New optimization models for optimal classification trees

Zacharie Alès, Valentine Huré, Amélie Lambert

▶ **To cite this version:**

Zacharie Alès, Valentine Huré, Amélie Lambert. New optimization models for optimal classification trees. Computers and Operations Research, 2023, pp.106515. 10.1016/j.cor.2023.106515 . hal-03865931v2

HAL Id: hal-03865931
https://hal.science/hal-03865931v2

Submitted on 23 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

New optimization models for optimal classification trees

Zacharie Ales, Valentine Huré, Amélie Lambert

November 23, 2023

Abstract

The most efficient state-of-the-art methods to build classification trees are greedy heuristics (e.g. CART) that may fail to find underlying characteristics in datasets. Recently, exact linear formulations that have shown better accuracy were introduced. However they do not scale up to datasets with more than a few thousands data points. In this paper, we introduce four new formulations for building optimal trees. The first one is a quadratic model based on the well-known formulation of Bertsimas et al. We then propose two different linearizations of this new quadratic model. The last model is an extension to real-valued datasets of a flow-formulation limited to binary datasets (Aghaei et al.). Each model is introduced for both axis-aligned and oblique splits. We further prove that our new formulations have stronger continuous relaxations than existing models. Finally, we present computational results on 22 standard datasets with up to thousands of data points. Our exact models have reduced solution times with learning performances as strong or significantly better than state of the art exact approaches.

Keywords: combinatorial optimization, optimal classification trees, mixed binary programming, quadratic programming, linearizations

1 Introduction

1.1 Related works

A supervised classification problem considers a dataset where each data is a pair composed of a feature vector and a target class. The objective is then to determine a *classifier* to best predict the class of data from their feature vectors. Interpretability is a concept that is increasingly considered in supervised classification see [Carvalho et al., 2019]. Although several definitions of interpretability can be considered, it can be summarised as the ability to explain or to present in understandable terms to a human how a classifier works [Doshi-Velez and Kim, 2017]. There is currently no clear metric to measure interpretability but rather a consensus on which models are (or are not) interpretables [Zhou et al., 2021]. A variety of reasons explain the rising interest of interpretability. For example, in the General Data Protection Regulation (GDRP), the notion of a *right to explanation* was introduced [Goodman and Flaxman, 2017,Wachter et al., 2016]. Interpretability may also allow users to have more confidence in the results of a classifier which is particularly important when taking sensitive decisions (e.g. medical or legal applications).

Two main approaches can generally be considered to analyze the prediction of a classifier in supervised classification. The first approach, applied to the most complex classifiers, consists in developing post-hoc models that can, to some extent, explain a classifier predictions, this is explanability. For example, in [Ribeiro et al., 2016], instead of training a global surrogate model, the method `LIME (Local Interpretable Model-Agnostic Explanation)`, focuses on training local surrogate models to explain individual predictions. The second approach consists in considering classifiers with a human-understandable decision process. In other words classifiers that are intrinsically interpretable [Rudin, 2019]. Note that interpretability and prediction performances are usually conflicting goals as the most efficient classifiers tend to be very complex (e.g., deep neural networks). This has been amplified by the fact that, for the last decades, research in supervised classification has mainly focused on performances rather than interpretability. The work presented in this paper aims to obtain good classifiers that are interpretable, and for this we focus on the construction of Optimal Classification Trees (OCT).

A classification tree is an oriented binary tree of a maximum given depth that associates a split function to each of its internal node and a class to each of its leaves. Splitting rules route data to the left or right child of the

internal nodes and a data is classified by the path it follows from the root to a leaf. The split functions are generally linear functions. When they only involve a single feature they are *axis-aligned* (or *univariate*). Otherwise, they are *oblique* (or *multivariate*).

The problem of the construction of an OCT is NP-complete [Hyafil and Rivest, 1976] and the most used approaches are greedy heuristics such as `CART (Classification And Regression Trees)` [Breiman et al., 1984] that constructs univariate classification trees. The basic idea of `CART` is to start with a tree composed only of its root node. Then, at each step one leaf is transformed into an internal node which split function is determined by optimizing an impurity measure over all the training data reaching it (e.g., the Gini index [Jost, 2006]). This process is recursively applied to the two new generated nodes until all the data reaching a leaf have the same class. Additional stopping criterion are usually considered such as a maximal depth or a minimal number of data reaching each leaf. Further, improved heuristics that use different impurity functions were introduced, such as `ID3` [Quinlan, 1986] or `C5.0` [Quinlan, 1993] algorithms, or that are based on statistical tests, such as `GUIDE` [Loh, 2002,Loh, 2009]. These greedy approaches are fast but do not guarantee the optimality of their solutions. In order to improve the prediction performances of these heuristics, tree ensemble methods such as `Random Forests` [Breiman, 2001], `TreeBoost` [Friedman, 2001] or `XGBoost` [Chen and Guestrin, 2016] were then proposed. These approaches clearly improve the performances of the predictions but the fact that several trees are required to classify the data lead to a loss of interpretability. Other heuristics that use oblique hyperplane splits were then introduced (see for instance [Brodley and Utgoff, 1995,Murthy et al., 1994,Orsenigo and Vercellis, 2003,Wickramarachchi et al., 2016]). These approaches allow to obtain smaller trees, but they can be computationally costly.

Recently, exact solution methods for the construction of OCT that are based on Mixed-Integer Linear Programming (MILP) formulations were introduced. For the case of real-valued datasets, the formulation of [Bertsimas and Dunn, 2017] considers the most general case of oblique splits, with an objective function that is a trade-off between minimizing the number of misclassifications on the *train set* and the complexity of the tree in order to preserve interpretability. The constraints and variables of the formulation can be decomposed in 3 sets: those that define the structure of the tree, those that follow the path of a data in the tree, and those that count the number of misclassifications. Another MILP formulation, called `BinOCT`*, that builds univariate trees was further introduced in [Verwer and Zhang, 2019]. The main difference with the method of [Bertsimas and Dunn, 2017] is that the variable associated to the coefficient of a function does not represent its exact value but an interval in which it can take any value. These two exact methods find better solutions than standard greedy heuristic approaches such as `CART`, `ID3`, or `C4.5`. However, since their sizes depend on the number of data points, they become untractable for datasets of more than a few thousand points. A way to scale up these formulations is to solve the MILPs heuristically using methods such as local search (see `TAO` [Carreira-Perpinan and Tavallali, 2018] or [Dunn, 2018]), or column generation (see algorithm `CGH` [Firat et al., 2020]). In a slightly different context, optimal randomized classification tree [Blanquero et al., 2021] consider a variant of the problem in which the path of a data in the tree is not deterministic which leads to a less interpretable model. We introduce in this paper a new compact formulation for OCT where a quadratic formulation of the objective function allows us to remove all the constraints and variables relative to the counting of the missclassifications. The obtained formulation has thus a smaller size than the original formulation of [Bertsimas and Dunn, 2017]. Then, we propose two different linearizations of the objective function, leading to two MILP formulations. We finally compare our new formulations from the continuous relaxation point of view with the original formulation of [Bertsimas and Dunn, 2017], and we prove that we always have better lower bounds, allowing us to exactly solve the OCT problem faster.

Several works are also devoted to binary datasets. Exact approaches for solving this specific case have been introduced based either on MILP [Aghaei et al., 2020, Lin et al., 2020], on constraint programming [Aglin et al., 2020]), or on dynamic programming [Demirović et al., 2022]. In particular, the work of [Aghaei et al., 2020] uses the specific structure of the binary dataset to model the OCT problem as a maximum flow problem. This model has smaller solution times and similar performances than the more general MILP approaches that can be applied to real-valued datasets. However, a direct extension of this flow model to the case of real-valued datasets requires a binary expansion of the features leading, either to information loss, or to a prohibitive number of features. We introduce in this paper a new maximum flow formulation that applies on real-valued datasets for both cases of axis-aligned and oblique splits. In our new model, we keep the characteristics as they are (i.e. real-valued), which allows us to have a formulation of moderate size. We then prove for both cases that the new formulations lead to better continuous relaxation bounds than the original formulation of [Bertsimas and Dunn, 2017]. Moreover, a significant result is that in the oblique case, our new formulation can even compute strictly better integer solutions than the formulation of [Bertsimas and Dunn, 2017].

All the above-mentioned formulations depend on several parameters, including tree depth and the weight (in

the objective function) of the term used to control the tree complexity. To compute these parameters, an iterative learning algorithm was introduced in [Bertsimas and Dunn, 2017] where at each iteration a MILP is solved. In this work, we propose a new iterative learning algorithm which, by iterating in the reverse order, reduces the total number of iterations. This leads to a significantly smaller number of MILPs to be solved and further reduces the total computational time.

1.2 Our Contribution

In this paper, we propose a novel approach to construct Optimal Classification Trees in the sense of [Bertsimas and Dunn, 2017], using new optimization formulations. Our main contributions are:

1. Four new compact and efficient formulations to compute Optimal Classification Trees (with axis-aligned or oblique splits).
2. A theoretical comparison of the introduced formulations with the state-of-the-art.
3. An iterative `Compact Tree Training (CTT)` algorithm that fits the parameters of the models in fewer iterations than the algorithm proposed in [Bertsimas and Dunn, 2017].
4. An extensive battery of computational experiments on 22 standard datasets comparing our approaches to other methods of the literature: `OCT`, `CGH`, `BinOCT`, `CART`, `C5.0`, `GUIDE` and `TAO`.

The paper is organised as follows. In Section 2, we present in details the formulations our work is based on, in particular the models (T) of [Bertsimas and Dunn, 2017] and (F^b) of [Aghaei et al., 2020]. Then, in Section 3, we present our new compact quadratic formulation based on (T). For this, we introduce a quadratic formulation of the objective function allowing us to remove all the constraints and variables dedicated to counting misclassifications. We further linearize it in two different ways obtaining two new formulations. We then prove that these linear programs have a tighter continuous relaxation than (T). Further, in Section 4, we propose an extension of the flow-formulation (F^b) from [Aghaei et al., 2020] to real-valued datasets. Here again, we prove that the continuous relaxation of our new formulation is tighter than that of (T). We also prove that our model can be extended to the *oblique* case and leads to strictly better integer solutions than (T). All these formulations generally have several optimal solutions. To select a good one, we propose in Section 5 a heuristic that locally shifts the split functions as far as possible from the data. We then present our `Compact Tree Training (CTT)` algorithm that fits the parameters of our formulations. This iterative algorithm performs fewer iterations than the original algorithm proposed in [Bertsimas and Dunn, 2017]. Finally, in Section 6, we provide extensive experimental results on 22 standard datasets comparing our approaches to 7 methods of the literature. We show that our models have reduced solution times with learning performances as strong or significantly better than state of the art approaches. Section 7 draws as conclusion.

2 Preliminaries

In this section we present in more detail formulations (T) [Bertsimas and Dunn, 2017] and (F^b) [Aghaei et al., 2020] that compute Optimal Classification Trees. We consider a dataset $(X_i, y_i)_{i\in\mathcal{I}}$ such that $X_i = (x_{i,j})_{j\in\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}$ is the feature vector of data $i \in \mathcal{I}$ and $y_i \in \mathcal{K}$ is its associated class. A decision tree is an oriented binary tree $\mathcal{T} = (\mathcal{N} \cup \mathcal{L}, E)$ which associates a split function $f_t : \mathbb{R}^{|\mathcal{J}|} \to \{true, false\}$ to each of its internal node $t \in \mathcal{N}$ and a class $k \in \mathcal{K}$ to each of its leaves $\ell \in \mathcal{L}$. We consider linear split functions of the form $a^\top X_i < b$ with $a \in \mathbb{R}^{|\mathcal{J}|}$ and $b \in \mathbb{R}$. A data $i \in \mathcal{I}$ is classified by the path it follows from the root to a leaf by applying the split functions to the feature vector X_i. Data i follows the left branch of node $t \in \mathcal{N}$ if $(a^\top X_i < b)$ and the right branch otherwise. The class predicted by the classifier is the one associated with the leaf reached by the path of data i.

For a given dataset and a given maximal depth, the structure of an optimal classification tree is unknown. Consequently, both of these formulations consider a complete binary tree (i.e., each internal node has two child nodes) and associate a first set of binary variables to each of its nodes that indicate whether the node is *active* (i.e. if it applies a split) or not in the solution. Another set of variables serves to compute the value of the coefficients (a and b) of the split functions, and to determine the class assigned to each leaf. Finally, the last set of variables is dedicated to follow the path of the data in the tree, in order to count the number of classification errors.

The main difference between formulations (T) and (F^b) is that the latter can only be applied to binary datasets and only consider axis-aligned split functions. Consequently, explicitly computing the coefficients (a and b) of the split function is not required, since the value of the features directly routes the data into the tree (i.e. it follows the left branch if $x_{i,j} = 0$, and the right otherwise). Their idea is thus to associate a flow to each data only if the data is correctly classified (i.e. if it is routed to a leaf of the correct class). This is not the case in (T), and additional real variables are required to calculate the coefficients a and b of the split functions and the path of a data is modeled thanks to binary variables indicating which leaf it reaches.

2.1 A first formulation for Optimal Classification Tree [Bertsimas and Dunn, 2017]

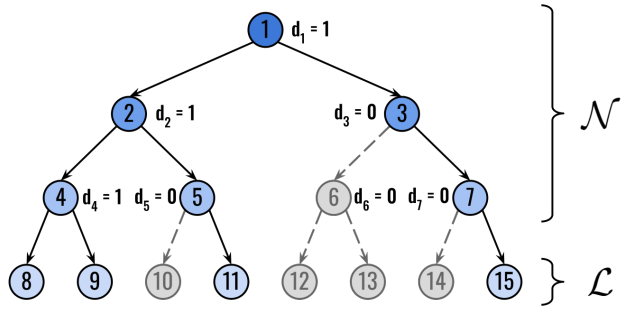
Bertsimas et al. [Bertsimas and Dunn, 2017] introduced two models that applies to real-valued datasets. Without loss of generality, we consider in this section that the value of each feature belongs to $[0, 1]$. For readability, we start by the introduction of the simpler model that computes OCT with axis-aligned split functions, (i.e. when $a_j \in \{0, 1\}$ and $\sum_{j\in\mathcal{J}} a_j = 1$). Then, we present its extension to the case of oblique splits (i.e. when $a_j \in [-1, 1]$ and $\sum_{j\in\mathcal{J}} |a_j| \leq 1$).

2.1.1 The axis-aligned case

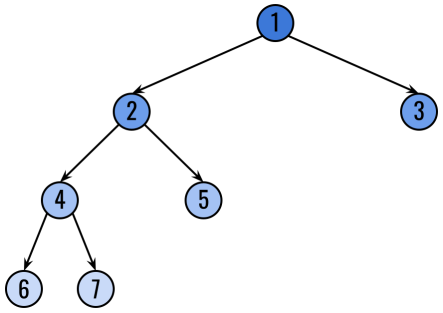
This formulation considers variables that define the structure of the tree and associate a class $k \in \mathcal{K}$ to each leaf. Each data i, is assigned to the class associated with the leaf at the end of its path in the tree. Formulation $(T_{\alpha,\beta,\delta})$ considers three parameters which control the structure of the tree: δ the depth of the tree, α the penalty limiting the number of splitting nodes, and β the minimal number of data reaching a leaf. In the following, we denote by $r \in \mathcal{N}$ the root of the tree, and by $a(t)$ the direct ancestor of node $t \in \mathcal{N}\backslash\{r\}$. Let P_ℓ be the path from r to a leaf $\ell \in \mathcal{L}$, we denote by $A_L(\ell)$ ($A_R(\ell)$ respectively), the subset of P_ℓ whose left (right resp.) child is in P_ℓ.

Since the structure of an optimal tree of depth δ is not known a priori, $(T_{\alpha,\beta,\delta})$ associates variables to each node of the complete tree (i.e. $|\mathcal{N}| = 2^\delta - 1$ and $|\mathcal{L}| = 2^\delta$). The split function of each internal node $t \in \mathcal{N}$ is determined by $|\mathcal{J}|$ variables $a_{j,t} \in \{0, 1\}$ and one variable $b_t \in [0, 1]$. The class predicted by a leaf $\ell \in \mathcal{L}$ is determined by variables $c_{k,\ell}$ which are equal to 1 if and only if class k is assigned to leaf ℓ. A data i assigned to leaf ℓ has to satisfy $\sum_{j\in\mathcal{J}} a_{j,t_r} x_{i,j} \geq b_{t_r}$ for all $t_r \in A_R(\ell)$, and $\sum_{j\in\mathcal{J}} a_{j,t_l} x_{i,j} < b_{t_l}$ for all $t_l \in A_L(\ell)$. To satisfy the latter strict inequalities, a vector $\mu \in [0, 1]^{|\mathcal{J}|}$ as well as two scalars $\mu^- = \min_{j\in\mathcal{J}}(\mu_j)$ and $\mu^+ = \max_{j\in\mathcal{J}}(\mu_j)$ are introduced. Each μ_j is the smallest positive interval between values of the training data on feature j (i.e. $\mu_j = \min\{|x_{i_1,j} - x_{i_2,j}|, \text{ s.t. } x_{i_1,j} \neq x_{i_2,j}, i_1, i_2 \in \mathcal{I}^2\}$). To follow the path of $X_i \in \mathcal{I}$ in the tree, binary variables $z_{i,\ell}$ that indicate if X_i reaches leaf ℓ are introduced.

To avoid overfitting, solutions representing non-complete trees are allowed, i.e. some nodes may not apply a split if the trade-off between decreasing misclassifications and increasing complexity is not beneficial. We refer to nodes that apply a split as *active*. Variable d_t is equal to 1 if and only if node $t \in \mathcal{N}$ is active. In the model, all data reaching an *inactive* node $t \in \mathcal{N}$ is sent to its right branch by setting $a_{j,t}$ and b_t to 0 for all $j \in \mathcal{J}$. Thus, some leaves may not be reached by any data. We denote leaves reached by data as *opened*. Variable l_ℓ is equal to 1 if and only if leaf $\ell \in \mathcal{L}$ is opened. Let us illustrate these notations by considering a solution in which the variables d_t take the values represented in Figure 1a. Four of the splitting nodes are inactive (3, 5, 6, and 7) and four of the leaves are closed (10, 12, 13, 14) leading to a decision tree which structure is represented in Figure 1b. Since all the data reaching an inactive node are sent to its right branch, leaf 3 in Figure 1b will be assigned the class predicted by node 15 in Figure 1a.



(a) Value of variables d_t.



(b) Structure of the corresponding decision tree.

Figure 1: Link between the value of variables d_t of $(T_{\alpha,\beta,\delta})$ and the structure of the decision tree obtained.

Additional variables are considered to count the number of misclassifications. Variable N_ℓ is equal to the number of data reaching leaf $\ell \in \mathcal{L}$ and variable $N_{k,\ell}$ is the number of these data whose class is $k \in \mathcal{K}$. Finally, variables L_ℓ is equal to the number of misclassifications in leaf $\ell \in \mathcal{L}$.

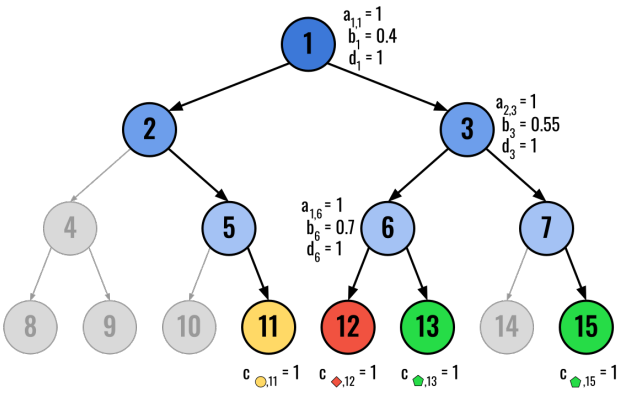
The objective function is a weighted sum of two criteria. The first criterion minimizes the number of misclassifications weighted by a constant $\hat{L}$ that corresponds to the number of misclassifications for an optimal tree of depth $\delta = 0$. The second one minimizes the number of active nodes weighted by parameter $\alpha \geq 0$, and allows to control both the classifier interpretability and the overfitting by penalizing the number of active nodes. Indeed, reducing the number of split functions make the classifier easier to understand. Moreover, performing too many splits, may produce a tree which is very good on the training set but generalizes poorly to the test set. Note that α should be smaller than 1, otherwise, due to constant $\hat{L}$, the solution will always be a tree reduced to its root. Model $(T_{\alpha,\beta,\delta})$ is formulated as follows:

$$
(T_{\alpha,\beta,\delta}) \left\{
\begin{array}{lll}
\min \ \frac{1}{\hat{L}} \sum_{\ell \in \mathcal{L}} L_\ell + \alpha \sum_{t \in \mathcal{N}} d_t & & \\
\text{s.t.} \sum_{j \in \mathcal{J}} a_{j,t} = d_t & t \in \mathcal{N} & (1) \\
0 \leq b_t \leq d_t & t \in \mathcal{N} & (2) \\
d_t \leq d_{a(t)} & t \in \mathcal{N} \setminus \{r\} & (3) \\
\sum_{k \in \mathcal{K}} c_{k,\ell} = l_\ell & \ell \in \mathcal{L} & (4) \\
\sum_{i \in \mathcal{I}} z_{i,\ell} \geq \beta l_\ell & \ell \in \mathcal{L} & (5) \\
z_{i,\ell} \leq l_\ell & \ell \in \mathcal{L}, i \in \mathcal{I} & (6) \\
\sum_{\ell \in \mathcal{L}} z_{i,\ell} = 1 & i \in \mathcal{I} & (7) \\
\sum_{j \in \mathcal{J}} a_{j,t} \left(x_{i,j} + \mu_j - \mu^- \right) + \mu^- \leq b_t + (1 + \mu^+)(1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_L(\ell) & (8) \\
\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t - (1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_R(\ell) & (9) \\
N_{k,\ell} = \sum_{i \in \mathcal{I}, y_i = k} z_{i,\ell} & \ell \in \mathcal{L}, k \in \mathcal{K} & (10) \\
N_\ell = \sum_{i \in \mathcal{I}} z_{i,\ell} & \ell \in \mathcal{L} & (11) \\
L_\ell \geq N_\ell - N_{k,\ell} - |\mathcal{I}|(1 - c_{k,\ell}) & \ell \in \mathcal{L}, k \in \mathcal{K} & (12) \\
L_\ell \leq N_\ell - N_{k,\ell} + |\mathcal{I}| c_{k,\ell} & \ell \in \mathcal{L}, k \in \mathcal{K} & (13) \\
L_\ell \geq 0 & \ell \in \mathcal{L} & (14) \\
a_{j,t} \in \{0,1\}, d_t \in \{0,1\} & t \in \mathcal{N}, j \in \mathcal{J} & (15) \\
c_{k,\ell} \in \{0,1\}, l_\ell \in \{0,1\}, z_{i,\ell} \in \{0,1\} & \ell \in \mathcal{L}, k \in \mathcal{K}, i \in \mathcal{I} & (16)
\end{array}
\right.
$$

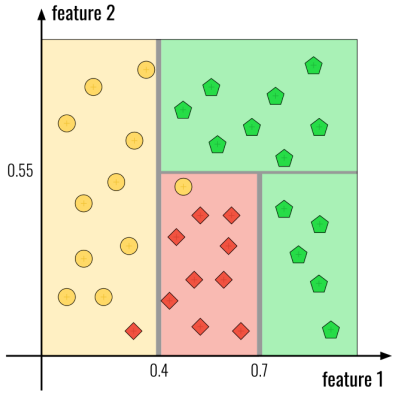
The first sets of constraints fix the structure of the tree. Constraints (1) and (2) ensure that the variables

defining the split function of $t \in \mathcal{N}$ vanish when t is inactive. Constraints (3) inactivate all descendants of an inactived node. Constraints (4) ensure that one and only one class is assigned to each opened node. The second part of the model follows the path of the data in the tree. Constraints (5) and (6) open a leaf only if at least β data reach it. Constraints (7) ensure that each data is assigned to one and only one leaf, and Constraints (8) - (9) that the path of each data is consistent with the split functions. The last set of constraints counts the number of misclassifications L_ℓ (Constraints (12) and (14)), by first fixing the counting variables $N_{k,\ell}$ and N_ℓ with Constraints (10) and (11), respectively. Note that since the misclassifications are minimized, an optimal solution always assigns to a leaf ℓ the most represented class among the data reaching ℓ.

As an example, Figure 2a represents the optimal solution of (T) for the training set represented in Figure 2b. Each split function on a feature $j \in \mathcal{J}$ is represented by a grey line of thickness μ_j. In this example, we have $\delta = 3$, $\alpha = 0.1$ and $\beta = 4$, and the optimal tree makes 2 misclassifications. Observe that the missclassified red square could be properly classified by adding a split function on node 2, but this could be seen as overfitting. This solution is thus cut off by Constraint (5) since $\beta = 4$.



(a) Value of variables a, b, c, and d that are non-zero in the solution.



(b) The training set. For any point in $[0,1]^2$, the background color corresponds to the class predicted by the decision tree.

Figure 2: Optimal tree obtained by $(T_{0.1,\,4,\,3})$ on a training set containing 3 classes, 2 features and 34 data.

2.1.2 The oblique case

An extension to the oblique case was also proposed in [Bertsimas and Dunn, 2017] which allows split functions that use a linear combination of more than one feature. We label the associated model as $(T^H_{\alpha,\beta,\delta,\mu})$ since oblique splits are in fact hyperplanes.

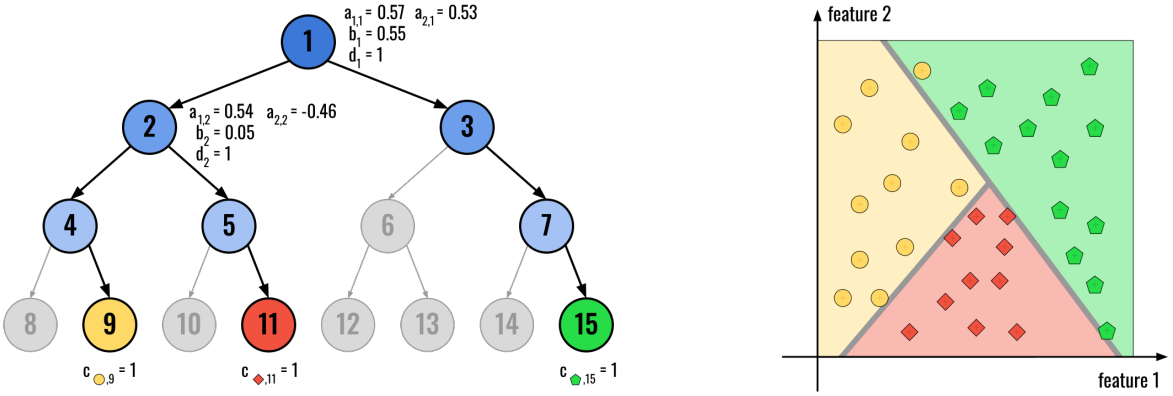
We present the differences with Formulation $(T_{\alpha,\beta,\delta})$. In this extension, variables $a_{j,t}$ are now defined in $[-1,1]$ rather than in $\{0,1\}$. As in the previous model, a node $t \in \mathcal{N}$ is inactive if and only if its split function only has null coefficients. To model it, the authors use the constraints $\sum_{j\in\mathcal{J}} |a_{j,t}| \leq d_t$ that they linearized with auxiliary variables $\hat{a}_{j,t} = |a_{j,t}|$. Then, the strict inequalities of the split functions are now handled by a new parameter $\mu \in \mathbb{R}$. Fixing a relevant value to parameter μ is a difficult task. If it is too small, in particular smaller than the accuracy of the solver, it may cause numerical issues. Note that in this formulation, it is not possible to generate split functions within the interval $[b_t - \mu, b_t]$, so if μ is chosen too large this may affect the objective value of an optimal solution. Finally, since a split function can now involve up to $|\mathcal{J}|$ non-zero coefficients, minimizing the number of active node is not a relevant second objective anymore. Instead, the authors minimize the number of non-zero coefficients $\{a_{j,t}\}_{j\in\mathcal{J},\,t\in\mathcal{N}}$ using auxiliary binary variables $s_{j,t}$ equal to 1 if and only if $a_{j,t} \neq 0$. Formulation $(T^H_{\alpha,\beta,\delta,\mu})$ is the

following:

$$
(T^H_{\alpha,\beta,\delta,\mu}) \begin{cases}
\min \dfrac{1}{\hat{L}} \displaystyle\sum_{\ell \in \mathcal{L}} L_\ell + \alpha \sum_{t \in \mathcal{N}, j \in \mathcal{J}} s_{j,t} & \\
\text{s.t.} (3)-(7), (10)-(14), (16) & \\
\displaystyle\sum_{j \in \mathcal{J}} \hat{a}_{j,t} \leq d_t & t \in \mathcal{N} \quad (17) \\
-\hat{a}_{j,t} \leq a_{j,t} \leq \hat{a}_{j,t} & j \in \mathcal{J}, t \in \mathcal{N} \quad (18) \\
-s_{j,t} \leq a_{j,t} \leq s_{j,t} & j \in \mathcal{J}, t \in \mathcal{N} \quad (19) \\
s_{j,t} \leq d_t & j \in \mathcal{J}, t \in \mathcal{N} \quad (20) \\
\displaystyle\sum_{j \in \mathcal{J}} s_{j,t} \geq d_t & t \in \mathcal{N} \quad (21) \\
-d_t \leq b_t \leq d_t & t \in \mathcal{N} \quad (22) \\
\displaystyle\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} + \mu \leq b_t + (2+\mu)(1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_L(\ell) \quad (23) \\
\displaystyle\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t - 2(1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_R(\ell) \quad (24) \\
s_{j,t} \in \{0,1\}, d_t \in \{0,1\} & t \in \mathcal{N}, j \in \mathcal{J} \quad (25)
\end{cases}
$$

In $(T^H_{\alpha,\beta,\delta,\mu})$, Constraints (17) and (18) ensure that $\hat{a}_{j,t} = |a_{j,t}|$. The definition of variables $s_{j,t}$ and their link with d_t are enforced by constraints (19) - (21). Constraints (22) enables variable b_t to be non-positive, and Constraints (23) and (24) define oblique split functions.

Figure 3a represents an optimal solution of (T^H) on the training set of Figure 3b (same as in Figure 2b). In this example, we have $\mu = 0.02$, $\delta = 3$, $\alpha = 0.04$ and $\beta = 4$, and all split functions have a thickness of μ. Observe that for the same value of parameters δ, α and β, and for $\mu \leq \min_{j \in \mathcal{J}} \mu_j$, all the feasible solutions of $(T_{\alpha,\beta,\delta})$ are always feasible for $(T^H_{\alpha,\beta,\delta,\mu})$.



(a) Value of variables a, b, c, and d that are non-zero in the solution.

(b) The training set.

Figure 3: Optimal tree obtained by $(T^H_{0.04,\,4,\,3,\,0.02})$ on a training set containing 3 classes, 2 features and 34 data.

2.1.3 Learning parameters α, β, δ

Formulations $(T_{\alpha,\beta,\delta})$ and (T^H) depend on 3 parameters: the depth of the tree δ, the minimal number of data in a leaf β, and the weight α of the second objective that penalizes the complexity of the tree. In [Bertsimas and Dunn, 2017], the authors propose an algorithm called `TreeTraining` and represented in Algorithm 1 that fits the parameters and allows to train the decision trees. Its input are the considered model $M \in \{(T), (T^H)\}$, $\overline{\delta}$ the maximal depth of the tree, and a fixed β (the minimal number of data points per open leaf). The data are split in 3 sets: the training set, the test set and the validation set. Model M is then solved on the training set for different values of the parameters. Finally, the best parameters are determined through the validation set. Since parameter α is continuous, the algorithm can not iterate on all its possible values. To overcome this, the authors remove the

second objective and bound the total number of non-zero coefficients in the split functions by $C \in \mathbb{Z}^+$ through one of the following constraints depending on the type of split functions considered:

$$\sum_{j,t \in \mathcal{J} \times \mathcal{N}} a_{j,t} \leq C \tag{26}$$

$$\sum_{j,t \in \mathcal{J} \times \mathcal{N}} s_{j,t} \leq C \tag{27}$$

Note that in the oblique case the number of iterations is significantly higher. Indeed, in that case $\overline{C}_\delta$, the highest possible value for C, is equal to $(2^\delta - 1)|\mathcal{J}|$ while it is only equal to $2^\delta - 1$ in the axis-aligned case. In Section 5.2 we introduce a new version of this algorithm that enables to reduce the number of iterations.

Algorithm *TreeTraining*$(M, \overline{\delta}, \beta)$
- $\mathcal{S} \leftarrow \emptyset$
- **for** $\delta = 1$ *to* $\overline{\delta}$ **do**
  - **for** $C = 1$ *to* $\overline{C}_\delta$ **do**
    - $\hat{T} \leftarrow$ a feasible solution for warm starting (given by CART, or set $\mathcal{S}$).
    - $T \leftarrow$ an optimal solution of M with $\alpha = 0$, Constraint (26) or (27), and $\hat{T}$ as a warm start.
    - $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
- Remove within $\mathcal{S}$ all dominated solutions for M.
- $T^* \leftarrow$ the tree in $\mathcal{S}$ that best performs on the validation set.
- **return** T^*

Algorithm 1: Algorithm `TreeTraining` of [Bertsimas and Dunn, 2017] to train a tree, originally designed for models $M \in \{(T), (T^H)\}$

2.2 A Strong Max-Flow Formulation [Aghaei et al., 2020]

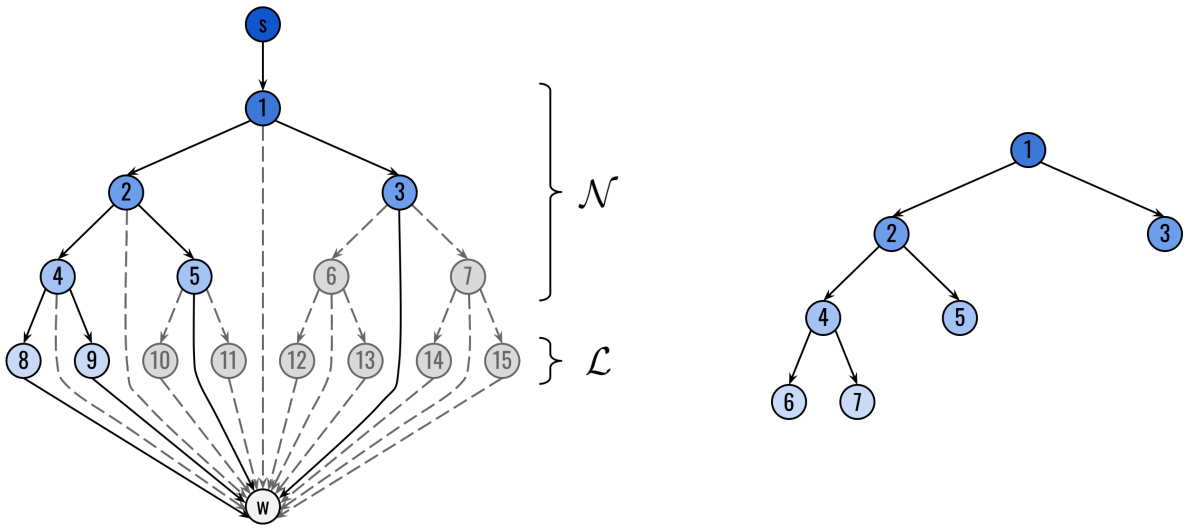
We now present an alternative linear program that is based on max-flow formulation [Aghaei et al., 2020]. It only handles binary features vectors (i.e. $X_i \in \{0,1\}^{\mathcal{J}}$), and axis-aligned split functions. The main idea of this formulation is to associate a unitary flow to each data i that is correctly classified.

The authors first define a flow network $\mathcal{G} = (\mathcal{V} = \mathcal{N} \cup \mathcal{L} \cup \{s, w\}, \mathcal{E} = E \cup \{(s,r), (t,w)_{t \in \mathcal{N} \cup \mathcal{L}}\})$, where a source s is connected by an arc (s,r) to the root r of the tree, and a sink w is connected by an arc (t,w) to all nodes $t \in \mathcal{N} \cup \mathcal{L}$ (see example in Figure 4a). In this model any node $t \in \mathcal{N} \cup \mathcal{L}$ can predict a class $k \in \mathcal{K}$, and the binary variable $g_{k,t}$ indicates if node t is assigned to class k. Then, to model the split functions, a binary variable $h_{j,t}$ states if the split function of node $t \in \mathcal{N}$ is performed on feature $j \in \mathcal{J}$. If it is, a data i reaching node t will go to its left child $l(t)$ if $x_{i,j} = 0$, and to its right child $r(t)$ if $x_{i,j} = 1$. Let P_i, be the path of a data $i \in \mathcal{I}$ in $\mathcal{G}$. A binary flow $\{u^i_{v,v'}\}_{(v,v') \in \mathcal{E}}$ is associated to i, and is equal to 1 if and only if i is correctly classified and $(v,v') \in P_i$. The class assigned to i being that of the direct ancestor of w in P_i.

The objective function is similar to that of $(T_{\alpha,\beta,\delta})$, but the parameter which weights the criterion is a scalar $\lambda \in [0,1[$. As in model $(T_{\alpha,\beta,\delta})$, the depth of the tree δ is a parameter of this formulation, but there is no parameter β controlling the minimal number of data reaching a leaf. Formulation of is as follows:

$$
(F^b_{\delta,\lambda})\begin{cases}
\max\,(1-\lambda)\sum_{i\in\mathcal{I}} u^i_{s,r} - \lambda\sum_{t\in\mathcal{N}}\sum_{j\in\mathcal{J}} h_{j,t} & & \\
\text{s.t.}\ \sum_{j\in\mathcal{J}} h_{j,t} + \sum_{k\in\mathcal{K}} g_{k,t} = 1 & t\in\mathcal{N} & (28)\\
\sum_{k\in\mathcal{K}} g_{k,\ell} = 1 & \ell\in\mathcal{L} & (29)\\
u^i_{a(t),t} = u^i_{t,l(t)} + u^i_{t,r(t)} + u^i_{t,w} & t\in\mathcal{N}, i\in\mathcal{I} & (30)\\
u^i_{a(\ell),\ell} = u^i_{\ell,w} & \ell\in\mathcal{L}, i\in\mathcal{I} & (31)\\
u^i_{t,l(t)} \le \sum_{j\in\mathcal{J}:x_{i,j}=0} h_{j,t} & t\in\mathcal{N}, i\in\mathcal{I} & (32)\\
u^i_{t,r(t)} \le \sum_{j\in\mathcal{J}:x_{i,j}=1} h_{j,t} & t\in\mathcal{N}, i\in\mathcal{I} & (33)\\
u^i_{t,w} \le g_{y_i,t} & i\in\mathcal{I}, t\in\mathcal{N}\cup\mathcal{L} & (34)\\
h_{j,t}\in\{0,1\} & t\in\mathcal{N}, j\in\mathcal{J} & (35)\\
g_{k,t}\in\{0,1\} & t\in\mathcal{N}\cup\mathcal{L}, k\in\mathcal{K} & (36)\\
u^i_{t,t'}\in\{0,1\} & i\in I, (t,t')\in\mathcal{E} & (37)
\end{cases}
$$

Constraints (28) ensure that each node either performs a split or predicts a class, and Constraints (29) that one and only one class is assigned to each leaf. Constraints (30) and (31) hold for the flow conservation. Constraints (32) and (33) ensure the consistency of the split functions. Constraints (34) impose that the flow of a misclassified data vanishes. Note that when a node $t \in \mathcal{N}$ is inactive, any data reaching it necessarily follows the arc (t, w). This is illustrated in the flow network represented in Figure 4a. In this solution of (F^b), nodes 3 and 5 are inactive and, consequently, the flow of the data cannot go through arcs $(3, 6)$, $(3, 7)$, $(5, 10)$ and $(5, 11)$. This leads to a decision tree which structure is represented in Figure 4b.



(a) The flow network. Grayed arcs and vertices are not reached by any training data.

(b) Structure of the corresponding decision tree.

Figure 4: Link between the flow of the data in the graph of a solution of Formulation (F^b) and the structure of the decision tree obtained.

In [Aghaei et al., 2020], it is proven that $(F^b_{\delta,\lambda})$ has a stronger relaxation than a restriction of $(T_{\alpha,\beta,\delta})$ to the case of binary feature vectors (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$).

3 A new quadratic formulation based on $(T_{\alpha,\beta,\delta})$

In this section, we introduce a new compact formulation for OCT where a quadratic formulation of the objective function allows us to remove all the constraints and variables relative to the counting of the missclassifications. More precisely, since an error of classification comes from the fact that a data reaches a leaf with a different class, it can be modeled with the product of the binary variable that assigns a class to a leaf by the one that follows the path of the data in the tree. With this new quadratic writing we do not need the error-counting variables of [Bertsimas and Dunn, 2017] (i.e. N_ℓ, $N_{k,\ell}$ and L_ℓ) anymore. Then, by considering two distinct linearization techniques, we deduce two linear reformulations that are stronger than $(T_{\alpha,\beta,\delta})$ and $(T^H_{\alpha,\beta,\delta,\mu})$ from the continuous relaxation point of view.

3.1 A quadratic formulation for the axis-aligned case

In $(T_{\alpha,\beta,\delta})$, variables L_ℓ that model the number of misclassified data reaching leaf $\ell \in \mathcal{L}$, are fixed thanks to N_ℓ, the total number of data reaching leaf ℓ, and $N_{k,\ell}$, the number of data of class $k \in \mathcal{K}$ reaching leaf ℓ. More formally:

$$L_\ell = N_\ell - \max_{k \in \mathcal{K}} N_{k,\ell} \tag{38}$$

This equality is true since the class assigned to leaf ℓ in an optimal solution of $(T_{\alpha,\beta,\delta})$ is precisely $\arg\max_{k \in \mathcal{K}} N_{k,\ell}$. In $(T_{\alpha,\beta,\delta})$, Equation (38) is linearized thanks to Constraints (10)-(14).

We introduce a new writting of variable L_ℓ in order to define a more compact formulation. Recall that variables $c_{k,\ell} = 1$ if class $k \in \mathcal{K}$ is assigned to leaf $\ell \in \mathcal{L}$, and variables $z_{i,\ell} = 1$ if data i reaches leaf ℓ. For a given $k \in \mathcal{K}$, let $\mathcal{I}_k$ be the subset of $\mathcal{I}$ restricted to data of class k (i.e., $\mathcal{I}_k = \{i \in \mathcal{I} \mid y_i = k\}$). Variable L_ℓ can be expressed as:

$$L_\ell = \sum_{k \in \mathcal{K}} c_{k,\ell} \left(N_\ell - N_{k,\ell}\right) = \sum_{k \in \mathcal{K}} (c_{k,\ell} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z_{i,\ell}) = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} c_{k,\ell} z_{i,\ell} \tag{39}$$

We propose to use Equation (39) to model the number of misclassified data, which we directly include into the objective function. Doing this allows us to reduce the size of the formulation by removing variables L_ℓ, N_ℓ, $N_{k,\ell}$ and Constraints (10)-(14). We obtain the following quadratic formulation:

$$(Q_{\alpha,\beta,\delta}) \begin{cases} \min & F_{\hat{L},\alpha}(c,z,d) = \dfrac{1}{\hat{L}} \displaystyle\sum_{\ell \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} c_{k,\ell} z_{i,\ell} + \alpha \sum_{t \in \mathcal{N}} d_t \\ \text{s.t.} & (1)-(9), (15), (16) \\ & l_\ell \leq d_t \qquad\qquad \ell \in \mathcal{L},\ t \in A_L(\ell) \end{cases} \tag{40}$$

where Constraints (40) are valid inequalities that enable strengthen our formulation. Note that this is not the case for $(T_{\alpha,\beta,\delta})$ since these valid inequalities do not impact the value of its continuous relaxation. The quadratic objective function $F_{\hat{L},\alpha}$ of $(Q_{\alpha,\beta,\delta})$ is a non-convex function which we propose to convexify through two different approaches: the linearization of Fortet [Fortet, 1960] and that of Glover [Glover, 1975]. Further, we compare the continuous relaxation of the two corresponding linear programs.

Fortet's linearization [Fortet, 1960] consists in replacing each bi-linear term $z_{i,\ell}c_{k,\ell}$ by an auxiliary variable $\theta_{i,k,\ell}$. The equality $\theta_{i,k,\ell} = z_{i,\ell}c_{k,\ell}$ is then enforced by the following sets of linear inequalities:

$$\begin{cases} \theta_{i,k,\ell} \leq c_{k,\ell} & i \in \mathcal{I} \setminus \mathcal{I}_k,\ k \in \mathcal{K}, \ell \in \mathcal{L} \quad (41) \\ \theta_{i,k,\ell} \leq z_{i,\ell} & i \in \mathcal{I} \setminus \mathcal{I}_k,\ k \in \mathcal{K}, \ell \in \mathcal{L} \quad (42) \\ \theta_{i,k,\ell} \geq c_{k,\ell} + z_{i,\ell} - 1 & i \in \mathcal{I} \setminus \mathcal{I}_k,\ k \in \mathcal{K}, \ell \in \mathcal{L} \quad (43) \\ \theta_{i,k,\ell} \geq 0 & i \in \mathcal{I} \setminus \mathcal{I}_k,\ k \in \mathcal{K}, \ell \in \mathcal{L} \quad (44) \end{cases}$$

It is easy to verify that inequalities (41)-(44) are equivalent to $\theta_{i,k,\ell} = z_{i,\ell}c_{k,\ell}$ as $z_{i,\ell}$ and $c_{k,\ell}$ are binary variables. Moreover, since in $(Q_{\alpha,\beta,\delta})$, the products $z_{i,\ell}c_{k,\ell}$ are only involved into the objective function, and are weighted

with non-negative coefficients, Constraints (41) and (42) are not necessary to get the equivalence. We finally obtain the following linear reformulation of $(Q_{\alpha,\beta,\delta})$:

$$(QF_{\alpha,\beta,\delta}) \begin{cases} \min & FL^1_{\hat{L},\alpha}(\theta,d) = \dfrac{1}{\hat{L}} \displaystyle\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta_{i,k,\ell} + \alpha \sum_{\ell\in\mathcal{N}} d_\ell \\ \text{s.t.} & (1)-(9),(15),(16),(40),(43),(44) \end{cases}$$

To further reduce the size of the formulation, we propose to use Glover's procedure [Glover, 1975] to linearize the products $c_{k,\ell}\Big(\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}\Big)$ by use of auxiliary variables $\Theta_{k,\ell}$. It is easy to prove that $|\mathcal{I}\setminus\mathcal{I}_k|$ is a valid upper bound for $\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}$. Using these bounds and the fact that $\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} \geq 0$, the equality $\Theta_{k,\ell} = c_{k,\ell}\Big(\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}\Big)$ is then enforced by the following set of linear inequalities:

$$\begin{cases} \Theta_{k,\ell} \leq |\mathcal{I}\setminus\mathcal{I}_k| c_{k,\ell} & k\in\mathcal{K},\ \ell\in\mathcal{L} & (45) \\ \Theta_{k,\ell} \leq \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} & k\in\mathcal{K},\ \ell\in\mathcal{L} & (46) \\ \Theta_{k,\ell} \geq 0 & k\in\mathcal{K},\ \ell\in\mathcal{L} & (47) \\ \Theta_{k,\ell} \geq \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} - |\mathcal{I}\setminus\mathcal{I}_k|(1-c_{k,\ell}) & k\in\mathcal{K},\ \ell\in\mathcal{L} & (48) \end{cases}$$

Here again we only need inequalities (47) and (48) to get an equivalent linear reformulation:

$$(QG_{\alpha,\beta,\delta}) \begin{cases} \min & FL^2_{\hat{L},\alpha}(\Theta,d) = \dfrac{1}{\hat{L}} \displaystyle\sum_{k\in K}\sum_{\ell\in\mathcal{L}} \Theta_{k,\ell} + \alpha \sum_{t\in\mathcal{N}} d_t \\ \text{s.t.} & (1)-(9),(15),(16),(40),(47),(48) \end{cases}$$

We now compare our two linear reformulations of $(Q_{\alpha,\beta,\delta})$. We first observe that the size of $(QG_{\alpha,\beta,\delta})$ (i.e. $\mathcal{O}(|\mathcal{K}|\times|\mathcal{L}|)$ auxiliary variables and constraints) is reduced by a factor of $|\mathcal{I}|$ in comparison to the size of $(QF_{\alpha,\beta,\delta})$ (i.e. $\mathcal{O}(|\mathcal{K}|\times|\mathcal{L}|\times|\mathcal{I}|)$ auxiliary variables and constraints). Since $|\mathcal{I}|$ is higher than both $|\mathcal{K}|$ and $|\mathcal{L}|$ in non-trivial problems, this reduction can be significant. Let us denote by $v(P)$ the optimal value of a problem (P), and $\overline{P}$ the continuous relaxation of problem P. We prove in Proposition 1 that problem $(QF_{\alpha,\beta,\delta})$ has a better continuous relaxation than problem $(QG_{\alpha,\beta,\delta})$.

Proposition 1. $v(\overline{QF}_{\alpha,\beta,\delta}) \geq v(\overline{QG}_{\alpha,\beta,\delta})$.

Proof. Let $(d^*,a^*,b^*,l^*,c^*,z^*,\theta^*)$ be an optimal solution of $(\overline{QF}_{\alpha,\beta,\delta})$. We build a feasible solution $(d=d^*, a=a^*, b=b^*, l=l^*, c=c^*, z=z^*, \Theta)$ of $(\overline{QG}_{\alpha,\beta,\delta})$ such that $\Theta_{k,\ell} = \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta^*_{i,k,\ell}$ for all $(\ell,k)\in(\mathcal{L}\times\mathcal{K})$. The only constraints of $(\overline{QG}_{\alpha,\beta,\delta})$ which are not obviously satisfied by this solution are Constraints (48). Let us prove that they are satisfied:

$$\begin{array}{rcl} \Theta_{k,\ell} = \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta^*_{i,k,\ell} & \geq & \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \Big(c^*_{k,\ell} + z^*_{i,\ell} - 1\Big) \\ & \geq & |\mathcal{I}\setminus\mathcal{I}_k| c_{k,\ell} + \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z^*_{i,\ell} - |\mathcal{I}| + |\mathcal{I}_k| \\ & \geq & \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} - |\mathcal{I}\setminus\mathcal{I}_k|(1-c_{k,\ell}) \end{array}$$

We now compare the objective values $\Delta = FL^2_{\hat{L},\alpha}(\Theta,d) - FL^1_{\hat{L},\alpha}(\theta^*,d^*)$:

$$\Delta = \frac{1}{\hat{L}}\sum_{k\in K}\sum_{\ell\in\mathcal{L}}\Theta_{k,\ell} - \frac{1}{\hat{L}}\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell} = \frac{1}{\hat{L}}\Big(\sum_{k\in K}\sum_{\ell\in\mathcal{L}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell} - \sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}\Big) = 0$$

□

We want to compare the value of $(\overline{QG}_{\alpha,\beta,\delta})$ and $(\overline{QF}_{\alpha,\beta,\delta})$ to the value of $(\overline{T}_{\alpha,\beta,\delta})$. To do so, we first show that the value of $(\overline{T}_{\alpha,\beta,\delta})$ is always 0.

Lemma 1. $v(\overline{T}_{\alpha,\beta,\delta}) = 0$

Proof. Since all the variables are positive and weighted by positive constants, necessarily $v(\overline{T}_{\alpha,\beta,\delta}) \geq 0$. Let ℓ_r denote the right-most leaf of the tree. We build the following solution of value 0:

- $a_{j,t} = b_t = d_t = 0 \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $z_{i,\ell_r} = l_{\ell_r} = 1$, and $z_{i,\ell} = l_\ell = 0 \quad \forall i \in \mathcal{I}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $N_{\ell_r} = |\mathcal{I}|$, and $N_\ell = 0 \quad \forall \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $N_{k,\ell_r} = |\mathcal{I}_k|$, and $N_{k,\ell} = 0 \quad \forall k \in \mathcal{K}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $c_{k,\ell_r} = \frac{|\mathcal{I}_k|}{|\mathcal{I}|}$, and $c_{k,\ell_r} = 0 \quad \forall k \in \mathcal{K}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $L_\ell = 0 \quad \forall \ell \in \mathcal{L}$.

Constraints (1)-(7) and (10)-(14) are obviously satisfied, and Constraints (8) and (9) are verified since all data go to the right-most leaf. The value of this solution is 0. □

As mentioned before, the valid inequalities (40) do not change the value of the continuous relaxation of $(T_{\alpha,\beta,\delta})$. The solution provided by the proof satisfies (40).

We state in Proposition 2 that our new linear formulations provide better continuous relaxation bounds than the one of $(T_{\alpha,\beta,\delta})$.

Proposition 2. *If* $\alpha > 0$, $v(\overline{QF}_{\alpha,\beta,\delta}) \geq v(\overline{QG}_{\alpha,\beta,\delta}) > v(\overline{T}_{\alpha,\beta,\delta})$.

Proof. Considering Lemma 1, it remains to prove that $v(\overline{QG}_{\alpha,\beta,\delta}) > 0$. Let us distinguish two cases:

Case 1: If $\sum_{t \in \mathcal{N}} d_t > 0$, since α is positive, we necessarily have $v(\overline{QG}_{\alpha,\beta,\delta}) > 0$.

Case 2: If $\sum_{t \in \mathcal{N}} d_t = 0$, we have $v(\overline{QG}_{\alpha,\beta,\delta}) = \frac{1}{L} \sum_{k \in K} \sum_{\ell \in \mathcal{L}} \Theta_{k,\ell}$. Constraints (1) and (2) impose for every node t and feature j that $a_{j,t} = b_t = 0$. By Constraints (40), we have that for all $\ell \in \mathcal{L} \setminus \{\ell_r\}$ the value of $z_{i,\ell}$ is 0 and $z_{i,\ell_r} = 1$. Therefore, $l_\ell = 1$ if and only if $\ell = \ell_r$, giving us $\sum_{k \in \mathcal{K}} c_{k,\ell_r} = 1$.

Since $(\overline{QG}_{\alpha,\beta,\delta})$ is a minimisation problem and the coefficients of $\theta_{\ell,k}$ in FL^2 are positive, Constraints (47) and (48) enforce $\Theta_{\ell,k} = 0 \; \forall \ell \neq \ell_r$. For ℓ_r we obtain:

$$\Theta_{\ell_r,k} = \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z_{i,\ell_r} - |\mathcal{I} \setminus \mathcal{I}_k|(1 - c_{k,\ell_r}) = |\mathcal{I} \setminus \mathcal{I}_k| \, c_{k,\ell_r}$$

Thus, $v(\overline{QG}_{\alpha,\beta,\delta}) = \sum_{k \in \mathcal{K}} |\mathcal{I} \setminus \mathcal{I}_k| c_{k,\ell_r} > 0$ since $\sum_{k \in \mathcal{K}} c_{k,\ell_r} = 1$.

□

3.2 Handling the *oblique* case

Equation (39) remains true for *oblique* split functions, and thus the oblique extension of our three formulations is straightforward, since the differences are the same than those between $(T_{\alpha,\beta,\delta})$ and $(T^H_{\alpha,\beta,\delta,\mu})$. Thus, we build $(Q^H_{\alpha,\beta,\delta,\mu})$, $(QF^H_{\alpha,\beta,\delta,\mu})$ and $(QG^H_{\alpha,\beta,\delta,\mu})$ the oblique extensions of $(Q_{\alpha,\beta,\delta})$, $(QF_{\alpha,\beta,\delta})$ and $(QG_{\alpha,\beta,\delta})$, respectively:

$$(Q^H_{\alpha,\beta,\delta,\mu}) \begin{cases} \min \quad F^H_{\hat{L},\alpha}(c,z,s) = \frac{1}{\hat{L}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} c_{k,\ell} z_{i,\ell} + \alpha \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{N}} s_{j,t} \\ s.t. \quad (3)-(7)(16)-(25) \end{cases}$$

$$(QF^H_{\alpha,\beta,\delta,\mu}) \begin{cases} \min \quad FL^{1H}_{\hat{L},\alpha}(\theta,s) = \frac{1}{\hat{L}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta_{i,k,\ell} + \alpha \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{N}} s_{j,t} \\ s.t. \quad (3)-(7)(16)-(25)(43)-(44) \end{cases}$$

$$(QG^H_{\alpha,\beta,\delta,\mu}) \begin{cases} \min \quad FL^{1H}_{\hat{L},\alpha}(\Theta,s) = \frac{1}{\hat{L}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} \Theta_{k,\ell} + \alpha \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{N}} s_{j,t} \\ s.t. \quad (3)-(7)(16)-(25)(47)-(48) \end{cases}$$

Similarly to the *axis-aligned* case, we compare in Proposition 3 the continuous relaxation values of $(QF^H_{\alpha,\beta,\delta,\mu})$, $(QG^H_{\alpha,\beta,\delta,\mu})$, and $(T^H_{\alpha,\beta,\delta,\mu})$, denoted by $(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$, $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$, and $(\overline{T}^H_{\alpha,\beta,\delta,\mu})$, respectively. To do so, we show that the value of $(\overline{T}^H_{\alpha,\beta,\delta,\mu})$ is always 0.

Lemma 2. $v(\overline{T}^H_{\alpha,\beta,\delta,\mu}) = 0$

Proof. Since all the variables are positive and weighted by positive constants, we have $v(\overline{T}^H_{\alpha,\beta,\delta,\mu}) \geq 0$. Let t_r denote the right-most leaf, we build the following solution of value 0:

- $a_{j,t} = \hat{a}_{j,t} = s_{j,t} = b_t = d_t = 0 \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $z_{i,\ell_r} = l_{\ell_r} = 1$, and $z_{i,\ell} = l_\ell = 0 \quad \forall i \in \mathcal{I}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $N_{\ell_r} = |\mathcal{I}|$, and $N_\ell = 0 \quad \forall \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $N_{k,\ell_r} = |\mathcal{I}_k|$, and $N_{k,\ell} = 0 \quad \forall k \in \mathcal{K}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $c_{k,\ell_r} = \frac{\mathcal{I}_k}{|\mathcal{I}|}$, and $c_{k,\ell_r} = 0 \quad \forall k \in \mathcal{K}, \ell \in \mathcal{L}, \ell \neq \ell_r$;
- $L_\ell = 0 \quad \forall \ell \in \mathcal{L}$.

Constraints (3)-(7), (10)-(14), (17)-(22) are obviously satisfied, and Constraints (23)-(24) are verified since all data go to the right-most leaf. □

We state in Proposition 3 that our new linear formulations provide better continuous relaxation bounds than the original formulation $(T^H_{\alpha,\beta,\delta,\mu})$.

Proposition 3. *If* $\alpha > 0$, $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu}) \geq v(\overline{QG}^H_{\alpha,\beta,\delta,\mu}) > v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$.

Proof. We first prove that $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu}) \geq v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$.
The proof is similar to that of Proposition 1. Let $(d^*, a^*, b^*, l^*, c^*, z^*, s^*, \theta^*)$ be an optimal solution of $(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$. We build a feasible solution to $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$: $(d = d^*, a = a^*, b = b^*, l = l^*, c = c^*, z = z^*, s = s^*, \Theta)$, with for

all $(\ell, k) \in (\mathcal{L} \times \mathcal{K})$ $\Theta_{k,\ell} = \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} \theta^*_{i,k,\ell}$. This solution obviously satisfies all constraints of $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$ and its objective value is equal to $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$.

Secondly, we prove that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu}) > v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$. Considering Lemma 2, it remains to prove that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu}) > 0$. Let us distinguish two cases:

Case 1: If $\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{N}} s_{j,t} > 0$, since α is positive, we necessarily have $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu}) > 0$.

Case 2: If $\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{N}} s_{j,t} = 0$, it means that all data are sent to the right-most leaf. With a similar reasoning as the proof of Proposition 2, we get that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu}) > 0$.

□

4 Extension of $(F^b_{\delta,\lambda})$ to the case of real-valued data

Formulation $(F^b_{\delta,\lambda})$ only applies to datasets in which all features are binary (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$) and only consider axis-aligned split functions. We propose in this section to extend this formulation to the case of real-valued data (i.e. $X_i \in [0,1]^{|\mathcal{J}|}$) for both the axis-aligned and oblique cases.

4.1 The axis-aligned case

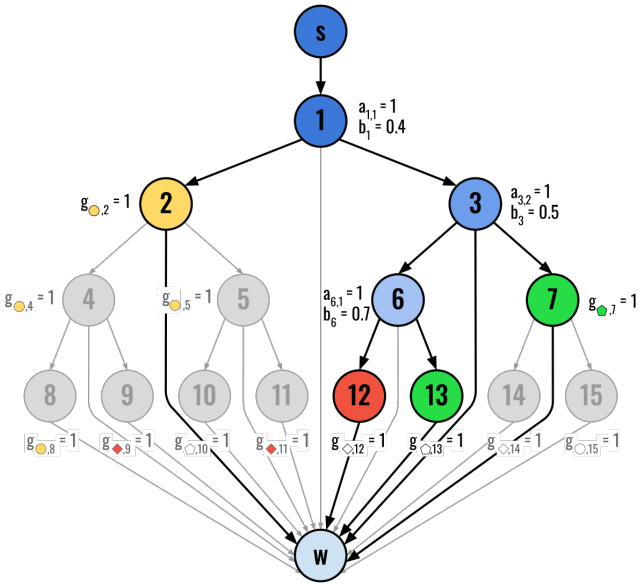
For our extension, we keep the unitary flow variables $u^i_{t,t'}$ that model whether data i is correctly classified and goes through arc (t,t'). Variables $h_{j,t}$ are not relevant anymore, we replace them by variables $a_{j,t}$ and b_t that model the split functions $\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t$ and $\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} < b_t$. The obtained formulation $(F_{\alpha,\delta})$ is the following:

$$
(F_{\alpha,\delta}) \begin{cases}
\min \quad f^F_{\hat{L},\alpha}(u,a) = \dfrac{1}{\hat{L}} \left(|\mathcal{I}| - \sum_{i \in \mathcal{I}} u^i_{s,r} \right) + \alpha \sum_{t \in \mathcal{N}} \sum_{j \in \mathcal{J}} a_{j,t} & \\
\text{s.t.} \quad (29)-(31), (34), (36), (37) & \\
\quad \sum_{j \in \mathcal{J}} a_{j,t} + \sum_{k \in \mathcal{K}} g_{k,t} = 1 & t \in \mathcal{N} \quad (49) \\
\quad 0 \leq b_t \leq \sum_{j \in \mathcal{J}} a_{j,t} & t \in \mathcal{N} \quad (50) \\
\quad \sum_{j \in \mathcal{J}} a_{j,t} \left(x_{i,j} + \mu_j - \mu^- \right) + \mu^- \leq b_t + (1+\mu^+)(1 - u^i_{t,l(t)}) & t \in \mathcal{N}, i \in \mathcal{I} \quad (51) \\
\quad \sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t - (1 - u^i_{t,r(t)}) & t \in \mathcal{N}, i \in \mathcal{I} \quad (52) \\
\quad u^i_{t,l(t)} \leq \sum_{j \in \mathcal{J}} a_{j,t} & i \in \mathcal{I}, t \in \mathcal{N} \quad (53) \\
\quad u^i_{t,r(t)} \leq \sum_{j \in \mathcal{J}} a_{j,t} & i \in \mathcal{I}, t \in \mathcal{N} \quad (54) \\
\quad a_{j,t} \in \{0,1\} & t \in \mathcal{N}, j \in \mathcal{J} \quad (55)
\end{cases}
$$

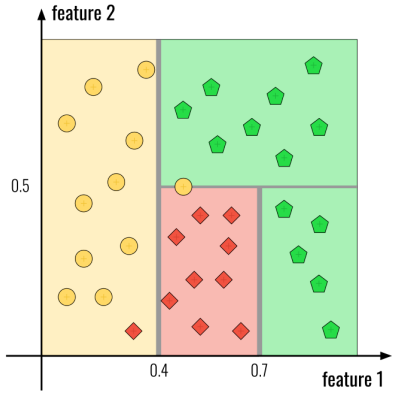
We keep in this extension Constraints (29) which force each leaf to be associated to a class, as well as the flow conservation constraints (30)-(31). As in $(F^b_{\delta,\lambda})$, a misclassified data will have a null flow by Constraints (34). Then, we adapt the capacity constraints (32)-(33) with Constraints (53)-(54). Note that Constraints (53) are redundant but are valid inequalities in the linear relaxation. Finally, to model the split functions, we rely on Formulation $(T_{\alpha,\beta,\delta})$ by adapting Constraints (8)-(9) to our case, where the difference is that variables $z_{i,\ell}$ are replaced by variables $u^i_{t,\ell(t)}$ ($u^i_{t,r(t)}$ respectively) in (51) ((52) resp.). We also adapt Constraints (28) into Constraints (49), that ensure that a node either predicts a class or performs a split. In this last case, these constraints additionally force

$\sum_{j\in\mathcal{J}} a_{j,t}$ to be equal to 1. For the objective function, we have chosen to use the same parameters as in $(T_{\alpha,\beta,\delta})$, so we minimize a function weighted by parameters $\hat{L}$ and α.

Figure 5a represents an optimal solution of (F), for parameters $\alpha = 0.1$, and $\delta = 4$, on the same training set considered in Figures 2 and 3. Note that the missclassified right-most circle in Figure 5b is located on a split function of thickness μ_2 (i.e. $x_{i,2} \in]0.5 - \mu_2, 0.5[$). This is possible in (F) since the flow of any misclassified data $i \in \mathcal{I}$ is null which disables the Constraints (51) and (52). Observe that the same split function is not feasible for (T) since each data is assigned to a path for which the branching rules must be satisfied.



(a) Value of variables a, b and g that are non-zero in the solution.



(b) The training set.

Figure 5: Optimal tree obtained from $(F_{0.1,4})$ on a training set containing 3 classes, 2 features, and 34 data.

The following proposition proves that, in the axis-aligned case, the optimal value of formulations $(T_{\alpha,\beta,\delta})$ and $(F_{\alpha,\delta})$ are identical.

Proposition 4. *If $\beta = 0$, $v(T_{\alpha,\beta,\delta}) = v(F_{\alpha,\delta})$.*

Proof. **1.** We first prove that $v(T_{\alpha,\beta,\delta}) \geq v(F_{\alpha,\delta})$. We denote by $Leaves(t)$ the set of leaves that can be reached from $t \in \mathcal{N}$ and by $leaf(t)$ the right-most leaf in $Leaves(t)$. Let $S_T = (a^*, b^*, c^*, d^*, l^*, L^*, N_t^*, N_{k,t}^*, z^*)$ be an optimal solution of $(T_{\alpha,\beta,\delta})$. Without loss of generality, let us assume that $d_t = 1$ if and only if node $t \in \mathcal{N}$ does split data, i.e. if $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(l(t))} z_{i,\ell} \geq 1$ and $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(r(t))} z_{i,\ell} \geq 1$. This is always possible since if there exists a node $t \in \mathcal{N}$ such that either $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(l(t))} z_{i,\ell} = 0$ or $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(r(t))} z_{i,\ell} = 0$ it is possible to set d_t to 0 without altering the predictions of the data. The solution remains optimal as the objective value cannot be increased by this transformation. We build a solution $S_F = (a, b, g, u)$ of $(F_{\alpha,\delta})$ as follows:

- $a_{j,t} = a^*_{j,t}$ and $b_t = b^*_t \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $u^i_{t,t'}$ is defined differently depending on $(t, t') \in \mathcal{E}$:
  - $u^i_{s,r} = \sum_{\ell\in\mathcal{L}} z^*_{i,\ell} c^*_{y_i,\ell} \quad \forall i \in \mathcal{I}$;
  - $u^i_{\ell,w} = u^i_{a(\ell),\ell} = d^*_{a(\ell)} z^*_{i,\ell} u^i_{s,r} \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
  - $u^i_{a(t),t} = d^*_{a(t)} u^i_{s,r} \sum_{\ell\in Leaves(t)} z^*_{i,\ell} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
  - $u^i_{t,w} = d^*_{a(t)}(1 - d^*_t) z^*_{i,leaf(t)} u^i_{s,r} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;

- $u^i_{r,w} = (1 - d^*_1) z^*_{i,leaf(r)} u^i_{s,r} \quad \forall i \in \mathcal{I}$;

- $g_{k,t}$ is defined differently depending on $t \in \mathcal{N} \cup \mathcal{L}$:
  - $g_{k,t} = (1 - d^*_t) d^*_{a(t)} c^*_{k,leaf(t)} + (1 - d^*_{a(t)}) \mathbf{1}[k = 1] \quad \forall t \in \mathcal{N} \setminus \{r\}, k \in \mathcal{K}$;
  - $g_{k,r} = (1 - d^*_1) c^*_{k,leaf(r)} \quad \forall k \in \mathcal{K}$;
  - $g_{k,\ell} = d^*_{a(\ell)} c^*_{k,\ell} + (1 - d^*_{a(\ell)}) \mathbf{1}[k = 1] \quad \forall \ell \in \mathcal{L}$.

Let us first note that the integrity of S_T. Moreover, Constraints (7) ensures the integrity of S_F, Constraints (31) are satisfied by definition, and Constraints (1) and (2) imply Constraints (50).

Constraints (51) **and** (52)**.** When $u^i_{t,l(t)}$ is equal to 0, the associated Constraint (49) is necessarily satisfied. From its definition we know that $u^i_{t,l(t)} = 1$ if and only if there exists a leaf $\ell \in Leaves(l(t))$ such that $z^*_{i,\ell} = 1$. In such cases Constraints (8) ensure that Constraints (51) are satisfied. The satisfaction of Constraints (52) can be proved similarly.

Constraints (29)**.** By definition of variables $g_{k,\ell}$, a Constraint (29) is satisfied if $d^*_{a(\ell)} = 0$. Otherwise, we know by assumption that there exists $i \in \mathcal{I}$ such that $z^*_{i,\ell} = 1$. From Constraints (4) and (6), we deduce that $\sum_{k \in \mathcal{K}} c^*_{k,\ell} = 1$ which leads to the same result.

Constraints (30)**.** By definition of variables $u^i_{t,t'}$, a Constraint (30) is satisfied if $d^*_{a(t)} = d^*_t$ or $u^i_{s,r} = 0$. Otherwise, because of Constraints (3), the only possible case is $d^*_{a(t)} = 1$, $d^*_t = 0$ and $u^i_{s,r} = 1$. By definition of variables $u^i_{t,t'}$, the right-hand side of the corresponding Constraint (30) is equal to $z^*_{i,leaf(t)}$ and because of Constraints (1) and (8), $z^*_{i,leaf(t)} = \sum_{\ell \in Leaves(t)} z^*_{i,\ell}$. As the left-hand side of the Constraint(30) is equal to $\sum_{\ell \in Leaves(t)} z^*_{i,\ell}$, by definition of variables $u^i_{t,t'}$, Constraints (30) are satisfied.

Constraints (49)**.** For a given $t \in \mathcal{N}$, if both $d^*_{a(t)}$ and d_t are equal to 1, the associated Constraint (47) is satisfied as $\sum_{k \in \mathcal{K}} g_{k,t} = 0$ by definition of $g_{k,t}$ and $\sum_{j \in \mathcal{J}} a_{j,t} = 1$ from Constraints (4) and (6). Otherwise, either $d^*_{a(t)}$ or d^*_t is equal to 0 and Constraints (1) and (3) ensure that $\sum_{j \in \mathcal{J}} a_{j,t} = 0$. If $d^*_{a(t)} = 0$, $\sum_{k \in \mathcal{K}} g_{k,t} = 1$ by definition of g. If $d^*_t = 0$, $\sum_{k \in \mathcal{K}} g_{k,t} = \sum_{k \in \mathcal{K}} c^*_{k,leaf(t)}$ which is also equal to 1. Thus, Constraints (49) are satisfied in all cases.

Constraints (53) **and** (54)**.** If $d^*_t = 1$, Constraints (53) and (54) are satisfied as Constraints (1) ensure that $\sum_{j \in \mathcal{J}} a_{j,t} = 1$. These constraints are also satisfied if $d^*_t = 0$ as it implies $u^i_{t,l(t)} = u^i_{t,r(t)} = 0$ by definition of these variables.

Constraints (34)**.** For any node $t \in \mathcal{N}$, these constraints are satisfied by definition of $u^i_{t,w}$ if either, $d_{a(t)} = 0$, $d_t = 1$, $z_{i,leaf(t)} = 0$ or $u^i_{s,r} = 0$. Otherwise, we know by definition of $g_{k,t}$ that $g_{k,t} = c^*_{k,leaf(t)}$ and by definition of $u^i_{s,r}$ that there exists $\overline{\ell} \in \mathcal{L}$ such that $z^*_{i,\overline{\ell}} = c^*_{y_i,\overline{\ell}} = 1$. Since $z_{i,leaf(t)} = 1$, we deduce from Constraints (7) that $\overline{\ell} = leaf(t)$. The associated Constraint (34) is also satisfied in that case as this leads to $g_{y_i,t} = 1$. A similar reasoning provides the same result for any $\ell \in \mathcal{L}$.

We now compare the objective values of S_T and S_F. Since the second criterion in both objectives are identical, we only compare the first criterion. For this, we consider the quadratic Equation (39):

$$L^*_\ell = \sum_{k \in \mathcal{K}} c^*_{k,\ell} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z^*_{i,\ell} \quad \forall \ell \in \mathcal{L}$$

The two solutions have the same value since:

$$\sum_{\ell \in \mathcal{L}} L^*_\ell = \sum_{\ell \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} z^*_{i,\ell} c^*_{k,\ell} - \sum_{\ell \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} z^*_{i,\ell} c^*_{k,\ell} = |\mathcal{I}| - \sum_{i \in \mathcal{I}} \sum_{\ell \in \mathcal{L}} z^*_{i,\ell} c^*_{y_i,\ell} = |\mathcal{I}| - \sum_{i \in \mathcal{I}} u^i_{s,r}$$

2. We now prove that $v(T_{\alpha,\beta,\delta}) \leq v(F_{\alpha,\delta})$. Let $S_F = (a^*, b^*, g^*, u^*)$ be an optimal solution of $(F_{\alpha,\delta})$. Without loss of generality, let us assume that $\sum_{j \in \mathcal{J}} a^*_{j,t} = 1$ if and only if node $t \in \mathcal{N}$ does split data. We build $S_T = (a, b, c, d, l, L_t, N_t, N_{k,t}, z)$ a feasible solution to $(T_{\alpha,\beta,\delta})$ with the same objective value:

- $d_t = \prod_{t' \in A(t) \cup \{t\}} \left(\sum_{j' \in \mathcal{J}} a^*_{j,t'} \right)$, $a_{j,t} = d_t a^*_{j,t}$ and $b_t = d_t b^*_t \quad \forall t \in \mathcal{N}, j \in \mathcal{J}$, where $A(t)$ is the set of nodes on the path from r to t (r included);

- $z_{i,\ell} = \prod_{t \in A_L(\ell)} \mathbf{1}[\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} < b_t] \times \prod_{t \in A_R(\ell)} \mathbf{1}[\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t] \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
- $l_\ell = \max_{i \in \mathcal{I}} z_{i,\ell}$, $N_\ell = \sum_{i \in \mathcal{I}} z_{i,\ell}$, and $N_{k,\ell} = \sum_{i \in \mathcal{I}_k} z_{i,\ell} \quad \forall \ell \in \mathcal{L}, k \in \mathcal{K}$;
- $c_{k,\ell} = d_{a(\ell)} g^*_{k,\ell} + \sum_{t \in \mathcal{N}:\ leaf(t) = \ell} B_t g^*_{k,t}, \forall \ell \in \mathcal{L}, k \in \mathcal{K}$ where $B_t = d_{a(t)}(1 - d_t)$, if $t \neq r$ and $B_r = (1 - d_r)$;
- $L_\ell = \sum_{k \in \mathcal{K}} (N_\ell - N_{k,\ell}) c_{k,\ell} \quad \forall \ell \in \mathcal{L}$.

Let us first note that the integrity of S_F ensures the integrity of S_T. Constraints (1), (2), (3), (5), (6), (10), (11) and (14) are satisfied by definition of S_T.

Constraints (4)**.** First note that Constraints (3) ensure that variables d_t are decreasing from the root to the leaf of every branch and that by definition of $c_{k,\ell}$ and Constraints (29), we have $\sum_{k \in \mathcal{K}} c_{k,\ell} = d_{a(\ell)} + \sum_{t \in \mathcal{N}: leaf(t) = \ell} B_t \sum_{k \in \mathcal{K}} g_{k,t}$. We consider three cases according to the possible values of B and d in the sub-branch $SB(\ell) = \{t \in \mathcal{N} \ : \ leaf(t) = \ell\}$ and prove that Constraints (4) are always satisfied:

- *Case 1:* $B_t = 0$ *for all* $t \in SB(\ell)$ *and* $d_{a(\ell)} = 1$.
  In that case, $d_t = 1$ for all $t \in SB(\ell)$. Therefore, $\sum_{k \in \mathcal{K}} c_{k,\ell} = 1$. Moreover, $d_{a(\ell)} = 1$ also ensures that $\sum_{j \in \mathcal{J}} a^*_{j,a(\ell)} = 1$ leading with our assumption to the existence of $i \in \mathcal{I}$ such that $z_{i,\ell} = 1$. Consequently, l_ℓ is also equal to 1.
- *Case 2:* $B_t = 0$ *for all* $t \in SB(\ell)$ *and* $d_{a(\ell)} = 0$.
  In that case $d_t = 0$ for all $t \in SB(\ell)$ and $r \notin SB(\ell)$. By definition of z and since for all $t \in SB(\ell)$ $d_{a(t)}$ is equal to 0, we obtain $\sum_{i \in \mathcal{I}} z_{i,\ell} = 0$. Therefore $\sum_{k \in \mathcal{K}} c_{k,\ell} = l_\ell = 0$.
- *Case 3:* $\exists \bar{t} \in SB(\ell)$ *such that* $B_{\bar{t}} = 1$.
  Here $\bar{t}$ is necessarily the first node in sub-branch $SB(\ell)$ such that $d_t = 0$. By definition of a, we have $\sum_{j \in \mathcal{J}} a_{j,\bar{t}} = 0$. Therefore, $\sum_{k \in \mathcal{K}} c_{k,\ell} = \sum_{k \in \mathcal{K}} g_{k,\bar{t}}$ which is equal to 1 from Constraints (49). From the definitions of a, b, z, our assumption and Constraints (51) and (52), we have $l_\ell = 1$.

Constraints (12) **and** (13)**.** Because the solution of $(F_{\alpha,\delta})$ is optimal, a class assigned to a node $t \in \mathcal{N} \cup \mathcal{L}$ is necessarily one of the most represented among the data reaching t. Therefore $c_{k,\ell}$ is equal to $argmax_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} z_{i,\ell}$. By definition of L_ℓ, N_ℓ and $N_{k,\ell}$, Constraints (12) and (13) are satisfied.

Constraints (8) **and** (9)**.** The provided solution may not satisfy all constraints (8) and (9). Indeed, unlike in $(T_{\alpha,\beta,\delta})$, Formulation $(F_{\alpha,\delta})$ does not follow the path of misclassified data. Consequently, there may exist a data $i \in \mathcal{I}$ missclassified by $(F_{\alpha,\delta})$ such that for a given leaf $\ell \in \mathcal{L}$ and a node $t \in A_L(\ell)$, $z_{i,\ell} = 1$ and $x_{i,j} \in]b_t - \mu_j, b_t]$. In that case the associated Constraint (8) is violated. However, it is easy to adjust the value of b_t so that Constraint (8) is satisfied. More precisely, let $j \in \mathcal{J}$ be the feature such that $a_{j,t} = 1$. By construction of μ_j, $x_{i,j}$ is the only value of the dataset for feature j in the interval $[x_{i,j}, x_{i,j} + \mu_j[$. Consequently, the constraint can be satisfied by setting the value of b_t to $x_{i,j} + \mu_j$.

By use of Equation (39) it follows that the two solutions have the same values.

□

We now prove that the continuous relaxation $(\overline{T}_{\alpha,\beta,\delta})$ of $(T_{\alpha,\beta,\delta})$ is worse than that of $(F_{\alpha,\delta})$.

Proposition 5. *If* $\alpha > 0$, $\beta > 0$, *and* $|\mathcal{K}| > 1$, $v(\overline{F}_{\alpha,\delta}) > v(\overline{T}_{\alpha,\beta,\delta})$.

Proof. By Lemma 1, it amounts to prove that $v(\overline{F}_{\alpha,\delta}) > 0$. Let us distinguish two cases:

Case 1: If $\sum_{(j,t) \in \mathcal{J} \times \mathcal{N}} a_{j,t} > 0$, since $\alpha > 0$, the value of the relaxation is necessarily positive.

Case 2: If $\sum_{(j,t)\in\mathcal{J}\times\mathcal{N}} a_{j,t} = 0$, Constraints (53) and (54) impose that $u^i_{t,t'} = 0$ for any arc (t,t') such that $t \neq s$ and $t' \neq w$. From the flow conservation constraints (30), we deduce that for all $i \in \mathcal{I}$, $u^i_{s,r} = u^i_{r,w}$. Considering Constraints (29) and (34), we have that, for any $(i_1, i_2) \in \mathcal{I}^2$ such that $y_{i_1} \neq y_{i_2}$:

$$u^{i_1}_{s,r} + u^{i_2}_{s,r} = u^{i_1}_{r,w} + u^{i_2}_{r,w} \leq g_{y_{i_1},r} + g_{y_{i_2},r} \leq 1$$

Therefore, $|\mathcal{I}| - \sum_{i\in\mathcal{I}} u^i_{s,r} > 0$.

□

4.2 Extension to the oblique case

We now extend Formulation $(F_{\alpha,\delta})$ to the case of oblique splits. We once more rely on Formulation $(T^H_{\alpha,\beta,\delta,\mu})$. In particular, we use variables $\hat{a}_{j,t} = |a_{j,t}|$, binary variables $s_{j,t}$ to count the number of $a_{j,t} \neq 0$, and binary variables d_t that indicate if node t is active, together with constraints (17)-(22) of $(T^H_{\alpha,\beta,\delta,\mu})$. We also use Constraints (29)-(31), (34) of $(F_{\alpha,\delta})$ leading to the following formulation $(F^H_{\alpha,\delta,\mu})$:

$$(F^H_{\alpha,\delta,\mu}) \begin{cases} \min \quad f^{FH}_{\hat{L},\alpha}(u,s) = \dfrac{1}{\hat{L}}\left(|\mathcal{I}| - \sum_{i\in\mathcal{I}} u^i_{s,r}\right) + \alpha \sum_{t\in\mathcal{N}}\sum_{j\in\mathcal{J}} s_{j,t} & \\ \text{s.t.} (17)-(22), (25), (29)-(31), (34), (36), (37) & \\ \quad d_t + \sum_{k\in\mathcal{K}} g_{k,t} = 1 & t \in \mathcal{N} \quad (56) \\ \quad \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} + \mu \leq b_t + (2+\mu)(1 - u^i_{t,l(t)}) & t \in \mathcal{N}, i \in \mathcal{I} \quad (57) \\ \quad \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \geq b_t - 2(1 - u^i_{t,r(t)}) & t \in \mathcal{N}, i \in \mathcal{I} \quad (58) \\ \quad u^i_{t,l(t)} \leq d_t & i \in \mathcal{I}, t \in \mathcal{N} \quad (59) \\ \quad u^i_{t,r(t)} \leq d_t & i \in \mathcal{I}, t \in \mathcal{N} \quad (60) \end{cases}$$

In this formulation $\sum_{j\in\mathcal{J}} a_{j,t}$ has no longer a binary value, we thus adapt Constraints (49), (53), (54) into Constraints (56), (59), (60) by use of binary variables d_t. Finally, to model the split functions, we rely on Formulation $(T^H_{\alpha,\beta,\delta,\mu})$ by adapting Constraints (23)-(24). Variables $z_{i,\ell}$ are replaced by variables $u^i_{t,l(t)}$ in (57) and by variables $u^i_{t,r(t)}$ in (58). For the objective function, we have chosen to use the same parameters as in $(T^H_{\alpha,\beta,\delta,\mu})$, so we minimize a function weighted by parameters $\hat{L}$ and α.

The following proposition proves that, in the oblique case, the optimal value of formulation $(F^H_{\alpha,\delta,\mu})$ can be strictly better than the optimal value of formulation $(T^H_{\alpha,\beta,\delta,\mu})$.

Proposition 6. *For $\delta > 0$, $\beta = 0$ and $\mu > 0$, $v(F^H_{\alpha,\delta,\mu}) \leq v(T^H_{\alpha,\beta,\delta,\mu})$. There exists datasets for which this inequality is strict if and only if $\alpha < 1$.*

Proof. Let $(a^*, \hat{a}^*, s^*, b^*, c^*, d^*, l^*, L^*, N^*_t, N^*_{k,t}, z^*)$ be an optimal solution of $(T^H_{\alpha,\beta,\delta,\mu})$. Without loss of generality, let us assume that $d_t = 1$ if and only if node $t \in \mathcal{N}$ does split data. We build $(a, \hat{a}, s, b, d, u, g)$ a feasible solution of $(F^H_{\alpha,\delta,\mu})$:

- $a_{j,t} = a^*_{j,t}$, $\hat{a}_{j,t} = \hat{a}^*_{j,t}$, $s_{j,t} = \tilde{s}_{j,t} \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $b_t = b^*_t$, $d_t = d^*_t \quad \forall t \in \mathcal{N}$;
- $u^i_{t,t'}$ is defined differently depending on $(t,t') \in \mathcal{E}$:
  - $u^i_{s,r} = \sum_{\ell\in\mathcal{L}} z^*_{i,\ell} c^*_{y_i,\ell} \quad \forall i \in \mathcal{I}$;

- $u^i_{\ell,w} = u^i_{a(\ell),\ell} = d^*_{a(\ell)} z^*_{i,\ell} u^i_{s,r} \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
- $u^i_{a(t),t} = d^*_{a(t)} u^i_{s,r} \sum_{\ell \in Leaves(t)} z^*_{i,\ell} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
- $u^i_{t,w} = d^*_{a(t)}(1 - d^*_t) z^*_{i,leaf(t)} u^i_{s,r} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
- $u^i_{r,w} = (1 - d^*_1) z^*_{i,leaf(r)} u^i_{s,r} \quad \forall i \in \mathcal{I}$;

- $g_{k,t}$ is defined differently depending on $t \in \mathcal{N} \cup \mathcal{L}$:
  - $g_{k,t} = (1 - d^*_t) d^*_{a(t)} c^*_{k,leaf(t)} + (1 - d^*_{a(t)}) \mathbf{1}[k = 1] \quad \forall t \in \mathcal{N} \setminus \{r\}, k \in \mathcal{K}$;
  - $g_{k,r} = (1 - d^*_1) c^*_{k,leaf(r)} \quad \forall k \in \mathcal{K}$;
  - $g_{k,\ell} = d^*_{a(\ell)} c^*_{k,\ell} + (1 - d^*_{a(\ell)}) \mathbf{1}[k = 1] \quad \forall \ell \in \mathcal{L}$.

With an very similar reasoning as used in the proof of Proposition 4, one can check that this solution satisfies all constraints of $(F^H_{\alpha,\delta,\mu})$ and that the two solutions have the same value. Consequently, $v(F^H_{\alpha,\delta,\mu}) \leq v(T^H_{\alpha,\beta,\delta,\mu})$.

If $\alpha < 1$, we now exhibit datasets for which $v(F^H_{\alpha,\delta,\mu}) < v(T^H_{\alpha,\beta,\delta,\mu})$.

We consider a training set $\mathcal{I} = \{(X_{i_1}, k_1), (X_{i_2}, k_2), (X_{i_3}, k_3), (X_{i_4}, k_3)\}$ composed of three classes $\mathcal{K} = \{k_1, k_2, k_3\}$ such that:

- $x_{i_1,1} = 0$, $x_{i_2,1} = \frac{\overline{\mu}}{2}$, $x_{i_3,1} = \overline{\mu}$, $x_{i_4,1} = 1$;
- $x_{i_1,j} = x_{i_2,j} = x_{i_3,j} = x_{i_4,j} \quad \forall j \in \mathcal{J} \setminus \{1\}$.

We prove that data i_1 and i_3 necessarily reach the same leaf of a tree obtained by $(T^H_{\alpha,\beta,\delta,\mu})$ while they can be separated by $(F^H_{\alpha,\delta,\mu})$.

Let t be the highest node of a decision tree which split function separates i_1 and i_3. Let us assume without loss of generality that $\sum_{j \in \mathcal{J}} a_{j,t} x_{i_1,j}$ is lower than $\sum_{j \in \mathcal{J}} a_{j,t} x_{i_3,j}$. Consequently,

$$\begin{array}{c} \sum_{j \in \mathcal{J}} a_{j,t} x_{i_1,j} < \quad b_t \quad \leq \sum_{j \in \mathcal{J}} a_{j,t} x_{i_3,j} \\ 0 < b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} \leq a_{1,t} \overline{\mu} \end{array} \tag{61}$$

In $(T^H_{\alpha,\beta,\delta,\mu})$, the Constraint (23) associated with node t and the leaf reached by data i_1 leads to

$$\overline{\mu} \leq b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} \tag{62}$$

From inequalities (61) and (62) we deduce that

$$b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} = \overline{\mu} \tag{63}$$

Given the features vectors of the data and the fact that t is the highest node which separates nodes i_1 and i_3, we deduce that data i_2 necessarily reaches node t. Consequently, i_2 either reaches the left or the right child of t. We prove that in both cases Equation (63) can not be satisfied, thus proving that i_1 and i_3 can not be separated in $(T^H_{\alpha,\beta,\delta,\mu})$.

If i_2 reaches $l(t)$, we deduce from the Constraint (23) associated with node t and the leaf reached by i_2 that:

$$\begin{aligned} \textstyle\sum_{j \in \mathcal{J}} a_{j,t} + \overline{\mu} &< b_t \\ a_{1,t} \tfrac{\overline{\mu}}{2} + \overline{\mu} &< \overline{\mu} \\ a_{1,t} \overline{\mu} &< 0 \end{aligned}$$

which is not possible as $a_{1,t}$ must be greater than 0 according to (61).

If i_2 reaches $r(t)$:

$$\begin{aligned}\textstyle\sum_{j\in\mathcal{J}} a_{j,t}x_{i_2,j} &\geq b_t\\ a_{1,t}\frac{\overline{\mu}}{2} &\geq \overline{\mu}\\ a_{1,t} &\geq 2\end{aligned}$$

which is not either possible as $a_{1,t} \leq 1$.

Since the distances between i_1 and i_2 or i_2 and i_3 are both lower than the one between i_1 and i_3, $(T^H_{\alpha,\beta,\delta,\mu})$ can not either separate these data. Thus, all three data necessarily reach the same leaf of a tree obtained by $(T^H_{\alpha,\beta,\delta,\mu})$ which leads to at least 2 misclassifications.

However, i_1 and i_3 can be separated by a tree obtained with $(F^H_{\alpha,\delta,\mu})$. Indeed, if i_2 is missclassified, its flow is equal to 0 and all Constraints (51) and (52) involving this data are satisfied. A tree with a single split function $x_{1,t} \leq \overline{\mu}$ can, thus, be obtained by $(F^H_{\alpha,\delta,\mu})$ and leads to only one misclassification. Consequently, if $\alpha = 0$, $v(F^H_{\alpha,\delta,\mu}) < v(T^H_{\alpha,\beta,\delta,\mu})$.

If α is in $]0,1[$. Let T_r be the tree reduced to a root node which predicts the most represented class $\overline{k}$ in the training set (i.e., $\overline{k} = \arg\max_{k\in\mathcal{K}} |\mathcal{I}_k|$). The same reasoning applies for this value of α provided that we can ensure that T_r is not an optimal tree (otherwise $v(F^H_{\alpha,\delta,\mu}) = v(T^H_{\alpha,\beta,\delta,\mu}) = 1$). This can be obtained by adding to the dataset a sufficient number of data equal to (X_{i_1}, k_1) and (X_{i_4}, k_3).

If $\alpha \geq 1$, T_r is an optimal tree. Consequently, $v(F^H_{\alpha,\delta,\mu}) = v(T^H_{\alpha,\beta,\delta,\mu})$. □

In terms of value of the linear relaxation, we obtain a result similar to the one of the axis-aligned case.

Proposition 7. *If $\alpha > 0$, $\beta = 0$ and $|\mathcal{K}| > 1$, $v(\overline{F}^H_{\alpha,\delta,\mu}) > v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$.*

Proof. By Lemma 2, it amounts to prove that $(\overline{F}^H_{\alpha,\delta,\mu}) > 0$. Let us distinguish two cases.

Case 1: If $\displaystyle\sum_{(t,j)\in\mathcal{N}\times\mathcal{J}} s_{j,t} > 0$, since α is positive, the value of the relaxation is necessarily positive.

Case 2: If $\displaystyle\sum_{(t,j)\in\mathcal{N}\times\mathcal{J}} s_{j,t} = 0$, the proof is the same in Case 2 in the proof of Proposition 5.

□

5 From MIP solution to classification tree

The optimisation models introduced in this previous sections have a large set of optimal solutions. In this section, we present a post-processing method that selects a "good" optimal tree within the set of optimal solutions (Section 5.1). Moreover, the new formulations depend on 3 parameters: the depth of the tree δ, the minimal number of data in a leaf β, and the weight α of the second objective. In Section 5.2, we introduce a new parameters fitting algorithm `CompactTreeTraining` that is more efficient than algorithm `TreeTraining` presented in Section 2.1.3.

5.1 Post-processing the MILP solutions for better performances

Since the variables that define the split functions are continuous, the considered models have a large number of equivalent optimal solutions. We observed that the solvers often provide split functions that are very close to the data. To reduce the risks of having test data of a same class on both sides of a split function, our post-processing follows the idea of the well-known Support Vector Machine (SVM) approaches (see [Vapnik, 1963]) and identify split functions that are as far as possible from the training data. Note that several approaches based on SVM for finding Optimal Classification Trees were proposed. For instance, maximum margin classifiers of the SVM type have been proposed in [Blanco et al., 2021] where a new formulation is introduced in which, except for the leaf nodes, the labels

are temporarily left out and grouped into two classes by means of a SVM separating hyperplane. In [Blanco et al., 2022b] a Mixed-Integer Non Linear formulation is introduced. The aim is to build a robust tree classifier, where the splitting rules are based on the possibility of relabeling some samples as described in [Blanco et al., 2022a]. Finally, in [D'Onofrio et al., 2022] a novel mixed-integer quadratic formulation to train multivariate optimal classification trees which employ maximum margin hyperplanes by following the soft SVM paradigm is introduced. In our case, we are only interested to focus on the post-processing phase for our formulations, in contrast to the approaches mentioned above that include the full process of computing an OCT. This is why, for our post-processing step we solve an optimization problem similar to that of [Zhou et al., 2002] that was introduced for linear SVM problems.

Our post-processing phase is applied independently on each splitting node $t \in \mathcal{N}$. Let $\mathcal{I}_L$ ($\mathcal{I}_R$ resp.) be the set of data reaching the left (right resp.) child of node t. We determine the split function between $\mathcal{I}_L$ and $\mathcal{I}_R$ which is the furthest from all data in $\mathcal{I}_L \cup \mathcal{I}_R$.

In the oblique case, the new split function of node t is obtained by solving the mixted-integer linear problem (S). For all $j \in \mathcal{J}$, let $s^*_{j,t}$ be the value of variable $s_{j,t}$ before the post-processing. Variables $a_{j,t}$, b_t and $s_{j,t}$ characterise the new split function. Variable e_i is equal to the distance between data $i \in \mathcal{I}$ and the new split function and e_{min} is equal to $\min_{i\in\mathcal{I}} e_i$.

$$
(S)\begin{cases}
\max\limits_{a_{j,t},b_t,s_{j,t},e_i,e_{min}} \quad e_{min} & \\
\text{s.t.} \quad e_i \geq e_{min} & i \in \mathcal{I}_L \cup \mathcal{I}_R \quad (64)\\
e_i = b_t - \sum\limits_{j\in\mathcal{J}} a_{j,t}x_{i,j} & i \in \mathcal{I}_L \quad (65)\\
e_i = \sum\limits_{j\in\mathcal{J}} a_{j,t}x_{i,j} - b & i \in \mathcal{I}_R \quad (66)\\
-s_{j,t} \leq a_{j,t} \leq s_{j,t} & j \in \mathcal{J} \quad (67)\\
\sum\limits_{j\in\mathcal{J}} s_{j,t} \leq \sum\limits_{j\in\mathcal{J}} s^*_{j,t} & \quad (68)\\
b \in [-1, 1] & \quad (69)\\
s_{j,t} \in \{0, 1\} & j \in \mathcal{J} \quad (70)
\end{cases}
$$

Constraints (64) ensure the equality $e_{min} = \min_i e_i$. Constraints (65) ((66) resp.) set the value of e_i for data points going through the left branch (right branch resp.). Constraints (67) define variables $s_{j,t}$ as indicators of whether $a_{j,t}$ is null or not. Constraint (68) ensures that the split keeps the same number of non-zero coefficients. Constraints (67) and (69) define respectively the domain of b_t and $s_{j,t}$. Problem (S) has $|\mathcal{J}|$ integer variables and is thus fast to solve with a strandard MILP solver.

In the case of axis-aligned splits, only the right-hand sides $\{b_t\}_{t\in\mathcal{N}}$ of the split functions are modified which enables to avoid solving MILPs. Let $j^* \in J$ be the feature on which the separation is performed in node $t \in \mathcal{N}$, $b^m_t = \max_{i\in\mathcal{I}_L} x_{i,j^*}$, and $b^M_t = \min_{i\in\mathcal{I}_R} x_{i,j^*}$. Since any $b_t \in [b^m_t, b^M_t[$ is optimal, we maximize the minimal distance between the new split function and the data by fixing $b_t = \frac{b^m_t + b^M_t}{2}$.

5.2 A Compact tree training algorithm

Our formulations depend on 3 parameters: the depth of the tree δ, the minimal number of data in a leaf β, and the weight α of the second objective that penalizes the complexity of the tree. Following the ideas of [Bertsimas and Dunn, 2017], we propose a new algorithm that fits the parameters and allows to train the decision trees. Algorithm `TreeTraining` (described in Section 2.3) computes relevant values of the parameters, at the cost of a heavy computational time, since each iteration requires the resolution of a MILP. This is especially true for oblique splits as the highest possible number of non-zero coefficients in the split functions $\overline{C}_\delta$ is proportional to $|\mathcal{J}|$. We thus introduce a new algorithm which enables to reduce the number of iterations.

Observe that the number of misclassifications is a decreasing piecewise constant function of C. Hence, the loop on parameter C can skip some values while leading to the same solutions than `TreeTraining`. Unlike `TreeTraining`, in our new algorithm we keep the second objective and set α to a value small enough to prioritise the first objective. Consequently, among all the solutions with an optimal number of misclassifications, we obtain one for which the number of splits is minimal. Thus, at iteration C, if the number of splits $\hat{C}$ of the solution is lower than C,

iterations $\{\hat{C}, ..., C-2, C-1\}$ are skipped. This leads to our new Algorithm `CompactTreeTraining` represented in Algorithm 2. It has 4 inputs: the considered model $M \in \{(Q), (Q^H), (QF), (QF^H), (QG), (QG^H), (F), (F^H)\}$, $\overline{\delta}$ the maximal depth of the tree, a fixed β (the minimal number of data points per open leaf), and α the weight of the second term of the objective function. Note that as in `TreeTraining`, $\overline{C}_\delta$ is equal to $2^\delta - 1$ in the axis-aligned case, and to $(2^\delta - 1)|\mathcal{J}|$ in the oblique case.

Algorithm *CompactTreeTraining*(M,$\overline{\delta}$,β,$\overline{\alpha}$)

- $\mathcal{S} \leftarrow \emptyset$
- **for** $\delta = 1$ *to* $\overline{\delta}$ **do**
  - $C \leftarrow \overline{C}_\delta$
  - **while** $C \geq \delta$ **do**
    - $\hat{T} \leftarrow$ a feasible solution for warm starting (given by CART, from set $\mathcal{S}$, or built from T obtained at the previous step).
    - $T \leftarrow$ an optimal solution of M with $\alpha = \overline{\alpha}$, Constraint (26) or (27), and $\hat{T}$ as a warm start.
    - $T \leftarrow$ PostProcess(T)
    - $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
    - $C \leftarrow$ (Number of non-null coefficients in the splits of T) $- 1$.
- Remove within $\mathcal{S}$ all dominated solutions for M.
- $T^* \leftarrow$ the tree in $\mathcal{S}$ that best performs on the validation set.
- **return** T^*

Algorithm 2: Algorithm `CompactTreeTraining` to train a tree with model $M \in \{(Q), (Q^H), (QF), (QF^H), (QG), (QG^H), (F), (F^H)\}$

6 Computational results

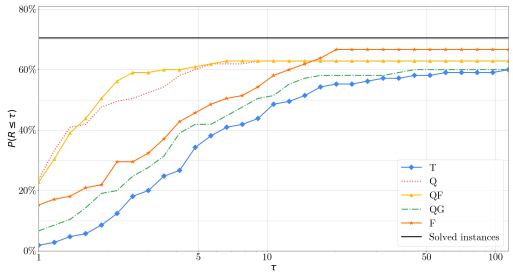
We now present computational experiments on several datasets from the literature. We start in Section 6.1 by presenting the reduction of the CPU times between the formulations introduced in this paper and the formulations of [Bertsimas and Dunn, 2017]. Then, in Section 6.2, we evaluate the efficiency of our new learning algorithm `CompactTreeTraining` and we show that it significantly reduces the learning time compared to that of `TreeTraining`. Finally, we present in Section 6.3 how, combined with our post-processing, it provides similar or better accuracy results than 7 state-of-the-art methods.

Experimental setup The experiments were performed on a server with 2 Intel Xeon CPUs each with 16 cores and 32 threads of 2.3 GHz and $8 * 16$ GB of RAM running the Linux OS. The experiments were implemented in C++ and we used the solver `Gurobi 9.1.1` [Gurobi Optimization, LLC, 2023] for solving the optimisation problems.

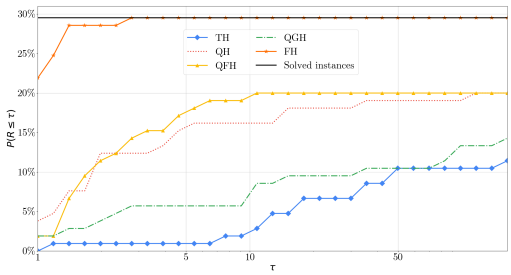
6.1 Efficiency of our new formulations

In this section, we compare the CPU times of the direct submission to the solver `Gurobi` of the formulations (T), (Q), (QF), (QG), and (F) (axis-aligned case), and (T^H), (Q^H), (QF^H), (QG^H), and (F^H) (oblique case). We run our experiments on 7 different training sets for each considered dataset: Blood donation, Breast cancer, Car evaluation, Dermatology, Iris, Tic tac toe and Wine whose characteristics are described in Table 1. We set parameters to the following values: $\alpha = 0.1$, depths $\delta \in \{2, 3, 4\}$, $\beta = 0$ (in order to make a comparison with flow-based formulations), and for the oblique-case $\mu = 10^{-4}$.

We present in Figure 6 the performance profiles [Dolan and Moré, 2002] of the CPU time of the considered formulations. In a performance profile of CPU times, each curve corresponds to a formulation, where each point of a curve gives, for a given factor τ, the percentage of instances whose CPU time was at most τ times greater than the minimal CPU time within the considered formulations. In particular, for $\tau = 1$, we have the proportion of instances on which the formulation was the best on the criterion.



(a) Performance Profile in axis-aligned case.



(b) Performance Profile in oblique case.

Figure 6: Performance Profiles of CPU Times over the 105 considered instances - Time limit : 1 h

In both the axis-aligned and the oblique cases, we observe that formulation (T) is the slowest followed by (QG). In the oblique case (F^H) clearly outperforms (QF^H) and (Q^H) which have similar CPU times. However, in the axis-aligned case, (QF) and (Q) are generally faster than (F). In both cases, (F) solves the most instances to optimality.

In conclusion, the linearisation (QG) is not a clear improvement over (T). Formulations (Q), (QF) and (F) significantly outperform (T) in terms of CPU times. Given, the similar performances of (Q) and (QF) we only focus in the following on formulations (QF) and (F).

6.2 Detailed comparison of algorithms `TreeTraining` and `CompactTreeTraining`

We run our experiments on datasets from the UCI Repository [Dua and Graff, 2017] whose characteristics are summed up in Table 1. For each dataset, we consider the 5 partitions from [Verwer and Zhang, 2019] taken from Github. A partition is described by a training set, a validation set and a test set representing 50%, 25% and 25% of the original datasets, respectively. As in the experiments of [Bertsimas and Dunn, 2017], we set the time limit to 1800 seconds for the axis-aligned case, and to 300 seconds for the oblique case. The maximum depth is 4 and parameter β is fixed at $5\%|\mathcal{I}|$.

We start with a comparison of the CPU times and of the number of iterations of algorithm `TreeTraining` run with $M \in \{(T), (T^H)\}$, and of our new algorithm `CompactTreeTraining` run with $M \in \{(QF), (QF^H), (F), (F^H)\}$. In Figure 7 (axis aligned case) and in Figure 8 (oblique case), we plot the ratios of CPU time (in blue) and of the number of iterations (in red) for (QF) and (F) compared to (T) for $\delta \in \{2, 3, 4\}$. More precisely, each point is $\frac{CPU(M)}{CPU(T)}$, where $M \in \{(QF), (QF^H), (F), (F^H)\}$ and $T = (T)$ for the axis-aligned case, and $T = (T^H)$ for the oblique case. Hence, if a point has a value smaller than 1 it means that M outperforms T. In the axis-aligned case (Figure 7), we observe that `CompactTreeTraining` with either (QF) or (F) always reduces the number of iterations in comparison to `TreeTraining`. Moreover, it is often faster, and the reduction of CPU times grows when the depth of the tree increases. This speed-up is also due to the fact that, at each iteration (QF) and (F) are faster to solve than (T). For the oblique case (Figure 8), the improvement in terms of CPU times is even greater. Concerning the ratio of the number of iterations, it is either smaller than or equal to the ratio of the CPU times, meaning that the reduction in CPU times is mostly due to the reduction of the number MILP solved.

Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{K}\rvert$
Balance scale	625	4	3
Bank marketing 10%	4521	51	2
Car evaluation	1728	6	4
Ionosphere	351	34	2
Iris	150	4	3
Monk1	124	6	2
Monk2	169	6	2
Monk3	122	6	2
Pima Indians diabetes	1151	19	2
Qsar biodegradation	1055	41	2
Seismic bumps	2584	18	2
Spambase	4601	57	2
Statlog satellite	4435	36	7
Tic tac toe	958	18	2
Wine	178	13	3
Blood transfusion	748	4	2
Breast cancer	683	9	2
Dermatology	358	34	6
Ecoli	336	7	8
German	1000	24	2
Haberman	306	3	2
Seeds	210	6	3

Table 1: Datasets used for tests

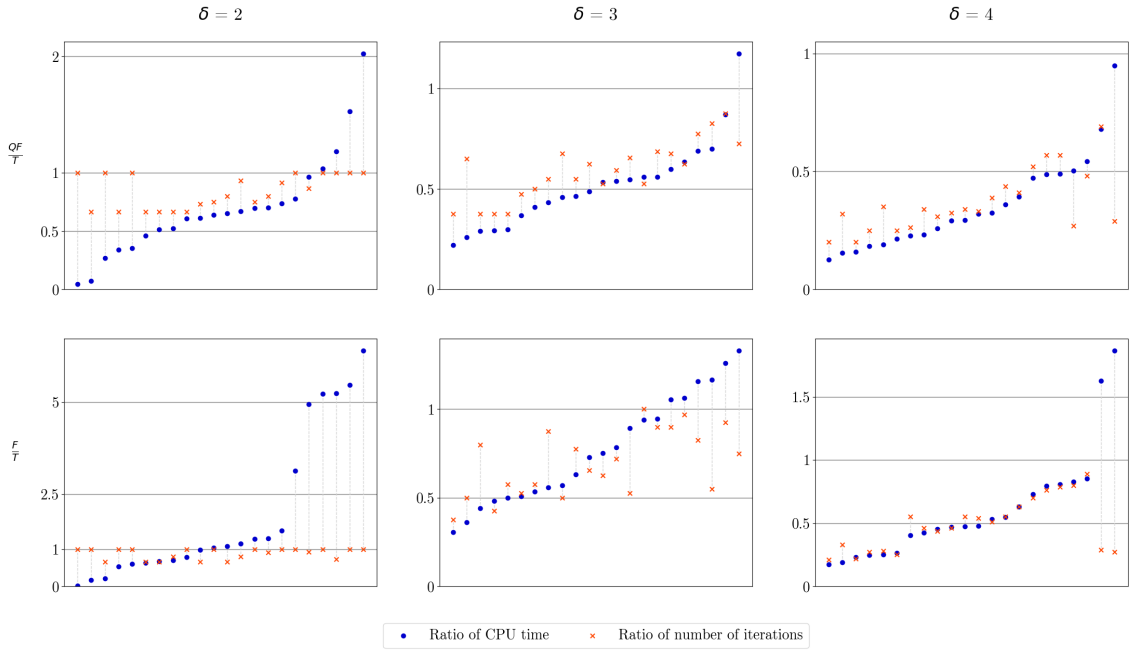


Figure 7: Ratio of CPU times and ratio of number of iterations for `CompactTreeTraining` with (QF) and (F) compared to `TreeTraining` with (T) (in axis-aligned case)

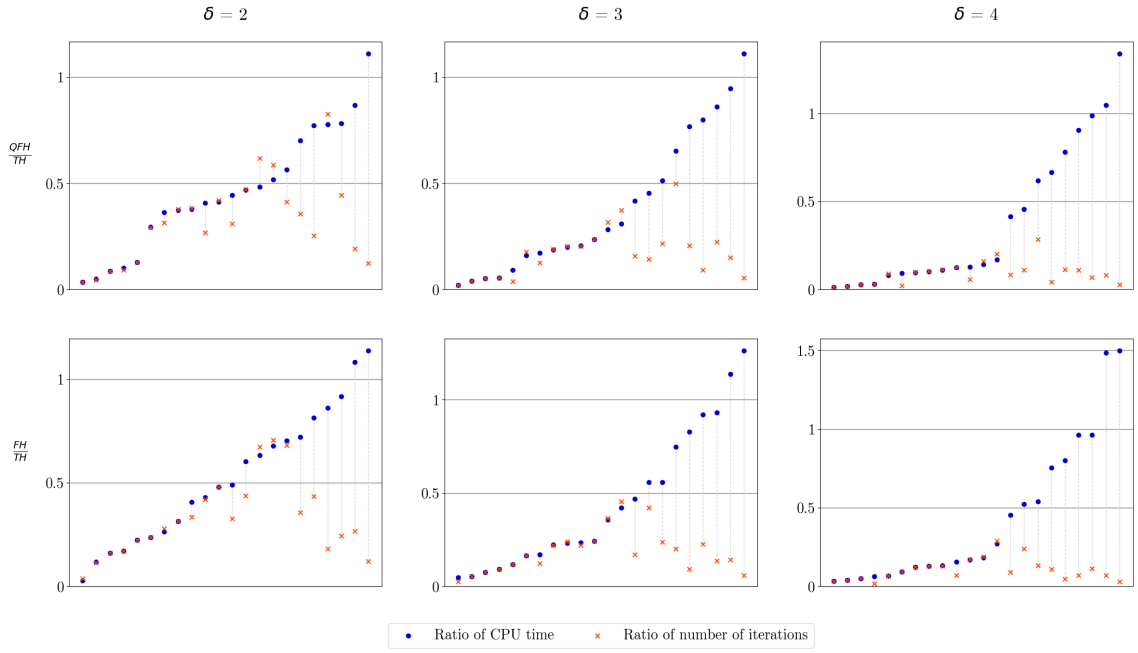


Figure 8: Ratio of CPU times and ratio of number of iterations for `CompactTreeTraining` with (QF^H) and (F^H) compared to `TreeTraining` with (T^H) (in oblique case)

In Section 5.1, we introduce a post-processing that finds split functions of a tree as far as possible from the data which does not alter the path of data from the training set. In our algorithm `CompactTreeTraining` this post-processing is applied at each iteration. However, we observe empirically that it is more efficient to only consider the post-processed tree whenever it does not deteriorate the performances on the validation set. We use this variant in the following.

6.3 Comparison with exact state-of-the-art methods for learning performances

In this section, we compare the accuracy of our new algorithm `CompactTreeTraining (CTT)` with the following set of models: $M \in \{(QF), (QF^H), (F), (F^H)\}$ to that of the following state-of-the-art methods: `BinOCT*` [Verwer and Zhang, 2019], `CGH` [Firat et al., 2020], `TreeTraining (TT)` [Bertsimas and Dunn, 2017] with $M \in \{(T), (T^H)\}$ and `CART` [Breiman et al., 1984]. We run our experiments on the datasets presented in Table 1.

We present in Tables 2-4 the testing accuracy of the methods on each dataset for depth $\delta \in \{2, 3, 4\}$, respectively. In order to provide a relevant comparison, for each dataset the accuracies reported correspond to averages over the same 5 dataset partitions taken from [Verwer and Zhang, 2019]. Note that the results of `BinOCT*` and `CGH` are directly taken from [Verwer and Zhang, 2019] and [Firat et al., 2020] that were run with the same partitions. Algorithms `CTT` and `TT` are implemented with the C++ interface of `Gurobi 911` [Gurobi Optimization, LLC, 2023].

Column *Dataset* of Tables 2-4 indicates the name of the considered dataset. The next 6 columns report the average percentage of correctly classified data for axis-aligned models, and the last 3 columns that of the oblique models. For each dataset, the highest accuracies among the 8 considered algorithms are in bold. In order to take into account the scope of each method, we also indicate with a "$*$" the best accuracies within each axis-aligned and oblique methods. The symbol "-" means that we do not have the result for the considered method, this is the case for the last 7 datasets of Table 1 that were not tested for methods `BinOCT*` [Verwer and Zhang, 2019] and `CGH` [Firat et al., 2020]. To analyze these results, we compute for each dataset the *Relative Gap (RG)* between the performance of each algorithm and that of the best one, i.e. $RG = \left|\frac{B-A}{B}\right| * 100$, where B is the highest accuracy (in bold), and A the accuracy of the considered method. We report in lines *Average RG*, the average relative gap in % over the first 15 datasets, then over the last 7, and we finally give in line *Total average RG* the average over all 22 datsets.

Dataset	Axis-aligned					Oblique		
	BinOCT*	CGH	TT(T)	CTT(QF)	CTT(F)	TT(T^H)	CTT(QF^H)	CTT(F^H)
Balance scale	69.3*	69.2	68.8	68.8	67.8	89.6	89.6	**90.6**
Bank marketing 10%	**90.3**	-	88.2	88	88.5	89.2*	89.1	88.9
Car evaluation	77.8*	77.8*	77.8*	77.8*	76.4	83.1	83.5	**84.4**
Ionosphere	87.7	86.4	**89.3**	88.9	88	87	88.2	**89.3**
Iris	95.8	95.8	92.6	95.8	96.8*	**97.9**	95.8	95.8
Monk1	80*	71	77.4	77.4	76.1	93.5	**95.5**	93.5
Monk2	54.4*	52.6	53	53	53	65.6	**71.6**	67.4
Monk3	**93.5**	**93.5**	**93.5**	**93.5**	**93.5**	89.7	89.7	90.3*
Pima Indians diabetes	75.3*	75.2	75.3*	75.3*	75.3*	73.6	**75.5**	74.6
Qsar biodegradation	78.1	-	77.6	77.4	79.3*	85.5	85.3	**85.7**
Seismic bumps	93.8	93.6	**94.1**	**94.1**	94	**94.1**	**94.1**	93.7
Spambase	85.7	86.5*	85.2	85.5	83.4	92.3	85.9	**92.6**
Statlog satellite	65.7	63.9	65.1	64.5	68.3*	65.5	63.7	**69.3**
Tic tac toe	67.3	67.7*	67.6	67.6	67.6	94.8	**96.2**	95.3
Wine	91.1	87.6	91.1	92*	92*	91.1	**94.2**	93.8
Average RG	*8.7*	*10.8*	*9.4*	*9.2*	*9.1*	*2.2*	*1.7*	*1.2*
Blood transfusion	-	-	76.1	76.2	77.7*	**80.1**	79.8	78.6
Breast cancer	-	-	93.3	93.5	95.1*	94.9	**95.2**	94.6
Dermatology	-	-	73.6	73.3	74*	**76.2**	73.6	74.4
Ecoli	-	-	77.7*	77.7*	77.1	**78.6**	**78.6**	77.1
German	-	-	72.2	72.4*	70.6	**75**	73	72.2
Haberman	-	-	72.7	72.7	73*	73	**74**	73
Seeds	-	-	89.8	89.1	91.3*	94.3	94.7	**95.1**
Average RG			*2.8*	*2.8*	*2.5*	*0.3*	*1.1*	*2*
Total average RG			*7.5*	*7.3*	*7.2*	*1.7*	*1.5*	*1.4*

Table 2: Percentage of correctly classified data on 22 datasets for 8 MILP based exact methods with depth $\delta = 2$.

For the depth $\delta = 2$ (Table 2), we observe that algorithms that compute oblique trees outperforms axis aligned algorithms on 20 datasets over 22. Moreover, the average relative gap is significantly smaller for the oblique case since it is reduced by a factor of about 5 on average in comparison to the axis-aligned case. When focusing on the 15 first datasets and the axis-aligned case, we see that all algorithms have at least once the best accuracy, but BinOCT* and CTT(F) stand out with the most number of datasets on which they have the best accuracy (5 each). Consequently, they have the lowest average relative gap (respectively 8.7% and 9.1%). Lastly, note that both CTT(QF) and CTT(F) have a smaller average relative gap than TT(T). For the oblique case and over the 22 datasets, CTT(QF^H) is the most often the best performing algorithm (9 times), followed by CTT(F^H) (8 times) and TT(T^H) (7 times). Note however that CTT(F^H) has the smallest average relative gap, with CTT(QF^H) coming up second.

For depth $\delta = 3$ (Table 3), we still observe that algorithms computing oblique trees have better accuracy (on 15 datasets) than axis-aligned algorithms, but the reduced average relative gaps of axis-aligned algorithms show that their performances are improved. In the axis-aligned case and for the 15 first datasets, the results reveals a similar trend than for $\delta = 2$. BinOCT* performs better on most datasets (6 datasets), but TT(T) has the smallest average relative gap. In the oblique case and over all datasets, CTT(F^H) outperforms algorithms CTT(QF^H) and TT(T^H), with 11 datasets on which it has the best accuracy and with the smallest total average relative gap (1.4%).

The results for $\delta = 4$ are given in Table 4. We observe a similar trend than for depth 2 and 3 for the axis-aligned case, but the average relative gap is again reduced for axis-aligned algorithms in comparison to depth 3. For oblique algorithms, CTT(F^H) clearly outperforms algorithms CTT(QF^H) and TT(T^H), with 11 datasets on which it has the best accuracy and a total average relative gap reduced by a factor of 2 and 2.4 in comparison to that of methods CTT(QF^H) and TT(T^H) respectively.

6.4 Comparison with heuristic state-of-the-art methods for learning performances

In this section, we compare our best performing algorithms of each axis-aligned and oblique case, that are CTT(QF) and CTT(F^H), with 4 state-of-the arts heuristic methods: CART [Breiman et al., 1984], C5.0 [Quinlan, 1993],

Dataset	Axis-aligned					Oblique		
	BinOCT*	CGH	TT(T)	CTT(QF)	CTT(F)	TT(T^H)	CTT(QF^H)	CTT(F^H)
Balance scale	71.3	72.4	72.4	70.7	73.2*	90.2	89.8	**90.6**
Bank marketing 10%	88.5*	-	88.2	88	88.4	**89.2**	89.1	**89.2**
Car evaluation	80.4*	79	80.1	79.3	77.7	83.1	83.5	**84.3**
Ionosphere	85.5	86.4	**90**	87.3	88.2	88.6	89.1	89.8*
Iris	**97.9**	95.4	96.8	97.4	96.8	**97.9**	95.8	95.8
Monk1	80	92.2*	88.4	88.4	90.3	98.1	**100**	99.4
Monk2	55.3	56.3	56.7	57.7*	57.2	67.4	70.7	**72.6**
Monk3	89.7	90.3	**93.5**	**93.5**	91.6	89.7	89.7	90.3*
Pima Indians diabetes	74.4	**75.6**	75.5	75.4	75.4	73.2	75.5*	74.5
Qsar biodegradation	81.3	-	82.1*	79.9	78.7	85.5	85.3	**86.2**
Seismic bumps	92.8	92.9	**94.1**	**94.1**	94	**94.1**	**94.1**	93.6
Spambase	88.9*	88.8	86.4	86.3	83.8	92.3	86.1	**92.7**
Statlog satellite	**79.2**	77.7	78.7	76.9	68.3	77.3*	76.8	77.3*
Tic tac toe	70.6	71.3	72.4	71.5	73.1*	94.8	**96.2**	95.9
Wine	**92**	88	91.5	90.2	90.7	89.8	90.2	91.1*
Average RG	*7.9*	*8*	*6.6*	*7.3*	*8.1*	*1.8*	*1.7*	*0.8*
Blood transfusion	-	-	80.2	79.3	**80.6**	79.9*	79.5	79
Breast cancer	-	-	94.6*	94.6*	94.2	94.9	**95.2**	94.6
Dermatology	-	-	86.7	87.1	88*	92	**93.3**	81.3
Ecoli	-	-	78	79.5	82.7*	**83.9**	81.5	83.3
German	-	-	73	73.3*	71	**74.4**	73.7	74.2
Haberman	-	-	72.7	**74**	**74**	72.2	73*	72.7
Seeds	-	-	90.2	89.8	91.3*	94.7	94.7	**95.1**
Average RG			*3.1*	*2.6*	*2.1*	*0.8*	*1.1*	*3*
Total average RG			*5.5*	*5.9*	*6.4*	*1.5*	*1.4*	*1.4*

Table 3: Percentage of correctly classified data on 22 datasets for 8 MILP based exact methods with depth $\delta = 3$.

Dataset	Axis-aligned					Oblique		
	BinOCT*	CGH	TT(T)	CTT(QF)	CTT(F)	TT(T^H)	CTT(QF^H)	CTT(F^H)
Balance scale	78.9	79.1*	77.2	75.4	77.5	90.1	89.8	**90.6**
Bank marketing 10%	88.5*	-	88.2	88	88.5*	89.2	89.1	**89.6**
Car evaluation	**86.5**	86	86.3	84.4	77.7	85.3	85.2	85.9*
Ionosphere	88.6	85.7	**89.8**	88.6	88.9	89.5	89.1	**89.8**
Iris	**98.4**	97.9	96.3	97.4	96.8	97.9*	95.8	95.8
Monk1	87.1	84.5	91	92.3	95.5*	98.1	**100**	99.4
Monk2	63.3*	62.3	52.1	55.8	61.9	65.1	**72.6**	72.1
Monk3	84.5	89	**94.8**	93.5	93.5	89.7	89.7	90.3*
Pima Indians diabetes	73	75.1	**76**	75.4	74.5	73.2	**76**	75.3
Qsar biodegradation	81*	-	80.4	79.9	80.2	**85.5**	85.3	85.3
Seismic bumps	92.6	92.5	**94.1**	**94.1**	94	**94.1**	**94.1**	93.8
Spambase	89.5	89.6*	86.7	86.3	83.8	92.3	86.1	**92.7**
Statlog satellite	79.9	80*	78.7	77.9	68.3	78.1	77.9	**80.1**
Tic tac toe	78.8	79.3*	79.2	74.8	75	94.8	**96.2**	95.9
Wine	89.8	89.8	**91.5**	90.2	90.7	89.8	90.2	91.1*
Average RG	*5.8*	*6.1*	*5.9*	*6.4*	*7*	*2.1*	*1.6*	*0.8*
Blood transfusion	-	-	79.7*	78.5	79.1	**79.9**	79.5	79
Breast cancer	-	-	94.6	94.6	**95.2**	94.9	**95.2**	95.1
Dermatology	-	-	88	88.9	92*	92.4	93.3	**93.8**
Ecoli	-	-	80.7*	80.4	80.1	83	82.4	**84.2**
German	-	-	73*	73*	72.9	**74.8**	73.4	74.2
Haberman	-	-	**74**	73.5	**74**	72.2	73	72.7*
Seeds	-	-	90.9	92.8*	91.3	94.7	94.7	**95.1**
Average RG	-	-	*2.3*	*2.5*	*1.7*	*0.9*	*1.1*	*0.6*
Total average RG			*4.8*	*5.3*	*5.4*	*1.7*	*1.4*	*0.7*

Table 4: Percentage of correctly classified data on 22 datasets for 8 MILP based exact methods with depth $\delta = 4$.

GUIDE [Loh, 2002, Loh, 2009] and TAO [Carreira-Perpinan and Tavallali, 2018]. More precisely, we run CART and C5.0 with R libraries on the same partitions used for CTT(QF) and CTT(F^H). For CART, we compute the maximum tree and prune it, and we fit the complexity parameter via the validation set. For C5.0, we took the computed tree as is. For GUIDE and TAO, results are taken from the survey [Zharmagambetov et al., 2019] and come from different partitions.

In Table 5, we present the testing accuracy or the 8 considered algorithms, where each line corresponds to the average over 5 dataset partitions on the dataset reported in Column *Dataset*. For these experiences, we consider the two versions of TAO and GUIDE algorithms for both axis aligned and oblique cases, denoted by TAO-A, GUIDE-A and TAO-O, GUIDE-O, respectively. We also report the average Relative Gaps (RG) that are defined as described in Section 6.3. The first reported *Average RG* corresponds to the average over the first 8 datasets, and the second over the last 14. Finally, the *Total average RG* is the average over all the 22 datasets. If the datasets were not tested in [Zharmagambetov et al., 2019], we put - in the corresponding column.

Clearly, oblique tree algorithms once again outperform axis-aligned ones. Indeed, except for TAO-A, the axis-aligned algorithms lead to average relative gaps about two times larger than that of oblique ones. Consequently, the heuristic TAO-A significantly outperforms the other axis-aligned algorithms in terms of accuracy on the first 8 datasets. For the oblique case, TAO-O is again the best performing approach, but our exact algorithm CTT(QF) has an average relative gap that is close.

Our goal in this paper is to compute trees that are both accurate and interpretable. Thus, we present in Tables 6 and 7 for each dataset and each algorithm the average depth and the average number of leaves of the computed trees in order to evaluate their complexity. We observe that both algorithms TAO-A and TAO-O have the worst complexity, both in terms of depth and of number of leaves, since they are increased by a factor of 2.9 and 6.6, respectively, in comparison to the trees given by our new algorithm CTT(QF) which are the simplest. Indeed, the trees computed by CTT(QF) have on average a depth 3 and less than 4 leaves, and lead to performances almost similar than that of the best performing heuristic.

	Axis-aligned					Oblique		
Dataset	TAO-A	GUIDE-A	CART	C5.0	CTT(QF)	TAO-O	GUIDE-O	CTT(F^H)
Balance scale	82.2*	77.4	77.6	75.9	75.4	88.5	85.6	**90.6**
Blood transfusion	77.9	78.8*	76.1	75.7	78.5	78.9	**79.9**	79
Breast cancer	95.9*	94.6	94.6	94.4	94.6	**97.7**	95.6	95.1
Car Evaluation	**96.7**	70.2	85.8	86.6	84.4	91.6*	70.2	85.9
Dermatology	96.1*	95.5	94.2	94.4	88.9	92.3	**97.8**	93.8
Iris	95.4	96.8	95.2	93.7	**97.4**	94.4	94.3	95.8*
Spambase	92.7	92.8*	89.2	91.9	86.3	**93.3**	92.2	92.7
Wine	91.2	91.9*	89.2	91.5	90.2	92	**93.3**	91.1
Average RG	*2.5*	*6.4*	*6*	*5.8*	*6.8*	*2.4*	*4.9*	*3*
Bank marketing 10%	-	-	**89.7**	**89.7**	88	-	-	89.6
Ecoli	-	-	77.3	80.2	80.4*	-	-	**84.2**
German	-	-	71.6	69.5	73*	-	-	**74.2**
Haberman	-	-	66.3	68.4	**73.5**	-	-	72.7
Ionosphere	-	-	88.6*	86.3	88.6*	-	-	**89.8**
Monk1	-	-	74.7	83.1	92.3*	-	-	**99.4**
Monk2	-	-	55.7	60.3*	55.8	-	-	**72.1**
Monk3	-	-	90.2	90.2	**93.5**	-	-	90.3
Pima Indians diabetes	-	-	74.8	74.1	**75.4**	-	-	75.3
Qsar biodegradation	-	-	81.3*	81.1	79.9	-	-	**85.3**
Seeds	-	-	86.8	89.4	92.8*	-	-	**95.1**
Seismic bumps	-	-	92.3	93.2	**94.1**	-	-	93.8
Statlog satellite	-	-	80.8	**84.2**	77.9	-	-	80.1
Tic tac toe	-	-	87.7	91.1*	74.8	-	-	**95.9**
Average RG	-	-	*7.3*	*5.5*	*5.5*	-	-	*0.7*
Total average RG	-	-	*5.9*	*4.7*	*5.1*	-	-	*0.6*

Table 5: Comparison with heuristics : percentage of correctly classified data on 22 datasets for 8 different methods.

	Axis-aligned					Oblique		
Dataset	TAO-a	GUIDE-a	CART	C5.0	CTT(QF)	TAO-o	GUIDE-o	CTT(F^H)
Balance scale	7.2	6.3	4.4	7.6	4*	3	5.8	**1.8**
Blood transfusion	7.4	4.8	5.8	3.2	2.5*	5	**2.4**	3
Breast cancer	3.4	4	2.8*	5	3.2	3	**1.8**	2
Car Evaluation	12.7	4.6	6.4	8	4*	4	3.6	**3.4**
Dermatology	7	6.7	5	6.4	**4**	7	5.1	**4**
Iris	3	2.3	**2**	2.6	3.4	3	2.3	**2**
Spambase	14.4	12.3	5.2	13.2	2.6*	4	15.4	**1.6**
Wine	2.8	2.7*	2.8	3	2.8	5	**2.4**	**2.4**
Average	*7.2*	*5.5*	*4.3*	*6.1*	*3.3*	*4.3*	*4.9*	*2.5*
Bank marketing 10%	-	-	5	4.4	**1**	-	-	3
Ecoli	-	-	**3**	4.8	3.5	-	-	3.8
German	-	-	8	14.6	**2.8**	-	-	3
Haberman	-	-	3	**1.4**	1.8	-	-	1.8
Ionosphere	-	-	2.6*	6.8	2.8	-	-	**1.8**
Monk1	-	-	**2.4**	4.4	4	-	-	2.8
Monk2	-	-	3.2	5.4	**1.6**	-	-	3
Monk3	-	-	2*	2*	2.4	-	-	**1.8**
Pima Indians diabetes	-	-	5.6	7	2.2*	-	-	**1.2**
Qsar biodegradation	-	-	5.2	11.6	**3**	-	-	3.2
Seeds	-	-	2.4*	3.6	3.4	-	-	**2**
Seismic bumps	-	-	4.6	2.2	**0**	-	-	1.4
Statlog satellite	-	-	5.8	15.8	**4**	-	-	**4**
Tic tac toe	-	-	5.2	7.8	4*	-	-	**1.6**
Average			*7.3*	*5.5*	*5.5*			*0.7*

Table 6: Comparison with heuristics : average depth on 22 datasets for 8 different methods.

	Axis-aligned					Oblique		
Dataset	TAO-a	GUIDE-a	CART	C5.0	CTT(QF)	TAO-o	GUIDE-o	CTT(F^H)
Balance scale	24.6	18.1	8.8	23.4	6.8*	5.6	8.3	**2.8**
Blood transfusion	20	8.2	8.2	4.2	4*	10.8	3.5	**3.8**
Breast cancer	5.4	7.6	3.8*	8.2	4.2	7.8	3.2	**2.6**
Car Evaluation	68	6.7	10.6	18.6	5.6*	14.3	5	**4.8**
Dermatology	9.6	8.6	6	7.6	**5**	64	6.2	6*
Iris	4.4	3.3	**3**	3.6	5	8.1	3.3	**3**
Spambase	53.6	53.5	9	39	4.2*	14.8	48	**2**
Wine	5.6	4.6	4*	5.2	4.2	16	3.5	**3.4**
Average	*23.9*	*13.8*	*6.7*	*13.7*	*4.9*	*17.7*	*10.1*	*3.6*
Bank marketing 10%	-	-	11.4	5.8	**2**	-	-	**2**
Ecoli	-	-	**5.6**	9.8	6.5	-	-	7.5
German	-	-	14.2	46.6	4*	-	-	**2**
Haberman	-	-	4.4	**2.4**	3.4	-	-	3
Ionosphere	-	-	3.6*	9.4	4.2	-	-	**2.4**
Monk1	-	-	**3.4**	6.4	9	-	-	4.4
Monk2	-	-	4.2	10	**3.6**	-	-	4.4
Monk3	-	-	**3**	**3**	3.6	-	-	3.2
Pima Indians diabetes	-	-	11	12.8	3.8*	-	-	**2.2**
Qsar biodegradation	-	-	10.8	31	5.5*	-	-	**2**
Seeds	-	-	3.4*	5.2	6.8	-	-	**3.2**
Seismic bumps	-	-	7.6	2.6	**1**	-	-	1.6
Statlog satellite	-	-	9	129.8	**6**	-	-	14.4
Tic tac toe	-	-	17.4	34.6	10.2*	-	-	**2.4**
Average			*7.4*	*19.1*	*4.9*			*3.8*

Table 7: Comparison with heuristics : average number of leaves on 22 datasets for 8 different methods.

7 Conclusion

We introduce four new formulations in order to build Optimal Classification Trees, each with two variants depending on whether the split functions are axis-aligned or oblique. The first formulation is a quadratic formulation of (T), the model introduced by Bertsimas et al. [Bertsimas and Dunn, 2017], that we linearize with Fortet and Glover linearizations. The last formulation is an extension to real-valued datasets of a flow-based formulation [Aghaei et al., 2020]. We prove that these new formulations have a better continuous relaxation than (T) in the axis-aligned case and than (T^H) in the oblique case. We also highlight that for some instances the flow-based formulations have better performances. We present a more efficient version of the algorithm introduced by Bertsimas et al. [Bertsimas and Dunn, 2017] to fit parameters. To enhance the performances on the test sets, we design a post-processing algorithm that move the split functions of a tree as far as possible from the data. Our numerical experiments show that both the Fortet linearization of the quadratic formulation of (T) and the flow-based formulation are faster than (T). Using those formulations together with our parameter fitting algorithm, we significantly reduce the resolution time while maintaining accuracy on test sets. In particular, for oblique trees, the time reduction is greater and the accuracy is even slightly improved. We moreover show that the performances of our new algorithms are competitive with several heuristics of the literature with the advantage of providing much simpler trees. In future work we will focus on further reducing the resolution time to enable the exact resolution of larger datasets.

References

Aghaei, S., Gomez, A., and Vayanos, P. (2020). Learning Optimal Classification Trees: Strong Max-Flow Formulations. arXiv:2002.09142 [cs, math, stat].

Aglin, G., Nijssen, S., and Schaus, P. (2020). Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3146–3153. Number: 04.

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

Blanco, V., Japón, A., and Puerto, J. (2021). Multiclass optimal classification trees with svm-splits. *arXiv preprint arXiv:2111.08674*.

Blanco, V., Japón, A., and Puerto, J. (2022a). A mathematical programming approach to svm-based classification with label noise. *Computers & Industrial Engineering*, 172:108611.

Blanco, V., Japón, A., and Puerto, J. (2022b). Robust optimal classification trees under noisy labels. *Advances in Data Analysis and Classification*, 16(1):155–179.

Blanquero, R., Carrizosa, E., Molero-Río, C., and Romero Morales, D. (2021). Optimal randomized classification trees. *Computers & Operations Research*, 132:105281.

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

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. Taylor & Francis.

Brodley, C. E. and Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning*, 19:45–77.

Carreira-Perpinan, M. A. and Tavallali, P. (2018). Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, 8(8):832.

Chen, T. and Guestrin, C. (2016). Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM.*

Demirović, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., and Stuckey, P. J. (2022). MurTree: Optimal Decision Trees via Dynamic Programming and Search. *Journal of Machine Learning Research*, 23(26):1–47.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

D'Onofrio, F., Grani, G., Monaci, M., and Palagi, L. (2022). Margin optimal classification trees. *arXiv preprint arXiv:2210.10567*.

Doshi-Velez, F. and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv:1702.08608 [cs, stat].

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dunn, J. W. (2018). *Optimal trees for prediction and prescription*. Thesis, Massachusetts Institute of Technology. Accepted: 2018-11-28T15:25:46Z.

Firat, M., Crognier, G., Gabor, A. F., Hurkens, C. A. J., and Zhang, Y. (2020). Column generation based heuristic for learning classification trees. *Computers & Operations Research*, 116:104866.

Fortet, R. (1960). L'algebre de Boole et ses applications en recherche operationnelle. *Trabajos de Estadistica*, 11(2):111–118.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

Glover, F. (1975). Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. *Management Science*, 22:455–460.

Goodman, B. and Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57. arXiv:1606.08813 [cs, stat].

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

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

Jost, L. (2006). Entropy and diversity. *Oikos*, 113(2):363–375. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2006.0030-1299.14714.x.

Lin, J., Zhong, C., Hu, D., Rudin, C., and Seltzer, M. (2020). Generalized and Scalable Optimal Sparse Decision Trees. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6150–6160. PMLR. ISSN: 2640-3498.

Loh, W.-Y. (2002). Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386.

Loh, W.-Y. (2009). Improving the precision of classification trees. *The Annals of Applied Statistics*, 3(4):1710–1737.

Murthy, S. K., Kasif, S., and Salzberg, S. L. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32.

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

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. R. (1993). C4.5: programs for machine learning. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938 [cs, stat].

Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. arXiv:1811.10154 [cs, stat].

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*.

Verwer, S. and Zhang, Y. (2019). Learning Optimal Classification Trees Using a Binary Linear Program Formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1625–1632. Number: 01.

Wachter, S., Mittelstadt, B., and Floridi, L. (2016). Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation.

Wickramarachchi, D. C., Robertson, B. L., Reale, M., Price, C. J., and Brown, J. (2016). Hhcart: An oblique decision tree. *Computational Statistics & Data Analysis*, 96:12–23.

Zharmagambetov, A., Hada, S. S., Carreira-Perpiñán, M. Á., and Gabidolla, M. (2019). An Experimental Comparison of Old and New Decision Tree Algorithms. *arXiv*.

Zhou, J., Gandomi, A. H., Chen, F., and Holzinger, A. (2021). Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics. *Electronics*, 10(5):593.

Zhou, W., Zhang, L., and Jiao, L. (2002). Linear programming support vector machines. *Pattern Recognition*, 35(12):2927–2936.