Learning Customised Decision Trees for Domain-knowledge Constraints

Géraldin Nanfack, Paul Temple, Benoît Frénay

Learning Customised Decision Trees for Domain-knowledge Constraints

Géraldin Nanfack[a,*], Paul Temple[a], Benoît Frénay[a]

[a] *PReCISE Research Center, Namur Digital Institute (NADI), University of Namur, Belgium*

Abstract

When applied to critical domains, machine learning models usually need to comply with prior knowledge and domain-specific requirements. For example, one may require that a learned decision tree model should be of limited size and fair, so as to be easily interpretable, trusted, and adopted. However, most state-of-the-art models, even on decision trees, only aim to maximising expected accuracy. In this paper, we propose a framework in which a diverse family of prior and domain knowledge can be formalised and imposed as constraints on decision trees. This framework is built upon a newly introduced tree representation that leads to two generic linear programming formulations of the optimal decision tree problem. The first one targets binary features, while the second one handles continuous features without the need for discretisation. We theoretically show how a diverse family of constraints can be formalised in our framework. We validate the framework with constraints on several applications and perform extensive experiments, demonstrating empirical evidence of comparable performance w.r.t. state-of-the-art tree learners.

Keywords: Decision Trees, Constraints, Domain Knowledge

*Corresponding author

Email addresses: geraldin.nanfack@unamur.be (Géraldin Nanfack), paul.temple@unamur.be (Paul Temple), benoit.frenay@unamur.be (Benoît Frénay)

1. Introduction

Decision trees are one of the most well-known classes of machine learning models thanks to their interpretability and ability to learn decision rules with relevant features [1, 2, 3]. They are even applied in critical domains involving high-stakes decision-making such as medical diagnosis and finance [4, 5]. Yet, in these critical domains, machine learning models need to be simple to explain, and meet ethical and domain-specific requirements [6]. For example, to predict heart disease, similar to a medical doctor, learned decision trees should take into consideration domain constraints like economic criteria when selecting features. Moreover, to validate decision trees, practitioners and domain experts may require a certain level of performance guarantee for an underrepresented target group of sample [7]. In these scenarios, the lack of satisfaction with domain-specific constraints and goals makes decision trees untrustworthy and likely to be rejected [8]. As an illustration of its importance, the issue of trustworthy artificial intelligence is also a major concern at the EU level [1]. To learn more trustworthy models, constraint enforcement represents a theoretically supported framework to customise the shape of the hypothesis set of models under a wide range of settings. However, designing learning algorithms may be complex, even for decision trees, when ethical and domain constraints must be met.

Recently, several decision tree methods [9, 10, 11, 12] have proposed to formalise the optimal decision tree problem as a mixed integer programming (MIP), a satisfiability (SAT) or a dynamic programming problem whose solutions often lead to small and accurate trees. A strong emphasis has been put on the speed of optimisation, neglecting the possibility of easily modeling prior and domain knowledge. Hence, these current methods are little to no applicable in settings where prior knowledge needs to be formalised, and enforced in order to learn more comprehensible and trustworthy trees. Yet, in domains as critical

[1] Communication from the Commission of 8 April 2019, Building Trust in Human-Centric Artificial Intelligence, COM (2019) 168.

as medicine and justice, the satisfaction of domain-knowledge constraints may be more important than providing a good level of accuracy [13, 14, 15]. This suggests that a broader family of constraints should be considered when modeling the decision tree learning problem. In this paper, we, therefore, model the optimal decision tree problem so as to easily formalise constraints while allowing an explicit control of their complexity through both the number of leaf nodes and the maximum depth. We present a tree representation based on constrained matrices that leads to two generic linear formulations of the optimal decision tree problem. The first formulation targets binary features while the second one copes with continuous features without the need for discretisation. Moreover, in contrast to recent depth-centric models, our tree representation is a branch-like model. We show its flexibility to straightforwardly formalise and enforce a broad class of constraints that include but are not limited to ordering on features, exclusion of features over branches, feature costs, and fairness.

The contributions of this paper can be summed up as follows: (1) we propose a tree representation based on constrained matrices that leads to two new generic linear programming formulations of the optimal decision tree problem allowing to easily integrate domain-knowledge constraints; (2) both formulations give the possibility for users to easily control complexity through the constraints on the number of leaf nodes and the maximum depth of the decision tree; (3) we theoretically show that elements of these matrices give the ability to easily formalise domain knowledge as constraints to improve trustworthiness.

The rest of the paper is organised as follows: Section 2 presents related works, Section 3 introduces our tree representation, and our first generic linear programming formulation for binary features (the second formulation is detailed in Appendix[2] A.1); Section 4 formalises the expression of domain-knowledge constraints; Section 5 shows application to real-world applications; Section 6 benchmarks the performance of our models w.r.t. recent models and discusses the results; Section 7 analyses computational time before concluding.

[2]Appendix is added via the link to "Supplementary Material for on-line publication only".

2. Related Works

Several works exist to learn decision trees under constraints. We thoroughly present them in what follows.

Building upon standard top-down induction trees such as CART [16] and C4.5 [17], many works have attempted to enforce constraints on decision trees by learning in a greedy top-down fashion. The minimum description length (MDL) presented by Quinlan and Rivest [18], inspired the work of Garofalakis et al. [19] that proposed to find a trade-off between the size of the tree and its accuracy with a branch-and-bound algorithm. It learns either the smallest decision tree given the accuracy or, the other way around, retrieves the most accurate decision tree given the size of the tree. Although it is possible to incorporate some domain-knowledge constraints like test and misclassification costs on decision trees via data and feature engineering (*e.g.*, by creating virtual instances or features), the such process remains highly non-trivial [20], supporting, therefore, the common approach of enforcing constraints directly through the model formulation. The former approach is all the more challenging for complex misclassification costs in multiclass settings and for feature acquisition costs. Also, data engineering may raise stability issues in learning procedures, *e.g.*, when misclassification costs are highly different. Related to constraint enforcement through the model formulation, Núñez [21] encodes a hierarchy of features with ISA (Is A) relationships and feature costs on a cost-sensitive measure to learn decision trees in a greedy top-down fashion. By doing that, they suppose that learned trees will make more sense for domain experts. López-Vallverdú et al. [22, 23] also present an algorithm based on the priority and relevance of features. They propose to modify the list of features on a given node and also formalise background knowledge using healthcare criteria. However, top-down algorithms often produce sub-optimal trees which may result in poor solutions when constraints need to be enforced [24].

Some works try to learn optimal solutions under tree-structure constraints by enumerating possible solutions via dynamic programming. Garofalakis et al.

[25] build up on their previous work and learn an overfitted decision tree that they prune to satisfy constraints. One major drawback of post-pruning methods is that new interesting rules are not learned since the top of the tree remains unchanged while the bottom is cut-off. To get rid of it, Nijssen and Fromont [26] present the DL8 algorithm using dynamic programming. Later, it is transposed into a more general framework [24] which uses an item-set mining approach. They are able to learn optimal decision trees for different types of constraints (e.g., size of the tree and test costs). As stated by the authors, DL8 needs memory to encapsulate the huge amount of item sets. To address the limitations of DL8, Aglin et al. [27] propose DL8.5 that focuses on the task of minimising the misclassification error under depth constraint. To enforce the depth constraint, the authors had to drastically modify the DL8 algorithm. Authors highlight that *"optimisation [of DL8] is hard to combine with a constraint on the depth of a tree"* [27], to argue why they had to completely change DL8. As a result, the size of the tree is left aside and cannot be constrained, at the profit of accelerating learning through a branch-and-bound search. Moreover, without requiring another drastic change, DL8.5 cannot enforce constraints over test costs or the hierarchy of features which are both tree-structure dependent constraints [24].

Inspired by Angelino et al. [28], Hu et al. [29] proposed optimal sparse decision trees (OSDT) that also use branch-and-bound search, but are limited to binary classification. Analytic bounds are used to prune the search space while the number of leaves is constrained using a regularised loss function that balances accuracy and the number of leaves. Because OSDT and its extended version GOSDT [30] use a customised branch-and-bound search, similarly to DL8.5, integrating tree-structure dependent constraints that are not directly expressed in the objective function will need to completely accommodate the learning algorithm. Nonetheless, our work shares similarities with OSDT since our model also allows constraining directly the number of leaves.

Because of the speed improvement of machines, several works [10, 9, 12, 31] propose to exploit MIP and SAT solvers to learn optimal decision trees by constraining the depth. More specifically, Narodytska et al. [11] and Florent

Constraints/Setting	BinOCT	[32]	DL8.5	OSDT[3]
Number of leaves	✗	✗	✗	✓
Depth	✓	✓	✓	✓
Hierarchy/ordering	✓	✓	✗	✓
Features/Misclassification costs	✗	✗	✗	✗
Minimum #instances in the same leaf	✗	✓	✓	✗
Instances belonging to the same leaf	✓	✓	✗	✗
Range of threshold splits cont. features	✗	✗	✗	✗
Multiclass setting	✓	✗	✓	✗

Table 1: Comparison of state-of-the-art tree learning methods in terms of their ability to enforce the constraints considered in this paper. ✓ (resp. ✗) indicates that the current method allows (resp. does not allow) the integration of a specific constraint or is (resp. is not) able to deal with the multiclass setting.

[31] use SAT solvers to learn the smallest tree for perfect classification. We do not tackle the same problem in this paper. Bertsimas and Dunn [10] present a MIP formulation of the optimal decision tree problem for a given depth. Their model (called OCT) handles both univariate and multivariate splits. As stated in [29], OCT is not easily reproducible and no public code is available [29].

A more recent MIP formulation (BinOCT) has been proposed by [9] to be efficient in computations. To speed up optimisation, BinOCT considers full and complete binary trees under depth constraints. This raises a question about the optimality of their learned trees when the optimal solution is not full and complete. In particular, in presence of domain constraints, BinOCT will not find a solution if trees do not need to be full and complete. Moreover, in order to accelerate optimisation, BinOCT does not keep track of misclassified instances through decision variables, making it impractical to change objective function in order to integrate instance-dependent constraints like misclassifition costs. The work [32] uses constraint programming to learn optimal decision trees, but only under depth constraint, and targets only binary classification.

To date, previous models have focused on accelerating the learning of *optimal* decision trees. This work instead focuses on the enforcement of a broad class of

[3]PyGOSDT [30], an improvement of OSDT, does multiclass but only through one-vs-all.

constraints for trustworthiness. As it is not the focus of previous works, they do not natively offer mechanisms to do that. For example, they may allow the setting of a depth constraint, but shallow decision trees may be over-simplistic and not meet domain-knowledge constraints. Such decision trees may be rejected by domain experts [1, 15]. Table 1 illustrates this with a representative sample of types of constraints that the above methods are (or are not) able to cope with, so as to incorporate a broad class of domain knowledge. For example, BinOCT, DL8.5 and Verhaeghe et al. [32] are depth-centric models that are not directly formulated to find the optimal decision tree for a fixed number of leaf nodes. Also, while BinOCT with its discretisation heuristic can cope with continuous features, all other methods require binary features. As a result, they (BinOCT included) cannot integrate the constraint that threshold splits must belong to a given interval, which is useful when seeking for relevant decision rules with respect to the domain expertise [33, 4, 34]. Some of the above methods could be modified at a significant cost to take into account some of the considered constraints (or even other ones). However, as they are not designed for that purpose, it is neither formally nor technically obvious how to do so (e.g., the transition from DL8 to DL8.5). This motivates our work to make constraint enforcement easier, thanks to a specifically designed tree representation.

The above analysis of the state of the art shows the need for a general framework to i) efficiently handle complexity control for generalisation of decision trees; ii) and learn decision trees under domain constraints that are important for safety and trustworthiness. Therefore, we propose a tree representation based on constrained matrices that leads to two generic flexible linear programming formulations of the optimal decision tree problem, which eases the integration of a broad class of constraints. Indeed, based on a branch-like tree representation, our formulations take into account structural constraints such as the number of leaves and the maximum depth. Furthermore, thanks to the constrained matrices encoding this representation, our method allows us to enforce a broad class of constraints, which include those listed in Table 1.

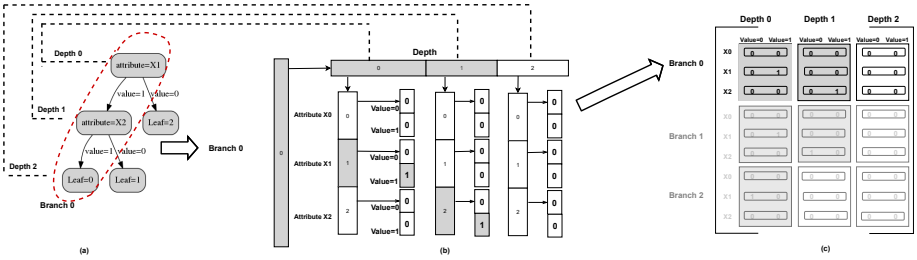


Figure 1: Tree encoding detailed for a branch. (a) An example of a decision tree on which depths are depicted and the first branch of the tree is circled. (b) The corresponding encoding for only the branch 0 (circled in (a)) with pointers representing how to pass from one dimension to another (c) The encoding matrix Q for all branches. The branch label is at dimension 0, the depth at dimension 1, attributes and their values are at dimension 3 and 4. Note that here, attribute X_0 is not selected on the tree in (a) but it is encoded in the matrix Q.

3. Tree Representation and the New Formulation to Enforce Constraints on Decision Trees

This section presents our tree representation and the first formulation of the tree learning classification problem via binary variables and linear constraints.

Let us consider a dataset where all features have been transformed into binary features (numeric features being discretised and binarised). A decision tree from this dataset is characterised by its number of leaf nodes L, its maximum depth K, and its size. In what follows, $\mathbf{X} \in \{0,1\}^{N \times M \times V}$ denotes the dataset (without labels), N is the number of instances, M is the number of features and V is the number of values which can be taken by a feature. Here, it is assumed that data only have binary features, thus $V = 2$. However, the following can be easily extended to $V > 2$ for V-array trees. $\mathbf{T} \in \{0,1\}^{N \times C}$ is the matrix providing the labels of instances (into their one-hot-encoding form) and C is the number of classes. Finally, $[n]$ denotes the set $\{0, 1, ..., n-1\}$, for $n \in \mathbb{N}$.

3.1. Tree Representation

We need a convenient and efficient way to encode trees so as to easily formalise domain constraints such as those related to the path of decisions (e.g., ordering on tests and test costs on branches). Rather than a tree representation

that decomposes a tree into a set of nodes, our decision trees are decomposed in terms of the set of branches. We propose to encode branches with the (multidimensional) matrix $Q \in \{0,1\}^{L \times K \times M \times V}$. To illustrate, $Q_{ijlp} = 1$ if the p-th value of the l-th feature is selected, at depth j of the i-th branch. Therefore, $Q_{i,*}$ encodes the i-th branch of the tree, $Q_{ij,*}$ encodes the selection of features and its values at depth j of branch i, and $Q_{ijl,*}$ is the vector encoding whether the l-th feature has been selected[4] at depth j of branch i. Figure 1 shows an example of a decision tree with its encoding matrix. As an illustration, on the branch 0 at depth 0 of the decision tree of Figure 1 (a), the attribute X_1 is selected with value 1. This means that $Q_{0,0,1,1} = 1$, as shown in Figure 1 (c).

From this description, Q can be represented as a 4-dimensional binary matrix. The following constraints need to be added in order to encode the selection of features over branches of trees. We give an example for the first constraint. Similar examples can be derived for the rest of the constraints.

- The same feature is chosen at the top of all branches:

$$\forall i \in [L], a \in [L], l \in [M] \colon \sum_{p \in [V]} Q_{i,0,l,p} = \sum_{p \in [V]} Q_{a,0,l,p}. \tag{1}$$

  Linking it to Figure 1, at depth $j = 0$, the attribute X_1 (i.e., $l = 1$) is selected on branches $i = 0$ and $a = 1$, i.e., $\sum_{p \in [V]} Q_{i,0,l,p} = \sum_{p \in [V]} Q_{a,0,l,p}$.

- No more than one feature must be chosen on a branch i, at depth j:

$$\forall i \in [L], j \in [K] \colon \sum_{l \in [M]} \sum_{p \in [V]} Q_{ijlp} \leq 1. \tag{2}$$

- Each feature is chosen at most once on a branch:

$$\forall i \in [L], l \in [M] \colon \sum_{j \in [K]} \sum_{p \in [V]} Q_{ijlp} \leq 1. \tag{3}$$

- For every branch, if a feature is chosen at depth j, on all previous depths $0, ..., j-1$, a feature must be selected:

$$\forall i \in [L], j \in [K-1] \colon \sum_{l \in [M]} \sum_{p \in [V]} Q_{ijlp} \geq \sum_{l \in [M]} \sum_{p \in [V]} Q_{i(j+1)lp}. \tag{4}$$

[4] The l-th feature is selected if at least one its value is $\neq 0$.

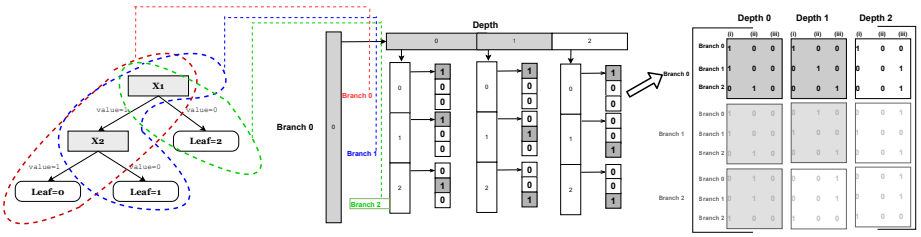


Figure 2: Relationship between branches (matrix Z). Left: branches of the tree. Center: the relationship between the first branch (branch 0) and the other branches; the deepest vector (with 3 bits horizontally aligned) defines the 3 possibilities of pair of branches given a depth: (i) branches are equal up to that depth, (ii) branches are siblings, (iii) branches are different with a variable or a value before the given depth. Right: the corresponding matrix Z which encodes the relationship between all the branches.

From what has been described, Q models a set of L branches with at most K features selected per branch. However, for Q to represent a valid tree, relations between branches need to be defined. In particular, two branches i and a are *siblings at depth* j if all features and values that are selected on branch i and a until depth j, are equal, but on depth j, chosen features to remain the same between the two branches and only their values differ (e.g., branch 1 and 2 at depth 0). Formally, branch relations are encoded using the 4-dimensional matrix [5] $Z \in \{0,1\}^{L\times K\times L\times 3}$ and we define three types of relations: (i) $Z_{ija,0} = 1$ if all the features and their values on branches i and a are equals up to and including depth j; (ii) $Z_{ija,1} = 1$ if branches i and a are siblings at depth j; and (iii) $Z_{ija,2} = 1$ if it exists one feature or one value of feature selected on depth $j_1 < j$ which differs from branches a and i. For example, in Figure 2, at depth 1, branches 0 and 1 are siblings, so $Z_{011,1} = 1$; and at depth 1 branches 0 and 2 fall into the third case; so $Z_{012,2} = 1$.

With the above definitions for the Q and Z matrices, we can now express the constraints that a valid tree representation should satisfy.

[5] The Symmetry on this matrix Z w.r.t. branch indexes i and a at their respective dimensions 0 and 2 has been technically broken in the implementation by considering that $i < a$.

- All branches must be pairwise different:

$$\forall i \in [L], a \in [L] \setminus \{i\} \colon Z_{i,K-1,a,0} = 0. \tag{5}$$

- Every pair of branches matches only one of the previous cases:

$$\forall i, a \in [L], j \in [K] \colon \sum_{q \in \{0,1,2\}} Z_{ijaq} = 1. \tag{6}$$

- Every pair of branches in case (i) at depth j must have selected on depths $j_1 \leq j$ the same feature and have the same values of those features:

$$\forall i, a \in [L], j \in [K], j_1 \in [j+1], p \in [V] \colon Z_{ija,0} = 1 \Rightarrow Q_{ij_1lp} = Q_{aj_1lp}. \tag{7}$$

- On a branch, if no feature is chosen at a specific depth j, no other branch can be in case (i) at depth $j-1$:

$$\forall i \in [L], a \in [L] \setminus \{i\}, j \in [K-1] \colon Z_{ija,0} \leq \sum_{l \in [M]} \sum_{p \in [V]} Q_{i,j+1,lp}. \tag{8}$$

- Every pair of branches in case (iii), which are different on at least one depth before j must have a depth where feature/value is different. This relationship appears when earlier on depth $j_1 < j$, the branches were siblings. Therefore, the constraint can be simply written as:

$$\forall i \in [L], j \in [K], j_1 \in [j], a \in [L] \setminus \{i\} \colon Z_{ija,2} \geq Z_{ij_1a,1}. \tag{9}$$

- If a feature is selected on a depth of a branch, the branch must have at least one sibling:

$$\forall i \in [L], j \in [K] \colon \sum_{a \in [L] \setminus \{i\}} Z_{ija,1} \geq \sum_{l \in [M]} \sum_{p \in [V]} Q_{ijlp}. \tag{10}$$

- Every pair of sibling branches must be equal (in terms of features and values) up to depth $j-1$:

$$\forall i, a \in [L], j \in [K] \setminus \{0\}, j_1 \in [j], l \in [M], p \in [V] \colon Z_{ija,1} = 1 \Rightarrow Q_{ij_1lp} = Q_{aj_1lp}. \tag{11}$$

- Every pair of sibling branches on depth j must have the same feature chosen on this depth:

$$\forall i, a \in [L], j \in [K], l \in [M] \colon Z_{ija,1} = 1 \Rightarrow \sum_{p \in [V]} Q_{ijlp} = \sum_{p \in [V]} Q_{ajlp}. \quad (12)$$

- Values of features for each pair of sibling branches at depth j must be different if the feature is selected and equal to zero otherwise:

$$\forall i, a \in [L], l \in [M], j \in [K], p_1 \in [V] \colon Z_{ija,1} = 1 \Rightarrow Q_{ijlp_1} + Q_{ajlp_1} = \sum_{p \in [V]} Q_{ijlp}. \quad (13)$$

- If no feature is selected at depth j of the branch i, no branch can be a sibling of another branch i:

$$\forall i \in [L], a \in [L] \setminus \{i\}, l \in [M], j \in [K], p \in [V] \colon Z_{ija,1} \leq \sum_{p \in [V], l \in [M]} Q_{ijlp}. \quad (14)$$

Our tree representation is composed of matrices Q and Z on which constraints (1–14) are applied. Matrices Q and Z respectively have $L{\times}K{\times}M$ and $L^2{\times}K$ variables. To learn trees, one needs to create an objective function where instance-dependent variables need to be explicitly introduced, as shown hereafter.

3.2. Encoding Global Objective Functions

This section introduces variables and constraints used to encode the global objective function. Section 4 will show how constraints can be easily enforced.

Let $S \in \{0,1\}^{N \times L}$ denote the mapping between the set of examples of the dataset and the set of leaves (e.g., $S_{ei} = 1$ if example e belongs to the i-th leaf/branch and 0 otherwise). Let $H \in \{0,1\}^{L \times C}$ denote the mapping between the set of branches and the set of classes (e.g., $H_{ic} = 1$ if c is the predicted class of branch i and 0 otherwise). Finally, $R \in \{0,1\}^N$ defines the $0-1$ loss over the example e. For recall, $\mathbf{T} \in \{0,1\}^{N \times C}$ represents labels of the dataset, as defined at beginning of this section.

The following constraints describe how to encode global objective functions.

- Each example must belong to one leaf:

$$\forall e \in [N] \colon \sum_{i \in [L]} S_{ei} = 1. \tag{15}$$

- If an example belongs to a leaf node, all the features/values chosen on the corresponding branch must be the same as this example:

$$\forall e \in [N], i \in [L], j \in [K], l \in [M], p \in [V] \colon$$

$$Q_{ijlp} = 1 \Rightarrow S_{ei} \leq 1 - \mathbf{X}_{elp} + Q_{ijlp}, \tag{16a}$$

$$Q_{ijlp} = 1 \Rightarrow S_{ei} \leq 1 + \mathbf{X}_{elp} - Q_{ijlp}. \tag{16b}$$

- Each branch of a leaf node must predict exactly one class:

$$\forall i \in [L] \colon \sum_{c \in [C]} H_{ic} = 1. \tag{17}$$

- The class of a leaf node (or branch) is the majority vote of examples belonging to this branch:

$$\forall i \in [L], c_1, c_2 \in [C] \colon H_{ic_1} = 1 \Rightarrow \sum_{e \in [N]} S_{ei} * \mathbf{T}_{ec_1} \geq \sum_{e \in [N]} S_{ei} * \mathbf{T}_{ec_2}. \tag{18}$$

- The error of the predicted class of an example is equal to one if the class of its branch is different from the true class and equal to zero, otherwise:

$$\forall e \in [N], c \in [C], i \in [L] \colon$$

$$S_{ei} = 1 \Rightarrow R_e \geq -H_{ic} + \mathbf{T}_{ec}, \tag{19a}$$

$$S_{ei} = 1 \Rightarrow R_e \leq 2 - H_{ic} - \mathbf{T}_{ec}. \tag{19b}$$

Using the above results, the misclassification error of the tree is

$$\sum_{e \in [N]} R_e. \tag{20}$$

This objective function can be used by solvers to search for an optimal tree.

Proposition 1. *The complexity $\mathcal{C}$ in terms of the number of variables of the entire constraint program of our formulation is $\mathcal{O}\Big(L \times K(L+M) + N \times L\Big)$. It can be reduced to $\mathcal{O}\Big(L(L+M+N)\Big)$ for shallow trees when prior values of L and K are known.*

Proof. The tree encoding complexity (in terms of the number of variables) is $\mathcal{O}(L \times K \times M + L^2 \times K)$. The objective function encoding is $\mathcal{O}(N \times L)$. Then, the total complexity is

$$\mathcal{C} = \mathcal{O}(L \times K \times M + L^2 \times K + N \times L) = \mathcal{O}\Big(L \times K(L + M) + N \times L\Big)$$

For shallow trees, K is small, therefore less than a fixed maximum value. So, $\mathcal{C} = \mathcal{O}(L^2 + L \times M + L \times N)$. □

The time to find the optimal solution increases proportionally with the size of the dataset and the number of leaves. Another important insight from Proposition 1 is the fact that, in terms of the number of variables, the complexity $\mathcal{C}$ highly varies depending on which term is the biggest term between N and $K(L + M)$. In particular, N is not usually tunable (except with sub-sampling) whereas M could be because of the binarisation step. Hence, Proposition 1 shows that in practical implementations with binary features, M should be ideally negligible w.r.t. N (i.e., a $o(N)$), when prior values of K and L are known.

Constraints 7, 11, 12, 13, 16a, 16b, 18, 19a and 19b are not directly expressed in a linear form. They are presented in a form called *indicator* constraints, but modern linear programming solvers generally prefer this form. Nonetheless, they can be linearised using big-M constraints (e.g., constraint 7 can be written as $Q_{ijlp} - Q_{ajlp} \leq \mathbf{M} * (1 - Z_{ija,0})$, with $\mathbf{M}$ being a big positive number so that when $Z_{ija,0} = 1$, the left part of the inequality must be null).

It is worth mentioning that, in contrast to [10, 32, 9], our first model does not rely on a specific type of solver since we exploit only binary variables and linear constraints. Therefore, it has the advantage to be implementable on CP solvers as well as MIP and ILP solvers.

4. Formalising Domain Knowledge as Constraints

With the tree representation and the first formulation (the second one is detailed in Appendix A.1) presented in the previous section, in the following,

we show how useful domain knowledge can be straightforwardly formalised as constraints and integrated into our models.

4.1. Ordering of Features

It often appears that domain experts have a prior belief in a specific ordering of features. This prior may directly come from their background knowledge or hard material constraints which favor asking patients to perform tests before others. As an illustration, to predict liver diseases, doctors will ask patients their age before performing a *bilirubin test* since the interpretation of this test depends on the age of the patient [35]. Therefore, a comprehensible decision tree for domain experts should encode this prior knowledge. The constraint "feature $\mathbf{X_{l_2}}$ must not appear before $\mathbf{X_{l_1}}$" is expressed as

$$\forall i \in [L], j \in [K] \setminus \{0\}, \sum_{j_1 \in [j]} \sum_{p \in [V]} Q_{ij_1 l_2 p} \leq 1 - \sum_{p \in [V]} Q_{ijl_1 p}.$$

4.2. Test Costs on Features

Experts can also be constrained by economic considerations to ask patients to do tests before taking decisions [21]. A good illustration of this importance is the case of the application of machine learning algorithms in domains with a lack of available material resources to perform medical tests. Hence, one can specify test costs on features in order to limit the total cost to perform before reaching a decision. With our framework, it is possible to learn trees with a maximum classification cost τ on a cost-sensitive dataset. This constraint becomes

$$\forall i \in [L] \colon \sum_{j \in [K]} \sum_{l \in [M]} \sum_{p \in [V]} Q_{ijlp} * \text{cost(l)} \leq \tau.$$

where cost(l) denotes the test cost of feature $\mathbf{X_l}$.

4.3. Expected Cost for Classification

As for the same reasons with test costs, certain applications may require to constrain the expected classification cost. The weighted test cost of a branch i

is $\frac{1}{N}\sum_{e\in[N]}\sum_{l\in[M]} S_{ei} * \text{cost}(l) * \left(\sum_{p\in[V]} Q_{ijlp}\right)$. As this previous term is non-linear, it is possible to linearise it with variables N_{ij} representing the weighted test cost of the branch i at depth j, with the constraints:

$$\forall i \in [L], j \in [M], l \in [L], \sum_{p\in[V]} Q_{ijlp} = 1 \Rightarrow N_{ij} = \text{cost}(l) * \sum_{e\in[N]} S_{ei},$$
$$\text{and } \forall i \in [L], j \in [M], \sum_{l\in[M]p\in[V]} Q_{ijlp} = 0 \Rightarrow N_{ij} = 0.$$

Therefore, the expected cost for classification is $\text{Ecost} = \frac{1}{N}\sum_{i\in[L]}\sum_{j\in[K]} N_{ij}$. Imposing a maximum expected cost for a decision tree can be stated as

$$\sum_{i\in[L]}\sum_{j\in[K]} N_{ij} \leq N * \tau.$$

4.4. Number of Instances on Leaf Nodes

To reduce the growth of a tree, one can also give the minimum number of instances on leaf nodes. This can be easily done by adding

$$\forall i \in [L], \sum_{e\in[N]} S_{ei} \geq \text{ minNumberInstances}.$$

4.5. Instances that Must Be in the Same Leaf

It often appears that datasets come with instance names and domain experts may have background knowledge of specific instances. Similarly to clustering [36], practitioners may want to impose that certain instances satisfy the same decision rule, i.e., belong to the same leaf node. One can also specify that two instances must be in the same leaf node with the proposed framework using the constraint expressed as

$$\forall i \in [L], S_{e_1,i} = S_{e_2,i}, \text{ if } e_1 \text{ and } e_2 \text{ has to be in the same leaf node.}$$

4.6. Presence or Exclusion of a Feature Over the Tree or Over a Branch

It often appears that in several situations (e.g., due to noisy data), the selected features of a decision tree are not properly the domain-related relevant features [23]. In such situations, domain experts may not trust the learned tree.

Using our formulation, it is possible to impose the selection of a feature $\mathbf{X}_l$ (without even knowing where it should be selected) thanks to the constraint

$$\sum_{i\in[L]}\sum_{j\in[K]}\sum_{p\in[V]} Q_{ijlp} \geq 1.$$

Additionally, two or more features may be redundant in a decision rule (i.e., over a specific branch). For some specific reasons, they should not be selected simultaneously in a branch (i.e., decision-wise) or in the whole tree (i.e., model-wise). Domain experts can provide this knowledge which can be infused as constraints. It can be guaranteed that two features $\mathbf{X}_{l_1}$ and $\mathbf{X}_{l_2}$ do not appear on the same branch of a decision tree with

$$\forall i \in [L], \sum_{j\in[K]}\sum_{p\in[V]} Q_{ij,l_1,p} + Q_{ij,l_2,p} \leq 1.$$

$\mathbf{X}_{l_1}$ and $\mathbf{X}_{l_2}$ will not appear simultaneously into the tree if

$$\sum_{i\in[L]}\sum_{j\in[K]}\sum_{p\in[V]} Q_{ij,l_1,p} + Q_{ij,l_2,p} \leq 3.$$

4.7. Fairness through Demographic Parity and Minimum Accuracy for a Group

4.7.1. Demographic Parity

One of the most popular measures to quantify the unfairness is the demography parity [37, 38]. Demographic parity fairness aims to ensure that the predictions of a classifier do not depend on a sensitive feature z such as gender or race, i.e., $p(\hat{y} = 1|\mathbf{x}_z) = p(\hat{y} = 1|\mathbf{x}_{\bar{z}})$ [39]. In practice, the demographic parity fairness is evaluated on a sample, through the difference of demography parity (DDP) given by

$$\text{DDP} = \Big|\frac{\big|\{\mathbf{x}\in\mathcal{D}; \hat{y}=1, \mathbf{x}_z=1\}\big|}{\big|\{\mathbf{x}\in\mathcal{D}; \mathbf{x}_z=1\}\big|} - \frac{\big|\{\mathbf{x}\in\mathcal{D}; \hat{y}=1, \mathbf{x}_z=0\}\big|}{\big|\{\mathbf{x}\in\mathcal{D}; \mathbf{x}_z=0\}\big|}\Big|.$$

Let N_z (resp. $N_{\bar{z}}$) be the number of samples belonging to the first (second) category of the binary sensitive feature, N_z^+ (resp. $N_{\bar{z}}^+$) be the number of positively predicted samples belonging to the first (resp. second) category of the sensitive feature. Therefore, the DDP can be expressed as $\text{DDP} = \big|N_z^+/N_z -$

$N^+_{\bar{z}}/N_{\bar{z}}|$. Looking back at our tree formulation, thanks to the partitioning of the branch representation, we can partition samples according to branches/leaves $i \in [L]$, create variables N^+_{iz} (resp. $N^+_{i\bar{z}}$) for each branch and express the DDP as $\left|\frac{N_{\bar{z}} * \sum_{i\in[L]} N^+_{iz} - N_z * \sum_{i\in[L]} N^+_{i\bar{z}}}{N_{\bar{z}} * N_z}\right|$. Hence, using our tree formulation, a decision tree satisfies the ϵ-DDP if

$$-\epsilon * N_{\bar{z}} * N_z \leq N_{\bar{z}} * \sum_{i\in[L]} N^+_{iz} - N_z * \sum_{i\in[L]} N^+_{i\bar{z}} \leq \epsilon * N_{\bar{z}} * N_z,$$

where N^+_{iz} and $N^+_{i\bar{z}}$ are variables defined thanks to the following constraints:

$$\forall i \in [L], H_{i,1} = 0 \Rightarrow \begin{cases} N^+_{i\bar{z}} = 0 \\ N^+_{iz} = 0 \end{cases}, H_{i,1} = 1 \Rightarrow \begin{cases} N^+_{i\bar{z}} = \sum_{e\in[N]} S_{ei} * (1 - \mathbf{X}_{ez}) \\ N^+_{iz} = \sum_{e\in[N]} S_{ei} * \mathbf{X}_{ez} \end{cases}.$$

4.7.2. Minimum Accuracy on an Underrepresented Group

In the same direction, if one would like to guarantee that a certain percentage ϵ of instances from a targeted group G should not be misclassified, it can be easily imposed using the following constraint: $\sum_{e\in G} r_e \leq (1 - \epsilon) * |G|$.

More broadly, it is important to note that, by drawing inspiration from demographic parity measures and minimum accuracy, other constraints called in [7] as *rate constraints* (evaluated using input/outputs of models) can be formalised and expressed within the proposed formulation.

5. Domain-knowledge Constraint Enforcement on Real-world Data

This section demonstrates empirically the capability of our model to enforce various types of domain-knowledge constraints. We now refer to our decision trees as *CPTrees*. Following the methodology in [40] and [7], we experimentally validate the ability of our framework to enforce a broad class of domain-knowledge constraints on several real-world applications with prior domain knowledge that should be enforced. In the following, for each application, we briefly discuss the results obtained with CPTree and constrained CPTree in terms of trees and decision rules. In order to have a point of comparison, we also include CART in our experiments, although any other method benchmarked in

Section 6 could be chosen. We report the accuracy in our experiments to study whether constraint enforcement impacts it: constrained decision trees should remain reliable and accurate. Finally, datasets are split using the $67 - 33\%$ train-test percentage (except on COMPAS with the $40 - 60\%$).

5.1. Ordering Constraint Applied to the Prediction of Breast Cancer Survival

The first application is related to the *Haberman's survival* dataset [41], which contains features like *age* and *positive_nodes* (number of positive axillary nodes detected), etc. The goal is to predict whether a patient survives after surgery for breast cancer. In this domain, breast cancer is extremely violent for younger patients because it can weaken the patient and make the surgery dangerous. On the other hand, older patients usually have difficulties recovering from surgeries (in general) since it puts the body under high stress. In between these two categories, patients have more chances to survive breast cancer surgery [42]. Moreover, it is established that knowing the age, the evolution of *positive_nodes* is piece-wise linear [39] and that the feature *age* precedes the feature *positive_nodes* on a causal directed graph [43]. Therefore, a medically valid decision tree should first select the feature *age* before the feature *positive_nodes*.

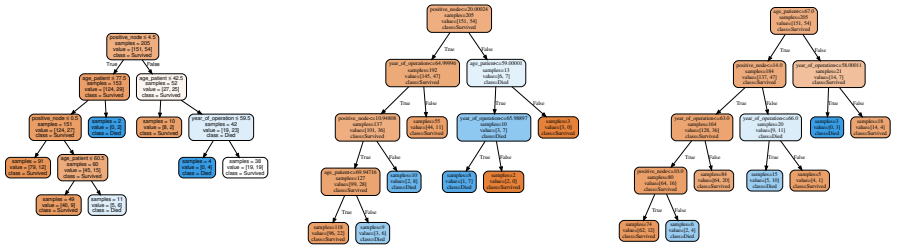
Results. Figure 3 shows learned trees (CART, unconstrained, and constrained CPTrees) from this dataset. It can be observed that both CART and the unconstrained CPTree violate the ordering constraint. In contrast, the constrained CPTree does not only satisfy the constraint but also, the new top feature is the feature *age*, which appears to be to some extent more natural since it is likely to be the first question a medical doctor would ask a breast-cancer patient. Additionally, Table 2 confirms that this prior knowledge is in phase with data since predictive accuracy is slightly improved for the constrained CPTree.

5.2. Must-be-selected Constraint Applied to Diabetes Prediction

Here, we study the problem of predicting diabetes on patients of the dataset *Pima Indian diabetes* [41]. Patients of this dataset are women of at least twenty-one years old [44]. The dataset contains socio-demographic features such as

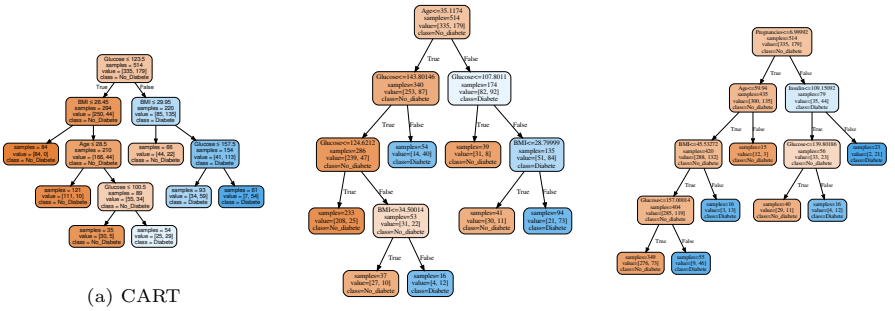
	Haberman's survival		Diabetes	
	Train	**Test**	**Train**	**Test**
CART	77.07	71.28	79.96	68.50
CPTree	80.97	68.31	81.90	68.89
C-CPTree	78.53	73.26	79.57	74.40

Table 2: Accuracy of CART, (unconstrained) CPTree and (constrained CPTree) C-CPTree.



(a) CART (b) Unconstrained CPTree (c) Constrained CPTree

Figure 3: Ordering constraints. Decision trees were obtained on Haberman's survival dataset using (a) CART, (b) CPTree without constraints, and (c) CPTree with the constraint: "feature *age_patient* must appear before feature *positive_nodes*".



(a) CART (b) Unconstrained CPTree (c) Constrained CPTree

Figure 4: Must-be-selected constraints. Decision trees were obtained on the Diabetes dataset using (a) CART, (b) CPTree without constraints, and (c) CPTree with the constraint: "feature *pregnancy* must appear on the tree". Zoomed versions of the trees are left in Appendix B.2.

age, clinical features such as body max index (BMI), and pregnancy-related features. In this use-case, we impose the constraint that a feature related to the pregnancy (*pregnancy*) should be selected on the tree. Indeed, women who have been pregnant may have developed *gestational diabetes*, which may result

	Train	Test
CART	83.33	53.33
CPTree	85.00	56.67
P-CART	80.00	63.63
C-CPTree	83.33	66.67

Table 3: Accuracy of CART, CPTree, P-CART (trained without the *L-CORE* feature) and constrained CPTree (C-CPTree) on Post-operative data.

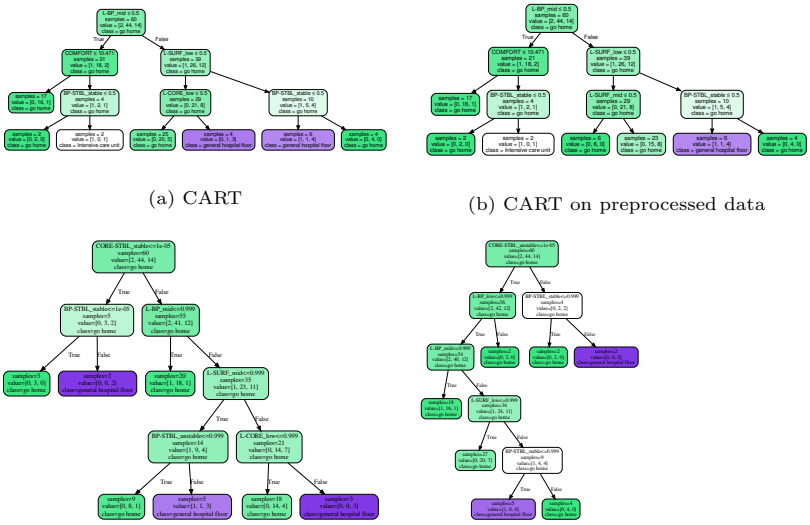
in diabetes after giving birth [45].

Results. Figure 4 shows learned trees from this diabetes dataset. According to the figure both CART and unconstrained CPTree select the same features but with different decision rules. They also fail to select a feature related to pregnancy. With these trees, it is difficult to quickly differentiate patients who are likely to have developed gestational diabetes. In contrast, when infusing the prior information of the selection of the *pregnancy* feature the constrained CPTree provides decision rules where it may be possible to differentiate those patients. Furthermore, as shown in Table 2, this prior knowledge does not harm predictive accuracy but helps to better generalise on unseen data.

5.3. Exclusion Constraint Applied to Prediction of Post-operative Action

This application aims to predict whether a patient should stay in the same service, go to an intensive care unit or go back home for recovery after surgery. We use the *Post-operative* dataset [41]. Here, from prior knowledge, on average, the difference between the core temperature (*L-CORE*) and the surface temperature (*L-SURF*) is generally constant. So it would be surprising and useless to select these two (strongly correlated) features on the same branch.

Results. Table 3 shows the performance of CART, CPTree, and P-CART (CART with the feature *L-CORE* removed). It can be seen that adding this prior knowledge does not impair predictive accuracy. It further improves generalisation. Moreover, as shown in Figure 5, both CART and CPTree violate the constraint while P-CART and C-CPTree satisfy by design.



(a) CART

(b) CART on preprocessed data

(c) Unconstrained CPTree

(d) Constrained CPTree (C-CPTree).

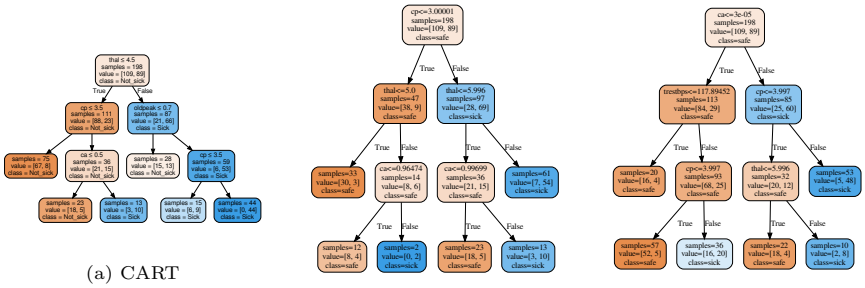
Figure 5: Exclusion constraints. Decision trees obtained on the Post-operative patient dataset with (a) CART without constraints, (b) CPTree without constraints, (c) CART with one of the two highly correlated features removed, (d) CPTree with the constraint: "*L-SURF* (surface temperature) and *L-CORE* (internal temperature) are mutually exclusive in a branch".

5.4. Minimum Accuracy on an Underrepresented Group and Test Cost Constraints Applied to Heart Disease Prediction

In this application, we use the *heart disease dataset* for heart disease prediction. In this application, men are usually of higher risk to develop heart disease than women. Sick women represent therefore an underrepresented group in this domain application and very often they present atypical symptoms compared to men [46]. Since even medical doctors have to be cautious [47] when exam-

	Train	**Test**	**Train Group**	**Test Group**
CART	82.32	77.78	68.75	55.56
CPTree	85.35	83.84	68.75	66.67
C-CPTree	81.82	79.80	93.75	88.89

Table 4: Accuracy of CART, CPTree, and constrained CPTree (C-CPTree) on heart disease. Group represents the group of sick women patients.



(a) CART (b) CPTree (c) Constrained CPTree

Figure 6: Minimum accuracy on a targeted group. Decision trees obtained on the heart disease dataset with (a) CART without constraints and (b) CPTree without constraint and (c) CPTree with the constraint: "90% of sick women should not be misclassified".

	Train	**Test**	**Train DDP**	**Test DDP**
CART	83.60	83.76	8.09	7.49
CPTree	83.14	83.45	12.25	12.80
C-CPTree	81.64	82.18	1.68	1.37

Table 5: Accuracy and DDP of CART, CPTree, constrained CPTree (C-CPTree) on Compas.

ining women patients for this disease, it is likely that a classifier will struggle to correctly classify these examples. In order to enhance the possible trust of learned decision trees, we, therefore, impose the constraint that a high percentage (90%) of sick women should not be misclassified. We additionally impose the constraint on test costs, which we leave in Appendix C.3.

Results. Table 4 shows results obtained on CART, unconstrained CPTree, and the constrained one. Without the constraint, both CART and CPTree correctly classify only 68.75% of sick women. However, when imposing the constraint, the constrained CPTree does not only increase this percentage on the training distribution (93.75%), but also on the test distribution (88.89%). Nonetheless, in this case, the accuracy of the constrained CPTree is slightly below the one of the unconstrained CPTrees.

5.5. Fairness constraint Applied to Recidivism Prediction

In this application, we use the ProPublicas COMPAS recidivism data. The task is to predict recidivism based on historical crime and demographic features. Studies have reported that the system built around this dataset was racially biased against African American defendants [40]. Inspired by Cotter et al. [7], we impose the constraint that the DDP (with race as the protected feature) should be less than 5%.

Results. Table 5 shows results[6] obtained on CART, unconstrained CPTree, and constrained CPTree. It appears on one hand that, both CART and unconstrained have approximately the same predictive accuracy, but the unconstrained CPTree is less fair than CART according to DDP. On the other hand, when enforcing the fairness constraint DDP decreases by approximately at least 6% (down to $\pm 1\%$) while keeping the same level of accuracy of unconstrained trees.

In summary for these use cases, this section showed that our framework allows enforcing domain-knowledge constraints in diverse real-world applications. For each application, our constrained CPTrees enforce constraints without loss in accuracy performance with respect to the unconstrained ones and CART baselines. In what follows, we show that if no constraints are enforced, CPTrees obtain competitive results with respect to state-of-the-art tree learners. We aim to show that domain experts can safely use our approach to obtain trees that (i) are reliable and (ii) straightforwardly enforce constraints that the models need to comply with.

6. Comparison to State-of-the-Art Decision Tree Learners

This section benchmarks the proposed CPTrees with respect to state-of-the-art decision tree learners in order to validate their relevance from an accuracy perspective. Indeed, Section 5 has shown that CPTrees can enforce domain

[6] Trees are left in Appendix D.4. due to a lack of space.

Dataset name	N	M	Type of features	C
Balance scale	625	4	categorical	3
Bankote authentication	1372	5	numeric	2
Car evaluation	1728	6	categorical	3
Credit approval	690	15	categorical, numeric	2
Hepatitis	155	19	categorical, numeric	2
Ionosphere	351	34	numeric	2
Iris	150	4	numeric	3
Mammographic masses	961	6	categorical, numeric	2
Monk1	432	6	categorical	2
Monk2	432	6	categorical	2
Monk3	432	6	categorical	2
Pima indian diabetes	768	10	numeric	2
QSAR biodegradation	1055	41	numeric	2
Post operative patient	90	8	categorical, numeric	3
Seismic-bumps	2584	19	numeric	2
Spambase	4601	57	numeric	2
Spect heart	267	22	categorical	2
Thoracy surgery	470	17	numeric	2
Tic tac toe	958	9	categorical	2
Wine	178	13	numeric	3

Table 6: Datasets. N, M, C denote respectively the number of instances, features and classes.

knowledge constraints, but they must also provide competitive accuracy to be of practical interest.

6.1. Experimental Settings

Most of the experiment settings that we use in this section have been set in accordance with [9] and [10]. Experiments have been performed on 20 UCI datasets [41] (mostly taken from the list of datasets used by Verwer and Zhang [9] plus additional ones as seen in Table 6). Depending on the datasets, the number of classes varies from 2 to 3 and the number of instances from 90 to 4601.

No code for preprocessing datasets was found on any of the following source code repositories: BinOCT [7][9], Verhaeghe et al. [32][8] , OSDT[9] [29] and DL8.5[10] [27]. However, some datasets that we have used, were found on the BinOCT

[7]https://github.com/SiccoVerwer/binoct

[8]https://bitbucket.org/helene_verhaeghe/classificationtree

[9]https://github.com/xiyanghu/OSDT

[10]https://github.com/aglingael/dl8.5

Dataset	CART	BinOCT[1]	BinOCT[2]	[32]	DL8.5	OSDT	CPTree-1	CPTree-1*	CPTree-2
Balance	61.15	63.06	65.35	N/A	65.35	N/A	65.35	66.50	66.62
Bank. A.	86.30	88.28	91.31	89.21	88.28	87.07	88.28	87.46	90.38
Biodeg	75.53	75.98	77.05	76.52	75.99	76.67	75.76	76.67	69.32
Car	77.13	77.13	77.13	N/A	77.13	N/A	77.13	77.13	72.08
Credit A.	85.37	84.27	84.63	85.37	84.03	85.37	84.27	85.61	74.27
Hepatitis	78.97	83.59	83.08	83.08	83.08	80.51	84.10	86.67	82.05
Ionosphere	78.64	81.82	88.86	81.82	81.82	78.18	82.27	80.45	89.55
Iris	93.68	93.68	90.53	N/A	94.21	N/A	93.68	93.68	92.11
Mam. M.	82.12	82.40	83.85	82.40	82.40	82.21	82.40	82.60	82.12
Monk1	76.83	75.02	75.02	74.55	75.02	74.59	75.97	75.54	75.97
Monk2	65.56	64.30	64.30	60.65	64.58	65.88	65.56	65.56	65.56
Monk3	95.83	95.70	95.70	95.96	95.70	95.70	95.83	95.83	95.83
Pima	74.58	74.58	74.38	74.58	74.58	73.75	74.58	74.58	73.02
Post O.	73.64	73.64	73.64	N/A	70.91	N/A	65.45	66.36	67.27
Seismic	93.44	93.34	93.10	93.03	93.34	93.44	93.34	93.34	93.19
Spambase	77.98	77.65	85.14	77.65	77.65	77.98	77.65	77.98	T/O
Spect H.	76.42	77.51	77.51	71.12	77.50	77.51	76.42	76.42	76.42
Thoracy S.	83.22	83.73	82.37	83.90	83.90	83.90	83.90	83.56	83.05
Tic T. T.	68.92	67.50	67.50	67.33	67.50	68.58	67.50	68.00	68.67
Wine	88.89	91.56	93.33	N/A	91.55	N/A	91.56	89.78	92.00

(a) Average test accuracy with maximum depth 2 over 5 repetitions.

Balance	66.75	68.15	68.79	N/A	69.94	N/A	69.68	69.68	64.33
Bank. A.	86.30	92.54	96.15	94.46	92.54	92.30	92.59	90.50	92.13
Biodeg	76.67	78.41	78.33	82.58	80.08	80.53	79.09	80.30	66.67
Car	78.89	80.0	80.00	N/A	79.82	N/A	79.81	79.81	T/O
Credit	84.39	85.12	85.49	87.20	85.74	85.61	85.98	86.10	76.52
Hepatitis	79.49	81.03	80.51	82.05	81.54	80.51	81.03	81.03	82.56
Ionosphere	81.14	85.45	87.05	89.32	88.64	80.91	86.36	86.82	80.91
Iris	93.16	94.21	92.11	N/A	93.68	N/A	95.26	93.68	96.32
Mammo.	82.98	83.56	83.46	83.56	83.46	83.75	83.37	83.65	80.29
Monk1	80.43	86.27	86.27	86.70	86.27	85.16	85.18	80.86	82.45
Monk2	63.71	58.92	57.59	59.08	59.31	63.89	57.75	61.46	63.44
Monk3	98.42	99.14	99.14	99.28	96.36	98.58	98.99	98.99	97.41
Pima	73.96	72.71	73.65	70.52	70.52	73.75	70.62	73.02	71.48
Post O.	72.73	69.09	69.09	N/A	66.36	N/A	59.09	62.73	61.82
Seismic	93.44	93.28	93.13	93.28	93.19	93.44	93.22	93.19	93.34
Spambase	83.25	83.35	84.36	83.75	83.76	83.84	83.75	83.75	T/O
Spect H.	75.82	76.86	76.86	74.62	77.67	77.51	78.51	79.10	77.91
Thoracy S.	82.54	82.54	81.86	81.86	80.85	83.39	81.53	82.54	83.73
Tic T. T.	72.83	72.00	71.75	73.17	73.33	73.67	73.17	74.25	T/O
Wine	87.56	92.00	90.22	N/A	92.44	N/A	90.22	93.33	94.22

(b) Average test accuracy with maximum depth 3 over 5 repetitions.

Balance	65.48	72.61	71.08	N/A	72.49	N/A	72.61	71.46	61.31
Bank.	92.36	93.94	97.26	95.63	94.58	92.77	94.46	93.76	93.41
Biodeg	77.42	78.79	79.09	81.06	80.38	78.56	78.26	79.17	T/O
Car	79.44	82.59	83.29	N/A	82.82	N/A	82.18	82.36	T/O
Credit A.	85.85	85.49	84.63	84.76	85.25	85.61	84.76	85.73	T/O
Hepatitis	78.97	74.87	82.56	78.46	78.46	76.92	82.56	83.08	82.05
Ionosphere	87.27	85.45	88.18	84.32	84.32	86.36	86.59	85.91	89.77
Iris	93.16	93.68	95.26	N/A	93.68	N/A	95.26	94.21	95.79
Mammo.	82.79	83.17	82.31	82.60	82.60	83.46	81.92	84.13	83.17
Monk1	82.88	100.00	100.00	100.00	100.00	100.00	100.00	100.00	81.58
Monk2	65.03	60.35	59.11	60.59	60.13	64.02	56.82	62.78	63.25
Monk3	98.99	97.47	97.47	96.19	97.28	98.58	98.56	98.99	96.83
Pima	71.88	71.77	71.25	69.27	70.52	72.92	69.38	72.08	74.35
Post O.	65.45	60.91	60.91	N/A	65.45	N/A	56.36	63.64	60.91
Seismic	93.44	93.13	92.94	93.13	93.06	93.44	93.07	93.16	93.30
Spambase	83.79	83.28	83.37	84.40	84.40	81.06	83.54	83.72	T/O
Spect H.	77.31	75.86	75.56	74.37	76.73	77.51	74.93	77.91	75.75
Thoracy S.	81.86	83.22	82.20	80.68	80.17	84.24	81.02	81.86	84.18
Tic T. T.	81.75	78.42	78.92	80.83	81.25	77.92	80.33	77.08	T/O
Wine	92.00	92.89	89.33	N/A	89.33	N/A	88.44	92.89	92.89

(c) Average test accuracy with maximum depth 4 over 5 repetitions.

Table 7: Test accuracy for several maximum depths. N/A means that the method cannot do multiclass classification. T/O means that no solution was found within the time limit (600s).

Dataset	CART	BinOCT[1]	BinOCT[2]	[32]	DL8.5	OSDT	CPTree-1	CPTree-1*	CPTree-2
Balance	65.64	69.87	69.10	N/A	69.10	N/A	69.10	68.72	68.68
Bank.	87.27	88.80	92.93	88.34	88.80	87.77	88.80	87.81	92.30
Biodeg	78.66	79.32	80.43	79.27	79.32	79.24	79.32	79.24	70.47
Car	77.99	77.99	77.99	N/A	77.99	N/A	77.99	77.99	71.40
Credit A.	86.95	87.28	87.53	86.71	87.28	86.71	87.28	87.03	75.01
Hepatitis	82.76	86.21	89.83	86.21	86.21	80.69	86.21	85.17	89.66
Ionosphere	80.99	86.01	91.94	86.01	86.01	83.88	86.01	85.32	90.42
Iris	95.18	95.54	92.86	N/A	95.54	N/A	95.54	95.18	93.93
Mammo.	83.18	84.08	84.66	84.08	84.08	83.47	84.08	83.89	84.12
Monk1	73.91	78.99	78.99	81.62	79.00	78.32	78.13	77.46	78.13
Monk2	65.78	65.88	65.88	76.80	65.88	65.05	65.78	65.78	65.78
Monk3	96.58	96.09	96.09	96.42	96.09	96.09	96.58	96.58	96.58
Pima	76.70	76.70	78.68	76.70	76.70	76.32	76.70	76.70	78.23
Post O.	72.62	75.38	75.38	N/A	80.00	N/A	75.38	73.85	75.38
Seismic	93.42	93.43	93.68	93.45	93.43	93.42	93.43	93.43	93.61
Spambase	78.67	78.99	85.43	78.99	78.99	78.67	78.99	78.67	T/O
Spect H.	80.00	79.35	79.35	85.12	79.35	79.35	80.00	80.00	80.00
Thoracy S.	86.08	86.48	86.82	86.48	86.48	85.85	86.48	86.14	86.42
Tic T. T.	70.75	71.23	71.23	71.23	71.23	70.81	71.23	71.17	69.67
Wine	93.08	93.83	97.29	N/A	93.83	N/A	93.83	93.53	96.99

(a) Average train accuracy with maximum depth 2 over 5 repetitions.

Balance	70.34	75.00	74.87	N/A	75.00	N/A	75.09	73.80	66.32
Bank.	87.27	93.37	97.26	92.61	93.37	93.00	93.37	92.23	93.22
Biodeg	79.87	82.23	83.34	83.06	83.54	82.23	83.16	82.78	68.27
Car	79.63	81.20	81.20	N/A	81.57	N/A	81.57	81.57	T/O
Credit A.	87.28	88.71	88.63	89.16	89.57	86.99	89.57	88.63	78.53
Hepatitis	87.07	91.03	93.28	91.55	91.38	81.38	91.55	88.10	90.86
Ionosphere	85.40	92.02	93.92	93.16	93.08	85.32	92.17	90.57	86.84
Iris	95.54	98.39	99.64	N/A	98.22	N/A	98.39	96.79	98.75
Mammo.	84.50	85.14	85.34	85.31	85.27	83.92	85.31	84.57	78.91
Monk1	77.84	90.71	90.71	91.86	90.72	90.55	90.46	85.28	83.65
Monk2	66.09	69.71	69.76	73.59	69.71	65.63	68.62	67.47	66.71
Monk3	98.41	98.15	98.15	98.22	98.15	97.82	98.89	98.89	98.07
Pima	76.77	77.78	80.69	78.33	78.33	76.32	78.33	77.74	78.78
Post O.	75.38	81.85	81.85	N/A	84.92	N/A	82.15	77.54	77.54
Seismic	93.42	93.47	93.80	93.47	93.47	93.42	93.47	93.47	93.53
Spambase	84.08	83.86	84.61	84.22	84.22	84.22	84.22	84.22	T/O
Spect H.	81.00	82.25	82.25	84.29	82.00	79.35	82.20	82.10	81.90
Thoracy S.	86.76	87.44	87.90	88.12	88.01	86.02	88.12	86.82	86.31
Tic T. T.	75.96	77.30	77.19	78.80	78.50	76.77	78.77	76.96	T/O
Wine	95.19	97.89	99.85	N/A	97.89	N/A	97.89	96.39	99.40

(b) Average train accuracy with maximum depth 3 over 5 repetitions.

Balance	72.01	78.21	77.95	N/A	79.06	N/A	78.93	76.79	61.75
Bank.	93.26	94.52	98.10	94.27	94.83	93.59	94.83	94.36	94.50
Biodeg	83.08	83.84	83.69	86.60	86.78	81.95	82.76	83.19	T/O
Car	80.19	83.63	83.53	N/A	84.55	N/A	83.89	83.80	T/O
Credit A.	89.98	89.61	89.08	91.41	91.74	88.02	90.67	89.53	T/O
Hepatitis	90.00	94.83	97.41	97.76	97.76	84.48	97.59	90.00	91.38
Ionosphere	90.11	94.52	95.59	97.26	97.26	89.05	95.59	92.02	91.33
Iris	97.50	98.39	100.00	N/A	98.39	N/A	98.39	97.68	97.68
Mammo.	85.14	85.92	86.11	86.43	86.40	84.21	86.21	85.24	81.94
Monk1	83.69	100.00	100.00	100.00	100.00	100.00	100.00	100.00	81.97
Monk2	68.53	73.93	73.83	77.78	74.51	69.40	72.22	70.13	66.89
Monk3	98.89	98.48	98.48	98.92	98.81	97.82	98.94	98.89	96.92
Pima	78.44	79.79	80.87	81.11	80.97	77.22	79.90	79.06	77.82
Post O.	77.85	86.46	86.77	N/A	91.69	N/A	90.46	79.08	78.77
Seismic	93.42	93.52	93.89	93.57	93.57	93.42	93.56	93.51	93.50
Spambase	84.60	84.84	84.01	85.50	85.50	81.88	84.57	84.52	T/O
Spect H.	82.90	85.95	85.95	88.54	86.95	79.35	86.90	84.60	82.25
Thoracy S.	88.01	88.81	88.86	90.28	90.17	85.51	89.77	88.12	86.08
Tic T. T.	83.62	83.82	83.87	87.05	86.69	81.78	84.43	80.84	T/O
Wine	98.80	99.10	100.00	N/A	100.00	N/A	100.00	97.44	98.80

(c) Average train accuracy with maximum depth 4 over 5 repetitions.

Table 8: Train accuracy obtained for several maximum depths. N/A means that the method cannot do multiclass classification. T/O means that no solution was found within the time limit (600s).

repository[7], were already preprocessed. In order to extend preproprecessing for other datasets, we had to preprocess the datasets, which may explain little differences with published performances (e.g., slightly different accuracy) on a few datasets. Additionally, the small differences with published performances may also be explained by the division, which does not match exactly the one done in the papers. For methods that require binary features (DL8.5, OSDT, Verhaeghe et al. [32] and our first model), datasets are preprocessed by transforming numeric features into (3 bins using the quantile discretiser from Scikit-learn[11]) categorical ones and then transforming all categorical features into binary features through one-hot-encoding.

Datasets have been divided into 3 sets: training (50%), validation (25%), and testing (25%). BinOCT, DL8.5 and Verhaeghe et al. [32] do not have any additional hyperparameters nor the number of leaves to tune. We ran these models directly on training plus validation sets. Since BinOCT has an intrinsic heuristic to binarise numeric features, we keep datasets with their numeric features for experiments with BinOCT. This gives two versions of BinOCT (BinOCT^1 for only binary features and BinOCT^2 with numeric features). Other methods only work with binary features according to their released code. In the experiments, we also include CART [16] from Scikit-learn using the entropy as a heuristic.

OSDT and CPTree require cross-validation to select hyperparameters λ and L, respectively. Therefore, λ has been selected after validation according to the default range of values provided by Hu et al. [29]. The number of leaves L of our model CPTrees has also been validated considering values from $3(K-1)$ to 2^K, where K is the maximum depth. Once tuning is done, we ran OSDT and our model on training plus validation sets, according to the best λ and L, respectively. Since our model is more general than [9], we made three versions of CPTrees. CPTree-1 and CPTree-1* come from the first formulation and are respectively the model for complete tree structures ($L = 2^K$ as BinOCT) and the model for which the number of leaves has been validated. Second, CPTree-2

[11] We used KBinsDiscretizer from the preprocessing package.

is the third model related to the second formulation (i.e., with numeric features) detailed in Appendix A.1.

To be fair, all models ran without a warm start such as CART. No additional constraint has been added since the goal of this section is to assess the ability of our method to produce decision trees that achieve similar accuracy to those obtained by state-of-the-art methods described in Section 2.

Since our first formulation uses linear constraints and binary variables, it can be implemented into CP or MIP solvers. We used the CP-SAT solver of the Google OR-Tools library [48], which is freely available in Python. For our second formulation, we used the MIP solver of Gurobi [49].

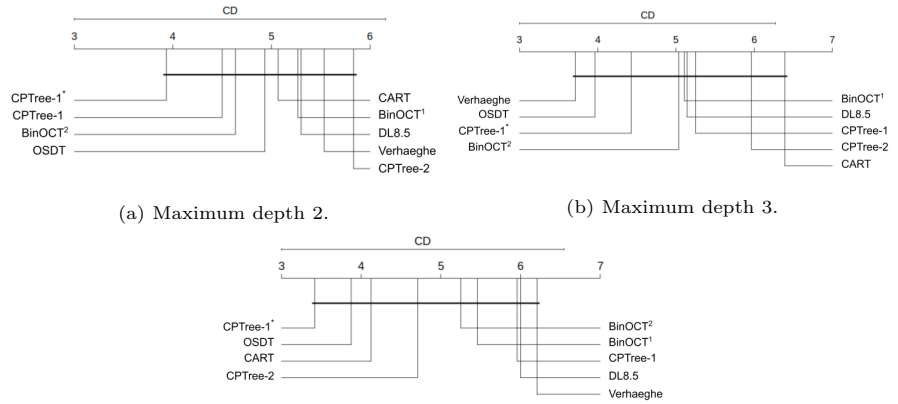
Based on the work of BinOCT [9], we set to 10 minutes the time limit for each run of all the methods. Experiments have been conducted on 5 independent runs by dataset, by depth (the maximal depth was set to 2, 3, and 4) also according to [9]. We ran them sequentially (to be fair with all methods) on a server with a Common KVM processor (2.294 GHz) and 16GB of RAM. Running completely all the evaluations took more than 2 weeks[12] of extensive computations.

The code for CPTrees is available[13] and learned trees can be inspected and visualised as in Scikit-learn. All scripts that we used to benchmark all these recent models are also available for future reproducibility.

6.2. How do CPTrees Perform Comparatively to BinOCT, OSDT and DL8.5?

Our benchmarking procedure aims to evaluate the generalisation of optimal decision tree models rather than how close they are to optimal solutions. Therefore, Table 7a, 7b and 7c present the average test accuracy (on depth 2, 3 and 4, respectively) over 5 independent runs as detailed in Section 6.1. Similarly, Table 8 shows the average training accuracy. Here, the training accuracy of optimal methods differs because, as a reminder, in accordance with Verwer and Zhang [9], all runs were done within the time limit of 10 minutes. Figure 7a, Figure 7b

[12] The total number of runs is $\approx 20(\text{datasets}) \times 3(\text{depths}) \times 8(\text{models}) \times 5(\text{runs}) = 2400$.
[13] https://github.com/gerald4/CPTree



(a) Maximum depth 2.

(b) Maximum depth 3.

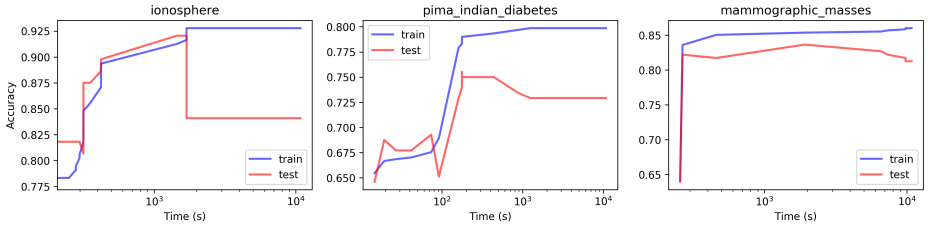
(c) Maximum depth 4.

Figure 7: Nemenyi statistical significance test. From left to right, algorithms are ranked from best to worst. The bold horizontal line indicates no significant difference between algorithms.

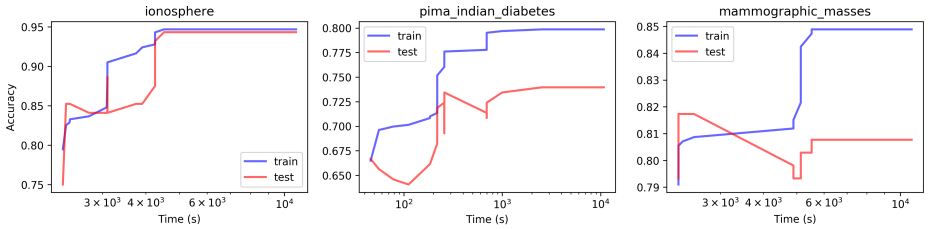
and Figure 7c show results of the Nemenyi statistical significant test, which is a non-parametric test that compares algorithms pairwise of their performance. Figure 7 shows that none of the compared methods outperforms the others, in terms of generalisation, according to the Nemenyi test. This is confirmed by the analysis of Table 7, which shows predictive performance. In terms of test accuracy, Table 7 shows that OSDT and CPTree-1* are usually close to each other, which is not surprising since they both are branch-like models. In terms of train accuracy, Table 8 shows that CPTrees are similar to state-of-the-art learners that are designed to find optimal decision trees.

It is also worth noting that, CPTree-2 is the only model (with CART) in Table 7 and Table 8, which does not involve preprocessing or heuristic discretisation of features. It is usually slow to reach an optimal solution, in particular for datasets with categorical features. This is due to the existence of multiple choices of splits that give the same semantic explanation. However, for pure numeric features (e.g., *Bank.*, *Ionosphere*, *Iris*, *Pima*, etc.), it usually provides good results, making it especially suitable for enforcing domain constraints in cases where datasets present numeric features.

Overall, CPTree-1, CPTree-1*, and CPTree-2 generally have similar perfor-



(a) Maximum depth 3.



(b) Maximum depth 4.

Figure 8: Accuracy curve within 3 hours of optimisation.

mances. They perform sometimes better (or worse) than state-of-the-art methods, but overall, there is no significant difference, according to the Nemenyi statistical tests. However, the particularity of CPTrees is to be as flexible as possible to incorporate a broad class of domain constraints as in Section 5.

7. Computational Time

This section examines how much computational time does CPTree requires to find a good decision tree that generalises well. We ran additional experiments on three datasets with maximum depths ($K = 3, 4; L = 2 * K$) and a time limit of 3 hours to explore more feasible space. We implemented a *callback* to keep track of both the training and testing accuracy during optimisation.

Figure 8 shows results obtained along optimisation path. At the first glance, it appears that predictive accuracy saturates and sometimes decreases after a reasonable amount of time. This confirms that we can avoid increasing the time limit. More specifically, at depth 3, Figure 8a shows that for *Pima Indian diabetes* and *Mammographic masses*, the time required to learn a good CPTree is less than $10^3 s$. This time is slightly higher for *Ionosphere*. On the other

hand, at depth 4, from Figure 8b, this time is less than or around $6 \times 10^3 s$. For *Ionosphere* (maximum depth 3) and *Mammographic masses* (maximum depth 4), the occasional drop in predictive accuracy is due to the oversearching problem, a well-known issue for optimal searching methods [50]. This question needs further investigation, especially on tree learners and we leave it for future work.

In brief, our finding from this optimisation time/path analysis is that it is not necessary to reach optimality with CPTree since this may harm predictive performance without any strategies to counter overfitting from oversearching.

8. Conclusion

This paper introduces a tree representation that leads to two new formulations to enforce domain-knowledge constraints on decision trees. With these formulations, we are able to enforce a broader family of constraints compared to recently proposed methods. These constraints include but are not limited to the number of leaf nodes, the maximum depth, domain-knowledge constraints like the ordering of features on a branch of the tree or costs on features, or even regarding fairness. These formulations provide a flexible framework in which several constraints both regarding the complexity and domain knowledge can be easily formulated seeking to learn more interpretable and trustworthy trees. The learned CPTrees ensure that the constraints are satisfied while keeping the same level of accuracy with baselines. However, users should also make sure, when adding constraints, that the optimisation problem has feasible solutions. Otherwise, the problem may not have a feasible solution simply because it is overconstrained. Finally, future work includes more experiments to validate interpretability's improvement directly with users or domain experts.

Acknowledgement. This work has been funded by the EOS-VeriLearn, project number 30992574 of the Fonds de la Recherche Scientifique (F.R.S-FNRS) in Belgium. The authors also thank the medical Doctor Drem's Tailor Fomekong for domain knowledge, Prof. Hendrik Blockeel, Prof. Sebastijan Dumancic, and Kshitij Goyal for their useful comments that helped to improve the framework.

References

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

[2] H. K. Sok, M. P.-L. Ooi, Y. C. Kuang, S. Demidenko, Multivariate alternating decision trees, Pattern Recognition 50 (2016) 195–209.

[3] L. Ma, S. Destercke, Y. Wang, Online active learning of decision trees with evidential data, Pattern Recognition 52 (2016) 33–45.

[4] J. Y. Verbakel, M. B. Lemiengre, T. De Burghgraeve, A. De Sutter, B. Aertgeerts, D. M. Bullens, B. Shinkins, A. Van den Bruel, F. Buntinx, Validating a decision tree for serious infection: diagnostic accuracy in acutely ill children in ambulatory care, BMJ open 5 (2015).

[5] S. Y. Sohn, J. W. Kim, Decision tree-based technology credit scoring for start-up firms: Korean case, Expert Systems with Applications 39 (2012) 4007–4012.

[6] L. Floridi, Establishing the rules for building trustworthy ai, Nature Machine Intelligence 1 (2019) 261–262.

[7] A. Cotter, H. Jiang, M. Gupta, S. Wang, T. Narayan, S. You, K. Sridharan, Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals, Journal of Machine Learning Research 20 (2019) 1–59.

[8] D. Martens, J. Vanthienen, W. Verbeke, B. Baesens, Performance of classification models from a user perspective, Decision Support Systems 51 (2011) 782–793.

[9] S. Verwer, Y. Zhang, Learning optimal classification trees using a binary linear program formulation, in: 33rd AAAI Conference on Artificial Intelligence, 2019.

[10] D. Bertsimas, J. Dunn, Optimal classification trees, Mach. Learn. 106 (2017) 1039–1082.

[11] N. Narodytska, A. Ignatiev, F. Pereira, J. Marques-Silva, Learning optimal decision trees with sat, in: 27th International Joint Conference on Artificial Intelligence, 2018.

[12] S. Aghaei, M. J. Azizi, P. Vayanos, Learning optimal and fair decision trees for non-discriminative decision-making, in: 33rd AAAI, 2019.

[13] G. K. Dziugaite, S. Ben-David, D. M. Roy, Enforcing interpretability and its statistical impacts: Trade-offs between accuracy and interpretability, ArXiv:2010.13764 (2020).

[14] M. T. Ribeiro, S. Singh, C. Guestrin, Model-agnostic interpretability of machine learning, in: ICML Workshop on Human Interpretability in Machine Learning, 2016.

[15] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82 – 115.

[16] L. Breiman, J. Friedman, C. J. Stone, R. Olshen, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984.

[17] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann publishers Inc., San Francisco, CA, USA, 1993.

[18] J. R. Quinlan, R. L. Rivest, Inferring decision trees using the minimum description length principle, Information and Computation 80 (1989) 227–248.

[19] M. Garofalakis, D. Hyun, R. Rastogi, K. Shim, Efficient algorithms for constructing decision trees with constraints, in: 6th KDD, 2000.

[20] P. Niyogi, F. Girosi, T. Poggio, Incorporating prior information in machine learning by creating virtual examples, Proceedings of the IEEE 86 (1998) 2196–2209.

[21] M. Núñez, The use of background knowledge in decision tree induction, Mach. Learn. 6 (1991) 231–250.

[22] J. A. López-Vallverdú, D. Riaño, A. Collado, Increasing acceptability of decision trees with domain attributes partial orders, in: 20th IEEE CBMS, 2007.

[23] J. A. López-Vallverdú, D. Riaño, J. A. Bohada, Improving medical decision trees by combining relevant health-care criteria, Expert Syst. Appl. 39 (2012) 11782–11791.

[24] S. Nijssen, E. Fromont, Optimal constraint-based decision tree induction from itemset lattices, Data Mining and Knowledge Discovevery 21 (2010) 9–51.

[25] M. Garofalakis, D. Hyun, R. Rastogi, K. Shim, Building decision trees with constraints, Data Mining and Knowledge Discovery 7 (2003) 187–214.

[26] S. Nijssen, E. Fromont, Mining optimal decision trees from itemset lattices, in: 13th SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.

[27] G. Aglin, S. Nijssen, P. Schaus, Learning optimal decision trees using caching branch-and-bound search, in: 34th AAAI Conference on Artificial Intelligence, 2020.

[28] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, C. Rudin, Learning certifiably optimal rule lists for categorical data, Journal of Machine Learning Research 18 (2018) 1–78.

[29] X. Hu, C. Rudin, M. Seltzer, Optimal sparse decision trees, in: Neurips, 2019.

[30] J. Lin, C. Zhong, D. Hu, C. Rudin, M. Seltzer, Generalized and scalable optimal sparse decision trees, in: 37th International Conference on Machine Learning, 2020.

[31] A. Florent, Efficient inference of optimal decision trees, in: 34th AAAI, 2020.

[32] H. Verhaeghe, S. Nijssen, G. Pesant, C.-G. Quimper, P. Schaus, Learning optimal decision trees using constraint programming, in: 25th CP, 2019.

[33] H. Liu, F. Hussain, C. L. Tan, M. Dash, Discretization: An enabling technique, Data mining and knowledge discovery 6 (2002) 393–423.

[34] I. N. M. Shaharanee, F. Hadzic, T. S. Dillon, Interestingness measures for association rules based on statistical validity, Knowledge-Based Systems 24 (2011) 386–392.

[35] J. M. Hodgson, V. H. van Someren, C. Smith, A. Goyale, Direct bilirubin levels observed in prolonged neonatal jaundice: a retrospective cohort study, BMJ Paediatr. open 2 (2018).

[36] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al., Constrained k-means clustering with background knowledge, in: 18th ICML, 2001.

[37] T. Calders, S. Verwer, Three naive bayes approaches for discrimination-free classification, Data mining and knowledge discovery 21 (2010) 277–292.

[38] T. Calders, F. Kamiran, M. Pechenizkiy, Building classifiers with independency constraints, in: ICDM Workshop on Domain Driven Data Mining, 2009.

[39] M. Lohaus, M. Perrot, U. Von Luxburg, Too relaxed to be fair, in: 37th ICML, 2020.

[40] W. Yang, L. Lorch, M. Graule, H. Lakkaraju, F. Doshi-Velez, Incorporating interpretable output constraints in bayesian neural networks, in: Neurips, 2020.

[41] D. Dua, C. Graff, UCI mach. learn. repo., 2017. URL: `http://archive.ics.uci.edu/ml`.

[42] M. Tina Binesh, M. Sydney, Toronto Notes for Medical Students: comprehensive medical reference and review for MCCQE and USMLE II, 34th ed ed., Toronto Notes 2018, 2018.

[43] R. Li, R. Daniel, B. Rachet, How much do tumor stage and treatment explain socioeconomic inequalities in breast cancer survival? applying causal mediation analysis to population-based data, European journal of epidemiology 31 (2016) 603–611.

[44] J. W. Smith, J. E. Everhart, W. Dickson, W. C. Knowler, R. S. Johannes, Using the adap learning algorithm to forecast the onset of diabetes mellitus, in: annual symposium on computer application in medical care, American Medical Informatics Association, 1988.

[45] S. H. Read, L. C. Rosella, H. Berger, D. S. Feig, K. Fleming, P. Kaul, J. G. Ray, B. R. Shah, L. L. Lipscombe, Diabetes after pregnancy: a study protocol for the derivation and validation of a risk prediction model for 5-year risk of diabetes following pregnancy, Diagnostic and Prognostic Research 5 (2021) 1–8.

[46] N. K. Wenger, L. Speroff, B. Packard, Cardiovascular health and disease in women, New England Journal of Medicine 329 (1993) 247–256.

[47] V. Okunrintemi, J. Valero-Elizondo, B. Patrick, J. Salami, M. Tibuakuu, S. Ahmad, O. Ogunmoroti, S. Mahajan, S. U. Khan, M. Gulati, et al., Gender differences in patient-reported outcomes among adults with atherosclerotic cardiovascular disease, Journal of the American Heart Association 7 (2018) e010498.

[48] L. Perron, V. Furnon, Google: Or-tools, `https://developers.google.com/optimization/`, 2019.

[49] G. Optimization, Gurobi optimizer reference manual, `http://www.gurobi.com`, 2021.

[50] J. Quinlan, R. Cameron-Jones, Oversearching and layered search in empirical learning, in: 14th International Joint Conference on Artificial Intelligence 95, 1995.