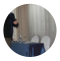
Optimal Randomized Classification Trees



Optimal Randomized Classification Trees

Rafael Blanquero
Instituto de Matemáticas de la Universidad de Sevilla (IMUS)
rblanquero@us.es

Emilio Carrizosa
Instituto de Matemáticas de la Universidad de Sevilla (IMUS)
ecarrizosa@us.es

Cristina Molero-Río *
Instituto de Matemáticas de la Universidad de Sevilla (IMUS)
mmolero@us.es

Dolores Romero Morales
Copenhagen Business School (CBS)
drm.eco@cbs.dk

August 8, 2018

Abstract

Classification and Regression Trees (CARTs) are an off-the-shelf technique in modern Statistics and Machine Learning. CARTs are traditionally built by means of a greedy procedure, sequentially deciding the splitting predictor variable(s) and the associated threshold. This greedy approach trains trees very fast, but, by its nature, the classification accuracy may suffer. Moreover, controlling critical issues, such as the misclassification rates in each of the classes, is difficult. To address these shortcomings, we propose an enhanced version of CART whose parameters are obtained by solving a continuous optimization problem. Our classifier can be seen as a randomized tree, since at each node of the decision tree a random decision is made. The computational experience reported shows the outperformance of our approach not only against CART, but also against recent proposals based on Integer Programming.

Key words Classification and Regression Trees; Cost-sensitive Classification; Nonlinear Programming

*Corresponding author

1 Introduction

Extracting knowledge from data is a crucial task in Statistics and Machine Learning, which has applications in areas such as Biomedicine [29, 41], Criminal Justice [30, 32, 55], Fraud Detection [49], Privacy Protection [33], Health Care [6, 15, 45, 48], Risk Management [1, 36], Social Networks [19], and Computational Optimization [31]. Mathematical Optimization plays an important role in building such models and interpreting their output, see, e.g., [4, 5, 7, 9, 12, 13, 14, 17, 20, 39].

Decision trees constitute a set of methods which can be considered as one of the most powerful tools for Classification and Regression in Statistics and Machine Learning [27]. They are popular because they are rule-based and, when they are not very deep, deemed to be easy-to-interpret, see [1, 21, 23, 30, 43, 48].

Since constructing optimal binary decision trees is known to be an NP-complete problem [28], researchers have traditionally focused on the design of heuristic procedures. CART [11] is a popular algorithm for growing decision trees which, due to the complexity and the available computer technology at the time it was introduced, is based on a simple greedy and sequential partitioning procedure. In addition, CART, like many other tree algorithms [54], uses orthogonal cuts, that is, each branching rule involves one single predictor variable. These characteristics of CART are aimed to achieve a low computational cost, but at the expense of accuracy. One of the most popular extension of CARTs is Random Forests (RFs) [8, 10, 18, 22], which bag unpruned decision trees. In this way, better accuracies are typically obtained, see e.g. [18], but having a complex, hardly interpretable neither manageable, decision structure. Nevertheless, there are ways to maintain the tree-like structure while improving accuracy. This includes combining several predictor variables in each split (oblique cuts), as well as choosing the splits globally along the tree, and not sequentially. These two strategies are at the core of this research.

Oblique cuts are more versatile than orthogonal cuts and tend to generate smaller trees, since several orthogonal cuts may be reduced to one single oblique cut. However, by their nature, oblique trees are harder to interpret and computationally more expensive than plain CARTs. While orthogonal cuts are easily chosen by an enumerative process, several proposals based on Mathematical Optimization are found in the literature to build oblique cuts. In some cases the oblique cuts are obtained as the output of a baseline classification procedure, such as a Support Vector Machine classifier [40] or a Logistic Regression classifier [47]. A different perspective dealing with oblique splits can be found in [53], where the authors propose the so-called oblique treed sparse additive models. These are tree-structured probabilistic mixture of expert models, which are tackled using factorized asymptotic Bayesian inference, through EM-like iterative optimization.

Due to the NP-completeness of the problem and the appeal of small trees to ease interpretability, optimal trees are grown up to a given maximum depth level. Two bottom-up approaches for finding optimal trees with the smallest classification error within the class of all possible trees of a specified depth are outlined in [44], being exponentially complex in the number of predictor variables. In [2, 38], the construction of optimal oblique trees in one step (non-greedily) is considered. In [2], the topology of the tree is fixed, including the assignment of classes to leaf nodes, and represents it as a set of disjunctive linear inequalities. This yields a non-linear non-convex continuous optimization problem over a polyhedral region. Since the local search (performed by Frank-Wolfe algorithm) may get stuck at local optima, a heuristic search approach, the so-called Extreme Points Tabu Search, is also proposed. Standard greedily built trees are outperformed by this approach for medium-size two-class problems. A method for non-greedy learning of oblique trees is developed in [38], in which a convex-concave upper bound on the tree's empirical loss is optimized using Stochastic Gradient Descent, enabling effective training with larger data sets.

Some attempts on bi-criteria optimization of decision trees have been proposed recently. For instance, [16] describes algorithms which allow to construct the set of Pareto optimal points for a given decision table and different objective functions, modeling the size of the tree, by controlling the depth and number of nodes, among others. One of its limitations is that it only handles categorical predictor variables and works efficiently only with medium-size decision tables. Besides dealing with optimal decision trees, optimization of tree ensembles has also been carried out recently. In [37], a mixed-integer formulation models the performance of a collection of trees in order to obtain the optimal prediction of the tree ensemble model. They show that this formulation can be solved to optimality for small to medium-size instances within minutes. For tackling large-scale instances, two different algorithms are presented: the former is based on solving a Benders reformulation of the problem using constraint generation and the latter, on applying lazy constraint generation directly to the formulation, which can drastically reduce the computing time.

Constructing an optimal decision tree involves, in principle, many discrete decisions, such as deciding whether splitting a branch node or indicating the path in the tree from the root node to the leaf node that each observation will follow. This is the approach followed in recent papers such as [3, 24, 50, 51], which consider mixed-integer formulations. Our approach instead considers a continuous optimization formulation, yielding the so-called Optimal Randomized Classification Tree (ORCT). ORCT is a randomized optimal version of CART since a random decision is made at each non-terminal node of the tree with a certain probability. By saving these probabilities along the tree, probabilistic information for each individual is provided. There are other worth mentioning similarities/differences between our method and those mentioned above. First, in terms of predictor variables, and as in [3], we deal with any type of predictor variables, linearly scaling to the 0-1 interval and creating dummies when needed. In [50, 51], they propose a non-linear scaling that assigns a unique integer to each value of each predictor variable, maintaining the original order of those values. Among others, one of the benefits of doing this scaling is that thresholds in branch nodes will be also represented by integers and therefore mixed-integer programming solvers will find it easier to branch on these values. The approach in [24] focuses on categorical predictor variables by exploiting the resulting combinatorial structure of considering every possible subset of categories of a given predictor variable. However, when there is a predictor variable with many categories, the number of resulting 0-1 predictor variables exploits. Second, in terms of type of trees, orthogonal ones are grown in [3, 24]. The latter also constructs oblique trees, as we do. In [50, 51], the authors construct oblique trees where the 0–1 coefficients in orthogonal cuts are generalized to integers. Third, in terms of the number of classes, multiclass problems can be handled with any approach, except for [24], which is restricted to two classes. Fourth, in terms of the Machine Learning task, Regression problems are also addressed by [50, 51]. Fifth, in relation to the topology of the tree, [3, 50, 51] specify its depth, and penalize the size of the tree in the objective function so that trees may be smaller than the pre-established depth. In [24], the authors specify a priori every leaf node in the tree and, as in our approach, a maximal tree for a given depth is constructed. Finally, we also provide, as in [24, 50, 51], the flexibility we can borrow from Mathematical Optimization [14], by controlling the correct classification rate of a given class.

A serious drawback of existing optimization-based procedures to build CARTs is the computing times required. This is particularly critical in Integer Programming-based strategies, for which the running times may dramatically increase with the data dimensionality. For this reason it is commonly assumed that the training sets are small, e.g., $\min\{0.9n, 600\}$ observations, n being the sample size, [24]. Moreover, a CPU time limit is imposed to the solver, e.g. 30 minutes per run, [50, 51, 24], while [3] uses from 30 minutes to 2 hours for orthogonal trees and from 5 to 15 minutes for oblique trees. Instead, using, as we do, a continuous-optimization based method, allows us to perform local search, yielding, as reported in this paper, better results in a more reasonable time.

In summary, the main contributions of this paper are outlined below:

1. A novel continuous-based approach for building optimal classification trees is provided. The inclusion of oblique cuts supplies more versatility to the model and allows to remove integer decision variables from the optimization scenario.

2. The numerical results in this paper illustrate the effectiveness of our approach: a better performance in terms of accuracy is generally achieved within a much lower running time than recent proposals.

3. Probabilities outputs, i.e., class membership probability estimators for each individual in the sample, are obtained. This enhanced prediction is not provided by greedy approaches like CART, neither by the recent mathematical optimization-based approaches proposed in the literature.

4. Preferences on classification rates in some classes are successfully handled, unlike heuristic approaches like CARTs or RFs which can not address this issue explicitly.

The remainder of the paper is organized as follows. In Section 2, we detail the construction of an ORCT. In Section 3, we introduce an extension of OR-

CTs, which allows us to control the classification performance in those classes where misclassification errors are more critical. In Section 4, computational experiments with ORCTs are reported. The results obtained are compared with CART, the Integer-Programming based approach in [3], and RFs. Finally, conclusions and possible lines of future research are provided in Section 5.

2 Optimal Randomized Classification Trees

Suppose we are given a training sample of n individuals, $I = \{(\boldsymbol{x}_i, y_i)\}_{1 \leq i \leq n}$, on which p predictor variables are measured, $\boldsymbol{x}_i \in \mathbb{R}^p$, and one class label is associated with each one, $y_i \in \{1, \ldots, K\}$. For simplicity, numerical predictor variables are considered. Without loss of generality, we assume that $\boldsymbol{x}_i \in [0,1]^p$.

Classic decision tree methods consist of sequentially, and greedily, partitioning the predictor space $[0,1]^p$ into disjoint sets or nodes by imposing certain conditions on the predictor variables. The usual splitting criterion is to take the split that makes descendant nodes purer, i.e., nodes with observations more and more homogenous in terms of class membership. The process of partitioning finishes when a stopping criterion is satisfied. Then, leaf nodes are labeled with a class label, $1, \ldots, K$. Commonly, a leaf node is labelled with the most frequent class in the individuals that have fallen into the node. Once the tree is built, the prediction of future unlabeled data is done in a deterministic way. Given a new observation $\boldsymbol{x}$, starting from the root node, it will end up in a leaf node, depending on the values the predictor variables take, and its predicted class will be the class label of that leaf node.

The approach proposed here is different: prediction is randomized. At each node of an Optimal Randomized Classification Tree (ORCT), a random variable will be generated to indicate by which branch one has to continue. Since we are building binary trees, the Bernoulli distribution is appropriate, whose probability of success will be determined by the value of a cumulative density function (CDF), evaluated over the vector of predictor variables. In this way, each leaf node will not contain a subset of individuals but all the individuals in the training sample, for which the probability of falling into such leaf node is known. Finally, the class label of each leaf node will be a decision variable, which will be found by minimizing the expected misclassification cost over the whole training sample.

The distinctive element in our approach is the fact that the yes/no rule in decision tree methods is replaced by a soft rule, induced by a CDF. In this way, a more stable rule is obtained. Indeed, suppose that the first cut of a classic classification tree forces individuals with $X \leq b$ to go down the tree by the left branch. Then, for an incoming individual with $X = b + \varepsilon$, $\varepsilon > 0$ sufficiently small, classic decision tree methods would give them probability of going down the branch $X > b$ equal to 1 and 0, otherwise, as in the orange line in Figure 1. Smoothing the probabilities of going left or right in a neighbourhood of b can be achieved through a CDF, see the green line in Figure 1.

After having introduced how ORCTs work, we will next formulate the prob-

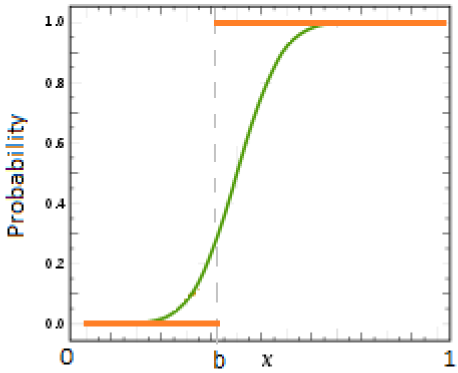


Figure 1: The probability of an individual going down by the right branch is depicted for both types of trees: the classic approach (orange line) and the proposed ORCT (green line).

lem.

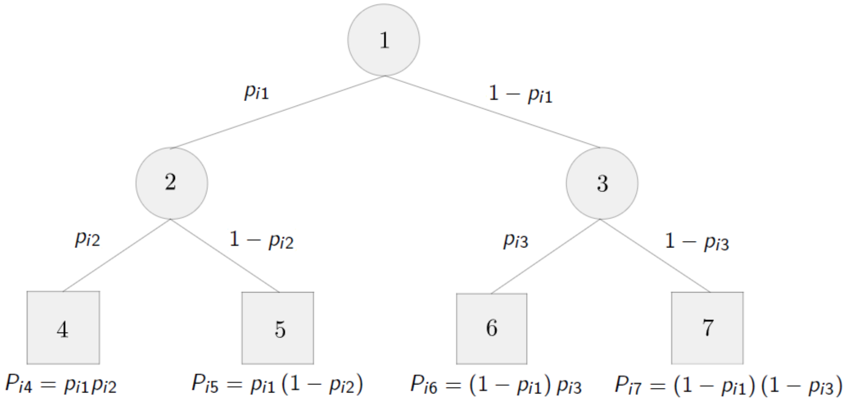

1
p_{i1}
$1-p_{i1}$
2
3
p_{i2}
$1-p_{i2}$
p_{i3}
$1-p_{i3}$
4
5
6
7
$P_{i4} = p_{i1}p_{i2}$
$P_{i5} = p_{i1}(1-p_{i2})$
$P_{i6} = (1-p_{i1})\,p_{i3}$
$P_{i7} = (1-p_{i1})(1-p_{i3})$

Figure 2: Optimal Randomized Classification Tree for depth $\boldsymbol{D = 2}$.

Our approach starts from the maximal binary tree of depth D, i.e., a binary tree in which each branch node has two children and terminal nodes all have the same depth, namely, D. For instance, Figure 2 shows the maximal binary tree of depth $D = 2$. Given D, the total number of nodes is known in advance, $T = 2^{(D+1)} - 1$. The sets of branch and leaf nodes are known and numbered as follows:

Branch nodes: nodes $t \in \tau_B = \{1, \ldots, \lfloor T/2 \rfloor\}$.

Leaf nodes: nodes $t \in \tau_L = \{\lfloor T/2 \rfloor + 1, \ldots, T\}$.

Oblique splits are modeled through linear combinations of the predictor variables. To do that that, we need to define, for each $j = 1, \ldots, p$ and each $t \in \tau_B$, the decision variables $a_{jt} \in [-1, 1]$ to indicate the value of the coefficient of predictor variable j in the oblique cut over branch node $t \in \tau_B$. The intercepts of the linear combinations correspond to decision variables $\mu_t \in [-1, 1]$, seen as the location parameter at every branch node $t \in \tau_B$.

Now, a univariate CDF $F(\cdot)$ is assumed. Then, for each individual $i = 1, \ldots, n$ at each branch node $t \in \tau_B$, the parameter of their corresponding Bernoulli distribution is obtained as follows:

$$p_{it} = F\left(\frac{1}{p}\sum_{j=1}^{p} a_{jt}x_{ij} - \mu_t\right),\ i = 1, \ldots, n,\ t \in \tau_B.$$

The value p_{it} will be used in the corresponding left branch and $1 - p_{it}$ in the right one, as seen in Figure 2. We denote as $N_L(t)$ the set of ancestor nodes of node t whose left branch takes part in the path from the root node to node t. Respectively, $N_R(t)$ is the set of ancestor nodes of node t whose right branch takes part in the path from the root node to t. If $N(t)$ denotes the set of ancestors of node t, we have that $N(t) = N_L(t) \cup N_R(t)$. For leaf node $t = 5$ in Figure 2: $N_L(5) = \{1\}$, $N_R(5) = \{2\}$ and $N(5) = \{1, 2\}$.

Once these sets are defined, we can obtain the probability of an individual falling into a given leaf node:

$$P_{it} \equiv \mathbb{P}(\boldsymbol{x}_i \in t) = \prod_{t_l \in N_L(t)} p_{it_l} \prod_{t_r \in N_R(t)} (1 - p_{it_r}),\ i = 1, \ldots, n,\ t \in \tau_L. \quad (1)$$

Now, it is necessary to define, for each leaf node $t \in \tau_L$, the decision variables $\boldsymbol{C}_t \in \{0, 1\}^K$ that model the class label assigned to each of them, where

$$C_{kt} = \left\{ \begin{array}{ll} 1, & \text{if node } t \text{ is labeled with class } k \\ 0, & \text{otherwise} \end{array} \right. ,k = 1, \ldots, K,\ t \in \tau_L.$$

We must add the following set of constraints for making a single class prediction at each leaf node:

$$\sum_{k=1}^{K} C_{kt} = 1,\ t \in \tau_L.$$

As a natural strengthening, we force each class $k = 1, \ldots, K$ to be identified by, at least, one terminal node, by adding the set of constraints below:

$$\sum_{t \in \tau_L} C_{kt} \geq 1,\ k = 1, \ldots, K.$$

It remains to describe the objective function of our model. As said before, the objective is to minimize the expected misclassification cost over the sample, so we need to introduce a misclassification cost for classifying an individual from class k in class k':

$$W_{kk'} \geq 0,\ k, k' = 1, \ldots, K,\ k \neq k'. \quad (2)$$

Thus, the objective function takes the following form:

$$\sum_{k=1}^{K} \sum_{i\in I_k} \sum_{t\in \tau_L} P_{it} \sum_{k'\neq k} W_{kk'} C_{k't}, \tag{3}$$

where $I_k,\ k = 1,\ldots,K$, is the subset of I of individuals from class k.

Thus, given the sample I split into K classes, the CDF F, the depth of the tree D and the misclassification costs $W_{kk'}$, a Mixed-Integer Non-Linear Optimization (MINLO) problem to build the proposed classification tree arises:

$$\begin{array}{ll} \min & \displaystyle\sum_{k=1}^{K} \sum_{i\in I_k} \sum_{t\in \tau_L} P_{it} \sum_{k'\neq k} W_{kk'} C_{k't} \\ \text{s.t.} & \displaystyle\sum_{k=1}^{K} C_{kt} = 1,\ t \in \tau_L, \\ & \displaystyle\sum_{t\in\tau_L} C_{kt} \geq 1,\ k = 1,\ldots,K, \\ & a_{jt} \in [-1,1],\ j = 1,\ldots,p,\ t \in \tau_B, \\ & \mu_t \in [-1,1],\ t \in \tau_B, \\ & C_{kt} \in \{0,1\},\ k = 1,\ldots,K,\ t \in \tau_L, \end{array} \tag{4}$$

where P_{it} is defined as in (1).

The presence of binary decision variables in a framework where the objective function is highly complex non-convex could appear to be discouraging. Nevertheless, without loss of optimality, we can relax the binary variables $C_{kt},\ k = 1,\ldots,K,\ t \in \tau_L$, to the $0-1$ interval, yielding to the continuous formulation we were looking for. Theorem 1 guarantees the equivalence of the resulting Non-Linear Continuous Optimization (NLCO) problem and the MINLO problem.

The NLCO problem, to which we will refer from now on as an Optimal Randomized Classification Tree (ORCT), reads as follows:

$$\begin{array}{ll} \min & \displaystyle\sum_{k=1}^{K} \sum_{i\in I_k} \sum_{t\in \tau_L} P_{it} \sum_{k'\neq k} W_{kk'} C_{k't} \\ \text{s.t.} & \displaystyle\sum_{k=1}^{K} C_{kt} = 1,\ t \in \tau_L, \\ & \displaystyle\sum_{t\in\tau_L} C_{kt} \geq 1,\ k = 1,\ldots,K, \\ & a_{jt} \in [-1,1],\ j = 1,\ldots,p,\ t \in \tau_B, \\ & \mu_t \in [-1,1],\ t \in \tau_B, \\ & C_{kt} \in [0,1],\ k = 1,\ldots,K,\ t \in \tau_L, \end{array} \tag{5}$$

where $C_{kt},\ k=1,\ldots,K,\ t\in\tau_L$, can be seen as the probability that the class label k is assigned to leaf node t.

Theorem 1 *There exists an optimal solution to* (5) *such that* $C_{kt}\in\{0,1\},\ k=1,\ldots,K,\ t\in\tau_L$.

Proof. The continuity of the objective function, defined in (5) over a compact set, ensures the existence of an optimal solution of the optimization problem, by the Weierstrass Theorem. Let $a^*_{jt},\ j=1,\ldots,p,\ t\in\tau_B,\ \mu^*_t,\ t\in\tau_B$, and $C^*_{kt},\ k=1,\ldots,K,\ t\in\tau_L$, be an optimal solution to (5). Fixed $a^*_{jt},\ j=1,\ldots,p,\ t\in\tau_B$, and $\mu^*_t,\ t\in\tau_B$, we have the following problem on the decision variables $C_{kt},\ k=1,\ldots,K,\ t\in\tau_L$:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K}\sum_{i\in I_k}\sum_{t\in\tau_L} P_{it}\sum_{k'\neq k} W_{kk'}C_{k't}\\
\text{s.t.} \quad & \sum_{k=1}^{K} C_{kt}=1,\ t\in\tau_L,\\
& \sum_{t\in\tau_L} C_{kt}\geq 1,\ k=1,\ldots,K,\\
& C_{kt}\in[0,1],\ k=1,\ldots,K,\ t\in\tau_L,
\end{aligned}
$$

which leads to a transportation problem, to which the integrality of the optimal solution is well-known to hold, i.e., there exist $\overline{C}_{kt}\in\{0,1\}$ such that $\left(a^*_{jt},\mu^*_t,\overline{C}_{kt}\right)$ is also optimal for (5). □

The NLCO formulation (5) allows us to train the ORCT classifier. Once the optimization problem (5) has been solved, ORCT predicts the class of a new unlabeled observation $\boldsymbol{x}$ as follows. The probability of $\boldsymbol{x}$ falling into each leaf node ($P_{\boldsymbol{x}t},\ t\in\tau_L$) is known from the probabilities of $\boldsymbol{x}$ falling into each branch node ($p_{\boldsymbol{x}t},\ t\in\tau_B$). Besides, given a leaf node, we know the probability of belonging to every class ($C_{kt},\ k=1,\ldots,K,\ t\in\tau_L$). Thus, we make use of the Total Probability rule and we assign to $\boldsymbol{x}$ the class for which probability is maximum:

$$\arg\max_{C_k}\left\{\sum_{t\in\tau_L}\mathbb{P}\left(\boldsymbol{x}\in C_k|\boldsymbol{x}\in t\right)\mathbb{P}\left(\boldsymbol{x}\in t\right)\right\},$$

which, in our formulation, turns into

$$\arg\max_{C_k}\left\{\sum_{t\in\tau_L} C_{kt}\cdot P_{\boldsymbol{x}t}\right\}. \tag{6}$$

Although, the final prediction is performed by assigning each individual to its most probable class, i.e., the class assignment is deterministic, a key contribution of our approach is the fact that we can also provide a probabilistic class assignment. For instance, Figure 3 represents the probabilities of class membership our ORCT gives to a subset of individuals from the Spambase data set, used in our numerical results, see Table 1. As we can see, for most of the individuals the probabilities are discriminatory. However, for some of them the probabilities are close to 0.5. This useful information is not directly provided either by CARTs or more recent optimation-based approaches.

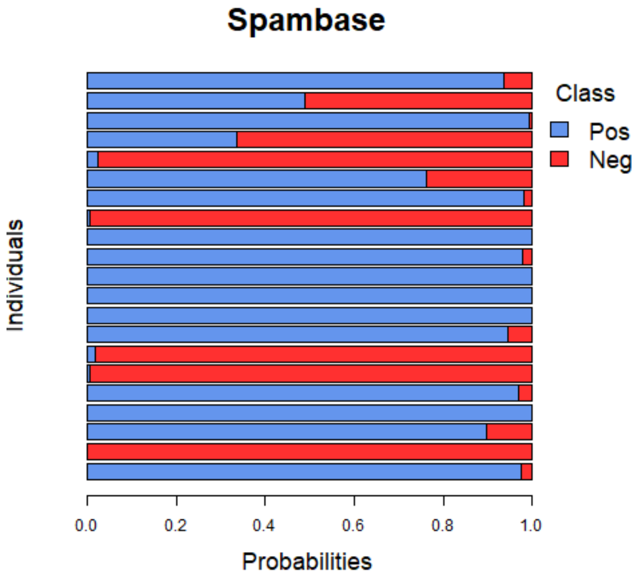


Figure 3: Probability output for a subset of individuals from the Spambase data set.

We remind the reader that along this text, we have considered numerical predictor variables; however, we can handle ordinal predictor variables, as well as categorical ones modeled, as usual, through dummy variables.

3 ORCT with performance constraints

Though classifiers seek a rule yielding a good overall classification rate, there are many cases in which misclassification for different classes has different consequences. It is then more appealing not to obtain the best overall classification, but obtaining an acceptable overall performance and ensuring a certain level of performance in some classes. One can modify the misclassification costs (2) in the objective function (3), but it is hard to control in this way the misclassification rates on critical classes. While preferences on classification rates in different classes can not be addressed directly by heuristic approaches like CARTs or RFs, ORCTs allow us to impose performance constraints over the different classes explicitly. Indeed, define the random variable O_i,

$$O_i = \begin{cases} 1, & \text{if individual } i \text{ is correctly classified} \\ 0, & \text{otherwise.} \end{cases}$$

Given $k = 1, \dots, K$, a Correct Classification Rate (CCR) over the $k-$th class, namely, ρ_k, is desired:

$$\frac{1}{|I_k|} \sum_{i \in I_k} O_i \geq \rho_k.$$

The expectation of achieving this performance can be written as:

$$\mathbb{E}\left[\frac{1}{|I_k|} \sum_{i \in I_k} O_i\right] = \frac{1}{|I_k|} \sum_{i \in I_k} \mathbb{E}\left[O_i\right] = \frac{1}{|I_k|} \sum_{i \in I_k} \sum_{t \in \tau_L} P_{it} C_{kt} \geq \rho_k.$$

Hence, given the class $k = 1, \dots, K$ to be controlled and its desired performance ρ_k, the following constraint would need to be added to the model:

$$\sum_{i \in I_k} \sum_{t \in \tau_L} P_{it} C_{kt} \geq \rho_k |I_k|. \tag{7}$$

4 Computational experiments

The purpose of this section is to show empirically that, with low running times and in terms of accuracy, our approach outperforms CART and a benchmark decision tree approach based on Integer Programming, and is competitive for some data sets against Random Forests, with the advantage of being much easier to interpret. Moreover, cost-sensitive constraints are easily handled as well.

Several well-known data sets from the UCI Machine Learning Repository [35] have been chosen for the computational experiments. Table 1 reports the size, the number of predictor variables, and the number of classes as well as the corresponding class distribution. This selection comprises data sets of different nature: from small-size data sets, such as Iris, to larger data sets, such as Spambase and Thyroid-disease-ann-thyroid; the class distribution is also diverse, from Iris or Seeds, which are balanced data sets, to Ozone-level-detection-one, which is highly imbalanced; lastly, and in addition to numerical (continuous as well as integer) predictor variables, we have also considered data sets with categorical predictor variables.

The NLCO model (5) has been implemented using Pyomo optimization modeling language [25, 26] in Python 3.5 [42]. As a solver, we have used IPOPT 3.11.1 [52]. Our experiments have been conducted on a PC, with an Intel® Core™ i5-4200M CPU 2.50GHz processor and 4 GB RAM. The operating system is 64 bits.

To train the ORCT, we solve the NCLP problem 20 times, starting from different random initial solutions.

Table 1: Information about the data sets considered.

Data set	n	p	K	Class distribution
Connectionist-bench-sonar	208	60	2	55% - 45%
Wisconsin	569	30	2	63% - 37%
Credit-approval	653	37	2	55% - 45%
Pima-indians-diabetes	768	8	2	65% - 35%
Statlog-project-German-credit	1000	48	2	70% - 30%
Ozone-level-detection-one	1848	72	2	97% - 3%
Spambase	4601	57	2	61% - 39%
Iris	150	4	3	33.3%-33.3%-33.3%
Wine	178	13	3	40%-33%-27%
Seeds	210	7	3	33.3%-33.3%-33.3%
Thyroid-disease-ann-thyroid	3772	21	3	92.5%-5%-2.5%
Car-evaluation	1728	15	4	70%-22%-4%-4%

The CDF chosen has been the logistic one:

$$F\left(\cdot;\gamma\right) = \frac{1}{1+\exp\left(-\left(\cdot\right)\gamma\right)},\ \gamma > 0.$$

In our computational experience, we have observed that γ needs to be a large value in order to obtain proper results. We have taken $\gamma = 512$, which yields satisfactory results.

Equal misclassification weights, $W_{kk'} = 0.5,\ k,k' = 1,\dots,K,\ k \neq k'$, have been used for the experiments.

We have used *accuracy* as the performance measure to compare ORCT with the benchmark methods. Each data set has been split into two subsets: the training subset (75%) and the test subset (25%). The ORCT is built over the training subset and, then, its performance is evaluated by determining the out-of-sample accuracy over the test subset. This procedure has been repeated ten times in order to avoid the effect of the initial splitting of data.

We will compare our ORCT with: the CART, as implemented in the `rpart` R package [46], the OCT-H proposed in [3] at the same depth as ORCT, and Random Forests from `randomForest` R package [34]. In our pursuit of building not very deep trees, and supported by the results in [3], we will restrict the construction of ORCTs to the minimum depth as possible according to the number of classes in each data set. That is, if $K = 2$, building an ORCT with depth $D = 1$ would be enough since there are precisely two leaf nodes; for $K = 3$ or 4, we will need $D = 2$.

4.1 Results for ORCT

Table 2 presents a comparison of ORCT at depth $D = 1$ against the entire CART and the OCT-H [3] at depth $D = 1$, as well as Random Forests, for two-class problems.

Table 2: Results for $D = 1$ in terms of the out-of-sample accuracy.

Data set	Average time (in secs)	Out-of-sample accuracy			
		ORCT	CART	OCT-H	RF
Connectionist-bench-sonar	22	**76.3**	70.0	70.4	83.1
Wisconsin	24	**96.4**	92.0	93.1	95.5
Credit-approval	22	83.7	85.7	**87.9**	86.7
Pima-indians-diabetes	21	**75.8**	74.2	71.6	76.3
Statlog-project-German-credit	28	**72.8**	72.1	71.6	75.2
Ozone-level-detection-one	94	96.7	95.6	**96.8**	96.4
Spambase	72	**89.8**	89.2	83.6	95.1

The average out-of-sample accuracy over the ten runs for ORCT, CART and RFs is complemented with the average execution time for ORCT. This time involves the data reading, the scaling of the training set, the random generation of initial solutions, the optimization time, the scaling of the test set using the scale parameters obtained in the training and the evaluation of performance. The average solving time of an instance of a specific optimization problem is on the order of seconds. Results for OCT-H are added and taken from [3]. The best accuracies among the one-single-tree-based methods are boldfaced. ORCT generally outperforms CART, even with depth 1, i.e., one single oblique cut. This is the case for all data sets except for Credit-approval. With respect to OCT-H, ORCT is better except for Credit-approval and Ozone-level-detection-one. Note that for the latter, the difference is 0.1 percentage points. With respect to RFs, although we are comparable in some data sets and even competitive in others (Wisconsin, Pima-indians-diabetes, Ozone-level-detection-one), increasing the depth of the tree may be required before we can compete against the bagged trees.

Table 3 presents similar results for depth $D = 2$ for three- and four-class problems. ORCT obtains the best classification rates for depth $D = 2$ compared to CART and OCT-H, except for the Thyroid-disease-ann-thyroid data set, in which we are comparable to OCT-H (difference equal to 0.3 percentage points) although CART outperforms these two methods. With respect to RFs, we are again comparable or even competitive in some of the data sets (Iris, Seeds and

Table 3: Results for $D = 2$ in terms of the out-of-sample accuracy.

Data set	Average time (in secs)	Out-of-sample accuracy			
		ORCT	CART	OCT-H	RF
Iris	17	**95.9**	92.7	95.1	95.4
Wine	23	**96.6**	88.6	91.1	98.6
Seeds	20	**94.2**	90.2	90.6	92.5
Thyroid-disease-ann-thyroid	145	92.2	**99.1**	92.5	99.1
Car-evaluation	71	**90.8**	88.1	87.5	88.0

Car-evaluation).

4.2 Results for variable importance measures via ORCTs

Measuring the importance of predictor variables in RFs has been thoroughly studied in the literature. In particular, the two most popular importance measures for forests are outlined in [8]: the Mean Decrease Impurity (MDI) and the Mean Decrease Accuracy (MDA). The MDI takes advantage of the splitting criterion used for growing classic decision trees: given an impurity measure, the predictor variable that maximizes the decrease of impurity together with its corresponding splitting threshold are chosen. Thus, the MDI of predictor variable j is the average over every tree built of the decrease in impurity of splits along that variable, weighted with the fraction of individuals falling in the corresponding branch node. The MDA is based on the following notion: if a predictor variable is not influential, permuting the values it takes should not affect the prediction accuracy of the forest. Thus, the MDI of predictor variable j is the average over every tree built of the difference in accuracy before and after the permutation.

In our case we have two straightforward and inexpensive ways of measuring the importance of predictor variables by analyzing the distribution of the absolute values of coefficients $\boldsymbol{a_{jt}}$. The first metric to measure the importance of predictor variable $j,\ j = 1, \dots, p$, is to sum the absolute values of the coeffientes a_{jt} for all branch nodes $t \in \tau_B$, called hereafter the Sum Importance Measure (SIM):

$$\mathrm{SIM}_j = \sum_{t \in \tau_B} |a_{jt}|.$$

The second metric is the so-called Maximum Importance Measure (MIM), which takes instead the maximum among all the branch nodes $t \in \tau_B$:

$$\mathrm{MIM}_j = \max_{t \in \tau_B} |a_{jt}|.$$

For illustrative purposes, these variable importance measures have been evaluated for the Wine and Car-evaluation data sets in Table 1, see Figures 4 and 6. Both measures have been evaluated over the resultant ORCT of the tenth run. The variable importance measures for RFs, the MDA and the MDI using the Gini index as the impurity measure, are also depicted for the aforementioned data sets. These can be found in Figures 5 and 7 and have been obtained with the `randomForest` R package. The message conveyed by these plots is, in general, in agreement. For instance, V8 in the Wine data set (Car-evaluation data set) is not as important as V10.

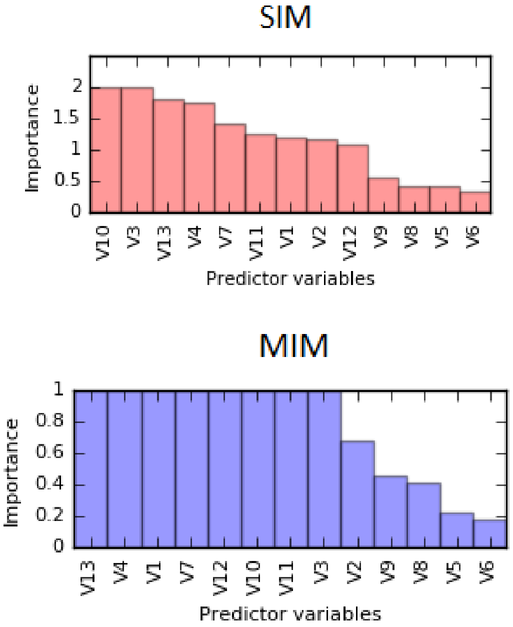


Figure 4: ORCT variable importance measures for the Wine data set.

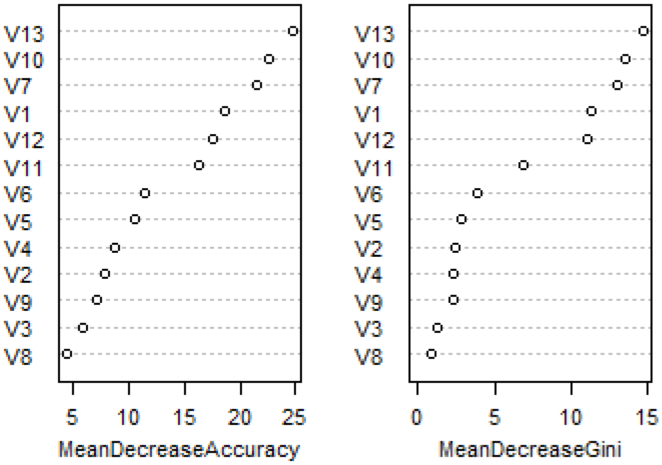


Figure 5: RF variable importance measures for the Wine data set.

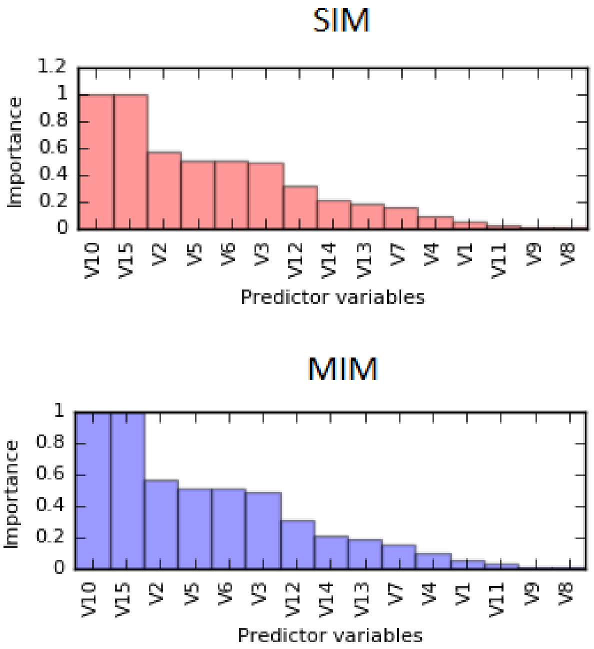


Figure 6: ORCT variable importance measures for the Car-evaluation data set.

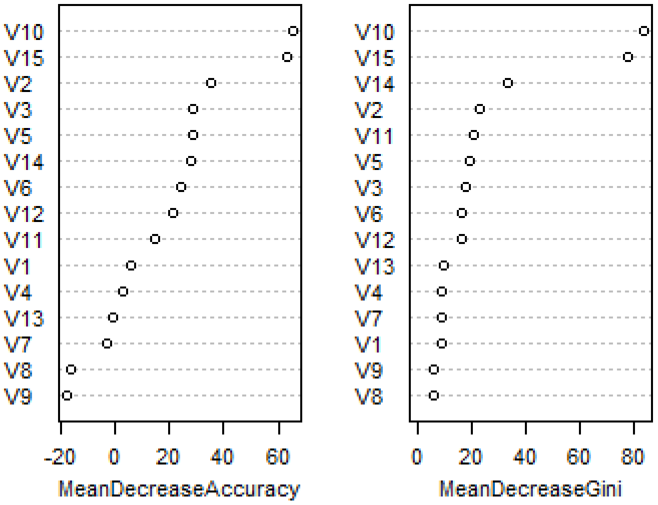


Figure 7: RF variable importance measures for the Car-evaluation data set.

4.3 Results for ORCT with performance constraints

The Pima-indians-diabetes data set, from [35], consists of a sample of 768 patients of Pima Indian heritage in the USA, on which 8 indicators (predictor variables) are measured. The target variable is whether the patient shows signs of diabetes or not. Diabetics are the positive class and represent the 35% of the entire sample. Firstly, we have run our ORCT without performance constraints by imposing a correct classification percentage over the positive class, ρ_+, equal to zero. See the first row in Table 4. The positive class, diabetics, is the worst classified, with an average True Positive Rate (TPR) of 60.7 over the ten training subsets and 55.6 over the ten test subsets. The negative class, non-diabetics, presents an average True Negative Rate (TNR) of 90.5 and 80.7 over the ten training and test subsets, respectively. In this setting, it is preferred to better classify diabetic patients, since diagnosing a diabetic as non-diabetic is more critical (in terms of missing medical treatment) than the other way around.

Table 4: Results with performance constraints over the positive class in the Pima-indians-diabetes data set.

Imposed TPR (ρ_+)	TPR_{train}	TPR_{test}	TNR_{train}	TNR_{test}	CCR_{train}	CCR_{test}
0	60.7	55.6	90.5	87.7	80.3	75.8
62.5	63.8	59.0	88.7	85.8	80.1	76.0
65.0	65.8	60.9	87.3	84.7	79.9	75.9
67.5	68.4	62.5	85.5	83.1	79.6	75.5
70.0	71.1	64.3	83.6	81.8	79.3	75.3
72.5	73.6	67.8	81.4	80.5	78.7	75.8
75.0	75.8	68.1	79.3	77.4	78.1	73.9
77.5	78.9	72.9	77.2	75.9	77.8	74.6
80.0	81.3	73.4	74.5	72.4	76.8	72.6
82.5	84.0	77.2	71.9	69.2	76.0	71.9
85.0	86.3	80.2	68.5	66.8	74.6	71.5

In this regard, a performance constraint over the positive class has been added to the ORCT for several values of the threshold ρ_+. We have considered a grid of values for ρ_+ varying from 62.5, a slightly higher value than the training TPR obtained with $\rho_+ = 0$, to 85.0 in steps of 2.5 units. Note that performance constraints (7) are imposed over the training sample, so they might not be satisfied on an independent sample. Indeed, results in Table 4 show how these thresholds are satisfied in the training subsets but this is not necessarily the case in the test subsets; even so, we observe that the test TPR increases with ρ_+. Figure 8 supports this observation, in which the $\mathrm{TPR}_{\mathrm{train}}$ and the $\mathrm{TPR}_{\mathrm{test}}$ are depicted as a function of the imposed TPR (ρ_+). There, we can see that there exists a good linear fit. In fact, the regression models' coefficients and their corresponding coefficients of determination are the following:

$$\mathrm{TPR}_{\mathrm{train}} = -0.47 + 1.02\rho_+, R^2 = 0.9994,$$

$$\mathrm{TPR}_{\mathrm{test}} = -0.55 + 0.94\rho_+, R^2 = 0.9873.$$

A clear overfitting, almost independent of the threshold imposed, is also detected; but it is possible to determine the required imposed TPR in order to obtain a successful $\mathrm{TPR}_{\mathrm{test}}$. There is a price to pay for achieving such high TPRs: the TNRs decrease as we demand larger thresholds, see Figure 9.

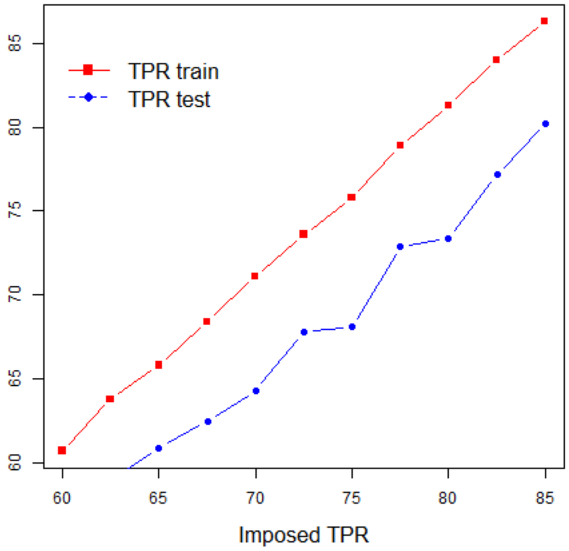


Figure 8: $\mathrm{TPR}_{\mathrm{train}}$ and $\mathrm{TPR}_{\mathrm{test}}$ drawn as a function of the imposed TPR $(\boldsymbol{\rho_+})$ for Pima-indians-diabetes data set.

.

5 Conclusions and future research

Several papers have been proposed in recent years to build classification trees in which the greedy suboptimal construction approach is replaced by solving an optimization problem, usually in integer variables. These procedures, while successful against CART, are very time consuming and can only address problems of moderate size. In this paper we have proposed a new optimization-based approach to build classification trees. By replacing the binary decisions with randomized decisions, the resulting optimization problem is smooth and only contains continuous variables, allowing one to use gradient information. The computational experience reported shows that, with low running times, we outperform recent benchmarks, getting closer to and some times better than the performance of Random Forests. Moreover, we can easily address cost-sensitive

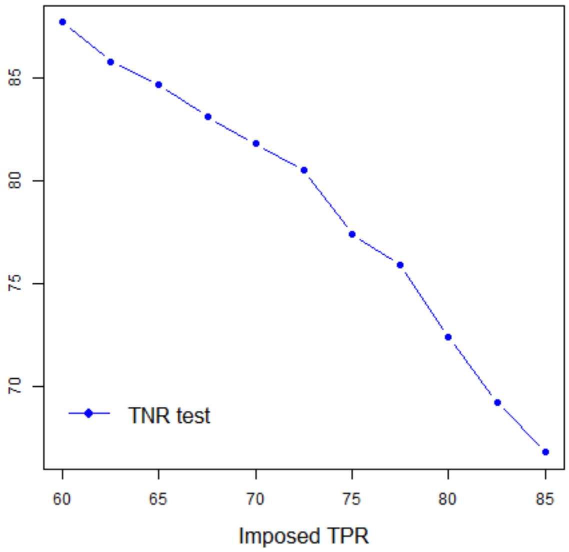


Figure 9: $\mathrm{TNR}_{\mathrm{test}}$ depicted as a function of the imposed TPR (ρ_+) for Pima-indians-diabetes data set.

constraints to control the accuracy of a specific class, in detriment of the accuracy in the remaining classes.

Several extensions of our approach are promising and deserve further investigation. First, it is known that bagging trees (i.e., using Random Forests instead of decision trees) tends to enhance the accuracy. An appropriate bagging scheme for our approach is a nontrivial design question. Second, it is difficult in decision trees to control the number of predictor variables used. Making our approach sparse by means of an ℓ_1 regularisation, i.e., by using a lasso-type objective, is also an interesting open question.

Acknowledgements

This research has been financed in part by research projects COSECLA - Fundación BBVA, MTM2015-65915R, Spain, P11-FQM-7603 and FQM-329, Junta de Andalucía, the last three with EU ERF funds. This support is gratefully acknowledged.

References

[1] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[2] K. P. Bennett and J. Blue. Optimal decision trees. *Rensselaer Polytechnic Institute Math Report*, 214, 1996.

[3] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[4] D. Bertsimas and A. King. An algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2015.

[5] D. Bertsimas and R. Mazumder. Least quantile regression via modern optimization. *The Annals of Statistics*, 42(6):2494–2525, 2014.

[6] D. Bertsimas, A. O'Hair, S. Relyea, and J. Silberholz. An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531, 2016.

[7] D. Bertsimas and R. Shioda. Classification and regression via integer optimization. *Operations Research*, 55(2):252–271, 2007.

[8] G. Biau and E. Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[9] L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv: 1606.04838* 2016.

[10] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[11] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[12] J. P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations Research*, 59(2):467–479, 2011.

[13] E. Carrizosa, V. Guerrero, and D. Romero Morales. Visualizinsg data as objects by dc (difference of convex) optimization. *Mathematical Programming*, 169(1):119–140, 2018.

[14] E. Carrizosa and D. Romero Morales. Supervised classification and mathematical optimization. *Computers & Operations Research*, 40(1):150–165, 2013.

[15] W. A. Chaovalitwongse, Y.-J. Fan, and R. C. Sachdeo. Novel optimization models for abnormal brain activity classification. *Operations Research*, 56(6):1450–1460, 2008.

[16] I. Chikalov, S. Hussain, and M. Moshkov. Bi-criteria optimization of decision trees with applications to data analysis. *European Journal of Operational Research*, 266(2):689–701, 2018.

[17] X. Fang, O. R. L. Sheng, and P. Goes. When is the right time to refresh knowledge discovered from data? *Operations Research*, 61(1):32–44, 2013.

[18] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181, 2014.

[19] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[20] K. Fountoulakis and J. Gondzio. A second-order method for strongly convex ℓ_1-regularization problems. *Mathematical Programming*, 156(1):189–219, 2016.

[21] A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.

[22] R. Genuer, J.-M. Poggi, C. Tuleau-Malot, and N. Villa-Vialaneix. Random Forests for Big Data. *Big Data Research*, 9:28–46, 2017.

[23] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

[24] O. Günlük, J. Kalagnanam, M. Menickelly, and K. Scheinberg. Optimal Decision Trees for Categorical Data via Integer Programming. *arXiv preprint arXiv:1612.03225v2*, 2018.

[25] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo–Optimization Modeling in Python*, volume 67. Springer Science & Business Media, second edition, 2017.

[26] W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[27] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. 2nd edition, New York: Springer, 2009.

[28] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[29] A. Jakaitiene, M. Sangiovanni, M. Guarracino, and P. Pardalos. *Multidimensional Scaling for Genomic Data*, pages 129–139. Springer International Publishing, Cham, 2016.

[30] J. Jung, C. Concannon, R. Shroff, S. Goel, and D. G. Goldstein. Simple rules for complex decisions. *arXiv preprint arXiv:1702.04690*, 2017.

[31] E. B. Khalil, P. L. Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 724–731. AAAI Press, 2016.

[32] J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan. Human decisions and machine predictions. *The Quarterly Journal of Economics*, 133(1):237–293, 2018.

[33] X.-B. Li and S. Sarkar. Against classification attacks: A decision tree pruning approach to privacy protection in data mining. *Operations Research*, 57(6):1496–1509, 2009.

[34] A. Liaw and M. Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002.

[35] M. Lichman. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences, 2013.

[36] D. Martens, B. Baesens, T. Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[37] V. V. Mišić. Optimization of Tree Ensembles. *arXiv preprint arXiv:1705.10883v*, 2017.

[38] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli. Efficient non-greedy optimization of decision trees. In *Advances in Neural Information Processing Systems*, pages 1729–1737, 2015.

[39] S. Olafsson, X. Li, and S. Wu. Operations research and data mining. *European Journal of Operational Research*, 187(3):1429–1448, 2008.

[40] C. Orsenigo and C. Vercellis. Multivariate classification trees based on minimum features discrete support vector machines. *IMA Journal of Management Mathematics*, 14(3):221–234, 2003.

[41] P. Pardalos, V. Boginski, and A. Vazacopoulos, editors. *Data Mining in Biomedicine*. Springer Optimization and Its Applications. Springer, 2007.

[42] Python Core Team. Python: A dynamic, open source programming language. Python Software Foundation, 2015.

[43] G. Ridgeway. The pitfalls of prediction. *National Institute of Justice Journal*, 271:34–40, 2013.

[44] P. Savickỳ, J. Klaschka, and J. Antoch. Optimal classification trees. In *COMPSTAT*, pages 427–432. Springer, 2000.

[45] W. Souillard-Mandar, R. Davis, C. Rudin, R. Au, D. Libon, R. Swenson, C. Price, M. Lamar, and D. Penney. Learning classification models of cognitive conditions from subtle behaviors in the digital clock drawing test. *Machine Learning*, 102(3):393–441, 2016.

[46] T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. R package version 4.1-10.

[47] A. Truong. *Fast growing and interpretable oblique trees via logistic regression models.* PhD thesis, University of Oxford, 2009.

[48] B. Ustun and C. Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[49] V. Van Vlasselaer, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens. GOTCHA! Network-based fraud detection for social security fraud. *Management Science*, 63(9):3090–3110, 2017.

[50] S. Verwer and Y. Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 94–103. Springer, 2017.

[51] S. Verwer, Y. Zhang, and Q. C. Ye. Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence*, 244:368–395, 2017.

[52] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[53] J. Wang, R. Fujimaki, and Y. Motohashi. Trading interpretability for accuracy: Oblique treed sparse additive models. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1245–1254. ACM, 2015.

[54] L. Yang, S. Liu, S. Tsoka, and L. G. Papageorgiou. A regression tree approach using mathematical programming. *Expert Systems with Applications*, 78:347–357, 2017.

[55] J. Zeng, B. Ustun, and C. Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A*, 180(3):689–722, 2017.