0.008* | 0.19317 |
| | 4 | | 0.21508 | | 0.21508 |
| | 5 | | 0.22935 | | 0.22935 |

**Table 5.** Comparison of the two Branch & Bound methods for the databases

The conclusions from Table 5 are very similar to those from Table 3:

- The two Branch & Bound methods managed to reach a conclusion in databases 4, 6, 7, 8 and 10, which are those with the lowest numbers of variables. In these databases both methods obtained all the optimal solutions. However, with these databases we can also see that *BnB Variant* always manages to reduce the computation time taken by *BnB Original*. Table 6 shows the percentage reduction. As we can see, this percentage reduction ranges from 11% to nearly 87%.

- For the remaining databases the execution of both methods was interrupted when they reached the set time of 1800 seconds, and therefore the optimal result is not guaranteed. Nevertheless, in all these databases we can see that the results obtained by *BnB Variant* are always better than or as good as those obtained by the *BnB Original* method. Specifically, of the 47 instances the same value was reached in 9 and the value obtained by *BnB Variant* was strictly better in the other 38.

| *DataBase* | % reduction |
|---|---|
| *Parkinson Speech Dataset with Multiple Types of Sound Recordings* | 50.08 |
| *Facebook Comment Volume Dataset - Feature_Variant_1* | 11.41 |
| *Facebook Comment Volume Dataset - Feature_Variant_2* | 69.49 |
| *Facebook Comment Volume Dataset - Feature_Variant_3* | 86.55 |
| *Parkinson Telemonitoring* | 57.89 |

**Table 6.** Time reductions achieved by *BnB Variant* in the databases where the execution was not interrupted

In conclusion, it can be seen that in instances with a moderate number of variables ($n \leq 36$ in the matrices and as much as $n$ = 53 in the databases from the literature) both Branch & Bound methods manage to finish and therefore to reach the optimal solution, although *BnB Variant* is faster in reaching the solution in a substantially shorter time. For the instances with a higher number of variables (n > 36 in the fictitious matrices and n $\geq$ 68 in the databases from the literature) neither of the two Branch & Bound methods guarantees that the optimal solution is obtained, since the procedure was interrupted to avoid excessive computation times. Nevertheless, in these instances *BnB Variant* always achieved the better result. The *BnB Original* method, in turn, reached the best result in 63 of the total of 126 instances (79 simulated and 47

from the literature). In other words, *BnB Variant* was strictly better than *BnB Original* in half of the total of 126 instances and the two methods were equally good in the other half of the instances. In short, using the information supplied by a heuristic executed beforehand (such as the one proposed in section 5) makes the resulting Branch & Bound method (in this case *BnB Variant*) more efficient and effective than the original method.

### 6.4 Comparison of the Branch & Bound method (BnB Variant) with other variable selection strategies

In this subsection the results obtained by the *BnB Variant* method are compared with other well-known methods and selection strategies in the literature for regression, as well as with general-purpose optimization software. Specifically, the methods with which it is compared are the following:

- Forward (Fwd): This method, Efroymson (1960), is a very well-known classical method present in well-known statistical software such as SPSS, StatGraphics, etc.

- GARROTE: As indicated in subsection 1.2 this method was proposed in Breiman (1996). In our case we used the algorithm proposed by Yuan and Lin (2006, 2007).

- LASSO: As indicated in subsection 1.2 this method was proposed in Tibshirani (1996). In this study the adaptation of the Coordinate Descent algorithm proposed in Wu and Lange (2009) was implemented.

- LARS: As indicated in subsection 1.2 this method was proposed in Efron et al. (2004).

- LocalSolver: In this case, the commercial software LocalSolver (version 8.0 Academic-Desktop) optimizer was chosen, since it has been used successfully in various fields, as presented in www.localsolver.com. LocalSolver uses its own programming language with data structures that are especially useful in routing problems.

Table 7 shows the results obtained by the *BnB Variant* method and these 4 classical methods for the matrices defined in subsection 6.1. The best result in indicated in bold type.

| $n$ | $p$ | *BnB Variant* | Fwd | GARROTE | LASSO | LARS | LocalSolver |
|---|---|---|---|---|---|---|---|
| 18 | 2 | **0.54267** | **0.54267** | 0.54066 | 0.54066 | 0.54066 | 0.43903 |
| | 3 | **0.69606** | 0.69593 | 0.69593 | 0.69593 | 0.69593 | 0.65868 |
| | 4 | **0.81110** | 0.81062 | 0.81062 | 0.81062 | 0.81062 | 0.74651 |
| 21 | 3 | **0.65949** | **0.65949** | **0.65949** | **0.65949** | **0.65949** | 0.56768 |
| | 4 | **0.76851** | **0.76851** | 0.76765 | 0.76765 | 0.76765 | 0.68873 |
| | 5 | **0.85269** | **0.85269** | 0.85191 | 0.85191 | 0.85191 | 0.67849 |
| 24 | 3 | **0.63320** | **0.63320** | 0.63194 | 0.63194 | 0.63194 | 0.56213 |
| | 4 | **0.73722** | **0.73722** | 0.73524 | 0.73524 | 0.73524 | 0.59255 |
| | 5 | **0.81792** | **0.81792** | 0.81640 | 0.81640 | 0.81640 | 0.69482 |
| 27 | 3 | **0.61118** | **0.61118** | 0.60829 | 0.60975 | 0.60975 | 0.57576 |
| | 4 | **0.71220** | **0.71220** | 0.71010 | 0.71010 | 0.71010 | 0.66636 |
| | 5 | **0.79005** | **0.79005** | 0.78828 | 0.78838 | 0.78838 | 0.66949 |
| | 6 | **0.85219** | 0.85216 | 0.85097 | 0.85097 | 0.85097 | 0.73062 |
| 30 | 4 | **0.69208** | **0.69208** | 0.69108 | 0.69108 | 0.69108 | 0.64478 |
| | 5 | **0.76798** | **0.76798** | 0.76688 | 0.76688 | 0.76688 | 0.70922 |
| | 6 | **0.82844** | 0.82840 | 0.82758 | 0.82681 | 0.82681 | 0.71688 |

| $n$ | $p$ | *BnB Variant* | Fwd | GARROTE | LASSO | LARS | LocalSolver |
|---|---|---|---|---|---|---|---|
| | 7 | **0.87801** | 0.87788 | 0.87660 | 0.87660 | 0.87660 | 0.73669 |
| 33 | 4 | **0.67477** | 0.67472 | 0.67205 | 0.67205 | 0.67205 | 0.62256 |
| | 5 | **0.74914** | 0.74850 | 0.74740 | 0.74740 | 0.74740 | 0.62966 |
| | 6 | **0.80806** | 0.80757 | 0.80631 | 0.80689 | 0.80689 | 0.70167 |
| | 7 | **0.85630** | 0.85586 | 0.85436 | 0.85519 | 0.85519 | 0.75820 |
| 36 | 4 | **0.66261** | **0.66261** | 0.65935 | 0.66087 | 0.66087 | 0.53197 |
| | 5 | **0.73451** | **0.73451** | 0.73273 | 0.73273 | 0.73273 | 0.61930 |
| | 6 | **0.79233** | **0.79233** | 0.79176 | 0.79176 | 0.79176 | 0.66214 |
| | 7 | **0.83947** | 0.83930 | 0.83909 | 0.83909 | 0.83909 | 0.70202 |
| | 8 | **0.87850** | **0.87850** | **0.87850** | **0.87850** | **0.87850** | 0.81362 |
| 39 | 4 | **0.64955** | **0.64955** | 0.64919 | 0.64919 | 0.64919 | 0.53442 |
| | 5 | **0.72105** | 0.72040 | 0.71948 | 0.71896 | 0.71896 | 0.65749 |
| | 6 | **0.77763** | 0.77733 | 0.77580 | 0.77580 | 0.77580 | 0.69195 |
| | 7 | **0.82393** | **0.82393** | 0.82261 | 0.82261 | 0.82261 | 0.73780 |
| | 8 | **0.86273** | **0.86273** | 0.86188 | 0.86188 | 0.86188 | 0.75304 |
| 42 | 5 | **0.70964** | **0.70964** | 0.70766 | 0.70766 | 0.70766 | 0.66351 |
| | 6 | **0.76520** | **0.76520** | 0.76307 | 0.76307 | 0.76307 | 0.72921 |
| | 7 | **0.81105** | **0.81105** | 0.80938 | 0.80856 | 0.80856 | 0.68832 |
| | 8 | **0.84907** | 0.84891 | 0.84722 | 0.84722 | 0.84722 | 0.70835 |
| | 9 | **0.88116** | 0.88088 | 0.88015 | 0.88015 | 0.88015 | 0.80192 |
| 45 | 5 | **0.70046** | 0.70030 | 0.69927 | 0.69927 | 0.69927 | 0.61609 |
| | 6 | **0.75609** | **0.75609** | 0.75515 | 0.75515 | 0.75515 | 0.64211 |
| | 7 | **0.80077** | 0.80074 | 0.80049 | 0.79980 | 0.79980 | 0.73821 |
| | 8 | **0.83806** | 0.83798 | 0.83707 | 0.83695 | 0.83755 | 0.77931 |
| | 9 | **0.86954** | **0.86954** | 0.86887 | 0.86887 | 0.86887 | 0.75767 |
| | 10 | **0.89654** | **0.89654** | 0.89532 | 0.89532 | 0.89532 | 0.75821 |
| 48 | 5 | **0.69201** | 0.69161 | 0.68951 | 0.68951 | 0.68951 | 0.63415 |
| | 6 | **0.74630** | 0.74597 | 0.74423 | 0.74423 | 0.74423 | 0.64660 |
| | 7 | **0.79068** | 0.79063 | 0.78935 | 0.78935 | 0.78935 | 0.73581 |
| | 8 | **0.82773** | **0.82773** | 0.82645 | 0.82716 | 0.82716 | 0.74157 |
| | 9 | **0.85916** | **0.85916** | 0.85833 | 0.85867 | 0.85867 | 0.73368 |
| | 10 | **0.88585** | **0.88585** | 0.88541 | 0.88543 | 0.88543 | 0.78585 |
| 51 | 6 | **0.73570** | 0.73537 | 0.73360 | 0.73383 | 0.73383 | 0.63479 |
| | 7 | **0.77994** | 0.77976 | 0.77849 | 0.77849 | 0.77849 | 0.69594 |
| | 8 | **0.81687** | 0.81675 | 0.81468 | 0.81468 | 0.81468 | 0.74895 |
| | 9 | **0.84808** | 0.84800 | 0.84586 | 0.84586 | 0.84586 | 0.73550 |
| | 10 | **0.87477** | 0.87463 | 0.87304 | 0.87304 | 0.87304 | 0.80737 |
| | 11 | **0.89780** | 0.89760 | 0.89615 | 0.89615 | 0.89615 | 0.79860 |
| 54 | 6 | **0.73073** | 0.73033 | 0.72933 | 0.72933 | 0.72933 | 0.64310 |
| | 7 | **0.77405** | 0.77393 | 0.77256 | 0.77242 | 0.77242 | 0.67321 |
| | 8 | **0.81019** | 0.81012 | 0.80876 | 0.80876 | 0.80876 | 0.70052 |
| | 9 | **0.84091** | 0.84077 | 0.84019 | 0.83909 | 0.83909 | 0.73799 |
| | 10 | **0.86721** | 0.86716 | 0.86583 | 0.86583 | 0.86583 | 0.80871 |
| | 11 | **0.88999** | **0.88999** | 0.88879 | 0.88879 | 0.88879 | 0.78571 |
| 57 | 6 | **0.72443** | **0.72443** | 0.72213 | 0.72213 | 0.72213 | 0.69335 |
| | 7 | **0.76742** | **0.76742** | 0.76502 | 0.76502 | 0.76502 | 0.69931 |
| | 8 | **0.80340** | 0.80313 | 0.80088 | 0.80088 | 0.80088 | 0.75657 |
| | 9 | **0.83374** | 0.83337 | 0.83198 | 0.83191 | 0.83191 | 0.73174 |
| | 10 | **0.85973** | 0.85928 | 0.85861 | 0.85861 | 0.85861 | 0.78841 |
| | 11 | **0.88236** | 0.88181 | 0.88130 | 0.88130 | 0.88130 | 0.81375 |
| | 12 | **0.90196** | 0.90147 | 0.90064 | 0.90119 | 0.90119 | 0.81888 |
| 60 | 7 | **0.76180** | 0.76145 | 0.76004 | 0.76004 | 0.76004 | 0.70807 |
| | 8 | **0.79717** | 0.79703 | 0.79597 | 0.79597 | 0.79597 | 0.71227 |

| $n$ | $p$ | *BnB Variant* | Fwd | GARROTE | LASSO | LARS | LocalSolver |
|---|---|---|---|---|---|---|---|
| | 9 | **0.82706** | 0.82702 | 0.82608 | 0.82570 | 0.82570 | 0.73724 |
| | 10 | **0.85275** | 0.85263 | 0.85148 | 0.85148 | 0.85148 | 0.81139 |
| | 11 | **0.87506** | 0.87495 | 0.87413 | 0.87383 | 0.87383 | 0.75926 |
| | 12 | **0.89449** | **0.89449** | 0.89358 | 0.89348 | 0.89348 | 0.81451 |
| | 13 | **0.91176** | 0.91164 | 0.91067 | 0.91062 | 0.91062 | 0.83932 |
| 63 | 7 | **0.75669** | 0.75631 | 0.75469 | 0.75469 | 0.75469 | 0.72183 |
| | 8 | **0.79182** | 0.79180 | 0.78927 | 0.78927 | 0.78927 | 0.72149 |
| | 9 | **0.82176** | 0.82155 | 0.81912 | 0.81912 | 0.81912 | 0.77438 |
| | 10 | **0.84731** | **0.84731** | 0.84527 | 0.84487 | 0.84487 | 0.78130 |
| | 11 | **0.86910** | **0.86910** | 0.86749 | 0.86749 | 0.86749 | 0.82835 |
| | 12 | **0.88833** | **0.88833** | 0.88649 | 0.88649 | 0.88649 | 0.78806 |
| | 13 | **0.90529** | 0.90528 | 0.90416 | 0.90416 | 0.90416 | 0.83055 |
| 66 | 7 | **0.75217** | 0.75184 | 0.74989 | 0.74989 | 0.74989 | 0.67840 |
| | 8 | **0.78713** | 0.78664 | 0.78401 | 0.78401 | 0.78401 | 0.69603 |
| | 9 | **0.81650** | 0.81608 | 0.81441 | 0.81441 | 0.81441 | 0.77959 |
| | 10 | **0.84186** | 0.84128 | 0.84020 | 0.84020 | 0.84020 | 0.74353 |
| | 11 | **0.86367** | 0.86322 | 0.86204 | 0.86204 | 0.86204 | 0.76765 |
| | 12 | **0.88258** | 0.88242 | 0.88137 | 0.88137 | 0.88137 | 0.80298 |
| | 13 | **0.89946** | 0.89927 | 0.89832 | 0.89832 | 0.89832 | 0.83559 |
| | 14 | **0.91434** | 0.91417 | 0.91358 | 0.91358 | 0.91358 | 0.84046 |
| 69 | 7 | **0.74644** | 0.74558 | 0.74428 | 0.74428 | 0.74428 | 0.67092 |
| | 8 | **0.78119** | 0.78028 | 0.77928 | 0.77928 | 0.77928 | 0.70632 |
| | 9 | **0.81053** | 0.80961 | 0.80895 | 0.80895 | 0.80895 | 0.75631 |
| | 10 | **0.83562** | 0.83488 | 0.83384 | 0.83384 | 0.83384 | 0.79285 |
| | 11 | **0.85749** | 0.85680 | 0.85605 | 0.85577 | 0.85577 | 0.77502 |
| | 12 | **0.87640** | 0.87586 | 0.87508 | 0.87508 | 0.87508 | 0.80975 |
| | 13 | **0.89326** | 0.89275 | 0.89232 | 0.89232 | 0.89232 | 0.82294 |
| | 14 | **0.90820** | 0.90778 | 0.90752 | 0.90752 | 0.90752 | 0.79817 |
| 72 | 8 | **0.77640** | **0.77640** | 0.77558 | 0.77558 | 0.77558 | 0.71184 |
| | 9 | **0.80562** | **0.80562** | 0.80523 | 0.80523 | 0.80523 | 0.71434 |
| | 10 | **0.83068** | **0.83068** | 0.83012 | 0.83012 | 0.83012 | 0.75642 |
| | 11 | **0.85237** | **0.85237** | 0.85108 | 0.85108 | 0.85108 | 0.76582 |
| | 12 | **0.87134** | 0.87131 | 0.87007 | 0.87007 | 0.87007 | 0.80658 |
| | 13 | **0.88811** | 0.88810 | 0.88691 | 0.88691 | 0.88691 | 0.79588 |
| | 14 | **0.90302** | **0.90302** | 0.90183 | 0.90183 | 0.90183 | 0.82840 |
| | 15 | **0.91626** | **0.91626** | 0.91516 | 0.91516 | 0.91516 | 0.85412 |

**Table 7.** Comparison of the *BnB Variant* method with the *Fwd, GARROTE*, *LASSO,* LARS and *LocalSolver* methods in the fictitious matrices

From Table 7 the following conclusions can be drawn:

- Of the 105 instances analyzed the BnB Variant method achieved the best solution in all of them. In the instances where the execution of this method ended within the maximum time set ($n \leq 36$) this result is obvious, since the solutions it obtained are optimal. However, it is interesting that it also achieved the best results in all the instances where its execution was interrupted ($n \geq 39$) and where it was not guaranteed that the optimal result was obtained.

- The results obtained by the remaining methods are worse. Only the Fwd method obtained the best result in an acceptable number of cases (41) while LARS, LASSO and GARROTE only achieved

the best result in 2 and LocalSolver in none. Table 8 shows the number of instances in which each method obtained the best solution.

| ***BnB Variant*** | ***Fwd*** | **GARROTE** | **LASSO** | **LARS** | ***LocalSolver*** |
|---|---|---|---|---|---|
| 105 | 41 | 2 | 2 | 2 | 0 |

**Table 8.** Number of instances in which each method obtained the best solution in the matrices

To determine whether the solutions obtained by the BnB Variant method are significantly better, various tests of means (t-tests) were conducted with the values obtained in Table 7. Specifically, 5 tests were performed, one for each method with which the *BnB Variant* was compared. Table 9 shows the results of these tests. As we can see, the differences are significant in all 4 comparisons.

| Test | Mean | std | t-statistic | p-value |
|---|---|---|---|---|
| BnB Variant *versus* Fwd | 0.00018019 | 0.00023465 | 7.869 | < 0.001 |
| BnB Variant *versus* GARROTE | 0.00142981 | 0.00063621 | 23.029 | < 0.001 |
| BnB Variant *versus* LASSO | 0.00142981 | 0.00063621 | 23.029 | < 0.001 |
| BnB Variant *versus* LARS | 0.00142981 | 0.00063621 | 23.029 | < 0.001 |
| BnB Variant *versus* LocalSolver | 0.08424790 | 0.02927326 | 29.491 | < 0.001 |

**Table 9.** T-tests to compare the results of *BnB Variant* with those of the other methods (in the matrices)

In order to show more clearly some results of Table 7, two figures have been added (Figures 2 and 3). These figures correspond respectively with the instances of values $n$ = 66, $p$ = 14, and $n$ = 69, $p$ = 14.

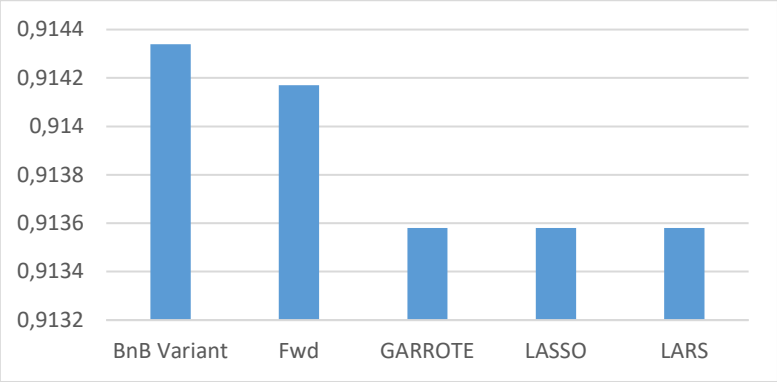


**Figure 2.** Results obtained by different methods with $n$ = 66, $p$ = 14

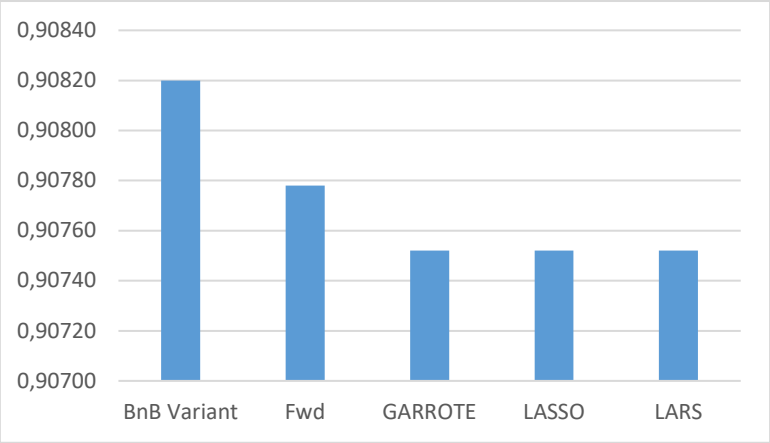


**Figure 3.** Results obtained by different methods with $n$ = 69, $p$ = 14

Finally, analogous tests were performed with the databases described in subsection 6.2. Table 10 shows the results.

| Bases | $p$ | BnB Variant | Fwd | GARROTE | LASSO | LARS | LocalSolver |
|---|---|---|---|---|---|---|---|
| *YearPredictionMsdn* | 10 | **0.20038** | 0.19679 | 0.11615 | 0.11055 | 0.18660 | 0.06749 |
| | 11 | **0.20321** | 0.20173 | 0.11913 | 0.11232 | 0.18782 | 0.11100 |
| | 12 | **0.20609** | 0.20469 | 0.12014 | 0.11328 | 0.19157 | 0.11101 |
| | 13 | **0.20842** | 0.20689 | 0.12033 | 0.11437 | 0.19303 | 0.07807 |
| | 14 | **0.21104** | 0.20874 | 0.12135 | 0.11542 | 0.19315 | 0.11164 |
| | 15 | **0.21271** | 0.21078 | 0.15439 | 0.11644 | 0.19574 | 0.08520 |
| | 16 | **0.21428** | 0.21215 | 0.15506 | 0.11915 | 0.19736 | 0.08527 |
| | 17 | **0.21594** | 0.21405 | 0.15620 | 0.15429 | 0.20164 | 0.05492 |
| | 18 | **0.21743** | 0.21560 | 0.15704 | 0.15433 | 0.20282 | 0.05611 |
| | 19 | **0.21885** | 0.21758 | 0.15779 | 0.15546 | 0.20320 | 0.05891 |
| *Buzz in social media-TomsHardware* | 10 | **0.96050** | 0.95970 | 0.95311 | 0.95777 | 0.95391 | 0.77995 |
| | 11 | **0.96078** | 0.96051 | 0.95311 | 0.95778 | 0.95391 | 0.81097 |
| | 12 | **0.96110** | 0.96093 | 0.95312 | 0.95778 | 0.95392 | 0.87183 |
| | 13 | **0.96144** | 0.96142 | 0.95312 | 0.95778 | 0.95398 | 0.80413 |
| | 14 | **0.96178** | 0.96151 | 0.95312 | 0.95784 | 0.95462 | 0.80768 |
| | 15 | **0.96195** | 0.96163 | 0.95312 | 0.95790 | 0.95485 | 0.87744 |
| | 16 | **0.96220** | 0.96176 | 0.95312 | 0.95791 | 0.95501 | 0.92331 |
| | 17 | **0.96264** | 0.96212 | 0.95312 | 0.95804 | 0.95501 | 0.88705 |
| | 18 | **0.96266** | 0.96232 | 0.95313 | 0.95815 | 0.95512 | 0.81928 |
| | 19 | **0.96292** | 0.96272 | 0.95313 | 0.95896 | 0.95513 | 0.90233 |
| | 20 | **0.96306** | 0.96283 | 0.95313 | 0.95910 | 0.95537 | 0.92878 |
| *Buzz in social media-Twitter* | 8 | **0.93380** | 0.93310 | 0.92327 | 0.92044 | 0.93124 | 0.86851 |
| | 9 | **0.93423** | 0.93347 | 0.92359 | 0.92061 | 0.93156 | 0.83065 |
| | 10 | **0.93454** | 0.93391 | 0.92360 | 0.92131 | 0.93358 | 0.82844 |
| | 11 | **0.93471** | 0.93429 | 0.92360 | 0.92132 | 0.93360 | 0.89971 |
| | 12 | **0.93489** | 0.93451 | 0.92360 | 0.92132 | 0.93384 | 0.89352 |
| | 13 | **0.93497** | 0.93469 | 0.92360 | 0.92136 | 0.93390 | 0.92516 |
| | 14 | **0.93505** | 0.93479 | 0.92360 | 0.92150 | 0.93391 | 0.91923 |
| | 15 | **0.93508** | 0.93487 | 0.92360 | 0.92150 | 0.93394 | 0.88079 |
| | 16 | **0.93524** | 0.93501 | 0.92360 | 0.92163 | 0.93427 | 0.92437 |
| *Parkinson Speech Dataset* | 3 | **0.38551** | **0.38551** | 0.38380 | 0.37728 | **0.38551** | 0.36817 |
| | 4 | **0.40038** | 0.39093 | 0.38835 | 0.38435 | 0.39016 | 0.00885 |
| | 5 | **0.40650** | 0.39490 | 0.38970 | 0.38476 | 0.39430 | 0.05730 |
| | 6 | **0.41360** | **0.41360** | 0.39216 | 0.38491 | 0.40044 | 0.05026 |
| | 7 | **0.16314** | 0.15560 | 0.10447 | 0.10104 | 0.15031 | 0.09945 |

| Bases | $p$ | BnB Variant | Fwd | GARROTE | LASSO | LARS | LocalSolver |
|---|---|---|---|---|---|---|---|
| *Geographical Original of Music* | 8 | **0.17202** | 0.17037 | 0.10579 | 0.10878 | 0.16089 | 0.08168 |
| | 9 | **0.18165** | 0.17994 | 0.10735 | 0.10899 | 0.17455 | 0.06500 |
| | 10 | **0.19030** | 0.18870 | 0.11046 | 0.10981 | 0.18027 | 0.07874 |
| | 11 | **0.19710** | 0.19533 | 0.11181 | 0.12520 | 0.18357 | 0.08605 |
| | 12 | **0.20471** | 0.20217 | 0.11339 | 0.12669 | 0.18996 | 0.10874 |
| | 13 | **0.21178** | 0.20970 | 0.11772 | 0.12737 | 0.19345 | 0.11381 |
| | 14 | **0.21606** | 0.21389 | 0.11852 | 0.12766 | 0.20281 | 0.10766 |
| *Facebook Comment Volume Dataset - Feature_Variant_1* | 6 | **0.31653** | **0.31653** | 0.29373 | 0.00058 | 0.00078 | 0.13749 |
| | 7 | **0.31813** | **0.31813** | 0.29438 | 0.00058 | 0.00081 | 0.13750 |
| | 8 | **0.31980** | 0.31959 | 0.29616 | 0.00098 | 0.00084 | 0.16228 |
| | 9 | **0.32139** | **0.32139** | 0.31171 | 0.00103 | 0.00087 | 0.16571 |
| | 10 | **0.32199** | **0.32199** | 0.31193 | 0.00103 | 0.02027 | 0.17638 |
| | 11 | **0.32260** | **0.32260** | 0.31199 | 0.00104 | 0.03658 | 0.27331 |
| *Facebook Comment Volume Dataset - Feature_Variant_2* | 6 | **0.31946** | 0.31654 | 0.29409 | 0.00035 | 0.00045 | 0.14157 |
| | 7 | **0.32249** | 0.31742 | 0.29560 | 0.00035 | 0.00055 | 0.14159 |
| | 8 | **0.32357** | 0.31829 | 0.31051 | 0.00059 | 0.00057 | 0.17901 |
| | 9 | **0.32439** | 0.31904 | 0.31208 | 0.00060 | 0.00057 | 0.18541 |
| | 10 | **0.32519** | 0.31963 | 0.31210 | 0.00061 | 0.00062 | 0.18863 |
| | 11 | **0.32587** | 0.32470 | 0.31322 | 0.00062 | 0.04576 | 0.27267 |
| *Facebook Comment Volume Dataset - Feature_Variant_3* | 6 | **0.34604** | 0.34469 | 0.32079 | 0.00059 | 0.00074 | 0.16549 |
| | 7 | **0.34720** | 0.34551 | 0.32111 | 0.00059 | 0.00098 | 0.16550 |
| | 8 | **0.34822** | 0.34659 | 0.32294 | 0.00101 | 0.00101 | 0.18558 |
| | 9 | **0.34902** | 0.34750 | 0.33927 | 0.00114 | 0.00107 | 0.18812 |
| | 10 | **0.34956** | 0.34832 | 0.33954 | 0.00115 | 0.00117 | 0.19416 |
| | 11 | **0.35002** | 0.34888 | 0.33957 | 0.00117 | 0.03881 | 0.29946 |
| *Superconductivity Data* | 9 | **0.67094** | 0.65799 | 0.49645 | 0.59077 | 0.63812 | 0.52677 |
| | 10 | **0.67866** | 0.66353 | 0.50139 | 0.59132 | 0.64373 | 0.48502 |
| | 11 | **0.68126** | 0.66736 | 0.50139 | 0.59515 | 0.65042 | 0.47507 |
| | 12 | **0.68469** | 0.67161 | 0.50498 | 0.60245 | 0.66113 | 0.60153 |
| | 13 | **0.68860** | 0.67625 | 0.50499 | 0.60827 | 0.66983 | 0.62785 |
| | 14 | **0.69165** | 0.67990 | 0.52905 | 0.60876 | 0.67219 | 0.47725 |
| | 15 | **0.69450** | 0.68617 | 0.52982 | 0.61103 | 0.67219 | 0.56584 |
| | 16 | **0.69901** | 0.68853 | 0.54157 | 0.61212 | 0.67315 | 0.57063 |
| | 17 | **0.69939** | 0.69281 | 0.55827 | 0.61282 | 0.67497 | 0.55451 |
| *Parkinson Telemonitoring* | 3 | **0.19317** | **0.19317** | 0.17146 | 0.17146 | **0.19317** | 0.02816 |
| | 4 | **0.21508** | **0.21508** | 0.17243 | 0.17441 | **0.21508** | 0.10168 |
| | 5 | **0.22935** | **0.22935** | 0.17499 | 0.17499 | **0.22935** | 0.12257 |

**Table 10.** Comparison of the *BnB Variant* method with the *Fwd*, *GARROTE*, *LASSO*, *LARS* and *LocalSolver* methods in the databases

The conclusions obtained from Table 10 are similar to those obtained from Table 7: the *BnB Variant* method achieved the best results in all the databases analyzed and for all the values of $p$ considered. Only the Fwd and LARS methods managed to reach the best value in some cases (10 and 4 respectively). The remaining methods, as shown in Table 11, did not achieve the best value in any cases.

| *BnB Variant* | *Fwd* | *GARROTE* | *LASSO* | *LARS* | *LocalSolver* |
|---|---|---|---|---|---|
| 72 | 10 | 0 | 0 | 4 | 0 |

**Table 11.** Number of instances in which each method obtained the best solution in the databases

As with the matrices, to determine whether the solutions obtained by the *BnB Variant* method are significantly better, various tests of means (t-tests) were conducted with the values obtained in Table 10. The results of these tests are shown in Table 12. As we can see, the differences are significant in all 5 comparisons.

| Tests | mean | std | t-statistic | p-value |
|---|---|---|---|---|
| BnB Variant *versus* Fwd | 0.00288319 | 0.00404840 | 6.043 | < 0.001 |
| BnB Variant *versus* GARROTE | 0.04968625 | 0.05360251 | 7.865 | < 0.001 |
| BnB Variant *versus* LASSO | 0.11795000 | 0.12730407 | 7.862 | < 0.001 |
| BnB Variant *versus* LARS | 0.08912236 | 0.13605584 | 5.558 | < 0.001 |
| BnB Variant *versus* LocalSolver | 0.12635014 | 0.07185972 | 14.920 | < 0.001 |

**Table 12.** T-tests to compare the results of *BnB Variant* with those of the other methods (databases)

In short, the *BnB Variant* method always obtained the best results. Only in a few instances did some of the proposed methods manage to equal this best result (especially Fwd). Moreover, the statistical tests show that the mean results obtained by *BnB Variant* are also significantly better. Figure 4 shows the results corresponding to the database *Buzz in social media – TomsHardware* and $p$ = 20.

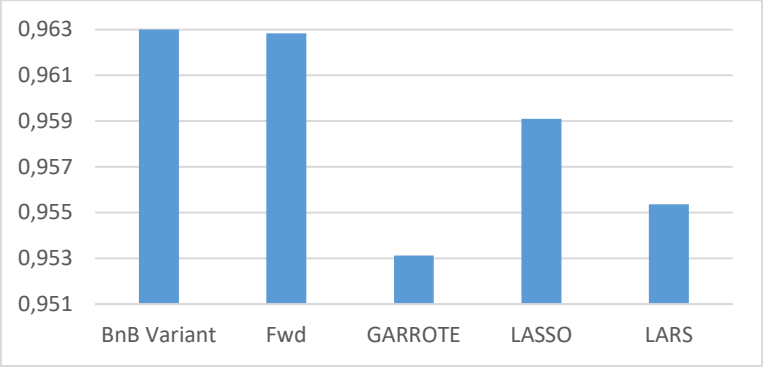


**Figure 4.** Results obtained by different methods with the *Buzz in social media – TomsHardware* database and $p$ = 20

It is important to point out that in our method, unlike others, the computational complexity does not depend on the number of cases (as explained in the Appendix), which makes it possible to work with databases containing a large number of cases (10000 cases in the simulated data and more than 500000 in some of the databases from the literature).

In addition, it should be noted that since most applications of this problem (statistical studies of various kinds) do not require obtaining solutions immediately, a moderate maximum execution time can be set. In our case a maximum time of 1800 seconds was considered appropriate.

### 6.5 Experiments in high dimension matrices

The matrices and databases used in the previous sub-sections have different dimensions. Thus the matrices defined in sub-section 6.1 have a number of cases $m$ = 100000 and the number of variables varies from $n$ = 18 to $n$ = 72. Among the databases defined in subsection 6.2, the smallest are *Parkinson Speech Dataset with Multiple Types of Sound Recordings* ($m$ = 1040, $n$ = 27), and Parkinson Telemonitoring ($m$ = 5875, $n$ = 20). While the largest databases are *YearPredictionMsdn* ($m$ = 515345, $n$ = 90) and *Buzz in social media-Twitter* ($m$ = 583250, $n$ = 77).

That is, the two largest databases have about 45-46 million real numbers. In short, some of the matrices and databases have a dimension that can be considered high or at least moderately high. However, in many real applications, larger databases are used. It would be interesting to check the performance of our method in these larger databases or matrices.

Thus, in this section we perform some experiments with the aim of checking the performance of our method in larger databases. For this purpose, a set of 4 matrices are designed in the same way as those designed in sub-section 6.1, but with the following differences: the number of cases is $m$ = 1000000 – that is, 10 times more than the matrices defined in sub-section 6.1; and the number of variables varies from $n$ = 120 to $n$ = 210. So the number of real numbers varies from 120 million to 210 million.

As in previous sections values of $p$ was considered from $\lfloor n \cdot 0.1 \rfloor + 1$ to $\lfloor n \cdot 0.2 \rfloor + 1$. These values are shown in Table 13.

| **Matrix #** | $n$ | $p$ range |
|---|---|---|
| **1** | 120 | 13 – 25 |
| **2** | 150 | 16 – 3 1 |
| **3** | 180 | 19 – 37 |
| **4** | 210 | 22 – 43 |

**Table 13.** Matrices and values of $p$ considered

Table 14 shows the results obtained in theses matrices by the *BnB Variant* method and the 4 classical methods considered in previous sections. The best result is indicated in bold type.

| $n$ | $p$ | ***BnB Variant*** | **Fwd** | **GARROTE** | **LASSO** | **LARS** |
|---|---|---|---|---|---|---|
| 120 | 13 | **0.80527** | 0.80522 | 0.80442 | 0.80442 | 0.80450 |
| | 14 | **0.81893** | 0.81891 | 0.81824 | 0.81817 | 0.81817 |
| | 15 | **0.83116** | **0.83116** | 0.83045 | 0.83045 | 0.83046 |
| | 16 | **0.84216** | **0.84216** | 0.84128 | 0.84128 | 0.84150 |
| | 17 | **0.85209** | **0.85209** | 0.85134 | 0.85133 | 0.85152 |
| | 18 | **0.86111** | **0.86111** | 0.86022 | 0.86018 | 0.86061 |
| | 19 | **0.86936** | 0.86935 | 0.86859 | 0.86847 | 0.86876 |
| | 20 | **0.87692** | **0.87692** | 0.87617 | 0.87607 | 0.87632 |
| | 21 | **0.88387** | **0.88387** | 0.88317 | 0.88315 | 0.88335 |
| | 22 | **0.89029** | **0.89029** | 0.8896 | 0.88952 | 0.88976 |
| | 23 | **0.89623** | **0.89623** | 0.89554 | 0.89550 | 0.89580 |
| | 24 | **0.90176** | **0.90176** | 0.90107 | 0.90097 | 0.90133 |
| | 25 | **0.90689** | **0.90689** | 0.90631 | 0.90627 | 0.90650 |
| 150 | 16 | **0.83991** | 0.83989 | 0.83907 | 0.83899 | 0.83907 |
| | 17 | **0.84980** | **0.84980** | 0.84901 | 0.84886 | 0.84892 |
| | 18 | **0.85878** | **0.85878** | 0.85794 | 0.85790 | 0.85798 |
| | 19 | **0.86695** | **0.86695** | 0.86620 | 0.86605 | 0.86619 |
| | 20 | **0.87446** | 0.87444 | 0.87366 | 0.87365 | 0.87377 |
| | 21 | **0.88138** | 0.88133 | 0.88062 | 0.88050 | 0.88069 |
| | 22 | **0.88776** | 0.88769 | 0.88698 | 0.88697 | 0.88700 |
| | 23 | **0.89366** | 0.89358 | 0.89290 | 0.89276 | 0.89285 |
| | 24 | **0.89913** | 0.89905 | 0.89844 | 0.89819 | 0.89834 |
| | 25 | **0.90423** | 0.90416 | 0.90351 | 0.90350 | 0.90353 |
| | 26 | **0.90899** | 0.90891 | 0.90826 | 0.90819 | 0.90833 |
| | 27 | **0.91345** | 0.91336 | 0.91273 | 0.91271 | 0.91287 |

| $n$ | $p$ | *BnB Variant* | Fwd | GARROTE | LASSO | LARS |
|---|---|---|---|---|---|---|
| | 28 | **0.91762** | 0.91753 | 0.91699 | 0.91696 | 0.91707 |
| | 29 | **0.92155** | 0.92146 | 0.92092 | 0.92091 | 0.92108 |
| | 30 | **0.92525** | 0.92515 | 0.92469 | 0.92454 | 0.92479 |
| | 31 | **0.92873** | 0.92863 | 0.92825 | 0.92812 | 0.92825 |
| | 19 | **0.86354** | 0.86350 | 0.86288 | 0.86281 | 0.86288 |
| | 20 | **0.87108** | 0.87101 | 0.87035 | 0.87022 | 0.87035 |
| | 21 | **0.87799** | 0.87791 | 0.87725 | 0.87715 | 0.87725 |
| | 22 | **0.88437** | 0.88428 | 0.88363 | 0.88358 | 0.88363 |
| | 23 | **0.89025** | 0.89018 | 0.88948 | 0.88946 | 0.88961 |
| | 24 | **0.89572** | 0.89567 | 0.89502 | 0.89485 | 0.89502 |
| | 25 | **0.90083** | 0.90077 | 0.90010 | 0.89999 | 0.90011 |
| | 26 | **0.90559** | 0.90552 | 0.90485 | 0.90478 | 0.90487 |
| | 27 | **0.91004** | 0.90997 | 0.90929 | 0.90917 | 0.90931 |
| 180 | 28 | **0.91420** | 0.91415 | 0.91352 | 0.91347 | 0.91354 |
| | 29 | **0.91812** | 0.91807 | 0.91746 | 0.91738 | 0.91746 |
| | 30 | **0.92180** | 0.92177 | 0.92121 | 0.92100 | 0.92116 |
| | 31 | **0.92527** | 0.92523 | 0.92469 | 0.92454 | 0.92469 |
| | 32 | **0.92855** | 0.92851 | 0.92800 | 0.92788 | 0.92797 |
| | 33 | **0.93165** | 0.93161 | 0.93111 | 0.93105 | 0.93107 |
| | 34 | **0.93457** | 0.93455 | 0.93406 | 0.93399 | 0.93404 |
| | 35 | **0.93735** | 0.93734 | 0.93684 | 0.93674 | 0.93684 |
| | 36 | **0.93999** | 0.93998 | 0.93950 | 0.93940 | 0.93950 |
| | 37 | **0.94250** | 0.94249 | 0.94202 | 0.94201 | 0.94205 |
| | 22 | **0.88130** | 0.88126 | 0.88043 | 0.88038 | 0.88050 |
| | 23 | **0.88717** | 0.88713 | 0.88637 | 0.88628 | 0.88637 |
| | 24 | **0.89263** | 0.89258 | 0.89183 | 0.89163 | 0.89183 |
| | 25 | **0.89771** | 0.89767 | 0.89696 | 0.89680 | 0.89697 |
| | 26 | **0.90243** | 0.90242 | 0.90175 | 0.90158 | 0.90175 |
| | 27 | **0.90687** | 0.90686 | 0.90615 | 0.90599 | 0.90619 |
| | 28 | **0.91101** | **0.91101** | 0.91029 | 0.91028 | 0.91033 |
| | 29 | 0.91490 | **0.91491** | 0.91421 | 0.91400 | 0.91421 |
| | 30 | **0.91857** | 0.91856 | 0.91790 | 0.91769 | 0.91790 |
| | 31 | **0.92202** | 0.92200 | 0.92135 | 0.92117 | 0.92135 |
| | 32 | **0.92528** | 0.92525 | 0.92459 | 0.92450 | 0.92469 |
| 210 | 33 | **0.92837** | 0.92833 | 0.92774 | 0.92763 | 0.92775 |
| | 34 | **0.93129** | 0.93125 | 0.93066 | 0.93065 | 0.93072 |
| | 35 | **0.93406** | 0.93402 | 0.93345 | 0.93338 | 0.93350 |
| | 36 | **0.93669** | 0.93665 | 0.93615 | 0.93597 | 0.93615 |
| | 37 | **0.93919** | 0.93916 | 0.93865 | 0.93856 | 0.93867 |
| | 38 | **0.94157** | 0.94154 | 0.94101 | 0.94098 | 0.94106 |
| | 39 | **0.94387** | 0.94381 | 0.94332 | 0.94319 | 0.94340 |
| | 40 | **0.94604** | 0.94598 | 0.94555 | 0.94547 | 0.94555 |
| | 41 | **0.94811** | 0.94806 | 0.94763 | 0.94755 | 0.94765 |
| | 42 | **0.95009** | 0.95005 | 0.94963 | 0.94942 | 0.94966 |
| | 43 | **0.95199** | 0.95194 | 0.95154 | 0.95149 | 0.95156 |

**Table 14.** Comparison of the *BnB Variant* method with the *Fwd, GARROTE*, L*ASSO and* LARS

The conclusions obtained from Table 14 are similar to those obtained in sub-section 6.4: the *BnB Variant* method achieved the best results in almost all the matrices analyzed and for all the values of $p$ considered (69 of 70 instances). Only the Fwd method managed to reach the best value in some cases (15). The remaining methods, as shown in Table 15, did not achieve the best value in any cases.

| ***BnB Variant*** | ***Fwd*** | **GARROTE** | **LASSO** | **LARS** |
|---|---|---|---|---|
| 69 | 15 | 0 | 0 | 0 |

**Table 15.** Number of instances in which each method obtained the best solution in the matrices

As in previous sections, to determine whether the solutions obtained by the *BnB Variant* method are significantly better, various tests of means (t-tests) were conducted with the values obtained in Table 14. The results of these tests are shown in Table 16. As we can see, the differences are significant in all 4 comparisons

| Test | Mean | std | t-statistic | p-value |
|---|---|---|---|---|
| BnB Variant *versus* Fwd | 0.00003771 | 0.00003051 | 10.342 | < 0.0001 |
| BnB Variant *versus* GARROTE | 0.00067529 | 0.00011173 | 50.568 | < 0.0001 |
| BnB Variant *versus* LASSO | 0.00076085 | 0.00011939 | 53.318 | < 0.0001 |
| BnB Variant *versus* LARS | 0.00062629 | 0.00012120 | 43.232 | < 0.0001 |

**Table 16.** T-tests to compare the results of *BnB Variant* with those of the other methods (in the matrices)

In short, the *BnB Variant* method always obtained the best results. Only in a few instances did some of the proposed methods manage to equal this best result (especially Fwd). Moreover, the statistical tests show that the mean results obtained by *BnB Variant* are also significantly better. Figures 5 and 6 show the results corresponding respectively with the instances of values $n$ = 150, $p$ = 31, and $n$ = 210, $p$ = 43.

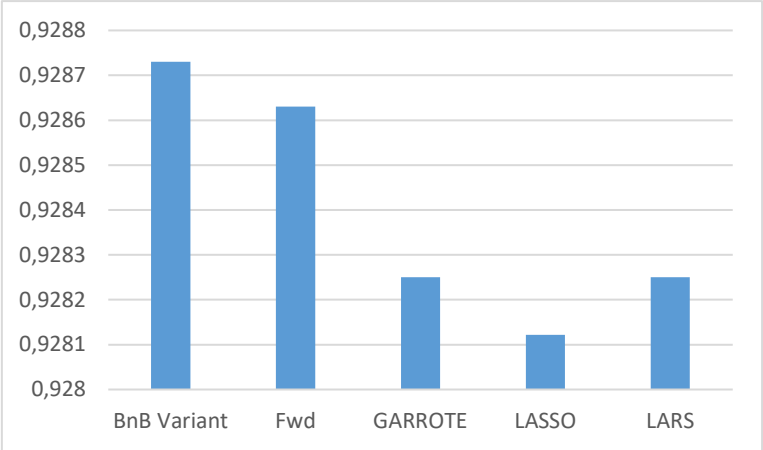


**Figure 6.** Results obtained by different methods with $n$ = 150, $p$ = 31

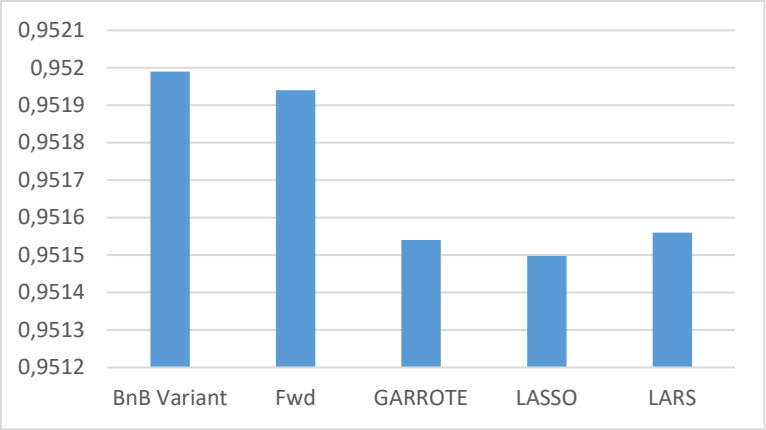


**Figure 7.** Results obtained by different methods with $n$ = 210, $p$ = 43

As mentioned above, the objective of our method is not so much to find the most relevant variables, but rather the subset of variables that achieve the highest prediction accuracy. However, sometimes it can also be important to determine the most relevant variables. It is therefore interesting to examine the ability of our method to select the most relevant variables. From the definition of the coefficients $\beta_i$ and the correlation population matrix $L$ (see sub-section 6.1) the most significant variables are 3, 6, 9, 12, .... The variables selected by our method in the tests of this section have been recorded and, in fact, all of them have been verified as belonging to this set (index variables multiple of 3). Therefore, we understand that our method is also capable of selecting the most relevant variables.

## 7. Conclusions

This study deals with the variable selection problem for linear regression. This model has a wide range of applications, as explained in detail in the introduction. In this study, two Branch & Bound methods have been proposed to obtain optimum solutions: an original method and a variant. The computational effort of both methods does not depend on the number of cases. It allows for the use of these methods in large database. In addition, these methods obtain the optimal solution in a short time in databases with a moderate number of initial variables. The main difference with respect to the original method is that the variant uses the information provided by a fast heuristic executed beforehand. Computational experiments show that the use of this information makes the variant a more efficient and effective method.

On the other hand, several computational experiments show that our Branch & Bound variant performs better compared with other well-known feature selection methods for linear regression.

In summary, appropriate use of the information provided by a heuristic can improve the effectiveness and efficiency of some exact methods. This statement has been proved for the problem of variable selection in linear regression in this paper. A challenge for the future is to apply this idea to other complex problems.

## Acknowledgments

This work was partially supported by FEDER funds and the Spanish Ministry of Economy and Competitiveness (Projects ECO2016-76567-C4-2-R and PID2019-104263RB-C44), the Regional

Government of “Castilla y León”, Spain (Project BU329U14 and BU071G19), the Regional Government of “Castilla y León” and FEDER funds (Project BU062U16 and COV2000375).

**References**

Aneiros, G., Ferraty, F., & Vieu, P. 2015. Variable selection in partial linear regression with functional covariate. *Statistics*, *49*(6), 1322-1347.

Bandura, R., 2008. A Survey of composite indices measuring country performance: 2008 Update. Office of Development Studies. United Nations Development Programme, Working Paper.

Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., & Chanussot, J., 2012. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE journal of selected topics in applied earth observations and remote sensing*, *5*(2), 354-379.

Bioucas-Dias, J. M., & Plaza, A., 2010. Hyperspectral unmixing: Geometrical, statistical, and sparse regression-based approaches. Image and Signal Processing for Remote Sensing XVI (Vol. 7830, p. 78300A). International Society for Optics and Photonics.

Blancas Peral, F.J., Gonzalez Lozano M., Guerrero Casas F.M. and Lozano Oyola M. 2010. Indicadores Sintéticos de Turismo Sostenible: Una aplicación para los destinos turísticos de Andalucia. *Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA*, Rect@ 11, 85- 118.

Breiman, L., 1995. Better subset regression using the nonnegative garrote. *Technometrics* 37, 4, 373-384.

Brusco, M.J., 2014. A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. *Computational Statistics & Data Analysis* 77, 38-53.

Brusco, M.J., Singh, R., Steinley, D., 2009. Variable neighborhood search heuristics for selecting a subset of variables in principal component analysis. *Psychometrika* 74, 705-726.

Brusco, M.J., Steinley, D., 2011. Exact and approximate algorithms for variable selection in linear discriminant analysis. *Computational Statistics & Data Analysis* 55 (1), 123-131.

Chatterjee, S., Steinhaeuser, K., Banerjee, A., Chatterjee, S., & Ganguly, A. (2012, April). Sparse group lasso: Consistency and climate applications. In *Proceedings of the 2012 SIAM International Conference on Data Mining* (pp. 47-58). Society for Industrial and Applied Mathematics.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., 2004. Least angle regression. *Annals of Statistics* 32 (2), 407-499.

Efroymson, M., 1960. Multiple regression analysis. *Mathematical methods for digital computers*, 1:191–203.

Fan, J. and Li, R., 2001. Variable selection via non concave penalized likelihood and its oracle properties. *Journal of the American Statistical Association* 96, 1348-1360.

Févotte, C., Torrésani, B., Daudet, L., & Godsill, S. J., 2008. Sparse linear regression with structured priors and application to denoising of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(1), 174-185.

Févotte, C., Daudet, L., Godsill, S. J., & Torrésani, B., 2006. Sparse regression with structured priors: Application to audio denoising. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* (Vol. 3, pp. III-III). IEEE.

Filzmoser, P., Gschwandtner, M., & Todorov, V., 2012. Review of sparse methods in regression and classification with application to chemometrics. *Journal of Chemometrics*, 26(3-4), 42-51.

Gijbels, I. and Vrinssen, I., 2015. Robust nonnegative garrote variable selection in linear regression. *Computational Statistics & Data Analysis* 85, 1-22.

Hans, C., Dobra, A. and West, M., 2007. Shotgun stochastic search for "large p" regression. *Journal of the American Statistical Association,* 102 **(**478), 507-516.

Iordache, M. D., Bioucas-Dias, J. M., & Plaza, A., 2014. Collaborative sparse regression for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(1), 341-354.

Kilinc, B. K., Asikgil, B., Erar, A., and Yazici, B., 2016. Variable selection with genetic algorithm and multivariate adaptive regression splines in the presence of multicollinearity. *International Journal of Advanced and Applied Sciences*, 3(12), 26-31

Li, Y., Nan, B., & Zhu, J. (2015). Multivariate sparse group lasso for the multivariate multiple linear regression with an arbitrary group structure. *Biometrics*, *71*(2), 354-363.

Luo, S. and Ghosal, S. (2016). Forward selection and estimation in high dimensional single index models. *Statistical Methodology*, 33, 172-179

Mateos, G., Bazerque, J. A., & Giannakis, G. B., 2010. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, *58*(10), 5262-5276.

Meiri, R. and Zahavi, J., 2006. Using simulated annealing to optimize the feature selection problem in marketing applications. *European Journal of Operational Research* 171, 842-858.

Mundry, R. and Nunn, C.L., 2009. Stepwise model fitting and statistical inference: Turning noise into signal pollution. *The American Naturalist* 173 (1), 119-123.

Nardo, M., Saisana, M., Saltelli, A., Tarantola, S., Hoffman, A. and Giovannini, E., 2005a. Handbook on constructing composite indicators: methodology and user guide. OECD Statistics, Working Paper 2005/3.

Nardo, M., Saisana, M., Saltelli, A. and Tarantola, S., 2005b. Tools for composite indicators building. European Commission. Joint Research Centre. Working Paper 21682.

Naylor, T., 1977. Técnicas de simulación en computadoras. Limusa.

Pacheco J., Casado S. and Núñez L., 2009. A Variable Selection Method based on Tabu Search for Logistic Regression Models. *European Journal of Operational Research* 199 (2), 506–511.

Pacheco, J., Casado, S., and Porras, S., 2013. Exact methods for variable selection in principal component analysis: Guide functions and pre-Selection. *Computational Statistics & Data Analysis* 57, 95-111.

Parada Rico, S.E., Fiallo Leal, E. and Blasco-Blasco, O. 2015. Construcción de indicadores sintéticos basados en juicio experto: aplicación a una medida integral de excelencia académica. *Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA*, Rect@, 16, 51-67.

Rish, I., & Grabarnik, G. (2014). *Sparse modeling: theory, algorithms, and applications*. CRC press.

Rubinstein, R. Y., 1981. Simulation and the Monte Carlo method, Wiley.

Sayed, G. I., Khoriba, G., & Haggag, M. H. (2018). A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence*, *48*(10), 3462-3481.

Sayed, G. I., Tharwat, A., & Hassanien, A. E. (2019). Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. *Applied Intelligence*, 49(1), 188-205.

Tibshirani, R., 1996. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B* 58 (1), 267-278.

Vounou, M., Nichols, T. E., Montana, G., & Alzheimer's Disease Neuroimaging Initiative. (2010). Discovering genetic associations with high-dimensional neuroimaging phenotypes: A sparse reduced-rank regression approach. *Neuroimage*, *53*(3), 1147-1159.

Wang, Y., & Feng, L. (2019). A new hybrid feature selection based on multi-filter weights and multi-feature weights. *Applied Intelligence*, 49(12), 4033-4057.

Wu, T. T., & Lange, K. 2008. Coordinate descent algorithms for lasso penalized regression. The Annals of Applied Statistics, *2*(1), 224-244.

Yuan, M., & Lin, Y., 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(1), 49-67.

Yuan, M., & Lin, Y., 2007. On the non-negative garrotte estimator. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *69*(2), 143-161.

**Appendix 1. Corollary used by our Branch & Bound methods**

Corollary

$\forall\ p, p' \in \{1, \ldots, n\}$, if $p < p'$ then $g(p) \leq g(p')$.

Proof:

By simplifying, we can define $p' = p + 1$, and $S_p^* = \{1, .., p\}$.

Let us also define $S' = \{1, \ldots, p, p+1\}$. Obviously, $S_p^* \subset S'$. Therefore

$$g(p) = f(S_p^*) \leq f(S') \leq max\{f(S): S \subset V, |S| = p+1\} = g(p+1).$$

**Appendix 2. Calculation of the objective function**

a) Pre-process for calculating the objective function

To facilitate the calculation of the objective function $f(S)$ a pre-calculation is initially performed before beginning to execute the algorithms. The matrix of independent variables $X$ and the vector of the dependent variable $Y$ are considered, as defined in section 2. In other words

$X = (x_{ij})_{i=1,\dots,m;j=1,\dots,n}$ e $Y = (y_i)_{i=1,\dots,m}$

The pre-process consists of the following steps:

- The matrix $X^* = (x_{ij}^*)_{i=1,\dots,m;j=1,\dots,n}$ is calculated

where $x_{ij}^* = \left(\frac{x_{ij}-\overline{x}_j}{\sqrt{m}\cdot s_j}\right), i = 1,\dots,m; j = 1,\dots,n$

and $\overline{x}_j$ and $s_j^2$ are respectively the mean and the sample variance of the variable $j; j = 1,\dots,n$.

- The matrix $Y^* = (y_i^*)_{i=1,\dots,m}$ is calculated

where $y_i^* = \left(\frac{y-\overline{y}}{\sqrt{m}\cdot s_y}\right), i = 1,\dots,m;$

and $\overline{y}$ and $s_y^2$ are respectively the mean and the sample variance of the variable $Y$.

- The matrix $R = X^{*\prime} \cdot X^*$ and the vector $H = X^{*\prime} \cdot Y^*$ are calculated. Note that $R$ is the matrix of correlations of the independent variables. We denote the elements of $R$, $j, j' = 1,\dots,n$, as $r_{jj'}$, l; and the elements of $H$, $j = 1,\dots,n$, as $h_j$.

The matrix $R$ and the vector $H$ will be used in calculating $f(S)$ for the various sets $S$ in the algorithms proposed in this paper. This pre-process requires $\Theta(n^2 \cdot m)$ operations. However, it is executed only once.

b) Calculation of the objective function $f(S)$

Let there be a set $S \subset V$ of size $p$. We denote $S = \{s(1), s(2), \dots, s(p)\}$. The calculation of $f(S)$ consists of the following steps:

- The matrix $R^S = (r_{jj'}^S)$ is constructed, where $r_{jj'}^S = r_{S(j)S(j')}$, $j, j' = 1,\dots,p$;

- The vector $H^S = (h_j^S)$, where $h_j^S = h_{S(j)}$ $\quad j = 1,\dots,p$;

- Find the inverse of the matrix $R^S$: $(R^S)^{-1}$

- Calculate the vector of coefficients $B = (R^S)^{-1} \cdot H^S$. We denote the elements of B, $j = 1,\dots,p$ as $\beta_j$.

- Calculate the value of $f(S) = \sum_{j=1}^{p}\sum_{j'=1}^{p} \beta_j \cdot \beta_{j'} \cdot r_{jj'}^S$ .

This calculation requires $\Theta(p^2)$ operations and is therefore independent of the number of cases $m$ and of the initial number of variables $n$.

**Appendix 3. Analysis of the complexity of our Branch & Bound methods**

As described in Section 3, the Branch & Bound methods are based on a recursive exploration in the set of solutions. This set is represented by a search tree. Each node of this tree corresponds with a subset of solutions. In the exploration of the set of solutions corresponding to a node (Pseudocode 1), a variable $a$ is selected and the corresponding set is divided into two subsets: one with the variable $a$ fixed and another with the variable $a$ forbidden. The process is then repeated with each subset. Since at most $p_0$ variables are selected ($p_0$ fixed variables), there are $p_0$ divisions until a solution with maximum size $p_0$ is reached. Therefore, $\theta(2^{p_0})$ nodes are explored. On the other hand, the number of variables that are examined to be selected is limited by $n$, so determining the variable $a$ assumes $\theta(n)$ calculations of the function $f$ (line 5 of Pseudocode 1). Therefore, the Branch & Bound methods calculate the objective function of $\theta(n \cdot 2^{p_0})$ solutions $S$, with $|S| \leq p_0$. As the calculation of each $f(S)$ requires $\theta(|S|^2)$ operations, the complexity of our methods is $\theta({p_0}^2 \cdot n \cdot 2^{p_0})$.

**Appendix 4. Basic ideas of traditional methods and our Branch & Bound methods**

- Our Branch & Bound methods find the global optimum solution to the problem defined by (1) – (3) in Section 2.

- The Forward Method finds a solution to the same problem defined by (1) – (3), but this solution is and approximate solution (not necessarily the global optimal). The method works as follows:

1. Do $S = \emptyset$

Repeat

2. Determine $i^* = argmax\{ f(S \cup \{i\}): i \in V - S\}$
3. Make $S = S \cup \{i^*\}$

Until $|S| = p$

Note that the problem defined by (1) - (3) can also be defined compactly as:

$$min_\beta \|Y - X^*\beta\|_2$$

subject to:

$$\|\beta\|_0 = p$$

where

$X^* = (\mathbf{1}|X)$ and $\mathbf{1^t} = (1,1, \ldots 1) \in R^m$, that is $X^*$ is matrix $X$ extended with vector $\mathbf{1}$

$\beta^t = (\beta_0, \beta_1, \cdots, \beta_n)$ is the vector of the coefficients of the variables and the independent coefficient

$\|\|_r$ = $r$ - Norme

- The LASSO Method solves the following model:

$$min_\beta \ \|Y - X^*\beta\|_2 + \lambda \cdot \|\beta\|_1$$

where $\lambda$ is a previously established positive parameter. The higher the value of $\lambda$ the fewer the variables with a coefficient $\beta_i \neq 0$ are selected – that is, the fewer variables are selected. To find the solution, methods based on the Coordinate Descent algorithm are usually used.

- The LARS method solves the same model as LASSO but using strategies analogous to the Forward method – that is, selecting in each step a variable to enter in the solution.

- The GARROTE method calculates its coefficients $\beta_i^g$ as follows:

$$\beta_i^g = \left[1 - \frac{\lambda}{\left(\beta_i^r\right)^2}\right]_+ \cdot \beta_i^r$$

where the values $\beta_i^r$ are the coefficients of the linear regression model with all variables, $[z]_+$ is the positive part of a real number $z$, and $\lambda$ is a previously established positive parameter. As in previous model, the higher the value of $\lambda$ the fewer the variables with coefficient $\beta_i \neq 0 -$ that is, the fewer variables are selected.