Chapter 3

# Differential Evolution Algorithm in the Construction of Interpretable Classification Models

Rafael Rivera-Lopez and Juana Canul-Reich

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75694

**Abstract**

In this chapter, the application of a differential evolution-based approach to induce oblique decision trees (DTs) is described. This type of decision trees uses a linear combination of attributes to build oblique hyperplanes dividing the instance space. Oblique decision trees are more compact and accurate than the traditional univariate decision trees. On the other hand, as differential evolution (DE) is an efficient evolutionary algorithm (EA) designed to solve optimization problems with real-valued parameters, and since finding an optimal hyperplane is a hard computing task, this metaheuristic (MH) is chosen to conduct an intelligent search of a near-optimal solution. Two methods are described in this chapter: one implementing a recursive partitioning strategy to find the most suitable oblique hyperplane of each internal node of a decision tree, and the other conducting a global search of a near-optimal oblique decision tree. A statistical analysis of the experimental results suggests that these methods show better performance as decision tree induction procedures in comparison with other supervised learning approaches.

**Keywords:** machine learning, classification, evolutionary algorithms

## 1. Introduction

Knowledge discovery refers to the process of nontrivial extraction of potentially useful and previously unknown information from a dataset [1]. Within the stages of this process, data mining stands out since it allows analyzing the data and producing models for their representation. In particular, machine learning provides data mining with useful procedures to build these models, since many of the techniques aimed at information discovery are based on inductive learning. Decision trees (DTs), artificial neural networks (ANN), and support vector machines (SVMs), as well as clustering methods, have been widely used to build predictive models. The ability to track and evaluate every step in the information extraction process is one of the most crucial factors for relying on the models gained from data mining methods [2]. In particular, DTs are classification models characterized by their high levels of comprehensibility and robustness. Knowledge learned via a DT is understandable due to its graphical representation [3], and also DTs can handle noise or data with missing values and make correct predictions [4].

On the other hand, soft-computing-based approaches have been widely used to solve complex problems in almost all areas of science and technology. These approaches try to imitate the process of human reasoning when solving a problem with the objective of obtaining acceptable results in a reasonable time. For the case of data mining, soft computing techniques such as ANN, metaheuristics (MHs), fuzzy logic methods, and other approaches have been used as tools to solve the data mining challenges. In particular, an MH is a general algorithmic template based on intelligent processes and behaviors observed in both nature and other disciplines [5]. Evolutionary algorithms (EAs) are one type of MH that have been successfully applied for providing near-optimal solutions for many computationally complex problems in almost all areas of science and technology. The effectiveness of EAs is due to two factors: (1) they combine a clever exploration of the search space to identify promising areas and (2) they perform an efficient exploitation of these areas aiming to improve the known solution or solutions. EAs are inspired by evolutionary theories that synthesize the Darwinian evolution through natural selection with the Mendelian genetic inheritance. In particular, differential evolution (DE) algorithm is an EA designed for solving optimization problems with variables in continuous domains that, instead of implementing traditional crossover and mutation operators, it applies a linear combination of several randomly selected candidate solutions to produce a new solution [6].

MHs have been previously applied to build DTs, and there exist several surveys that describe their implementation [7–11]. Some approaches apply a recursive partitioning strategy in which an MH finds a near-optimal test condition for each internal node of a DT; however, the approach most commonly used is to perform a global search in the solution space with the aim of finding near-optimal DTs. Since DE is one of the most powerful EA to solve real-valued optimization problems, and the task of finding a near-optimal oblique hyperplane with real-valued coefficients is an optimization problem in a continuous space, in this chapter, two DE-based methods to induce oblique DTs are described: one implementing a recursive partitioning strategy to find the most suitable oblique hyperplane of each internal node of a decision tree, and the other conducting a global search of a near-optimal oblique decision tree. A statistical analysis of the experimental results suggests that these methods show better performance as decision tree induction procedures in comparison with other supervised learning approaches.

The rest of this chapter is organized as follows: Section 2 provides a set of basic definitions about DTs and the DE algorithm. The induction of oblique DTs by means of MH-based approaches is described in Section 3. The constituent elements of the DE-based methods described in this chapter is discussed in Section 4, and the experimental results are discussed in Section 5. Finally, Section 6 describes the conclusion and the future work.

## 2. Background

Machine learning methods are an essential tool in emerging disciplines such as data science [12] and business intelligence [13] since they provide efficient predictive models constructed from the data previously collected. DTs, ANN, and SVMs, as well as clustering methods, have been widely used to build these models. A DT is a hierarchical model using an ordered sequence of decisions to predict the class membership of new unclassified instances. An ANN consists of many nonlinear elements connected by links associated with weighted variables operating in parallel [14], in which learning is performed iteratively as the network processes the training instances, trying to simulate the way a human being learns from previous experiences. Finally, one SVM finds the hyperplane that best separates the training instances into two different classes using a set of functions called kernels. The optimal hyperplane is described with a combination of entry points known as support vectors [15].

A DT is an acyclic connected graph with a single root node used as one classification model induced through a set of training instances. A DT contains zero or more internal nodes and one or more leaf nodes [16]. Each internal node evaluates a test condition consisting of a combination of one or more attributes of the dataset, and each leaf node has a class label. The arcs joining an internal node with their successor nodes are labeled with the possible outcomes of its test condition. Each DT branch represents a sequence of decisions made by the model to determine the class membership of a new unclassified instance. The DT induction (DTI) process commonly implements a recursive partition strategy. In each stage of this process, the most appropriate test condition to split the training set is selected according to some partition criterion. As a result of evaluating the training instances with this test condition, two or more instances subsets are created which are assigned to the successor nodes of the current internal node. This process is recursively applied until a stop criterion is reached. If the number of attributes used in the test conditions of the tree internal nodes is regarded, two types of DT can be constructed: axis-parallel or multivariate DTs. An axis-parallel DT is a univariate DT that evaluates a single attribute in each test condition to split the training set. On the other hand, oblique DTs and nonlinear DTs are multivariate DTs in which a linear combination and a nonlinear composition of attributes are utilized in the test conditions of a DT, respectively. Multivariate DTs commonly show better performance, and they are smaller than univariate DTs, but they require more computational effort to induce them. In particular, an oblique hyperplane divides the instance space into two halfspaces, and it is defined as follows:

$$\sum_{j=1}^{d} h_j x_j + b > 0 \tag{1}$$

where $d$ is the number of attributes in the dataset, $x_j$ is the value of the $j$-th attribute, $h_j$ is a real-valued coefficient in the hyperplane, and $b$ represents the independent term of the hyperplane. **Figure 1** shows an axis-parallel DT induced from the iris dataset [17] using the J48 method [18], and **Figure 2** shows a near-optimal oblique DT constructed from the same dataset by the DE-based method implementing a global search strategy. Iris dataset has four attributes, three

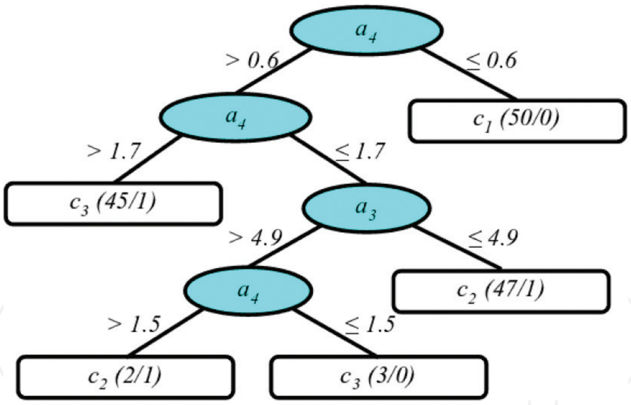

**Figure 1.** An axis-parallel DT induced from the iris dataset.

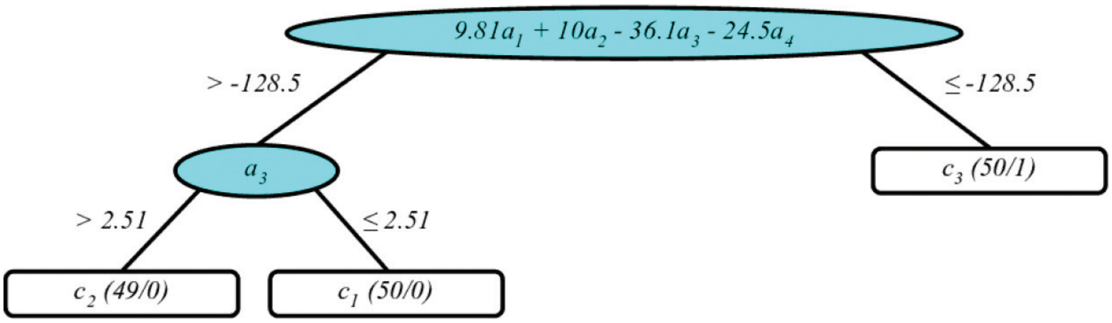

**Figure 2.** An oblique DT induced from the iris dataset.

class labels, and 150 instances. It is clear that the oblique DT is more compact and more accurate than its axis-parallel version, but it has been proved that to find the oblique hyperplane that minimizes the number of misclassified instances both above and below is an NP-hard problem [19].

On the other hand, MHs are general algorithmic templates that can be easily adapted to solve almost all optimization problems [20]. MHs are nature-inspired procedures using stochastic components to find a near-optimal solution and have several parameters that need to be fitted to the specific problem [21]. In accordance with the number of candidate solutions used in its search procedure, MHs have been grouped in single-solution-based MHs and population-based MHs [22]. Single-solution-based MHs implement intelligent search procedures that iteratively replace a candidate solution with a neighboring solution with the aim of reaching a near-optimal solution. Simulated annealing (SA) and Tabu search (TS) are two well-known single-solution-based MHs. Population-based MHs use a group of candidate solutions in each step of their iterative process. The most commonly used population-based MHs are related to EAs and Swarm intelligence (SI) methods. Genetic algorithms (GA), genetic programming (GP), evolutionary strategies (ES) and DE are the most prominent EAs, and ant colony optimization (ACO) and particle swarm optimization (PSO) are examples of SI methods.

In particular, DE is an effective EA designed to solve optimization problems with real-valued parameters [6]. DE evolves a *population* $X = \{x^1, , x^2, \dots, , x^{\text{NP}}\}$ of NP *chromosomes* by applying mutation, crossover, and selection operators with the aim to reach a near-optimal solution. Several DE variants differing in the implementation of the mutation and crossover operators have been described in existing literature. In this chapter, the standard DE algorithm, named `DE/rand/1/bin` in agreement with the nomenclature adopted to refer DE variants, is used as a procedure to find a near-optimal solution. At each iteration of this evolutionary process, known as a *generation*, a new population of chromosomes is generated from the previous one. For each $i \in \{1, \dots, \mathrm{NP}\}$ in the $g$-th generation, $x^i$ is taken from the $X_{g-1}$ population, and it is used to build a new vector $u^i$ by applying the mutation and crossover operators. Vectors $x^i$ and $u^i$ are known as the *target vector* and the *trial vector*, respectively. To build a new chromosome, instead of implementing a traditional mutation operator, DE first applies a linear combination of several chromosomes randomly chosen from the current population ($x^{r_1}, x^{r_2}$, and $x^{r_3}$) to construct a mutated vector $v^i = x^{r_1} + \mathrm{F}(x^{r_2} - x^{r_3})$, where F is a user-specified value representing a scale factor applied to control the differential variation. Next, the crossover operator determines each parameter in $u^i$ from either $x^i$ or $v^i$, based on a stochastic decision. If a random value is less than a crossover factor (CF), the $j$-th parameter value of $u^i$ is taken from $v^i$, otherwise its value is $u^i_j = x^i_j$. Finally, a one-to-one tournament is applied to determine which vector, between $x^i$ and $u^i$, is selected as a member of the new population $X_g$. **Figure 3** shows a scheme of the application of the DE operators to build a new chromosome for the next population.

DE has been used in conjunction with several machine learning techniques to implement classification methods [23–27]. It has been mainly applied to optimize the parameters of classification methods or to conduct preprocessing tasks in a data mining process. DE has several advantages in comparison with other MHs, and since mutation operator is based on a linear combination of several randomly chosen individuals, DE exhibits a good trade-off between its exploitation and exploration skills [28]. On the other hand, although DE requires the definition of a smaller number of parameters compared to other MHs, its performance is sensitive to the values selected for CR, F, and NP.

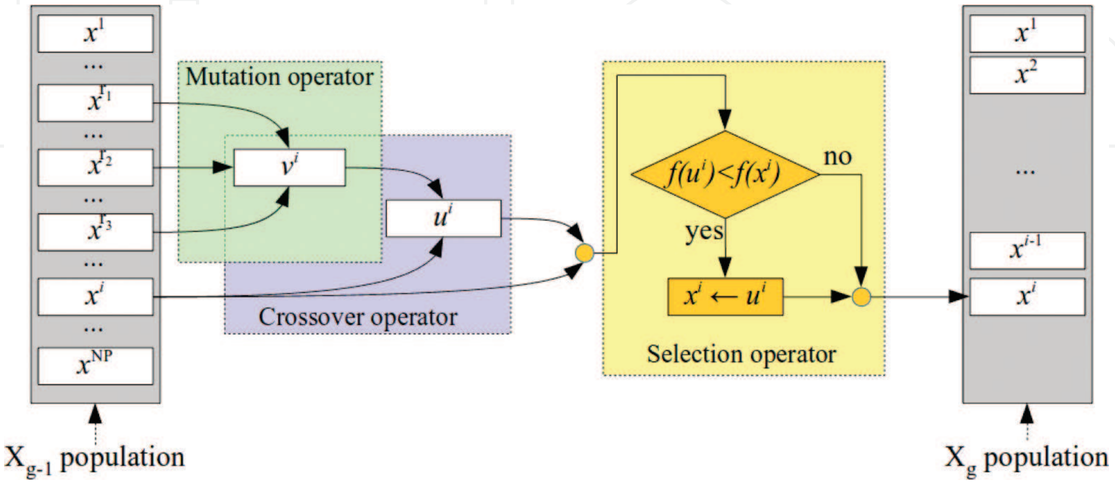

**Figure 3.** DE operators applied to build a new chromosome for the next population.

## 3. Induction of oblique decision trees using metaheuristics

Several MHs have been used to induce DTs with methods implementing a recursive partitioning strategy. One timeline of these methods is shown in **Figure 4**. Single-solution-based MHs such as SA and TS have been used to induce DTs through this strategy. SA is applied in the simulated annealing of decision trees (SADT) method [29] that iteratively perturbs one randomly selected coefficient to build a new hyperplane, and in one variant of the oblique classifier 1 (OC1) system [30], named OC1-SA [31], that disturbs simultaneously several coefficients of the best axis-parallel hyperplane found by the OC1 algorithm. TS is used in the linear discriminant and TS (LDTS) method [32] and in the linear discrete support vector DT with TS ($\text{LDSDT}_{TS}$) method [33]. Furthermore, EAs such as ES, GA, and DE also have been applied to build an oblique DT through this strategy. The OC1-ES algorithm [31] and the multimembered ES oblique DT (MESODT) method [34] obtain a near-optimal hyperplane using the $(1+1)$-ES and the $(\mu, \lambda)$-ES, respectively. Furthermore, GA evolves a population of hyperplanes encoded: (1) with a binary chromosome in the binary tree-GA (BTGA) algorithm [35] and in the HereBoy for DT (HBDT) method [36] and (2) with a real-valued chromosome in the OC1-GA algorithm [31] and in the procedures described by Kr&#281;towski [37], and by Pangilinan and Janssens [38]. Finally, DE is applied in an OC1 variant named OC1-DE algorithm [39].

On the other hand, several MH-based approaches implementing a global search strategy have been described in the existing literature. GA evolves a population of variable-length chromosomes in the generalized decision tree inducer (GDTI) method [40] and in the evolutionary full tree induction (EFTI) method [41]. Other GA-based approaches such as the Global EA for oblique DTI (GEA-ODT) procedure [42, 43] and the tree analysis with randomly generated

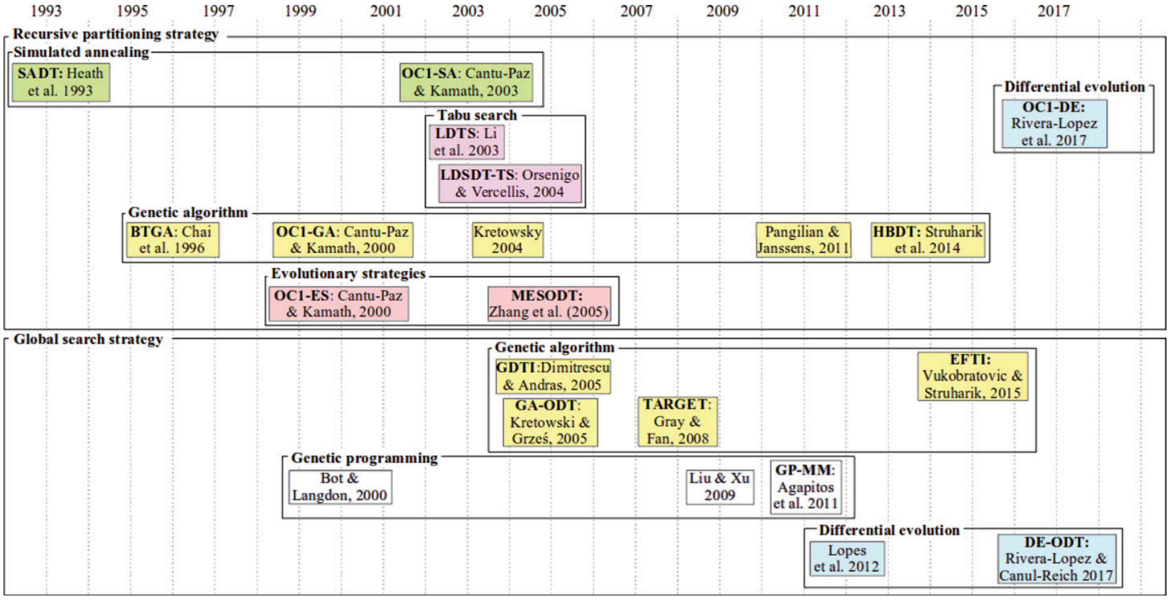
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
Recursive partitioning strategy
Simulated annealing
SADT: Heath et al. 1993
OC1-SA: Cantu-Paz & Kamath, 2003
Tabu search
LDTS: Li et al. 2003
LDSDT-TS: Orsenigo & Vercellis, 2004
Differential evolution
OC1-DE: Rivera-Lopez et al. 2017
Genetic algorithm
BTGA: Chai et al. 1996
OC1-GA: Cantu-Paz & Kamath, 2000
Kretowsky 2004
Pangilian & Janssens, 2011
HBDT: Struharik et al. 2014
Evolutionary strategies
OC1-ES: Cantu-Paz & Kamath, 2000
MESODT: Zhang et al. (2005)
Global search strategy
Genetic algorithm
GDTI:Dimitrescu & Andras, 2005
GA-ODT: Kretowski & Grześ, 2005
TARGET: Gray & Fan, 2008
EFTI: Vukobratovic & Struharik, 2015
Genetic programming
Bot & Langdon, 2000
Liu & Xu 2009
GP-MM: Agapitos et al. 2011
Differential evolution
Lopes et al. 2012
DE-ODT: Rivera-Lopez & Canul-Reich 2017

**Figure 4.** Timeline of the MH-based approaches to induce oblique DTs.

and evolved trees (TARGET) algorithm [44] use trees as chromosomes. Furthermore, the standard GP is applied by Liu and Xu [45], the strongly typed GP is used by Bot and Langdon [46, 47], and the grammar-based GP is utilized in the GP with margin maximization (GP-MM) method [48]. Finally, DE is implemented in the perceptron DT (PDT) method [49, 50] and in the DE for inducing oblique DTs (DE-ODT) method [51].

## 4. DE-based methods to induce oblique decision trees

In this chapter, two methods to induce an oblique DT using the DE/rand/1/bin algorithm are described. The first method, named OC1-DE, is similar to the OC1 system and its variants, but it applies DE to find a near-optimal hyperplane at each internal node of an oblique DT [39]. The second one, named DE-ODT method, implements a global search strategy to induce oblique DTs [51].

### 4.1. OC1-DE method to search near-optimal oblique hyperplanes

The OC1-DE method is based on the OC1 system [30] and the OC1-GA procedure [31]. The OC1 system applies a two-step process to find a better hyperplane. First, it finds the best axis-parallel hyperplane splitting the instance set. Next, it applies two procedures to disturb the hyperplane coefficients:

- *Sequential perturbation*: This is a deterministic rule that adjusts the hyperplane coefficients, taking one at a time and looking for its optimal value.
- *Random vector perturbation*: When the sequential perturbation reaches a local optimum, a random vector is added to the current hyperplane with the aim of looking elsewhere in the solutions space.

Finally, this procedure returns as the best hyperplane to the one selected between the best-perturbed hyperplane and the best axis-parallel hyperplane. OC1 uses several criteria to evaluate the quality of the candidate hyperplanes such as information gain [52] and three criteria introduced by Heath [19]: max minority, minority sum, and sum of impurities. Induced DT is pruned by removing sub-trees whose impurity value is less than a predefined threshold value. An improved OC1 version [53] uses the elements defined in the CART method [54]: the Gini index and the twoing rule as splitting criteria and the cost-complexity pruning method.

On the other hand, the OC1-GA method is an OC1 variant encoding the hyperplane coefficients in a real-valued chromosome. First, the axis-parallel hyperplane that best splits the training instances is obtained. This hyperplane is copied to 10% of the initial population and the remaining hyperplanes are randomly created. Then, OC1-GA evolves this population to find a near-optimal hyperplane evaluating its quality through the twoing rule. Oblique DT is then induced in a recursive partitioning strategy, and it is pruned using the cost-complexity pruning method.

The DE implementation to find a near-optimal hyperplane at each internal node of an oblique DT is shown in the **Algorithm 1**. First, the axis-parallel hyperplane that best splits a set of training instances is obtained (line 1). It is copied to 10% of the initial population, as is proposed in [31], and the remaining hyperplanes are randomly created (line 2). Each random hyperplane is constructed considering that almost two instances with different class labels are separated by the hyperplane. Next, this population is evolved through several generations using the DE operators (lines 3–19), and the best hyperplane in the population is selected (line 20). Finally, the OC1-DE algorithm returns the hyperplane selected between the best axis-parallel hyperplane and the best oblique hyperplane produced by DE (line 21).

The hyperplane returned by the OC1-DE is used as the test condition of a new internal node that is added in an oblique DT. This hyperplane is used to split the training instances into two subsets. The OC1-DE method is recursively applied using each subset until a leaf node is created as all instances in the subset have the same class label or a threshold value of unclassified instances is reached. The quality of the hyperplane is obtained using the twoing rule. Finally, a pruning procedure is applied in order to reduce the overfitting of DT produced and to improve its predictive power.

**Algorithm 1** Hyperplane selection using the OC1-DE algorithm.

```
 1: $h^{\mathrm{AP}} \leftarrow$ Best axis-parallel hyperplane for the training instances
 2: $X_0 \leftarrow$ Initial population of random hyperplanes with 10% of copies of $h^{\mathrm{AP}}$
 3: for $g$ in $[1,\ldots,\mathrm{it}]$ do
 4:     $X_g \leftarrow \emptyset$
 5:     for each $i \in \{1,\ldots,\mathrm{NP}\}$ do
 6:         $h^i \leftarrow$ The trial hyperplane from $X_{g-1}$
 7:         $\{h^{r_1},h^{r_2},h^{r_3}\} \leftarrow$ Randomly selected hyperplanes from $X_{g-1}$
 8:         $v^i \leftarrow h^{r_1} + \mathrm{F}(h^{r_2} - h^{r_3})$                ▷ DE mutation operator
 9:         for each $j \in \{1,\ldots,|h^i|\}$ do
10:             $r \leftarrow$ A uniformly distributed random number from $[0,1]$
11:             $l \leftarrow$ A randomly chosen index from $\{1,\ldots,|x^i|\}$
12:             $u^i_j \leftarrow \begin{cases} v^i_j & \text{if } r \le \mathrm{CR} \text{ or } j = l, \\ h^i_j & \text{otherwise.} \end{cases}$    ▷ DE crossover operator
13:         end for
14:         if $fitness(u^i) < fitness(h^i)$ then                ▷ DE selection operator
15:             $h^i \leftarrow u^i$
16:         end if
17:         $X_g \leftarrow X_g \cup \{h^i\}$
18:     end for
19: end for
20: $h^{\mathrm{best}} \leftarrow$ Best hyperplane in $X_g$
21: return $\begin{cases} h^{\mathrm{AP}} & \text{if } fitness(h^{\mathrm{AP}}) < fitness(h^{\mathrm{best}}), \\ h^{\mathrm{best}} & \text{otherwise.} \end{cases}$
```

### 4.2. DE-ODT method to induce oblique decision trees

The DE-ODT method implements a global search strategy in which the DE algorithm is applied to find a near-optimal oblique DT, where each real-valued chromosome encodes only a feasible oblique DT.

### *4.2.1. Linear representation of oblique decision trees*

In the DE-ODT method, each chromosome represents the internal nodes of a complete binary oblique DT stored in a fixed-length real-valued vector (**Figure 5**). This vector encodes the set of hyperplanes used as test conditions of the oblique DT. Vector size is determined using both the number of attributes and the number of class labels of the training set whose model is induced. Since each internal node of an oblique DT has a hyperplane as its test condition, the size of the real-valued vector $x^i$ used to encode each $i$-th candidate solution in the population is fixed as $n_e(d+1)$, where $n_e$ is the estimated number of internal nodes of a complete binary oblique DT. Considering that: (1) an oblique DT is more compact than its univariate version and since (2) the DT size is related to the structure of the training set, the DE-ODT method determines the value of $n_e$ based on both the number of attributes and the number of class labels ($s$) in it.

If the number of internal nodes of a complete binary DT with height $H$ is $2^{H-1}-1$ and the number of leaf nodes of the same DT is $2^{H-1}$, two heights can be obtained: $H_i = \lceil \log_2(d+1) +1 \rceil$, and $H_l = \lceil \log_2(s)+1 \rceil$. Using these equations, $n_e$ is determined as follows:

$$n_e = 2^{\max(H_i,H_l)-1} - 1, \tag{2}$$

and, the size of the real-valued parameter vector representing a sequence of $n_e$ hyperplanes for a training set with $d$ attributes is computed as follows:

$$n = n_e(d+1). \tag{3}$$

As an example, if a hypothetical dataset with three numerical attributes and three class labels is used to induce an oblique DT, then $d=3$ and $s=3$. In this case, $H_i = \lceil \log_2(4)+1 \rceil = 3$ and $H_l = \lceil \log_2(3)+1 \rceil = 3$. Finally, $n_e = 2^{\max\{3,3\}} - 1 = 7$. This implies that the oblique DT could have seven internal nodes. Finally, one chromosome representing a candidate solution in the evolutionary process of this problem has 28 real-valued parameters.

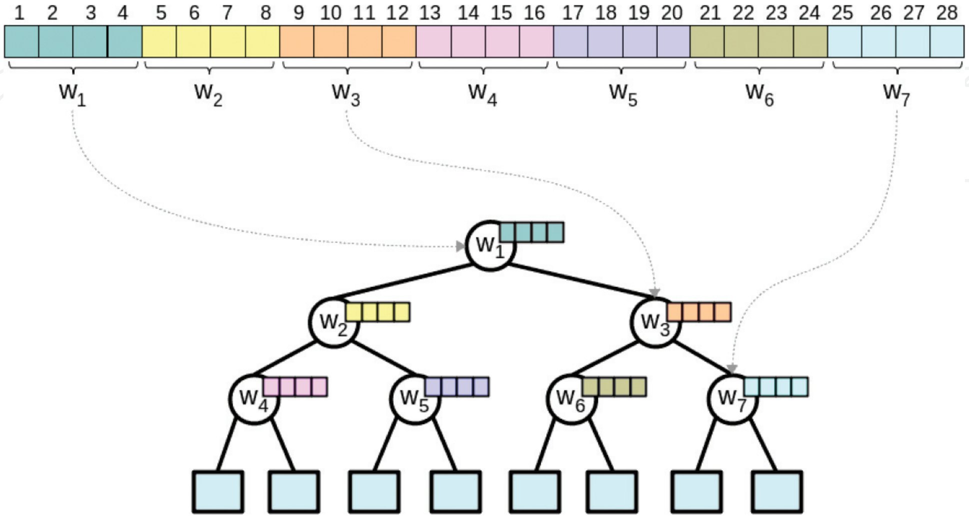

**Figure 5.** Linear encoding scheme for the internal nodes of a complete binary oblique tree.

### *4.2.2. Induction of feasible oblique decision trees*

The DE-ODT method applies the following steps to map an oblique DT from a chromosome $x^i$ of the population:

1. **Hyperplanes construction:** $x^i$ is used to build the vector $w^i$ representing the sequence of candidate hyperplanes utilized in the internal nodes of a partial DT. Since the values of $x^i$ represent the hyperplane coefficients contained in these nodes, the following criterion applies: Values $\{x_1^i, \dots, x_{d+1}^i\}$ are assigned to the hyperplane $h^1$, the values $\{x_{d+2}^i, \dots, x_{2d+2}^i\}$ are assigned to the hyperplane $h^2$, and so on. For each $j \in \{1, \dots, n_e\}$, and for each $k \in \{1, \dots, d+1\}$, the $k$-th coefficient of $h^j$ is designed as follows

$$h_k^j = x_{(j-1)(d+1)+k}^i. \tag{4}$$

   These hyperplanes are assigned to the elements of $w^i$: $h^1$ is assigned to $w_1^i$, $h^2$ is assigned to $w_2^i$, and so on. **Figure 6** shows an example of the construction of a set of hyperplanes from one chromosome for the hypothetical dataset previously described. Once $w^i$ is completed, it is used to create a partial DT with only internal nodes.

2. **Partial oblique decision tree construction:** $w^i$ is used to create the partial tree ($pT^i$). First, the element in the initial location of $w^i$ is used as the root node of $pT^i$. Next, the remaining elements of $w^i$ are inserted in $pT^i$ as successor nodes of those previously added so that each new level of the tree is completed before placing new nodes at the next level, in a similar way to the breadth-first search strategy. **Figure 7** shows an example of the construction of a partial oblique DT from $w^i$. In this figure, it can be observed that $w_1^i$ is selected as the tree root node, $w_2^i$ and $w_3^i$ are placed as the successor nodes of $w_1^i$, $w_4^i$ and $w_5^i$ are designed as the successor nodes of $w_2^i$, and so on.

3. **Oblique decision tree completion:** The final stage of the mapping scheme adds leaf nodes in $pT^i$ using the training set. In this stage, one instance set is assigned to a node (the

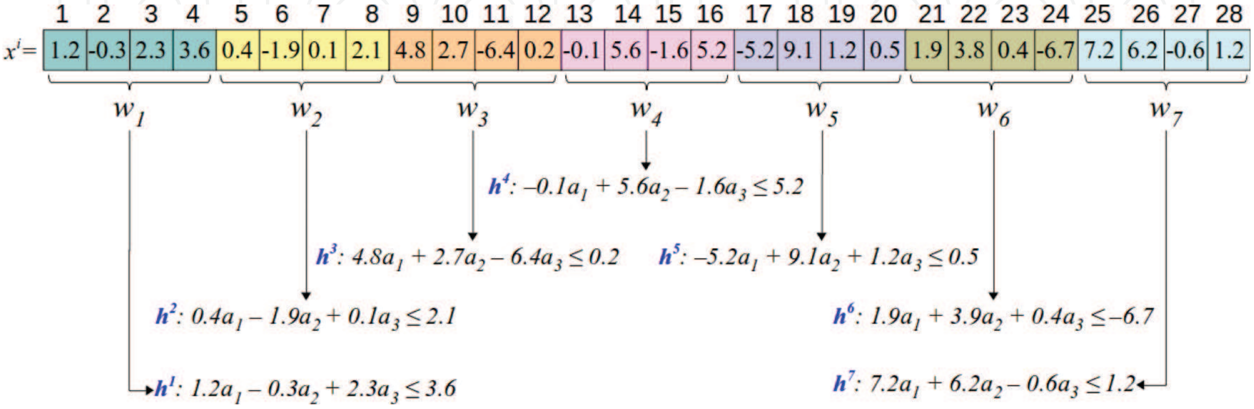
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
$x^i$=
1.2 -0.3 2.3 3.6 0.4 -1.9 0.1 2.1 4.8 2.7 -6.4 0.2 -0.1 5.6 -1.6 5.2 -5.2 9.1 1.2 0.5 1.9 3.8 0.4 -6.7 7.2 6.2 -0.6 1.2
$w_1$
$w_2$
$w_3$
$w_4$
$w_5$
$w_6$
$w_7$
$h^4$: $-0.1a_1 + 5.6a_2 - 1.6a_3 \leq 5.2$
$h^3$: $4.8a_1 + 2.7a_2 - 6.4a_3 \leq 0.2$
$h^5$: $-5.2a_1 + 9.1a_2 + 1.2a_3 \leq 0.5$
$h^2$: $0.4a_1 - 1.9a_2 + 0.1a_3 \leq 2.1$
$h^6$: $1.9a_1 + 3.9a_2 + 0.4a_3 \leq -6.7$
$h^1$: $1.2a_1 - 0.3a_2 + 2.3a_3 \leq 3.6$
$h^7$: $7.2a_1 + 6.2a_2 - 0.6a_3 \leq 1.2$

**Figure 6.** Construction of a set of hyperplanes from $x^i$.

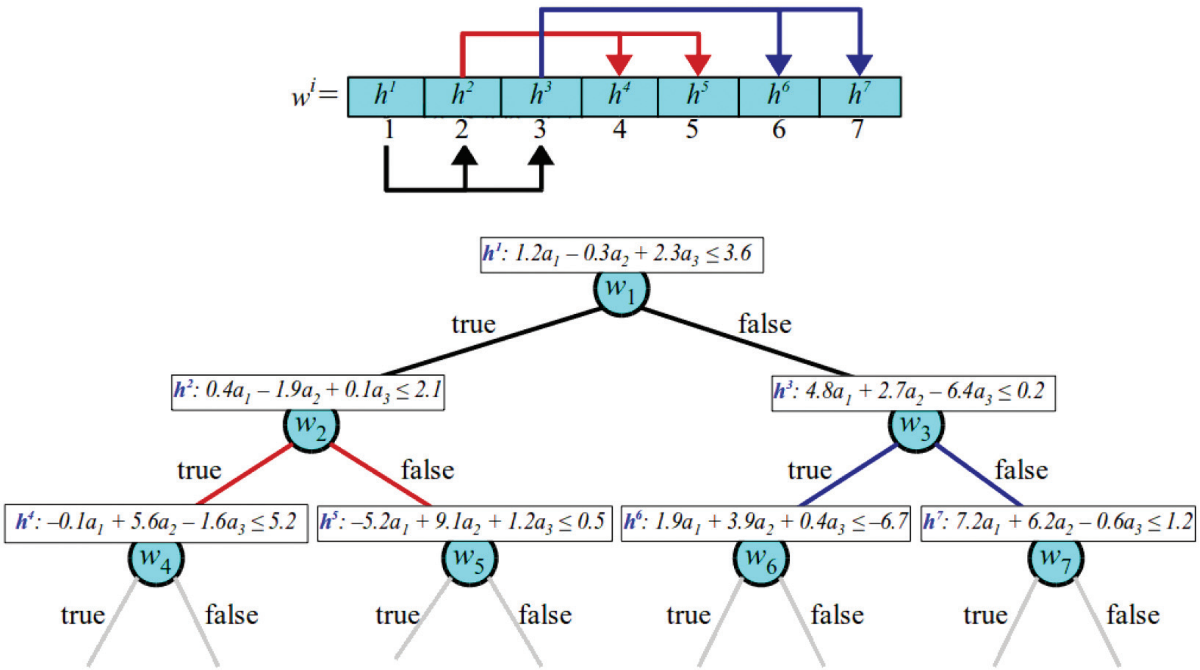
$w^i =$
$h^1$ $h^2$ $h^3$ $h^4$ $h^5$ $h^6$ $h^7$
1 2 3 4 5 6 7
$h^1$: $1.2a_1 - 0.3a_2 + 2.3a_3 \le 3.6$
$w_1$
true
false
$h^2$: $0.4a_1 - 1.9a_2 + 0.1a_3 \le 2.1$
$w_2$
$h^3$: $4.8a_1 + 2.7a_2 - 6.4a_3 \le 0.2$
$w_3$
true
false
true
false
$h^4$: $-0.1a_1 + 5.6a_2 - 1.6a_3 \le 5.2$
$w_4$
$h^5$: $-5.2a_1 + 9.1a_2 + 1.2a_3 \le 0.5$
$w_5$
$h^6$: $1.9a_1 + 3.9a_2 + 0.4a_3 \le -6.7$
$w_6$
$h^7$: $7.2a_1 + 6.2a_2 - 0.6a_3 \le 1.2$
$w_7$
true
false
true
false
true
false
true
false

**Figure 7.** Construction of a partial oblique DT with only internal nodes.

complete training set for the root node of the tree), and it is labeled as an internal node. To evaluate each instance in this set using the hyperplane associated to the internal node, two instances subsets are created, and they are assigned to the successor nodes of this node. This assignment is repeated for each node of the partial DT. If the internal node is located at the end of a branch of the DT, then two leaf nodes are created, and they are designated as successor nodes of this node. The instances subsets created are assigned to these leaf nodes. On the other hand, if all instances in the set assigned to the internal node have the same class label, it is labeled as a leaf node and its successor nodes are removed, if they exist. **Figure 8** shows an example of this tree-completion procedure. **Figure 8** shows that all the instances assigned to $w_3$ and $w_5$ have the same class label, so they are designated as leaf nodes, and the successor nodes of $w_3$ are removed from the tree. On the other hand, since $w_4$ is the ending node of a branch, its instance set is split using its hyperplane, the instances subsets produced are assigned to two new leaf nodes, and their majoritarian classes are assigned as their class labels. It can be observed that this tree has three internal nodes and four leaf nodes.

### *4.2.3. General structure of the DE-ODT method*

The **Algorithm 2** shows the structure of the DE-ODT method described in this chapter. This procedure requires to identify the training set used to induce an oblique DT, as well as the three control parameters applied by the DE algorithm (CR, F, and NP) and the threshold value ($\tau$) used to determine if a node is labeled as a leaf node. First, the DE-ODT method gets the attributes vector ($a$), the vector of class labels ($c$), and the instance set ($\iota$) from the dataset whose

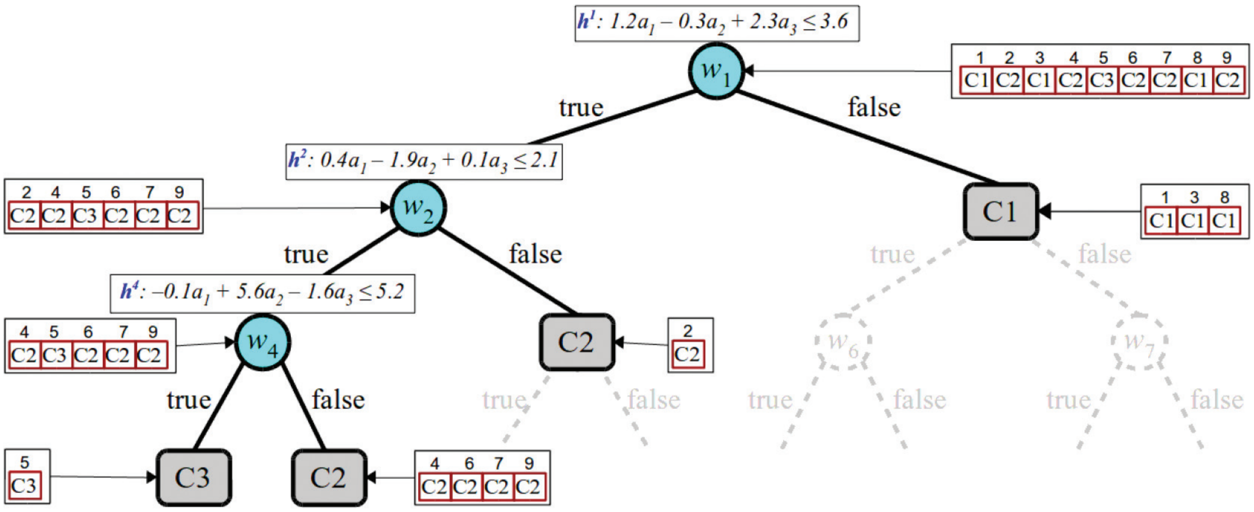

**Figure 8.** Completion of an oblique DT using $pT^i$ and the training instances.

model must be built (line 1). Next, the value of $d$ and $n$ are computed (lines 2–4). Then, the DE algorithm evolves a population of real-valued chromosomes encoding oblique DTs. DE selects the best candidate solution $x^{\text{best}}$ in the last population as the result of its evolutionary process (line 5). After that, a near-optimal oblique DT is constructed applying the procedures described in the previous paragraphs (lines 6–8). Since the DE-ODT method uses an a priori definition of the size of the chromosome, it is possible that some leaf nodes in the DT do not meet the following conditions: that the size of its instances subset is less than $\tau$ or that all the instances in the subset belong to the same class. The DE-ODT method refines the DT by replacing nonoptimal leaf nodes with sub-trees (line 9). Finally, the oblique DT is pruned to reduce the possible overfitting generated by applying this refinement (line 10).

This procedure allows inducing feasible oblique DTs with a different number of nodes, although they are represented with a fixed-length parameter vector.

**Algorithm 2** General structure of DE-ODT method.

```
1:  $(a,c,\iota) \leftarrow$ READTRAININGSET($trainingSet$)
2:  $d \leftarrow |a|$                                                   ▷ Number of attributes
3:  $n_e \leftarrow$ Number of estimated internal nodes computed using (1.2)
4:  $n \leftarrow n_e(d+1)$                                   ▷ Size of the real-valued chromosome
5:  $x^{\text{best}} \leftarrow$ DIFFERENTIALEVOLUTION(CR, F, NP. $n$)
6:  $w \leftarrow$ HYPERPLANESCONSTRUCTION($x^{\text{best}}$)                    ▷ Nodes vector
7:  $pT \leftarrow$ PARTIALDTCONSTRUCTION($w$)                     ▷ DT with only internal nodes
8:  $T \leftarrow$ DTCOMPLETION($pT,\iota,\tau$)                              ▷ Complete DT
9:  $T \leftarrow$ DTREFINEMENT($T,\tau$)                                      ▷ Refined DT
10: $T \leftarrow$ DTPRUNING($T,\tau$)                                         ▷ Pruned DT
```

## 5. Experimental study

In this chapter, the experimental study carried out to analyze the performance of the DE-based methods for DTI is detailed. First, a description of the datasets used in this study, as well as the definition of the parameters of each method, are given. Then, both the model validation technique used in the experiments and the statistical tests applied to evaluate the results obtained are outlined. Finally, a discussion about the performance of the DE-based methods is provided.

### 5.1. Experimental setup

A benchmark of 20 datasets chosen from the UCI machine learning repository [55] is used to carry out the experimental study. These datasets have been selected as their attributes are numerical, their instances are classified into two or more classes, and most of them are imbalanced datasets. **Table 1** shows the description of these datasets. To ensure that the comparison of the results achieved by the DE variants with those produced by other approaches is not affected by the treatment of the data, all datasets used in this study do not have missing values. Also, the data are not preprocessed, filtered, or normalized, that is, they are used as they are obtained from the UCI repository.

The DE-based methods are implemented in the Java language using the JMetal library [56]. The mutation scale factor is linearly decreased from 0.5 to 0.1 as the evolutionary process progresses, and the crossover rate is fixed at 0.9. The decrement in the *F* value allows more exploration of search space at the beginning of the evolutionary process, and with the passage of the generations, it tries to make a better exploitation of promising areas of this space [57]. The population size is adjusted to $5n$, with 250 and 500 chromosomes as lower and upper bound, respectively. These bounds are used to ensure that the population is not so small as not to allow a reasonable exploration of the search space and is not so large as to impact the runtime of the algorithm. Furthermore, the fitness function used in the DE-ODT method computes the training accuracy of each DT in population, and the twoing rule is used as fitness value in the OC1-DE method. The best oblique DT induced by these methods is pruned using the error-based pruning (EBP) approach [58]. Finally, the threshold value used to determine

| Dataset | Instances | Attributes | Classes | Class distribution | Dataset | Instances | Attributes | Classes | Class distribution |
|---|---|---|---|---|---|---|---|---|---|
| Glass | 214 | 9 | 7 | 70\|76\|17\|0\|13\|9\|29 | Diabetes | 768 | 8 | 2 | 500\|268 |
| Balance-scale | 625 | 4 | 3 | 288\|49\|288 | Heart-statlog | 270 | 13 | 2 | 150\|120 |
| Iris | 150 | 4 | 3 | 50 instances per class | Australian | 690 | 14 | 2 | 307\|383 |
| Ionosphere | 351 | 34 | 2 | 126\|225 | Wine | 178 | 13 | 3 | 59\|71\|48 |
| Sonar | 208 | 60 | 2 | 97\|111 | Vehicle | 846 | 18 | 4 | 212\|217\|218\|199 |
| Liver-disorder | 345 | 6 | 2 | 145\|200 | Page-blocks | 5473 | 10 | 5 | 4913\|329\|28\|88\|115 |
| Blood-t | 748 | 4 | 2 | 570\|178 | Breast-tissue-6 | 106 | 9 | 6 | 22\|21\|14\|15\|16\|18 |
| Movement-libras | 360 | 90 | 15 | 24 instances per class | Parkinsons | 195 | 22 | 2 | 48\|147 |
| Seeds | 210 | 6 | 3 | 70 instances per class | Segment | 2310 | 19 | 7 | 330 instances per class |
| Ecoli | 336 | 7 | 8 | 143\|77\|52\|35\|20\|5\|2\|2 | Spambase | 4601 | 57 | 2 | 1813\|2788 |

**Table 1.** Description of datasets used in the experiments.

whether a node should be labeled as one leaf node is set to be two instances, and the DT size is defined as the number of leaf nodes of the oblique DT.

In this study, a repeated stratified 10-fold cross-validation (CV) procedure is used to estimate the predictive performance of the DE-based methods, and the Friedman test [59] is applied to carry out a statistical analysis of the results produced by these methods as compared to them with those obtained by other classification methods. This nonparametric statistical test evaluates the statistical significance of the experimental results through computing the *p*-value without making any assumptions about the distribution of the analyzed data. This *p*-value is used to accept or to reject the null hypothesis $H_0$ of the experiment which holds that the performance of the compared algorithms does not present significant differences. If the *p*-value does not exceed a predefined significance level, $H_0$ is rejected and the Bergmann-Hommel (BH) post hoc test [60] is conducted to detect the differences between all existing pairs of algorithms. These statistical tests are applied using the *scmamp* R library [61].

| Dataset | J48 | | sCART | | OC1-DE | | DE-ODT | |
|---|---|---|---|---|---|---|---|---|
| Glass | 67.62 | (4) | 71.26 | (2) | **71.31** | (1) | 68.97 | (3) |
| Diabetes | 74.49 | (3) | 74.56 | (2) | 73.37 | (4) | **75.79** | (1) |
| Balance-scale | 77.82 | (4) | 78.74 | (3) | **93.92** | (1) | 91.97 | (2) |
| Heart-statlog | 78.15 | (2) | 78.07 | (3) | 74.11 | (4) | **81.11** | (1) |
| Iris | 94.73 | (3) | 94.20 | (4) | 96.73 | (2) | **97.17** | (1) |
| Australian | 84.35 | (4) | 85.19 | (2.5) | 85.19 | (2.5) | **85.61** | (1) |
| Ionosphere | 89.74 | (3) | 88.86 | (4) | 91.11 | (2) | **92.28** | (1) |
| Wine | **93.20** | (1) | 89.49 | (4) | 92.58 | (2) | 91.88 | (3) |
| Sonar | 73.61 | (3) | 70.67 | (4) | 77.65 | (2) | **79.34** | (1) |
| Vehicle | 72.28 | (2) | 69.91 | (4) | **72.32** | (1) | 71.33 | (3) |
| Liver-disorders | 65.83 | (4) | 66.64 | (3) | 67.63 | (2) | **71.16** | (1) |
| Page-blocks | 96.99 | (2) | 96.76 | (4) | 96.88 | (3) | **97.07** | (1) |
| Blood-t | 78.20 | (2) | 77.86 | (3) | 76.35 | (4) | **78.70** | (1) |
| Breast-tissue-6 | 34.81 | (3) | 32.45 | (4) | 34.91 | (2) | **38.85** | (1) |
| Movement-libras | 69.31 | (2) | 65.64 | (3) | **75.11** | (1) | 55.63 | (4) |
| Parkinsons | 84.72 | (4) | 86.31 | (3) | **87.95** | (1) | 86.43 | (2) |
| Seeds | 90.90 | (3.5) | 90.90 | (3.5) | **93.76** | (1) | 91.79 | (2) |
| Segment | **96.79** | (1) | 95.83 | (3) | 95.93 | (2) | 94.78 | (4) |
| Ecoli | 82.83 | (4) | 83.15 | (3) | 83.51 | (2) | **84.72** | (1) |
| Spambase | 92.68 | (2) | 92.35 | (3) | 92.19 | (4) | **93.94** | (1) |
| **Average ranking** | | 2.825 | | 3.250 | | 2.175 | | 1.750 |

**Table 2.** Average accuracies obtained by the DTI algorithms and the DE-based methods.

The results obtained with the DE-based methods are compared with those achieved by several supervised learning methods available on the WEKA data mining software [62]. First, the accuracy and the size of the DTs got by these algorithms are compared with those obtained by the J48 method [63] and by the SimpleCART (sCART) [54] procedure. Next, the accuracy of the DTs constructed with the DE-based procedures is compared with those achieved using the following classification methods: Naïve Bayes (NB) [64], multilayer perceptron (MLP) [65], radial basis function neural network (RBF-NN) [66], and random forest (RF) [67].

### 5.2. Comparison with DTI methods

**Table 2** and **Figure 9** show the average accuracies of the DTs induced by the DTI algorithms as well as those achieved by the OC1-DE method. In **Table 2**, the best result for each dataset is highlighted with bold numbers, and the numbers in parentheses refer to the ranking reached by each method for each dataset. The last row in this table indicates the average ranking of each method. It is observed that the DE-based methods produce better results than those generated by the other DTI algorithms.

A statistical test of the experimental results is conducted to evaluate the performance of the DE-based methods. First, the Friedman test is run and its resulting statistic value is 16.197 for four methods and 20 datasets, which has a $p$-value of $1.033 \times 10^{-3}$. When evaluating this $p$-value with a significance level of 5%, $H_0$ is rejected. Next, the BH post hoc test is applied to

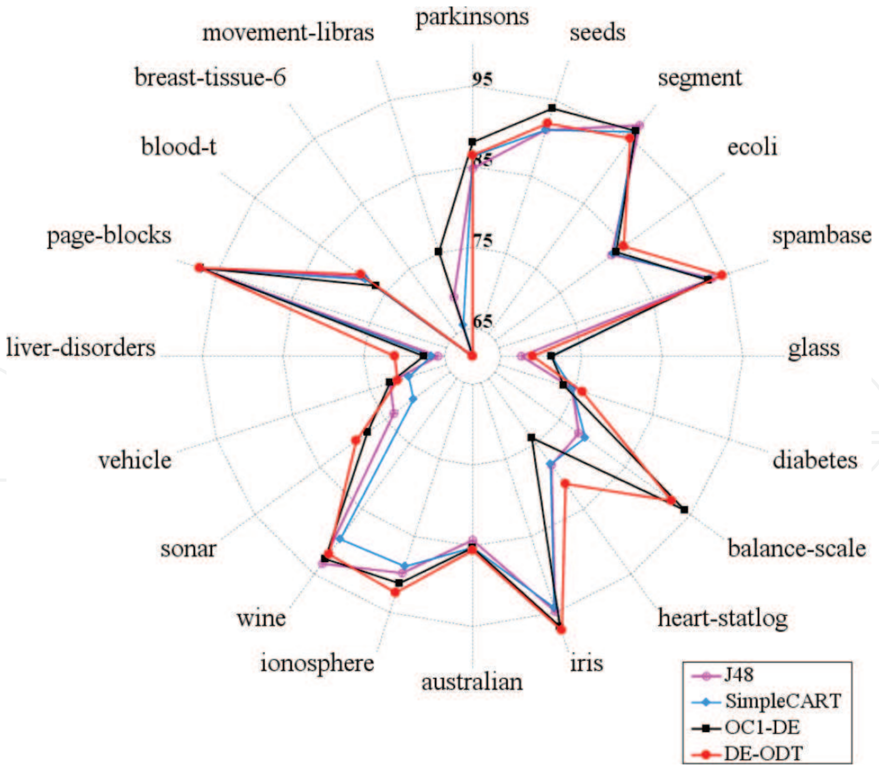

**Figure 9.** Graphical comparison of the average accuracies obtained by the DTI algorithms and the DE-based methods.

| **Method** | **AR** | **OC1-DE** | | **DE-ODT** | |
|---|---|---|---|---|---|
| | | **Unadjusted** | **BH** | **Unadjusted** | **BH** |
| J48 | 2.825 | 1.1134e-01 | 1.1134e-01 | 8.4584e-03 | **2.5375e-02** |
| sCART | 3.250 | 8.4584e-03 | **2.5375e-02** | 2.3856e-04 | **1.4131e-03** |
| OC1-DE | 2.175 | — | — | 2.9786e-01 | 5.9572e-01 |
| DE-ODT | 1.750 | 2.9786e-01 | 5.9572e-01 | — | — |

**Table 3.** The *p*-values for multiple comparisons among DTI algorithms and the DE-based methods.

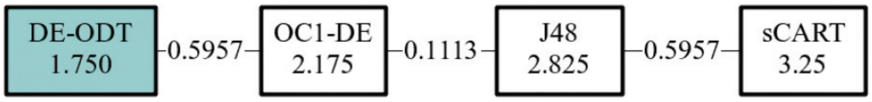

**Figure 10.** The *p*-value graph of the DTI algorithms and the DE-based methods.

| **Dataset** | **J48** | | **sCART** | | **OC1-DE** | | **DE-ODT** | |
|---|---|---|---|---|---|---|---|---|
| Glass | 23.58 | (4) | **8.00** | (1) | 21.61 | (3) | 11.08 | (2) |
| Diabetes | 22.20 | (3) | **3.00** | (1) | 41.55 | (4) | 14.77 | (2) |
| Balance-scale | 41.60 | (4) | 13.00 | (2) | 15.24 | (3) | **5.01** | (1) |
| Heart-statlog | 17.82 | (4) | 16.00 | (2) | 17.43 | (3) | **7.23** | (1) |
| Iris | 4.64 | (3) | 5.00 | (4) | **3.00** | (1) | 3.37 | (2) |
| Australian | 25.75 | (4) | **5.00** | (1) | 21.90 | (3) | 15.64 | (2) |
| Ionosphere | 13.87 | (4) | **3.00** | (1) | 7.20 | (2) | 7.73 | (3) |
| Wine | 5.30 | (3) | 5.00 | (2) | 5.48 | (4) | **4.71** | (1) |
| Sonar | 14.45 | (4) | 10.00 | (2) | 10.24 | (3) | **6.13** | (1) |
| Vehicle | 69.50 | (3) | 80.00 | (4) | 56.74 | (2) | **44.25** | (1) |
| Liver-disorders | 25.51 | (4) | **3.00** | (1) | 22.65 | (3) | 6.60 | (2) |
| Page-blocks | 42.91 | (4) | **22.00** | (1) | 38.70 | (3) | 24.56 | (2) |
| Blood-t | **6.50** | (1) | 10.00 | (3) | 22.39 | (4) | 8.46 | (2) |
| Breast-tissue-6 | 22.45 | (4) | **8.00** | (1) | 14.09 | (3) | 8.97 | (2) |
| Movement-libras | 47.52 | (4) | 30.00 | (3) | **27.46** | (1) | 29.07 | (2) |
| Parkinsons | 10.24 | (4) | 7.00 | (2) | 7.11 | (3) | **4.85** | (1) |
| Seeds | 7.42 | (4) | 6.00 | (3) | 4.78 | (2) | **3.17** | (1) |
| Segment | 41.21 | (4) | 41.00 | (3) | 30.53 | (2) | **27.91** | (1) |
| Ecoli | 18.59 | (4) | 15.00 | (3) | 12.57 | (2) | **7.06** | (1) |
| Spambase | 103.37 | (4) | 75.00 | (3) | 74.42 | (2) | **31.70** | (1) |
| **Average ranking** | | 3.65 | | 2.15 | | 2.65 | | 1.55 |

**Table 4.** Average DT sizes obtained by the DTI methods.

find all the possible hypotheses which cannot be rejected. **Table 3** shows both the average rank (AR) of the results yielded by each method and the $p$-values computed by comparing the average accuracies achieved by the DE-based procedures versus those obtained by the other DTI methods. The $p$-values highlighted with bold numbers indicate that $H_0$ is rejected for this pair of methods since they show different performance. Unadjusted $p$-values are calculated with the average ranks of the two methods being compared, as is described by Demšar in [68]. These values are used by the BH post hoc test to compute the corresponding adjusted $p$-values. **Table 3** shows that the DE-ODT method has a better performance than the other DTI methods since it has the lowest average rank (1.750), and its results are statistically different than these methods. **Figure 10** shows a graph where the nodes represent the compared methods and the edges joining two nodes indicate that the performance of these methods does not present significant differences. The values shown in the edges are the $p$-values computed by the BH post hoc test. This figure is based on that obtained using the *scmamp* library, and in it is observed that the DE-based methods are not statistically different between them, and that the DE-ODT method is statistically different with the DTI methods. This statistical results indicate that the DE-ODT method is the better DTI method to build oblique DT.

On the other hand, the average sizes of the DTs constructed by the DE-based algorithms and also of those induced by the J48 and the sCART methods are shown in **Table 4** and **Figure 11**. Similar to **Table 2**, the best result for each dataset in **Table 4** is highlighted with bold numbers, and the numbers in parentheses refer to the ranking reached by each method for each dataset. These results indicate that the DE-ODT method produces the most compact DTs. Also, it is observed that the size of the DTs built for the OC1-DE method has less complexity than those yielded by the J48 method.

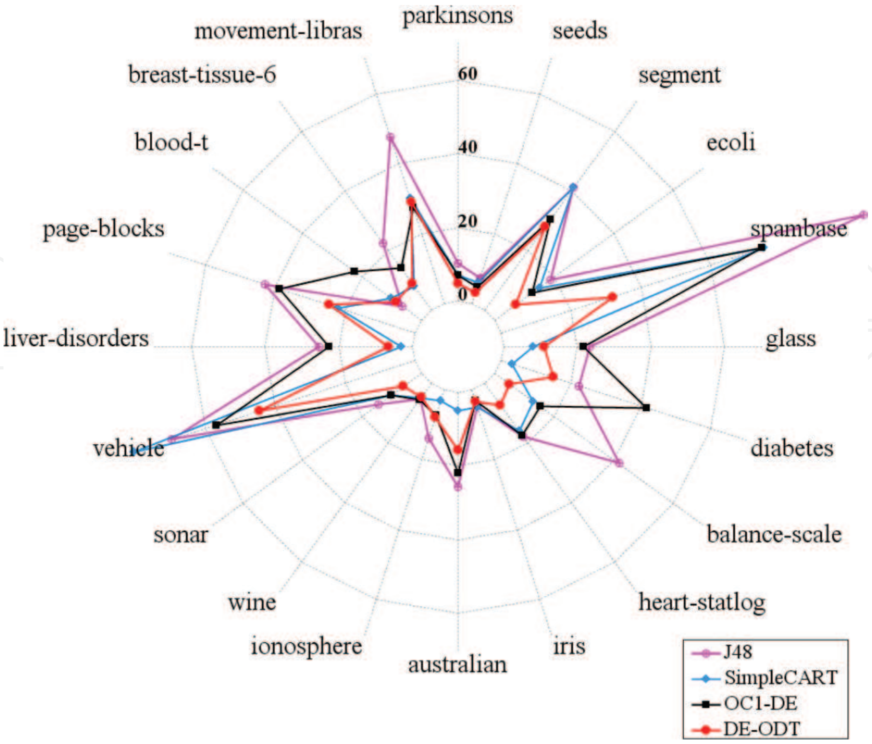

**Figure 11.** Average DT sizes of several DTI methods.

| Dataset | NB | | MLP | | RBF-NN | | RF | | OC1-DE | | DE-ODT | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Glass | 49.44 | (6) | 67.29 | (4) | 65.09 | (5) | **79.95** | (1) | 71.31 | (2) | 68.97 | (3) |
| Diabetes | 75.76 | (3) | 74.75 | (4) | 74.04 | (5) | **76.18** | (1) | 73.37 | (6) | 75.79 | (2) |
| Balance-scale | 90.53 | (4) | 90.69 | (3) | 86.34 | (5) | 81.71 | (6) | **93.92** | (1) | 91.97 | (2) |
| Heart-statlog | **83.59** | (1) | 79.41 | (5) | 83.11 | (2) | 82.41 | (3) | 74.11 | (6) | 81.11 | (4) |
| Iris | 95.53 | (5) | 96.93 | (2) | 96.00 | (4) | 94.73 | (6) | 96.73 | (3) | **97.17** | (1) |
| Australian | 77.19 | (6) | 83.42 | (4) | 82.55 | (5) | **86.77** | (1) | 85.19 | (3) | 85.61 | (2) |
| Ionosphere | 82.17 | (6) | 91.05 | (5) | 91.71 | (3) | **93.39** | (1) | 91.11 | (4) | 92.28 | (2) |
| Wine | 97.47 | (4) | **98.03** | (1.5) | 97.70 | (3) | **98.03** | (1.5) | 92.58 | (5) | 91.88 | (6) |
| Sonar | 67.69 | (6) | 81.59 | (2) | 72.60 | (5) | **84.47** | (1) | 77.65 | (4) | 79.34 | (3) |
| Vehicle | 44.68 | (6) | **81.11** | (1) | 65.35 | (5) | 75.14 | (2) | 72.32 | (3) | 71.33 | (4) |
| Liver-disorders | 54.87 | (6) | 68.72 | (3) | 65.04 | (5) | **72.99** | (1) | 67.63 | (4) | 71.16 | (2) |
| Page-blocks | 90.01 | (6) | 96.28 | (4) | 94.91 | (5) | **97.54** | (1) | 96.88 | (3) | 97.07 | (2) |
| Blood-t | 75.28 | (5) | 78.46 | (2) | 78.22 | (3) | 73.62 | (6) | 76.35 | (4) | **78.70** | (1) |
| Breast-tissue-6 | **46.42** | (1) | 35.47 | (5) | 41.13 | (3) | 45.19 | (2) | 34.91 | (6) | 38.85 | (4) |
| Movement-libras | 64.14 | (5) | 80.50 | (2) | 75.50 | (3) | **82.89** | (1) | 75.11 | (4) | 55.63 | (6) |
| Parkinsons | 70.10 | (6) | **91.44** | (1) | 81.49 | (5) | 91.38 | (2) | 87.95 | (3) | 86.43 | (4) |
| Seeds | 90.52 | (6) | **95.24** | (1) | 91.67 | (5) | 93.57 | (3) | 93.76 | (2) | 91.79 | (4) |
| Segment | 80.17 | (6) | 96.21 | (2) | 87.31 | (5) | **98.07** | (1) | 95.93 | (3) | 94.78 | (4) |
| Ecoli | 85.51 | (2) | 84.85 | (3) | 83.30 | (6) | **86.25** | (1) | 83.51 | (5) | 84.72 | (4) |
| Spambase | 79.56 | (6) | 91.19 | (4) | 81.31 | (5) | **95.65** | (1) | 92.19 | (3) | 93.94 | (2) |
| **Average ranking** | | 4.800 | | 2.925 | | 4.350 | | 2.125 | | 3.700 | | 3.100 |

**Table 5.** Average accuracies obtained by several classification methods.

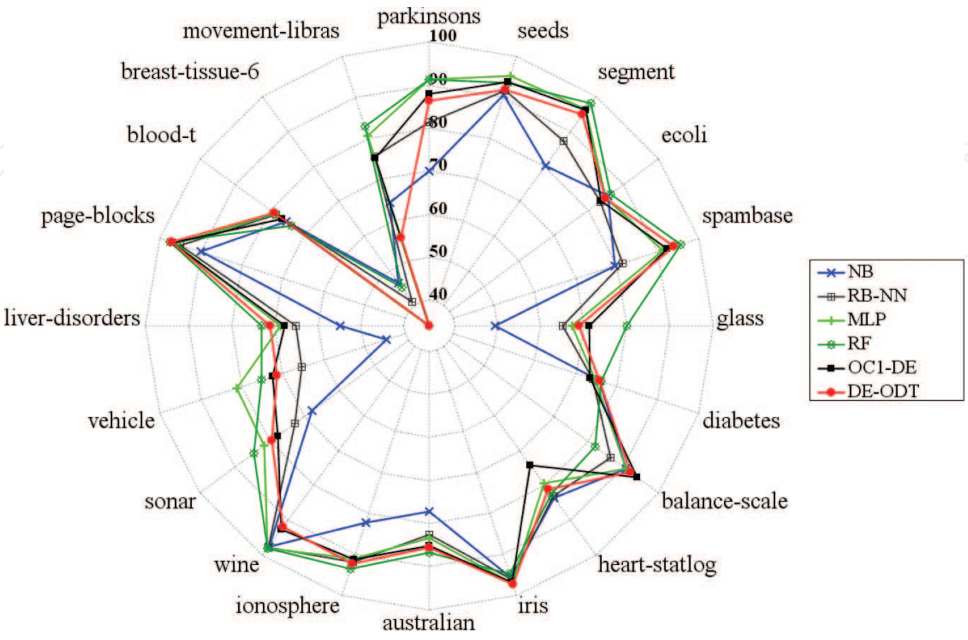

**Figure 12.** Graphical comparison of the average accuracies obtained by several classification methods.

| Method | AR | OC1-DE | | DE-ODT | |
|---|---|---|---|---|---|
| | | Unadjusted | BH | Unadjusted | BH |
| NB | 4.800 | 6.2979e-02 | 3.7787e-01 | 4.0591e-03 | **2.8414e-02** |
| MLP | 2.925 | 1.9019e-01 | 7.6079e-01 | 7.6737e-01 | 8.9374e-01 |
| RBF-NN | 4.350 | 2.7118e-01 | 7.7679e-01 | 3.4610e-03 | 1.3844e-01 |
| RF | 2.125 | 7.7623e-03 | 5.4336e-03 | 0.9342e-02 | 3.9736e-01 |
| OC1-DE | 3.700 | — | — | 3.1049e-01 | 7.6079e-01 |
| DE-ODT | 3.100 | 3.1049e-01 | 7.6079e-01 | — | — |

**Table 6.** The $p$-values for multiple comparisons among several classification methods.

### 5.3. Comparison with other classification methods

**Table 5** and **Figure 12** show the average accuracies got by several classification methods as well as those obtained by the DE-based methods. In this table, we can observe that the RF algorithm and the MLP method construct more accurate classifiers than the others, and also that the DE-based procedures induce DTs with better accuracy than the models built by both the RBF-NN algorithm and the NB method.

The Friedman statistics computed by analyzing the results got by these six methods with 20 datasets is 27.661, and the corresponding $p$-value is $4.24 \times 10^{-5}$ so that $H_0$ is rejected. The BH post hoc test is then applied to find all possible hypotheses that cannot be refused. **Table 6** shows the results of these tests, and **Figure 13** shows the graph corresponding to these $p$-values. The value highlighted with bold in **Table 6** indicates that the DE-ODT method is statistically different with the NB method, only.

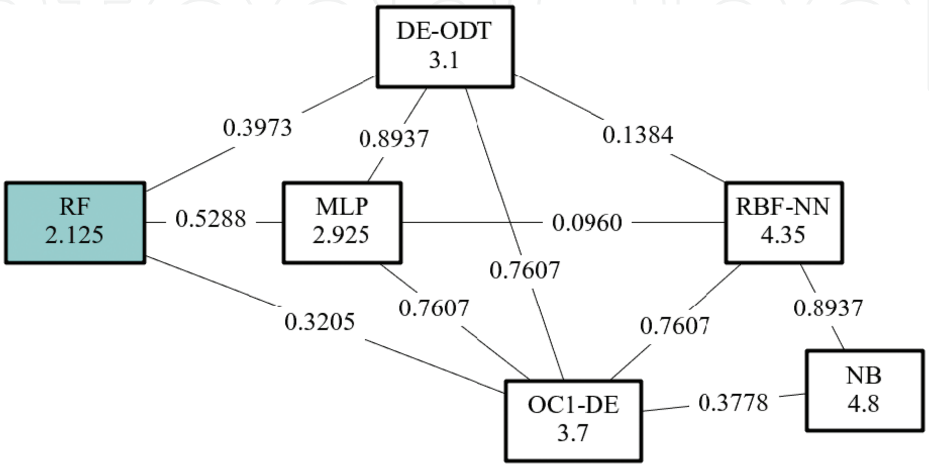

**Figure 13.** The $p$-value graph of the classification methods.

The *p*-values obtained by the BH post hoc test point out that the RF method is statistically different only with the RBF-NN algorithm and the NB method, and that both the MLP method and the DE-ODT procedure are statistically different with the NB method. The comparison between the remaining pairs of algorithms indicates that they have a similar performance. The RF method is the best ranked in this comparison, and the AR of the DE-ODT procedure places it as the third best classification method.

## 6. Conclusions

In this chapter, two DE-based methods to induce oblique DTs are described. The OC1-DE method implements a recursive partitioning strategy to find a near-optimal hyperplane which is used as test condition of an oblique DT. On the other hand, in the DE-ODT method, a global search in the space of oblique DTs is conducted with the aim of finding a near-optimal tree. The DE-ODT method estimates the size of the chromosome encoding a complete tree based on both the number of attributes and the number of classes of the dataset whose model is constructed. This method also defines a scheme to map feasible oblique DTs from this chromosome.

The experimental results obtained indicate that these DE-based methods are better DTI methods, since they build more accurate and compact oblique DTs than those induced by the J48 and the sCART procedures. The DE-ODT method is better than the OC1-DE since its search procedure uses intelligent search procedures combining their exploration and exploitation skills, thus providing a better way to discover the relationships between the attributes used in the training set, and although the search process is only guided by the accuracy of the DT, the models constructed are more compact than those produced by the methods that implement a recursive partitioning strategy. Among the other compared methods, the results got by the OC1-DE method are better than those obtained by the other methods, since it uses a linear combination of attributes in each test condition of the tree, and it produce better hyperplanes than the axis-parallel hyperplanes.

Even though the results yielded by the DE-based variants are not better than those produced by the RF algorithm and the MLP-based classifier, they are statistically equivalent. An advantage of these methods is that it constructs models whose decisions and operations are easily understood, and although the RF method also builds DTs, its voting scheme makes it very difficult to trace the way in which the model takes its decisions.

In this chapter, an analysis of the run time of the algorithms is not performed, since it is known that MHs consume more computational time than other approaches because they work with a group of candidate solutions, unlike the traditional methods where only one DT is induced from the training set. It is important to mention that for many practical applications, the construction of the model is conducted in one offline procedure, so the time of its construction is not a parameter that usually impacts the efficiency of the built model.

## Author details

Rafael Rivera-Lopez[1] and Juana Canul-Reich[2]*

*Address all correspondence to: juana.canul@ujat.mx

1 Departamento de Sistemas y Computación, Instituto Tecnológico de Veracruz, Veracruz, VER, México

2 División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, Cunduacán, TAB, México

## References

[1] Frawley WJ, Piatetsky-Shapiro G, Matheus CJ. Knowledge discovery in databases: An overview. AI Magazine. 1992;**13**(3):57

[2] Stiglic G, Kocbek S, Pernek I, Kokol P. Comprehensive decision tree models in bioinformatics. PLoS One. 2012;**7**(3):1-13

[3] Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decision Support Systems. 2011;**51**(1):141-154

[4] Nettleton DF, Orriols-Puig A, Fornells A. A study of the effect of different types of noise on the precision of supervised learning techniques. Artificial Intelligence Review. 2010;**33**(4): 275-306

[5] Du KL, Swamy MNS. Search and Optimization by Metaheuristics. Switzerland: Springer; 2016

[6] Storn R, Price K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997;**11**(4):341-359

[7] Galea M, Shen Q, Levine J. Evolutionary approaches to fuzzy modelling for classification. The Knowledge Engineering Review. 2004;**19**(1):27-59

[8] Espejo PG, Ventura S, Herrera F. A survey on the application of genetic programming to classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2010;**40**(2):121-144

[9] Kokol P, Pohorec S, Štiglic G, Podgorelec V. Evolutionary design of decision trees for medical application. Data Mining and Knowledge Discovery. 2012;**2**(3):237-254

[10] Barros RC, Basgalupp MP, Carvalho ACPLF, Freitas AA. A survey of evolutionary algorithms for decision-tree induction. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2012;**42**(3):291-312

[11] Kolçe E, Frasheri N. The use of heuristics in decision tree learning optimization. International Journal of Computer Engineering in Research Trends. 2014;**1**(3):127-130

[12] Alurkar AA, Ranade SB, Joshi SV, Ranade SS, Sonewar PA, Mahalle PN, Deshpande AV. A proposed data science approach for email spam classification using machine learning techniques. In: 2017 Internet of Things Business Models, Users, and Networks; November 2017; pp. 1-5

[13] Mishan MT, Kushan AL, Fadzil AFA, Amir ALB, Anuar NB. An analysis on business intelligence predicting business profitability model using naive Bayes neural network algorithm. In: 2017 7th IEEE International Conference on System Engineering and Technology (ICSET). Shah Alam, Malasya: IEEE; 2017; pp. 59-64

[14] Lippmann RP. An introduction to computing with neural nets. ASSP Magazine. 1987;**4**(2): 4-22

[15] Abe S. Support Vector Machines for Pattern Classification. London, UK: Springer; 2005

[16] Murthy SK. On Growing Better Decision Trees from Data. PhD thesis, The Johns Hopkins University; 1997

[17] Fisher RA. The use of multiple measurements in taxonomic problems. Annals of Eugenics. 1936;**7**(2):179-188

[18] Witten IH, Frank E. Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, CA, USA: Morgan Kaufmann; 2005

[19] Heath DG. A Geometric Framework for Machine Learning [PhD thesis]. Johns Hopkins University; 1993

[20] Birattari M. Tuning Metaheuristics: A Machine Learning Perspective, Volume 197 of Studies in Computational Intelligence. Berlin Heildelberg: Springer; 2009

[21] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Information Sciences. 2013;**237**:82-117

[22] Talbi EG. Metaheuristics: From Design to Implementation. Hoboken, NY, USA: Wiley; 2006

[23] Li J, Ding L, Li B. Differential evolution-based parameters optimisation and feature selection for support vector machine. International Journal of Computational Science and Engineering. 2016;**13**(4):355-363

[24] Leema N, Nehemiah HK, Kannan A. Neural network classifier optimization using differential evolution with global information and back propagation algorithm for clinical datasets. Applied Soft Computing. 2016;**49**:834-844

[25] Geetha K, Baboo SS. An empirical model for thyroid disease classification using evolutionary multivariate Bayesian prediction method. Global Journal of Computer Science and Technology. 2016;**16**(1):1-9

[26] García S, Derrac J, Triguero I, Carmona CJ, Herrera F. Evolutionary-based selection of generalized instances for imbalanced classification. Knowledge-Based Systems. 2012; **25**(1):3-12

[27] Tušar T. Optimizing accuracy and size of decision trees. In: Proceedings of the 16th International Electrotechnical and Computer Science Conference (ERK-2007), Portorož; Slovenia, 2007; pp. 81-84

[28] Neri F, Tirronen V. Recent advances in differential evolution: A survey and experimental analysis. Artificial Intelligence Review. 2010;**33**(1–2):61-106

[29] Heath DG, Kasif S, Salzberg S. Induction of oblique decision trees. In: Bajcsy R, editor. Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93); Chambéry; France, 1993. pp. 1002-1007

[30] Murthy SK, Kasif S, Salzberg S, Beigel R. OC1: A randomized algorithm for building oblique decision trees. In: AAAI'93. Vol 93. AAAI press; 1993. pp. 322-327

[31] Cantú-Paz E, Kamath C. Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation. 2003;**7**(1):54-68

[32] Li XB, Sweigart JR, Teng JTC, Donohue JM, Thombs L, Wang SM. Multivariate decision trees using linear discriminants and tabu search. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans. 2003;**33**(2):194-205

[33] Orsenigo C, Vercellis C. Discrete support vector decision trees via tabu search. Computational Statistics & Data Analysis. 2004;**47**(2):311-322

[34] Zhang K, Xu Z, Buckles BP. Oblique decision tree induction using multimembered evolution strategies. In: Dasarathy BV, editor. Proceeding of Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, SPIE 2005. Vol. 5812. Orlando:, Florida SPIE; 2005. pp. 263-270

[35] Chai BB, Zhuang X, Zhao Y, Sklansky J. Binary linear decision tree with genetic algorithm. In: Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96). Track D: Parallel and Connectionist Systems. Vol. IV. Vienna: IEEE; 1996. pp. 530-534

[36] Struharik R, Vranjkovic V, Dautovic S, Novak L. Inducing oblique decision trees. In: Proceedings of the 12th International Symposium on Intelligent Systems and Informatics (SISY–2014), Subotica, Serbia: IEEE; 2014. pp. 257-262

[37] Krętowski M. An evolutionary algorithm for oblique decision tree induction. In: Rutkowski L, Siekmann J, Tadeusiewicz R, Zadeh LA, editors. Proceedings of the 7th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2004). LNAI. Vol. 3070. Zakopane, Poland: Springer; 2004. pp. 432-437

[38] Pangilinan JM, Janssens GK. Pareto-optimality of oblique decision trees from evolutionary algorithms. Journal of Global Optimization. 2011;**51**(2):301-311

[39] Rivera-Lopez R, Canul-Reich J, Gámez JA, Puerta JM. OC1-DE: A differential evolution based approach for inducing oblique decision trees. In: Rutkowski L, Korytkowski M, Scherer R, Tadeusiewicz R, Zadeh LA, Zurada JM, editors. Proceedings of the 16th International Conference in Artificial Intelligence and Soft Computing (ICAISC 2017). LNCS. Vol 10245. Zakopane, Poland: Springer; 2017. pp. 427-438

[40] Dumitrescu D, András J. Generalized decision trees built with evolutionary techniques. Studies in Informatics and Control. 2005;**14**(1):15-22

[41] Vukobratovic B, Struharik R. Evolving full oblique decision trees. In: Proceedings of the 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2015). Budapest, Hungary: IEEE; 2015. pp 95-100

[42] Krętowski M, Grześ M. Global induction of oblique decision trees: An evolutionary approach. In: Kłopotek MA et al editors. IIPWM'05, Volume 31 of ASC. Berlin Heildelberg: Springer; 2005. pp. 309-318

[43] Krętowski M, Grześ M. Evolutionary learning of linear trees with embedded feature selection. In: Rutkowski L et al editors. ICAISC 2006. LNAI. Volume 4029 of LNAI-Springer; 2006. pp. 400-409

[44] Gray JB, Fan G. Classification tree analysis using TARGET. Computational Statistics & Data Analysis. 2008;**52**(3):1362-1372

[45] Liu KH, Xu CG. A genetic programming-based approach to the classification of multiclass microarray datasets. Bioinformatics. 2009;**25**(3):331-337

[46] Bot MCJ, Langdon WB. Application of genetic programming to induction of linear classification trees. In: Poli R et al., editors. EuroGP 2000. LNCS. Vol. 1802. Berlin Heildelberg: Springer; 2000. pp. 247-258

[47] Bot MCJ, Langdon WB. Improving induction of linear classification trees with genetic programming. In: Whitley LD et al., editors. GECCO-2000. San Francisco, CA, USA: Morgan Kaufmann; 2000. pp. 403-410

[48] Agapitos A, O'Neill M, Brabazon A, Theodoridis T. Maximum margin decision surfaces for increased generalisation in evolutionary decision tree learning. In: Silva S et al editors. EuroGP 2011. LNCS. Volume 6621. Berlin Heildelberg: Springer; 2011. pp. 61-72

[49] Lopes RA, Freitas ARR, Silva RCP, Guimarães FG. Differential evolution and perceptron decision trees for classification tasks. In: Yin H, Costa JAF, Barreto G, editors. Proceedings of the 13th International Conference Intelligent Data Engineering and Automated Learning (IDEAL 2012). LNCS. Vol. 7435. Natal, Brazil: Springer; 2012. pp. 550-557

[50] Freitas ARR, Silva RCP, Guimarães FG. Differential evolution and perceptron decision trees for fault detection in power transformers. In: Snášel V et al, editors. SOCO Models in Industrial & Environmental Applications. AISC. Volume 188. Berlin Heildelberg: Springer; 2013. pp. 143-152

[51] Rivera-Lopez R, Canul-Reich J. A global search approach for inducing oblique decision trees using differential evolution. In: Mouhoub M, Langlais P, editors. Proceedings of the 30th Canadian Conference on Artificial Intelligence (AI 2017), volume 10233 of LNCS. Edmonton, Canada: Springer; 2017. pp. 27-38

[52] Quinlan JR. Induction of decision trees. Machine Learning. 1986;**1**(1):81-106

[53] Murthy SK, Kasif S, Salzberg S. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research. 1994;**2**(1):1-32

[54] Breiman L, Friedman J, Olshen R, Stone C. Classification and Regression Trees. Boca Raton, FL, USA: Chapman and Hall; 1984

[55] Lichman M. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences; 2013

[56] Durillo JJ, Nebro AJ. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software. 2011;**42**(10):760-771

[57] Das S, Konar A, Chakraborty UK. Two improved differential evolution schemes for faster global search. In: Beyer HG, editor. Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05), Washington, DC, USA: ACM; 2005. pp. 991-998

[58] Quinlan JR. C4.5: Programs for Machine Learning. San Mateo, CA, USA: Morgan Kaufmann; 1993

[59] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association. 1937;**32**(200):675-701

[60] Hommel G. A stagewise rejective multiple test procedure based on a modified Bonferroni test. Biometrika. 1988;**75**(2):383-386

[61] Calvo B, Guzmán-Santafé R. scmamp: Statistical comparison of multiple algorithms in multiple problems. The R Journal. 2016;**8**(1):248-256

[62] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: An update. SIGKDD Explorations Newsletter. 2009;**11**(1):10-18

[63] Witten IH, Frank E, Trigg LE, Hall MA, Holmes G, Weka SJC. Practical machine learning tools and techniques with Java implementations. Technical Report 11, Department of Computer Science. New Zeland: Waikato; 1999

[64] John GH Langley P. Estimating Continuous Distributions in Bayesian Classifiers. In: Besnard P, Hanks S, editors. Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95), San Francisco, CA, USA: Morgan Kaufmann; 1995. pp. 338-345

[65] Murtagh F. Multilayer perceptrons for classification and regression. Neurocomputing. 1991;**2**(5):183-197

[66] Frank E. Fully supervised training of Gaussian radial basis function networks in WEKA. Technical Report 04, Department of Computer Science. New Zeland: Waikato; 2014

[67] Breiman L. Random forests. Machine Learning. 2001;**45**(1):5-32

[68] Demšar J. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research. 2006;**7**:1-30

IntechOpen