Artificial Intelligence Review (2023) 56:4765–4800
https://doi.org/10.1007/s10462-022-10275-5

Recent advances in decision trees: an updated survey

Vinícius G. Costa[1] · Carlos E. Pedreira[1]

Published online: 10 October 2022

Abstract

Decision Trees (DTs) are predictive models in supervised learning, known not only for their unquestionable utility in a wide range of applications but also for their interpretability and robustness. Research on the subject is still going strong after almost 60 years since its original inception, and in the last decade, several researchers have tackled key matters in the field. Although many great surveys have been published in the past, there is a gap since none covers the last decade of the field as a whole. This paper proposes a review of the main recent advances in DT research, focusing on three major goals of a predictive learner: issues regarding the fitting of training data, generalization, and interpretability. Moreover, by organizing several topics that have been previously analyzed in isolation, this survey attempts to provide an overview of the field, its key concerns, and future trends, serving as a good entry point for both researchers and newcomers to the machine learning community.

Keywords Decision trees · Machine learning · Interpretable models · Classification algorithms

1 Introduction

In recent years, there has been a growing wealth of various data sources: social media, smartphones, Internet of Things (IoT) devices, health monitors, and many others. By leveraging this new resource, the *machine learning* field has grown rapidly, allowing an increasing number of intelligent applications to learn their functions directly from data instead of having them explicitly programmed. This has resulted in systems capable of controlling self-driving cars (Bojarski et al. 2016), recognizing speech (Amodei et al. 2016) and surpassing humans at the game Go (Silver et al. 2016), among others. Underlying these results is not only an abundance of available data, but also important advances in the learning models themselves (Sarker 2021).

✉ Carlos E. Pedreira
pedreira56@gmail.com

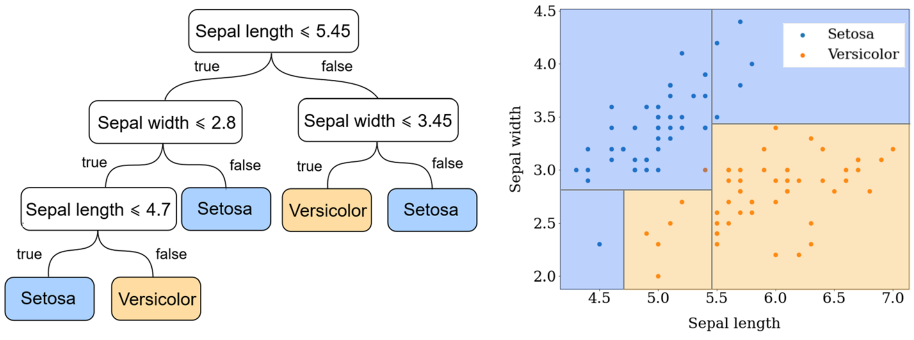
Vinícius G. Costa
vgcosta@cos.ufrj.br

[1] Systems Engineering and Computer Science Department, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil

One of the most popular learning models are Decision Trees (DTs). These models are usually represented in a flowchart-like structure, in which every internal node is a logical test (called a *split*), and every leaf is a prediction (see Fig. 1a). During inference, each observation starts at the root and ends in one of the leaves, following a path that is completely transparent to the user. The simplicity of this approach belies its robustness: DTs are known not only as an interpretable model that is easy to grasp, but also as an accurate method that has stood the test of time, ever since their original proposal in the 1960s (Morgan and Sonquist 1963). Trees present many other advantages as well, such as low computational cost, being able to deal with missing values, and out-of-the-box handling of mixed data (Hastie et al. 2009).

Although DTs have been widely-used for many decades, recently these models have attracted more attention than usual. With the growing ubiquity of machine learning and automated decision systems, there has been a rising interest in *explainable machine learning*: building models that can be, in some sense, understood by humans (Rudin 2019; Roscher et al. 2020). Since DTs are known for their interpretability, many researchers have turned their attention towards this field. The following uptick in interest, in addition to the usual flow of publishing, has resulted in a rich collection of contributions that may benefit from a broad review (see Fig. 2 for an illustration of this increasing interest in the field). Several great reviews have already been written on the subject, such as the ones by Murthy (1998), Rokach and Maimon (2007), Kotsiantis (2013) and Loh (2014), however, the last decade of research remains uncovered. More recent surveys do exist, but they focus on specific tree approaches, like fuzzy trees (Sosnowski and Gadomer 2019) and mathematical programming trees (Carrizosa et al. 2021), therefore not covering the field as a whole. In an attempt to fill this gap, this survey aims for two contributions: first, to review the recent work in DT research, and second, to position those works in relation to well-established previous developments in the field.

The platform Web of Science was used to locate current research papers on the topic of DTs. To further refine the results and guarantee both quality and relevance, the authors defined inclusion and exclusion criteria, such that a paper must satisfy all inclusion criteria and none of the exclusion criteria:



(a) Tree visualization **(b)** Partitioning visualization

Fig. 1 A classification tree induced on a reduced version of the Iris dataset

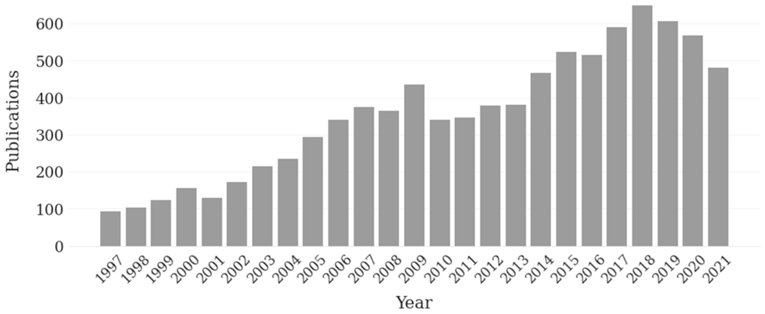


Fig. 2 Number of publications found through the search term "TI=(tree*) AND TS=(classif* OR regress* OR decision)", organized by year of publication. These results only consider the search term and a filter by venue (Computer Science or Engineering categories), therefore applications are also included. The graph was obtained through the Web of Science database

Inclusion criteria

- The paper proposes a method in the DT family.
- The paper was published within the last 10 years in a high-standard journal (as their findings provide the most relevant domain knowledge), or within the last 3 years in a well-reputed conference (as they present the latest advancements). Both sources were given the same relevance.

Exclusion criteria

- The paper was not written in English.
- The paper was not published in a venue related to Computer Science or Engineering.
- The paper only presents an application of a DT method.

Furthermore, certain subjects were not considered, such as fuzzy trees, trees for online learning, and ensembles of trees, because although these fields were clearly linked to DT research at first, they have since become sizeable subjects with their own sets of considerations and challenges. Additionally, throughout the process of writing the survey, two other types of papers were included: conference papers older than 3 years that were relevant enough to be frequently cited by other papers in the survey (e.g. Frosst and Hinton 2017), and papers that either directly continue or precede other papers in the survey (e.g. Demirović et al. 2021 directly expands on Aglin et al. 2020). At the end of this process, 83 recent papers were selected for discussion, in addition to several older seminal papers that were also selected to provide context.

Given the high number of relevant papers, the authors aimed to keep the survey concise by giving more detail to approaches that could be more complex to understand, while giving less detail to approaches that are either more straightforward or based on well-established methods. The level of detail given to a certain paper does not correlate to its quality or novelty, and interested readers are referred to the original papers for further mathematical and technical details.

It also must be noted that this survey does not empirically compare the cited methods. Such comparisons would be valuable since, to the best of the authors' knowledge, there is no large-scale study comparing different DT methods in a wide range of datasets. However, there are two great obstacles to such a study: first, the majority of papers on the topic do not make the code openly available nor provide implementations for the proposed methods, and second, the papers frequently use different datasets. These two issues make it very difficult to compare a large number of DT methods to each other, and it is the opinion of the authors that comparing only a small subset of these methods would lead to weaker and possibly misleading conclusions with regards to the superiority of a method or family of methods. As such, this survey refrains from empirical comparisons and focuses instead on comparing the approaches from a conceptual and algorithmic point of view, aiming to identify recent trends and research gaps while leaving comparative analyses to future studies.

Next, an outline is presented for the topics and key questions that support the structure of this survey (see also Fig. 3).

- *Approximation* (Sect. 3): Issues regarding the fitting of training data.
  - *Multivariate splits* (Sect. 3.1): Traditionally, the internal nodes of a tree are simple and involve a single variable. Is there any valuable gain in using more complex splits?
  - *Model trees* (Sect. 3.2): In regression tasks, each leaf in the tree corresponds to a constant number, and consequently the tree's outputs are both limited and discontinuous. However, most real-life regression tasks require more complex representations. Are there advantages in employing functions, instead of constants, at the leaves?
  - *Trees for data with specific characteristics* (Sect. 3.3): Traditional trees work well with most kinds of data. But what if the dataset is unlabeled, or if certain classes are more important than others? These additional considerations sometimes require a different approach.
  - *Non-greedy trees* (Sect. 3.4): Traditional DTs are constructed in a greedy way that is both efficient and easy to implement. Can one build better trees by using other optimization approaches?

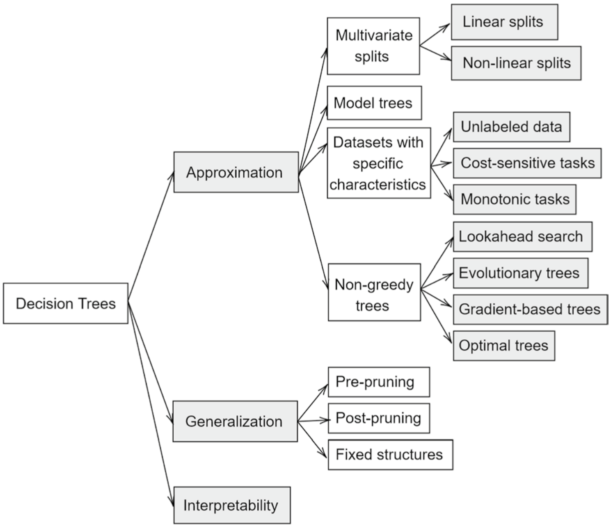


Fig. 3 Visual outline of the survey

- *Generalization* (Sect. 4): Left unattended, most tree construction algorithms will produce a large tree that overfits the training data.
  - *Pre-pruning* (Sect. 4.1): Should one interrupt the algorithm before the tree grows too large?
  - *Post-pruning* (Sect. 4.2): Should one reduce the final tree by eliminating some of its branches?
  - *Fixed structure* (Sect. 4.3): Should one fix the tree structure a priori, therefore eliminating growth altogether?
- *Interpretability* (Sect. 5): Indeed, DTs are often praised for their interpretability. But are they always interpretable? Which design decisions affect interpretability?

As can be seen from the outline, this survey is split into two levels. At the outer level, sections are organized by three major goals of a predictive learner: approximation, generalization, and interpretability. At the inner level, the survey is organized by different design decisions in DT algorithms, either regarding the tree structure (e.g. What are the splits? What are the leaves?) or the induction process (e.g. How is the tree constructed? How is it pruned?).

This organization was chosen to propose a view of the area and ease the flow of reading, but it is not without its problems. The outer levels sometimes overlap since some decisions contribute to more than one major goal (e.g. by reducing tree size, post-pruning enhances both generalization and interpretability). Furthermore, since many of the design decisions at the inner level are not mutually exclusive, they have occasionally been combined in interesting ways, which leads the survey to cite some works in multiple sections. In truth, it is difficult to provide a clear taxonomy for the field since many topics are interconnected, and it is noteworthy that Murthy (1998) identified similar difficulties in his seminal survey.

2 Basic concepts

Intending to make this text as self-contained as possible, this section offers a brief description of how traditional DT algorithms work. By "traditional", this survey refers to the greedy univariate trees built by CART (Classification and Regression Trees) (Breiman et al. 1984), which remains one of the most popular algorithms in the field, and also serves as the base for several other contributions. A DT of this kind is illustrated in Fig. 1a.

Regarding terminology, this survey denotes that the goal of predictive models is to navigate a *hypothesis set* (e.g. the set of possible trees) and select a *hypothesis* (e.g. a particular tree) that is close to the *target function* that one wants to predict. To this end, a *training dataset* consisting of *observations* is used, with each observation being composed of *attributes* (that admit numerical, categorical, or binary *values*) and a *target attribute* (a *label*, in classification tasks, or a constant number, in regression tasks). An ideal hypothesis not only *fits* the training data (i.e. approximates), but also has similar performance in out-of-sample data taken from the same distribution (i.e. generalizes).

Structurally, traditional DTs are composed of two elements: *internal nodes* and *leaves*. Each internal node contains a logical test, called a *split*, of the form "attribute ≤ value" (for numerical attributes) or "attribute = value" (for categorical attributes). Since every split has a binary outcome of true or false, every node has two outgoing branches and the

Table 1 Example dataset for a medical diagnosis task with three individuals, and attributes "temperature" and "age"

Temperature	Age	Label
39	58	Sick
40	19	Sick
36	50	Healthy

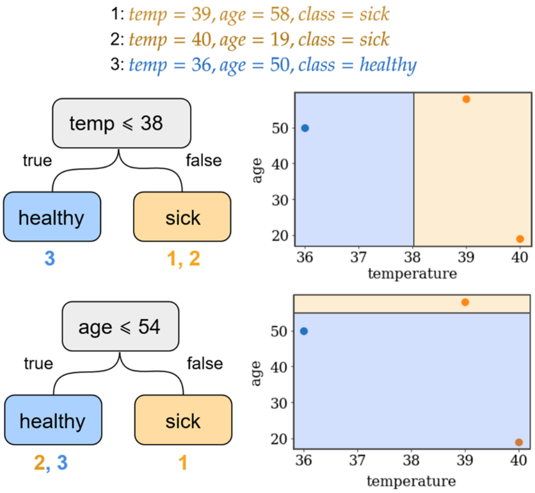


Fig. 4 Comparison between two candidate splits for a medical diagnosis example

tree itself is also binary. The leaves, on the other hand, contain either a label or a constant number, and they serve as the tree's possible outcomes, selected during the inference step through successive application of the splits.

The process of building DTs from training data is usually called *decision tree induction*, and to understand how it is traditionally done, it is useful to visualize a DT as a specific partitioning of the input space. This interpretation is more readily apparent with numerical attributes, and it is illustrated in Fig. 1b alongside the usual flowchart depiction. The root node splits the space into two disjoint partitions, which are passed along to its left and right child nodes. Each of these nodes further splits the partitions they received by applying their own tests and then passing them along to their children. This process is repeated until a leaf is reached, as the leaves correspond to the partitions themselves and contain the predictions associated with each region of the input space. In this context, the induction process can be seen as an optimization problem, in which the goal is to partition the input space in a way that correctly predicts the target function, while using as few partitions as possible.

Traditional algorithms solve this problem through a greedy approach that iteratively selects the most informative splits. To illustrate this notion of *informativeness*, consider a medical diagnosis example in which the attributes are "temperature" and "age", the prediction is either "healthy" or "sick", and the dataset is composed of the three individuals displayed in Table 1. In this case, the split "temperature $\leq$ 38" is more informative than the split "age $\leq$ 54", because the first divides the dataset perfectly into healthy and sick individuals, while the latter cannot explain why individuals 2 and 3 have different labels (see Fig. 4). In other words, the first split is more informative because it divides the dataset into *pure partitions* (i.e. every element has the same label), whereas the second split does not. This impurity is captured by a measure called *splitting criterion*, which for classification tasks is usually either entropy-based information gain (Quinlan 1986) or Gini impurity (Breiman et al. 1984).

From this measure, an induction algorithm follows naturally: the tree can be built one node at a time, in a greedy way that always selects the best split according to the splitting criterion. Because each node considers only the data that has been passed to it from its parent, the procedure fits well with a recursion, and it is aptly called *recursive partitioning*. For classification tasks, it can be summarized as follows:

```
(1) Let $X$ be a dataset with attributes $(x_1, \dots, x_n)$
(2) If $X$ is a pure partition with label $y$:
    (a) Create a leaf that predicts label $y$, and return.
(3) Otherwise:
    (a) For each attribute $x_i$:
        (i) For each possible cutoff point or value $c$:
            (a) Select the best candidate split $(x_i \leq c)$ or $(x_i = c)$ according to the splitting criterion.
            (b) Create an internal node with $(x_i, c)$ being the split.
            (c) Divide $X$ into two datasets $X_{\text{left}}$ and $X_{\text{right}}$, according to the split $(x_i, c)$.
            (d) Create a child node for each of the two newly-created datasets.
            (e) For each child node, execute step 1.
```

Since this procedure will not stop until all the leaves contain pure partitions, it is guaranteed to induce a tree that perfectly classifies the training data. This is great news for the fitting capacity of the model, as the tree will adapt to the complexity of the target function and grow as large as it is necessary to represent it. However, it is also terrible news concerning generalization: in the worst case, pure partitions can only be provided by an exponentially-large tree that has one leaf for every observation in the dataset, which is analogous to fitting N points using a polynomial function of the $(N-1)$th degree. For this reason, the procedure presented above usually results in overfitting, which is one of the reasons why DTs have been accused of having low bias and high variance (Breiman et al. 1984, Sect. 3.1). Fortunately, this weakness of DT algorithms (sometimes called *overpartitioning*) has been extensively studied, and indeed one of the reasons for CART's popularity is precisely the way by which it alleviates this problem: instead of stopping the procedure before the tree overfits (called *pre-pruning*), the algorithm lets the overfitting occur and then eliminates some of the branches from the tree (called *post-pruning*). These topics are explored in Sect. 4.

Furthermore, it can be seen that the above procedure contains some arbitrary decisions which could be reasonably modified. For instance, instead of numerical splits following the form $x_i \leq c$, they could involve a linear combination of attributes such as $w_i x_i + w_j x_j \leq c$. Or perhaps splits could be stochastic, instead of deterministic. Or, maybe, the stopping criterion could be minimum partition size, instead of pure partitions. The possibilities are huge, and in a certain sense, much of DT research can be understood as extensions of this framework. They are the topic of the rest of the survey.

3 Approximation

A fundamental topic in machine learning is approximation: the ability of a model to select a hypothesis that can fit a training dataset by minimizing the associated error. A model with low approximation capacity is unable to approximate complex target functions, regardless of the size of the training dataset (e.g., a linear model is unable to adequately approximate a sinusoidal target function). On the other hand, a model with an excessive high approximation capacity, for a given training set size, can end up overfitting the training data.

There are many aspects of DTs that impact their ability to approximate. The most immediate one is size: a tree with a single split (also called a *stump*) can only represent a single line that is orthogonal to its associated attribute; on the other hand, a tree with many nodes is capable of representing this single line and much more. Although growing larger trees has a positive effect concerning approximation, several other aspects may impact this ability. The following sections consider multivariate splits, model leaves, trees for datasets with particular data characteristics, and non-greedy trees.

3.1 Multivariate splits

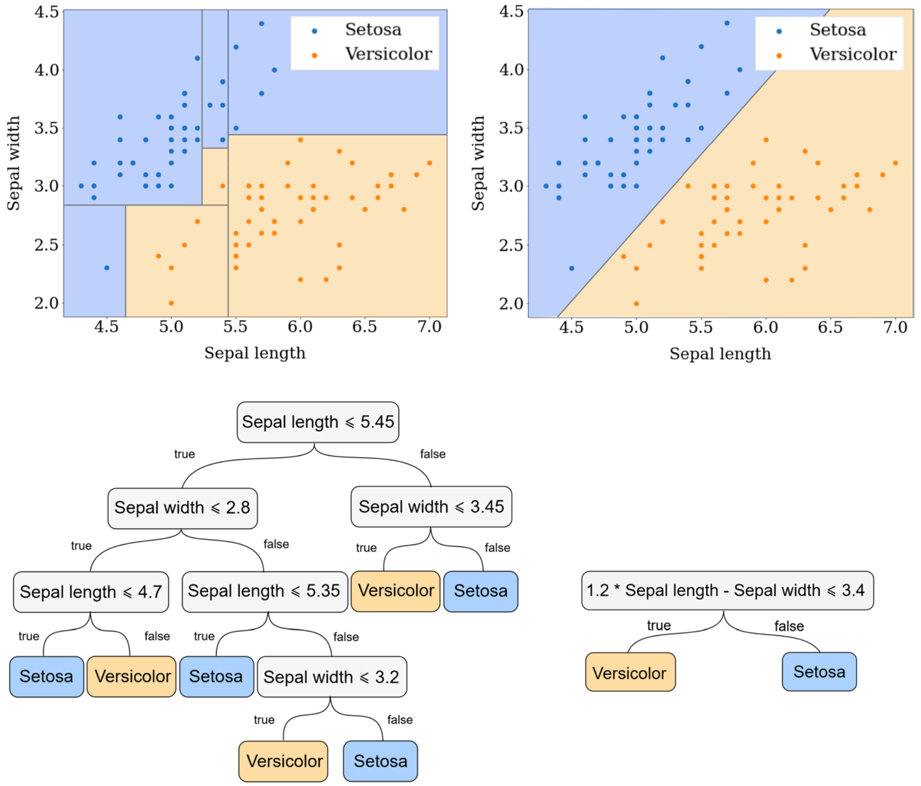
Traditionally, the splits in a DT are *univariate*: they evaluate a single attribute at a time, generally in the form of $x_i \leq c$ for numerical attributes and $x_i = c$ for categorical ones. As a result, these univariate splits correspond to hyperplanes that are orthogonal to the tested attributes and parallel to all others, which renders them the alternative name *axis-parallel splits*. Splits of this kind are employed because they are simple to interpret and optimize, but they have their drawbacks: a DT with univariate splits (called a *univariate tree*) is a composition of orthogonal lines, and as such, it is restricted to representing hyper-rectangles in the attribute space, an effect illustrated in Fig. 5a. When facing target functions such as a simple linear combination between two attributes, univariate DT algorithms may end up inducing unnecessarily large trees that may be not only hard to interpret, but also unable to adequately approximate the target function.

The most popular solution to this problem is the induction of trees that consider multiple attributes in each split, called *multivariate trees*. Since the splits are no longer limited to orthogonal lines, the tree can represent much more than hyper-rectangles, therefore enhancing its approximation capacity. However, this improvement also brings its own set of disadvantages: multivariate splits are considered to be less interpretable than their univariate counterparts (Breiman et al. 1984, Sect. 5.2.2), and they are significantly more time-consuming to induce (since there are many more attributes to consider). Consequently, the use of multivariate trees should be evaluated depending on the complexity of the task and the size of the available sample. For an in-depth survey of different aspects related to multivariate trees, please refer to Brodley and Utgoff (1995).

The next two subsections explore how recent contributions deal with some of these issues. These contributions are divided into two categories: those which employ linear combinations of the attributes at each split, and those which employ non-linearities.

3.1.1 Linear splits

Most multivariate DTs employ a linear combination of attributes in some (or all) of their splits, resulting in the so-called *linear* or *oblique* trees. Accordingly, their splits follow the



(a) Univariate tree **(b)** Linear multivariate tree

Fig. 5 Two classification trees induced on a reduced version of the Iris dataset

form $\sum_{\forall i} w_i x_i \leq c$, where w_i is the weight associated with the ith attribute x_i, and c is a threshold—please see Fig. 5b for an example of such a linear split. In this context, finding the best linear split amounts to finding a set of weights and a threshold such that the splitting criterion is maximized, but since this problem is NP-Complete for many criteria (Heath et al. 1993), heuristics are often employed. Solutions to this problem can be divided into three main approaches: optimization techniques, linear discriminant analysis, and geometrically inspired heuristics.

The *optimization* approach is exemplified by CART-LC (Breiman et al. 1984), a variant of CART that incorporates linear splits induced by a deterministic hill-climbing algorithm. Since it is not straightforward to linearly combine categorical data, only numerical attributes are considered for linear splits, like most other linear tree algorithms. Although the authors reported that CART-LC is competitive with other non-DT methods in tasks where the class structure depends on a linear combination of attributes, some researchers later argued that the deterministic approach of CART-LC makes it vulnerable to getting stuck in low-quality local optima (Heath et al. 1993; Murthy et al. 1993). This motivated the use of randomized search heuristics, such as SADT (Heath et al. 1993), which employs simulated annealing optimization, and OC1 (Murthy et al. 1993), which combines CART-LC and SADT by employing hill-climbing with random perturbations.

Other multivariate DTs use *Linear Discriminant Analysis* (LDA; Tharwat et al. 2017) techniques to find linear splits. Traditionally, these techniques are used in statistics to find a linear combination of attributes that separates classes, through the construction of a lower-dimensional space that maximizes the between-class variance and minimizes the within-class variance. Because they directly output a linear split, LDA methods fit naturally within the traditional greedy DT framework as a replacement for evaluating every candidate split. Among DTs that employ LDA family methods, this paper highlights FACT (Loh and Vanichsetakul 1988), CRUISE (Kim and Loh 2001), QUEST (Loh and Shih 1997), and GUIDE (Loh 2009).

More recent contributions to the LDA family are Sparse ADTrees (Sok et al. 2015) and Fisher's ADTree (Sok et al. 2016). Both are multivariate extensions of the Alternating Decision Tree model (a type of DT that uses boosting, proposed by Freund and Mason 1999), but they differ in the type of LDA used. Sparse ADTrees employ Sparse LDA (Clemmensen et al. 2011) to induce linear splits with few attributes, whereas Fisher's ADTree uses the more traditional Fisher's linear analysis to determine the linear splits at each node, akin to another DT in the LDA family, Fisher's Decision Tree (López-Chau et al. 2013). Since ADTrees are constructed using boosting, both extensions modify their LDA methods to allow for sample weighting—this is the main difference in the splitting procedures of Fisher's Decision Tree and Fisher's ADTree.

The third group of heuristics obtains the linear split by bringing inspiration from the problem's *geometric* structure. The GDT algorithm (Manwani and Sastry 2012) does this by determining two 'clustering hyperplanes' that are as close to one class and as far to the other one as possible; then, only two splits are ever considered at each node, those being the two angles that bisect the hyperplanes. This considerably lowers the computational cost of finding the splits and is shown to produce intuitive results in many scenarios. Wickramarachchi et al. (2016) proposed a different geometric approach in the HHCART algorithm, which creates a Householder matrix from the vector orientations of each class and uses this matrix to rotate the input space. Similar to LDA techniques, the oblique split in the input space is obtained by finding an axis-parallel split in the rotated space. Subsequently, Wickramarachchi et al. (2019) combined HHCART and GDT into the HHCART(G) algorithm: the induction process is similar to the original HHCART's, but instead of creating a rotated space based on eigenvectors, it bases the rotation on a modified version of GDT's angle bisector.

Up to this point, all the cited linear DT algorithms have followed the traditional greedy and iterative approach to constructing trees. However, more recently, some authors have proposed *non-greedy* trees that employ linear splits, using integer programming (Bertsimas and Dunn 2017; Blanquero et al. 2021; Zhu et al. 2020; Aghaei et al. 2019) and gradient-based optimization (Norouzi et al. 2015). Approaches of this sort are discussed in Sects. 3.4.3 and 3.4.4.

3.1.2 Non-linear splits

Aiming to further enhance DT approximation capacity, some contributions pointed at non-linear splits. Fuzzy Rule-Based DTs (Wang et al. 2015c) employ non-linear splits by representing them as fuzzy conjunction rules, such that the non-linear split $f_1(x_1)f_2(x_2) \leq 0.6$ (where f_1 and f_2 are fuzzy membership functions) can be expressed in the much more interpretable form "does the observation have short sepal length and short sepal width?". Although the reported accuracy was not significantly higher than other multivariate approaches, FRDT stands out for its superior interpretability resulting from the fuzzy rule usage. Subsequently, Mu et al. (2020) proposed an extension to FRDT called MR-FRBDT, which applies several fuzzy rules at each split instead of a single one, and employs the parallel framework of Map-Reduce to speed up tree induction. Although accuracy and scalability were improved, it may be noted that interpretability was diminished due to the use of multiple fuzzy rules.

Paez et al. (2019) proposed a strategy by which linear and non-linear splits can be introduced to any DT algorithm. This is done by adding Interactive Basis Functions (IBFs) as artificial attributes to the original dataset, such that the final tree may include splits like $e^{x_1} + e^{x_2} \leq 2$, or $x_1 x_2 \leq 7$. Since the user needs to specify which IBFs will be considered, the strategy turns DTs into a semi-parametric approach, therefore requiring domain knowledge that may not always be present. The authors reported that IBFs have better performance in datasets with more attributes, more observations, and fewer labels. It can be noted that FRDT's non-linearity may be seen as a particular case of the IBF strategy.

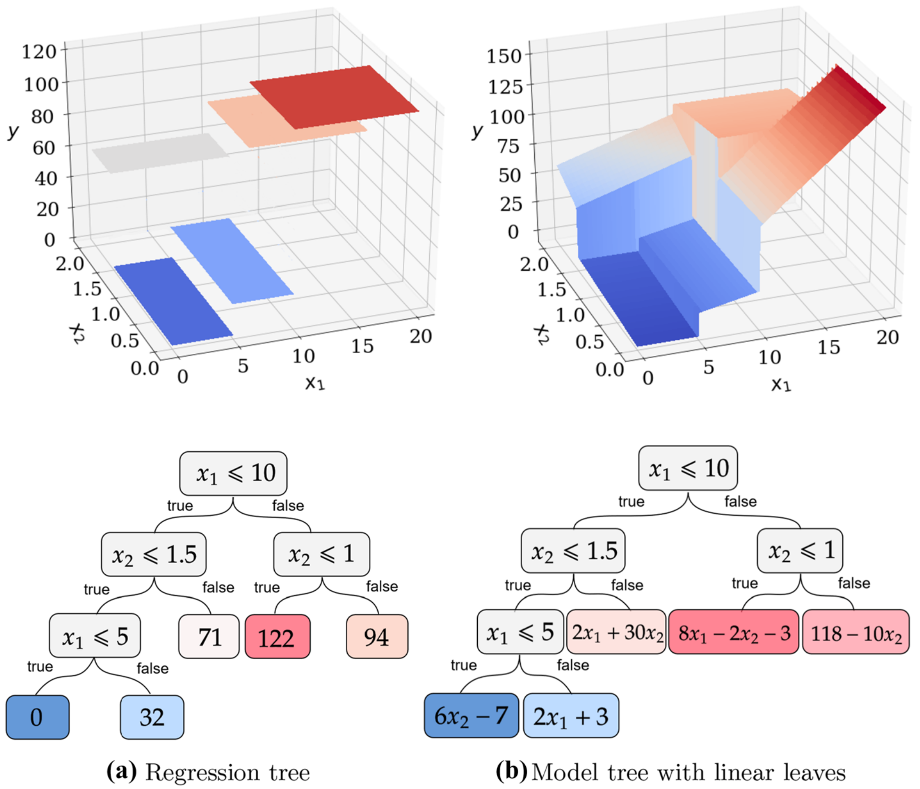
Finally, almost every DT trained with gradient-based optimization employs non-linear splits in some way, either by applying a sigmoid function over a linear combination of input attributes (Suarez and Lutsko 1999; Irsoy et al. 2012, 2014; Norouzi et al. 2015; Frosst and Hinton 2017), or by employing entire neural networks at internal nodes (Hehn et al. 2020; Tanno et al. 2019). These approaches are further explored in Sect. 3.4.3.

3.2 Model trees

While multivariate trees employ more complex splits, other trees employ more complex leaves. *Model trees* are DTs with predictive models in their leaves, instead of labels or constant numbers. Although these trees were originally presented as an extension to regression trees that would not be limited to representing piecewise constant functions (please see Fig. 6) (Quinlan 1992; Breiman et al. 1984, Sect. 8.8), they have since been applied to classification tasks as well, through the use of predictive models such as Naive Bayes (Kohavi 1996) and logistic regression (Landwehr et al. 2005). More recently, research has aimed at inducing more accurate and efficient model trees (Wang et al. 2015b; Zhou and Yan 2019; Czajkowski and Kretowski 2016; Yang et al. 2017; Broelemann and Kasneci 2019), as well as integrating this approach with other DT frameworks (Ikonomovska et al. 2011; Frank et al. 2015).

One of the main challenges of model trees is efficiency: if the greedy procedure is naively employed, then two predictive models (one for each child node) have to be trained to evaluate the quality of a single candidate split (Loh 2011). This is intractable for most real-world tasks, as the number of candidate splits usually scales with the size of the training dataset and the dimensions of its input space. Often, researchers tackle this problem either by growing a standard tree and adding models in post-processing (Quinlan 1992), or by inducing a model for each internal node and then using this model to warm-start the node's children, therefore reducing the cost of training a new model (Landwehr et al. 2005). Both approaches are reflected in recent work.

Broelemann and Kasneci (2019) followed the *warm-starting* approach and proposed an algorithm for inducing trees with any differentiable predictive model for leaves. The authors argued that traditional splitting criteria are inadequate for model trees since they directly attempt to obtain the best partitions; instead, they proposed a novel gradient-based splitting criterion that explicitly considers the model loss of each leaf, reportedly achieving better accuracy than other model trees. After applying the criterion and obtaining a split,



(a) Regression tree

(b) Model tree with linear leaves

Fig. 6 An illustrative comparison between traditional regression trees and model trees

the algorithm trains a model for each new child node and warm-starts it by using the parent's trained model. Warm-start is also used in FIMT-DD (Ikonomovska et al. 2011), an online algorithm for inducing model trees capable of dealing with concept drift. Each leaf has a corresponding perceptron that uses the Delta rule to update its weights with every observation that falls into that path. Every time a concept drift is detected, the leaf is split further, and its perceptron is copied over to both its children in a warm-starting procedure.

The *post-processing* approach is reflected in the work of Zhou and Yan (2019): their algorithm constructs a traditional C4.5 tree and then, after the post-pruning step, replaces the newly-created pruned leaves with ELM classification models (Huang et al. 2006) trained on the corresponding data partitions. The authors reported that this approach was more effective than an earlier model tree algorithm called ELM-trees (Wang et al. 2015b), which instead employed ELM models during pre-pruning and additionally tackled efficiency by employing parallelism in the evaluation of candidate splits.

Other approaches have been proposed to avoid training an intractable number of models. Alternating Model Trees (Frank et al. 2015) consider only the median value of each attribute as a potential split, therefore severely reducing the number of models that need to be trained. Furthermore, the predictive models themselves are univariate linear functions of the form $ax + b$, and consequently can be quickly trained using linear regression. In contrast, MPTrees (Yang et al. 2017) employ a mixed-integer optimization (MIO) formulation to find, given an input attribute, both the optimal split point and the linear regression coefficients of its two children. This offers an optimality guarantee that is absent in other model tree approaches, and since only the input attributes need to be iterated, the efficiency of the overall algorithm depends only on MIO optimization. It bears to note, however, that only datasets with less than 5000 observations were included in the paper, perhaps due to the well-known expensive computational cost of guaranteeing optimality—a similar discussion is presented in Sect. 3.4.4.

Finally, Czajkowski and Kretowski (2016) avoided training multiple models by using an evolutionary algorithm that induces model trees in a global way. After evaluating different combinations of DT structure decisions, the authors found that the best model is a fully mixed tree, in which splits can be either uni- or multivariate, and leaves can be either constants or linear predictors.

3.3 Datasets with specific characteristics

Although traditional DT algorithms were proposed as a general tool for predictive learning, they may not be well-suited or applicable to datasets with certain characteristics, such as unlabeled data and monotonic constraints. Therefore, some authors proposed extensions to better fit these situations, usually by introducing new splitting criteria and pruning mechanisms. The following subsections bundle together some of these contributions.

3.3.1 Unlabeled data

As seen in Sect. 2, classification DT algorithms evaluate candidate splits based on their ability to separate the labels. However, there are situations where the data is either completely unlabeled (unsupervised tasks) or partially unlabeled (semi-supervised tasks). In these situations, it is difficult to use traditional splitting criteria, as they depend on knowing the correct prediction for any observation. To effectively tackle these tasks, the traditional DT framework needs to be modified.

The *unsupervised* task has been tackled in the past by DT algorithms such as CLTrees (Liu et al. 2000) and PCT (Blockeel et al. 1998). More recently, Fraiman et al. (2013) proposed Clustering Unsupervised Binary Trees (CUBT), a CART-based algorithm that uses a deviance splitting criterion based on a heterogeneity measure derived from the input's covariance matrices. After using this criterion to construct a CART tree, CUBT prunes the model by merging pairs of leaves with low dissimilarity, according to another proposed measure based on the Euclidean distance. In contrast to traditional pruning procedures, CUBT includes a procedure for merging even non-adjacent leaves, as this enables different leaves throughout the tree to be labeled as part of the same cluster. It bears to note that both deviance and dissimilarity were defined only for continuous attributes, which is why Ghattas et al. (2017) extended this approach to use entropy-based splitting criteria when the attributes are categorical.

Semi-supervised DT learning, on the other hand, had not received much attention until recent years. Tanha et al. (2017) proposed tree induction through a self-training framework: at each iteration, a DT is induced using the labeled data, and the resulting tree is used to classify the remaining unlabeled data. The unlabeled observations with the most confident predictions are incorporated into the labeled set, and the process is repeated until a stopping criterion is reached. Since correctly estimating the confidence of a prediction is a crucial part of self-training, the authors employed techniques that have been previously reported to enhance the probability estimation capacity of DTs, such as Laplacian Correction (Provost and Domingos 2003), no-pruning (Provost and Domingos 2003), and grafting (Webb 1997). The resulting approach is found to surpass traditional DT algorithms in the self-training semi-supervised task.

However, since self-training is computationally costly and prone to error propagation, Levatić et al. (2017) employed an alternative approach to semi-supervised trees, namely an extension of the previously-cited PCT (Blockeel et al. 1998). The proposed algorithm exploits both labeled and unlabeled data through a splitting criterion that considers not only class impurity (like traditional algorithms) but also attribute impurity. Unlabeled data were used only for the latter type, while labeled data were used for both. A hyper-parameter w weighs these two sources of impurity, effectively controlling the relevance of supervised vs. unsupervised learning. This approach was subsequently extended to multi-target regression (Levatić et al. 2018).

Finally, a similar weighted supervised-vs.-unsupervised splitting criterion was employed by Kim (2016) to produce semi-supervised DTs. However, the goal of the author was not to take advantage of additional unlabeled data (as in a traditional semi-supervised task), but to produce well-separated partitions in which other models could be constructed (as in a model tree environment, discussed in Sect. 3.2). The author argued that labels may be distributed differently in different subspaces, therefore a criterion based only on the output distribution may fail to produce adequate partitions. As such, the input space distribution has to be considered to enhance the subspace partitioning capability of trees.

3.3.2 Cost-sensitive learning

Several real-world problems include some type of specific cost, e.g. the cost of incorrectly classifying an observation (the misclassification cost), or the cost of obtaining an attribute value (the test cost). DTs that consider these two costs have been extensively proposed in the past, and a comprehensive survey on the topic has been written by Lomax and Vadera (2013).

More recently, some contributions have aimed at balancing cost-sensitive learning with *resource constraints*. Chen et al. (2016) proposed trees in which every test costs a time resource and every path from the root to a leaf must not exceed a maximum time. Splitting is done by minimizing not only test and misclassification costs, but also the time used by the node, with the final tree then being post-processed to further grow the nodes that did not exploit the maximum time allotted. Subsequently, the authors extended this idea to consider not only time but multiple arbitrary resource constraints (Wu et al. 2019), though by a different approach: at first, a frequent-pattern extraction method mines rules from the training data that satisfy the resource constraints, and then, a tree is induced using the extracted rules as possible splits. This guarantees that the final tree satisfies all constraints without explicitly considering them during the growing step.

Other authors tackle the issue of *efficiently* building cost-sensitive trees. The ACSDT algorithm (Li et al. 2015) employs two procedures with this aim: first, it considers only a subset of candidate splits by using a gradient-descent-inspired method, and second, it employs feature selection by removing attributes that present low splitting criterion values at some point in the tree construction. This latter idea is also explored in the Batch-Deleting Adaptive DT algorithm (BDADT; Zhao and Li 2017), an algorithm that uses the same feature selection step but considers class imbalance by incorporating weights during splitting.

Finally, Correa Bahnsen et al. (2015) have focused on *example-dependency*: a third type of cost in which the penalty for misclassification depends not only upon the classes but also upon which individual observation is being considered. To tackle this problem, the authors proposed novel splitting criteria and pruning mechanisms that allow for example-specific weights.

3.3.3 Monotonicity constraints

In certain tasks (called *monotonic*), it is known that the target attribute is a monotonic function of the input attributes for at least some subset of the data. Since traditional splitting criteria fail to address such constraints, several authors have proposed extensions over the years, such as Potharst and Bioch (1999) and Cao-Van and De Baets (2003). More recently, Hu et al. (2010) introduced Rank Mutual Information (RMI), a new information measure that combines Shannon's entropy with dominance rough set theory in an attempt to capture the ordinal consistency of the data's partitions. Based on this measure, the authors then proposed a DT algorithm called REMT (Hu et al. 2012) which was reported to be less sensitive to noise than previously proposed algorithms.

Subsequently, Marsala and Petturiti (2015) generalized the RMI approach to splitting criteria other than Shannon's entropy, formally defining a group of rank discrimination measures that includes RMI and a newly proposed Rank Gini Impurity (RGI). In the same paper, the authors also proposed RMDT(H), a tree induction algorithm that evaluates splits based on any rank discrimination measure H from this group of measures. Finally, Pei et al. (2016) proposed a linear multivariate DT called MMT that can employ both RMI and RGI to tasks with partially monotonic constraints. Such constraints are handled by projecting the observations unto the multivariate split vector, consequently creating a space in which non-monotone samples can be seen as monotonic. The splits themselves were obtained in a manner reminiscent of the linear discriminant analysis approach discussed in Sect. 3.1.1, with the difference being that the non-negative least squares method is used, so as to guarantee monotonicity throughout the tree.

3.4 Non-greedy trees

As discussed in Sect. 2, DT induction can be seen as an optimization problem, in which the goal is to construct a tree that correctly fits the data while using a minimal number of splits. However, direct optimization is not a convenient approach, since this task has been identified as NP-Complete in certain cases, such as for binary trees (Hyafil and Rivest 1976). Therefore, heuristics are often employed to efficiently navigate the search space, with top-down greedy algorithms being the most popular by a large margin (e.g., CART, ID3, C4.5, GUIDE).

However, there has been some debate over the effectiveness of greedy approaches. In particular, it can be argued that such algorithms are only one-step optimal and not overall optimal, since the construction procedure only considers the quality of the next split and not of future splits on the same path (Breiman et al. 1984, Sect. 2.8.2). Consequently, several authors have proposed non-greedy heuristics for navigating the DT search space more appropriately, as well as optimizing techniques that improve the tractability of direct optimization. The following subsections explore lookahead search, evolutionary methods, gradient-based optimization, and optimal trees.

3.4.1 Lookahead trees

Lookahead search is an intuitive approach to reduce the shortsightedness of greedy algorithms: instead of selecting the optimal split with regards to the next iteration, a *k*-lookahead selects the optimal split relating to the next *k* iterations. Satisfactory results with this approach were initially reported by Norton (1989) and Ragavan and Rendell (1993), with the latter showing particular success in tasks with high attribute interaction. Apparently, the only downside of lookahead search was the increase in computational cost (since a greater number of splits need to be considered).

Nevertheless, Murthy and Salzberg (1995a) demonstrated that one-level lookahead exhibits *pathology*: not only does it fail to produce significantly better trees, but also it may produce trees that are worse than the purely greedy methods. The authors hypothesized that this happens because optimizing common splitting criteria (such as information gain) does not necessarily improve the tree as a whole, therefore a better local optimization can lead to even more sub-optimal solutions. In face of these results, interest in the lookahead approach waned, and other non-greedy methods were explored in the literature.

However, it is not a consensus whether lookahead is overall harmful or if it may have some kind of potential. Esmeir and Markovitch (2007) argued that Murthy and Salzberg's paper only studied datasets with simple target functions, while lookahead is more advantageous when there is high attribute interaction (an observation reflected by Ragavan and Rendell's analysis). They then proposed LSID3, an anytime DT algorithm that constructs trees using a dynamic lookahead and employs a splitting criterion that is biased towards shallower trees. This criterion explicitly considers the size of the subtree under each candidate path, which is estimated via a Monte Carlo sampling of the possible trees. LSID3 differs from the approach studied by Murthy and Salzberg in two main ways: it employs a dynamic lookahead instead of a fixed one-level, and it is used in datasets with high attribute interaction. Perhaps due to these differences, the authors reported that LSID3 outperforms greedy algorithms, especially when a larger time budget is available.

A related and more recent approach is the UCT-DT algorithm (Nunes et al. 2020), which combines Monte Carlo sampling with an anytime property in a different way: namely, it uses the Monte Carlo Tree Search (MCTS) algorithm to navigate the space of possible DTs. In MCTS the search space is represented as a game tree, which is iteratively built by selecting the node with the highest value and expanding it. UCT-DT applies the same idea, but the search space is composed of possible DTs instead of game states, and the value of a particular tree is defined as its accuracy on a validation set. Lookahead search is applied in two ways: first, in the usage of the tree-structured approach of MCTS itself, and second, in the experimental usage of C4.5 to complete the tree before measuring its accuracy. Although the authors found that UCT-DT outperforms greedy approaches on datasets larger than 1000 observations, they also reported that using C4.5 to complete the tree was not superior to using the performance of the current DT, which can be taken to indicate that at least part of the lookahead approach was not beneficial. Overall, there seems to be space for further investigating the effects of lookahead search, especially since the computational constraints are progressively less limiting.

3.4.2 Evolutionary trees

A different approach for avoiding sub-optimality is inducing trees through *evolutionary* algorithms (EAs; Mitchell 1998). Instead of building a single tree, these methods build several trees and randomly modify and combine them using “genetic operators” until a satisfactory solution is found. Moreover, instead of using a splitting criterion to compare candidate splits at each internal node, these methods compare multiple DTs at each iteration using a “fitness function” (usually a weighted sum of tree size and accuracy). Overall, it has been argued that by evaluating the performance of entire trees at once, EAs avoid the sub-optimality trap, and therefore produce more accurate trees (Barros et al. 2012). The more complex search comes with two main downsides: a higher computational cost, and a large number of hyperparameters that need to be somehow adequately determined.

Barros et al. (2012) provided a comprehensive survey on the usage of EAs for the induction of DTs. Since this contribution, many authors have introduced new algorithms to the field, most of which can be adequately analyzed through the taxonomic lenses proposed in that survey. For example, Barros et al. noted the popular usage of multi-objective fitness functions, and some of the more recent algorithms follow the same approach (Brunello et al. 2017; Chabbouh et al. 2019; Karabadji et al. 2017). Furthermore, the survey noted how many EAs employ a random initial population, and this is repeated in some recent contributions (Chabbouh et al. 2019; Karabadji et al. 2017).

Some advances in evolutionary trees deserve to be emphasized: Barros et al. highlighted the need for *efficient* implementations that alleviated the computational cost of EAs, and more recent contributions tackled this goal through parallelism. Czajkowski et al. (2015) combined the parallel approaches of shared memory and message passing to induce DTs using the GDT framework (Kretowski and Grzes 2007), attaining a speedup close to x15 for CPUs with 64 cores. In a subsequent paper (Jurczuk et al. 2017), the same authors improved this speedup by using general-purpose GPUs for evaluating the fitness of each tree, which they argued to be the most time-consuming step of the evolutionary approach. Speedups ranging from x148 to x781 were reported, depending on the dataset and GPU used. Notably, this approach induced trees in less than 10 min for datasets with close to 20 million observations.

Another aspect that has been recently receiving considerable attention is the usage of EAs that also deal with *hyperparameter optimizations*, such as BiLeGA (Adibi 2019), a two-level EA that alternates between feature selection and DT induction. During the feature selection phase, the algorithm generates groups of attributes by applying genetic operators, while in the DT induction phase, a tree is evolutionarily constructed using only the attributes of the corresponding attribute group. Karabadji et al. (2017) similarly considered feature selection, but went one step further by providing genes for many other hyperparameters, such as which subset of training and testing data are to be used, and even which induction algorithm itself (e.g., CART, C4.5, etc). This hyperparameter-evolution approach reached its most ambitious version with HEAD-DT (Barros et al. 2015), an algorithm that aims to evolve entire DT induction methods by including genes for design decisions such as splitting criterion, stopping criterion, post-pruning method, and tree branching factor. By doing this, the authors aimed to obtain not only a better-optimized tree, but a better-optimized DT algorithm for certain domains.

3.4.3 Gradient-based trees

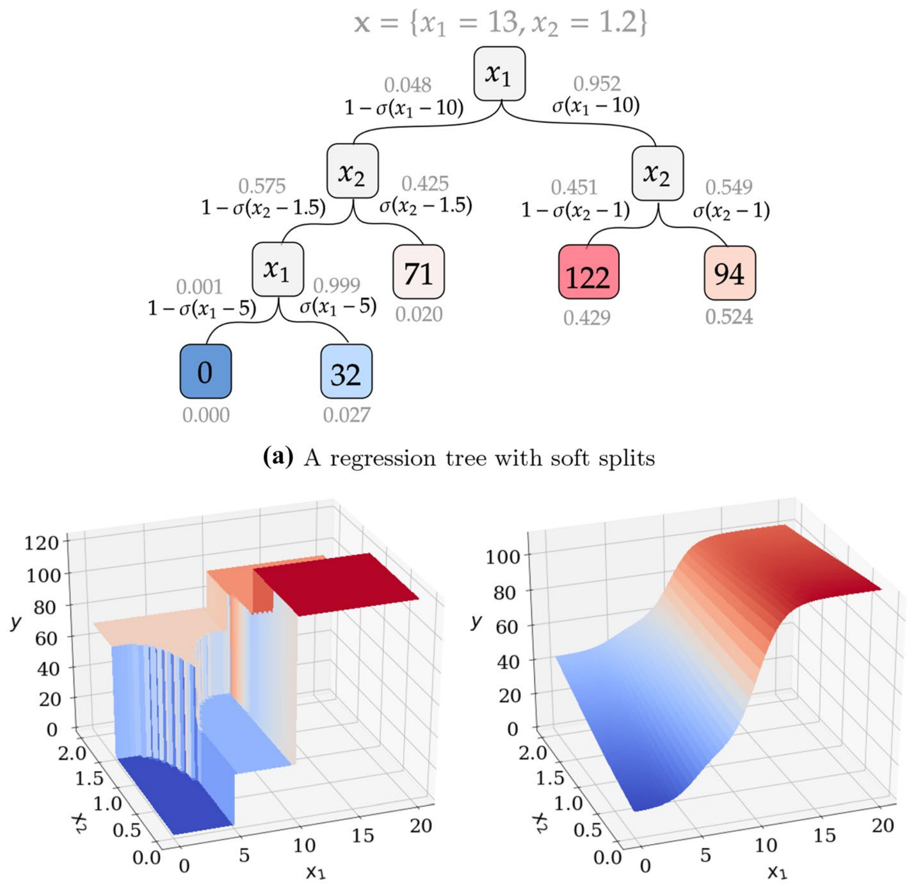
Gradient-based optimization has been increasingly explored as an alternative to greedy DT induction. Although this idea is not new (Jordan and Jacobs 1994; Suarez and Lutsko 1999), the last decade's renaissance in deep learning has prompted many members of the artificial neural networks (ANNs) community to experiment with using backpropagation-inspired approaches in other methods, such as DTs. Most of these contributions are motivated by the sub-optimality argument—greedy algorithms produce sub-optimal trees, therefore the non-greedy gradient-based optimization may help (Suarez and Lutsko 1999; Irsoy et al. 2012, 2014; Norouzi et al. 2015), but with the recent interest in explainable AI, an increasing number of works also report being inspired by the potential of a hybrid model that combines DT's interpretability with ANN's accuracy (Frosst and Hinton 2017; Yang et al. 2018; Silva et al. 2020; Hehn et al. 2020; Wan et al. 2020).

Regardless of the motivation, all such approaches face a common challenge: gradient-based optimization requires continuous and differentiable functions, but traditional DTs are defined by discrete and non-linear decisions, such as selecting which attribute will be used in a split or if an observation goes left or right. To resolve this contradiction, gradient-based methods replace the traditional *hard* DT architecture with a *softer* version that is more amenable for differentiation. This is done via two major modifications: the nature of the splits, and how they are interpreted.

Instead of using the traditional non-differentiable splits $x_i \leq c$, soft DTs usually employ the differentiable function $\sigma(f(\mathbf{x}))$, with $\sigma(\cdot)$ being the sigmoid function and $f(\mathbf{x})$ being the linear combination plus bias $\sum_{\forall i} w_i x_i + b$ (Suarez and Lutsko 1999; Irsoy et al. 2012, 2014; Frosst and Hinton 2017; Wan et al. 2020). The inspiration for this is clear: the tree's internal nodes now mimic the network's neurons, and a backpropagation-inspired derivation is even more immediate. In some cases this analogy is eschewed since f is defined as a convolutional ANN (Hehn et al. 2020; Tanno et al. 2019), however, the overall idea remains the same since these networks are also differentiable.

The sigmoid function brings differentiability, but also creates another problem: how to use its output (a real number between 0 and 1) to decide if an observation goes left or right? The most popular solution is to interpret $\sigma(f(\mathbf{x}))$ (resp. $1-\sigma(f(\mathbf{x}))$) as the probability of an observation $\mathbf{x}$ going right (resp. left), therefore turning the traditional mutually exclusive splits (called *crisp*) into weighted ones (called *soft*). There are three ways in which these soft splits can be used to output a prediction. In the first way, the outputted prediction is determined only by the leaf with the highest probability (Frosst and Hinton 2017; Wan et al. 2020), whereas in the second way, the prediction is determined by an aggregation of all leaves, for example through a weighted sum (Suarez and Lutsko 1999; Irsoy et al. 2012, 2014; Hehn et al. 2020). These two different modes are illustrated in Fig. 7. The third way differs from the previous two by virtue of being stochastic: the probabilities are sampled at each split, and a path throughout the tree is chosen (Tanno et al. 2019). It may be noted that the first two approaches bring additional computational costs since they require visiting every node in the tree, whereas the last approach avoids this problem altogether and introduces a new potential disadvantage (stochasticity).

There are exceptions to the soft splits solution. Norouzi et al. (2015) managed to use gradient-based optimization in a DT without resorting to soft splits, by establishing a link between traditional DT optimization and structured prediction with latent attributes. This provided their method with a continuous upper bound that serves as a surrogate



(a) A regression tree with soft splits

(b) Resulting partitioning when the leaf with maximum probability is returned

(c) Resulting partitioning when leaves are aggregated by a weighted sum

Fig. 7 An illustrative comparison between two ways to interpret soft splits from a regression tree. Gray values correspond to an example observation $\mathbf{x} = (x_1 = 13, x_2 = 1.2)$

objective for gradient descent, therefore bypassing the discontinuous error inherent to traditional crisp splits. Hehn et al. (2020) took a different approach and employed soft splits only during training time, by combining the Expectation-Maximization approach with an annealing scheme that makes the nodes increasingly deterministic. Therefore, at test time, all nodes produce either 0 or 1 as output (for left or right).

Another aspect that differs between these methods is how the *tree induction* occurs. Most algorithms are divided into two phases: greedy tree construction, and fine-tuning. In the first phase, a tree is greedily constructed as usual, except that after each split, the parameters of the new subtree are updated through gradient-based optimization, while the rest of the tree's parameters are fixed. In the second phase (fine-tuning), the algorithm runs a final gradient update through the entire tree, updating all of its weights while fixing the structure. Some algorithms employ both phases (Irsoy et al. 2012; Tanno et al. 2019; Hehn et al. 2020), while others restrict themselves to the fine-tuning step, and assume that a tree structure is given instead of greedily constructing it (these "fixed tree structure" approaches are later discussed in Sect. 4.3). Still other approaches exist. NBDT (Wan et al. 2020) applies fine-tuning, but induces a tree differently: first, it generates the tree's leaves based on the weights at the output layer of a neural network, and then, it constructs the rest of the tree by merging nodes in a bottom-up process (akin to hierarchical agglomerative clustering). Budding Trees (Irsoy et al. 2014) eschew both greedy induction and fine-tuning by considering every node to be both a leaf and an internal node, therefore conflating tree construction with pruning and updating all weights at every iteration.

Furthermore, one of the benefits of employing gradient-based optimization is the possibility of explicitly using *regularization* to guide the training process. This idea has been explored by a few authors: Norouzi et al. (2015) used norm regularization to constrain the learned weights in the nodes, Hehn et al. (2020) followed a similar approach but through a spatial regularization particular to image inputs, and Frosst and Hinton (2017) employed a penalty that encourages internal nodes to make equal use of both its subtrees.

Overall, gradient-based optimization has been reported to result in trees that are more accurate and shallower, apparently delivering on their promise of less sub-optimal DTs. However, this advantage comes with a cost: by employing all input attributes at every node, it becomes humanly impossible to understand why the model made a certain decision, therefore giving up the traditional notion of DT transparency. This problem and proposed solutions are discussed in Sect. 5.4 on interpretability.

3.4.4 Optimal trees

When taken to the extreme, the idea of avoiding sub-optimality leads to optimal trees: algorithms that search the space of possible models and find the tree that maximizes some measure (e.g. accuracy) while under a size constraint (e.g. maximum depth). Since the search space is impossibly large, methods in this category are more computationally expensive than their heuristical counterparts, which is why despite similar ideas having been explored in the past (Meisel and Michalopoulos 1973; Bennett and Blue 1996), the approach only started to flourish with the hardware advancements of recent years (Bertsimas and Dunn 2017). Currently, there are three main lines of research being investigated: mathematical programming, Boolean satisfiability, and specialized approaches. This subsection explores each in turn and then comments on issues common to all three.

In the *mathematical programming* (MP) approach, DT induction is formulated as a mathematical program, which is then passed to generic commercial solvers such as Gurobi or CPLEX who return the optimal tree. Since the DT induction problem involves several discrete choices, the formulations themselves are usually classified as either MIO (Verwer and Zhang 2017; Bertsimas and Dunn 2017; Rhuggenaath et al. 2018; Aghaei et al. 2019; Firat et al. 2020; Günlük et al. 2021) or purely binary integer programming (BIP; Verwer and Zhang 2019). Although approaches of this line of research can be traced back to the '90s (Bennett 1992; Bennett and Blue 1996), the idea has only gained steam with the technical advances of modern optimization solvers (Bertsimas and Dunn 2017).

This recent interest was ignited in 2017, with the publication of OCT (Bertsimas and Dunn 2017) and DTIP (Verwer and Zhang 2017), two algorithms that induce univariate trees using MIO. DTIP employs fewer variables and restrictions than OCT, which makes it more scalable with regards to sample size; OCT, on the other hand, finds the tree structure in a more sophisticated way and is more thoroughly investigated in relation to the traditional greedy approaches. Both algorithms were subsequently built upon: OCT was extended for regression (Dunn 2018) and prescriptive tasks (Bertsimas et al. 2019), while

DTIP was extended for fuzzy trees (Rhuggenaath et al. 2018) and its authors proposed a novel binary formulation called BinOCT (Verwer and Zhang 2019). BinOCT stands out for replacing the dependency on the dataset size with a dependency on the number of different valued attributes present in the dataset, therefore improving the scalability of optimal trees even further. For a detailed view of MP for DT induction, please refer to Carrizosa et al. (2021).

A second line of research poses DT induction as a *Boolean satisfiability* (SAT) problem, in which the goal is to assign values to Boolean variables such that a certain set of clauses is satisfied. This approach is similar to the previous one in the sense that generic solvers are also employed (SAT solvers in this case), but it notably differs in the definition of optimality that is often used: in the SAT approach, a tree is optimal if it is consistent (i.e. perfectly predicts all observations) while having a minimal size. This focus on consistency is inherited from technical considerations of the SAT framework, and brings with it some downsides: it makes the approach infeasible for noisy datasets (where inconsistent data often appears) and for datasets that have many observations (in which case the consistent trees would be impossibly large). It has also has been reported to result in overfitting, even when the trees are small (Hu et al. 2020).

This line of work started with Bessiere et al. (2009), who proposed an initial formulation for DT induction as an SAT problem, and further showed that it was only practical for small datasets with less than a hundred observations. The works that followed all tried to tackle this issue of scalability: Narodytska et al. (2018) built upon their work and provided a tighter formulation that solved these small instances orders of magnitude faster; Avellaneda (2020) proposed an iterative strategy that considers incrementally larger subsets of the training data, until either the whole training set is considered, or no optimal tree can be found under the size constraints; Janota and Morgado (2020) proposed a formulation that explicitly considers the tree's paths, and was reported to outperform Narodytska et al.'s approach. However, despite this chain of improvements, scalability remains a challenge. More recently, works have considered the SAT approach from other angles: Schidler and Szeider (2021) used SAT techniques to progressively prune a greedy-induced tree, while Hu et al. (2020) used the MaxSAT framework to maximize accuracy instead of capturing it perfectly, resulting in an algorithm closer to the MP approach.

Instead of passing formulations to generic solvers, a third group of works is concerned with more *specialized* approaches. Two lines of research are highlighted.

DL8 (Nijssen and Fromont 2010) frames DT induction as a frequent itemset mining problem: each path corresponds to an itemset, and ideas similar to dynamic programming can be employed to find the optimal tree. This is made possible due to the decomposable nature of DT induction: if a tree is optimal, then any subtree it contains is also optimal. Subsequently, Aglin et al. (2020) built upon these ideas to propose DL8.5, an algorithm that achieves higher efficiency by using more sophisticated caching techniques and bounds to prune the search space. MurTree (Demirović et al. 2021) is currently the last word in this line of research: caching and bounding are further refined, and the problem is explicitly framed in the conventional dynamic programming framework.

A parallel line of specialized algorithms started with OSDT (Hu et al. 2019). Leveraging their previous work on optimal rule lists (Angelino et al. 2017), the authors established several bounds for tree accuracy (e.g. if a tree has a lower bound on its accuracy, then further splitting the tree will maintain the bound, etc.) which the algorithm then uses in an explicit branch-and-bound approach to prune the search space and search it more efficiently. A collection of specialized data structures was also developed to reduce computational costs, similar to MurTree. An extension called GOSDT (Lin et al. 2020) generalized

OSDT to objective measures other than accuracy and further adapted it to treat continuous attributes more efficiently.

This subsection concludes with comments on optimal trees as a whole. An advantage of these algorithms is *flexibility*: since the objective functions are explicitly defined, it is easy to adapt them to domains where accuracy is not the measure to be optimized. This is not the case for traditional greedy trees, where the global objective is not tackled directly but indirectly through the splitting criterion (due to the local optimality itself Breiman et al. 1984, Sect. 4.1). So far, the flexibility of optimal trees has been showcased for objectives like discrimination measures (Aghaei et al. 2019), imbalanced accuracy (Verwer and Zhang 2017; Lin et al. 2020; Günlük et al. 2021), and general combinations of accuracy and size (Bertsimas and Dunn 2017; Hu et al. 2020; Lin et al. 2020; Demirović et al. 2021). A limitation of this approach is that most optimal trees are limited to representing linear measures, as their methods are either based upon linear programming or exploit mathematical properties of linear measures. Non-linear goals require alternative approaches like the bi-objective optimization considered by Demirović and Stuckey (2021).

Another theme commonly discussed is the required pre-processing of *attribute types*. Because splits of the form “attribute ≤ value” can be easily formulated as linear constraints, the majority of MP approaches assume that all attributes are numerical, relying on transformations such as one-hot encoding to handle categorical data (Bertsimas and Dunn 2017; Verwer and Zhang 2017, 2019; Firat et al. 2020; Blanquero et al. 2021). This works well but is in stark contrast to traditional greedy algorithms like CART, which handle categorical and numerical data out-of-the-box. The closest optimal tree algorithm in this regard is the formulation proposed by Aghaei et al. (2019), which proposes an MIO problem that handles mixed data.

Inspired by similar technical considerations, other algorithms assume that all attributes are categorical (Günlük et al. 2021) or, more often, binary (Bessiere et al. 2009; Nijssen and Fromont 2010; Narodytska et al. 2018; Verwer and Zhang 2019; Aglin et al. 2020; Janota and Morgado 2020; Demirović et al. 2021). For categorical attributes the encoding is straightforward, but numerical attributes require special consideration, because encoding every different value is infeasible. A proposed solution is “bucketization”: the process of ordering the observed values and only considering splits between pairs with different labels (Verwer and Zhang 2019; Aglin et al. 2020). However, although this idea is attractive, Lin et al. (2020) have shown that it sacrifices optimality. More sophisticated approaches include efficiently considering every possible threshold via specialized approaches (Lin et al. 2020), using supervised discretization algorithms (Demirović et al. 2021), or decile binning (Günlük et al. 2021).

A major challenge to all optimal tree algorithms is *scalability*: the high computational cost of searching the entire tree space (albeit implicitly) significantly increases the time necessary to process datasets with more attributes and observations. As previously noted, this is particularly problematic in the SAT line of research, where certificates of optimality are restricted to datasets with hundreds of observations. However, scalability is also an issue in the MP approach, where algorithms only tackle datasets with less than 5000 observations. This effect can also be seen in the depth of the trees considered: most works consider depths of at most 4 or 5 (Bertsimas and Dunn 2017; Verwer and Zhang 2017; Rhuggenaath et al. 2018; Verwer and Zhang 2019; Aglin et al. 2020; Avellaneda 2020; Firat et al. 2020).

Several authors have proposed ways of sidestepping the scalability issue, for example by inducing trees on a subsample of the training data (Bessiere et al. 2009; Narodytska et al. 2018; Avellaneda 2020) or by reducing the input space (Narodytska et al.

2018; Firat et al. 2020). However, these techniques often sacrifice optimality, which seems counter to the initial idea of optimal trees. The line of specialized algorithms seem to fare better in this regard: by avoiding the use of generic solvers in favor of highly specialized implementations that exploit characteristics of the DT induction problem, recent proposals like MurTree and GOSDT can handle datasets with tens of thousands of observations and dozens of attributes, often achieving certificates of optimality.

Interestingly, the scalability issue of optimal trees might not be so problematic. Murthy and Salzberg (1995b) empirically found that the gap between greedy trees and their optimal counterparts is most significant when the target function is complex but there are few observations—the gap diminishes when the training dataset grows larger. If true, this would mean that optimal trees are most useful especially when there are few observations, which is where they already flourish. It remains to be seen if the hypothesis holds for more thorough investigations of this kind.

Lastly, although the previous discussion has focused on univariate trees, it is noteworthy that multivariate trees (seen in Sect. 3.1) have also been adapted to the optimal framework, such as OCT-H (Bertsimas and Dunn 2017), ORCT (Blanquero et al. 2021, 2020), SVM-1 ODT (Zhu et al. 2020) and the formulation by Aghaei et al. (2019). Additionally, Aghaei et al. also considered optimal trees with model leaves, such as the ones seen in Sect. 3.2.

4 Generalization

The previous sections focused on different ways of enhancing the approximation capacity of DTs, but generalization is also a key concern. There is an intuitive trade-off between approximation and generalization: if a model has high representational power, it has a higher chance of perfectly representing the training data (overfitting) possibly at the expense of a poor generalization performance on the out-of-sample data. Conversely, if a model has low representational power, overfitting is less probable and the out-of-sample results will be probably comparable to the in-sample accuracy, although one should be aware that the models may end up underfitted. This situation is analogous to the bias–variance trade-off, and it is also related to the number of degrees of freedom involved in the models.

In the context of DTs, there is a strong connection between generalization and tree size: shallower trees have a lower representational capacity and tend to generalize better, whereas deeper trees correspondingly tend to generalize worse. Because of this, any technique that directly or indirectly reduces tree size can be seen as partly beneficial to generalization, which includes the previously seen multivariate splits (by enhancing the representational capacity of the nodes, the final tree may end up smaller) and non-greedy trees (a more optimal tree may have a lower number of nodes). But generalization concerns more than tree size: many of the topics presented as advantageous to approximation (Sect. 3) can also be understood as contributing negatively to generalization, since they enhance the representational capacity of the tree as a whole. Amid these multiple lenses through which generalization can be seen, this section constrains itself only to approaches that aim at generalization by directly involving tree size, namely: pre-pruning, post-pruning, and fixed structures.

4.1 Pre-pruning

Possibly the most intuitive approach to tree size reduction is simply to stop growing the tree when it becomes too large (according to a pre-determined criterion). This approach is called *pre-pruning*, and it can be seen as an instance of the traditional early stopping procedure. In its most basic form, pre-pruning algorithms amount to applying a minimum threshold on the splitting criterion (Gleser and Collen 1972; Quinlan 1986), but more complex methods have also been employed.

Recently, Wu et al. (2016) proposed SCDT (Size Constrained DT), a DT induction algorithm that grows the tree until all leaves are pure or until the number of leaves reaches the user-specified maximum. As a result, the order of node evaluation becomes highly relevant, and nodes are evaluated in ascending splitting criterion order. This is also the case for other pre-pruning algorithms, such as the recently proposed MSI (Garcia Leiva et al. 2019). In MSI, Kolmogorov complexity is employed during induction to estimate the complexity of the current tree, and when the tree starts to accumulate unnecessary complexity, the growing procedure is interrupted. The authors reported that this approach leads MSI to induce trees that are slightly less accurate but significantly shorter than traditional algorithms, all the while having no hyper-parameters. This trade-off appears to be in line with the reported "horizon effect" of pre-pruning algorithms: since good splits may be hiding behind bad splits, local information is occasionally unable to find more accurate trees (Breiman et al. 1984, Sect. 3.1). This effect is responsible for the inconsistent accuracy of the pre-pruning approach, and it is also why pre-pruning has been largely abandoned in favor of post-pruning methods (Breslow and Aha 1997).

4.2 Post-pruning

The most popular tree simplification approach is post-pruning, sometimes simply called *pruning*. The main idea is to allow the DT to grow unimpeded, and then eliminate the subtrees that are introducing needless complexity to the model; this complexity is often measured on a validation set. Although slower than other methods such as pre-pruning, post-pruning has consistently been shown to improve accuracy, especially in tasks with a high level of noise (Mingers 1989; Quinlan 1987). One classic pruning method is CART's cost-complexity pruning (Breiman et al. 1984), which has two phases: in the first phase, the complete tree is used to create several trees with increasing levels of simplification, and in the second phase, one of these simplified trees is selected according to a measure that combines its accuracy (cost) with its number of nodes (complexity).

Although post-pruning remains an essential element of DT learning, the issue has not received much innovation recently. Most recent algorithms employ traditional post-pruning methods, such as the previously cited cost-complexity pruning (Loh and Shih 1997; Barros et al. 2015; Wickramarachchi et al. 2016), ID3's pessimistic error pruning (López-Chau et al. 2013), C4.5's error-based pruning (Esmeir and Markovitch 2007), or even no post-pruning at all (Hehn et al. 2020; Wang et al. 2015c; Bertsimas and Dunn 2017; Verwer and Zhang 2019).

An exception to this trend is visual pruning (Iorio et al. 2019). In this method, the tree is visualized as a dendrogram-like structure, with the branches having a length proportional to the impurity reduction they bring. The pruning is done by selecting a threshold beyond which all subtrees are pruned to leaves, akin to applying a minimum splitting criterion to an already constructed tree. The process renders a direct visual interpretation and can notably be employed without the use of a separate validation set, which frees up data that can be used for training. Although advantageous at first, this idea can bring in its own set of problems, since pruning approaches that exclusively use the training data have been found to be more optimistic and therefore prone to overfitting (Mingers 1989). For this reason, pruning approaches of this kind usually employ some type of pessimistic correction (Quinlan 1987). Because such corrections are absent in visual pruning, it remains to be seen if the approach could be enhanced with such corrections.

4.3 Fixed structure

Fixed structure methods take a tree structure as input and search for an accurate set of splits that fits the given structure. Seen from another angle, this approach obtains simplified models by removing trees with undesirable structures from the hypothesis space. Suarez and Lutsko (1999) stated that this procedure is analogous to fixing the architecture and then finding the weights in an ANN.

This idea is common in gradient-based (Sect. 3.4.3) and optimal trees (Sect. 3.4.4), but for different reasons. In gradient-based trees, fixed structures appear to be a consequence of bringing inspiration from ANNs, where the architectures are also fixed. Meanwhile, in optimal trees, fixing the structure is both a technical consideration (the mathematical programming formulations are simpler) and a practical one (optimizing a fixed structure is already computationally costly, therefore iterating over many structures is not prioritized).

Regardless of origin, a key issue is how to determine the fixed structure itself, which either involves user-defined parameters or some sort of automated procedure. User-centric approaches include optimizing the maximal tree of user-specified depth (Verwer and Zhang 2017; Frosst and Hinton 2017; Verwer and Zhang 2019; Blanquero et al. 2021), assuming that the entire structure is provided by the user (Aghaei et al. 2019), or using cross-validation to select a tree from a small group of pre-determined fixed structures (Günlük et al. 2021). Automatic procedures, on the other hand, usually run a greedy induction algorithm like CART and employ the resulting tree's structure as the one to be fixed (Suarez and Lutsko 1999; Silva et al. 2020; Norouzi et al. 2015). A more sophisticated approach can be found in the OCT algorithm (Bertsimas and Dunn 2017), which combines the two ideas: it selects fixed structures from a pool of CART trees trimmed to various depths, up to a maximum depth specified by the user.

In general, fixed structure algorithms are employed more out of necessity than out of concern for generalization, though they are nevertheless positive in that regard (since the provided tree structures are often shallow). However, it has been argued that if the goal is to reduce tree size, then fixing the number of nodes is too rigid a strategy, since the model may become unaware of reasonable trade-offs (e.g. adding a single node may double accuracy) (Freitas 2014). Overall, fixed structures can be seen as both a boon and a barrier to be surpassed.

5 Interpretability

DTs are closely related to interpretability: not only are they often regarded as a particularly interpretable model (Freitas 2014), but also interpretability itself is regarded as the "biggest single advantage of the tree-structured approach" (Breiman and Friedman 1988).

However, not all DTs are equally interpretable, or even interpretable at all, e.g. a tree with thousands of nodes is much more difficult to grasp than a shallower version of it. Consequently, several authors have proposed ways to create more interpretable trees, exploring aspects such as tree size, multivariate splits, soft splits, and others. This section bundles such works.

5.1 Tree size

Tree size is the aspect most commonly associated with interpretability: shallow trees are regarded as very interpretable, whereas deep trees are not (Lipton 2018; Molnar 2022). Using the interpretability framework proposed by Lipton (2018), the issue is *simulatability*: it is difficult for a user to mentally simulate the tree as a whole when there are too many nodes involved. For trees with modest complexity (3 to 12 leaves), this claim was empirically observed in the survey done by Piltaver et al. (2016), in which a strong negative correlation was found between the number of leaves in a tree and a “comprehensibility” score given by the respondents.

But although tree size is important to interpretability, it does not capture this notion entirely. As argued by Freitas (2014), users may find a larger tree to be more interpretable than a shallower version of it, if the splits of the larger model better reflect the user’s domain knowledge. This exemplifies the fact that size is only a syntactic notion, and that the semantic aspects of the model (its contents) must be considered for a fuller picture of interpretability.

This complex relation between tree size and interpretability was also identified in the survey by Piltaver et al.: the authors found that if the respondent does not necessarily need to use the whole tree (e.g. they are manually using the tree for prediction), then the number of leaves is not as important as the depth of the deepest leaf required to answer the respondent’s question (as measured by both a subjective ”question difficulty” rating and the time required to answer a question). Therefore, depending on the task, a large tree may not be as uninterpretable if the most common use cases are closer to the root. Recently, Hwang et al. (2020) proposed a splitting criterion that tackles this same issue: instead of aiming to reduce the combined impurity of two partitions (as is usual in the traditional methods), their criterion focuses on only one of the partitions having low impurity and large observation coverage. As a result, the broadest and most accurate rules are pushed to the top of the tree, reducing the user’s need to investigate deeper leaves.

5.2 Multivariate splits

Another aspect that affects interpretability is the presence of multivariate splits: logical tests that involve more than one variable (discussed in Sect. 3.1 and illustrated in Fig. 5b). At first, their presence seems to harm interpretability, since multivariate splits are less interpretable than univariate ones (as can be easily illustrated by two hypothetical splits in a medical task: “temperature $\leq$ 38” and “$1.2 \times \text{age} + 0.3 \times \text{temperature} \leq 50$”). However, as noted by Brodley and Utgoff (1995), there is a flip-side to the inclusion of multivariate splits: the higher approximation ability of multivariate tree splits may make the trees much shallower than their univariate counterparts (as illustrated in Fig. 5), which would consequently increase the interpretability related to tree size. These opposite results complicate any claim to the general interpretability of these splits. However, it can also be argued that even a single multivariate split is enough to compromise the interpretability of the entire tree, because if the included multivariate function is uninterpretable, then every path that passes through it will contain an uninterpretable step—therefore harming what has been called the *decomposability* aspect of interpretable predictions, meaning that every part of the model must admit an intuitive explanation (Lipton 2018).

This negative impact on interpretability can be mitigated in several ways, the most common of which is to use multivariate splits that are simpler and more interpretable, such as linear functions. The splits can be further simplified through the use of "feature sparseness" techniques that reduce the number of attributes involved: for example, CART-LC (Breiman et al. 1984) employs a backward elimination process to remove attributes that do not contribute much to accuracy, whereas GUIDE (Loh 2009) considers only bivariate splits from the beginning. More recently, Sok et al. (2015) tackled sparse multivariate splits by using a variant of LDA called Sparse LDA (Clemmensen et al. 2011), while Wang et al. (2015a) employed sparse additive models. Another way to improve the interpretability of these trees is to focus on alternative ways of split interpretation: Fuzzy Rule-based DTs (Wang et al. 2015c) represent non-linear splits by using interpretable fuzzy rules, while Paez et al. (2019) proposed a visualization tool called "decision charts" that allows the user to visually interpret such splits.

5.3 Soft splits

Another design decision that affects interpretability is the possible inclusion of soft splits: logical tests that attribute a weight to each outgoing branch, instead of resulting in mutually exclusive outcomes (like the ones seen in traditional trees, called *crisp*). As discussed in Sect. 3.4.3 and illustrated in Fig. 7, the prediction step of such trees usually involves calculating the weight of every leaf, and either averaging the results or selecting the leaf with maximum weight. As a consequence, soft splits deal a huge blow to the *simulatability* aspect of interpretability (Lipton 2018), because to manually use such a tree for prediction, a user has to simulate every path in the tree while keeping track of all the weights.

However, soft splits also have a positive contribution: if the prediction is done via averaging, then soft splits restore a notion of locality that is absent in traditional splits (Suarez and Lutsko 1999). For example, consider the first tree in Fig. 4: a patient with a temperature of "38.00" will be classified as healthy, whereas the same patient with a temperature of "38.01" would be classified as sick. This difference could plausibly reduce the user's trust in the model, because similar inputs are expected to result in similar outputs (Alvarez-Melis and Jaakkola 2018). A soft split, in contrast, would avoid this problem entirely, because the slight temperature difference would be manifested as a slight change in the tree's weights, and therefore in its final continuous output. As such, if the domain of interest cannot be delimited by sharp boundaries, the soft approach might be more interpretable to the users. It is noteworthy that this is the domain where fuzzy logic is advantageous, as it captures the cognitive uncertainty of categories (Yuan and Shaw 1995), and indeed, the sub-field of DT research known as fuzzy trees (Sosnowski and Gadomer 2019) also employs soft splits (Yuan and Shaw 1995; Janikow 1998; Suarez and Lutsko 1999).

5.4 Other issues

Some trees, such as the ones obtained via gradient-based optimization (seen in Sect. 3.4.3), employ splits that are both soft and multivariate. This results in a model similar to a neural network which cannot be interpreted in the traditional DT way. An intuitive solution is to simply revert to the traditional structure: Silva et al. (2020) proposed inducing a multivariate soft tree using gradient-based techniques and then transforming it into a univariate crisp tree by selecting the attributes with higher weights. A more common solution is aiming at *post hoc interpretations* (Lipton 2018) of the resulting trees through two main approaches: either by focusing on computer vision tasks, in which the multivariate weights map to features that can be visually interpreted (Frosst and Hinton 2017; Hehn et al. 2020), or by labeling the tree's paths so that each path can be interpreted as representative of a concept (e.g. the left subtree handles man-made objects, while the right handles animals) (Tanno et al. 2019; Wan et al. 2020).

Another issue that affects interpretability is the instability of traditional DT algorithms (Li and Belford 2002): the notion that small changes in the training data might result in the induction of substantially different trees. Instability occurs for two main reasons: first, there might be near-ties during the selection of each split, which means that a split being selected depends upon a small fraction of the data. Second, a single different split during induction results in different subtrees, therefore propagating the difference throughout the model. Instability affects interpretability because it might decrease trust in the training algorithm, since humans expect that similar inputs result in similar outputs (Alvarez-Melis and Jaakkola 2018).

To tackle instability, Wang et al. (2014) proposed a splitting criterion that better disambiguates nearly-tied splits. This is done by introducing the concept of "segments", which allows the criterion to consider not only the purity of each partition but also the label distributions. Their criterion reportedly achieves less unstable trees with higher generalization and smaller size. Subsequently, Yan et al. (2019) extended this approach by proposing two novel splitting criteria, SPES1 and SPES2, both based on the idea of surpassing the previous approach through a better balance of "traditional splitting measures" and "number of segments". On a more theoretical front, Baranauskas (2015) showed through simulation that for traditional splitting criteria, the number of nearly-tied splits grows with the number of classes, with instability also depending on the imbalance of the dataset.

Several other issues fall under the general notion of interpretability. Since traditional trees do not capture causality, they are not useful for providing insight into the causal relationships of the data. To tackle this, Li et al. (2017) proposed a splitting criterion for inducing causal DTs. On the other hand, Johansson et al. (2018) aimed at improving trust in regression trees by guaranteeing a probability of error for each inference, through a framework called conformal prediction. Lastly, Carreira-Perpinan and Hada (2021) proposed an exact algorithm for finding counterfactual explanations in both univariate and multivariate DTs, therefore allowing the user to understand how the input could be changed to obtain a more desirable outcome.

6 Future directions

As shown throughout the paper, the field of DT research is still being actively researched, with proposals being made across many different fronts. This section points out some future trends and interesting problems.

- *Applications in Industry 4.0* With the growth of the IoTs and Industry 4.0 as a whole, there will be increasingly more opportunities to implement automated decision processes in day-to-day life, and it is expected that DTs will play an important role in many of those applications, especially those in which interpretability is a key concern. Recent applications of DTs in the industry range from validating data from smart energy meters (Elsisi et al. 2021) to detecting motor faults and cyber-attacks (Tran et al. 2021), and many more are expected to be seen.

- *Optimal trees* Currently, optimal trees are one of the most researched topics in the field. These methods sidestep the sub-optimality concerns that have always been present in DT research, and therefore offer an exciting new paradigm with limits that are still being actively explored. The major challenge is scalability: since guaranteeing optimality is much more expensive than settling for a heuristical solution, optimal trees are usually infeasible for larger datasets, a limitation that traditional trees do not have. Specialized algorithms such as GOSDT and DL8.5 appear to be paving the way in this regard, in contrast to approaches that employ generic commercial solvers.
- *Gradient-based trees* It is expected that gradient-based trees will continue to receive some attention, as researchers of the neural networks community attempt to find new ways to combine the interpretability of DTs with the recent developments in neural networks. Although there is fertile ground to be found in inducing more accurate trees by proposing novel gradient-based algorithms, the biggest challenge is not in reaching the accuracy of a neural network, but instead in maintaining the interpretability of the tree, as most gradient-based approaches employ soft multivariate splits that preclude any traditional notion of interpretability. Therefore, an interesting problem is how to take advantage of gradient-based optimization while keeping the final model interpretable.
- *Improving accuracy without compromising interpretability* As discussed throughout the survey, some DT design decisions generally improve approximation capacity while reducing interpretability (e.g. multivariate splits, model leaves, and larger tree size). This conflict hints at an accuracy-interpretability trade-off, but it is not clear that this trade-off is inevitable; indeed, in many situations, it is known to be more myth than fact (Rudin 2019). Therefore, an interesting challenge going forward is how to improve the accuracy of DT models without compromising that quality which is usually lauded as their key advantage: their transparency. This challenge can be handled in two ways: either by inducing more accurate models without significantly changing the structure of traditional trees, or by proposing new ways to interpret non-traditional tree structures. The former is the most common, exemplified through proposals of new splitting criteria, pruning techniques, or optimization methods (such as optimal trees). The latter is less explored, but with much potential nonetheless: recent works have explored new visualization tools (Paez et al. 2019), post hoc labeling techniques (Wan et al. 2020), restricting multivariate splits to user-provided functions (Paez et al. 2019), and exploiting novel representations for multivariate splits (Wang et al. 2015c). Future research could follow along both these paths.
- *Large-scale comparison studies* As pointed out throughout the survey, many different lines of DT research share the same goals: for example, many evolutionary, optimal, and gradient-based algorithms define their main motivation as "surpassing the sub-optimality of greedy decision trees". However, these methods are rarely compared against each other: most papers provide comparisons only to traditional methods such as CART and C4.5, and sometimes to a few other methods in the same family. Because of this lack of communication between different approaches, it is hard to ascertain what are the best algorithms in the field as a whole, which is why most users end up defaulting to traditional algorithms like CART and C4.5. As noted in the introduction, a comprehensive study of this kind is difficult, since many papers do not make their code openly available and/or provide empirical results on different groups of datasets.

These points were brought up by Rusch and Zeileis (2014) and they continue to be an issue, even though some researchers in the field have started to publish code alongside the papers themselves. Still, a large-scale study across many datasets would be valuable going forwards, as the community would be better able to understand the limitations and advantages of each family of algorithms.

7 Conclusion

Since the introduction of the first regression tree algorithm in the seminal paper by Morgan and Sonquist, close to six decades have passed; nevertheless, DTs have remained a widely-used model and an actively researched subject. This paper attempted to review the advances of the last decade in the field, since the most recent surveys do not feature topics like optimal trees and interpretability, which have gained steam in the last few years. In this scenario where increasingly more attention is given to interpretable AI and DT research, the authors hope that this review serves as a good entry point for both researchers and newcomers to the machine learning community.

Acknowledgements Both authors are grateful to the anonymous reviewers for their valuable suggestions and feedback.

Funding This work was partially funded by the Brazilian research agencies CNPq—National Council for Scientific and Technological Development (Grant Number 306258/2019-6), FAPERJ—Foundation for Research Support of Rio de Janeiro State (Grant Number E-26/200.840/2021), and a Ph.D. Scholarship from CAPES (PROEX).

Data availibility Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Adibi MA (2019) Single and multiple outputs decision tree classification using bi-level discrete-continues genetic algorithm. Pattern Recognit Lett 128:190–196. https://doi.org/10.1016/j.patrec.2019.09.001

Aghaei S, Azizi MJ, Vayanos P (2019) Learning optimal and fair decision trees for non-discriminative decision-making. In: Proceedings of the AAAI conference on artificial intelligence, vol 33(01), pp 1418–1426. https://doi.org/10.1609/aaai.v33i01.33011418

Aglin G, Nijssen S, Schaus P (2020) Learning optimal decision trees using caching branch-and-bound search. In: Proceedings of the AAAI conference on artificial intelligence, vol 34(04), pp 3146–3153. https://doi.org/10.1609/aaai.v34i04.5711

Alvarez-Melis D, Jaakkola TS (2018) On the robustness of interpretability methods. arXiv:1806.08049 [cs, stat]

Amodei D, Ananthanarayanan S, Anubhai R et al (2016) Deep speech 2: end-to-end speech recognition in English and Mandarin. In: International conference on machine learning, PMLR, pp 173–182

Angelino E, Larus-Stone N, Alabi D et al (2017) Learning certifiably optimal rule lists. https://doi.org/10.1145/3097983.3098047

Avellaneda F (2020) Efficient inference of optimal decision trees. In: Proceedings of the AAAI conference on artificial intelligence, vol 34(04), pp 3195–3202. https://doi.org/10.1609/aaai.v34i04.5717

Baranauskas JA (2015) The number of classes as a source for instability of decision tree algorithms in high dimensional datasets. Artif Intell Rev 43(2):301–310. https://doi.org/10.1007/s10462-012-9374-7

Barros RC, Basgalupp MP, de Carvalho ACPLF et al (2012) A survey of evolutionary algorithms for decision-tree induction. IEEE Trans Syst Man Cybern C 42(3):291–312. https://doi.org/10.1109/TSMCC.2011.2157494

Barros RC, de Carvalho ACPLF, Freitas AA (2015) Automatic design of decision-tree induction algorithms. Springer Briefs in computer science. Springer. https://doi.org/10.1007/978-3-319-14231-9

Bennett KP (1992) Decision tree construction via linear programming, Technical report. University of Wisconsin-Madison Department of Computer Sciences

Bennett KP, Blue JA (1996) Optimal decision trees. Technical report, R.P.I. Math Report No. 214. Rensselaer Polytechnic Institute

Bertsimas D, Dunn J (2017) Optimal classification trees. Mach Learn 106(7):1039–1082. https://doi.org/10.1007/s10994-017-5633-9

Bertsimas D, Dunn J, Mundru N (2019) Optimal prescriptive trees 1(2):164–183. https://doi.org/10.1287/ijoo.2018.0005

Bessiere C, Hebrard E, O'Sullivan B (2009) Minimising decision tree size as combinatorial optimisation. In: Gent IP (ed) Principles and practice of constraint programming—CP 2009. Lecture notes in computer science, vol 5732. Springer, Berlin, pp 173–187. https://doi.org/10.1007/978-3-642-04244-7_{1}6

Blanquero R, Carrizosa E, Molero-Río C et al (2020) Sparsity in optimal randomized classification trees. Eur J Oper Res 284(1):255–272. arXiv: 2002.09191

Blanquero R, Carrizosa E, Molero-Río C et al (2021) Optimal randomized classification trees. Comput Oper Res 132(105):281. https://doi.org/10.1016/j.cor.2021.105281

Blockeel H, Raedt LD, Ramon J (1998) Top-down induction of clustering trees. In: Proceedings of the fifteenth international conference on machine learning, 1998, pp 55–63

Bojarski M, Del Testa D, Dworakowski D et al (2016) End to end learning for self-driving cars. arXiv preprint. arXiv:1604.07316

Breiman L, Friedman JH (1988) Tree-structured classification via generalized discriminant analysis: comment. J Am Stat Assoc 83(403):725–727

Breiman L, Friedman J, Stone CJ et al (1984) Classification and regression trees. Taylor & Francis, Boca Raton

Breslow LA, Aha DW (1997) Simplifying decision trees: a survey. Knowl Eng Rev 12(01):1–40. https://doi.org/10.1017/S0269888997000015

Brodley CE, Utgoff PE (1995) Multivariate decision trees. Mach Learn 19(1):45–77. https://doi.org/10.1007/BF00994660

Broelemann K, Kasneci G (2019) A gradient-based split criterion for highly accurate and transparent model trees. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, 2019, pp 2030–2037. https://doi.org/10.24963/ijcai.2019/281

Brunello A, Marzano E, Montanari A et al (2017) Decision tree pruning via multi-objective evolutionary computation. Int J Mach Learn Comput 7(6):167–175. https://doi.org/10.18178/ijmlc.2017.7.6.641

Cao-Van K, De Baets B (2003) Growing decision trees in an ordinal setting. Int J Intell Syst 18(7):733–750. https://doi.org/10.1002/int.10113

Carreira-Perpinan MA, Hada SS (2021) Counterfactual explanations for oblique decision trees: exact, efficient algorithms. In: Proceedings of the AAAI conference on artificial intelligence, 2021, vol 35(8), pp 6903–6911

Carrizosa E, Molero-Río C, Romero Morales D (2021) Mathematical optimization in classification and regression trees. TOP 29(1):5–33. https://doi.org/10.1007/s11750-021-00594-1

Chabbouh M, Bechikh S, Hung CC et al (2019) Multi-objective evolution of oblique decision trees for imbalanced data binary classification. Swarm Evol Comput 49:1–22. https://doi.org/10.1016/j.swevo.2019.05.005

Chen YL, Wu CC, Tang K (2016) Time-constrained cost-sensitive decision tree induction. Inf Sci 354:140–152. https://doi.org/10.1016/j.ins.2016.03.022

Clemmensen L, Hastie T, Witten D et al (2011) Sparse discriminant analysis. Technometrics 53(4):406–413. https://doi.org/10.1198/TECH.2011.08118

Correa Bahnsen A, Aouada D, Ottersten B (2015) Example-dependent cost-sensitive decision trees. Expert Syst Appl 42(19):6609–6619. https://doi.org/10.1016/j.eswa.2015.04.042

Czajkowski M, Kretowski M (2016) The role of decision tree representation in regression problems—an evolutionary perspective. Appl Soft Comput 48:458–475. https://doi.org/10.1016/j.asoc.2016.07.007

Czajkowski M, Jurczuk K, Kretowski M (2015) A parallel approach for evolutionary induced decision trees. MPI+OpenMP implementation. In: Rutkowski L, Korytkowski M, Scherer R et al (eds)

Artificial intelligence and soft computing. Lecture notes in computer science, vol 9119. Springer, Cham, pp 340–349. https://doi.org/10.1007/978-3-319-19324-3_31

Demirović E, Stuckey PJ (2021) Optimal decision trees for nonlinear metrics. In: Proceedings of the AAAI conference on artificial intelligence, 2021, vol 35(5), pp 3733–3741

Demirović E, Lukina A, Hebrard E et al (2021) MurTree: optimal classification trees via dynamic programming and search. arXiv:2007.12652 [cs, stat] ArXiv: 2007.12652

Dunn JW (2018) Optimal trees for prediction and prescription. PhD Thesis, Massachusetts Institute of Technology

Elsisi M, Mahmoud K, Lehtonen M et al (2021) Reliable industry 4.0 based on machine learning and IoT for analyzing, monitoring, and securing smart meters. Sensors 21(2):487

Esmeir S, Markovitch S (2007) Anytime learning of decision trees. J Mach Learn Res 8:891–933

Firat M, Crognier G, Gabor AF et al (2020) Column generation based heuristic for learning classification trees. Comput Oper Res 116(104):866. https://doi.org/10.1016/j.cor.2019.104866

Fraiman R, Ghattas B, Svarc M (2013) Interpretable clustering using unsupervised binary trees. Adv Data Anal Classif 7(2):125–145. https://doi.org/10.1007/s11634-013-0129-3

Frank E, Mayo M, Kramer S (2015) Alternating model trees. In: Proceedings of the 30th annual ACM symposium on applied computing, Salamanca, Spain. ACM, pp 871–878. https://doi.org/10.1145/2695664.2695848

Freitas AA (2014) Comprehensible classification models: a position paper. ACM SIGKDD Explor Newsl 15(1):1–10

Freund Y, Mason L (1999) The alternating decision tree learning algorithm. In: Proceedings of the sixteenth international conference on machine learning, 1999, pp 124–133

Frosst N, Hinton G (2017) Distilling a neural network into a soft decision tree. arXiv:1711.09784 [cs, stat]

Garcia Leiva R, Fernandez Anta A, Mancuso V et al (2019) A novel hyperparameter-free approach to decision tree construction that avoids overfitting by design. IEEE Access 7:99978–99987. https://doi.org/10.1109/ACCESS.2019.2930235

Ghattas B, Michel P, Boyer L (2017) Clustering nominal data using unsupervised binary decision trees: comparisons with the state of the art methods. Pattern Recognit 67:177–185. https://doi.org/10.1016/j.patcog.2017.01.031

Gleser MA, Collen MF (1972) Towards automated medical decisions. Comput Biomed Res 5(2):180–189. https://doi.org/10.1016/0010-4809(72)90080-8

Günlük O, Kalagnanam J, Li M et al (2021) Optimal decision trees for categorical data via integer programming. J Glob Optim. https://doi.org/10.1007/s10898-021-01009-y

Hastie T, Tibshirani R, Friedman JH et al (2009) The elements of statistical learning: data mining, inference, and prediction, vol 2. Springer, New York

Heath D, Kasif S, Salzberg S (1993) Induction of oblique decision trees. J Artif Intell Res 1993:1002–1007

Hehn TM, Kooij JFP, Hamprecht FA (2020) End-to-end learning of decision trees and forests. Int J Comput Vis 128(4):997–1011. https://doi.org/10.1007/s11263-019-01237-6

Hu Q, Guo M, Yu D et al (2010) Information entropy for ordinal classification. Sci China Inf Sci 53(6):1188–1200. https://doi.org/10.1007/s11432-010-3117-7

Hu Q, Che X, Zhang L et al (2012) Rank entropy-based decision trees for monotonic classification. IEEE Trans Knowl Data Eng 24(11):2052–2064. https://doi.org/10.1109/TKDE.2011.149

Hu X, Rudin C, Seltzer M (2019) Optimal sparse decision trees. In: Advances in neural information processing systems (NeurIPS)

Hu H, Siala M, Hebrard E, et al (2020) Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence, pp 1170–1176. ISSN: 1045-0823. https://doi.org/10.24963/ijcai.2020/163

Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501

Hwang S, Yeo HG, Hong JS (2020) A new splitting criterion for better interpretable trees. IEEE Access 8:62762–62774. https://doi.org/10.1109/ACCESS.2020.2985255

Hyafil L, Rivest RL (1976) Constructing optimal binary decision trees is NP-complete. Inf Process Lett 5(1):15–17. https://doi.org/10.1016/0020-0190(76)90095-8

Ikonomovska E, Gama J, Džeroski S (2011) Learning model trees from evolving data streams. Data Min Knowl Discov 23(1):128–168. https://doi.org/10.1007/s10618-010-0201-y

Iorio C, Aria M, D'Ambrosio A et al (2019) Informative trees by visual pruning. Expert Syst Appl 127:228–240. https://doi.org/10.1016/j.eswa.2019.03.018

Irsoy O, Yıldız OT, Alpaydın E (2012) Soft decision trees. In: Proceedings of the 21st international conference on pattern recognition (ICPR2012), 2012, pp 1819–1822

Irsoy O, Yildiz OT, Alpaydin E (2014) Budding trees. In: 2014 22nd international conference on pattern recognition, Stockholm, Sweden, 2014. IEEE, pp 3582–3587. https://doi.org/10.1109/ICPR.2014.616

Janikow C (1998) Fuzzy decision trees: issues and methods. IEEE Trans Syst Man Cybern B 28(1):1–14. https://doi.org/10.1109/3477.658573

Janota M, Morgado A (2020) SAT-based encodings for optimal decision trees with explicit paths. In: Theory and applications of satisfiability testing—SAT 12178, pp 501–518. https://doi.org/10.1007/978-3-030-51825-7_35

Johansson U, Linusson H, Löfström T et al (2018) Interpretable regression trees using conformal prediction. Expert Syst Appl 97:394–404. https://doi.org/10.1016/j.eswa.2017.12.041

Jordan MI, Jacobs RA (1994) Hierarchical mixtures of experts and the EM algorithm. Neural Comput 6(2):181–214. https://doi.org/10.1162/neco.1994.6.2.181

Jurczuk K, Czajkowski M, Kretowski M (2017) Evolutionary induction of a decision tree for large-scale data: a GPU-based approach. Soft Comput 21(24):7363–7379. https://doi.org/10.1007/s00500-016-2280-1

Karabadji NEI, Seridi H, Bousetouane F et al (2017) An evolutionary scheme for decision tree construction. Knowl Based Syst 119:166–177. https://doi.org/10.1016/j.knosys.2016.12.011

Kim K (2016) A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree. Pattern Recognit 60:157–163. https://doi.org/10.1016/j.patcog.2016.04.016

Kim H, Loh WY (2001) Classification trees with unbiased multiway splits. J Am Stat Assoc 96(454):589–604

Kohavi R (1996) Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the second international conference on knowledge discovery and data mining. KDD'96, 1996. AAAI Press, pp 202–207

Kotsiantis SB (2013) Decision trees: a recent overview. Artif Intell Rev 39(4):261–283. https://doi.org/10.1007/s10462-011-9272-4

Kretowski M, Grzes M (2007) Evolutionary induction of mixed decision trees. IJDWM 3:68–82. https://doi.org/10.4018/jdwm.2007100104

Landwehr N, Hall M, Frank E (2005) Logistic model trees. Mach Learn 59(1):161–205. https://doi.org/10.1007/s10994-005-0466-3

Levatić J, Ceci M, Kocev D et al (2017) Semi-supervised classification trees. J Intell Inf Syst 49(3):461–486. https://doi.org/10.1007/s10844-017-0457-4

Levatić J, Kocev D, Ceci M et al (2018) Semi-supervised trees for multi-target regression. Inf Sci 450:109–127. https://doi.org/10.1016/j.ins.2018.03.033

Li RH, Belford GG (2002) Instability of decision tree classification algorithms. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '02, New York, NY, USA. Association for Computing Machinery, pp 570–575. https://doi.org/10.1145/775047.775131

Li X, Zhao H, Zhu W (2015) A cost sensitive decision tree algorithm with two adaptive mechanisms. Knowl Based Syst 88:24–33. https://doi.org/10.1016/j.knosys.2015.08.012

Li J, Ma S, Le T et al (2017) Causal decision trees. IEEE Trans Knowl Data Eng 29(2):257–271. https://doi.org/10.1109/TKDE.2016.2619350

Lin J, Zhong C, Hu D et al (2020) Generalized and scalable optimal sparse decision trees. In: Proceedings of the 37th international conference on machine learning, 2020. PMLR, pp 6150–6160. ISSN: 2640-3498

Lipton ZC (2018) The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. Queue 16(3):31–57

Liu B, Xia Y, Yu PS (2000) Clustering through decision tree construction. In: Proceedings of the ninth international conference on information and knowledge management, CIKM '00, New York, NY, USA, 2000. Association for Computing Machinery, pp 20–29. https://doi.org/10.1145/354756.354775

Loh WY (2009) Improving the precision of classification trees. Ann Appl Stat. https://doi.org/10.1214/09-AOAS260

Loh WY (2011) Classification and regression trees. Wiley Interdiscip Rev Data Min Knowl Discov 1(1):14–23

Loh WY (2014) Fifty years of classification and regression trees. Int Stat Rev 82(3):329–348. https://doi.org/10.1111/insr.12016

Loh WY, Shih YS (1997) Split selection methods for classification trees. Stat Sin 7(4):815–840

Loh WY, Vanichsetakul N (1988) Tree-structured classification via generalized discriminant analysis. J Am Stat Assoc 83(403):715–725. https://doi.org/10.1080/01621459.1988.10478652

Lomax S, Vadera S (2013) A survey of cost-sensitive decision tree induction algorithms. ACM Comput Surv (CSUR). https://doi.org/10.1145/2431211.2431215

López-Chau A, Cervantes J, López-García L et al (2013) Fisher's decision tree. Expert Syst Appl 40(16):6283–6291. https://doi.org/10.1016/j.eswa.2013.05.044

Manwani N, Sastry PS (2012) Geometric decision tree. IEEE Trans Syst Man Cybern B 42(1):181–192. https://doi.org/10.1109/TSMCB.2011.2163392

Marsala C, Petturiti D (2015) Rank discrimination measures for enforcing monotonicity in decision tree induction. Inf Sci 291:143–171. https://doi.org/10.1016/j.ins.2014.08.045

Meisel W, Michalopoulos D (1973) A partitioning algorithm with application in pattern classification and the optimization of decision trees. IEEE Trans Comput C–22(1):93–103. https://doi.org/10.1109/T-C.1973.223603

Mingers J (1989) An empirical comparison of pruning methods for decision tree induction. Mach Learn 4(2):227–243. https://doi.org/10.1023/A:1022604100933

Mitchell M (1998) An introduction to genetic algorithms. MIT Press, Cambridge

Molnar C (2022) Interpretable machine learning, 2nd edn. christophm.github.io/interpretable-ml-book/

Morgan JN, Sonquist JA (1963) Problems in the analysis of survey data, and a proposal. J Am Stat Assoc 58(302):415–434. https://doi.org/10.1080/01621459.1963.10500855

Mu Y, Liu X, Wang L et al (2020) A parallel fuzzy rule-base based decision tree in the framework of Map-Reduce. Pattern Recognit 103(107):326. https://doi.org/10.1016/j.patcog.2020.107326

Murthy SK (1998) Automatic construction of decision trees from data: a multi-disciplinary survey. Data Min Knowl Discov 2(4):345–389. https://doi.org/10.1023/a:1009744630224

Murthy S, Salzberg S (1995a) Lookahead and pathology in decision tree induction. In: Proceedings of the 14th international joint conference on artificial intelligence, IJCAI'95, vol 2. Morgan Kaufmann Publishers Inc., San Francisco, pp 1025–1031

Murthy SK, Salzberg S (1995b) Decision tree induction: how effective is the greedy heuristic? p 6

Murthy S, Kasif S, Salzberg S et al (1993) OC1: a randomized induction of oblique decision trees. In: AAAI, Citeseer, pp 322–327

Narodytska N, Ignatiev A, Pereira F et al (2018) Learning optimal decision trees with SAT. In: Proceedings of the twenty-seventh international joint conference on artificial intelligence. International Joint Conferences on Artificial Intelligence Organization, Stockholm, pp 1362–1368. https://doi.org/10.24963/ijcai.2018/189

Nijssen S, Fromont E (2010) Optimal constraint-based decision tree induction from itemset lattices. Data Min Knowl Discov 21(1):9–51. https://doi.org/10.1007/s10618-010-0174-x

Norouzi M, Collins M, Johnson MA et al (2015) Efficient non-greedy optimization of decision trees. In: Advances in neural information processing systems, vol 28. Curran Associates, Inc., Red Hook

Norton SW (1989) Generating better decision trees. In: IJCAI, pp 800–805

Nunes C, De Craene M, Langet H et al (2020) Learning decision trees through Monte Carlo tree search: an empirical evaluation. WIREs Data Min Knowl Discov. https://doi.org/10.1002/widm.1348

Paez A, López F, Ruiz M et al (2019) Inducing non-orthogonal and non-linear decision boundaries in decision trees via interactive basis functions. Expert Syst Appl 122:183–206. https://doi.org/10.1016/j.eswa.2018.12.041

Pei S, Hu Q, Chen C (2016) Multivariate decision trees with monotonicity constraints. Knowl Based Syst 112:14–25

Piltaver R, Luštrek M, Gams M et al (2016) What makes classification trees comprehensible? Expert Syst Appl 62:333–346. https://doi.org/10.1016/j.eswa.2016.06.009

Potharst R, Bioch JC (1999) A decision tree algorithm for ordinal classification. In: Goos G, Hartmanis J, van Leeuwen J et al (eds) Advances in intelligent data analysis. Lecture notes in computer science, vol 1642. Springer, Berlin, pp 187–198. https://doi.org/10.1007/3-540-48412-4_16

Provost F, Domingos P (2003) Tree induction for probability-based ranking. Mach Learn 52(3):199–215. https://doi.org/10.1023/A:1024099825458

Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106. https://doi.org/10.1007/BF00116251

Quinlan JR (1987) Simplifying decision trees. Int J Man–Mach Stud 27(3):221–234. https://doi.org/10.1016/S0020-7373(87)80053-6

Quinlan JR (1992) Learning with continuous classes. In: 5th Australian joint conference on artificial intelligence. World Scientific, pp 343–348

Ragavan H, Rendell LA (1993) Lookahead feature construction for learning hard concepts. In: Proceedings of the tenth international conference on international conference on machine learning, ICML'93, 1993. Morgan Kaufmann Publishers, Inc., San Francisco, pp 252–259

Rhuggenaath J, Zhang Y, Akcay A et al (2018) Learning fuzzy decision trees using integer programming. In: 2018 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–8. https://doi.org/10.1109/FUZZ-IEEE.2018.8491636

Rokach L, Maimon OZ (2007) Data mining with decision trees: theory and applications. World Scientific, Singapore

Roscher R, Bohn B, Duarte MF et al (2020) Explainable machine learning for scientific insights and discoveries. IEEE Access 8:42200–42216. https://doi.org/10.1109/ACCESS.2020.2976199

Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1(5):206–215. https://doi.org/10.1038/s42256-019-0048-x

Rusch T, Zeileis A (2014) Discussion on fifty years of classification and regression trees. Int Stat Rev 82(3):361–367

Sarker IH (2021) Machine learning: algorithms, real-world applications and research directions. SN Comput Sci 2(3):1–21

Schidler A, Szeider S (2021) SAT-based decision tree learning for large data sets. In: Proceedings of the AAAI conference on artificial intelligence, vol 35(5), pp 3904–3912

Silva A, Gombolay M, Killian T et al (2020) Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In: Proceedings of the twenty third international conference on artificial intelligence and statistics, 2020. PMLR, pp 1855–1865. ISSN: 2640-3498

Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489

Sok HK, Ooi MPL, Kuang YC (2015) Sparse alternating decision tree. Pattern Recognit Lett 60–61:57–64. https://doi.org/10.1016/j.patrec.2015.03.002

Sok HK, Ooi MPL, Kuang YC et al (2016) Multivariate alternating decision trees. Pattern Recognit 50:195–209. https://doi.org/10.1016/j.patcog.2015.08.014

Sosnowski ZA, Gadomer Lu (2019) Fuzzy trees and forests—review. Wiley Interdiscip Rev Data Min Knowl Discov. https://doi.org/10.1002/widm.1316

Suarez A, Lutsko J (1999) Globally optimal fuzzy decision trees for classification and regression. IEEE Trans Pattern Anal Mach Intell 21(12):1297–1311. https://doi.org/10.1109/34.817409

Tanha J, van Someren M, Afsarmanesh H (2017) Semi-supervised self-training for decision tree classifiers. Int J Mach Learn Cybern 8(1):355–370

Tanno R, Arulkumaran K, Alexander D et al (2019) Adaptive neural trees. In: Proceedings of the 36th international conference on machine learning, 2019. PMLR, pp 6166–6175. ISSN: 2640-3498

Tharwat A, Gaber T, Ibrahim A et al (2017) Linear discriminant analysis: a detailed tutorial. AI Commun 30(2):169–190. https://doi.org/10.3233/AIC-170729

Tran MQ, Elsisi M, Mahmoud K et al (2021) Experimental setup for online fault diagnosis of induction machines via promising IoT and machine learning: towards industry 4.0 empowerment. IEEE Access 9:115429–115441

Verwer S, Zhang Y (2017) Learning decision trees with flexible constraints and objectives using integer optimization. In: Salvagnin D, Lombardi M (eds) Integration of AI and OR techniques in constraint programming. Lecture notes in computer science, vol 10335. Springer, Cham, pp 94–103. https://doi.org/10.1007/978-3-319-59776-8_8

Verwer S, Zhang Y (2019) Learning optimal classification trees using a binary linear program formulation. In: Proceedings of the AAAI conference on artificial intelligence, vol 33(01), pp 1625–1632. https://doi.org/10.1609/aaai.v33i01.33011624

Wan A, Dunlap L, Ho D et al (2020) NBDT: neural-backed decision trees. arXiv:2004.00221

Wang R, Kwong S, Wang XZ et al (2014) Segment based decision tree induction with continuous valued attributes. IEEE Trans Cybern 45(7):1262–1275

Wang J, Fujimaki R, Motohashi Y (2015a) Trading interpretability for accuracy: oblique treed sparse additive models. In: Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp 1245–1254

Wang R, He YL, Chow CY et al (2015b) Learning ELM-Tree from big data based on uncertainty reduction. Fuzzy Sets Syst 258:79–100. https://doi.org/10.1016/j.fss.2014.04.028

Wang X, Liu X, Pedrycz W et al (2015c) Fuzzy rule based decision trees. Pattern Recognit 48(1):50–59. https://doi.org/10.1016/j.patcog.2014.08.001

Webb GI (1997) Decision tree grafting. In: Proceedings of the fifteenth international joint conference on artificial intelligence, IJCAI'97, vol 2. Morgan Kaufmann Publishers, Inc., San Francisco, pp 846–851

Wickramarachchi D, Robertson B, Reale M et al (2016) HHCART: an oblique decision tree. Comput Stat Data Anal 96:12–23. https://doi.org/10.1016/j.csda.2015.11.006

Wickramarachchi DC, Robertson BL, Reale M et al (2019) A reflected feature space for CART. Aust NZ J Stat 61(3):380–391. https://doi.org/10.1111/anzs.12275

Wu CC, Chen YL, Liu YH et al (2016) Decision tree induction with a constrained number of leaf nodes. Appl Intell 45(3):673–685. https://doi.org/10.1007/s10489-016-0785-z

Wu CC, Chen YL, Tang K (2019) Cost-sensitive decision tree with multiple resource constraints. Appl Intell 49(10):3765–3782. https://doi.org/10.1007/s10489-019-01464-x

Yan J, Zhang Z, Xie L et al (2019) A unified framework for decision tree on continuous attributes. IEEE Access 7:11924–11933. https://doi.org/10.1109/ACCESS.2019.2892083

Yang L, Liu S, Tsoka S et al (2017) A regression tree approach using mathematical programming. Expert Syst Appl 78:347–357. https://doi.org/10.1016/j.eswa.2017.02.013

Yang Y, Morillo IG, Hospedales TM (2018) Deep neural decision trees. arXiv:1806.06988 [cs, stat]

Yuan Y, Shaw MJ (1995) Induction of fuzzy decision trees. Fuzzy Sets Syst 69(2):125–139. https://doi.org/10.1016/0165-0114(94)00229-Z

Zhao H, Li X (2017) A cost sensitive decision tree algorithm based on weighted class distribution with batch deleting attribute mechanism. Inf Sci 378:303–316. https://doi.org/10.1016/j.ins.2016.09.054

Zhou X, Yan D (2019) Model tree pruning. Int J Mach Learn Cybern 10(12):3431–3444. https://doi.org/10.1007/s13042-019-00930-9

Zhu H, Murali P, Phan D et al (2020) A scalable MIP-based method for learning optimal multivariate decision trees. In: Larochelle H, Ranzato M, Hadsell R et al (eds) Advances in neural information processing systems, vol 33. Curran Associates, Inc., Red Hook, pp 1771–1781

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.