Bayesian Network-Based Multi-objective Estimation of Distribution Algorithm for Feature Selection Tailored to Regression Problems

José A. López[1], Felipe Morales-Osorio[2], Maximiliano Lara[3], Jonás Velasco[1,4], and Claudia N. Sánchez[3](✉)

[1] Centro de Investigación en Matemáticas (CIMAT), A.C., 20200 Aguascalientes, Mexico
{jose.portillo,jvelasco}@cimat.mx
[2] Massachusetts Institute of Technology, Cambridge, MA 02139, USA
fmorales@mit.edu
[3] Facultad de Ingeniería, Universidad Panamericana, 20296 Aguascalientes, Mexico
{0218259,cnsanchez}@up.edu.mx
[4] Consejo Nacional de Humanidades Ciencias y Tecnologías (CONAHCYT), 03940 Ciudad de México, Mexico

Abstract. Feature selection is an essential pre-processing step in Machine Learning for improving the performance of models, reducing the time of predictions, and, more importantly, identifying the most significant features. Sometimes, this identification can reduce the time and cost of obtaining feature values because it could imply buying fewer sensors or spending less human time. This paper proposes an Estimation of Distribution Algorithm (EDA) for feature selection tailored to regression problems with a multi-objective approach. The objective is to maximize the performance of learning models and minimize the number of selected features. We use a Bayesian Network (BN) as the EDA distribution probability model. The main contribution of this work is the process used to create this BN structure. It aims to capture the redundancy and relevance among features. Also, the BN is used to create the initial EDA population. We test and compare the performance of our proposal with other multi-objective algorithms: an EDA with a Bernoulli distribution probability model, NSGA II, and AGEMOEA, using different datasets. The experimental results show that the proposed algorithm found solutions with a considerably fewer number of features. Additionally, the proposed algorithm achieves comparable results on models' performance compared with the other algorithms. Our proposal generally expended less time and had fewer objective function evaluations.

Keywords: Feature selection · estimation distribution algorithms · bayesian network · multi-objective optimization · regression problems

1 Introduction

In Machine Learning, most real-world problems involve a large amount of data. Usually, this data has a high number of features, which makes the learning process difficult. However, not all features are essential since many of them are redundant or even irrelevant, which may reduce the performance of a learning algorithm. Feature Selection (FS) problems involve reducing the size of the original datasets by selecting a small subset of relevant features from the original dataset while maintaining model performance [1,23]. According to the mathematical definition presented on [1], an FS problem can be formulated as follows. Assume a dataset consists of a set of features $\mathcal{F}$ with exactly d number of features, $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots, \mathcal{F}_d\}$, the objective is to select the best subsets of features of size k from $\mathcal{F}$, where $k < d$. Feature selection becomes a difficult task as the number of features increases. For d features, there are 2^d subsets that can be selected to train a learning algorithm [6,9,23].

According to [8], there are three different approaches for solving the FS task: filter, wrapper, and embedded methods. Filter methods consist of finding the best subset of features according to the intrinsic characteristics of the data (e.g., correlation coefficient) that measure the relevance and redundancy of features. They are independent of the learning algorithms. Because of this, they are computationally faster than the other methods. Wrapper methods train a learning algorithm on a selected subset of features and use the algorithm's performance to evaluate the quality of the subset. They aim to find a subset of features that minimizes the learning algorithm's error. However, because they need to train on each subset of features, they are computationally more expensive than filter methods but generally offer better performance. Embedded methods incorporate feature selection into model training. For example, decision trees can be used in these models. After generating the prediction model, decision trees return a feature's importance, which can be used to create new solutions. As pointed out in [12], hybrid methods have been developed to exploit the benefits of different approaches (e.g., using a filter method to reduce computation time and a wrapper method to increase model performance).

Most of the documents tailored to feature selection have recently used a multi-objective approach [12]. Commonly, the first objective function is related to the performance of a learning model, and the second objective aims to reduce the number of features. These techniques return a set of feature subsets, and the end user can use the one better adapted to its application. Since FS is an optimization problem whose search space grows exponentially according to the number of features d, Evolutionary Algorithms (EAs) are techniques that can be used to solve this problem. Some of the EAs that have been applied for solving FS with a multi-objective approach are Genetic Algorithms [21], Particle Swarm Optimization (PSO) [22], and Differential Evolution (DE) [24]. The choice of the solution representation is highly related to the applied EA [23], but using continuous representations, such as the one used in PSO or DE, increases the search space. Genetic Algorithms and Estimation of Distribution Algorithms are techniques that can use binary representation. Specifically, NSGA

II (Nondominated Sorting Genetic Algorithm) [7] is a fast and powerful technique for solving multi-objective problems. It has been widely used for FS [10,19,21].

In this work, we explore the Estimation of Distribution Algorithms (EDAs) because they can combine filter metrics for measuring the redundancy and the relevance of features with the wrapper techniques evaluating the performance of learning models. Soliman and Rassem [20] proposed a filter technique that consisted of a quantum bio-inspired EDA for correlation-based feature selection to obtain optimal feature subsets. Maza and Touahria [16] proposed an EDA, a hybrid methodology, for FS in classification problems. Instead of randomly creating the initial population, they use the relevance between features and the class. In addition, they propose four probabilistic models for estimating the probability of each feature being selected. Those models use metrics of relevance and redundancy among features. However, they calculate the probability of features being selected separately. We propose using a Bayesian Network as the probabilistic model for sampling the selection values of features according to the selection values previously assigned to other features.

Some documents previously used Bayesian Networks for FS in EDAs. Their creation of the net is described as follows. Larrañaga *et al.* [11] used Estimation of Bayesian Networks Algorithm (EBNA) [14], which is a greedy search which starts with an arc-less structure and, at each step, adds the arc with the maximum improvement in the measure used. The algorithm stops when adding an arc would not increase the scoring measure. Castro and Von Zuben [3] begins with an initial network generated at random. Next, the probability distribution of each variable is estimated using the dataset, and the network score is computed. The search process generally proposes small changes to the structure to obtain a network with a higher score than the previous one. These small changes can be accomplished by adding or deleting an edge or reversing the edge direction. Every time a change is made, it is necessary to compute the probability distribution of the variables for the modified network. In contrast, our proposal creates the network structure as a tree, where each node represents an input feature and the arcs represent the redundancy among them. We use this graph structure because we are interested in establishing a unique dependence among the variables in order to create the simplest algorithm while keeping complexity as low as possible. The objective is to maximize a redundance metric among the connected features.

This paper proposes a Bayesian network-based multi-objective estimation of distribution algorithm for feature selection tailored to regression problems. The regression problems have numeric vectors from $\mathbb{R}^d$ as inputs, where d is the number of features, and numeric scalars from $\mathbb{R}$ as outputs. We focus on two criteria: maximizing the performance of the learning models and minimizing the number of selected features. The main contribution of our proposal is that the probabilistic model of the EDA is a Bayesian Network defined to capture the relevance and redundancy in the features. Mainly, the difference between our proposal and others using EDAs with Bayesian Networks is how the BN is created. In our case, it is a tree where the arcs represent the redundancy among features. In addition, the first generation is a filter method that creates solutions based on the BN structure, maximizing relevance and minimizing redundancy among features.

The rest of the paper is organized as follows. Section 2 presents the background concepts. Our proposal is described in the Methodology Sect. 3. The experiments and results are presented in Sect. 4. Finally, Sect. 5 concludes this document and describe the future work.

2 Background

In this section, we briefly review the fundamental concepts of this research. We started with Multi-objective Evolutionary Optimization, followed by Estimation of Distribution Algorithms (EDAs), and finalized with Bayesian Networks (BN).

2.1 Multi-objective Evolutionary Optimization

In multi-objective optimization [4], a variety of optimization problems are tackled, each involving one of the n distinct objective functions $f_1(s), \ldots, f_n(s)$. These functions operate on s, a vector of parameters from a specific domain. Consider $\mathcal{S}$ to be the potential solution space for such a multi-objective optimization problem. A solution set is classified as non-dominated (alternatively referred to as Pareto optimal) when there is no $t \in \mathcal{S}$ that, for any $s \in \mathcal{S}$, can satisfy the following:

– $\exists i$, where $i \in \{1, \ldots, n\}$, $f_i(t)$ enhances $f_i(s)$,
– $\forall j$, where $j \in \{1, \ldots, n\}$ and $j \neq i$, $f_j(s)$ cannot improve $f_j(t)$.

The main point is that one solution s is considered to dominate another t if s surpasses t on at least one objective and is not inferior on the remaining ones. We term s as non-dominated when no other solution outperforms it. The set of these non-dominated solutions within $\mathcal{S}$ is known as the Pareto front. The goals of multiobjective optimization is to find set of solutions as close as possible to Pareto-optimal front and to find a set of solutions as diverse as possible. When it comes to multi-objective optimization, multi-objective evolutionary algorithms are particularly beneficial as they concurrently pursue multiple best solutions. These algorithms can locate a collection of top solutions in the end population with just a single run, and once this set is ready, the most pleasing solution can be picked based on a preference criterion.

Feature selection consists of two main objectives: minimizing the cardinality of the subset of selected features and maximizing the model's performance. These objectives often conflict, giving space for multiobjective optimization techniques such as evolutionary multi-objective optimization. In these algorithms, more than one solution is returned, and each solution corresponds to a specific trade-off between the objectives. This way, of the reported solutions, some will represent a larger subset with better performance. In comparison, others will give a smaller subset but diminish the model's performance.

In this work, we use two algorithms: the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [7] and Adaptive Geometry Estimation based Multi-Objective Evolutionary Algorithms (AGEMOEA) [18]. The NSGA-II and AGEMOEA are examples of Evolutionary Algorithms designed for solving multi-objective optimization problems. The former finds a diverse set of solutions and covers near the true Pareto optimal set. The latter, AGEMOEA, modifies NSGA-II by replacing the fitness assigned to the solutions in each non-dominated front. In AGEMOEA, the crowding distance of NSGA-II is replaced by a survival score that combines both diversity and proximity of the solutions within the same non-dominated front.

2.2 Estimation of Distribution Algorithms (EDAs)

Introduced in [17], Estimation of Distribution Algorithms (EDAs) are based on Evolutionary Algorithms where a population of N individuals becomes better to their fitness value each iteration [14]. This algorithm replaces crossover and mutation operators of EAs by estimation of parameters of the M best individuals of the population. The probability distribution function is used to sample new N individuals who will become part of the next generation of the algorithm. This process is repeated until a stopping criterion is met. It could be the maximum number of iterations, convergence criterion based on stagnation, etc. The algorithm learns from the population and modifies the probability distribution in each generation. The overall fitness value of the population will be better in each iteration. It is important to see that exploitation and exploration in the search space are controlled by random sampling of new individuals.

Algorithm 1: The basic steps of an EDA.

```
Input  : population size $N$, selection size $M$, where $M < N$
Output: the best solution(s) found $S_{\text{Best}}$
1  $P_0 \leftarrow$ Generate initial population with $N$ random individuals
2  Evaluate each individual $s$ in $P_0$ using the objective function
3  $S_{\text{Best}} \leftarrow$ Get the best solution(s) from $P_0$
4  $t \leftarrow 0$
5  while termination criteria are not met do
6      $S_t \leftarrow$ Select $M$ individuals from $P_t$ according to a selection method
7      $\bar{p}_t \leftarrow$ Estimate the probability density of solutions in $S_t$
8      $P_{t+1} \leftarrow$ Sample $N$ individuals from $\bar{p}_t$
9      Evaluate $P_{t+1}$ using the objective function
10     Update $S_{\text{Best}}$ according to the solutions in $P_{t+1}$
11     $t \leftarrow t + 1$
12 end
13 return $S_{\text{Best}}$
```

Algorithm 1 shows the basic pseudo-code of a typical EDA. EDAs utilize several parameters, including the population size N, the number of generations (or iterations) G, and the number of individuals selected, M for estimate the probability density of solutions, $\bar{p}_t$. The set of selected solutions S_t serves as a training dataset to estimate the probabilistic model and leads the search towards regions with better fitness. The set of new solutions P_{t+1} is generated using the probabilities encoded in the probabilistic model in accordance with the statistics collected from the solutions in S_t.

Early EDAs were developed for discrete domains, as it is common in evolutionary algorithms to represent solutions with binary representations. Variations of EDAs are widely used for combinatorial optimization problems where more sophisticated distributions are used for sampling new individuals. For solving optimization problems in continuous domains, variations of the original EDAs approach also exist.

EDA algorithms are commonly grouped according to the degree of interaction among variables into univariate, bivariate, and multivariate EDAs. Univariate EDAs, such as Univariate Marginal Distribution Algorithm (UMDA), assume that all variables are independent and factorize the joint probability of the selected solutions as a product of univariate marginal probabilities. Multivariate EDAs do not necessarily limit the degree of interactions among variables and can be modelled with unrestricted Bayesian Networks. The choice of probabilistic model can have a major influence on the performance and efficiency of EDAs.

2.3 Bayesian Networks

A Bayesian Network (BN) [13] is a probabilistic graphical model which provides a robust general approach especially suited to modeling complex non-deterministic systems. A BN models the causal relationships between the features of a model. It consists of a Directed Acyclic Graph (DAG) $\mathcal{G}$ [13], and a set of parameters Θ, defining the strength and the shape of the relationships between features. To use a BN, one must define the graph $\mathcal{G}$ and then calculate its parameters Θ. Defining the graph $\mathcal{G}$ is a task that can be done by learning through data or by consulting human experts in a specific field [13].

$\mathcal{G}$ consists of a set of vertices $\mathcal{V}$ and a set of directed edges $\mathcal{E}$. The vertices in $\mathcal{V}$ represent the features whose relationship is modeled by the BN, and the edges in $\mathcal{E}$ represent the relationships between the features. A directed edge from V_i to V_j where $V_i, V_j \in \mathcal{V}$ is symbolically represented by the tuple $(V_i, V_j) \in \mathcal{E}$ or graphically represented as $V_i \rightarrow V_j$. The directed edge $(V_i, V_j) \in \mathcal{E}$ indicates that V_i is the parent of V_j and that V_j is the child of V_i. A BN models a parent-child relationship as the parent variable causing the child variable. Therefore, the directed edge (V_i, V_j) in a BN means that the parent variable V_i causes the child variable V_j. Additionally, every directed edge $(V_i, V_j) \in \mathcal{E}$ has a parameter $\theta_{i,j} \in \Theta$. The parameter $\theta_{i,j}$ associated with this edge is a matrix modeling $\Pr(V_j \mid V_i)$ (the conditional probability of V_j given V_i). If V_i, V_j can take on $|V_i|$, $|V_j|$ different values respectively, then this matrix $\theta_{i,j}$ will have $|V_i| \times |V_j|$ different entries.

Finally, a BN assumes that the joint probability distribution of the variables $P(V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n)$, defined in Eq. (1), can be decomposed as the product of the conditional distribution of the variables given the value of their parents $P\left(V_i = v_i \mid \mathbf{Pa}(V_i)\right)$.

$$P(V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n) = \prod_{i=1}^{n} P\left(V_i = v_i \mid \mathbf{Pa}(V_i)\right) \tag{1}$$

3 Methodology

Our proposal is an Estimation of Distribution Algorithm for feature selection tailored to regression problems with a multi-objective approach. Our regression problems have numeric vectors from $\mathbb{R}^d$ as inputs, where d is the number of features, and numeric scalars from $\mathbb{R}$ as outputs. We focus on two criteria: maximizing the performance of the learning models and minimizing the number of selected features. Our main contribution is using a Bayesian Network (BN) as the sample distribution model. The BN aims to capture the redundancy among features. Additionally, the first generation can be seen as a filter method that creates solutions that maximize relevance and minimize redundancy among features.

We follow the Algorithm 1, explained in the previous section. In this section, we describe the details of the implementation. First, in Subsect. 3.1, we explain the individuals' representation and evaluation. The selection of the individuals used for calculating the distribution model, in this case, the BN, is described in Subsect. 3.2. The creation of the structure of the BN is described in Subsect. 3.3. Subsection 3.4 describes the initialization of the population. In this case, instead of being totally random, it is initialized using the BN. The BN's probabilities calculation is explained in Subsect. 3.5. Finally, Subsect. 3.6 describes how the individuals of the new population are sampled using the BN.

3.1 Representing and Evaluating Individuals

Our proposal aim to solve the feature selection in regression problems. Regression can be mathematically expressed as follows. We are given a set $\mathcal{D} = \{(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n)\}$ of n input-output data pairs with d features in the inputs where $\vec{x_i} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for all $i \in \{1 \ldots n\}$. The regression model $\mathcal{M}$ aims to minimize $\sum_{(\overrightarrow{x}, y) \in \mathcal{D}} E(\mathcal{M}(\overrightarrow{x}), y)$, where E is a function that measures the difference or error between $\mathcal{M}(\overrightarrow{x})$ and y. The ideal scenario would be $\mathcal{M}(\overrightarrow{x}) = y \ \forall (\overrightarrow{x}, y) \in \mathcal{D}$. The model predictions $\mathcal{M}(\overrightarrow{x})$ also are defined as $\widehat{y}$. Feature selection consists of selecting the best subsets of features of size k, where $k < d$. An individual representing a solution for the feature selection problem is a binary vector of size d, where the i-th element corresponds to the i-th feature in the dataset. A value of 1 means that the i-th feature is selected for fitting the model, while 0 indicates that the i-th feature is left out.

We define two criteria as objective functions. The first is related to the regression models' performance, and the other to the number of selected features. Regression involves finding a mathematical model that relates input features to an output feature to reduce an error. The determination of coefficient R^2 (Eq. 2) can be used for measuring the performance of a regression model. If its value is close to 1 indicates a good performance of the model, or in other words, that the regression model explains a large portion of the variability in the response. A number near 0 indicates that the regression does not explain much of the variability in the response, and it is almost equal to random guesses according to the output feature distribution. Finally, negative values indicate that the regression model gives worse results than random guesses. In our proposal, for having minimization objective functions, we define $\overline{R^2} \equiv 1 - R^2$. By doing this, we focus on minimizing the performance of the regression model trained with a subset of features.

$$R^2(y, \widehat{y}) = 1 - \frac{\sum_i (y_i - \widehat{y}_i)^2}{\sum_i (y_i - \overline{y}_i)^2} \tag{2}$$

The second objective function is designed to minimize the number of selected features. It is defined as $|\mathcal{F}|$ and corresponds to the number of selected features in the subset divided by the total number of features in the data set. A number near to 0 indicates that the subset has a few features selected. On the other hand, a number close to 1 indicates that most of the features have been selected.

3.2 Selection of Individuals for Estimating the Parameters of the Bayesian Network

Step 6 of Algorithm 1 consists of selecting the best M individuals from the population to calculate the parameters of the distribution model, in this case, the probabilities of the Bayesian Network. For sorting the individuals, we use the fast non-dominated sorting algorithm presented in [7] and the non-domination rank that corresponds to the non-dominated front an individual belongs to.

3.3 Creating Bayesian Network Graph

A BN must be a DAG by definition. However, our BN is more restrictive because it is a tree. Our Bayesian Network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a tree whose vertices $V_i \in \mathcal{V}$ represent the feature at index i and whose edges $(V_i, V_j) \in \mathcal{E}$ represent the causal relationship between feature i and feature j. The strength of the causal relationship between V_i, V_j is approximated by $|C_{i,j}|$, corresponding to the absolute value of the Pearson correlation coefficient between the two features i, j in a dataset. Our objective is to construct a tree capturing the redundancy between variables. In other words, we want to construct a tree with an edge set $\mathcal{E}$ that maximizes $R(\mathcal{E})$ as defined in Eq. (3).

$$R(\mathcal{E}) = \sum_{(V_i, V_j) \in \mathcal{E}} |C_{i,j}| \tag{3}$$

Maximizing $R(\mathcal{E})$ is equivalent to finding the maximum spanning tree of $\mathcal{H}$ where $\mathcal{H}$ is a fully connected graph containing the vertices in our problem and the weighted edges (V_i, V_j) are the values of $|C_{i,j}|$ between features. We use a modified version of Prim's Minimum Spanning Tree [5] algorithm to find the maximum spanning tree. In particular, we set the root vertex to the feature that has maximum relevance with the output feature in a dataset. We approximate relevance with $|C_i|$, the absolute value of the Pearson correlation coefficient between the feature i and the output feature in the dataset.

Figure 1 illustrates the Bayesian Network produced by our algorithm using the Concrete Compressive Strength dataset (See Table 5). The root vertex corresponds to the feature Fly Ash because this is the one with the highest relevance to the output feature. In other words, Fly Ash had the highest absolute value of the Pearson correlation coefficient with the output variable. Next, an edge from the root to the vertex Blast Furnace Slag is drawn because it has the greatest redundancy with the root. This means it has the highest absolute value of the Pearson correlation coefficient with the root. Then, each successive edge added to the tree must not produce a cycle while also being the edge with the highest possible redundancy.

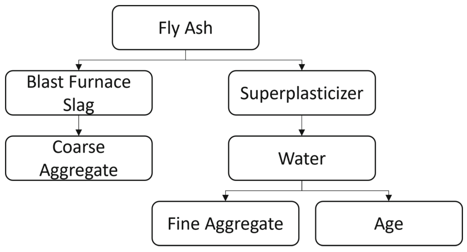

Fig. 1. Bayesian Network produced from concrete data.

3.4 Creating Initial Population

To create the initial population of individuals for the EDA, we use the parent-child structure of the BN and using the redundance and the relevance of features calculated with the Pearson correlation coefficient. Our proposal generates half of the individuals of the initial population via relevance and half via redundancy.

To generate individuals via redundancy, we want to avoid selecting two features that have a high redundancy with each other. To achieve this, we want every child variable to be more likely be the binary opposite of its parent variable if the parent-child redundancy metric is high. For example, if a parent $V_i = 1$, then the child $V_j = 0$ should have a high probability of occurring if the redundancy $|C_{i,j}|$ is high. Similarly, if a parent $V_i = 0$, then the child $V_j = 1$ should occur with high probability when $|C_{i,j}|$ is high. We model this expected probability distribution using a Bernoulli trial. `BERNOULLI`(p) runs a Bernoulli trial to generate a binary value. The single parameter p specifies the probability of a success. A success generates a 1 while a failure generates a 0. The desired outcome v_j for a child V_j given value v_i of its parent V_i is the result of the Bernoulli trial with the parameter $p = |v_i - |C_{i,j}||$ as $v_j = \texttt{BERNOULLI}\,(|v_i - |C_{i,j}||)$.

To generate individuals via relevance, we want the probability of feature i being selected to be proportional to its relevance $|C_i|$ with the output. If a feature has high relevance, then it should have a high probability of being selected. Again, we model this probability distribution using a Bernoulli trial. The binary value v_i of the variable V_i should be determined as $v_i = \texttt{BERNOULLI}(|C_i|)$.

3.5 Calculating Bayesian Network Parameters

The Bayesian Network parameters are related to probability distributions; each vertex has its values. The probabilities are calculated based on the best individuals selected for making the distribution model in each algorithm iteration. First, we estimate $P(V_r)$, which corresponds to the probability distribution of V_r, the root variable of the tree. Table 1 shows the equations for calculating the $P(V_r)$ values, where F represents the occurrence frequency of the feature values in the selected individuals. The rest of the vertices on the BN corresponds to conditional probabilities $P(V_j \mid V_i)$, where vertex V_i can be seen as the parent of vertex V_j. Table 2 shows the equations for calculating the $P(V_j \mid V_i)$ values.

Table 1. Probability distribution of the root vertex $P(V_r)$

V_r	$P(V_r)$
0	$F(V_r = 0)/(F(V_r = 0) + F(V_r = 1))$
1	$F(V_r = 1)/(F(V_r = 0) + F(V_r = 1))$

Table 2. Probability distribution $P(V_j \mid V_i)$

V_i	V_j	$P(V_j \mid V_i)$
0	0	$F(V_i = 0, V_j = 0)/F(V_i = 0)$
0	1	$F(V_i = 0, V_j = 1)/F(V_i = 0)$
1	0	$F(V_i = 1, V_j = 0)/F(V_i = 1)$
1	1	$F(V_i = 1, V_j = 1)/F(V_i = 1)$

3.6 Sampling from Bayesian Network

Once the structure of the Bayesian Network and its parameters are calculated, it can be used to generate a new population of individuals. To start, we generate the binary value v_r associated with the root V_r using the $\texttt{BERNOULLI}$ function based on $P(V_r)$. Once we have generated the root's binary value, we generate the binary values associated with the remaining vertices from top to bottom. This is done so that the values of parent vertices are decided before the values of child vertices. To generate the binary value v_j associated with vertex V_j we must have a value v_i associated with the parent vertex V_i. We generate the binary value associated with V_j using the `BERNOULLI` function based on $P(V_j \mid V_i = v_i)$.

4 Experiments and Results

In this section, we present our proposal's experiments and results. We used Linear Regression as the regression model because it is simple and fast. To analyze the performance of our proposal, we compare it with Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Adaptive Geometry Estimation based Multi-Objective Evolutionary Algorithms (AGEMOEA) previously mentioned in Sect. 2. We used the implementation of those algorithms provided by the library Pymoo [2]. The hyper-parameters of those algorithms, defined by default in Pymoo, are presented in Table 3.

In addition, we compare our results with an Estimation of Distribution Algorithm that uses a Bernoulli model as the probability distribution, defined as EDA Bernoulli. This implementation initializes the population P of individuals using Bernoulli trials. As mentioned in the previous section, an individual is a binary vector whose entry v_i associated with feature i is decided via a Bernoulli trial. In the initial population, the entry v_i is given by the Bernoulli trial $v_i = \texttt{BERNOULLI}(0.5)$. In the following generations of the algorithm, the population is sampled using the Bernoulli trial $v_i = \texttt{BERNOULLI}(\texttt{PROPORTION}(v_i))$ where $\texttt{PROPORTION}(v_i)$ is the proportion of times $v_i = 1$ in the selected population. The hyper-parameters of the EDA implementations are presented in Table 4.

Table 3. Hyper-parameters of NSGA II and AGEMOEA

Algorithm	Population size N	Max Iterations	Sampling	Crossover	Mutation
NSGA II	50	10	Binary random sampling	Two point crossover	Bitflip mutation
AGEMOEA	50	10	Binary random sampling	Two point crossover	Bitflip mutation

Our implementation was developed in the Python programming language, and we use the libraries Scikit-learn and numpy. Our computational experiments were run with an Intel Core i7-5500U Dual-Core Processor @ 2.40 GHz running the Windows 10 64-bit operating system based on an ×64 processor with 8.00 GB of RAM.

Table 4. Hyper-parameters of EDA Bernoulli and EDA Bayesian Network

Algorithm	Population size N	Max iterations	Individuals selected M	Probability distribution
EDA Bernoulli	50	5	25	Bernoulli
EDA Bayesian Network	50	5	25	Bayesian Network

4.1 Datasets

Five datasets are adopted from the UCI repository [15] (See Table 5). The datasets are of different dimensions, varying from 8 to 100, and the number of instances is 395 to 515345. Since we are solving a regression problem, we need the input features, and the output feature, to be of numerical type. For each dataset, the last column corresponds to the target feature. We had to preprocess this data because some of the features were categorical, or it contained missing values. The preprocess methodology is described as follows. First, columns with a high ratio of missing values were deleted. For example, the Communities and Crime dataset had many columns with around 84% missing values. Rows containing null values were removed from the datasets. Categorical variables with only two possible values were transformed into numerical variables by changing the value of one class to 0 and the other to 1. Those variables with three or more possible values were transformed into numerical variables by applying one-hot encoding. In the case the categorical variables had more than 30 categories, they were removed from the dataset. In the Forest Fires dataset, the values of the feature month were transformed from 'jan', 'feb', 'mar', ..., 'dec' to 1, 2, 3, ...,12. And for the feature day, values were transformed from 'mon', 'tue', 'wed', ..., 'sun' to 1, 2, 3, ...,7. We randomly divided the datasets into training (70%) and testing (30%) sets to validate the results.

Table 5. Datasets

Name	Before processing		After processing	
	# instances	# features	# instances	# features
Concrete Compressive Strength	1030	8	1030	8
Forest Fires	517	12	517	12
Student Performance Math	395	32	395	45
YearPredictionMSD	515345	90	515345	90
Communities and crime	1993	127	1993	100

4.2 Comparison of Our Proposal Against Other Techniques

To the best of our knowledge, no study has been conducted on feature selection for the datasets described in Table 5. Most of the documents related to feature selection are related to classification. However, we compare the results of our proposal, EDA Bayesian Network, with other multi-objective evolutionary algorithms: EDA Bernoulli, NSGA-II, and AGEMOEA. Each experiment was executed 100 times, except the experiment of the YearPredictionMSD dataset, which was executed only ten times for the expensive time required.

Figure 2, 3, 4, 5 and 6 show the non-dominated solutions found by the different algorithms. They contain the best solutions for different executions. In Fig. 2 and Fig. 3 can be observed that all the algorithms obtain similar results in the two smallest datasets, Concrete Compressive Strength and Forest Fires. On the medium dataset, Student Performance Math, EDA Bayesian Network found solutions with a less number of features (see Fig. 4). And in the two biggest datasets, YearPredictionMSD and Communities and crime, AGMOEA got better results in the regression model performance, but our proposal got considerably better results in the number of selected features (see Fig. 5 and Fig. 6).

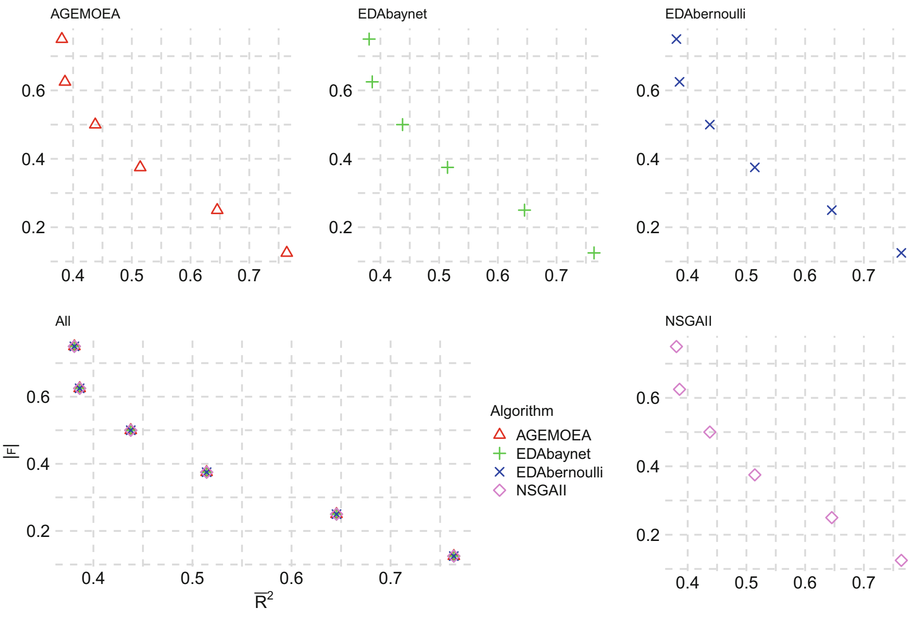

Fig. 2. Non-dominated solutions for the Concrete Compressive Strength dataset.

We also compare the algorithms based on the time (in seconds) and the number of evaluations of the regression model. For optimizing the execution time, we store the model's evaluation of different solutions aiming to evaluate only once time each different subset of features. Table 6 shows this comparison. It can be observed that in most cases, our implementation, EDA Bayesian Network, presented the best performance having less number of evaluations and the shortest time. In the case of the Communities Crime dataset, NSGA II was better in the

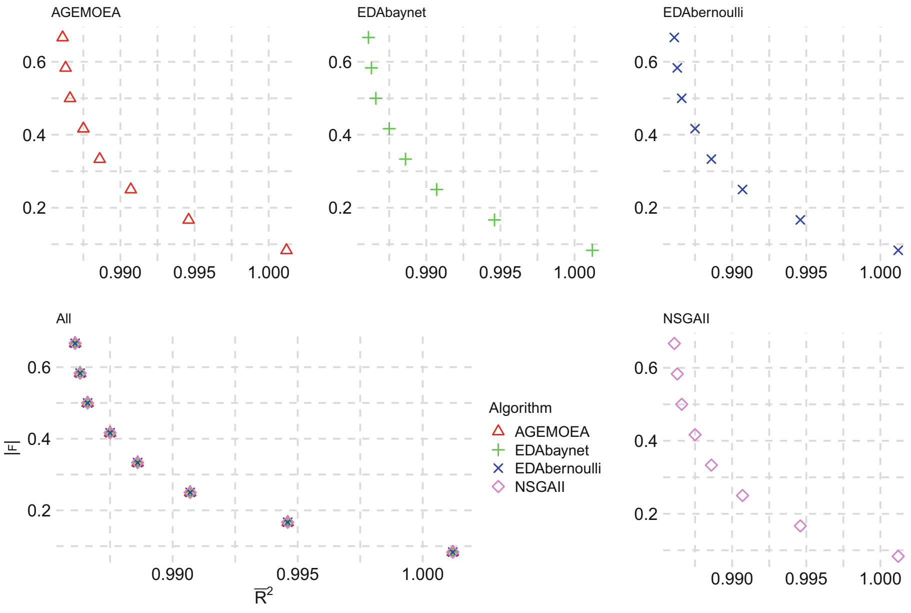

Fig. 3. Non-dominated solutions for the Forest Fires dataset.

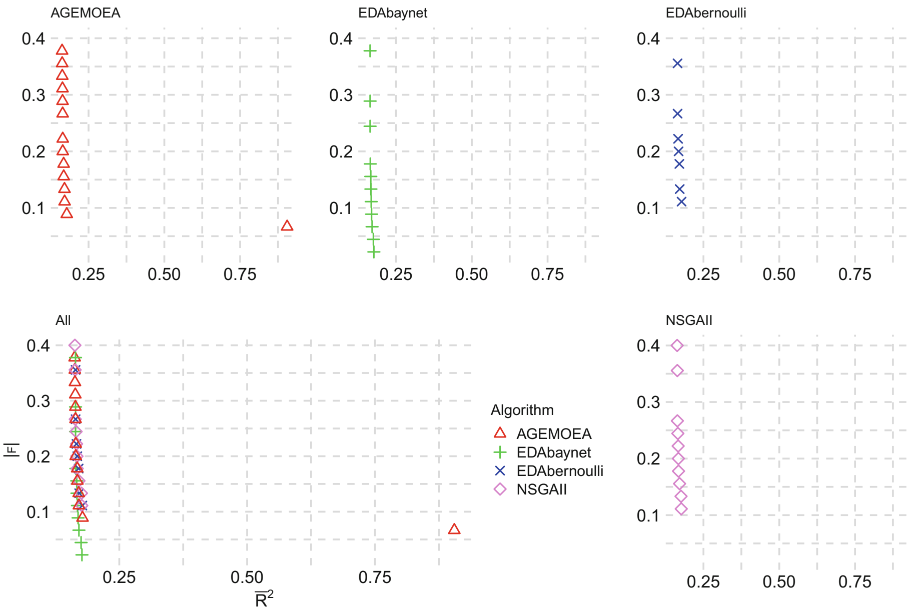

Fig. 4. Non-dominated solutions for the Student Performance Math dataset.

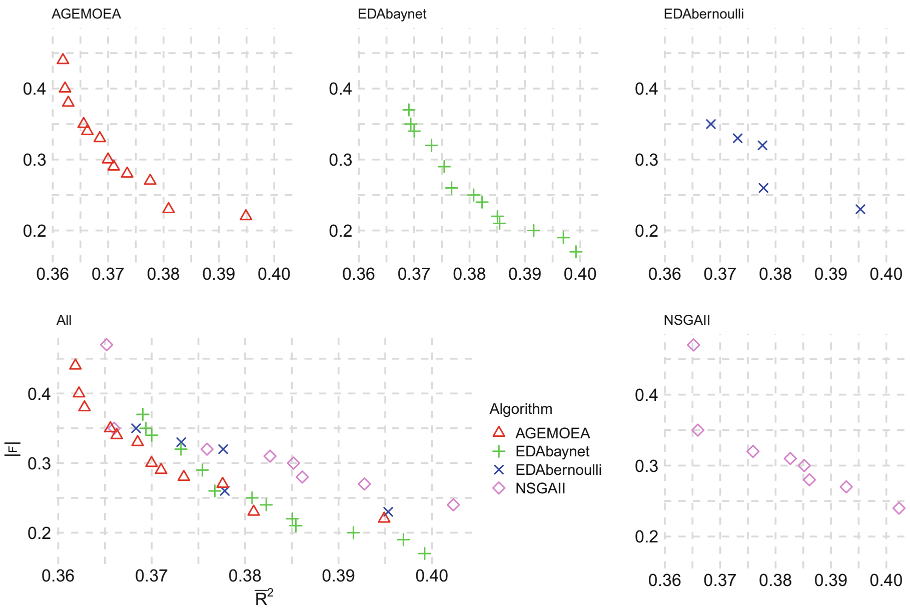

Fig. 5. Non-dominated solutions for the Communities and crime dataset.

number of evaluations, and EDA Bernoulli was better in time. In the case of the Student Performance dataset, EDA Bernoulli was better in time. The number of evaluations can be seen as how much the search space is explored because it is related to the different solutions found. In some cases, when the objective function is expensive, we wanted good results with a few evaluations. In this experiment, our proposal proportionate good results with the smallest number of evaluations. It indicates that we can improve the exploration in future work and maybe get better results.

Table 6. Comparison of time and number of evaluations between algorithms for all datasets. Bold numbers correspond to the smallest values in time or number of evaluations.

Algorithm	Concrete Compressive		Forest Fires		Student Performance		Year Prediction MSD		Communities Crime	
	Time	Eval	Time	Eval	Time	Eval	Time	Eval	Time	Eval
AGEMOEA	0.39	157.96	0.71	330.55	0.97	463.41	384.15	439.37	4.67	424.49
NSGA-II	0.47	172.35	0.73	366.89	0.91	431.24	211.57	312.50	1.52	**298.90**
EDA Bernoulli	0.28	122.79	0.42	241.60	**0.53**	299.99	211.28	300.00	**1.36**	300.00
EDA BayNet	**0.25**	**88.10**	**0.32**	**182.49**	0.57	**295.64**	**132.45**	**294.36**	1.50	299.97

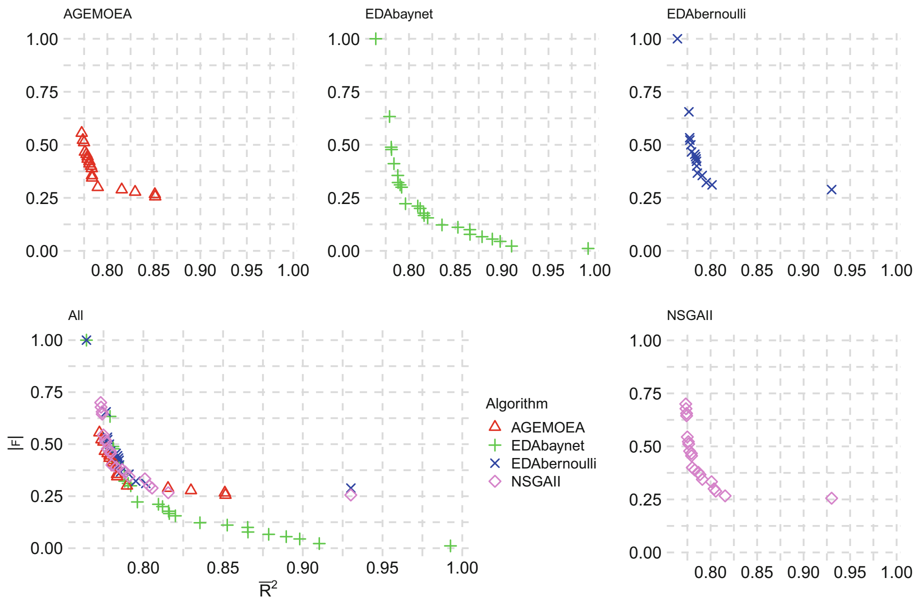

Fig. 6. Non-dominated solutions for the Year Predictions MSD dataset.

5 Conclusion

This paper proposes an Estimation of Distribution Algorithm for feature selection tailored to regression problems with a multi-objective approach. The main objective was maximizing the learning models' performance, calculated as the determination coefficient R^2, and minimizing the number of selected features. Our proposal used a Bayesian Network (BN) as the distribution model. The BN aims to capture the redundancy among features. The generation of the initial population can be seen as a filter method that randomly creates solutions maximizing the relevance and minimizing the redundancy among features. The relevance and the redundancy were measured using the Pearson correlation coefficient.

We compared our proposal with other multi-objective algorithms such as EDA Bernoulli, NSGA II, and AGEMOEA, and we used five different datasets. According to the performance of the regression and the number of features of the non-dominated solutions found, all the algorithms obtain similar results in the two smallest datasets. On the medium dataset, our proposal found solutions with a less number of features. Finally, in the two biggest datasets, AGMOEA got better results in the regression model performance, but our proposal got considerably better results in the number of selected features. However, our proposal generally expended less time and evaluated fewer times the objective function. The experimental results indicate that it could be improved for exploring more the search space.

In future work, we expect to improve the exploration of our proposed algorithm. It obtains good results in the number of features but can improve the model's performance. We plan to extend the experiments by trying different regression models, and we can optimize the hyper-parameters values of the evolutionary algorithms.

References

1. Agrawal, P., Abutarboush, H.F., Ganesh, T., Mohamed, A.W.: Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019). IEEE Access **9**, 26766–26791 (2021). https://doi.org/10.1109/ACCESS.2021.3056407
2. Blank, J., Deb, K.: Pymoo: multi-objective optimization in Python. IEEE Access **8**, 89497–89509 (2020). https://doi.org/10.1109/ACCESS.2020.2990567
3. Castro, P.A., Von Zuben, F.J.: Multi-objective feature selection using a Bayesian artificial immune system. Int. J. Intell. Comput. Cybern. **3**(2), 235–256 (2010). https://doi.org/10.1108/17563781011049188
4. Collette, Y., Siarry, P.: Multiobjective Optimization. Principles and Case Studies. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-662-08883-8
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press (2002)
6. Dash, M., Liu, H.: Feature selection for classification. Intell. Data Anal. **1**(1–4), 131–156 (1997). https://doi.org/10.1016/S1088-467X(97)00008-5. http://linkinghub.elsevier.com/retrieve/pii/S1088467X97000085
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Technical report 2 (2002)
8. Dhal, P., Azad, C.: A comprehensive survey on feature selection in the various fields of machine learning. Appl. Intell. **52**(4), 4543–4581 (2022). https://doi.org/10.1007/s10489-021-02550-9
9. Guyon, I., De, A.M.: An introduction to variable and feature selection André Elisseeff. Technical report (2003)
10. Hamdani, T.M., Won, J.M., Alimi, A.M., Karray, F.: LNCS 4431 - multi-objective feature selection with NSGA II. Technical report (2007)
11. Inza, I., Larrañaga, P., Etxeberria, R., Sierra, B.: Feature subset selection by Bayesian network-based optimization. Technical report (2000)
12. Jiao, R., Nguyen, B.H., Xue, B., Zhang, M.: A survey on evolutionary multiobjective feature selection in classification: approaches, applications, and challenges. IEEE Trans. Evol. Comput. (2023). https://doi.org/10.1109/TEVC.2023.3292527. https://ieeexplore.ieee.org/document/10173647/
13. Kitson, N.K., Constantinou, A.C., Guo, Z., Liu, Y., Chobtham, K.: A survey of Bayesian Network structure learning. Artif. Intell. Rev. **56**, 8721–8814 (2023). https://doi.org/10.1007/s10462-022-10351-w
14. Larragaña, P., Lozano, J.: Genetic algorithms and evolutionary computation. In: OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems (2002)
15. Markelle, K., Rachel, L., Kolby, N.: The UCI Machine Learning Repository. https://archive.ics.uci.edu
16. Maza, S., Touahria, M.: Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms. Appl. Intell. **49**(12), 4237–4257 (2019). https://doi.org/10.1007/s10489-019-01503-7

17. Mühlenbein, H.: The equation for response to selection and its use for prediction. Evol. Comput. **5**(3), 303–346 (1997). https://doi.org/10.1162/EVCO.1997.5.3.303. https://pubmed.ncbi.nlm.nih.gov/10021762/
18. Panichella, A.: An adaptive evolutionary algorithm based on non-Euclidean geometry for many-objective optimization. In: Proceedings of the 2019 Genetic and Evolutionary Computation Conference, GECCO 2019, July 2019, pp. 595–603. Association for Computing Machinery, Inc. (2019). https://doi.org/10.1145/3321707.3321839
19. Rehman, A.U., Nadeem, A., Malik, M.Z.: Fair feature subset selection using multiobjective genetic algorithm. In: Proceedings of the 2022 Genetic and Evolutionary Computation Conference, GECCO 2022 Companion, July 2022, pp. 360–363. Association for Computing Machinery, Inc. (2022). https://doi.org/10.1145/3520304.3529061
20. Soliman, O.S., Rassem, A.: Correlation based feature selection using quantum bio inspired estimation of distribution algorithm. Technical report (2012)
21. Spolaôr, N., Lorena, A.C., Lee, H.D.: Multi-objective genetic algorithm evaluation in feature selection. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 462–476. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19893-9_32
22. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans. Cybern. **43**(6), 1656–1671 (2013). https://doi.org/10.1109/TSMCB.2012.2227469
23. Xue, B., Zhang, M., Browne, W.N., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Trans. Evol. Comput. **20**(4), 606–626 (2016). https://doi.org/10.1109/TEVC.2015.2504420
24. Zhang, Y., Gong, D., Gao, X., Tian, T., Sun, X.: Binary differential evolution with self-learning for multi-objective feature selection. Inf. Sci. **507**, 67–85 (2020). https://doi.org/10.1016/J.INS.2019.08.040