A Comparative Study Between Feature Selection Algorithms

Víctor Hugo Medina Garcia[1(✉)], Jorge Rodriguez Rodriguez[1(✉)], and Miguel Angel Ospina Usaquén[2(✉)]

[1] District University "Francisco José de Caldas", Bogotá, Colombia
{vmedina, jerodriguezr}@udistrital.edu.co
[2] Santo Tomas University, Bogotá, Colombia
miguelospina@usantotomas.edu.co

Abstract. In this paper, we show a comparative study between four algorithms used in features selection; these are: decision trees, entropy measure for ranking features, estimation of distribution algorithms, and the bootstrapping algorithm. Likewise, the features selection is highlighted as the most representative task in the elimination of noise, in order to improve the quality of the dataset. Subsequently, each algorithm is described in order that the reader understands its function. Then the algorithms are applied using different data sets and obtaining the results in the selection. Finally, the conclusions of this investigation are presented.

Keywords: Features selection · Bootstrapping algorithm · Decision trees
Entropy theory · Estimation of distribution algorithms

1 Introduction

Usually, database contain millions of tuples and thousands of attributes, presenting dependencies among attributes [1]. The essential purpose of data preprocessing is to manipulate and transform each dataset, making the information contained within them more accessible and coherent [2, 3]. The data preprocessing process involves choosing an outcome measure to evaluate, potential influencer variables, cleansing the data, creating features and generating data sets to provide to beyond core for automated analysis. Data preprocessing is an important step in the knowledge discovery process, because quality decisions must be based on quality data. The main tasks of data preprocessing are: Data Cleaning, Missing Values, Noisy Data, Data Integration, Data Transformation and Feature Selection [37].

A feature selection algorithm is a computational solution that is motivated by a certain definition of relevance. However, the relevance of a feature as seen from the inductive learning perspective may have several definitions depending on the objective that is looked for. An irrelevant feature is not useful for induction, but not all relevant features are necessarily useful for induction [34]. The existing selection algorithms focus, mainly, on finding relevant features [4]. This is a process, in which the most relevant characteristics are selected, improving the knowledge discovery database.

Y. Tan et al. (Eds.): DMBD 2018, LNCS 10943, pp. 65–76, 2018.
https://doi.org/10.1007/978-3-319-93803-5_7

2 Problem

Feature selection, applied as a data preprocessing stage to data mining, proves to be valuable in that it eliminates the irrelevant features that make algorithms ineffective. Sometimes the percentage of instances correctly classified is higher if a previous feature selection is applied, since the data to be mined will be noise-free [5]. This is usually attributed to the "curse of dimensionality" or to the fact that irrelevant features decrease the signal to noise ratio. In addition, many algorithms become computationally intractable when the dimensionality is high [33].

Feature selection task is divided into four stages [6]: the first one determines the possible set of attributes to perform the representation of the problem, then the subset of attributes generated in step one is evaluated. Subsequently, it is examined whether the selected subset satisfies the search criteria. These processes can be classified differently depending on the stage in which we focus, in order to make this distinction in three categories [7]: Filters, Wrappers [8, 9] and hybrids [10].

In Filters methods, the selection procedure is performed independently of the evaluation function. In these can be distinguished four different evaluation measures: distance, information, dependence and consistency. Respective examples of each one of these measures can be found in [11–13]. Wrappers methods combine search in the attribute space with the machine learning evaluating the set of attributes and choosing the most appropriate. Hybrid models present the advantages that Filters and Wrappers models provide. Since the feature selection is applicable to dissimilar real situations, it is difficult to reach a consensus as to which is the best possible choice; this makes possible multiple algorithms of this type [14].

3 Methodology

This section describes the type of scientific research used in the article along with the research method and the development methodology. The type of scientific research applied in this paper is descriptive-exploratory with an experimental approach. According to the formal research process, a hypothetical - deductive method was used in which a hypothesis was formulated, which through empirical validation was validated through deductive reasoning. It was established, based on the experimentation, a mechanism for weighting the algorithm evaluation indicators in such a way that it was possible to evaluate said mechanism when changing the dataset.

The following tasks are defined to obtain the results, after applying the appropriate algorithms in feature selection:

Collection, integration, and data preprocessing. In this phase, the collection and integration of different datasets, data transformation; as the case may be, and cleaning in order to eliminate noise.

Definition and application of tests of the algorithms used for feature selection. Based on the tests performed with the synthetic data, the imputation and the evaluation of the imputation indicators were applied to apply the algorithms.

Review of test results. The analysis of the data obtained in the allocation with the algorithms was performed. Likewise, the complexity of the algorithms was calculated, in order to determine their feasibility of implementation.

The basis of this paper, is based on the analysis and choice of algorithms for feature selection. In this stage the case studies are of vital importance, the conclusions of this research and machine learning algorithms used in feature selection.

4 Feature Selection Algorithms

4.1 Decision Trees

A decision trees are a representation in which each set of possible conclusions is implicitly established by a list of known class samples [15]. Decision tree has a simple form that efficiently classifies new data [16, 17].

These trees are considered as an important tool for data mining; compared to other algorithms, decision trees are faster and more accurate [18]. Learning in a decision trees is a method to approximate an objective function of discrete values, in which the learning function is represented by a tree. These learning methods are the most popular inductive inference algorithms and they have thriving application in various machine learning tasks [19–21]. The theory of information provides a mathematical model (Eq. 1) to measure the total disorder in a database.

$$isAver = \sum_1^b \frac{nb}{nt} * \sum_1^c - \frac{nbc}{nb} log_c\left(\frac{nbc}{nb}\right) \tag{1}$$

Where: nb is the number of instances of attribute b, nt is the total number of instances, nbc is the number of instances of attribute b belonging to class c.

4.2 Entropy Measure for Ranking Features

Ranking methods may filter features leading to reduced dimensionality of the feature space. All the process is based on eliminating n attributes, without losing the basic characteristics of the dataset. This algorithm is based on the measure of similarity S that is inversely proportional to the distance D between two instances of n dimensions [22–24]. The distance D is small with instances close to 0 and large with instances close to 1. When the features are numerical, the measure of similarity S of two instances can be as shown in Eq. 2.

$$S_{ij} = e^{-\alpha D_{ij}} \tag{2}$$

Where D_{ij} is the distance between the instances X_{ij} and Y_{ij}, and α a parameter expressed in mathematic terms (Eq. 3).

$$\alpha = \frac{-\ln(0.5)}{D} \tag{3}$$

D is the average distance between the samples in the dataset. In practice it is close to 0.5. The Euclidean distance are calculated as follows (Eq. 4).

$$D_{ij} = \sqrt{\sum_{k=1}^{n} \left(\frac{X_{ik} - X_{jk}}{max_k - min_k} \right)^2} \tag{4}$$

Where n is number of attributes, $\max_k$ and $\min_k$ are the maximum and minimum value used for the normalization of k-attributes. When the attributes are categorical, the hamming distance is used (Eq. 5).

$$S_{ij} = \sum_{k=1}^{n} \frac{|X_{ik} = X_{jk}|}{n} \tag{5}$$

Where $|X_{ik} = X_{jk}|$ is 1, otherwise it is 0. The distribution of all similarities for a given data set is a characteristic of the organization and order of data in an n - dimensional space. This organization may be more or less ordered. Changes in the level of order in a data set are the main criteria for inclusion or exclusion of a feature from the feature set; these changes may be measured by entropy [33]. The algorithm in question compares the entropy for a set of data before and after deleting attributes. For a data set of N instances, the measure of the entropy is (Eq. 6).

$$E = -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(S_{ij} \log(S_{ij}) + (1 - S_{ij}) \log(1 - S_{ij}) \right) \tag{6}$$

The steps of the algorithm (Table 1) are based on sequential backward ranking, and they have been successfully tested on several real world applications.

Table 1. Algorithm steps [33]

1: Start with the initial full set of feature F.
2: For each feature f ∈ F, remove one feature f from F and obtain a subset F_f. Find the difference between entropy for F and entropy for all F_f.
3: Let f_k be a feature such that the difference between entropy for F and entropy for F_{fk} is minimum.
4: Update the set of feature F = F − {f_k}, where " – " is a difference operation on sets.
5: Repeat steps 2 – 4 until there is only one feature in F.

A ranking process may be stopped in any iteration, and may be transformed into a process of selecting features, using the additional criterion mentioned in step 4. This criterion is that the difference between entropy for F and entropy for F_f should be less than the approved threshold value to reduce feature f_k from set F. A computational complexity is the basic disadvantage of this algorithm, and its parallel implementation could overcome the problems of working with large data sets and large number of features sequentially.

4.3 Estimation of Distribution Algorithms (EDAs)

EDA algorithm is a stochastic search technique based on population, which uses a probability distribution model to explore candidates (instances) in a search space [25, 26]. EDAs have been recognized as a strong algorithm to optimize. They have shown a better performance in comparison with evolutionary algorithm, in problems where these have not presented satisfactory results. This is mainly due to the explicit nature of the relations or dependencies between the most important variables associated with some particular problems that are estimated through probability distributions [27, 28].

Table 2 shows the EDA algorithm; in the first place an initial population of individuals is generated. These individuals are evaluated according to an objective or aptitude function. This evaluates how appropriate each individual is as a solution to the problem. Based on this evaluation, a subset of the best individuals is selected. Thus, from this subset it is learnt a probability distribution to be used to sample another population [27].

Table 2. EDA Algorithm pseudo-code [29]

Requires: Candidate size n, the number variables l and the cost function f(·).
1: θ ← initialize (l)
2: repeat
3: D ← sample (P(X; θ), n)
4: C ← select (D, f(·))
5: θ ← estimate (θ, C)
6: until P(X; θ) has converged
Outputs: Probability distribution P(X; θ)

This kind of genetic algorithms are stochastic search techniques that evolve a probability distribution model from a pool of solution candidates, rather than evolving the pool itself. The distribution is adjusted iteratively with the most promising (sub-optimal) solutions until convergence. Hence, they are also known as Estimation of Distribution Algorithms. The generic estimation procedure is shown in Algorithm 1. Step (1) initializes the model parameters θ. Step (2) is the loop that updates the parameters θ until convergence. Step (3) samples a pool S of n candidates from the model. Step (4) ranks the pool according to a cost function f (·) and chooses the top-ranked into B. Step (5) re-estimates the parameters θ from this subset of promising solutions [32]. EDA [30, 31] approximately optimizes a cost function by building a probabilistic model of a pool of promising sub-optimal solutions over a given search space. For very-high dimensional search spaces, storing and updating a large population of candidates may imply a computational burden in both time and memory.

4.4 Bootstrapping Algorithm

An option for a suitable feature selection of is to perform an exhaustive search, but this entails a high complexity that makes it inaccessible. The methods of feature selection must perform a search among the candidates of subsets of attributes, these may be: complete, sequential or random. The complete search also known as exhaustive search, helps to find the optimal result according to the evaluation criteria used; the problem is its high complexity (i.e. O(2n)) that makes it inappropriate in case the number of attributes (n) is high. The random search does not ensure an optimal result, starts with a subset of randomly selected attributes and proceeds, from it, with a sequential search or randomly generating the rest of candidate subsets.

Bootstrapping algorithm replicates all of the classification experiments a large number of times and estimates the solution using the set of these experiments. Because this process is based on random selection, there is a probability that there will be some data from the original set that is not used and others that are involved in more than one subset. Bootstrapping algorithm is divided into four stages: generate subsets of attributes, evaluate each subset, update the weights of each attribute and order them by their weight [35].

In the first stage, the subsets of attributes are generated; each subset will contain a maximum number of attributes that randomly select among the original dataset (there cannot be the same attributes). Each subset is generated independently of the previous one (here can be similar subsets). Both the generated subset number and the maximum number of attributes contained in them will be established by the user.

The evaluation phase consists of classifying the original dataset with each of the subsets of attributes generated in the previous stage. The result is to assign to each subset a goodness, which is the percentage of success of each of the classifications.

In the update, the weights of the attributes of the original data set are assigned. The weight of an attribute is the benefits average of the subsets that contain that attribute.

The last step is to organize the attributes of form descendant according to their weight. This phase generates a list that will be the classification returned by the process.

The difference between the bootstrapping algorithm and the exhaustive method is in the first stage. For example, the exhaustive method with $K = 1$ generates a subset for each feature of the original set. However, if we choose a value of $K = 2$, a subset is generated for each pair of possible features. Therefore, this first step for an exhaustive method depends on the value of K, whereas in the random method it depends on the number of experiments and the maximum number of features. The Table 3 show bootstrapping algorithm, this algorithm divides into two tasks. First, *RankingGenerate*, is the main function and has as input parameters: the classification method that is used to evaluate each subset of features chosen U; the set of features to be treated X; the number of experiments to be performed Ne and the maximum number of features that will intervene in each Na experiment. This function generates a single output parameter L, which corresponds to the feature ranking obtained by applying the features selection algorithm. For this purpose, the *RankingGenerate* function sort the attributes according to their weight. To calculate this value, for each feature ai, the average of the success percentages resulting from applying the classification method U on those subsets that contain the feature ai is calculated. Second, to obtain these subsets of attributes we use

the *SubsetGenerate* function, where each subset is chosen randomly with an equally random size (greater than 1 and less than or equal to *Na*) so that there is no duplicate attribute in the same subset.

Table 3. Bootstrapping algorithm pseudo-code

```
Requires: Criteria evaluate U, features X, Experiments number Ne, Features number Na,
Feature list L.
Function RankingGenerate
    1: S ← SubsetGenerate (X, Ne, Na)
    2: while features subset Si ∈ S
    3: C ← Evaluate(Si, C)
    4: Feature Update Si ∈ S
    End while
    5: L ← Sort(X)
    End RankingGenerate Function
Function SubsetGenerate
    1: for i ← 1 until Ne
    2: n ← GenerateRandomNumber (1, Na)
    3: Si = ChooseFeatures (X, n)
    4: L ← L + Si
    End for
End SubsetGenerate Function
```

In [36] the authors motivate by the fact that the variability from randomization can lead to inaccurate outputs, they propose a deterministic approach. First, they establish several computational complexity results for the exact bootstrap method, in the case of the sample mean. Second, it shows the first efficient, deterministic approximation algorithm (FPTAS) for producing exact bootstrap confidence intervals which, unlike traditional methods, has guaranteed bounds on the approximation error. Third, develop a simple exact algorithm for exact bootstrap confidence intervals based on polynomial multiplication. Last, provide empirical evidence involving several hundred (and in some cases over one thousand) data points that the proposed deterministic algorithms can quickly produce confidence intervals that are substantially more accurate compared to those from randomized methods, and are thus practical alternatives in applications such as clinical trials.

5 Results

5.1 Decision Trees

You can see that they have totally different behaviors and the reasons are:

Soybean dataset: shows behavior in which a low percentage of feature is selected, the highest percentage of selection and the maximum of selected attributes does not exceed 70%. This problem has been called the curse of dimensionality.

Chess dataset: the behavior of these data is totally opposite to the Soybean dataset, since it covers the largest amount of attribute space. Means that most data is relevant and only begins to be deleted with a high features selection percentage.

5.2 Entropy Measure for Ranking Features

For test, this algorithm we used a dataset of four features (X1, X2, X3, and X4) with one thousand instances. Features contain categorical data; therefore, we are used the hamming distance to compute the similarity (Eq. 5) between the instances and then the entropy is computed (Eq. 6). The result generates that the feature X3 is the least relevant; since in computing the difference between the total entropy and the entropy without the feature three, it is the one closest to 0. Therefore, X3 must be removed from the dataset.

5.3 Estimation Distribution Algorithms (EDAs)

We used one dataset applied to solve the classical OneMax problem with 100 variables, using a population size of 30 and 1000 maximum number of evaluations, and 30 candidates per iteration. For the execution of the EDAs algorithm, used Orange Suit with the widget Goldenberry. The result show in the Table 4 [38]. The OneMax is a traditional test problem for evolutionary computation. It involves binary bitstrings of fixed length. An initial population is given. The objective is to evolve some bitstrings to match a prespecified bitstring [39].

Table 4. EDA applied result

Best:{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,0,0,0,0,1,1,0,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1}.
Cost: 89, *Evals:* 464, *Argmin:* 22, *Argmax:* 459, *Min val:* 44, *Max val:* 89, *Mean:* 68.4590517241, *Stdev:* 10.5445710397
Cost: 89, *Evals:* 464, *Argmin:* 22, *Argmax:* 459, *Min val:* 44, *Max val:* 89, *Mean:* 68.4590517241, *Stdev:* 10.5445710397

5.4 Bootstrapping Algorithm

DNA dataset: Results focused on the time it uses bootstrapping algorithm is directly related to the number of subsets selected; this is where the algorithm uses most of the time. Since the number of attributes is high, the number of subsets to generate affects the performance of this. The algorithm generates a ranking and chooses the percentage of attributes that are in better positions according to the ranking (Table 5).

Table 5. DNA dataset test

Selection	Features selection	Quant	Time
0%	90	1	144 seg.
5%	6, 16, 29, 74, 92, 115, 127, 142, 163	9	143 seg.
30%	1, 2, 7, 10, 15, 17, 20, 21, 30, 39, 40, 41, 42, 44, 47, 48, 52, 56, 61, 63, 64, 65, 66, 72, 73, 75, 84, 85, 95, 97, 98, 101, 108, 109, 110, 115, 116, 123, 125, 126, 128, 136, 139, 144, 146, 148, 149, 152, 155, 158, 165, 168, 176, 177	54	145 seg.
80%	1–9, 11–28, 30–42, 44–55, 59–62, 64, 66, 67–71, 73–76, 78–80, 82–94, 96, 97, 99–105, 107–119, 122, 125–131, 134, 137, 139–145, 147–151, 154–156, 158–160, 162–164, 166–177, 179, 180	144	145 seg.

CHESS Dataset: Feature selection with bootstrapping algorithm is very similar to the NURSERY dataset, in that most have a very similar ranking, because the entire space of possibilities is covered. The big difference between these two datasets is that the CHESS occupies a considerable time in the selection, since, first, it has more instances, second, each attribute has 8 possible values, and third the class has 18 possible values that increases the benefits of the naive bayes classifier (Table 6).

Table 6. Chess dataset test

Selection percentage	Features selection	Quantity	Time
50	2, 3, 5	3	13,7 seg.
80	1, 2, 3, 4	4	13,7 seg.
90	1, 2, 3, 4, 5	5	13,9 seg.

NURSEY Dataset: The test with a selection percentage of 80% eliminated the features 6 and 7, because the Naive bayes classifier provided these features with the lowest benefits, thus, when generating the ranking based on the weights granted according to this classifier, the dataset presents a deletion of the features mentioned above, these represent little relevant attributes, in the case of the *finance* feature, which contains the lowest number of classes in the set of attributes and next to *social* represent the last places in the ranking generated by the algorithm, which corresponds to the remaining 20% of the number of features of the dataset (Table 7).

Table 7. Nursey dataset test

Selection percentage	Features selection	Quantity	Time
30	1, 4	2	2.6 seg.
80	1, 2, 3, 4, 5, 8	6	3 seg.

6 Conclusions

Decision trees are used as an algorithm to feature selection are a good option. However, it must be taken into account that: the data set must be categorical, only applies for predictive problems, which limits the field of applications, and if the dataset is incomplete, the selection is not considered as good.

Feature selection based entropy measure for ranking features is applicable only to descriptive tasks, limiting the field of application just like decision trees. Its main weakness lies in the high complexity since it makes a comparison combining all the instances. As for the EDAs, these enjoy a good reputation within the feature selection algorithms; however, it has some weaknesses, such as: redundancy in generating the dependency trees, in free code applications has only been done with languages interpreters, and there is limited evidence of having used multivariate data.

The concept of randomness used by the Bootstrapping algorithm for the selection of features establishes the possibility that, due to the very concept of randomness, the algorithm presents unexpected results at some point, as opposed to its speed of execution. The ideal would be to use algorithms that indicate if the result generated by the algorithm is acceptable. It is emphasized that the Bootstrapping algorithm has a good behavior with large volumes of dataset.

References

1. Larose, D.: Data Mining: Methods and Models, pp. 1–3. Wiley-Interscience, New York (2006)
2. Pyle, D.: Data Preparation for Data Mining, p. 15. Morgan Kaufmann Publisher, San Francisco (1999)
3. Bradley, P., Mangasarian, O.: Feature selection via concave minimization and support vector machine. In: ICML, pp. 82–90 (1998). Journal Machine learning
4. Lei, Y., Huan, L.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. **5**, 1205–1224 (2004)
5. Guyon, I., Elissee, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
6. Dash, M., Liu, H.: Feature selection for classification. J. Intell. Data Anal. **1**(3), 131–156 (1996)
7. Liu, H., Lei, Y.: Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng. **17**, 491–502 (2005)
8. Kohavi, R., John, G.: Wrappers for feature subset selection. Artif. Intell. **97**(12), 273–324 (1997)
9. Jennifer, G.: Feature selection for unsupervised learning. J. Mach. Learn. **5**, 845 (2004)
10. Das, S.: Filters, wrappers and a boosting-based hybrid for feature selection. In: Proceedings of 18th International Conference on Machine Learning, pp. 74–81 (2001)
11. Cardie, C.: Using decision trees to improve case-based learning. In: Proceedings of 10th International Conference on Machine Learning, Utgo, pp. 25–32 (1993)
12. Mucciardi, A., Gose, E.: A comparison of seven techniques for choosing subsets of pattern recognition. IEEE Trans. Comput. **20**, 1023–1031 (1971)

13. Ruiz, R., Riquelme, J., Aguilar-Ruiz, J.: Projection-based measure for efficient feature selection. J. Intell. Fuzzy Syst. **12**, 175–183 (2003)
14. Pérez, I., Sánchez, R.: Adaptación del método de reducción no lineal LLE para la selección de atributos en WEKA. In: III Conferencia en Ciencias Informáticas, pp. 1–7 (2016)
15. Winston, P.: Inteligencia Artificial, pp. 455–460. Addison Wesley, Reading (1994)
16. Chourasia, S.: Survey paper on improved methods of ID3 decision tree classification. Int. J. Sci. Res. Publ. **3**, 1–4 (2013)
17. Rodríguez, J.: Fundamentos de minería de datos, pp. 63–64. Fondo de publicaciones de la Universidad Distrital Francisco José de Caldas (2010)
18. Changala, R., Annapurna, G., Yedukondalu, G., Raju, U.N.P.G.: Classification by decision tree induction algorithm to learn decision trees from the class labeled training tuples. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **2**(4), 427–434 (2012)
19. Michell, T.: Machine Learning, pp. 50–56. McGraw Hill, New York (1997)
20. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, pp. 331–336. McGraw Hill, New York (2012)
21. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, pp. 531–540. Prentice Hall, Englewood Cliffs (2012)
22. Kantardzic, M.: Data Mining: Concepts, Models, Methods and Algorithms, pp. 46–48. IEEE Press, Wiley-Interscience, Hoboken (2003)
23. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining, vol. 2, pp. 30–35. Kluwer Academic Publisher (2000)
24. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective, pp. 20–28. Kluwer Academic Publisher, Dordrecht (2000)
25. Larrañaga, P., Lozano, J.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, pp. 1–2. Kluwer Academic Publishers, Dordrecht (2002)
26. Pelikan, M., Sastry, K.: Initial-population bias in the univariate estimation of distribution algorithm. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, pp. 429–436 (2002)
27. Pérez, R., Hernández, A.: Un algoritmo de estimación de distribuciones para el problema de secuenciamiento en configuración jobshop, vol. 1, pp. 1–4. Mexico, Communic del CIMAT (2015)
28. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. binary parameters. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61723-X_982
29. Rodríguez, N.: Feature relevance estimation by evolving probabilistic dependency networks and weighted kernel machine, pp. 3–4. A thesis submitted to the District University Francisco José de Caldas in fulfillment of the requirements for the degree of Master of Science in Information and Communications (2013)
30. Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A.: Estimation of distribution algorithms: a new evolutionary computation approach for graph matching problems. In: Figueiredo, M., Zerubia, J., Jain, Anil K. (eds.) EMMCVPR 2001. LNCS, vol. 2134, pp. 454–469. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44745-8_30
31. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-34954-9
32. Rodríguez, N., Rojas-Galeano, S.: Discovering feature relevancy and dependency by kernel-guided probabilistic model-building evolution. BioData Mining **10**, 12 (2017)
33. Kantardzic, M.: Data Mining: Concepts, Models, Methods and Algorithms, Second edition, pp. 68–70. IEEE Press, Wiley-Interscience, Hoboken (2003)

34. Caruana, R.A., Freitag, D.: How useful is relevance? In: fall'94 AAAI Symposium on Relevance, New Orleans, Technical report (1994)
35. Aguilar, J., Díaz, N.: Selección de atributos relevantes basada en bootstrapping. España, Actas III Taller de Minería de Datos y Aprendizaje (TAMIDA 2005), pp. 21–30 (2005)
36. Bertsimas, D., Sturt, B.: Computation of exact bootstrap confidence intervals: complexity and deterministic algorithms. Optimization Online an e-print Site for the Optimization Community (2017)
37. Barrera, H., Correay, J., Rodríguez, J.: Prototipo de software para el preprocesamiento de datos - UDClear. In: IV Simposio de Sistemas de Información e Ingeniería de Software en la Sociedad del Conocimiento, Libro de Actas, vol. 1 (2006). ISBN 84–690-0258-9
38. Rodríguez, N., Rojas, S.: Goldenberry: EDA visual programming in orange. In: Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion (GECCO), pp. 1325–1332 (2013)
39. Wood, D.H., Chen, J., Antipov, E., Lemieux, B., Cedeño, W.: A design for DNA computation of the OneMax problem. Soft Comput. **5**(1), 19–24 (2001)