Constraint Enforcement on Decision Trees: A Survey

GÉRALDIN NANFACK, PAUL TEMPLE, and BENOÎT FRÉNAY, University of Namur, Belgium

Decision trees have the particularity of being machine learning models that are visually easy to interpret and understand. Therefore, they are primarily suited for sensitive domains like medical diagnosis, where decisions need to be explainable. However, if used on complex problems, then decision trees can become large, making them hard to grasp. In addition to this aspect, when learning decision trees, it may be necessary to consider a broader class of constraints, such as the fact that two variables should not be used in a single branch of the tree. This motivates the need to enforce constraints in learning algorithms of decision trees. We propose a survey of works that attempted to solve the problem of learning decision trees under constraints. Our contributions are fourfold. First, to the best of our knowledge, this is the first survey that deals with constraints on decision trees. Second, we define a flexible taxonomy of constraints applied to decision trees and methods for their treatment in the literature. Third, we benchmark state-of-the art depth-constrained decision tree learners with respect to predictive accuracy and computational time. Fourth, we discuss potential future research directions that would be of interest for researchers who wish to conduct research in this field.

CCS Concepts: • **Computing methodologies** → **Classification and regression trees**; **Learning settings**; *Search methodologies*; • **General and reference** → **Surveys and overviews;**

Additional Key Words and Phrases: Decision trees, constraints, interpretability, explainability, domain knowledge, privacy, fairness

ACM Reference format:
Géraldin Nanfack, Paul Temple, and Benoît Frénay. 2022. Constraint Enforcement on Decision Trees: A Survey. *ACM Comput. Surv.* 54, 10s, Article 201 (September 2022), 36 pages.
https://doi.org/10.1145/3506734

1 INTRODUCTION

Integrating constraints in learning algorithms has shown its importance in machine learning. Real-world applications indeed require models to comply with specified constraints and guarantees. This is even more becoming a major issue, as machine learning models are expected to be interpretable and trustworthy.[1] Regularisation of objective functions and prior beliefs

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

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.
Authors' address: G. Nanfack, P. Temple, and B. Frénay, University of Namur, NADI Institute, Faculty of Computer Science, PReCISE Research Center, 21 Rue Grandgagnage, Namur, 5000, Belgium; emails: {geraldin.nanfack, paul.temple, benoit.frenay}@unamur.be.

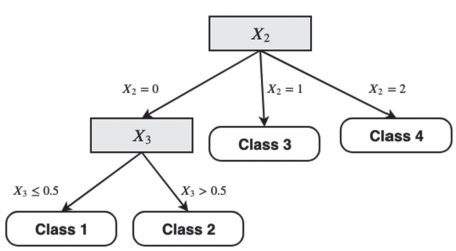
0360-0300/2022/09-ART201 $15.00
https://doi.org/10.1145/3506734

[WESEL et al. 2011] in Bayesian settings are examples of means to enforce constraints on a machine learning model. However, when constraints from diverse nature must be enforced, incorporating them into learning algorithms remains an open issue. For example, imposing the sparsity of a machine learning model for explainability is different from guaranteeing fair decisions or from ensuring privacy for ethical purposes. Currently, mainstream algorithms often only aim at providing accurate predictions and they are applied in ever more areas. Yet, they usually lack the ability to meet user and domain knowledge constraints, which are left aside [Gilpin et al. 2018; Lorenzi and Filippone 2018].

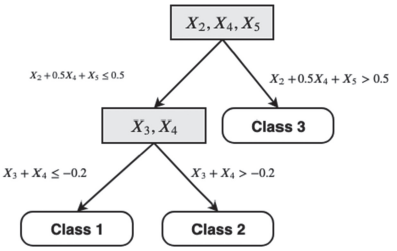
Decision trees are one of the most well-known and simplest machine learning algorithms. Their representability and their ability to produce rules with relevant attributes make them the most commonly used technique when seeking interpretable machine learning models [Freitas 2014]. Despite their simplicity, standard decision tree algorithms such as ID3 [Quinlan 1986], C4.5 [Quinlan 1993], and CART [Breiman et al. 1984] may produce trees that are very difficult to understand, because they may be large and complex. Moreover, learned trees may not satisfy some desirable constraints such as being small and accurate at the same time. Pruning methods can be applied to reduce the complexity of trees and to avoid overfitting but the resulting trees can be less accurate or can still fail to satisfy properties such as fairness or privacy desired by domain experts. If decision trees fail to satisfy these requirements, then they will likely be rejected. For example, in some critical domains such as medicine, guarantees are indeed needed to validate a particular machine learning model (e.g., in terms of prediction performances) and doctors may need to know and understand the logic behind the predictions to trust and use the model [Bibal and Frénay 2016; Martens et al. 2011]. As long as domain experts are involved in the validation loop of decision tree models, these trees will have to be neither too large nor too deep to be "humanly processable." Therefore, in this context, imposing constraints on decision trees is required to guarantee various aspects, which include interpretability, fairness, or being reliable and trustworthy. For instance, let us suppose that a person is willing to loan money and goes to see a bank advisor. The bank advisor will get pieces of information to establish a profile by asking, for example, their age first before asking whether they earn money (and probably how much) followed by other questions. The order and the type of questions are important: first to make people feel comfortable in exchanging information that may be considered personal; second, to ensure ethical safeguards such as making decisions regardless of gender, ethnicity, or religious beliefs of people. Supposing that a decision tree takes the loan decision, it must encode these constraints too to mimic the behaviour of bank advisors and ultimately be accepted by them. As an illustration, works such as López-Vallverdú et al. [2012, 2007]; Núñez [1991] enforce constraints on decision trees to make them more acceptable and comprehensible for domain experts.

What are the constraints that are applied to decision trees in the literature? What are the methods used to constrain a decision tree to comply with the desired properties? How can constraint enforcement serve to learn bias-free, interpretable, and trustworthy decision trees? Guided by these questions, this article presents a survey on how the literature attempts to solve the problem of learning optimal decision trees that satisfy a set of given constraints. Our contributions are summarised as follows: (1) to the best of our knowledge, we present the first survey on methods to enforce constraints on decision trees; (2) we present a categorisation of constraints used and a taxonomy of methods to enforce them; (3) we benchmark state-of-the-art depth-constrained decision tree learners with respect to predictive accuracy and computational time; (4) we point out and discuss open problems that we think are important to address and may help researchers who want to work in this domain.

Aside from this work, it is important to note that Barros et al. [2012], Buhrman and De Wolf [2002], Lomax and Vadera [2013], and Safavian and Landgrebe [1991] review different decision



(a) A univariate decision tree. The categorical variable X_2 (possible values are in $\{0, 1, 2\}$) has been selected on the root node while the real-valued variable X_3 has been selected on the only internal node of the tree.



(b) A multivariate decision tree. The real-valued variables X_2, X_4 and X_5 have been selected on the root node while X_3 and X_4 have been selected on the only internal node of the tree.

Fig. 1. Examples of univariate and multivariate decision trees. Nodes with a rectangular shape are the root node and the internal nodes. Nodes with a rounded rectangular shape are leaf nodes.

tree learning algorithms. However, they lack special attention to methods that learn trees with constraint enforcement, which is precisely the focus of this work. It is not the scope of our work to present all existing tree learning methods but to review how the literature integrates constraints in the learning process to get more trustworthy, understandable, robust decision trees.

The rest of the work is organised as follows: Section 2 introduces decision trees and motivates the importance to integrate constraints in the learning algorithm of the decision trees. Section 3 gives a taxonomy of the constraints and analyses previous works that try to enforce constraints on decision trees. Section 4 provides a general overview of methods used in the literature. Section 5 discusses potentially available libraries to learn constrained decision trees. Section 6 proposes a discussion on our view about key issues in the field, benchmarks state-of-the-art depth-constrained decision tree learners. Furthermore, we highlight future research directions, and more importantly, how can we get benefit from constraint enforcement on decision trees to learn more trustworthy models. Section 7 concludes the work.

2 DEFINITIONS AND IMPORTANCE OF CONSTRAINT ENFORCEMENT

This section presents decision trees and discusses the benefits of constraint enforcement on them.

2.1 Decision Trees in a Nutshell

This section is inspired by the work of Safavian and Landgrebe [1991] describing the graph formalism of a decision tree. A decision tree is a family of machine learning algorithms primarily designed for classification and regression, although they have also been extended to clustering [Blockeel et al. 1998]. Classification can be described as the task of mapping instances $\mathbf{x_i}$ of a dataset $\mathcal{D} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{N}$ to label y_i from a set of predefined label $\{1, \ldots, C\}$ [Murphy 2012]. Regression only differs from the previous description in the definition domain of y_i, which is $\mathbb{R}$. Decision trees try to learn the mapping between the different $\mathbf{x_i}$ of $\mathcal{D}$ and their expected output y_i. $\mathbf{x_i}$ are observations of M random variables $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_M}$ and are usually of the form $\mathbf{x_i} = (x_i^1, x_i^2, \ldots, x_i^M)$. These random variables can be categorical, integers, or real-valued.

A decision tree is encoded by a directed rooted tree structure $(\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of nodes that contains the root node r, the set of internal nodes $\mathcal{V}_{\text{int}}$ that must have at least two child nodes, and the non-empty set of leaf nodes $\mathcal{V}_{\text{leaf}}$. $\mathcal{E}$ represents the set of edges between nodes defining the hierarchy of the tree. A tree is said to be *binary* (e.g.,Figure 1(b)) if each internal node v_{int} and the root node r have exactly two child nodes. The depth K of a decision tree (also called the height

of the tree) is the number of edges of the *longest* path from the root node r to any of leaf nodes $v_{\text{leaf}} \in \mathcal{V}_{\text{leaf}}$. The size of the tree is the number of nodes $|\mathcal{V}|$ and, finally, we denote the number of leaf nodes as L. With respect to the dataset $\mathcal{D}$, each node $v \in \{r\} \cup V_{int}$ contains a list of attributes that are selected for a splitting criterion when traversing this node. Each edge is labelled by a splitting rule produced by the learning algorithm.

Splitting rules can involve one variable. In this case, the decision tree is said to be univariate. When several variables are used at the same time on splitting rules, the decision tree is multivariate. Figures 1(a) and 1(b), respectively, show an example of a univariate and a multivariate decision tree with splitting rules visible over the edges (e.g., the value of X_3 is larger than 0.5).

Given an instance $\mathbf{x} \in \mathcal{D}$, its prediction of the tree is computed as follows: starting from the root node, a test on one or several variables of $\mathbf{x}$ (e.g., $\mathbf{X}_2$ in Figure 1(a) or $\mathbf{X}_2$, $\mathbf{X}_4$, and $\mathbf{X}_5$ in Figure 1(b)) is performed. Depending on the result of the test, one branch is followed to the next node. A new test is performed and so on until a leaf node is reached. Each leaf node is associated with an output (e.g., "Class 1" for the leftmost leaf node of the tree in Figure 1(a)).

Learning a decision tree is equivalent to learning its structure and its decision rule function (the mapping from the domain of $\mathbf{x}$ to the domain of y), which is composed of splitting rules. It is possible to constrain the structure while learning decision rules of the tree. The search space of decision trees (space of possible decision trees for the dataset) depends therefore on both the possible structures and decision rules. By examining the complexity of this search space, Das and Goodrich [1997] (respectively, Hyafil and Rivest [1976]) show that finding the optimal decision tree that minimises the classification error (respectively, the size of tree) is NP-complete. However, algorithms exist to quickly compute a sub-optimal solution through heuristics. One popular method is the top-down induction approach [Quinlan 1986]. Top-down approaches learn trees by recursively creating a split node (starting with the root node) along with the chosen attribute that optimises a local heuristic. The splitting procedure stops when a specific criterion is reached (for example, the fact that a node is pure, meaning that all instances of a node have the same label) and a leaf node is created. Thus, traditional top-down approaches create trees by applying a depth-first search strategy with a greedy approach, since the tree is built level-by-level.

Several algorithms exist, such as ID3 [Quinlan 1986], C4.5 [Quinlan 1993], and CART [Breiman et al. 1984]. All of them use impurity measures from information theory to evaluate the homogeneity of the empirical class distribution when considering a splitting rule. By using this heuristic, learned greedy trees can be very accurate. However, they might also become large and lose their generalisation capability to predict correctly new instances (also known as overfitting). To avoid that, pruning techniques [Barros et al. 2015] can be applied to find a tradeoff between reducing the complexity of the tree and maintaining a certain level of accuracy.

2.2 On the Importance of Constraint Enforcement on Decision Trees

This section presents some challenges that motivate the enforcement of constraints on decision trees.

Constraints to Improve Interpretability. According to the previous description of traditional algorithms (top-down induction and pruning; see Section 2.1), learned trees can be large and deep. As a result, they may lose their interpretability. Piltaver et al. [2016] concluded that the size of the tree, the depth, and the number of leaf nodes directly impact the comprehensibility of decision trees. Thus, if it is possible to constrain a decision tree to be small, shallow (less deep) while keeping a good level of accuracy, the learned tree would be more interpretable. The work of Bessiere et al. [2009] showed that constraining the size of the decision tree can significantly improve the accuracy of traditional algorithms while halving the size of the tree.

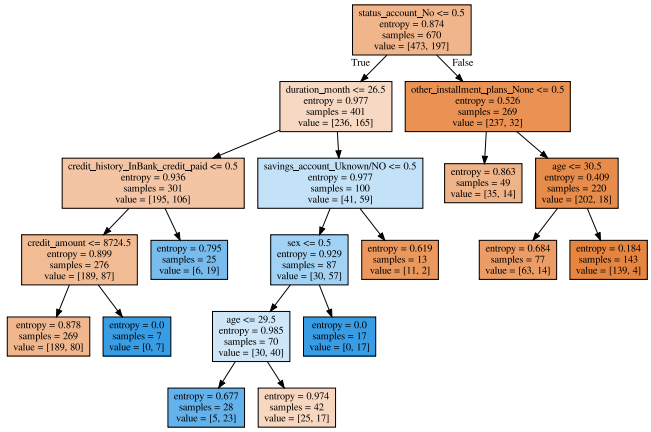


Fig. 2. Example of a decision tree trained on the German credit dataset. Here, split rules are visible inside (root) internal nodes.

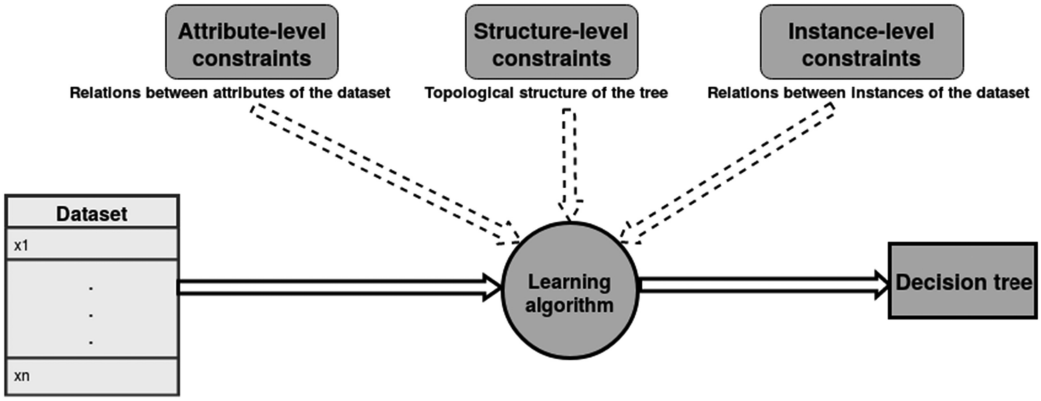


Fig. 3. Learning decision trees under constraints. The dashed arrows indicate optional constraints, while solid line arrows indicate mandatory concepts. For example, a learning algorithm may integrate attribute-level constraints to learn a decision tree.

Constraints for Explainability and Trustworthiness. In the field of health or banking, decision trees are widely used, since people need to understand how the algorithm has reached its decisions. In the medical domain, the expert (i.e., the doctor) validates a particular machine learning model by comparing it with his knowledge about the domain. Here, the doctor examines the sequence of decision rules to evaluate the comprehensibility of the decision tree. However, learned decision trees do not necessarily make sense from a medical or clinical point of view, because the algorithms only consider information that can be extracted from a medical dataset [López-Vallverdú et al. 2007]. Thus, if the rules of a decision tree do not match the needs of experts and their knowledge, then the entire tree will be rejected. However, if decision trees are learned by considering additional domain knowledge in the form of constraints, then the resulting tree could be more trustworthy and reliable. For example, works like López-Vallverdú et al. [2012, 2007] enforce constraints on decision trees by adding priority of relevance on attributes to make trees more comprehensible and trustworthy for users and domain experts.

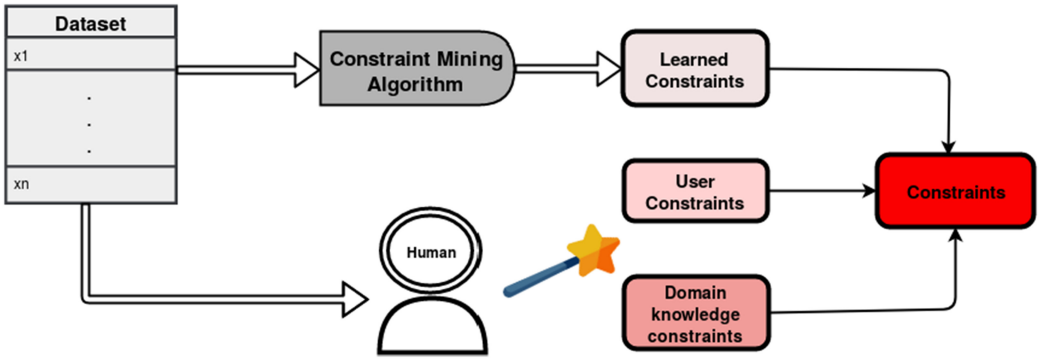


Fig. 4. Sources of constraints. Three different sources are shown: algorithms can discover and learn constraints from a given dataset; humans can define user constraints to the learning algorithm specific to their needs, for instance, to control the complexity of a tree; humans can also give constraints based on their knowledge regarding a dataset within a particular domain.

Constraints for Ethical Safeguards. Recently, because of the rise of machine learning models in society, research has been conducted to promote ethical guarantees on machine learning models. In fact, when applied in the real world, a machine learning model must ensure fair and equitable decisions (for instance, between men and women), as well as the protection of sensitive data information (called privacy). From the ethical point of view, using a black-box machine learning model may be unacceptable, because its decisions cannot be explained. Thus, decision trees are mainly suited for domains like justice to check easily and prevent illegal decisions [Ribeiro et al. 2016a]. Figure 2 gives an example of decision tree trained on the well-known *German credit* dataset. While sometimes it may make unfair predictions between women and men to grant a credit (due to the selected feature *sex*), such a decision tree could be learned and used in practice if a constraint on fairness is not enforced. However, machine learning models must meet ethical requirements such as fairness or even privacy and, most importantly, people must be able to assess that the models actually meet these requirements. Thus, if decision trees are learned with fairness and privacy constraints, they will be more susceptible to be accepted in a critical domain such as justice where the satisfaction of constraints may be more important than providing a good level of accuracy [Barredo Arrieta et al. 2020; Dziugaite et al. 2020; Ribeiro et al. 2016a]. Several works [Aghaei et al. 2019; Kamiran et al. 2010; Liu et al. 2009] attempted to impose fairness and privacy constraints on decision trees.

Constraints on Proxy Models. Similarly to the problem of understanding reasons why the decisions have been taken or to ensure some guarantees (such as fairness or being ethically correct), decision trees can also be used to approximate black-box models (e.g., neural networks) such that predictions can be explained [Nanfack et al. 2021b]. This kind of explanatory model is called a proxy model [Guidotti et al. 2018]. Without any constraints, decision trees may fail at providing clear and understandable explanations of the target black-box model. Indeed, the structure of decision trees (described in Section 2.1) might be too simple to approximate with enough accuracy black-box models that are more complex (such as neural networks) or might be too complex, resulting in trees that have lost their interpretability. Even worse, if the black-box model embeds guarantees such as fairness, without stating them to the decision tree algorithm, then the learned tree has little chance to meet these constraints. For example, a fair neural network could be approximated by an unfair decision tree simply because the approximation algorithm did not incorporate the non-discrimination or fairness constraints.

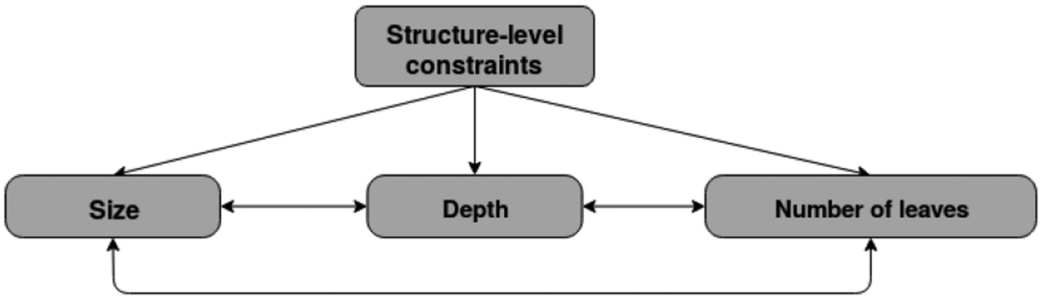


Fig. 5. Structure-level constraints in decision trees. Structure-level constraints are refined into three subcategories of constraints that have an impact of the structure of trees. The left and right horizontal arrows mean that the constraints have a mutual impact. For example, constraining the size of the tree limits its depth and reciprocally.

3 TAXONOMY OF CONSTRAINTS

The previous section introduced decision trees and presented the importance to enforce constraints on them. This section presents our taxonomy of constraints inspired by Nijssen and Fromont [2010] and Struyf and Džeroski [2007] who define, respectively, constraints on the structure of the tree and who define instance-level constraints (especially for clustering).

In our taxonomy, we distinguish three types of constraints (see Figure 3): structure-level constraints, attribute-level (or feature-level) constraints, and instance-level constraints. These constraints can be obtained from three sources (see Figure 4). First, constraints can be derived from another machine learning algorithm (for instance, a Bayesian structure learning algorithm or rule learning algorithm) that we call hereafter constraint mining algorithm. Second, constraints may also come from users; for example, they may want to limit the size of the tree. The term *user constraints* usually indicates in the literature that constraints do not require relevant domain expertise for the learning task [Fromont et al. 2007; Garofalakis et al. 2000, 2003]. Third, and most importantly, constraints can be provided by domain experts with domain knowledge (also known as background knowledge [López-Vallverdú et al. 2012; Núñez 1991]) who can define the constraints related to the domain with respect to the given dataset. With this taxonomy, structure-level constraints may be set by any user to control the complexity of the decision tree. Besides, attribute-level and instance-level constraints are more susceptible to be defined by a domain expert or by a constraint mining algorithm. This section presents the different types of constraints and several works that have studied them in the literature.

3.1 Structure-level Constraints

The structure of a tree relates to how the nodes are arranged relatively to each other (see in Section 2). Therefore, learning decision trees with structure-level constraints means designing a learning algorithm that can find a tree that satisfies a particular property over its structure. For example, trying to find the optimal binary decision tree with a depth of at most d that minimises the misclassification can be formalised as

$$\begin{aligned} \underset{T \in \mathcal{T}}{\text{minimise}} \quad & l(T; \mathcal{D}) \\ \text{subject to} \quad & \text{depth}(T) \leq d \\ & \text{binary}(T) = 1, \end{aligned}$$

where $\mathcal{T}$ denotes the set of decision trees with respect to the given dataset $\mathcal{D}$, $l : \mathcal{T} \times \mathcal{D} \rightarrow \mathbb{R}^+$ is a cost function penalising misclassification and thus is used to train a decision tree, depth : $\mathcal{T} \rightarrow \mathbb{N}$ is a function returning the depth of a specific tree and binary : $\mathcal{T} \rightarrow \{0, 1\}$ tells whether a decision tree is binary or not, and d is a fixed integer.
One of the main advantages of this type of constraint (see Figure 5), according to Piltaver et al. [2016], is that it directly influences the understandability of the decision tree. In what follows, we present approaches developed to enforce structure-level constraints in the learning algorithm and their impact on trees. Section 3.1.1 analyses methods that enforce the size of trees as structural constraints, Section 3.1.2 discusses methods related to the depth constraint, and Section 3.1.3 details methods used to restrict the number of leaves.

3.1.1 Size of the Tree. As defined in Section 2.1, the size of a decision tree is the number of nodes $|V|$ of the tree and is related to the readability of the entire tree [Piltaver et al. 2016]. Several works in the literature, presented hereafter, attempt to impose the size constraint on decision trees. We classify them into three types of methods (see Section 4 for a detailed discussion about optimisation methods): top-down greedy, safe enumeration, and linear (and constraint) programming methods.

Top-down Greedy Algorithms. Top-down greedy algorithms aim to optimise a local heuristic. Here, we focus on approaches that try to learn, in a top-down fashion, trees that do not exceed a maximum number of nodes. Quinlan and Rivest [1989] propose one of the first works in this direction by introducing the **minimum description length (MDL)**. The MDL principle comes from information theory and consists in adding a prior to the optimal tree formulation so as to have the **maximum a posteriori (MAP)** tree. This prior represents the belief of the length of encoding the tree. Since learning the exact MAP is NP-Complete, they propose an approximation based on a greedy top-down method. One might think naively that it could be sufficient to learn a complex (accurate) tree and then prune it to satisfy the size constraint. In contrast to this approach, Garofalakis et al. [2000] introduce a tree algorithm that *pushes* the size constraint into the building phase of the tree. The algorithm estimates a lower bound of the inaccuracy when deciding to split on a node using a top-down fashion. Interestingly, this algorithm can find an optimal tree given a maximum number of nodes, or the other way around: Given an accuracy, find the smallest tree. The minimum accuracy or the maximum size has to be defined by users.

Several works have also proposed to enforce the size constraint on decision trees used as proxy models. Proxy models [Gilpin et al. 2018] are machine learning models that are used for approximating and explaining predictions of black-box models (for instance, an SVM, a neural network, or a random forest). An early work is TREPAN [Craven and Shavlik 1995], which tries to approximate a neural network by a decision tree. To control the comprehensibility of the tree, TREPAN can accept a constraint on the number of internal nodes. Also, the learned rules are $m - of - n$ rules, which are chosen to maximise the information gain ratio of C4.5. Boz [2002] uses genetic algorithms to find interesting inputs of the neural network considered as a black-box classifier, and thereafter learn decision trees via a C4.5-like algorithm. Controlled by the user, the size of the final tree is enforced using post pruning. Yang et al. [2018] recently proposed **GIRP (global model interpretation via recursive partitioning)**. GIRP also uses a CART-like algorithm to learn binary decision trees through a contribution matrix that represents the contribution of input variables [Choi et al. 2016; Ribeiro et al. 2016b] of a black-box machine learning model. To control the size of the tree, authors use a pruning mechanism that adds the size of the tree as a penalising term to the average gain of the tree.

In summary, top-down greedy algorithms that aim to learn decision trees under a size constraint have the advantage to leverage well-known pruning methods and ultimately learn decision trees that meet the size constraint (even if they may grow large at first). Proxy models often apply these

top-down greedy approaches to control the size of the tree to learn clearer explanations of a black-box machine learning model. However, despite the ease of designing a top-down algorithm, these works suffer from the potential sub-optimality of the solution, which is inherently due to the usage of a (local) heuristic.

Safe Enumeration Methods. Safe enumeration methods are designed to enumerate all (or a subset of) possible trees by identifying possible splitting rules with a specific attention to complexity. This allows the exploration of richer trees in terms of accuracy or constraints.

In this direction, Bennett and Blue [1996] propose an algorithm, called global tree optimisation, which uses multivariate splits and models decision tree encoding as disjunctive inequalities with a fixed structure. By showing that they can use various types of objective functions, they present a search method (called extreme point tabu search) based on tabu search [Glover and Laguna 1998] (a heuristic method that uses a local search over the search-space by checking the immediate neighbors of a solution) to heuristically find a good solution. They also showed that it is possible to solve the problem with the Frank Wolfe algorithm and the Simplex method. While the former has the disadvantage of getting stuck into local optima, the latter is costly in terms of computations.

Garofalakis et al. [2003] propose another approach to add knowledge constraints (size of the tree, inaccuracy cost) into the learning phase based on a branch-and-bound algorithm. The authors also use a dynamic programming algorithm whose goal is to prune an accurate and large tree such that it will satisfy the constraints (size constraint). Struyf and Džeroski [2006] extend previous pruning methods to learn multi-objective regression trees with size and accuracy constraints. Fromont et al. [2007] use the analogy of itemset mining to learn decision trees with constraints (size of the tree, errors of the tree, syntactic constraints, i.e., the way attributes are ordered). They proposed two methods: CLUS, which is a greedy method, and CLUS-EX, which is based on an exhaustive search that enumerates possibilities expecting that user constraints are restrictive enough to limit the search space. Nijssen and Fromont [2007] present an algorithm called DL8 for optimal decision trees using dynamic programming. A more general framework [Nijssen and Fromont 2010] is given later by the same authors that uses an itemset mining approach to learn decision trees for various types of structure-level constraints (size of the tree, number of leaf nodes, etc.) and data-level constraints (see Section 3.3). Also based on dynamic programming, this extended version of DL8 needs enough memory to encapsulate all the lattices of variables.

This section focused on methods that aim at enumerating all (or a subset of) possible trees to enforce the size constraint. If these methods have the advantage of learning an optimal solution in certain circumstances, then they can become costly to use. In particular, when there is a large and/or complex space of possible trees (e.g., when the number of features is important), these methods can fail to provide a solution under a reasonable time.

Linear, SAT, and Constraint Programming Formulations. Instead of safely enumerating consistent trees, other methods prefer to formalise the tree encoding in terms of variables and constraints and use a solver to get an optimal tree that satisfies fixed or bounded structures in terms of their characteristics (for example, fixed number of nodes and/or leaves). Bessiere et al. [2009] propose a method to find decision trees with minimum size using constraint programming and **integer linear programming (ILP)**. By presenting an SAT-based encoding of a decision tree, they express all constraints that need to be satisfied by a decision tree. Translating the problem using linear constraints and integer variables makes it possible to use ILP to explore richer and smaller trees. However, computational time remains high. To speed up the search, Narodytska et al. [2018] propose another SAT-based encoding for optimal decision trees based on tree size. The heart of their method is to consider a perfect (error-free) binary classification where the selected features

on nodes and the valid tree topology are modelled with SAT formulae. They search for the smallest decision tree considering that it must perfectly classify examples in the dataset.

Again, the above methods seek optimal solutions, however, depending on the chosen formalisation, different types of solvers have to be used. SAT solvers may be very efficient but are limited to propositional formulae, while CP solvers can handle more complex problems but have difficulty to scale to large search spaces.

Summary about the Tree Size Constraint. To conclude, constraining the size of trees helps to control their complexity, making them more easily understandable and readable. This problem of retrieving decision trees optimised with respect to their size and accuracy has been vastly explored using local search through heuristics, enumeration, and constraint programming approaches. To effectively control the size of the tree, the last two approaches generally assume trees to be binary, even for non-binary categorical variables. This assumption allows to highly reduce the search space. However, top-down greedy approaches do not need this assumption.

3.1.2 Depth of the Tree. The depth constraint is important to control overfitting but also the comprehensibility of decision trees. It usually takes the form of learning a decision tree with a given maximum depth.

Top-down greedy Methods. Diverse algorithms have been proposed to learn interpretable and proxy decision trees under depth constraints. Trying to approximate a neural network, Zilke et al. [2016] propose a constrained and more elaborated version of CRED [Sato and Tsukimoto 2001] (a method that learns decision trees to interpret predictions of a decomposed neural network into hidden units). This version extracts rules for each hidden unit and approximates these local decisions by a decision tree with a depth $K \leq 10$ using a modified version of C4.5.

Safe Enumeration Methods. Enumeration-based methods have been proposed also to learn optimal trees (in terms of classification error). For example, the T2 [Auer et al. 1995] and T3 [Tjortjis and Keane 2002] algorithms, respectively, find optimal trees with a maximum depth to 2 and 3 using a careful exhaustive search based on agnostic learning. The authors of T2 are one of the few authors who proposed a constraint-based tree learning algorithms and to theoretically analyse the computational time complexity and the guarantees of the learning algorithm. T3C [Tzirakis and Tjortjis 2017] is an improved version, which changes the way T3 splits continuous attributes to four decision rules.

To enforce the depth constraint in DL8 (presented in Section 3.1.1), Aglin et al. [2020a] introduce DL8.5. This algorithm uses a branch-and-bound search with caching to safely enumerate trees under the depth constraint. However, unlike DL8, DL8.5 cannot enforce test cost constraints [Nijssen and Fromont 2010].

Linear, SAT, Constraint Programming Methods. For seeking richer and more accurate trees, Verwer and Zhang [2017] present a formulation of the optimal decision tree given a specific depth (i.e., depth constraint) as an integer linear program. By creating variables that link training instances to their leaf nodes, authors are able to formalise, in terms of linear constraints, notions that include, but are not limited to, the selection of features over internal nodes and splitting rules. Furthermore, using this formulation, a tree learned by existing algorithms such as CART can be used as a starting solution for the **mixed integer programming (MIP)** problem. Authors used CPLEX as a MIP solver. In a more general framework, Bertsimas and Dunn [2017] give a new formulation of optimal classification decision trees (called OCT) as a MIP problem. Authors also propose an adaptation to overcome the problem of multivariate splits. They define ancestors of nodes and divide them into two categories: left and right ones. Only considering left ones helped authors to

formalise decision rules and break a symmetry. They specified all the tree consistency constraints as linear constraints that can be pushed to the CPLEX solver to get the optimal tree. Authors claimed to outperform greedy top-down methods, yet there is a need to start with an "acceptable good" solution to reduce computation time. To overcome the computational time problem of OCT, Firat et al. [2020] propose an ILP formulation based on paths for trees with a fixed depth. While using column generation methods or variable pricing with CART to speed up the computations, their formulation allows defining flexible objectives coupled with a regularisation term based on the number of leaf nodes. Aghaei et al. [2021] translated the OCT model into a maximum flow problem, which is optimised with a MIP solver. Although this maximum flow formulation accelerates the optimisation, it only works with binary features and classes, unlike the general OCT.

Instead of making computations faster by looking for a warm-start solution, Verwer and Zhang [2019] propose an algorithm that finds an encoding whose number of decision variables is independent of the size of the dataset. This allows them to introduce a new binary linear program to find optimal decision trees given a fixed depth. Verhaeghe et al. [2019] rather prefer a **Constraint Programming (CP)** modelling inspired by the link on itemset mining of DL8 [Nijssen and Fromont 2010].

After noting that the SAT-based encoding of Narodytska et al. [2018] (see Section 3.1.1) was only applicable to the size-constraint, Avellaneda [2020] proposed a novel SAT-based encoding of the depth-constrained optimal decision tree that does not only minimise the classification error, but also the depth for error-free classification. In addition, this improved version incrementally adds data-related literals and clauses to reduce the computational time and memory requirement. Later, Hu et al. [2020] introduce a MaxSAT version of Narodytska et al. [2018] that integrates depth and size constraints to speed up computations using the Loandra state-of-the-art MaxSAT solver.

Summary about the Depth Constraint. In conclusion, methods to learn decision trees under depth constraints are generally based on enumerations, linear programming, and constraint programming. Their goal is primarily to learn more accurate and most importantly accelerate computations to reach optimal depth-constrained decision trees. Because these methods control the depth of decision trees, they prefer to learn shallow trees (trees with small depth) [Bertsimas and Dunn 2017; Firat et al. 2020] to improve accuracy and interpretability as well. Nevertheless, particular attention was paid to the scalability of these methods, in particular the over-simplicity of learned trees that we discuss later in this article (see Section 6).

3.1.3 Number of Leaf Nodes. The number of leaf nodes is an important factor for the summary of the decision rules and also to limit the growth of the tree that in turn can be important when trying to understand how the model predicts a particular class [Piltaver et al. 2016]. In the case of binary trees, the number of leaf nodes L is linked to the size of the tree $|V|$ by the formula $|V| = 2L - 1$. In other general cases, there is no such explicit formula. However, in this equality, L and $|V|$ do not relate to the same aspects of the explainability of decision trees. The size of the tree is related to the readability of the tree, while the number of leaf nodes is essential for the comprehensibility of the predictions of a given class. Very few studies have focused their interest on constraining the number of leaf nodes.

By drawing inspiration from the work of Angelino et al. [2018] on optimal rule lists, Hu et al. [2019] propose **optimal sparse decision trees (OSDT)** that uses a branch-and-bound search for 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. Thanks to the use of a structural empirical minimisation scheme and analytic bounds, OSDT and its extended version called GOSDT [Lin et al. 2020] learn efficiently trees whose structures are very sparse and therefore likely to generalise well.

Just as linear and constraint programming formulations are used to learn decision trees under size and depth constraints, they can also be used to learn decision trees with a given maximum number of leaf nodes. Namely, to overcome the problem of computation time of MIP solvers in Bertsimas and Dunn [2017], the work of Menickelly et al. [2016] proposes another formulation of decision trees for binary classification as integer programming problem with a predefined size adjusted by the number of leaf nodes. After creating variables that are directly linked to internal nodes, leaf nodes, and attributes, they encode the tree by imposing constraints on these variables. Authors have pushed those constraints into the CPLEX solver with a predefined topology (structure) and they chose the best topology through cross-validation. Their method finds the optimal solution but is limited to binary trees with categorical variables only.

In a completely different way, the work of Nijssen [2008] extends previous works on DL8 algorithm [Nijssen and Fromont 2007] in a Bayesian setting. It proposes a MAP formulation of the optimal decision tree problem under soft constraints (i.e., constraints that might be violated) on the maximum number of leaf nodes. Based on the link between itemsets and decision trees, the algorithm can find predictions using a single optimal tree or Bayesian predictions for several trees. To incorporate the constraints on the number of leaf nodes (although other types of constraints may be targeted), Chipman et al. [1998] present a Bayesian approach to learn decision trees. Their main contribution is to propose a prior over the structure of the tree and a stochastic search of the posterior to explore the search space and therefore find some "acceptable" good trees. The search consists in building a Markov Chain of trees by the Metropolis-Hasting algorithm considering the transitions: *grow* (i.e., split a terminal node), *prune* (i.e., choose a parent of terminal node and prune it), *change* (i.e., change the splitting rule of an internal node), and *swap* (i.e., swap splitting rules of parent-child pair), all randomly. To circumvent the local optima of the previous Bayesian formulation, Angelopoulos and Cussens [2005a] exploit **stochastic logic programming (SLP)** to integrate informative priors (that try to penalise unlikely transitions). They also extend this to a *tempere* version, which improves the convergence and predictive accuracy [Angelopoulos and Cussens 2005b]. However, even though their posterior predictive performs usually well, when selecting a single tree as the mode of their Bayesian formulation, this tree is less accurate than the one learned by greedy algorithms. To interpret Bayesian tree ensembles, inspired by the Bayesian formulation of Chipman et al. [1998], Schetinin et al. [2007] propose a new probabilistic interpretation of Bayesian decision trees, whose goal is to find the most confident tree within the ensemble of Bayesian trees.

In conclusion, the works that deal with the constraints on the number of leaf nodes are generally based on a probabilistic formulation of decision trees. Bayesian learning seems to be suitable for this kind of constraint. However, the challenge is to effectively model the learning of the structure and the selection of rules of the decision tree. Mentioned works [Chipman et al. 1998; Schetinin et al. 2007] learn trees using stochastic search. While having the advantage of integrating constraints in a rigorous and clear mathematical manner with priors, Bayesian formulations of decision trees are also computationally expensive when implemented (see Section 4.4 for more details).

3.1.4 Summary and Discussion about Structure-level Constraints. To sum up, the presented works focus on making decision trees more readable by constraining trees to be small, or limiting the number of decision rules or the number of attributes to take into account in the decision tree, since it is related to the abstraction capabilities of human beings. Also, since structural characteristics of the tree are linked to each other, setting one aspect constrains the others, but they all have different impacts on the interpretability of decision trees. If the size controls the readability of the tree, then the depth defines how easy the interpretability of a prediction can be, and the number of leaf nodes gives an idea of how understandable the prediction among a particular class

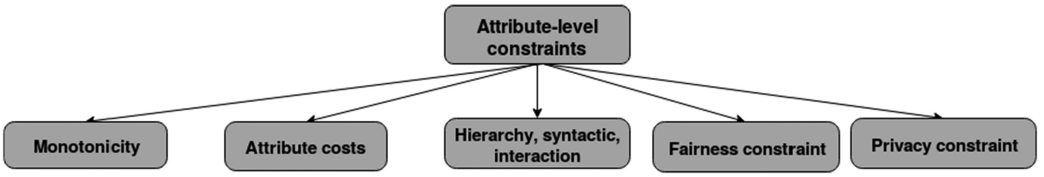


Fig. 6. Attribute-level constraints in decision trees. The top-down arrow shows the different sub-categories of constraints of attribute-level constraints.

is. Presented works that take into account these constraints try to find optimal solutions, while others leverage heuristic-based approaches to either explore richer trees or to learn more accurate ones compared to traditional methods. Besides, works in the direction of proxy models for black-box classifiers consider the structural constraints to make sure that the resulting tree will be easily understandable.

Nevertheless, the tree balance constraint (also depending on the branching factor of the tree) is understudied in the literature, despite its impact on the readability and therefore the comprehensibility of decision trees. Also, the majority of the works on structure-level constraints assume that decision trees are binary to better handle the number of leaf nodes and the sizes. This assumption is severely lacking in flexibility. In some cases, for the sake of interpretability, it would be useful to have nodes with more than two child nodes. For example, if a categorical variable like the number of doors of a car has three values (namely, 2, 4, and 6), then it may be important, for comprehensibility purposes, not to transform this variable into two binary variables so the knowledge "number of doors" appears only once in a branch of the tree. Additionally, another generally common assumption is that shallow trees (i.e., trees with small depth) and small trees (small size) enhance the interpretability and the comprehensibility. This question requires further study because, actually, in certain domains such as health care, a decision tree with a maximum depth of two such as in Bertsimas and Dunn [2017] and Tjortjis and Keane [2002] could not be comprehensible by experts [Freitas 2014; López-Vallverdú et al. 2007]. Indeed, rules may be too simple to fit domain-expert requirements. To overcome this problem, it may be necessary to add human and expert knowledge as constraints when learning decision trees.

3.2 Attribute-level Constraints

Attribute-level constraints are defined as properties over the features of the dataset. Therefore, they are directly linked to the rules and the parameters of the tree, since the rules, coupled with the structure, can be considered as the parameters learned by a decision tree algorithm (see Section 2). These constraints are more likely to come from an expert or a constraint mining algorithm. Most of the works on attribute-level constraints study monotonicity, attribute costs, and hierarchy constraints including interactions between multiple features, privacy, and fairness (see Figure 6).

3.2.1 Monotonicity Constraints. Monotonicity constraints are related to attributes that evolve in the same direction as the output variable. More formally, it is defined in Pei et al. [2016] as follows: Given a set of attributes $\mathcal{A} = \{\mathbf{X_1}, \ldots, \mathbf{X_M}\}$, one of its subset $\mathcal{B} = \{\mathbf{X_k}, \ldots, \mathbf{X_l}\}$ with $1 \leq k \leq l \leq m$, the decision rule $T_\mathbf{X}$ of a tree T is monotonically consistent in terms of $\mathcal{B} \subseteq \mathcal{A}$, in a set $\mathcal{U}$ of examples drawn from the dataset $\mathcal{D}$ if

$$\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{U}, \mathbf{x}_i \preceq_\mathcal{B} \mathbf{x}_j \Longrightarrow T_\mathbf{X}(\mathbf{x}_i) \leq T_\mathbf{X}(\mathbf{x}_j),$$

where $\preceq_\mathcal{B}$ defines a partial relation order on the instances regarding the subset of attributes $\mathcal{B}$.

For instance, let us suppose a dataset for a loan credit classification problem where the decision is whether to grant a credit or not. One would like the decision rule of the learned tree to be monotonically consistent in terms of the income on the entire dataset (in this context: $\mathcal{U}$ is the entire dataset, $\mathcal{A}$ is the set of attributes of the dataset, and $\mathcal{B}$ is the singleton of the income attribute). It is important to note that a monotonicity constraint is an attribute-level constraint: It is not targeted to specific instances but rather on particular attributes.

Classification with monotonicity constraints is an important and well-studied problem, since monotonicity improves the comprehensibility of classifiers like decision trees and therefore increases the acceptability of a classifier in certain applications [Freitas 2014].

Studies have been done to enforce monotonicity constraints on decision trees. Potharst and Feelders [2002] proposed a survey about decision tree methods and this type of constraint. A particular method is from Ben-David [1995], who designed a metric that can take into account the monotonicity constraint without loss of accuracy. This score, called the *total ambiguity score*, has two components. The first component is a non-monotonicity score based on a matrix representing a non-monotonicity constraint over branches of the trees. The second component is the ID3 score based on information theory. The total ambiguity score allows Ben-David [1995] to make a tradeoff between the accuracy given by ID3 and the non-monotonicity score. In this direction of a score-based constraint enforcement, Marsala and Petturiti [2015] proposed three measures using the notion of *dominance rough set* to integrate monotonicity constraints. These three discrimination measures can show the discrepancy of the monotonicity of a specific attribute variable from his class attribute. They come from a version of mutual information adapted for ranking in the context of ordinal classification [Hu et al. 2010]. In the same line, to take into account monotonicity constraints, Hu et al. [2012] developed the rank entropy, which is still based on dominance rough sets but performs better than the rank mutual information. Note that these score-based monotonicity techniques allow learning decision trees greedily in a top-down fashion.

Feelders and Pardoel [2003] rely on pruning methods to learn decision trees on monotone and non-monotone datasets. They develop several fixing methods that aim to make monotone subtrees using overfitted trees. They also show how these fixing methods could be combined with existing cost-complexity pruning methods. Similar work is done in Cardoso and Sousa [2010], where a relabelling technique (class assignment step) enforces the monotonicity of the tree after a tree has been learned from a greedy algorithm such as C4.5. In the same logic, Kamp et al. [2009] propose to relabel non-monotone leaf nodes, then prune the learned tree using properties of isotonic functions, so the final tree will satisfy monotonicity constraints.

Briefly, monotonicity constraints can improve the comprehensibility of decision trees [Freitas 2014]. Most of the works that learn trees under the monotonic constraints introduce a monotonicity score to make a heuristic (such as Gini or entropy) able to detect non-monotonicity. Other studies use post-pruning methods to force trees to satisfy this type of constraint. However, it remains difficult to learn accurate decision trees while satisfying the monotonicity constraint.

3.2.2 Attribute Costs. Cost constraints for attributes are related to the more general topic of cost-sensitive decision trees. The motivation of this field is essentially based on the medical domain in which doctors have to make an appropriate diagnostic by taking into account economic constraints to make patients pass tests. The aim of cost-sensitive decision trees [Lomax and Vadera 2013] is to learn decision trees with a good tradeoff between the misclassification error/cost and the sum of the cost of attributes that can be expressed as constraints. Lomax and Vadera [2013] categorise methods into greedy and non-greedy methods. Greedy methods can use the cost during the tree learning by modifying heuristics [Davis et al. 2006; Freitas et al. 2007; Li et al. 2015; Ling et al. 2004; Norton 1989; Núñez 1991; Pazzani et al. 1994; Tan 1993; Wang et al. 2018] or by

post-pruning [Ferri et al. 2002; Pazzani et al. 1994], which necessitate relabelling techniques. Non-greedy methods are usually based on genetic algorithms [Krętowski and Grześ 2006; Omielan and Vadera 2012; Turney 1995], wrapper methods [Estruch et al. 2002], and stochastic search methods [Esmeir and Markovitch 2004, 2006].

Enforcing cost constraints on decision trees makes them more natural, reliable for practical applications [Qiu et al. 2017], such as loan credits in finance or medical diagnoses in the health domain.

3.2.3 Hierarchy and Feature Interaction. The concept of hierarchy defines an ordering between variables selected on a decision tree. It is also mentioned in Fromont et al. [2007] as syntactic constraints and in Nijssen and Fromont [2007] as path constraints. Hence, it is expressed as an attribute that must be selected before another one over a branch or even over the entire decision tree. Some domain knowledge can refer to such kind of preferences where some attributes should be tested before others, hence affecting the hierarchy to be learned. For instance, in the medical domain, doctors might want to perform temperature checks or blood checks before going after more advanced tests. As a result, it makes the decision tree more reliable and more comprehensible for domain experts.

Núñez [1991] was one of the first works in this direction (because the issue was presented just a few years after ID3 and CART were introduced to the community). The purpose is to make decision trees more compliant with background knowledge using **ISA (Is A)** relationships. ISA relationships represent a hierarchy relationship between two attributes: one that belongs to the concept of the other (for example, an amphibian is a vertebrate). Núñez [1991] introduces a cost-sensitive measure that incorporates ISA relationships between variables to make the tree more comprehensible from a user perspective. Using this measure, Núñez [1991] can also integrate attribute costs and learn decision trees in an ID3 fashion.

López-Vallverdú et al. [2007] rather suggest to modify the list of available attributes at a given node of the tree during learning. This list comes from the pairwise relationship of attributes given by the expert. Iqbal et al. [2012] propose another way of integrating feature importance in the heuristic. They propose a score that estimates the probability for a feature to influence the target variable. Moreover, their method employs a top-down fashion. Since this score decreases when the number of instances increases, domain knowledge in the form of feature importance becomes dominant at the bottom of the tree, thus improving its performance.

In the same vein, to improve medical decisions, works of Torres et al. [2011] and López-Vallverdú et al. [2012] present a formalism to express background knowledge using health-care criteria. They propose a novel algorithm to learn more comprehensible trees using a formalism based on priority and relevance of features.

Another domain for which attribute-level constraints have shown to be important is intrusion detection. An intrusion detection system aims at detecting malicious activities in a computer network. There are two main methods used in this domain: anomaly detection and signature-based detection. While anomaly detection tries to discover patterns of sequences that lead to intrusions, signature-based methods have already probed the network and know which patterns lead to intrusions. Kruegel and Toth [2003] propose to learn signature rules through clustering and use these rules as constraints to learn more interesting rules by a decision tree. It is a signature-based detection. They learn decision trees in a top-down fashion using a modified version of entropy that takes into account the signature rules.

Imposing learned trees to be multivariate allows mixing decision trees with other classification algorithms. While Bennett and Mangasarian [1992] use MIP to find multivariate splits, Bennett and Blue [1998] formulate the multivariate tree learning problem given a number of nodes as learning

decision boundaries of SVMs. Furthermore, they show that this approach can be extended with kernel tricks to multivariate linear, nonlinear, polynomial decisions, and even neural networks with sigmoid functions. For another purpose, to circumvent the multi-class problem of SVM, Madzarov et al. [2009] propose to learn a decision tree along with SVMs as a splitting criterion, but their algorithm follows a clustering-based method to find the two classes whose instances are very far from each other. Then an SVM is learned inside an internal node considering these classes.

As a summary, imposing a specific hierarchy and feature interaction on decision trees can improve the comprehensibility and the trustworthiness of the tree. This usually requires additional knowledge either from experts or from a mining algorithm. The works presented here are usually applied in areas where comprehensibility may be more important than accurate predictions, for instance in the medical field. Besides, enforcing feature interactions leads to an improvement in the accuracy of predictions.

3.2.4 Privacy Constraints. In the context of either vertically distributed datasets (among different sources) or datasets with private sensitive features, it is essential to incorporate privacy constraints in the formulation of the decision tree learning problem. Privacy constraints guarantee that the learning algorithm may not access, discover, or use sensitive information to learn or predict [Vaghashia and Ganatra 2015]. Hence, it is related to both computer security and machine learning.

Various algorithms have been proposed in the literature for privacy constraints in machine learning. Du and Zhan [2002] propose a method to compute an impurity measure from two vertical/horizontal partitions of datasets using an untrusted third party. In the logic of horizontal partitioning data with privacy constraints, Gangrade and Patel [2012] propose a secure protocol with another untrusted third-party to learn decision trees on horizontal partitions of the dataset. In the setting of federated learning, Li et al. [2020a] propose a framework to learn decision trees and gradient boosting trees using hashing. Interestingly, they prove that their framework satisfies the privacy model. While Vaidya et al. [2008] and Gangrade and Patel [2009] present a modified version of ID3 and C4.5, respectively, with a secure channel to exchange sensitive attributes, Matwin et al. [2005] prefer early stopping and pruning methods to check the satisfiability of privacy constraints during the tree learning process. However, Friedman et al. [2006] propose the integration of the k-anonymisation [Sweeney 2002] (a process to obtain an anonymous dataset while maintaining the usefulness of the data) method into the learning process of the decision tree. Teng and Du [2007] come with a hybrid approach that uses both k-anonymisation and secure multiple channels. Alternatively, Zaman et al. [2016] propose a method that generalises values of sensitive attributes to anonymise the dataset. After proposing a new perturbation method of the training data, Liu et al. [2009] present a modified version of C4.5 based on a new estimation of information gain with noisy perturbed data. However, this version of C4.5 results in loss of performance. To address the issue of possible loss of performance, Li et al. [2020b] introduce two algorithms based on differential privacy to guarantee privacy on decision trees and gradient boosting trees.

Learning decision trees under privacy constraints becomes an issue as soon as preserving both the privacy and the comprehensibility of the tree is needed. Several works have been proposed to tackle this problem but, in certain cases, at the cost of sacrificing the comprehensibility of the tree. Aside from secure multi-channel and anonymisation methods, works in the sense of applying perturbations with random noise, and most importantly differential privacy [Friedman and Schuster 2010], seem increasingly promising. The reason is that these directions propose a rigorous mathematical way to tackle privacy constraints without focusing on security aspects.

3.2.5 Fairness Constraint. Another constraint that can be considered as an attribute-level constraint is the fairness constraint, because it is related to a sensitive attribute. However, actually, it is

in-between attribute-level and instance-level (see Section 3.3) constraints. The fairness constraint ensures that the probabilities of correctly classifying instances from different groups according to a sensitive attribute should be almost equal. A decision tree T is said to be formally fair in expectation [Fitzsimons et al. 2019], for two groups of instances $\mathcal{A}, \mathcal{B} \subseteq \mathcal{D}$, if its decision rule $T_{\mathbf{X}}$ verifies

$$\mathbb{E}[T_{\mathcal{X}}(\mathbf{x}_a)|\mathbf{x}_a \in \mathcal{A}] - \mathbb{E}[T_{\mathbf{X}}(\mathbf{x}_b)|\mathbf{x}_b \in \mathcal{B}] = 0.$$

In this case, the fair decision tree does not discriminate against a particular group (e.g., female vs. male for the sex attribute).

With the hypothesis that discrimination could not be justified in the task (that the tree is learning to perform), Kamiran et al. [2010] propose two methods to learn fair decision trees. The first one uses a modified version of information gain that incorporates the influence of a split on the discrimination. The second one is a relabelling technique that minimises the discrimination of a learned decision tree using the empirical joint distribution between the sensitive attribute and the class attribute on all the leaf nodes. Recently, in the same direction, Raff et al. [2018] adapted the learning algorithm of a C4.5 decision tree to comply with fairness constraints. They proposed two measures for fairness in the case of either categorical sensitive feature or continuous one and a new fair heuristic gain measure. Using linear constraints, Aghaei et al. [2019] proposed a general framework that encapsulates the learning problem of optimal fair decision trees through MIP. With their formulation, they can use it for both classification and regression. They integrate the fairness constraint as a regularisation term on the misclassification error or mean absolute error. However, this optimal formulation needs hours to provide good results, unlike the heuristic-based method of Raff et al. [2018].

3.2.6 Summary and Discussion about Attribute-level Constraints. To summarise, we have subcategorised attribute-level constraints into monotonicity constraints, cost of attributes, hierarchies, privacy constraints, or fairness constraints. Monotonicity constraints may be useful to learn trees that can be more interpretable and trustworthy [You et al. 2017]. Diverse works constrain the decision tree to be monotone by rewriting heuristics of traditional algorithms as measures that incorporate the monotonicity constraints [Hu et al. 2012, 2010; Marsala and Petturiti 2015; Pei et al. 2016]. Others prefer using a post-pruning method to enforce the monotonicity.

Hierarchy, feature interaction, and attribute cost constraints are well studied because of their necessity in real-world applications. While the two first constraints have the advantage of being defined by domain experts or by a constraint mining algorithm [Iqbal et al. 2012], the problem of enforcing attribute cost constraints is much more studied due to existing datasets (notably cost-sensitive datasets in the health care domain presented in Kachuee et al. [2019] and Turney [1995]) on cost-sensitive classification.

Privacy constraints aim at preventing attacks (e.g., to discover sensitive information) over instances and are generally considered for security purposes. Thus, works in this field can be seen in the pure security formulations with secure channels, perturbations, anonymisation [Fletcher and Islam 2019], and most importantly in mathematical formulations with differential privacy whose purpose is to guarantee that computations are invariant to (noisy) perturbations over data [Friedman and Schuster 2010]. The main difficulty here is to guarantee the balance between accuracy, privacy requirements, and interpretability of the decision tree. For example, finding an optimal depth while satisfying the differential privacy constraint is still an open issue [Fletcher and Islam 2019].

Finally, fairness constraints that act as non-discrimination constraints are widely understudied for decision trees, while being of particular interest in machine learning recently. In fact, very few works exist [Kamiran et al. 2010; Raff et al. 2018] even though diverse mechanisms in machine

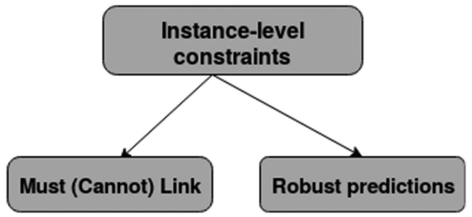


Fig. 7. Instance-level constraints in decision trees.

learning have been proposed [Zafar et al. 2017]. This is because standard decision tree algorithms do not optimise directly a global objective, and thus it is quite difficult to apply these methods to learn discrimination-aware decision trees. That is why learning fairer decision trees could be easily handled with constraint programming methods that optimise explicitly a global objective function, as proposed in Aghaei et al. [2019].

3.3 Instance-level Constraints

Instance-level constraints (see Figure 7) are specified on some particular instances of the dataset. For example, one can specify that certain examples may never be misclassified. Another example of such constraints is related to clustering and thus clustering trees: instance-level constraints, such as must link or cannot link [Wagstaff et al. 2001], which specify that some examples must belong to the same class or must belong to different classes.

3.3.1 Must (or Cannot) Link, Partitions and Robust Predictions for Certain Instances. The must-link and cannot-link constraints are popular in clustering. Since decision trees are mainly used for classification and regression, these constraints are not widely studied in the decision tree literature. Struyf and Džeroski [2007] propose an algorithm called ClusILC, which learns clustering trees by integrating domain knowledge in terms of instance-level constraints (must-link or cannot-link). These constraints are treated as soft constraints, since some of them can be violated. The authors also present a global heuristic that is decomposed by the average variance of the leaves nodes normalised by the total variance and the percentage of violated constraints. Therefore, they propose a greedy hill-climbing algorithm whose goal is to learn clustering trees by maximising this heuristic.

In another direction, one can specify constraints on measures (for example, information gain) directly linked to instances. This is the case for Sethi and Sarvarayudu [1982], who propose an algorithm that learns top-down induction trees for classification by using the average information gain on the tree. The algorithm attempts to find the best splitting values of features in nodes. It is a recursive partitioning algorithm, which aims to maximise the average mutual information gain of the tree, given a probability of error that integrates a belief on instances that may be misclassified.

To deal with the problem of scalability, Gehrke et al. [1999] present BOAT, a method that learns trees on a subset of the dataset to make the learning process of decision trees faster in the case of large datasets. Tolomei et al. [2017] leverage data-level constraints on learned decision trees to explain random forests.

Although adversarial examples applied on decision trees are very understudied in the literature, decision trees have shown to be naturally sensitive to adversarial examples [Chen et al. 2019; Cheng et al. 2019; Kantchelian et al. 2016; Papernot et al. 2016]. To overcome this problem, Chen et al. [2019] propose a robust version of information gain that can serve to improve the robustness of decision trees learned in a greedy top-down fashion. Calzavara et al. [2020] rather optimise an evasion-aware loss function as the heuristic, and Calzavara et al. [2019] show how to extend adversarial training for decision trees and gradient boosting trees.

3.3.2 Summary about Instance-level Constraints. In summary, it is possible to integrate domain knowledge through data-level instance constraints. But methods dealing with this type of constraints are very limited, because they require external information (usually from domain experts) on instances to integrate them into the learning algorithm of the tree. The presented methods deal with the problem of learning on partitioning and sampling data, must-link and cannot-link constraints, adversarial examples, and instances that must be well-classified. While the first kind of constraint aims to speed up the learning of the tree, the others contribute to learning trustworthy decision trees. A particular attention is given to the robustness of decision trees to adversarial noise. In fact, the works tackling this issue might be limited, because, if the rules of the decision trees are known, it is easy to generate adversarial examples. However, considering so-called white-box attacks, it should be interesting to learn more robust decision trees and the question becomes more challenging for the case of black-box and transferable attacks [Papernot et al. 2016] (i.e., adversarial examples generated from a particular classifier and which can be applied to another one).

3.4 Summary over the Taxonomy

We have presented a taxonomy of constraints (tree structure, attribute, and instance-levels) that may be important when designing efficient learning algorithms requiring the learned decision trees to satisfy explicitly formulated constraints. These levels seem orthogonal at first, sometimes constraining the structure of trees, sometimes ensuring high-level considerations such as data privacy. Structure-level constraints are mainly used to control the complexity of the model and might be set by the user. Attribute-level constraints and instance-level constraints require more expertise to be defined. Hence, there is often a need for an expert or a constraint mining algorithm to define them. The advantage of these types of constraints is their ability to produce trustworthy decision trees and thus enforce the interpretability of the resulting tree. Although they apply on various aspects of the learning algorithm, it is possible to mix all these types of constraints in a framework to have a human-oriented optimal tree that satisfies those constraints. In addition, a constraint on a measure (for instance, information gain must be less than a threshold) can be seen as a mixture of attribute, instance-level, or structure-level constraints. As another example, imposing that the number of instances inside a leaf node is more than a fixed number (which is a very useful constraint to reduce overfitting) can be seen as instance-level and structure-level constraints. So, rather than being orthogonal disjoint types of constraints, this taxonomy is flexible, since a particular level of constraints can potentially impact other levels of constraints. Hence, more constraints can be formulated as a mixture of different levels of constraints.

4 GENERAL OVERVIEW OF OPTIMISATION APPROACHES FOR CONSTRAINT ENFORCEMENT

In this section, we present our categorisation of approaches for learning decision trees through constraint enforcement. This section differs from Section 3 by the fact that it presents the general optimisation approaches of works that have been presented in the previous section. These works use either top-down greedy methods, safe enumeration methods, linear and constraint programming approaches, and discriminative and Bayesian approaches. Table 1 provides an overview of these approaches for the different types of constraints.

4.1 Top-down Greedy Approaches

These methods learn trees in a top-down manner by choosing for each node the test that optimises a specific heuristic (generally a local heuristic, which depends on a subset of data and a subset of attributes). The majority of previously formulated constraints can be incorporated into a heuristic

Table 1. Overview of the Approaches for the Different Types of Constraints

	Structure-level	Attribute-level	Instance-level
Greedy top-down	Boz [2002]; Craven and Shavlik [1995]; Garofalakis et al. [2000]; Quinlan and Rivest [1989]; Wu et al. [2016]; Yang et al. [2018]; Zilke et al. [2016]	Ben-David [1995]; Cardoso and Sousa [2010]; Daniëls and Velikova [2003]; Davis et al. [2006]; Feelders and Pardoel [2003]; Ferri et al. [2002]; Freitas et al. [2007]; Friedman et al. [2006]; Gangrade and Patel [2009, 2012]; Iqbal et al. [2012]; Kamp et al. [2009]; Kruegel and Toth [2003]; Li et al. [2020a, b, 2015]; Ling et al. [2004]; Liu et al. [2009]; López-Vallverdú et al. [2012, 2007]; Matwin et al. [2005]; Nanfack et al. [2021a]; Norton [1989]; Núñez [1991]; Pazzani et al. [1994]; Potharst and Feelders [2002]; Sweeney [2002]; Tan [1993]; Teng and Du [2007]; Vaidya et al. [2008]; Wang et al. [2018]; Zaman et al. [2016]	Calzavara et al. [2019, 2020]; Chen et al. [2019]; Gehrke et al. [1999]; Kamiran et al. [2010]; Raff et al. [2018]; Sethi and Sarvarayudu [1982]; Struyf and Džeroski [2007]
Safe enumeration methods	Auer et al. [1995]; Bennett and Blue [1996]; Fromont et al. [2007]; Garofalakis et al. [2003]; Hu et al. [2019]; Kocev et al. [2007]; Lin et al. [2020]; Struyf and Džeroski [2006]; Tjortjis and Keane [2002]; Tzirakis and Tjortjis [2017]	Esmeir and Markovitch [2004, 2006]; Estruch et al. [2002]; Krętowski and Grześ [2006]; Madzarov et al. [2009]; Omielan and Vadera [2012]; Turney [1995]	Gehrke et al. [1999]; Nijssen and Fromont [2007, 2010]
Linear, SAT, and constraint programming	Aghaei et al. [2019, 2021]; Avellaneda [2020]; Bertsimas and Dunn [2017]; Bessiere et al. [2009]; Firat et al. [2020]; Heidenberger [1996]; Hu et al. [2020]; Menickelly et al. [2016]; Narodytska et al. [2018]; Verhaeghe et al. [2019]; Verwer and Zhang [2017, 2019]	Aghaei et al. [2019]; Bennett and Mangasarian [1992]	[Bertsimas and Dunn 2017; Verhaeghe et al. 2019]
Discriminative and Bayesian learning	[Angelopoulos and Cussens 2005b, a; Buntine 1992; Chipman et al. 1998; Denison et al. 1998; Nijssen 2008; Nuti et al. 2019; Schetinin et al. 2007; Wu et al. 2007]	Angelopoulos and Cussens [2005b, a]	Angelopoulos and Cussens [2005b, a]; Buntine [1992]; Chipman et al. [1998]; Denison et al. [1998]; Nuti et al. [2019]

The table can be read horizontally to look for works using a particular optimisation method, or it can be read vertically to look for works that enforce a specific type of constraints. The cells of the table indicate works given the type of constraints in the header and the optimisation method in the table row.

to build trees greedily. They make sure to push local decisions towards a tradeoff between satisfying the stated constraints and the expected accuracy of the tree. If they have the advantage of being fast to compute, then they have the big disadvantage of providing usually sub-optimal decision trees. Since they do not optimise directly a global objective, they are not naturally suited for global constraints (for example, the size of the tree, monotonicity constraints for monotone

datasets). This is why post-strategies such as pruning methods are often applied to make sure that the global constraints are satisfied even when these constraints can be integrated into the heuristic [Choi et al. 2016; Garofalakis et al. 2003; Kamiran et al. 2010]. For structure-level constraints, this type of method is generally an extension of pruning methods. Top-down greedy algorithms that integrate structure-level constraints include Boz [2002], Craven and Shavlik [1995], Garofalakis et al. [2000], Quinlan and Rivest [1989], Wu et al. [2016], Zilke et al. [2016]. Few works handle structure-level constraints with top-down greedy approaches, as they are naturally designed to learn the most accurate trees even if the resulting trees can be large. For such methods that rely on top-down greedy approaches, they try first to guarantee a level of accuracy by learning possible large trees and, second, to enforce structure-level constraints using a (post) pruning strategy. This explains why they can be seen as pruning methods.

To enforce attribute-level constraints, top-down greedy algorithms are very popular for two reasons. First, the main algorithm is generic enough to be easily customisable. Second, in real-world applications, domain experts provide domain knowledge than can be transposed to attribute-level constraints. Then practitioners can easily develop models that comply with domain knowledge. Works in this sense include Ben-David [1995], Cardoso and Sousa [2010], Daniëls and Velikova [2003], Davis et al. [2006], Feelders and Pardoel [2003], Ferri et al. [2002], Freitas et al. [2007], Iqbal et al. [2012], Kruegel and Toth [2003], Li et al. [2015], Ling et al. [2004], López-Vallverdú et al. [2012, 2007], Norton [1989], Núñez [1991], Pazzani et al. [1994], Potharst and Feelders [2002], Tan [1993], and Wang et al. [2018].

The studied top-down greedy methods can also integrate instance-level constraints, as in the work of Chen et al. [2019], Gehrke et al. [1999], Kamiran et al. [2010], Raff et al. [2018], Sethi and Sarvarayudu [1982], and Struyf and Džeroski [2007], although the studies in instance-level constraints on decision trees are relatively limited.

4.2 Safe Enumeration Approaches

Apart from greedy algorithms, diverse works try to enumerate the possibilities of choosing splits through careful directives with respect to the constraints, while simultaneously proposing mechanisms to break the complexity. Beyond the brute force method, methods exist to choose the best split criterion randomly according to a minimum/maximum number of possibilities. Even though these methods have the advantage of leaving the greedy direction, a major drawback is the difficulty of easily incorporate constraints. Also, a thorough and practical study must be done to break the complexity of the proposed learning algorithm. Works tailored in this approach include Auer et al. [1995], Bennett and Blue [1996], Fromont et al. [2007], Garofalakis et al. [2003], Kocev et al. [2007], Tjortjis and Keane [2002], Tzirakis and Tjortjis [2017], Esmeir and Markovitch [2004, 2006], Estruch et al. [2002], Krętowski and Grześ [2006], Madzarov et al. [2009], Omielan and Vadera [2012], Turney [1995], and Gehrke et al. [1999]. Works using this approach [Fromont et al. 2007; Nijssen and Fromont 2007] can sometimes guarantee the optimality of decision trees with restrictive assumptions on the search space (for instance, the binarity of the tree, the number of constraints, the size of the dataset).

4.3 Linear, SAT, and Constraint Programming Approaches

The combinatorial nature of the search space and the logical constraints that can be imposed on the decision trees are highly related to **constraint satisfaction problem (CSP)** or linear programming problems or SAT as well. In contrast to top-down greedy and safe enumeration methods, these approaches formalise the problems in an adequate form (e.g., linear, SAT, MaxSAT, or CP) so it can be optimised by an appropriate solver.

At first glance, this method would provide optimal guarantees. In fact, formulating the tree learning problem in terms of global optimisation makes it possible to focus more on the modelling than on the algorithm. This is very suitable for providing mathematical guarantees such as fairness [Aghaei et al. 2019]. Also, it allows an easy way to integrate constraints as regularisation terms, even for structure-level constraints [Bertsimas and Dunn 2017]. However, optimality is usually not attained, as most works require acceptable initial solutions and the majority of CSP, ILP, and MIP problems are NP-complete, even with restrictive assumptions. Thus, the scalability of the problem and the sub-optimal solutions with time limits remain problematic. Related works include Aghaei et al. [2019], Bertsimas and Dunn [2017], Bessiere et al. [2009], Firat et al. [2020], Heidenberger [1996], Menickelly et al. [2016], Narodytska et al. [2018], Verwer and Zhang [2017, 2019], and Bennett and Blue [1996].

4.4 Discriminative and Bayesian Learning

Bayesian methods give a probabilistic formulation of the problem. They are efficient to integrate several constraints with priors. Few works on this approach exist, because the choice of the prior and computing the posterior are still open problems. Works that propose priors and approximations of the posterior include Buntine [1992], Chipman et al. [1998], Denison et al. [1998], Nijssen [2008], Nuti et al. [2019], and Schetinin et al. [2007]. The Bayesian formulation has the advantage to integrate constraints with a clear and rigorous mathematical way through priors. Using also a probabilistic formulation, discriminative learning transforms the non-parametric problem into a parametric one, and the combinatorial space into a real space, to learn trees with standard gradient descent optimisation. However, finding such transformation can be difficult and the learned tree may lose interpretability, since the gradient descent optimisation aims for high accuracy, resulting in potentially complex trees. Norouzi et al. [2015] is an example of learning multivariate decision trees subject to structure-level constraints with gradient descent.

4.5 Summary about Categorisation of Approaches

To summarise, we have identified several optimisation methods that we categorised in top-down greedy induction, safe enumeration approaches, mixed integer programming, and Bayesian and gradient-based approaches (see Table 1 for a global summary). Due to their low computational cost, historical developments in early methods such as CART and C4.5 and their ease of implementation and modification, greedy top-down methods are the most studied. The underlying top-down induction algorithm remains similar in structure and is easy to understand, yet it may provide sub-optimal solutions due to its greedy nature. However, safe enumeration methods are computationally costly but are likely to produce more accurate decision trees. However, finding a solution with safe enumeration may be difficult, because different constraints have to be simultaneously satisfied, making the search more complex. Bayesian approaches are challenging, because priors must be carefully designed to soundly enforce constraints. They are also rather expensive due to the use of sampling strategies. Finally, CP/SAT/MIP models offer an alternative to mathematically specify constraints. If decision trees are part of a bigger problem of a decision system that need to impose some constraints, then these additional constraints only need to be formulated in the SAT/CP/MIP modelling framework to learn trees that are consistent with those constraints by design. However, these CP/SAT/MIP methods are expensive to use and often require to be initialised with satisfactory solutions.

In summary, this survey shows that there is no one-size-fits-all solution to the problem of learning decision trees with user and domain knowledge constraints. Depending on the characteristics of the dataset itself, the type and number of constraints, the acceptable discrepancy in accuracy

Table 2. Available Libraries and Packages for Learning Decision Trees under Constraints

Methods	Library/Package
Garofalakis et al. [2003]	DecisionConstraints [Roşca 2019]
Wu et al. [2016]	Scikit-learn [Buitinck et al. 2013], Weka [Hall et al. 2009]
Hu et al. [2012]; Marsala and Petturiti [2015]	MonDT [González 2019]
Chen et al. [2019]	RobustTrees [Chen 2019]
Kocev et al. [2007]; Struyf and Džeroski [2007]	Clus [Struyf et al. 2017]
Bertsimas and Dunn [2017]	Pyoptree [Pan 2019]
Angelopoulos and Cussens [2005b, a]	Bims [Angelopoulos and Cussens 2016]
Nuti et al. [2019]	Bayesian_tree [Murray and Thommen 2019]
Aglin et al. [2020a]	PyDL8.5 [Aglin et al. 2020b]
Verhaeghe et al. [2019]	https://bitbucket.org/helene_verhaeghe/classificationtree
Verwer and Zhang [2019]	https://github.com/SiccoVerwer/binoct
Avellaneda [2020]	https://github.com/FlorentAvellaneda/InferDT
Aghaei et al. [2021]	https://github.com/pashew94/StrongTree
Hu et al. [2020]	https://gepgitlab.laas.fr/hhu/maxsat-decision-trees
Calzavara et al. [2020]	https://github.com/gtolomei/treant
Hu et al. [2019]	https://github.com/xiyanghu/OSDT
Lin et al. [2020]	https://github.com/Jimmy-Lin/GeneralizedOptimalSparseDecisionTrees

and the available computational time, one method will be preferred to the others. The high computational cost of non-greedy methods explains their infrequent use (until recently).

Table 1 confirms that top-down greedy algorithms have been largely studied. Many works have proposed to improve them, which is natural, since they were prominent in both the literature and practical applications, and they were easier to modify and to implement than their computational costly competitors. This has to be put in perspective of recent works that are focusing on linear and constraint programming approaches, thanks to efficient implementation supported by faster computations. We expect this trend to only increase in the future, leading to new developments.

5 LIBRARIES AND PACKAGES FOR LEARNING DECISION TREES UNDER CONSTRAINTS

This section briefly presents packages and source code of methods to learn constrained decision trees. It aims at helping readers and practitioners by showing a panel of libraries and packages, as well as their related references and papers or repositories permitting to download the tools. The reported packages and implementations are a representative sample, as they cover structure-level, attribute-level, and instance-level constraints. Most of these packages are implemented in Python or have a Python wrapper that makes them easy to use. However, we notice that each paper has produced its own library or package. To the best of our knowledge, we do not know any libraries or packages that would embed several works (either gathering different methods or that would implement a method in different ways) into the same artefact to make reproduction, adaptation, or comparison easier. Table 2 presents in the first column a list of methods whose implementations are currently accessible (via the second column) at the time of writing this document.

6 DISCUSSION

This survey reviewed how constraints can be defined and applied to decision trees, to make them safer, more accurate, more understandable, more robust, or more trustworthy. This section provides a discussion on learning decision trees through constraint enforcement. Specifically, we express the difficulty of traditional algorithms to enforce constraints on decision trees. The optimality and interpretability of learned decision trees under constraints are also discussed. Finally, we mention what could be future research directions in this field.

Table 3. A Benchmark for Depth-constrained Decision Tree Learners

	Top-down greedy			MIP						SAT			CP			Safe enumeration					
	CART			BinOCT			OST			MaxSAT_DT			[Verhaeghe et al. 2019]			DL8.5			OSDT		
	[Breiman et al. 1984]			[Verwer and Zhang 2019]			[Aghaei et al. 2021]			[Hu et al. 2020]						[Aglin et al. 2020a]			[Hu et al. 2019]		
Dataset	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime
Balance S.	70.51	66.24	0.0	74.36	68.79	600.0	–	–	–	–	–	–	–	–	–	75.00	**69.94**	0.0	–	–	–
Banknote A.	94.38	93.82	0.0	97.47	**96.15**	600.0	93.25	93.08	36.0	93.37	92.54	18.0	93.37	92.54	0.0	93.37	92.54	0.0	93.37	92.59	14.0
Biodeg	83.92	78.03	0.0	82.76	79.85	600.0	79.58	76.04	604.0	82.02	79.17	599.0	83.54	80.08	2.0	83.54	80.08	1.0	82.23	**80.53**	128.0
Car	80.61	**80.40**	0.0	80.94	79.21	600.0	–	–	–	–	–	–	–	–	–	81.57	79.82	0.0	–	–	–
Credit A.	87.28	84.89	0.0	88.63	85.24	600.0	86.71	85.67	602.0	89.20	85.49	598.0	89.61	**86.10**	1.0	89.57	85.74	0.0	86.71	85.37	0.0
Hepatitis	91.48	**83.76**	0.0	92.59	80.51	600.0	82.54	82.69	521.0	91.03	81.54	599.0	91.55	82.05	1.0	91.38	81.54	0.0	79.31	79.49	0.0
Ionosphere	93.03	88.89	0.0	93.00	87.95	600.0	87.55	83.24	602.0	91.56	86.82	598.0	93.16	**89.32**	6.0	93.08	88.64	2.0	82.66	77.73	29.0
Iris	97.22	**95.61**	0.0	92.86	88.95	110.0	–	–	–	–	–	–	–	–	–	98.22	93.68	1.0	–	–	–
Mammo. M.	84.87	83.23	0.0	85.34	**83.85**	600.0	84.45	83.65	602.0	85.31	83.65	599.0	85.31	83.56	1.0	85.27	83.46	0.0	84.21	82.79	9.0
Monk1	78.18	81.14	0.0	90.22	**86.33**	524.0	90.41	85.07	186.0	90.46	85.32	5.0	90.46	85.18	0.0	90.46	85.18	0.0	90.46	85.18	128.0
Monk2	66.12	63.50	0.0	68.04	59.34	600.0	65.78	**65.56**	601.0	68.27	59.34	600.0	68.62	57.75	0.0	68.40	58.67	0.0	65.78	**65.56**	0.0
Monk3	98.85	**99.12**	0.0	98.89	98.99	83.0	98.23	97.60	27.0	98.89	98.99	1.0	98.89	98.99	0.0	98.89	98.99	0.0	97.16	96.12	0.0
Pima I. D.	79.21	72.69	0.0	80.45	73.44	600.0	77.03	74.13	601.0	78.06	71.56	599.0	78.33	70.52	0.0	78.33	70.52	0.0	76.70	**74.58**	128.0
Post O. P.	76.24	71.21	0.0	81.54	**72.73**	600.0	–	–	–	–	–	–	–	–	–	84.92	66.36	0.0	–	–	–
Seismic	93.52	93.14	0.0	93.73	93.16	600.0	93.41	**93.45**	604.0	93.47	93.22	600.0	93.47	93.28	0.0	93.47	93.19	0.0	93.42	93.44	0.0
Spambase	89.04	**87.63**	0.0	85.54	85.09	601.0	83.76	83.67	607.0	83.77	83.30	600.0	84.22	83.75	0.0	84.22	83.76	0.0	84.22	83.84	128.0
Spect H.	81.33	75.46	0.0	82.10	77.91	600.0	81.67	75.62	29.0	82.20	75.52	255.0	82.20	**79.10**	0.0	82.20	**79.10**	0.0	80.80	77.01	128.0
Thoracy S.	87.25	83.24	0.0	87.61	82.20	600.0	85.23	**84.75**	601.0	87.90	81.86	599.0	88.12	81.86	1.0	88.01	80.85	0.0	85.23	**84.75**	0.0
Tic T. T.	75.91	72.92	0.0	77.19	71.67	600.0	75.49	73.06	602.0	76.99	73.92	599.0	78.80	73.17	1.0	78.50	73.33	0.0	77.52	**74.17**	128.0
Wine	99.33	**93.09**	0.0	99.85	91.56	381.0	–	–	–	–	–	–	–	–	–	97.89	92.44	0.0	–	–	–

Maximum depth is set to 3. Train and test columns indicate, respectively, the mean of training and testing accuracy over five independent runs, whereas ctime columns refer to the computational time (in seconds). CART and BinOCT are run with continuous features, while others require binarisation of continuous features.

Table 4. A Benchmark for Depth-constrained Decision Tree Learners

	Top-down greedy			MIP						SAT			CP			Safe enumeration					
	CART			BinOCT			OST			MaxSAT_DT			Verhaeghe et al. [2019]			DL8.5			OSDT		
	[Breiman et al. 1984]			[Verwer and Zhang 2019]			[Aghaei et al. 2021]			[Hu et al. 2020]						[Aglin et al. 2020a]			[Hu et al. 2019]		
Dataset	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime
Balance S.	72.86	67.07	0.0	77.39	70.83	600.0	–	–	–	–	–	–	–	–	–	79.06	**72.49**	0.0	–	–	–
Banknote A.	97.24	95.92	0.0	97.71	**96.73**	600.0	93.37	93.29	603.0	94.81	94.40	599.0	94.83	94.52	1.0	94.83	94.58	0.0	93.99	93.24	10.0
Biodeg	87.16	78.07	0.0	82.17	78.33	601.0	81.07	77.08	608.0	84.02	79.17	598.0	87.03	**80.45**	171.0	86.78	80.38	26.0	82.28	78.79	10.0
Car	81.23	80.38	0.0	83.02	82.18	600.0	–	–	–	–	–	–	–	–	–	84.55	**82.82**	0.0	–	–	–
Credit A.	90.14	85.03	0.0	89.12	84.88	600.0	87.58	85.67	604.0	89.98	85.00	598.0	91.86	85.12	42.0	91.74	85.25	6.0	89.86	**85.73**	10.0
Hepatitis	94.54	84.33	0.0	96.90	79.49	600.0	82.54	**85.26**	601.0	95.86	78.97	598.0	97.76	78.46	4.0	97.76	78.46	1.0	79.31	79.49	0.0
Ionosphere	95.31	88.89	0.0	94.75	**89.32**	600.0	89.45	85.23	604.0	92.24	85.91	597.0	97.26	84.32	549.0	97.26	84.32	44.0	82.66	77.73	3.0
Iris	99.01	**95.61**	0.0	100.00	94.74	9.0	–	–	–	–	–	–	–	–	–	98.39	93.68	0.0	–	–	–
Mammo. M.	85.41	82.80	0.0	85.98	82.60	600.0	84.53	**83.65**	603.0	85.88	82.98	598.0	86.43	82.60	2.0	86.40	82.60	0.0	84.50	82.60	10.0
Monk1	84.01	83.29	0.0	100.00	**100.00**	62.0	100.00	**100.00**	169.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0
Monk2	68.59	64.97	0.0	71.51	57.35	600.0	65.78	**65.56**	602.0	69.69	61.72	599.0	72.71	57.75	1.0	72.58	58.15	0.0	65.78	**65.56**	0.0
Monk3	98.85	**99.12**	0.0	98.89	98.99	600.0	98.37	98.02	64.0	98.94	98.56	488.0	98.94	98.56	1.0	98.94	98.56	0.0	97.16	96.12	0.0
Pima I. D.	81.48	71.47	0.0	79.97	71.77	600.0	76.87	74.09	603.0	78.92	71.46	599.0	81.11	69.27	5.0	80.97	70.52	1.0	76.70	**74.58**	10.0
Post O. P.	79.32	**70.20**	0.0	85.23	64.55	600.0	–	–	–	–	–	–	–	–	–	91.69	65.45	0.0	–	–	–
Seismic	93.95	92.71	0.0	93.78	93.25	600.0	93.41	**93.46**	608.0	93.55	93.07	599.0	93.57	93.13	2.0	93.57	93.06	0.0	93.42	93.44	0.0
Spambase	91.16	**89.75**	0.0	81.50	81.48	603.0	84.58	83.91	613.0	84.28	83.20	599.0	85.50	84.40	2.0	85.50	84.40	0.0	81.83	81.22	10.0
Spect H.	83.78	77.11	0.0	85.70	**79.40**	600.0	81.12	76.49	195.0	86.20	77.01	599.0	86.90	77.01	3.0	86.90	77.01	0.0	84.20	76.72	10.0
Thoracy S.	88.45	82.86	0.0	88.69	80.68	600.0	85.23	**84.75**	602.0	89.15	81.02	598.0	90.28	80.68	8.0	90.17	80.17	1.0	85.23	**84.75**	0.0
Tic T. T.	83.70	**82.31**	0.0	82.28	79.00	600.0	80.22	76.98	604.0	81.50	76.75	598.0	87.05	80.83	7.0	86.69	81.25	1.0	81.62	78.17	10.0
Wine	100.00	**92.10**	0.0	100.00	88.89	178.0	–	–	–	–	–	–	–	–	–	100.00	89.33	0.0	–	–	–

Maximum depth is set to 4. Train and test columns indicate, respectively, the mean of training and testing accuracy over five independent runs, whereas ctime columns refer to the computational time (in seconds). CART and BinOCT are run with continuous features, while others require binarisation of continuous features.

6.1 On the Weaknesses of Standard Top-down Induction Algorithms to Constrain Decision Trees

First, it is important to note that pruning techniques rarely produce more interpretable decision trees, even though they can reduce the complexity of the tree. In fact, pruning methods do not discover richer trees than traditional top-down greedy algorithms. In other words, pruned greedy top-down trees are unnatural and it is possible to find smaller and better (in terms of accuracy and interpretability) trees, even manually [Piltaver et al. 2016]. Second, according to what has been shown in Section 3, with traditional algorithms, there is a need to adapt each constraint to a new specific heuristic; for example, ranked version of information gain [Hu et al. 2012, 2010] for monotonicity constraint, information gain sensitivity for fairness [Kamiran et al. 2010], and so on. This is relatively inefficient when many constraints or properties must be guaranteed. Thus, further studies are required to propose flexible impurity measures that can integrate constraints more easily.

6.2 On the Optimality of the Learned Decision Trees under Constraints

Before discussing optimality of existing techniques, it may also be relevant to look back at their history. Regarding the techniques proposed by the literature and by looking at Table 1, we can note that the earliest methods [Quinlan and Rivest 1989; Núñez 1991; Garofalakis et al. 2000; Potharst and Feelders 2002] with constraint enforcement are top-down. A reason could be the fact that this period was extremely dominated by the greedy heuristic search for combinatorial problems. Bayesian [Buntine 1992; Chipman et al. 1998; Denison et al. 1998] and MIP methods Heidenberger [1996] also appeared early. Due to their computational cost, these methods have been less used than greedy methods. However, nowadays, the computing capabilities of machines and the efficiency of solvers have highly increased and methods that were deemed as inefficient because of their computation cost are now re-emerging. This is the case for MIP and Bayesian methods. By nature, greedy algorithms can rarely lead to optimal decision trees given a particular set of constraints. If correctly modelled, then MIP/CP/SAT approaches produce optimal decision trees given enough time. However, from what has been seen for current methods, there is still room for improvement in the modelling, since the majority of current CP/SAT methods require binary features and are limited to binary classification, whereas current MIP methods (such as Bertsimas and Dunn [2017]; Verwer and Zhang [2019]) make use of starting solutions to speed up the search. Added with the scalability problem, these issues remain current limitations of MIP/CP/SAT approaches.

Tables 3 and 4 show a benchmark of extensive experiments that we performed on current state-of-the-art depth-constrained[2] decision tree methods. We reported training, testing (75-25 train-test split percentage) and computational time over five independent runs (with 10 minutes as time limit for each run to prevent unlimited computations, in accordance with Verwer and Zhang [2019]). Cells of the table marked with "––" correspond to methods that do not work with multi-class classification. Methods are evaluated on well-known datasets (listed on the first column of the tables) from the UCI [Dua and Graff 2017] repository. Except CART and BinOCT (which are used with continuous features), all these methods have the strong requirement to work only with binary features, thus continuous features have to be discretised[3] [Hu et al. 2020; Verhaeghe et al. 2019]. It can be seen that the old top-down greedy CART method is still competitive in terms of predictive accuracy over MIP/SAT/CP and safe enumeration methods. However, regarding the training performance, it usually fails to provide optimal or near-to-optimal solutions, unlike MIP/SAT/CP and safe enumeration methods. It is also surprising that the performance improvement of MIP/SAT/CP and safe enumeration methods do not necessarily lead to good generalisation yet have a depth constraint, which is supposed to reduce the generalisation gap. This suggests that more inductive biases should be proposed on these methods, in order to further reduce their generalisation gap.

Regarding the computational time, it appears from the table that MIP and MaxSAT require more time to find (optimal) solutions (or at least to prove optimality) compared to CP, safe enumeration, and greedy methods. Indeed, CART uses a heuristic and its implementation in Scikit-learn is highly optimised with system calls using the C language. Regarding safe enumeration methods, it uses highly optimised C libraries, too. Therefore, it remains unclear whether this improvement of computational time comes from this use of C libraries or from a reduction of theoretical complexity. Therefore, we highlight that future work should be done to fairly analyse computational time in the lens of theoretical complexity evaluation, which is currently lacking in some extent.

[2]Without a constraint on the minimum number of instances on leaves.

[3]We used the KBinsDiscretizer from the Scikit-learn library, with three Bins.

Table 5. Methods, Their Number of Variables, and Their Number of Constraints

Methods	Number of variables/literals	Number of constraints/clauses
Bessiere et al. [2009]	$\mathcal{O}\left(S \times M\right)$	$\mathcal{O}\left(M \times S^2 \times N^2 + S \times M^2 + M \times S^3\right)$
Narodytska et al. [2018]	$\mathcal{O}\left(S^2 + M \times S\right)$	$\mathcal{O}\left(M \times S^2 + M \times N \times S\right)$
Avellaneda [2020]	$\mathcal{O}\left(2^K(M + N + C)\right)$	$\mathcal{O}\left(2^K(M^2 + N \times M + C\right)$
Bertsimas and Dunn [2017]	$\mathcal{O}\left(2^K(M + N + C)\right)$	$\mathcal{O}\left(2^K(C + N \times K + M)\right)$
Verwer and Zhang [2019]	$\mathcal{O}\left(2^K(M + C + \log(T_{\max})\right)$	$\mathcal{O}\left(N + 2^K(M \times T_{\text{all}} + C)\right)$
Aghaei et al. [2021]	$\mathcal{O}\left(2^K(N + M)\right)$	$\mathcal{O}\left(2^K(N + M)\right)$
Aghaei et al. [2019]	$\mathcal{O}\left(L(M \times S + N)\right)$	$\mathcal{O}\left(L(L \times N + M \times S)\right)$

N is the number of instances of the dataset, M is the number of features, T_{all} is the total number of splits, and $T_{\max}$ is the number of maximum split per feature in Verwer and Zhang [2019]. S is the number of nodes or the size of the tree, K is its depth, L is its number of leaves. Only the work in the last row tries to incorporate a novel constraint, while the rest of works aim to accelerate the learning.

6.3 On the Complexity of SAT/CP/MIP Formulations of the Optimal Decision Trees

Although being theoretically hard (NP-hard), the optimal decision tree problem under structure-level constraints is attracting researchers. Indeed, several works have been proposed as explained in Section 3.1 and work well in practice with a limited budget of computational time (usually minutes or hours). Table 5 compares several of these methods in terms of their number of variables/literals and number of constraints/clauses (that we have counted if they were not mentioned in the paper). The number of variables and number of constraints serve as a heuristic to assess which method is more attractive in terms of practical computational time and therefore time and space complexity. Among others, the formulation of Aghaei et al. [2019] on the last row of the table is the only method, among SAT/CP/MIP formulations, whose goal is not to speed up computational time, but rather to leverage global optimisation to enforce the fairness constraint. All other methods usually aim to speed up optimisation when learning optimal decision trees under structure-level constraints. The first three works are SAT-based methods. From the first row to the third, the number of literals or constraints is enhanced to improve computational time. This can also be noted for other methods. For example, the BinOCT formulation of Verwer and Zhang [2017] is an improved MIP version of the OCT formulation [Bertsimas and Dunn 2017] where the number of variables does not depend on the dataset size N.

As mentioned above, the number of variables and constraints are used here as simple heuristics to characterise the difficulty and scalability of a particular method. Therefore, future studies should be done to access whether this reduction in the number of variables and constraints of these methods leads to an improved computational complexity. This issue is also an open problem for safe enumeration methods. Among these methods, only Auer et al. [1995] accompanied their T2 algorithm with a polynomial time complexity for the class of decision tree of depth at most 2. Hence, in the future, newly proposed methods should also discuss their computational complexity rather than just comparing number of variables/constraints or reporting only the empirical computational time.

6.4 On the Interpretability, Trustworthiness, and Robustness of Decision Trees

According to what has been related previously in the literature (e.g., Bertsimas and Dunn [2017]; Verwer and Zhang [2017] in Section 3.1), the question of interpretability is generally related to the complexity of the decision tree. Thus, when authors talk about forcing trees to be more interpretable and easier to understand, they commonly think about reducing complexity, i.e., structure-level constraints [Bertsimas and Dunn 2017; Verwer and Zhang 2017, 2019]. However, decision

trees that are learned with a maximal depth of two can be too simple in certain contexts, such as in the medical domain. That is to say, domain experts or users will not trust the learned tree, since its decision rule is too simple [Freitas 2014]. Therefore, this conducts to a loss of interpretability, because the domain expert will consider "an incompleteness criterion of models" [Guidotti et al. 2018]. Thus, similarly to the fact that increasing the size, depth, and number of leaf nodes of decision trees may lead to a loss of interpretability, decision trees with a very small size and depth, even accurate, are likely to not be accepted because of their "over-simplistic explanations" [Freitas 2014].

Hence, trustworthy decision trees with domain knowledge constraints and user-defined constraints are needed to increase the level of interpretability of the decision tree. Despite the work of López-Vallverdú et al. [2012, 2007] and Núñez [1991] to learn more comprehensible and trustworthy decision trees, this direction is not yet sufficiently studied in the literature. In fact, further away than domain knowledge constraints, trustworthy decision trees also include decision trees with ethical guarantees like the fairness constraint. More work needs to be done on that topic. Furthermore, beyond complexity, there is a large gap in the general direction of the possibility to impose multiple constraints, although Aghaei et al. [2019] and Nanfack et al. [2021a] recently showed how to learn (optimal) decision tree under fairness constraints.

Robustness is another particular issue in machine learning and thus in decision trees. Recent challenges in machine learning have shown a stream of interest in making machine learning classifiers robust to adversarial examples. Although this is very frequent in deep learning, it has also been shown that those adversarial examples can be transferred to any classifier and thus to a decision tree [Papernot et al. 2016]. Furthermore, because adversarial examples can be generated via constraints [Biggio and Roli 2018; Kantchelian et al. 2016], improving the robustness of classifiers can also be done via constraint enforcement [Bastani et al. 2016]. And yet, there are only very few works, if not only one [Chen et al. 2019], on constraining the decision tree to be more robust to particular attacks of adversarial examples.

6.5 On the Usefulness of Constraint Enforcement for Approximating Black-box Machine Learning Algorithms

Several works presented in Section 3.1.1 get benefit from constraint enforcement to approximate and explain a black-box machine learning algorithm. In fact, explaining black-box machine learning is (obviously) necessary from an ethical perspective and for preventing a "black-box society" [Guidotti et al. 2018; Pasquale 2015] guided by senseless decisions of algorithms. Additionally, it would be useless to learn a decision tree that is difficult to understand if its purpose is to explain a black-box model. Works such as Boz [2002], Craven and Shavlik [1995], Yang et al. [2018], and Zilke et al. [2016] have been proposed to explain black-box models. They can guarantee clear explanations by constraining the tree to be small and shallow. Of course, constraining the size of trees can affect the level of comprehensibility of the decision tree for experts helping them to understand how the initial model works. However, restraining too much the size of the trees may have consequences on their prediction performances thus being so different from the initial model that they would become useless. It is, therefore, necessary to find a good balance between the tasks of providing clear explanations and getting closer to the initial model. In this direction, there is still a significant gap in the literature to provide theoretical guarantees on the fidelity of the interpretation of the model explanations.

6.6 Future Prospects

In addition to previously mentioned gaps and research directions that need to be explored, this section presents other relevant future directions where research should be conducted.

6.6.1 Tree Balance Constraint, Interpretability of Decision Trees. Regarding structure-level constraints, several open issues are identified. The first issue is related to the capability to impose the balance constraint on the structure of the decision tree. The second one targets the tradeoff between the small size, the depth, or the number of leaf nodes in the tree and the question of the degree of interpretability of the decision tree. In fact, forcing the smallness/sparsity of a tree can improve its interpretability, but when it becomes too small, even remaining very accurate, its interpretability can decrease, since learned rules may become unreliable w.r.t. the domain expertise. Third, it is important to look for flexible optimal decision tree formulations with fewer restrictions (categorical variables as well as real variables, binary as well as non-binary trees, classification as well as regression trees) and that require less computation time. Therefore, a good modelling for interpretable decision trees should be much more flexible in terms of binarity, type of variables, tasks, depth, size, number of leaf nodes to be closer to the specifications of the user and the domain experts.

6.6.2 Experimental Settings. Regarding attribute-level and instance-level constraints, experimental evaluations are relatively limited, because there do not exist enough datasets that provide domain knowledge as constraints, except for attribute costs (as mentioned in Section 3.2.6). Hence, more datasets should be proposed to benchmark algorithms.

6.6.3 Flexible Impurity Measures. In real-world applications, top-down greedy algorithms are frequently used, but we have previously mentioned the limitations of such approaches (see Section 4.1). In fact, the well-known impurity measures that are based on entropy and the Gini index do not easily integrate constraints. Even if pruning methods attempt to make the learned tree to satisfy the constraints, the final tree may seriously lose its performance. Therefore, more flexible heuristics that make it easier to incorporate constraints should be proposed.

6.6.4 Domain Knowledge Constraints (feature and instance-level) for Constraint Programming and Bayesian Formulations. Table 1 shows that few works have used probabilistic formulations to enforce attribute and instance-level constraints, which is still an open issue. Yet, Bayesian formulations have the advantage of learning decision trees with a global objective or, in certain cases, a global heuristic. This would allow to soundly express global constraints such as instance-level constraints. In Bayesian methods, one can enforce constraints in a clear mathematical way through priors. In this direction, Angelopoulos and Cussens [2005b] use informative priors to allow "box" constraints (which can be seen as rules constraints in attribute-level). Their formulation performs well on a Bayesian predictive model with an ensemble of trees, not for a single tree. However, Nijssen [2008] makes it possible to learn a single accurate tree with his Bayesian formulation, but without investigating how to integrate informative priors. Further studies should be done to allow informative priors with domain knowledge constraints, such as rules in the Bayesian formulations, to learn a single accurate and trustworthy decision tree.

Beyond Bayesian formulations, linear and constraint programming formulations are also used to learn decision trees directly with a global objective. Thus, this approach is well suited to integrate attribute and instance-level constraints at a global level (for instance, attribute costs, hierarchy, fairness). Such constraints are not trivial to implement with e.g., greedy methods [Struyf and Džeroski 2006]. However, the majority of the proposed formulations have focused their attention on enforcing structure-level constraints, such as imposing the tree to be small or shallow. Thus, more efforts should be done to propose flexible linear programming approaches, which can integrate various types of constraints, including hierarchy and rule constraints that are important for the trustworthiness of decision trees for critical domains.

6.6.5 Constraints for Proxy Models. In the motivational part of the survey (Section 2.2), we emphasised the necessity to impose constraints on decision trees used as a proxy for black-box models. As a reminder, a proxy decision tree that approximates and explains a black-box model can be too small and not be able to approximate the black-box model; or it can be so deep that it is not able to provide human-understandable explanations. According to the literature that has been examined throughout this article, it is clear that it is still an open issue for future research, because the learned trees might be unreliable and untrustworthy. Future works on decision trees as proxy models should get benefit from constraint enforcement on decision trees in general, to ensure that the approximating tree of a black-box model meets the same guarantees as the approximated model. In particular, further studies should enforce fairness constraints on decision trees as a proxy if the black-box model has to guarantee fair decisions.

7 CONCLUSION

This survey underlines the importance of constraint enforcement on decision trees, for instance to meet a specified level of interpretability or trustworthiness. We present a taxonomy that comprises structure-level constraints, attribute-level constraints, and instance-level constraints. Our findings reveal that a large part of methods that enforce structure-level constraints (i.e., through the number of leaf nodes, depth, and size) aim to improve the accuracy and interpretability of decision trees. The two other levels of constraints usually aim to guarantee that decision trees comply with the requirements of a particular domain, often provided by domain experts. For example, it may be necessary to enforce monotonicity for the predictions with respect to some attributes. This makes decision trees more reliable and predictions appear as more realistic and similar to those that would be made by humans in the same context.

Historically, top-down greedy algorithms such as CART and C4.5 have been prominent in early developments for decision tree induction. Quite naturally, they are therefore widely used to learn decision trees under constraints. Top-down greedy induction approaches reported in this work enforce constraints by applying pruning methods or with specifically modified heuristics. Despite being the most popular approach, top-down greedy induction methods usually produce sub-optimal solutions for the training performance, which is not the case of other approaches that we have identified. Linear programming, constraint programming, and Bayesian formulations provide optimal solutions when they are given enough time to explore solutions, but they are more computationally demanding. With recent improvements in implementations and computational power, these methods are gaining interest. Despite being able to learn more accurate trees than their top-down greedy counterpart, they do not necessarily easily integrate a broad class of constraints, leaving room for improvement.

We suggest that further research should be intensified in the following areas: First, providing humanly understandable explanations of black-box machine learning models should be performed with theoretical guarantees (regarding both the predictions and the constraints already satisfied in the black-box model). Second, research is also needed to study possible compatibility between different types of constraints. Third, our study encourages the scientific community to propose more datasets with domain knowledge to systematically validate methods without the need of experts.

ACKNOWLEDGMENTS

The authors thank Hendrik Blockeel for his fruitful comments and suggestions to finalise the survey. Thanks are also due to Adrien Bibal and Minh Vu Viet for their valuable comments and discussions. Authors also thank anonymous reviewers for their arguments, propositions, and additional references.

REFERENCES

Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. 2019. Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI Press, 1418–1426.

Sina Aghaei, Andrés Gómez, and Phebe Vayanos. 2021. Strong optimal classification trees. *arXiv preprint arXiv:2103.15965* (2021).

Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. 2020a. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, the 32nd Innovative Applications of Artificial Intelligence Conference, the 10th AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 3146–3153.

Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. 2020b. PyDL8.5: A library for learning optimal decision trees. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 5222–5224.

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2018. Learning certifiably optimal rule lists for categorical data. *J. Mach. Learn. Res.* 18, 234 (2018), 1–78.

Nicos Angelopoulos and James Cussens. 2005b. Exploiting informative priors for Bayesian classification and regression trees. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 641–646.

Nicos Angelopoulos and James Cussens. 2005a. Tempering for Bayesian C&RT. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. ACM Press, New York, NY, 17–24.

Nicos Angelopoulos and James Cussens. 2016. BIMS: Bayesian inference of model structure. Retrieved from http://stoics.org.uk/~nicos/sware/bims/.

Peter Auer, Robert C. Holte, and Wolfgang Maass. 1995. Theory and applications of agnostic PAC-learning with small decision trees. In *Proceedings of the 12th International Conference on International Conference on Machine Learning (ICML'95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 21–29.

Florent Avellaneda. 2020. Efficient inference of optimal decision trees. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 8.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* 58 (2020), 82–115.

R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas. 2012. A survey of evolutionary algorithms for decision-tree induction. *IEEE Trans. Syst., Man, Cyber., Part C (Applic. Rev.)* 42, 3 (May 2012), 291–312.

Rodrigo C. Barros, André C. P. L. F. De Carvalho, Alex A. Freitas et al. 2015. *Automatic Design of Decision-tree Induction Algorithms*. Springer-Verlag.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. 2016. Measuring neural net robustness with constraints. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., 2621–2629.

Arie Ben-David. 1995. Monotonicity maintenance in information-theoretic machine learning algorithms. *Mach. Learn.* 19, 1 (01 Apr. 1995), 29–43.

Kristin P. Bennett and Jennifer A. Blue. 1996. *Optimal Decision Trees*. Technical Report. R.P.I. Math Report No. 214, Rensselaer Polytechnic Institute.

K. P. Bennett and J. A. Blue. 1998. A support vector machine approach to decision trees. In *Proceedings of the IEEE International Joint Conference on Neural Networks Proceedings*, Vol. 3. IEEE, 2396–2401.

Kristin P. Bennett and Olvi L. Mangasarian. 1992. Robust linear programming discrimination of two linearly inseparable sets. *Optim. Meth. Softw.* 1, 1 (1992), 23–34.

Dimitris Bertsimas and Jack Dunn. 2017. Optimal classification trees. *Mach. Learn.* 106, 7 (01 July 2017), 1039–1082.

Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. 2009. Minimising decision tree size as combinatorial optimisation. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP'09)*. Springer-Verlag, Berlin, 173–187.

Adrien Bibal and Benoît Frénay. 2016. Interpretability of machine learning models and representations: An introduction. In *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN'16)*. 77–82.

Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recog.* 84 (2018), 317–331.

Hendrik Blockeel, Luc De Raedt, and Jan Ramon. 1998. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 55–63.

Olcay Boz. 2002. Extracting decision trees from trained neural networks. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*. ACM Press, New York, NY, 456–461.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. 1984. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Harry Buhrman and Ronald De Wolf. 2002. Complexity measures and decision tree complexity: A survey. *Theor. Comput. Sci.* 288, 1 (2002), 21–43.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: Experiences from the Scikit-learn project. In *Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

Wray Buntine. 1992. Learning classification trees. *Statist. Comput.* 2, 2 (1992), 63–73.

Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. 2019. Adversarial training of gradient-boosted decision trees. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2429–2432.

Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. 2020. Treant: Training evasion-aware decision trees. *Data Mining. Knowl. Discov.* 34, 5 (2020), 1390–1420.

Jaime S. Cardoso and Ricardo Sousa. 2010. Classification models with global constraints for ordinal data. In *Proceedings of the 9th International Conference on Machine Learning and Applications (ICMLA'10)*. IEEE Computer Society, Washington, DC, 71–77.

Hongge Chen. 2019. Robust Decision Trees Against Adversarial Examples. Retrieved from https://github.com/chenhongge/RobustTrees.

Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. 2019. Robust decision trees against adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 1122–1131.

Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2019. Query-efficient hard-label black-box attack: An optimization-based approach. In *Proceedings of the International Conference on Learning Representation (ICLR'19)*.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. 1998. Bayesian CART model search. *J. Amer. Statist. Assoc.* 93, 443 (1998), 935–948.

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., 3512–3520.

Mark W. Craven and Jude W. Shavlik. 1995. Extracting tree-structured representations of trained networks. In *Proceedings of the 8th International Conference on Neural Information Processing Systems (NIPS'95)*. The MIT Press, Cambridge, MA, 24–30.

H. A. M. Daniëls and M. V. Velikova. 2003. *Derivation of Monotone Decision Models from Non-monotone Data*. Technical Report. Tilburg University, Center for Economic Research.

Gautam Das and Michael T. Goodrich. 1997. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Comput. Geom.* 8, 3 (1997), 123–137.

Jason V. Davis, Jungwoo Ha, Christopher J. Rossbach, Hany E. Ramadan, and Emmett Witchel. 2006. Cost-sensitive decision tree learning for forensic classification. In *Proceedings of the 17th European Conference on Machine Learning (ECML'06)*. Springer-Verlag, Berlin, 622–629.

David G. T. Denison, Bani K. Mallick, and Adrian F. M. Smith. 1998. A Bayesian CART algorithm. *Biometrika* 85, 2 (1998), 363–377.

Wenliang Du and Zhijun Zhan. 2002. Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining (CRPIT'14)*. Australian Computer Society, Inc., 1–8.

Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.

Gintare Karolina Dziugaite, Shai Ben-David, and Daniel M. Roy. 2020. Enforcing interpretability and its statistical impacts: Trade-offs between accuracy and interpretability. *arXiv preprint arXiv:2010.13764* abs/2010.13764 (2020).

Saher Esmeir and Shaul Markovitch. 2004. Lookahead-based algorithms for anytime induction of decision trees. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*. ACM Press, New York, NY.

Saher Esmeir and Shaul Markovitch. 2006. Any time induction of decision trees: an iterative improvement approach. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*. AAAI Press, Boston, Massachusetts, 348–355.

V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramirez-Quintana. 2002. Re-designing cost-sensitive decision tree learning. In *Proceedings of the Workshop de Mineria de Datos y Aprendizaje*. 33–42.

Ad Feelders and Martijn Pardoel. 2003. Pruning for monotone classification trees. In *Proceedings of the 5th International Symposium on Intelligent Data Analysis (IDA'03)*. Springer-Verlag, Berlin, 1–12.

César Ferri, Peter A. Flach, and José Hernández-Orallo. 2002. Learning decision trees using the area under the ROC curve. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 139–146.

Murat Firat, Guillaume Crognier, Adriana F. Gabor, C. A. J. Hurkens, and Yingqian Zhang. 2020. Column generation based heuristic for learning classification trees. *Comput. Oper. Res.* 116 (Apr. 2020).

Jack Fitzsimons, AbdulRahman Al Ali, Michael Osborne, and Stephen Roberts. 2019. A general framework for fair regression. *Entropy* 21, 8 (2019), 741.

Sam Fletcher and Md. Zahidul Islam. 2019. Decision tree classification with differential privacy: A survey. *ACM Comput. Surv.* 52, 4 (Aug. 2019).

Alberto Freitas, Altamiro Costa-Pereira, and Pavel Brazdil. 2007. Cost-sensitive decision trees applied to medical data. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'07)*. Springer-Verlag, Berlin, 303–312.

Alex A. Freitas. 2014. Comprehensible classification models: A position paper. *SIGKDD Explor. Newslett.* 15, 1 (Mar. 2014), 1–10.

Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. ACM Press, New York, NY, 493–502.

Arik Friedman, Assaf Schuster, and Ran Wolff. 2006. K-anonymous decision tree induction. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD'06)*. Springer-Verlag, Berlin, 151–162.

Élisa Fromont, Hendrik Blockeel, and Jan Struyf. 2007. Integrating decision tree learning into inductive databases. In *Proceedings of the 5th International Conference on Knowledge Discovery in Inductive Databases (KDID'06)*. Springer-Verlag, Berlin, 81–96.

Alka Gangrade and Ravindra Patel. 2009. Building privacy-preserving C4. 5 decision tree classifier on multi-parties. *Int. J. Comput. Sci. Eng.* 1, 3 (2009), 199–205.

Alka Gangrade and Ravindra Patel. 2012. Privacy preserving two-layer decision tree classifier for multiparty databases. *Int. J. Comput. Inf. Technol.* 1, 1 (2012), 77–82.

Minos Garofalakis, Dongjoon Hyun, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for constructing decision trees with constraints. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*. ACM Press, New York, NY, 335–339.

Minos Garofalakis, Dongjoon Hyun, Rajeev Rastogi, and Kyuseok Shim. 2003. Building decision trees with constraints. *Data Mining Knowl. Discov.* 7, 2 (Apr. 2003), 187–214.

Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. 1999. BOAT—Optimistic decision tree construction. *SIGMOD Rec.* 28, 2 (June 1999), 169–180.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of the IEEE 5th International Conference on Data Science and Advanced Analytics (DAAS'18)*. IEEE, 80–89.

Fred W. Glover and Manuel Laguna. 1998. *Tabu Search*. Springer Science & Business Media.

Sergio González. 2019. MonDT - Decision Trees for Classification with Monotonicity Constraints. Retrieved from https://github.com/sergiogvz/MonDT.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *Comput. Surv.* 51, 5 (Aug. 2018).

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* 11, 1 (2009), 10–18.

Kurt Heidenberger. 1996. Dynamic project selection and funding under risk: A decision tree based MILP approach. *Eur. J. Oper. Res.* 95, 2 (1996), 284–298.

Hao Hu, Mohamed Siala, Emmanuel Hébrard, and Marie-José Huguet. 2020. Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*.

Qinghua Hu, Xunjian Che, Lei Zhang, David Zhang, Maozu Guo, and Daren Yu. 2012. Rank entropy-based decision trees for monotonic classification. *IEEE Trans. Knowl. Data Eng.* 24, 11 (2012), 2052–2064.

Qing Hua Hu, Mao Zu Guo, Da Ren Yu, and Jin Fu Liu. 2010. Information entropy for ordinal classification. *Sci. China, Series F: Inf. Sci.* 53, 6 (2010), 1188–1200.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. 2019. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 7265–7273.

Laurent Hyafil and Ronald L. Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.* 5, 1 (1976), 15–17.

Md. Ridwan Al Iqbal, Mohammad Saiedur Rahaman, and Syed Irfan Nabil. 2012. Construction of decision trees by using feature importance value for improved learning performance. In *Proceedings of the 19th International Conference on Neural Information Processing (ICONIP'12)*. Springer-Verlag, Berlin, 242–249.

Mohammad Kachuee, Kimmo Karkkainen, Orpaz Goldstein, Davina Zamanzadeh, and Majid Sarrafzadeh. 2019. Nutrition and health data for cost-sensitive learning. *CoRR* (2019). arXiv:1902.07102.

Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. 2010. Discrimination aware decision tree learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'10)*. IEEE Computer Society, Washington, DC, 869–874.

Rémon Kamp, Ad Feelders, and Nicola Barile. 2009. Isotonic classification trees. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis (IDA'09)*. Springer-Verlag, Berlin, 405–416.

Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. 2016. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)*. JMLR Press, New York, NY, 2387–2396.

Dragi Kocev, Jan Struyf, and Sašo Džeroski. 2007. Beam search induction and similarity constraints for predictive clustering trees. In *Proceedings of the 5th International Conference on Knowledge Discovery in Inductive Databases (KDID'06)*. Springer-Verlag, Berlin, 134–151.

Marek Krętowski and Marek Grześ. 2006. Evolutionary induction of cost-sensitive decision trees. In *Proceedings of the 16th International Conference on Foundations of Intelligent Systems (ISMIS'06)*. Springer-Verlag, Berlin, 121–126.

Christopher Kruegel and Thomas Toth. 2003. Using decision trees to improve signature-based intrusion detection. In *6th International Symposium on Recent Advances in Intrusion Detection (RAID'03)*. Springer-Verlag, Berlin, 173–191.

Qinbin Li, Zeyi Wen, and Bingsheng He. 2020a. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 4642–4649.

Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. 2020b. Privacy-preserving gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 784–791.

Xiangju Li, Hong Zhao, and William Zhu. 2015. A cost sensitive decision tree algorithm with two adaptive mechanisms. *Knowl.-based Syst.* 88 (2015), 24–33.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. 2020. Generalized and scalable optimal sparse decision trees. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 6150–6160.

Charles X. Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. 2004. Decision trees with minimal costs. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*. ACM Press, New York, NY, 69–77.

Li Liu, Murat Kantarcioglu, and Bhavani Thuraisingham. 2009. Privacy preserving decision tree mining from perturbed data. In *Proceedings of the 42nd Hawaii International Conference on System Sciences*. IEEE, 1–10.

Susan Lomax and Sunil Vadera. 2013. A survey of cost-sensitive decision tree induction algorithms. *Comput. Surv.* 45, 2 (Mar. 2013), 16:1–16:35.

Joan Albert López-Vallverdú, David Riaño, and John A. Bohada. 2012. Improving medical decision trees by combining relevant health-care criteria. *Expert Syst. Appl.* 39, 14 (Oct. 2012), 11782–11791.

Joan Albert López-Vallverdú, David Riaño, and Antoni Collado. 2007. Increasing acceptability of decision trees with domain attributes partial orders. In *Proceedings of the 20th IEEE International Symposium on Computer-based Medical Systems (CBMS'07)*. IEEE Computer Society, Washington, DC, 569–574.

Marco Lorenzi and Maurizio Filippone. 2018. Constraining the dynamics of deep probabilistic models. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3227–3236.

Gjorgji Madzarov, Dejan Gjorgjevikj, and Ivan Chorbev. 2009. A multi-class SVM classifier utilizing binary decision tree. *Informatica* 33, 2 (May 2009), 233–241.

Christophe Marsala and Davide Petturiti. 2015. Rank discrimination measures for enforcing monotonicity in decision tree induction. *Inf. Sci.* 291, C (Jan. 2015), 143–171.

David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. 2011. Performance of classification models from a user perspective. *Decis. Supp. Syst.* 51, 4 (2011), 782–793.

Stan Matwin, Amy Felty, István Hernádvölgyi, and Venanzio Capretta. 2005. Privacy in data mining using formal methods. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications (TLCA'05)*. Springer-Verlag, Berlin, 278–292.

Matt Menickelly, Oktay Günlük, Jayant Kalagnanam, and Katya Scheinberg. 2016. Optimal generalized decision trees via integer programming. *CoRR* (2016). arXiv:1612.03225.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Ryan Murray and Kaspar Thommen. 2019. A Bayesian Decision Tree Algorithm. Retrieved from https://github.com/UBS-IB/bayesian_tree.

Geraldin Nanfack, Valentin Delchevalerie, and Benoît Frénay. 2021a. Boundary-based fairness constraints in decision trees and random forests. In *Proceedings of the 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'21)*.

Geraldin Nanfack, Paul Temple, and Benoît Frénay. 2021b. Global explanations with decision rules: A co-learning approach. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI'21)*.

Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva. 2018. Learning optimal decision trees with SAT. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. AAAI Press, 1362–1368.

Siegfried Nijssen. 2008. Bayes optimal classification for decision trees. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. ACM Press, New York, NY, 696–703.

Siegfried Nijssen and Elisa Fromont. 2007. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. ACM Press, New York, NY, 530–539.

Siegfried Nijssen and Elisa Fromont. 2010. Optimal constraint-based decision tree induction from itemset lattices. *Data Mining Knowl. Discov.* 21, 1 (July 2010), 9–51.

Mohammad Norouzi, Maxwell D. Collins, Matthew Johnson, David J. Fleet, and Pushmeet Kohli. 2015. Efficient non-greedy optimization of decision trees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*. The MIT Press, Cambridge, MA, 1729–1737.

Steven W. Norton. 1989. Generating better decision trees. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 800–805.

Marlon Núñez. 1991. The use of background knowledge in decision tree induction. *Mach. Learn.* 6, 3 (May 1991), 231–250.

Giuseppe Nuti, Lluís Antoni Jiménez Rugama, and Andreea-Ingrid Cross. 2019. Efficient Bayesian decision tree algorithm. *Corr* (2019). arXiv:1901.03214

Adam Omielan and Sunil Vadera. 2012. ECCO: A new evolutionary classifier with cost optimisation. In *Proceedings of the 7th IFIP TC International Conference (IIP'12)*. Springer-Verlag, Berlin, 97–105.

Meng Pan. 2019. Python Optimal Tree. Retrieved from https://pypi.org/project/pyoptree/.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *CoRR* (2016). arXiv:1605.07277.

Frank Pasquale. 2015. *The Black Box Society: The Secret Algorithms That Control Money and Information*. Harvard University Press, Cambridge, MA.

Michael J. Pazzani, Christopher J. Merz, Patrick M. Murphy, Kamal M. Ali, Timothy Hume, and Clifford Brunk. 1994. Reducing misclassification costs. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 217–225.

Shenglei Pei, Qinghua Hu, and Chao Chen. 2016. Multivariate decision trees with monotonicity constraints. *Knowl.-based Syst.* 112, C (Nov. 2016), 14–25.

Rok Piltaver, Mitja Luštrek, Matjaź Gams, and Sanda Martinčić-Ipšić. 2016. What makes classification trees comprehensible? *Expert Syst. Appl.* 62, C (Nov. 2016), 333–346.

R. Potharst and A. J. Feelders. 2002. Classification trees for problems with monotonicity constraints. *SIGKDD Explor. Newslett.* 4, 1 (June 2002), 1–10.

Chen Qiu, Liangxiao Jiang, and Chaoqun Li. 2017. Randomly selected decision tree for test-cost sensitive learning. *Appl. Soft Comput.* 53, C (Apr. 2017), 27–33.

J. R. Quinlan. 1986. Induction of decision trees. *Mach. Learn.* 1, 1 (Mar. 1986), 81–106.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

J. R. Quinlan and R. L. Rivest. 1989. Inferring decision trees using the minimum description length principle. *Inf. Comput.* 80, 3 (Mar. 1989), 227–248.

Edward Raff, Jared Sylvester, and Steven Mills. 2018. Fair forests: Regularized tree induction to minimize model bias. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES'18)*. ACM Press, New York, NY, 243–250.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. Model-agnostic interpretability of machine learning. In *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning (WHI'16)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM Press, New York, NY, 1135–1144.

Valentin Roşca. 2019. Decision Tree Constraints. Retrieved from https://pypi.org/project/DecisionTreeConstraints/.

S. Rasoul Safavian and David Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE Trans. Syst., Man, Cyber.* 21, 3 (May/June 1991), 660–674.

Makoto Sato and Hiroshi Tsukimoto. 2001. Rule extraction from neural networks via decision tree induction. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'01)*. IEEE, 1870–1875.

V. Schetinin, J. E. Fieldsend, D. Partridge, T. J. Coats, W. J. Krzanowski, R. M. Everson, T. C. Bailey, and A. Hernandez. 2007. Confident interpretation of Bayesian decision tree ensembles for clinical applications. *Trans. Info. Tech. Biomed.* 11, 3 (May 2007), 312–319.

I. K. Sethi and G. P. R. Sarvarayudu. 1982. Hierarchical classifier design using mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.* 4, 4 (Apr. 1982), 441–445.

Jan Struyf and Sašo Džeroski. 2006. Constraint based induction of multi-objective regression trees. In *Proceedings of the 4th International Conference on Knowledge Discovery in Inductive Databases (KDID'05)*. Springer-Verlag, Berlin, 222–233.

Jan Struyf and Sašo Džeroski. 2007. Clustering trees with instance level constraints. In *Proceedings of the 18th European Conference on Machine Learning (ECML'07)*. Springer-Verlag, Berlin, 359–370.

Jan Struyf, Bernard Ženko, Hendrik Blockeel, Celine Vens, Matej Petković, Tomaž Stepišnik P., Vanja Mileski, Martin Breskvar, Jurica Levatić, Dragi Kocev, and Sašo Džeroski. 2017. Clus. Retrieved from www.cs.kuleuven.be/~dtai/clus.

Latanya Sweeney. 2002. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-based Syst.* 10, 5 (Oct. 2002), 557–570.

Ming Tan. 1993. Cost-sensitive learning of classification knowledge and its applications in robotics. *Mach. Learn.* 13, 1 (Oct. 1993), 7–33.

Zhouxuan Teng and Wenliang Du. 2007. A hybrid multi-group privacy-preserving approach for building decision trees. In *Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'07)*. Springer-Verlag, Berlin, 296–307.

Christos Tjortjis and John Keane. 2002. T3: A classification algorithm for data mining. In *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'02)*. Springer-Verlag, Berlin, 50–55.

Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 465–474.

Pere Torres, David Riaño, and Joan Albert López-Vallverdú. 2011. Inducing decision trees from medical decision processes. In *Proceedings of the ECAI Workshop: Knowledge Representation for Health-Care:*. Springer-Verlag.

Peter D. Turney. 1995. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *J. Artif. Int. Res.* 2, 1 (Apr. 1995), 369–409.

Panagiotis Tzirakis and Christos Tjortjis. 2017. T3C: Improving a decision tree classification algorithm's interval splits on continuous attributes. *Adv. Data Anal. Classif.* 11, 2 (June 2017), 353–370.

Hina Vaghashia and Amit Ganatra. 2015. A survey: Privacy preservation techniques in data mining. *Int. J. Comput. Applic.* 119, 4 (2015), 20–26.

Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. 2008. Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data* 2, 3 (Oct. 2008).

Hélène Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. 2019. Learning optimal decision trees using constraint programming. In *Proceedings of the 25th International Conference on Principles and Practice of Constraint Programming (CP'19)*.

Sicco Verwer and Yingqian Zhang. 2017. Learning decision trees with flexible constraints and objectives using integer optimization. In *Proceedings of the 14th International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'17)*. Springer-Verlag, Cham, 94–103.

Sicco Verwer and Yingqian Zhang. 2019. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*. AAAI Press.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. 2001. Constrained K-means clustering with background knowledge. In *Proceedings of the 15th International Conference on Machine Learning (ICML'01)*. Morgan Kaufmann, 577–584.

Nan Wang, Jinbao Li, Yong Liu, Jinghua Zhu, Jiaxuan Su, and Cheng Peng. 2018. Accurate decision tree with cost constraints. In *Proceedings of the First International Conference on Advanced Hybrid Information Processing (ADHIP'17)*. Springer, 154–165.

Floryt van Wesel, Herbert Hoijtink, and Irene Klugkist. 2011. Choosing priors for constrained analysis of variance: Methods based on training data. *Scand. J. Statist.* 38, 4 (2011), 666–690.

Chia-Chi Wu, Yen-Liang Chen, Yi-Hung Liu, and Xiang-Yu Yang. 2016. Decision tree induction with a constrained number of leaf nodes. *Appl. Intell.* 45, 3 (Oct. 2016), 673–685.

Yuhong Wu, Håkon Tjelmeland, and Mike West. 2007. Bayesian CART: Prior specification and posterior simulation. *J. Comput. Graphic. Statist.* 16, 1 (2007), 44–66.

Chengliang Yang, Anand Rangarajan, and Sanjay Ranka. 2018. Global model interpretation via recursive partitioning. In *Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS'18)*. IEEE, 1563–1570.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya R. Gupta. 2017. Deep lattice networks and partial monotonic functions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 2985–2993.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P. Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 962–970.

A. N. K. Zaman, Charlie Obimbo, and Rozita A. Dara. 2016. A novel differential privacy approach that enhances classification accuracy. In *Proceedings of the 9th International C* Conference on Computer Science & Software Engineering (C3S2E'16)*. ACM, New York, NY, 79–84.

Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. 2016. DeepRED–Rule extraction from deep neural networks. In *Proceedings of the 19th International Conference on Discovery Science (DS'16)*. Springer-Verlag, Cham, 457–473.

Received October 2019; revised December 2021; accepted December 2021