Data Mining and Knowledge Discovery (2023) 37:434–475
https://doi.org/10.1007/s10618-022-00884-7

Regularized impurity reduction: accurate decision trees with complexity guarantees

Guangyi Zhang[1] · **Aristides Gionis**[1]

Received: 9 February 2022 / Accepted: 25 October 2022 / Published online: 28 November 2022

Abstract

Decision trees are popular classification models, providing high accuracy and intuitive explanations. However, as the tree size grows the model interpretability deteriorates. Traditional tree-induction algorithms, such as C4.5 and CART, rely on impurity-reduction functions that promote the discriminative power of each split. Thus, although these traditional methods are accurate in practice, there has been no theoretical guarantee that they will produce small trees. In this paper, we justify the use of a general family of impurity functions, including the popular functions of entropy and Gini-index, in scenarios where small trees are desirable, by showing that a simple enhancement can equip them with complexity guarantees. We consider a general setting, where objects to be classified are drawn from an arbitrary probability distribution, classification can be binary or multi-class, and splitting tests are associated with non-uniform costs. As a measure of tree complexity, we adopt the expected cost to classify an object drawn from the input distribution, which, in the uniform-cost case, is the expected number of tests. We propose a tree-induction algorithm that gives a logarithmic approximation guarantee on the tree complexity. This approximation factor is tight up to a constant factor under mild assumptions. The algorithm recursively selects a test that maximizes a greedy criterion defined as a weighted sum of three components. The first two components encourage the selection of tests that improve the balance and the cost-efficiency of the tree, respectively, while the third impurity-reduction component encourages the selection of more discriminative tests. As shown in our empirical evaluation, compared to the original heuristics, the enhanced algorithms strike an excellent balance between predictive accuracy and tree complexity.

Responsible editor: Albrecht Zimmermann and Peggy Cellier.

✉ Guangyi Zhang
guaz@kth.se

Aristides Gionis
argioni@kth.se

[1] Division of Theoretical Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

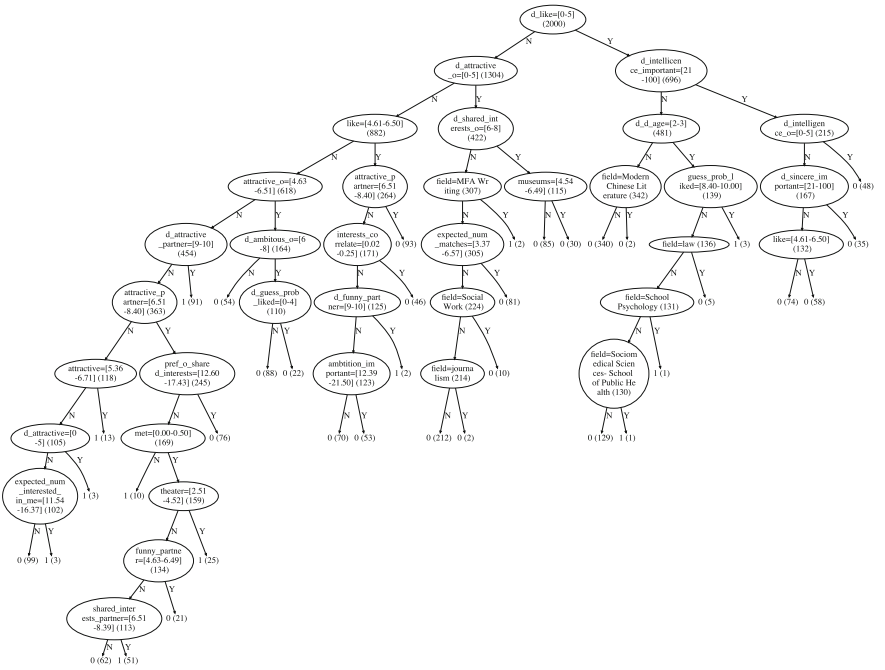
Keywords Decision trees · Impurity functions · Submodularity · Tree complexity · Approximation algorithms

1 Introduction

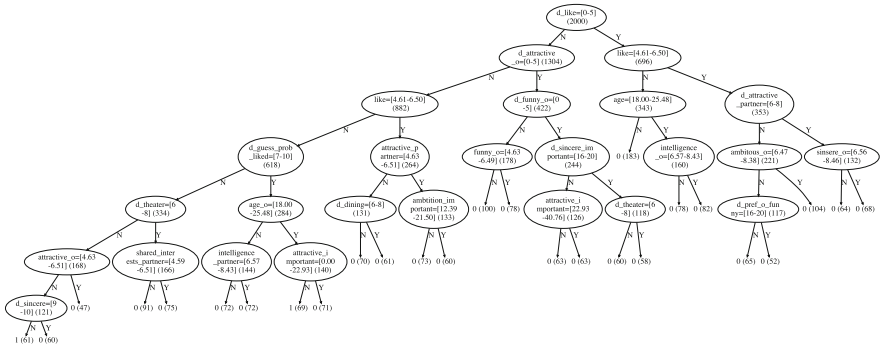
Decision trees are known to provide a good trade off between accuracy and interpretability. However, when their size grows, decision trees become harder to interpret, preventing their deployment in safety-critical applications and in domains where model transparency is highly valued, such as disease diagnosis. As interpretability still remains an ill-defined notion (Lipton 2018), in this paper we consider tree complexity, a commonly-accepted proxy, to quantify interpretability (Freitas 2014; Doshi-Velez and Kim 2017). In addition, low tree-complexity promotes cheaper and faster evaluation. Note that post-pruning techniques, such as the standard minimal cost-complexity pruning (Breiman et al. 1984), are heuristics performed mainly to avoid overfitting. Therefore, in order to produce interpretable trees, we aim for an integrated tree-induction algorithm that considers both the accuracy and complexity of the inferred trees.

More concretely, given a set of labeled objects (examples) drawn from an arbitrary probability distribution, our goal is to learn a decision tree that outputs the correct class of a given input object. Each internal tree node is equipped with a single test, e.g., a projection split along a feature, and each test is associated with a non-uniform cost, i.e., the cost of evaluating the outcome of the test. For example, an input object may represent a person, a test may correspond to a blood sugar test, and one possible outcome can be "high." Our aim is to learn trees that are accurate and have low complexity. The latter complexity objective is measured by the expected cost to classify an object drawn from the input distribution; if all tests incur the same cost, this measure is simply the expected number of tests to classify an object. This complexity measure reflects a form of "local" interpretability: the more tests are involved in an if-then rule for a given object, the more obscure the rule becomes to a user (Freitas 2014). Figure 1 helps demonstrating this intuition by juxtaposing two decision trees with different complexity. Note that non-uniform test costs may arise in different real-world scenarios; for example, in a medical-diagnosis application some tests can be significantly more expensive than others.

The problem of minimizing the expected cost of a tree for perfect class identification has been extensively studied. Typically, the assumption of *realizability* (or consistency) is being made, which states that for every two distinct objects there exists at least one test that can distinguish them. Thus, one can always expand the tree until it classifies every object in the training data perfectly. Then, the goal is to find the tree with the minimum expected cost that classifies each object perfectly. Note that, in practice, the realizability assumption can be easily fulfilled by data preprocessing, as we demonstrate later in our experiments. When each object belongs to a distinct class, the problem is referred to as *entity identification* (EI) (Gupta et al. 2017). Without this restriction, the problem is called *group identification* (GI) (Cicalese et al. 2014). A further generalization that is called *adaptive submodular ranking* (ASR) (Navidi et al. 2020) characterizes the tree-building process as interaction among multiple



(a) Standard C4.5 (AUC ROC 0.779 and expected height 5.9)



(b) Enhanced C4.5 (in this paper) (AUC ROC 0.790 and expected height 4.9)

Fig. 1 Decision trees for predicting if participants would like to see their date again after speed dating. Each internal node includes the test used and the number of participants in parentheses. Leaf nodes make a decision

submodular functions, one for each object, and achieves logarithmic approximation by a greedy algorithm. The above-mentioned works (Gupta et al. 2017; Cicalese et al. 2014; Navidi et al. 2020) consider the problem of building trees for the purposes of exact identification. They do not consider issues of accuracy and overfitting. In fact, exact identification on a set of (training) data leads precisely to overfitting.

The algorithm proposed for the ASR problem by Navidi et al. (2020) provides a very elegant solution for the identification task it has been designed for. However, in practice, it is not suitable for classification tasks in the context of statistical learning, because the chosen tests are geared towards small expected cost and are not necessarily discriminative. Discriminative power is generally measured by the homogeneity of the target variable within a tree node, and is essential for the generalization of model performance over unseen data. Their method selects splits that minimize the number of *heterogeneous pairs* (also known as impure pairs) of objects (Golovin et al. 2010; Cicalese et al. 2014). In Section A we provide a simple example where the criterion favors a non-discriminative (presumably random) test over a discriminative one. While random tests lead to a balanced tree with bounded expected depth, they are not "learning", that is, no statistical dependence is captured between tests and the target variable.

On the other hand, traditional decision-tree methods, such as CART (Breiman et al. 1984) and C4.5 (Quinlan 1993), rely on time-tested impurity-reduction heuristics that yield decision trees with high discriminative power. Although trees produced by these popular methods are accurate in practice, there has been no guarantee on the size, or depth, of the resulting trees. Actually, despite the popularity of these methods, their theoretical properties remain still poorly understood (Bellala et al. 2012; Brutzkus et al. 2019; Blanc et al. 2020).

In this paper we propose a general family of methods that achieve the best of both worlds: *it produces decision trees having both high accuracy and bounded depth*. Our key discovery is that the ASR framework can be extended to effectively analyze a broad range of impurity functions for tree induction.

More formally, we introduce the *non-overfitting group identification* (NGI) problem, which is a natural generalization of group identification (GI), where we further allow early termination during tree expansion to avoid overfitting. We propose a novel greedy algorithm that takes into consideration the impurity reduction and maintains the strong approximation guarantee on the complexity of the resulting tree. Specifically, our greedy algorithm admits the use of a *general family of decomposable impurity functions*, which is defined to be in the form of a weighted sum over impurity scores in each class. This family includes the popular functions of entropy and Gini-index. Therefore, our approach generalizes many traditional tree-induction algorithms such as CART and C4.5 into a complexity-aware method.

In concrete, in this paper we make the following contributions.

- We extend the adaptive submodular ranking (ASR) framework of Navidi et al. (2020) and we propose a novel greedy algorithm to select discriminative tests for the non-overfitting group identification (NGI) problem. Our algorithm offers an asymptotically tight approximation guarantee on the complexity of the inferred tree under mild assumptions.
- We define a general family of decomposable impurity functions, which can be used by our algorithm as a surrogate for discriminative power. As a result, our algorithm generalizes traditional tree-induction algorithms, such as CART and C4.5, into complexity-aware methods.

- We provide a comprehensive experimental evaluation in which we show that the enhanced C4.5 and CART strike an excellent balance between predictive accuracy and tree complexity, compared to their corresponding original heuristics. Furthermore, the ASR formulation yields inferior predictive accuracy, compared to other learning methods. Our implementation is publicly available.[1]

The rest of the paper is organized as follows. The related work is discussed in Sect. 2. The necessary notation and the formal definition of the NGI problem are introduced in Sect. 3. The main algorithm and its theoretical analysis follow in Sects. 4 and 5, respectively. Empirical experiments are conducted in Sect. 6, and we conclude in Sect. 7.

2 Related work

Decision-tree induction. Mainstream algorithms such as C4.5 and CART embrace a top-down greedy approach. Most of the greedy criteria proposed are essentially ad-hoc heuristics for measuring the strength of dependence between tests and the class, with no consideration for tree complexity (Murthy 1998). Theoretical understanding about such greedy methods is still lacking in the literature. A lower bound on the expected tree depth for C4.5 that depends on the shape of a given tree has been developed by Bellala et al. (2012). There also exist some recent results in the field of learning theory (Brutzkus et al. 2019; Blanc et al. 2020).

Tree complexity Popular measures include the number of nodes in the tree, the tree height and the expected path length. The first kind of measures are closer to a notion of "global" interpretability, in the sense that one could inspect the entire tree of a small size, while the second kind of measures provide a notion of "local" interpretability, in the sense that one could explain any given object using a small number of tests. Our choice in the paper, the third kind of a measure, combines elements from both global and local interpretability. First, it obviously enables a form of local interpretability, i.e., a guarantee of a small expected number of tests when explaining a given object. This choice is considered to be more natural and less strict compared to worst-case tree height, as it may not be possible to classify every object using a small number of tests. Second, it also enables a form of global interpretability, as the global model knowledge is acquired by understanding the decision for every example in the dataset, and also it leads to smaller trees in general. Unfortunately, these complexity measures are proven to lead to **NP**-hard tasks (Hancock et al. 1996; Laurent and Rivest 1976). In particular, the expected path-length measure with an arbitrary probability distribution over objects does not admit sub-logarithmic approximation (Chakaravarthy et al. 2007).

Identification The entity identification (EI) problem has been investigated in different contexts, including optimal decision trees, disease/fault diagnosis, and active learning (Adler and Heeringa 2008; Chakaravarthy et al. 2007; Dasgupta 2005; Garey 1972; Guillory and Bilmes 2009; Gupta et al. 2017; Kosaraju et al. 1999). A class-based generalization, the group identification (GI) problem, where objects are partitioned into groups (classes), has also been studied (Bellala et al. 2012; Cicalese et al. 2014; Golovin

[1] https://github.com/Guangyi-Zhang/low-expected-cost-decision-trees.

et al. 2010). The state-of-the-art method achieves $\mathcal{O}(\log n)$-approximation in a general setting with an arbitrary object distribution and non-uniform multi-way testing costs (Cicalese et al. 2014). Our paper further generalizes the latter work by considering the discriminative power of the selected tests. To the best of our knowledge, this is the first work to combine identification problems and traditional tree-induction algorithms.

Stochastic submodular coverage (StoSC) Tree induction can be seen as a sample-based stochastic submodular-coverage problem (Golovin and Krause 2011; Grammel et al. 2016), by relating a realization of items in the StoSC problem to an object in identification problems. The expected cost of a tree is then equivalent to the expected cost in item selection.

Adaptive submodular ranking (ASR) The ASR problem, proposed by Navidi et al. (2020), originates from the line of research of min-sum set cover (Feige et al. 2004; Im et al. 2012), and turns out to generalize the above-mentioned identification problems (Bellala et al. 2012; Cicalese et al. 2014; Golovin et al. 2010). Our formulation follows the framework of ASR, and extends its greedy criterion to incorporate an impurity-reduction component.

3 Problem definition

In this section, we first formalize the *non-overfitting group identification* (NGI) problem, and then define a family of decomposable impurity functions for tree induction.

An instance of the NGI problem is specified by a set of objects $X = \{x_1, ..., x_n\}$, a set of class labels $L = \{\ell_1, ..., \ell_k\}$, and a set of tests $D = \{d_1, ..., d_m\}$. The objects in X are drawn from a probability distribution p, i.e., object x in X occurs with probability $p(x)$. Each object $x \in X$ is associated with a class $\ell(x)$ in L. A test $d \in D$ performed on an object $x \in X$ returns a value $d(x) \in \{1, ..., v_d\}$. We assume that employing test d incurs cost $c(d)$. For simplicity and without loss of generality, we also assume that the cost function c takes integral values. A useful quantity in our later analysis is the minimum object probability $p_{\min} = \min_{x \in X} p(x)$. Finally, we assume that a threshold parameter $\theta \in [0, 1]$ is given as input, which determines a stopping condition for the decision-tree construction, as we will see shortly.

We write $T(X)$ to refer to a decision tree built to classify the objects in X. We omit the reference to the set X when it is clear from the context and just write T. We also write $T(S)$ to refer to a subtree of the decision tree to classify objects in a node S of the tree, where $S \subseteq X$ is the subset of objects. Each internal node S is equipped with a test d in D. Objects in S are partitioned by test d into multiple subnodes according to their testing outcomes $d(x)$. Using this convention we refer to the root of the decision tree simply as X, that is, the complete set of objects to be classified by the tree. Finally, we define $p(S) = \sum_{x \in S} p(x)$.

We stop splitting a node $S \subseteq X$ in the tree T when either (i) the node S is *homogeneous*, i.e., all objects in S belong to the same class, or (ii) the probability $p(S)$ is no greater than the threshold parameter θ, for instance, in the case of uniform p, the node S has at most θn objects. As a surrogate for homogeneity, we adopt a function $\boldsymbol{\phi}$ over pairs of objects. We define $\boldsymbol{\phi}(S)$ to be the number of *heterogeneous* pairs of

objects in the node S, i.e., pairs of objects with distinct classes. Note that $\boldsymbol{\phi}(S) = 0$ when S is homogeneous.

As a measure of complexity for a tree rooted at X, we adopt the measure of *expected cost*, which we denote by $c(T(X))$. In particular, we define $c(T, x)$ as the cost of evaluating an object x in T, which is the sum of costs of all tests that x goes through in T. The expected cost of a tree T for a set of objects X is then defined as $c(T(X)) = \sum_{x \in X} p(x)\, c(T, x)$.

We are now ready to define the NGI problem.

Problem 1 (*Non-overfitting group identification (NCI)*) Given a problem instance $I = (X, L, D, \ell, p, c, \theta)$, with set of objects X, set of class labels L, set of tests D, object labels ℓ, probability distribution p, cost function c, and a threshold θ, find a tree $T(X)$ that minimizes the expected cost $c(T(X))$ and for all leaf nodes S it satisfies either $\boldsymbol{\phi}(S) = 0$ or $p(S) \leq \theta$.

The NGI problem generalizes the GI problem by setting $\theta = 0$, and as stated in Sect. 2, the GI problem is **NP**-hard. Thus, we aim to find a tree T that is an approximate solution, i.e., whose cost $c(T)$ is bounded with respect to the cost $c(T^*)$ of the optimal tree T^*.

Our approach draws inspiration from the *adaptive submodular ranking* (ASR) problem (Navidi et al. 2020), which can be defined similarly, by replacing each object x_i in X with a non-decreasing submodular function $f_i : 2^D \to [0, 1]$ such that $f_i(\emptyset) = 0$ and $f_i(D) = 1$; recall that D is the set of tests, and thus, each function f_i takes as input a subset of tests. We denote the set of non-decreasing submodular functions by $F = \{f_i \mid x_i \in X\}$. We again consider a tree, which recursively partitions F. The tests D and the probability distribution p apply to the set of functions F in the same way that they apply to their corresponding objects. For example, a function f_i evaluated on a test $d \in D$ returns a value $d(f_i) = d(x_i)$, which determines the branch of the tree that f_i will follow. Given a tree T, a function f picks up all tests associated with the nodes it goes through and is *fully covered* when it reaches its maximum function value $f(D)$. Let $c(T, f)$ be the cost of *covering* f in T, defined as the sum of costs of all tests that f goes through in T before it is *fully covered*. Note that a function is not necessarily covered in a leaf node, it may be covered in an internal node. The expected cost of a tree T is defined in a similar manner as for the NGI problem. The *adaptive submodular ranking* problem is defined as follows.

Problem 2 (*Adaptive submodular ranking (ASR)* (Navidi et al. 2020)) Given a problem instance $I = (F, D, p, c)$, with set of submodular functions F, set of tests D, a probability distribution p, and cost function c, find a tree $T(F)$ that covers all functions in F and minimizes the expected cost $c(T(F)) = \sum_{f \in F} p(f) c(T, f)$.

Decomposable impurity functions When constructing decision trees for classification tasks, in addition to having small expected cost, the discriminative power of the selected tests is also vital. A number of different impurity measures have been widely used in deciding a discriminative test in decision trees, such as *entropy* and *Gini index*. Such impurity measures are defined as functions $h : [0, 1]^k \to \mathbb{R}_+$, taking as input the class distribution at a given tree node. Impurity functions are expected to satisfy certain conditions (Kearns and Mansour 1999), which capture the notion of "impurity."

All impurity functions mentioned in this paper satisfy the following conditions: (1) they obtain the maximum value if the class distribution is uniform, and the minimum value zero if a node is pure (i.e., homogeneous); (2) they are concave; and (3) they are symmetric.

A typical splitting criterion compares the change in impurity before and after performing a test d, defined as $h(S)$ and $h(S \mid d)$, respectively. The *impurity reduction* of a test d on a tree node S is defined as $d(S, d) = h(S) - h(S \mid d)$. A test that causes larger impurity reduction is considered more discriminative. Based on the concavity property of h, it is easy to show that $d(S, d) \geq 0$ for any tree node S and test d. We defer the proof of this claim to the Appendix, Section B.

Before we define a special family of impurity functions for our problem, we first introduce some additional notation. For a node S of the tree, where $S \subseteq X$, we define S_d^v as the child node of S by equipping S with test d and following the branch that takes on a specific testing value v. In particular, we define $S_d^{(i)} = S_d^{v=d(x_i)}$. Likewise, we define $S_{D'}^{(i)}$ as the ending node of a path that starts at $S \subseteq X$ and follows a sequence of nodes each equipped with a test d in $D' \subseteq D$ by taking on a value of $d(x_i)$. Note that the order of tests in D' does not matter in $S_{D'}^{(i)}$. Finally, we denote the total probability of objects in a specific class ℓ in S as $p_\ell(S) = \sum_{x \in S : \ell(x) = \ell} p(x)$.

We are now ready to define $h(S)$ and $h(S \mid d)$ for our problem. We require h to be *decomposable*, i.e., to be a weighted sum over impurity scores in each class. We define:

$$h(S) \;=\; \sum_{\ell} \frac{p_\ell(S)}{p(S)} h_\ell(S) \;=\; \frac{1}{p(S)} \sum_{x \in S} p(x) h_{\ell(x)}(S), \tag{1}$$

where $h_\ell(S)$ can be any function of $\frac{p_\ell(S)}{p(S)}$, the proportion of objects of class ℓ in S, which ensures that h satisfies the three requirements stated above (i.e., (1) being maximized at uniform class distribution and minimized at homogeneity, (2) concavity, and (3) symmetry).

A wide range of concave impurity functions adopt such a form. For example, h becomes the entropy function when $h_\ell(S) = -\log \frac{p_\ell(S)}{p(S)}$, and it becomes the Gini index when $h_\ell(S) = 1 - \frac{p_\ell(S)}{p(S)}$. With the impurity of a node S defined, $h(S \mid d)$ is just a weighted sum of the impurity of all child nodes of S when split by test d, i.e., $h(S \mid d) = \sum_{v \in [v_d]} \frac{p(S_d^v)}{p(S)} h(S_d^v)$.

A useful quantity for our analysis is the maximum value of $h_{\ell(x)}(S)$, which we denote by $\epsilon_h = \max_{S \subseteq X} \max_{x \in S} h_{\ell(x)}(S)$.

4 Algorithm

The main idea of our approach is to cast the NGI problem as an instance of the ASR problem (Navidi et al. 2020). We achieve this by defining a non-decreasing submodular function for each object. The ASR problem is solved by a greedy algorithm that picks tests to maximize the coverage of the submodular functions while encouraging a

Algorithm 1 Greedy tree-induction algorithm

Input: An instance $I = (S, L, D, \ell, p, c, \theta)$, a set of tests $D' \subseteq D$ used so far, impurity function h, trade-off parameter $\lambda \geq 0$
Output: A decision tree T

Return a decision tree TREE$(I, \emptyset)$ ▷ A recursive call at the top level with $D' = \emptyset$
function TREE(I, D')
  if S is homogeneous or $p(S) \leq \theta$ **then**
    Return a leaf node S, labeled with its majority class
  end if
  for $d \in D \setminus D'$ **do**
    $v^* \leftarrow \arg\max_{v \in [\nu_d]} |S_d^v|$ ▷ the largest child node by test value v^*
    Let $Z'(d) = \frac{1}{c(d)}\big(p(S) - p(S_d^{v^*}) + \sum_{i:x_i \in S} p(x_i) \frac{\tilde{f}_i(D' \cup \{d\}) - \tilde{f}_i(D')}{\tilde{f}_i(D) - \tilde{f}_i(D')}\big)$
    Let $Z(d) = Z'(d) + \frac{1}{c(d)} \lambda\, p(S)(h(S) - h(S \mid d))$
  end for
  $d^* \leftarrow \arg\max_{d \in D \setminus D'} Z(d)$
  for $v \in [\nu_{d^*}]$ **do**
    $T_v \leftarrow$ TREE$((S_{d^*}^v, L, D, \ell, p, c, \theta), D' \cup \{d^*\})$
  end for
  Return a tree rooted at S with children $\{T_v\}$ by test d^*
end function

balanced partition. We further incorporate the impurity-reduction objective into the greedy criterion to encourage the selection of discriminative tests, without losing the approximation guarantee.

Our algorithm for the NGI problem is demonstrated in Algorithm 1. It is a greedy algorithm, which, at each node S, selects a test d that maximizes a cost-benefit greedy score $Z(d)$ consisting of the following three terms:

$$Z(d) = \frac{1}{c(d)} \left(\underbrace{B(d)}_{\text{balance}} + \underbrace{E(d)}_{\text{efficiency}} + \lambda \underbrace{D(d)}_{\text{discrimination}} \right). \tag{2}$$

The first term, $B(d) = p(S) - p(S_d^{v^*})$, with $v^* = \arg\max_{v \in [\nu_d]} |S_d^v|$, is the sum of the branch probabilities except the largest-cardinality branch. Maximizing $B(d)$ encourages selecting a test d that yields a balanced split.

The second term,

$$E(d) = \sum_{i:x_i \in S} p(x_i) \frac{\tilde{f}_i(D' \cup \{d\}) - \tilde{f}_i(D')}{\tilde{f}_i(D) - \tilde{f}_i(D')},$$

is the re-weighted total sum of the marginal gain in each submodular function, which we will define for our objects shortly. Maximizing $E(d)$ accelerates the progress towards termination.

The last term, $D(d) = p(S)\,(h(S) - h(S \mid d))$, is the impurity reduction we defined in Sect. 3, which improves the discrimination of the selected test. The user-defined parameter $\lambda \geq 0$ controls the trade-off between tree complexity and discrimination.

One way to understand the greedy score $Z(d)$ is to view the B and E terms as a *regularizer*. Notice that maximizing only the first two terms, $Z'(d) = \frac{1}{c(d)}(B(d) + E(d))$ at Step 8 of Algorithm 1, is exactly the greedy criterion used by Navidi et al. (2020) to solve the ASR problem.

We finish the description of our method by showing how to define the submodular function f_i for each object x_i. We start by defining two monotonically non-decreasing submodular functions. For each object $x_i \in X$, both submodular functions take as input a subset of tests $D' \subseteq D$ and return a real value. The first function $f_i^p(D')$ is defined as the scaled total probability of the objects that do not fall into $S_{D'}^{(i)}$, i.e., the objects that disagree with x_i in at least one test in $D' \subseteq D$. Note that eventually, only object x_i itself stays in $S_D^{(i)}$. Formally, we define f_i^p as

$$f_i^p(D') = (1 - p(S_{D'}^{(i)}))/(1 - p(x_i)).$$

The second function $f_i^{\boldsymbol{\phi}}(D')$ is defined as the number of heterogeneous pairs that do not fall into $S_{D'}^{(i)}$. Eventually, no heterogeneous pair will exist in $S_D^{(i)}$ and the ending node is homogeneous. We define

$$f_i^{\boldsymbol{\phi}}(D') = (\boldsymbol{\phi}(X) - \boldsymbol{\phi}(S_{D'}^{(i)}))/\boldsymbol{\phi}(X).$$

The target (maximum) values for these two functions are both 1, for each object x_i. Thus, the functions f_i^p and $f_i^{\boldsymbol{\phi}}$ are *fully covered* for a subset of tests $D' \subseteq D$ for which $f_i^p(D') = 1$ and $f_i^{\boldsymbol{\phi}}(D') = 1$, respectively. It is easy to see that both functions are submodular and monotonically non-decreasing. When the termination constraint θ for minimum probability is in place, we use the fact that the monotonicity and submodularity properties remain valid when truncated by a constant. The truncated version of the f_i^p function is defined as

$$\bar{f_i^p}(D') = \min\left\{(1 - p(S_{D'}^{(i)}))/(1 - \max\{p(x_i), \theta\}), 1\right\}.$$

Next we define the *disjunction* function $\tilde{f}_i$ of $\bar{f_i^p}$ and $f_i^{\boldsymbol{\phi}}$, which remains monotonically non-decreasing and submodular (Deshpande et al. 2016; Guillory and Bilmes 2011). We set

$$\tilde{f}_i(D') = 1 - \left(1 - \bar{f_i^p}(D')\right)\left(1 - f_i^{\boldsymbol{\phi}}(D')\right).$$

It is easy to see that with a reasonable value of θ (e.g., a multiple of the greatest common divisor of $\{p(x)\}$), the minimum positive incremental value of any element and any $\tilde{f}_i$ is

$$\Delta = \min_{\substack{i \in [n],\, D' \subseteq D, \\ d \in D : \tilde{f}_i(D' \cup \{d\}) > \tilde{f}_i(D')}} \left\{\tilde{f}_i(D' \cup \{d\}) - \tilde{f}_i(D')\right\} = \Omega(p_{\min}/n^2).$$

Last, we examine the time complexity of Algorithm 1. The greedy score for a tree node S with respect to a test can be computed in $\mathcal{O}(|S|)$ time. At each level of the decision tree, the union of disjoint nodes has a total size n. Thus, the worst-case time complexity is $\mathcal{O}(Hmn)$, where m is the number of tests, n is the number of objects, and H is the tree height, which is upper bounded by n. In practice, the algorithm is more efficient than what this worst-case bound suggests; it has the same time complexity as standard tree-induction algorithms, such as CART.

5 Approximation guarantee

In this section we establish the approximation guarantee of Algorithm 1 for the NGI problem. Our main result is the following.

Theorem 1 *Algorithm 1 provides an* $\mathcal{O}(\log 1/p_{min} + \log n + \lambda\epsilon_h)$ *approximation guarantee for the NGI problem.*

As a practical consequence of Theorem 1, we have the following corollary, which is a consequence of the fact that for the popular impurity functions the factor ϵ_h in the approximation ratio of Theorem 1 can be effectively bounded. Note that in practice $p_{\min} \geq 1/n$ when given training data of n points, and thus $\mathcal{O}(\log 1/p_{\min}) = \mathcal{O}(\log n)$. We omit the constant λ for simplicity.

Corollary 2 *Algorithm 1 provides an* $\mathcal{O}(\log 1/p_{min} + \log n)$ *approximation guarantee for the NGI problem when the impurity function h is either the entropy or the Gini index function.*

Proof In addition to the result of Theorem 1 we can show that ϵ_h is small, compared to the other terms, or bounded by a constant. When h is the entropy function, we have

$$\begin{aligned}\epsilon_h &= \max_{S\subseteq X}\max_{x\in S}\left\{h_{\ell(x)}(S)\right\} = \max_{S\subseteq X}\max_{x\in S}\left\{-\log\frac{p_{\ell(x)}(S)}{p(S)}\right\}\\ &\leq -\log\frac{\min_{x\in X} p(x)}{p(X)} = \log 1/p_{\min}.\end{aligned}$$

When h is the Gini index function, we have

$$\epsilon_h = \max_{S\subseteq X}\max_{x\in S}\left\{h_{\ell(x)}(S)\right\} = \max_{S\subseteq X}\max_{x\in S}\left\{1-\frac{p_{\ell(x)}(S)}{p(S)}\right\} \leq 1.$$

□

Assuming $\mathbf{P} \neq \mathbf{NP}$, the result given by Theorem 1 is asymptotically the best possible among instances where $1/p_{\min}$ is polynomial in n. This follows directly from the hardness result of the EI problem (Chakaravarthy et al. 2007), which in turn, is proved via a reduction from the minimum set-cover problem. Recall that by specifying θ to be zero, NGI problem degenerates into EI or GI problems. The constructed EI instance in their reduction asks for a minimum object probability $p_{\min}$ such that

$1/p_{\min} = \Theta(n^3)$, and thus if NGI admits $o(\log 1/p_{\min}) = o(\log n)$ approximation, we could solve the set-cover problem with $o(\log n)$-approximation, which is conditionally impossible (Feige 1998).

Remark 1 The NGI problem does not admit an $o(\log n)$ approximation algorithm, unless $\mathbf{P} = \mathbf{NP}$.

5.1 Proof of Theorem 1

Our analysis is similar to the one by Navidi et al. (2020), except that we need a new proof of their key lemma for our new greedy selection rule (Eq. (2)). This is done by leveraging the special structure in the family of impurity functions (Eq. (1)) we employ.

In order to analyze the total cost along a path, we treat cost as discrete "time" — or continuous time if we allow continuous cost — and we divide time geometrically. We refer to the decision tree returned by Algorithm 1 as T_{A}, while we refer to the optimal decision tree as T^*. We denote the set of internal (i.e., unfinished) nodes up to time t in T_{A} as $\mathcal{C}(t)$, and similarly as $\mathcal{C}^*(t)$ in T^*.

We define $\mathcal{C}_k = \mathcal{C}(\gamma 2^k)$ and $\mathcal{C}^*_k = \mathcal{C}^*(2^{k-1})$, for a constant γ to be defined shortly. That is, we are interested in the set of unfinished nodes at the end of the k-th geometrically increasing time interval. Notice that the interval length for $\mathcal{C}$ is stretched by a factor of 2γ, compared to $\mathcal{C}^*$. We define $p(\mathcal{C}(t)) = \sum_{S\in\mathcal{C}(t)} p(S)$, i.e., the total probability of unfinished nodes at time t. Note that $p(\mathcal{C}(t))$ is non-increasing as t grows, and in the case of integral costs we have $p(\mathcal{C}^*_0) = p(\mathcal{C}^*(2^{-1})) = 1$, i.e., no test can be completed within a fractional cost.

The cost of some test may be truncated by the defined geometrical time intervals. To denote the actual cost of a test within an interval, we first define a path π_{ik} in T_{A} to be the sequence of tests involved within time $(\gamma 2^k, \infty)$ for each object x_i. A test d selected by object x_i appears in path π_{ik} during time interval $(\gamma 2^k, \infty) \cap (t_{i,<d}, t_{i,<d} + c(d)]$, where $t_{i,<d}$ is the total cost before test d for object x_i. The truncated cost of a test $d \in \pi_{ik}$ within that intersection is denoted by $c_{ik}(d)$. Note that $c_{ik}(d) \le c(d)$. We denote the set of tests before test d in path π_{ik} by $\pi_{ik,<d}$.

Our greedy algorithm is identical to the algorithm of Navidi et al. (2020) except that their greedy-selection score Z' at Step 8 is replaced by the new score Z at Step 9, in order to encourage impurity reduction of the selected tests. The key in the analysis of Navidi et al. is to show that $p(\mathcal{C}_{k+1}) \le 0.2p(\mathcal{C}_k) + 3p(\mathcal{C}^*_{k+1})$, which is proven via an intermediate value Z'_k defined below. We restate their technical result here.

Lemma 3 (Navidi et al. 2020, Lemma 2.4,2.5) *If Algorithm 1 is executed using the greedy score Z' at Step 8, then*

$$\begin{aligned} Z'_k &\ge \left(p(\mathcal{C}_{k+1}) - 3p(\mathcal{C}^*_{k+1})\right)\gamma'/3, \text{ and} \\ Z^{\infty\prime}_k &\le p(\mathcal{C}_k)\gamma'/15, \end{aligned}$$

where $Z'_k = \sum_{\gamma' 2^k < t \le \gamma' 2^{k+1}} \sum_{S\in\mathcal{C}(t)} Z'(d(S))$, $Z^{\infty\prime}_k = \sum_{t>\gamma' 2^k} \sum_{S\in\mathcal{C}(t)} Z'(d(S))$, $\gamma' = 15(1 + \ln 1/\Delta + \log n)$, *and $d(S)$ is the greedy test for node S. Besides, the first*

inequality holds regardless of the value of γ' and holds as long as $d(S)$ is a greedy test with respect to an additive score $Z' + D$, where D can be any non-negative function; the second inequality holds regardless of the choice of tests $d(S)$ in the decision tree.

Our new greedy score Z is in an additive form required above for the first inequality. Therefore, in our case, the difficulty mainly lies in the second inequality. We can prove that a similar lemma holds for our new greedy score.

Lemma 4 *Algorithm 1 ensures the following inequalities*

$$\begin{aligned} Z_k &\geq \left(p(\mathcal{C}_{k+1}) - 3p(\mathcal{C}^*_{k+1})\right)\gamma/3, \text{ and} \\ Z^\infty_k &\leq p(\mathcal{C}_k)\gamma/15, \end{aligned}$$

where $Z_k = \sum_{\gamma 2^k < t \leq \gamma 2^{k+1}} \sum_{S \in \mathcal{C}(t)} Z(d(S))$, $Z^\infty_k = \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} Z(d(S))$, $\gamma = 15(1 + \ln 1/\Delta + \log n + \lambda\epsilon_h)$, *and* $d(S)$ *is the greedy test for node* S.

Proof The first inequality is easy to show. Notice that compared to Z', Z introduces a third term of impurity reduction D in Eq. (2), which is always non-negative (see Sect. 3) and thus $Z(d) \geq Z'(d)$. Thus, the first inequality in Lemma 3 also holds for the tree generated by the new greedy score. Since the first inequality in Lemma 3 does not depend on the value of γ', we replace it with the new γ, which completes the proof.

The second inequality requires more work. We denote the sum of the D terms in Z^∞_k by

$$G = \lambda \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} \frac{p(S)}{c(d(S))} \left(h(S) - h(S \mid d(S))\right).$$

From Lemma 3 we know that $Z^\infty_k - G \leq \gamma' p(\mathcal{C}_k)/15$.

We now upper bound the additional term G. We omit S in $d(S)$ when it is clear from the context.

$$\begin{aligned} G &= \lambda \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} \frac{p(S)}{c(d)} \left(h(S) - h(S \mid d)\right) \\ &= \lambda \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} \frac{p(S)}{c(d)} \left(h(S) - \sum_{v \in [v_d]} \frac{p(S^v_d)}{p(S)} h(S^v_d) \right) \\ &= \lambda \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} \frac{p(S)}{c(d)} \left(\frac{1}{p(S)} \sum_{x \in S} p(x) h_{\ell(x)}(S) \right. \\ &\quad \left. - \sum_{v \in [v_d]} \frac{p(S^v_d)}{p(S)} \frac{1}{p(S^v_d)} \sum_{x \in S^v_d} p(x) h_{\ell(x)}(S^v_d) \right) \\ &= \lambda \sum_{t > \gamma 2^k} \sum_{S \in \mathcal{C}(t)} \frac{1}{c(d)} \left(\sum_{x \in S} p(x) h_{\ell(x)}(S) - \sum_{v \in [v_d]} \sum_{x \in S^v_d} p(x) h_{\ell(x)}(S^v_d) \right) \end{aligned}$$

$$
\begin{aligned}
&= \lambda \sum_{t>\gamma 2^k} \sum_{S\in\mathcal{C}(t)} \sum_{x\in S} \frac{p(x)}{c(d)} \left(h_{\ell(x)}(S) - h_{\ell(x)}\left(S_d^{d(x)}\right)\right) \\
&= \lambda \sum_{S\in\mathcal{C}_k} \sum_{i:x_i\in S} p(x_i) \sum_{d\in\pi_{ik}} \frac{c_{ik}(d)}{c(d)} \left(h_{\ell(x_i)}\left(S^{(i)}_{\pi_{ik,<d}}\right) - h_{\ell(x_i)}\left(S^{(i)}_{\pi_{ik,<d}\cup\{d\}}\right)\right) \qquad (3) \\
&\le \lambda \sum_{S\in\mathcal{C}_k} \sum_{i:x_i\in S} p(x_i) \sum_{d\in\pi_{ik}} \left(h_{\ell(x_i)}\left(S^{(i)}_{\pi_{ik,<d}}\right) - h_{\ell(x_i)}\left(S^{(i)}_{\pi_{ik,<d}\cup\{d\}}\right)\right) \qquad (4) \\
&\le \lambda \sum_{S\in\mathcal{C}_k} \sum_{i:x_i\in S} p(x_i)\, h_{\ell(x_i)}(S) \qquad (5) \\
&\le \lambda \sum_{S\in\mathcal{C}_k} \sum_{i:x_i\in S} p(x_i)\, \epsilon_h \\
&= \lambda\, p(\mathcal{C}_k)\, \epsilon_h,
\end{aligned}
$$

where step (3) follows by enumerating the summands in a different order, step (4) is due to $c_{ik}(d) \le c(d)$, and step (5) follows by considering the telescoping series of the impurity reduction along a path of an object. Putting everything together gives

$$Z_k^\infty = G + (Z_k^\infty - G) \le \lambda p(\mathcal{C}_k)\epsilon_h + p(\mathcal{C}_k)\gamma'/15 = p(\mathcal{C}_k)\gamma/15.$$

□

Next, we use another simple lemma that provides an upper bound on the expected cost C_{A} of T_{A}, and a lower bound on the optimal cost C^* of T^*. This result is a consequence of the geometrical division of time. For example, to obtain an upper bound for C_{A}, we assume that the set of unfinished nodes stays the same as $\mathcal{C}(\gamma 2^k)$ during the time interval $(\gamma 2^k, \gamma 2^{k+1}]$. Recall that $p(\mathcal{C}(t))$ is a non-increasing function of time t.

Lemma 5 (Navidi et al. 2020, Lemma 2.2) *The expected cost C_A of the tree T_A produced by Algorithm 1, and the cost C^* of the optimal tree T^* for the NGI problem, satisfy the following inequalities*

$$
\begin{aligned}
C_A &\le \gamma \sum_{k\ge 0} 2^k p(\mathcal{C}_k) + \gamma, \text{ and} \\
C^* &\ge \frac{1}{2} \sum_{k\ge 0} 2^{k-1} p(\mathcal{C}_k^*).
\end{aligned}
$$

We are now ready to prove our main result, Theorem 1, stated in Sect. 5. The proof relies on combining the results of Lemma 4 with the upper and lower bounds provided by Lemma 5.

Proof (Theorem 1) From Lemma 4, we obtain

$$\left(p(\mathcal{C}_{k+1}) - 3p(\mathcal{C}^*_{k+1})\right)\gamma/3 \le Z_k \le Z_k^\infty \le p(\mathcal{C}_k)\gamma/15.$$

By rearranging terms, we get

$$p(\mathcal{C}_{k+1}) \;\leq\; 0.2\, p(\mathcal{C}_k) + 3\, p(\mathcal{C}^*_{k+1}).$$

Define $Q = \gamma \sum_{k\geq 0} 2^k\, p(\mathcal{C}_k) + \gamma$, i.e., the upper bound of C_{A}. We have

$$\begin{aligned}
Q &= \gamma \sum_{k\geq 1} 2^k\, p(\mathcal{C}_k) + \gamma\, (p(\mathcal{C}_0) + 1) \\
&\leq \gamma \sum_{k\geq 1} 2^k \left(0.2\, p(\mathcal{C}_{k-1}) + 3\, p(\mathcal{C}^*_k)\right) + \gamma\, (p(\mathcal{C}_0) + 1) \\
&\leq \gamma \sum_{k\geq 0} 2^k\, 0.4\, p(\mathcal{C}_k) + \gamma \frac{1}{2} \sum_{k\geq 1} 2^{k-1}\, 12\, p(\mathcal{C}^*_k) + \gamma\, (p(\mathcal{C}_0) + 1) \\
&= \gamma \sum_{k\geq 0} 2^k\, 0.4\, p(\mathcal{C}_k) + \gamma \frac{1}{2} \sum_{k\geq 0} 2^{k-1}\, 12\, p(\mathcal{C}^*_k) - 3\,\gamma\, p(\mathcal{C}^*_0) + \gamma\, (p(\mathcal{C}_0) + 1) \\
&\leq \gamma \sum_{k\geq 0} 2^k\, 0.4 p(\mathcal{C}_k) + \gamma \frac{1}{2} \sum_{k\geq 0} 2^{k-1}\, 12\, p(\mathcal{C}^*_k) \\
&\leq 0.4\, Q + 12\, \gamma\, C^*,
\end{aligned}$$

where we note that $p(\mathcal{C}_0{}^*) = 1$ and $p(\mathcal{C}_0) \leq 1$. Together with Lemma 5, we obtain

$$C_{\mathrm{A}} \;\leq\; Q \;\leq\; \frac{12}{0.6}\, \gamma\, C^* \;=\; 20\, \gamma\, C^*.$$

□

6 Experimental evaluation

In this section, we evaluate the performance of our enhanced decision-tree algorithms by comparing them against strong baselines on a large collection of real-world datasets. Some additional experimental results are presented in the Appendix, including further experimental results for CART (Section C), further experimental results on tree size (Section D), additional statistical tests (Section E), and more visual examples (Section G). Our implementation and pre-processing scripts can be found in a Github repository.[2]

Datasets We evaluate our methods on 20 datasets from the UCI Machine Learning Repository (Dua and Graff 2017) and OpenML (Vanschoren et al. 2013). Information about the datasets is shown in Table 1. We experiment with datasets containing up to 0.6 million objects and 5 thousand features. We set the limit θ to be 0.005 for all datasets except for small ones, whose θ are set accordingly so that the minimum leaf size is 2. For all datasets, 70% of the data points are used for training, 10% for validation and the rest for testing. Numerical features are categorized into multiple bins

[2] https://github.com/Guangyi-Zhang/low-expected-cost-decision-trees.

Table 1 Datasets statistics; n, m, k: number of data points, binary features and classes

Dataset	n	m	k
iris	150	20	3
ilpd	583	46	2
breast-w	699	45	2
tic-tac-toe	958	27	2
obesity	2111	58	7
bioresponse	3751	5333	2
spambase	4601	285	2
phoneme	5404	25	2
musk	6598	830	2
speed-dating	8378	733	2
phishing-websites	11,055	46	2
shoppers	12,330	454	2
letter	20,000	80	26
default	30,000	112	2
bank-marketing	45,211	76	2
electricity	45,312	42	2
firewall	65,532	55	4
dota2	92,649	394	2
diabetic	101,766	264	3
covertype	581,012	94	7

Numerical features are categorized into 5 bins by the k-means strategy

by the k-means strategy, which can adapt to uneven data distributions. All categorical features are then binarized to avoid biases towards features with a large number of levels (Strobl et al. 2007). All identical objects are coalesced into a single object, and the sampling probability is set accordingly. To fulfill the realizability assumption, the majority class is assigned to each identical data point in the training set, which may have different classes otherwise, due to noise or feature discretization. Apart from the original datasets with unit test cost, we additionally create more challenging scenarios, where each test cost is independently drawn from the set $\{1, \ldots, 10\}$.

Algorithms and baselines A summary of the algorithms is displayed in Table 2. The algorithms that implement the proposed approach are denoted as *enhanced C*4.5 (EC4.5) and *enhanced CART* (ECART). Baselines include the following:

- The ASR method (Navidi et al. 2020), which is the greedy algorithm without the newly-introduced impurity-reduction term.
- Impure Pairs (IP), which maximizes the reduction in the number of impure pairs at each split, i.e., the unweighted edge cut among different classes (Golovin et al. 2010; Cicalese et al. 2014).
- BAL, which is an unsupervised balanced-tree algorithm that greedily selects the test that splits the current node most evenly.
- The two traditional algorithms C4.5 and CART, and their cost-benefit versions that select a test using a cost-weighted criterion (denoted with a prefix 'C').

Table 2 Summary of competing algorithms

Algorithm	Brief description
ASR	Greedy without impurity reduction (Navidi et al. 2020)
IP	Greedy in reducing the number of impure pairs (Golovin et al. 2010)
BAL	Greedy in the most balanced split
[p][C]CART	Traditional CART (Breiman et al. 1984)
[p][C]C4.5	Traditional C4.5 (Quinlan 1993)
[p]ECART	Enhanced CART (proposed method)
[p]EC4.5	Enhanced C4.5 (proposed method)

Prefix 'p-' indicates a variant with post-pruning

To ensure a meaningful comparison, we measure performance for all methods based on the same stopping criteria. All algorithms perform two-way splitting. Splitting of tree nodes stops if homogeneity is achieved or if the minimum-probability limit is reached. We examine the performance of C4.5 and CART with post-pruning (denoted with a prefix 'p') or without. We adopt the standard *minimal cost-complexity pruning* approach (Breiman et al. 1984), which prunes a tree node having many leaves if its impurity is no much larger than the total impurity of its leaves. The parameter that controls the stringency of the pruning is determined by cross-validation over a logspace from 10^{-5} to 1.

The only hyperparameter in our algorithm (λ) controls the trade-off between complexity and discrimination. The effect of λ is summarized in Fig. 2. For large values of λ our algorithms turn into the traditional tree-induction algorithms C4.5 and CART; on the other hand, if λ is zero, our algorithms turn into the greedy algorithm for the ASR problem. As we are working with a bi-criteria optimization problem, there is no golden rule in deciding the best value of the hyperparameter. In this experiment, we aim to decide a value of λ that preserves comparable accuracy while reduces the complexity as much as possible. Thus, we tune the hyperparameter λ by starting with a large λ and gradually decreasing it before a significant drop (larger than 1%) is seen in the predictive accuracy over the validation set. Note also that λ is invariant to the data size, as the greedy score only depends on the distributions before and after the split.

Results We evaluate all methods using ROC AUC as a measure of predictive power, expected cost as a measure of tree complexity, and tree size (i.e., the number of tree nodes) as an auxiliary measure of global tree complexity. A full result on tree size is deferred to Section D in Appendix. Reported results, shown in Fig. 5, are averages over 5 executions with random train-test splits. We conduct the Bonferroni-Dunn test with significance level $\alpha = 0.05$ for average ranks (Demšar 2006), and report the *critical difference diagram* in Figs. 3 and 4, where the proposed method pEC4.5 (or pECART) is tested against the other methods, and methods closer to the right end have a better rank. We see that the predictive power and tree complexity of pECART and pEC4.5 are statistically not significantly different from the respective best performer, while it is significantly better than most other baselines. Two methods C4.5 and CART

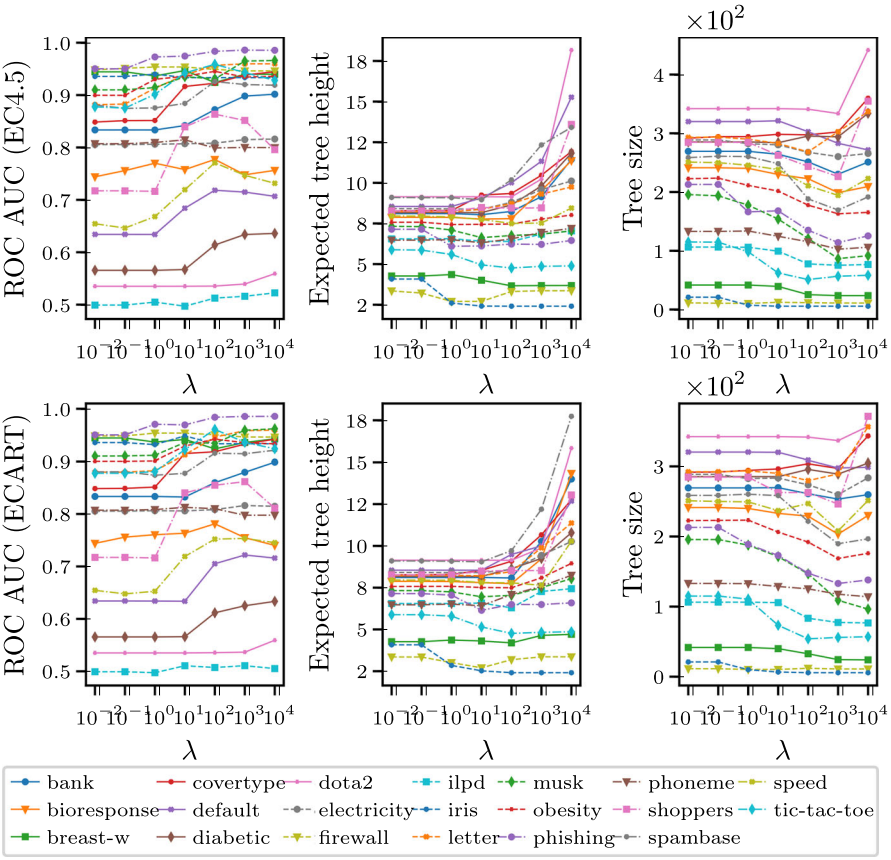


Fig. 2 The effect of the trade-off hyperparameter λ. ROC AUC is on validation data, and expected tree height or tree size on training data

lead to similar behavior; we focus on C4.5 below and discuss its results in details. Full results for CART and its enhancements are presented in the Appendix, Section C.

It can be seen that post-pruning has a noticeable positive effect on both the accuracy and complexity for the C4.5 algorithm. However, even after post-pruning, pC4.5 is still ranked closely to un-pruned EC4.5 in terms of the expected cost, and in some datasets, the expected cost of pC4.5 is about two times larger than that of EC4.5 in Fig. 5. This is reasonable because post-pruning mainly removes tree nodes near the bottom, but fails to rescue early bad splits near the root. On the other hand, post-pruning is significantly more beneficial than impurity reduction for the global tree size. Also note that post-pruning has less effect on EC4.5 in terms of accuracy, which indicates that un-pruned EC4.5 alone is robust to overfitting.

The decision tree produced by BAL is the worst in both aspects. This is expected for predictive power as BAL is an unsupervised method, but it is quite surprising for complexity. It turns out that BAL often has to keep expanding a balanced tree until the minimum leaf size is reached, as tree nodes rarely achieve homogeneity. This behavior

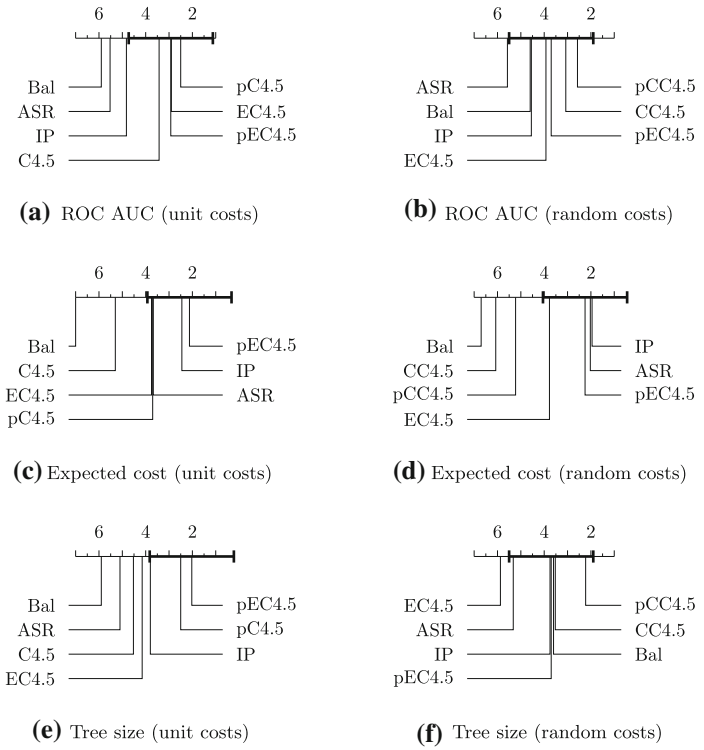


Fig. 3 Critical difference for the Bonferroni-Dunn test on significance level $\alpha = 0.05$ for average ranks of algorithms among 20 tested datasets. Methods closer to the right end have a better rank. The method that is compared with other methods is pEC4.5, and methods lying outside the thick interval are significantly different from pEC4.5

reinforces the argument that discriminative tests help accelerating termination and reducing expected cost.

The IP algorithm achieves better performance in both aspects than the ASR algorithm. However, IP has a too strong bias towards a balanced split, that it favors a random test over a discriminative one in the example we provide in Section A. This bias is also reflected in Fig. 5 where it falls behind ECART by more than 10% accuracy in some datasets. By further statistical tests we conduct in Section E, the predictive power of ASR and IP are statistically indistinguishable from the unsupervised BAL.

Finally, regarding running time, algorithm EC4.5 typically runs 3–4 times longer than C4.5, but there are instances that the latter algorithm constructs very skewed trees and it takes more time to complete (details in Appendix, Section F).

The benefit of the proposed method becomes more pronounced in non-uniform-cost scenarios, shown in Fig. 6. It turns out that the cost-benefit traditional trees fail to reduce the expected cost, which indicates the need for more sophisticated techniques like ours to tackle non-uniform costs. Our algorithms obtain comparable predictive power, while achieving up to 90% lower expected cost than the traditional trees.

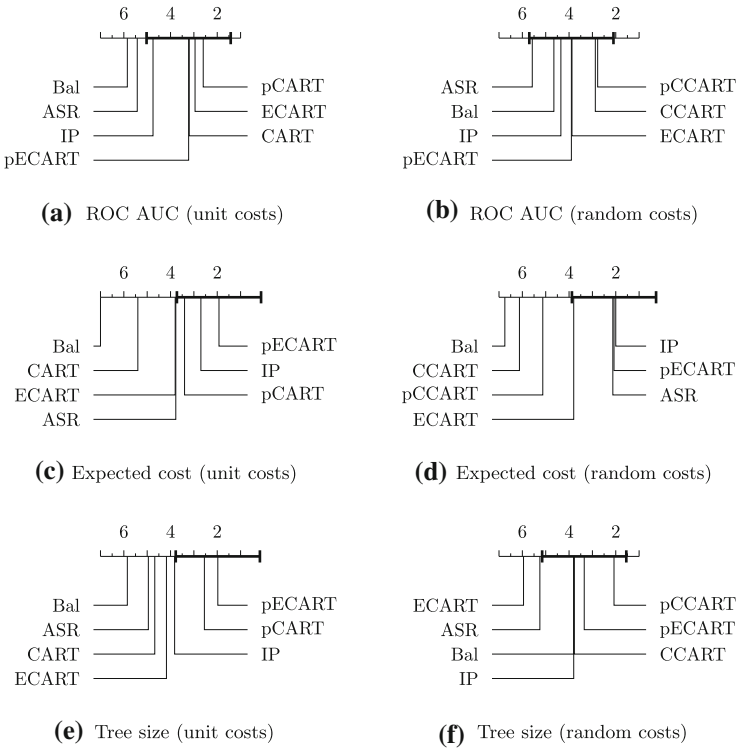


Fig. 4 Critical difference for the Bonferroni-Dunn test on significance level $\alpha = 0.05$ for average ranks of algorithms among 20 tested datasets. Methods closer to the right end have a better rank. The method that is compared with other methods is pECART, and methods lying outside the thick interval are significantly different from pECART

We conclude that our enhancement, given in the form of a regularizer, strikes an excellent balance between predictive power and expected tree height.

7 Conclusion

In this paper, we proposed a novel algorithm to construct a general decision tree with asymptotically tight approximation guarantee on expected cost under mild assumptions. The algorithm can be used to assimilate many existing standard impurity functions so as to enhance their corresponding splitting criteria with a complexity guarantee. Through empirical evaluation on various datasets and scenarios, we verified the effectiveness of our algorithm both in terms of accuracy and complexity. Potential future directions include the study of different complexity measures, further termination criteria, and incorporating a broader family of impurity functions.

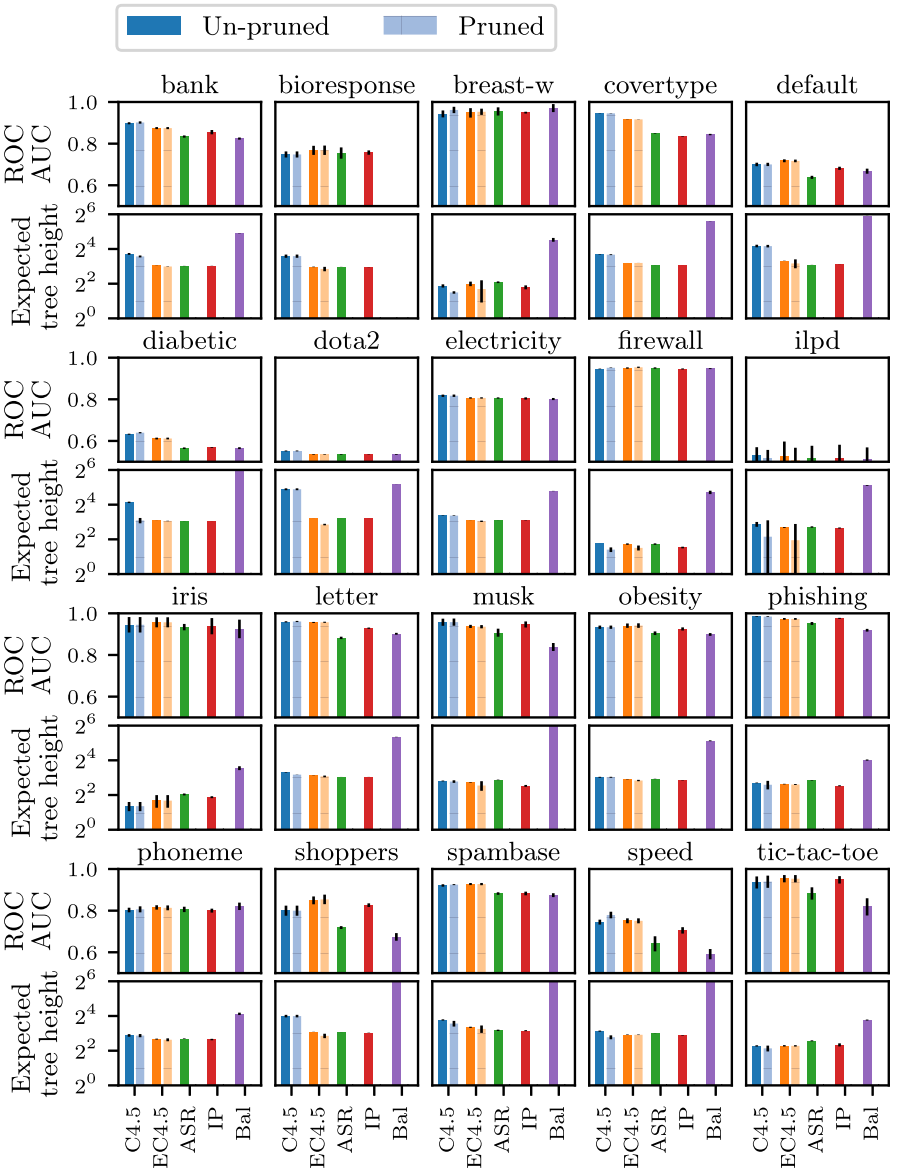


Fig. 5 Performance results with unit test costs. All plots in the same row share the same x- and y-axes. Error bars are also shown

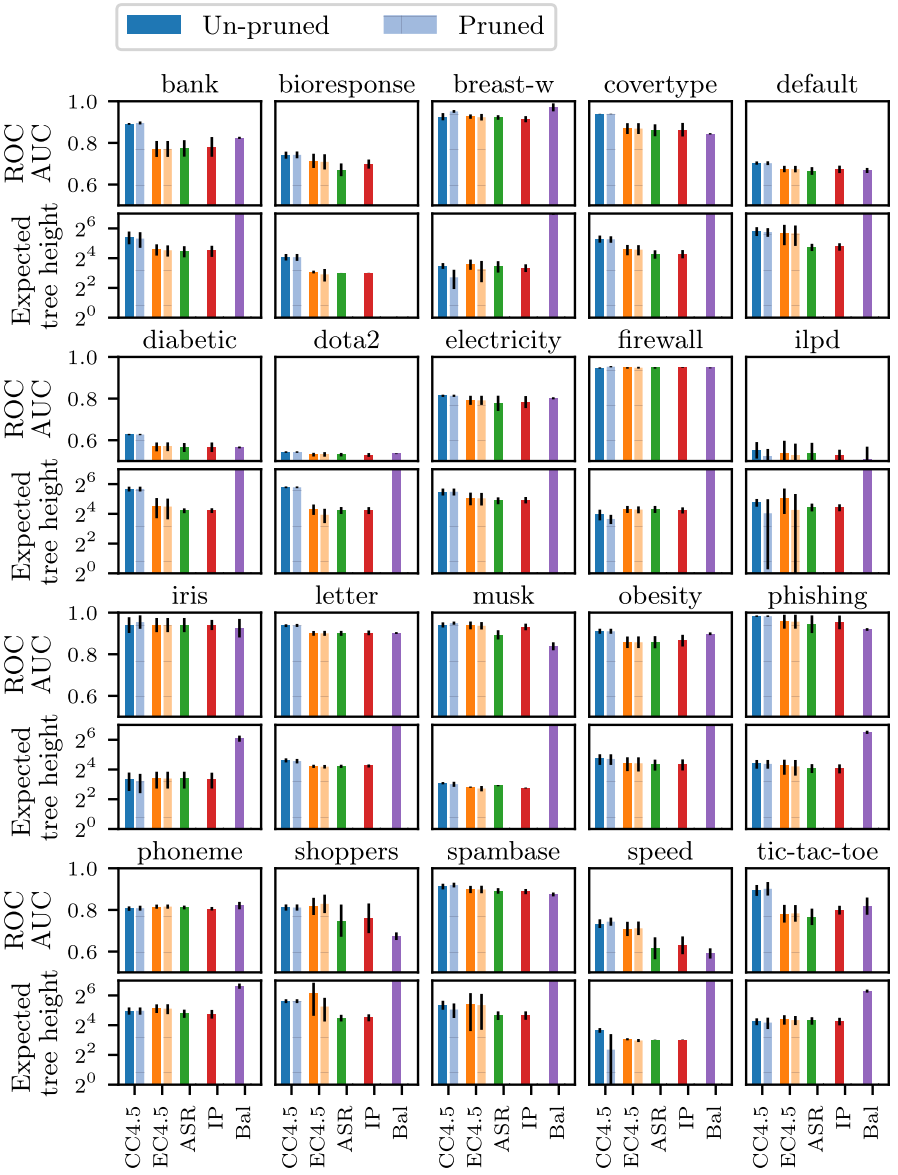


Fig. 6 Performance results with non-uniform test costs

Acknowledgements This research is supported by the Academy of Finland projects AIDA (317085) and MLDB (325117), the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Author Contributions GZ is responsible for the theoretical and experimental development. Both authors contribute significantly to the design and writing of the work.

Funding Open access funding provided by Royal Institute of Technology. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Availability of data and materials All datasets we use are publicly available in the UCI Machine Learning Repository (Dua and Graff 2017) and OpenML (Vanschoren et al. 2013).

Code Availability Our implementation is publicly available in the Github repository (https://github.com/Guangyi-Zhang/low-expected-cost-decision-trees).

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Appendix A Split criteria for adaptive submodular ranking (ASR) method may lead to non-discriminative decision trees

We present a simple example demonstrating that the ASR method, used for the group identification problem where the aim is to minimize the expected cost of a decision tree, may not select discriminative splits, which in turn may lead to decision trees with low predictive power. The splitting criterion of ASR consists of two terms (see Sect. 4 for more details). One term encourages balanced partitions and does not use label information. For the other term, two criteria have been considered in the literature: (1) maximize reduction in the number of *heterogeneous pairs* of objects after a split, or (2) maximize the number of excluded objects in other classes. These two criteria turn out to be equivalent up to a constant factor of 2 by a simple double counting. Apparently, when the number of heterogeneous pairs drops to zero or each object separates from all objects in other classes, we obtain perfect accuracy (in training data). Note that even random splits can lead to perfect accuracy as long as the tree is fully expanded. Here we discuss the second term, as random splits actually optimize balance, on expectation.

Our example is shown in Fig. 7, where we demonstrate two possible splits on a node. As we will see, ASR favors the non-discriminative split Fig. 7b over the more discriminative split Fig. 7a. We assume two classes, and we write (x, y) to indicate the

number of objects from the two different classes in a tree node. We now discuss the first criterion. In the left tree Fig. 7a, the root, the left, and the right node have 50×50, 0, and 26×50 heterogeneous pairs, respectively. In the right tree Fig. 7b, there are 50×50, 25×25, and 25×25 pairs, respectively. The left split decreases the number of heterogeneous pairs by $50 \times 50 - 0 - 26 \times 50 = 24 \times 50$. The right split decreases the number of heterogeneous pairs by $50 \times 50 - 25 \times 25 - 25 \times 25 = 50 \times 25$. Thus, the ASR criterion will select the (non-discriminative) split Fig. 7b.

Appendix B Impurity reduction is non-negative

By the concavity property of h, it is easy to show that the impurity-reduction function $d(S, d)$ is non-negative, for any tree node S and test d. In particular, we have

$$\begin{aligned} d(S,d) &= h(S) - h(S \mid d) \\ &= h(S) - \sum_{v \in [\nu_d]} \frac{p(S_d^v)}{p(S)} h(S_d^v) \\ &= h(p_S) - \sum_{v \in [\nu_d]} \frac{p(S_d^v)}{p(S)} h(p_{S_d^v}) \\ &\geq h(p_S) - h\left(\sum_{v \in [\nu_d]} \frac{p(S_d^v)}{p(S)} p_{S_d^v} \right) \\ &= h(p_S) - h(p_S) \\ &= 0, \end{aligned} \tag{B1}$$

where p_S denotes the class distribution vector of S, and Inequality (B1) is by concavity.

Appendix C Further experimental results for CART

See Figs. 8 and 9.

(a) A discriminative split **(b)** A non-discriminative split

Fig. 7 A simple example demonstrating that ASR split criteria may lead to non-discriminative decision trees

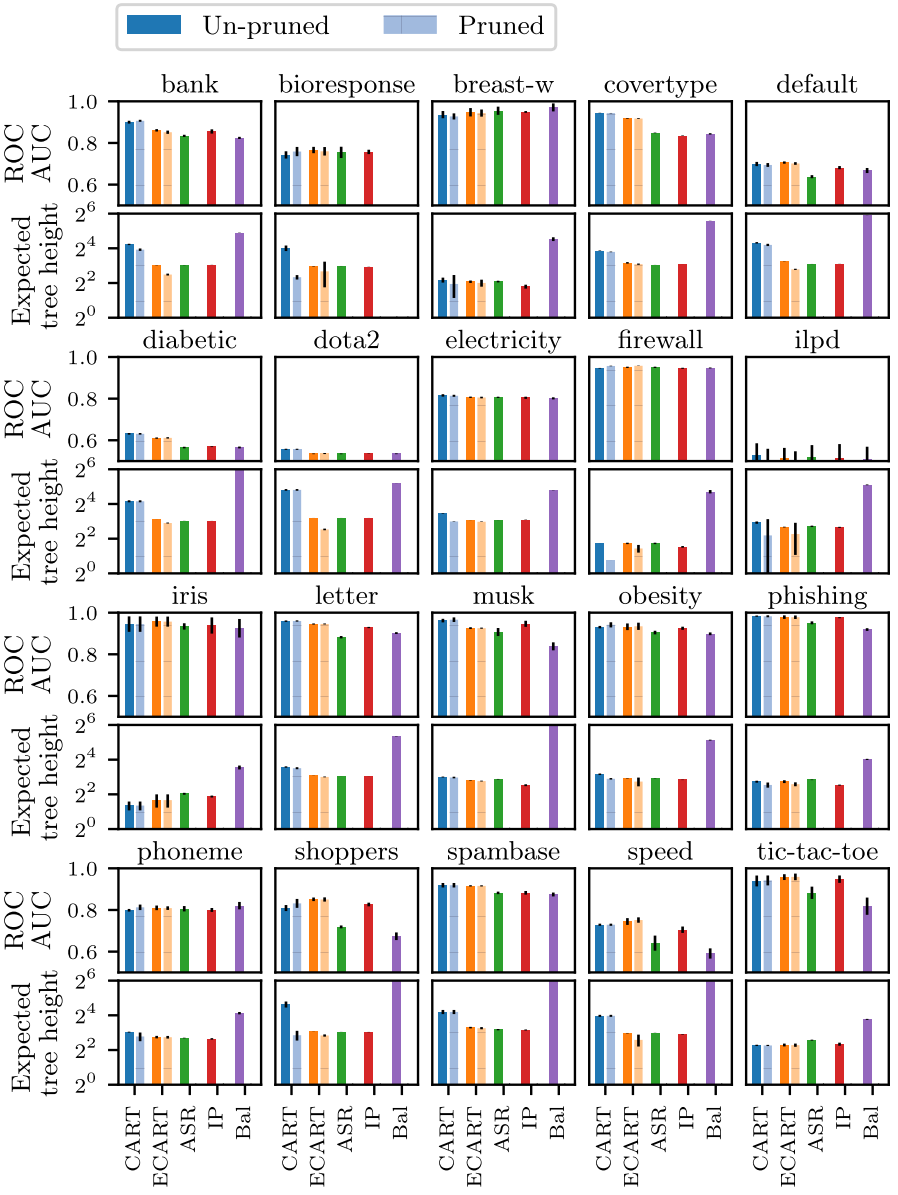


Fig. 8 Performance results with unit test costs. All plots in the same row share the same x- and y-axes. Error bars are also shown

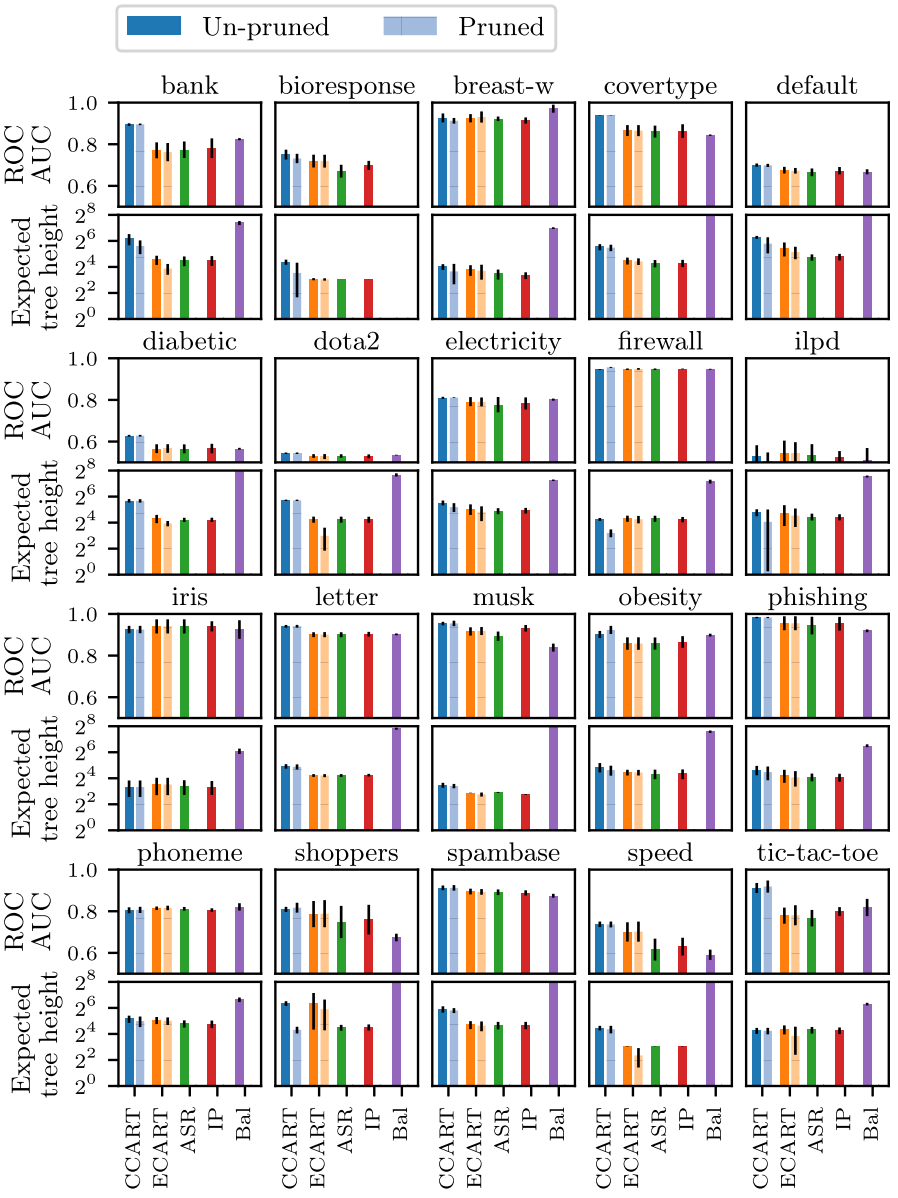


Fig. 9 Performance results with non-uniform test costs

Appendix D Further experimental results on tree size

See Figs. 10 and 11.

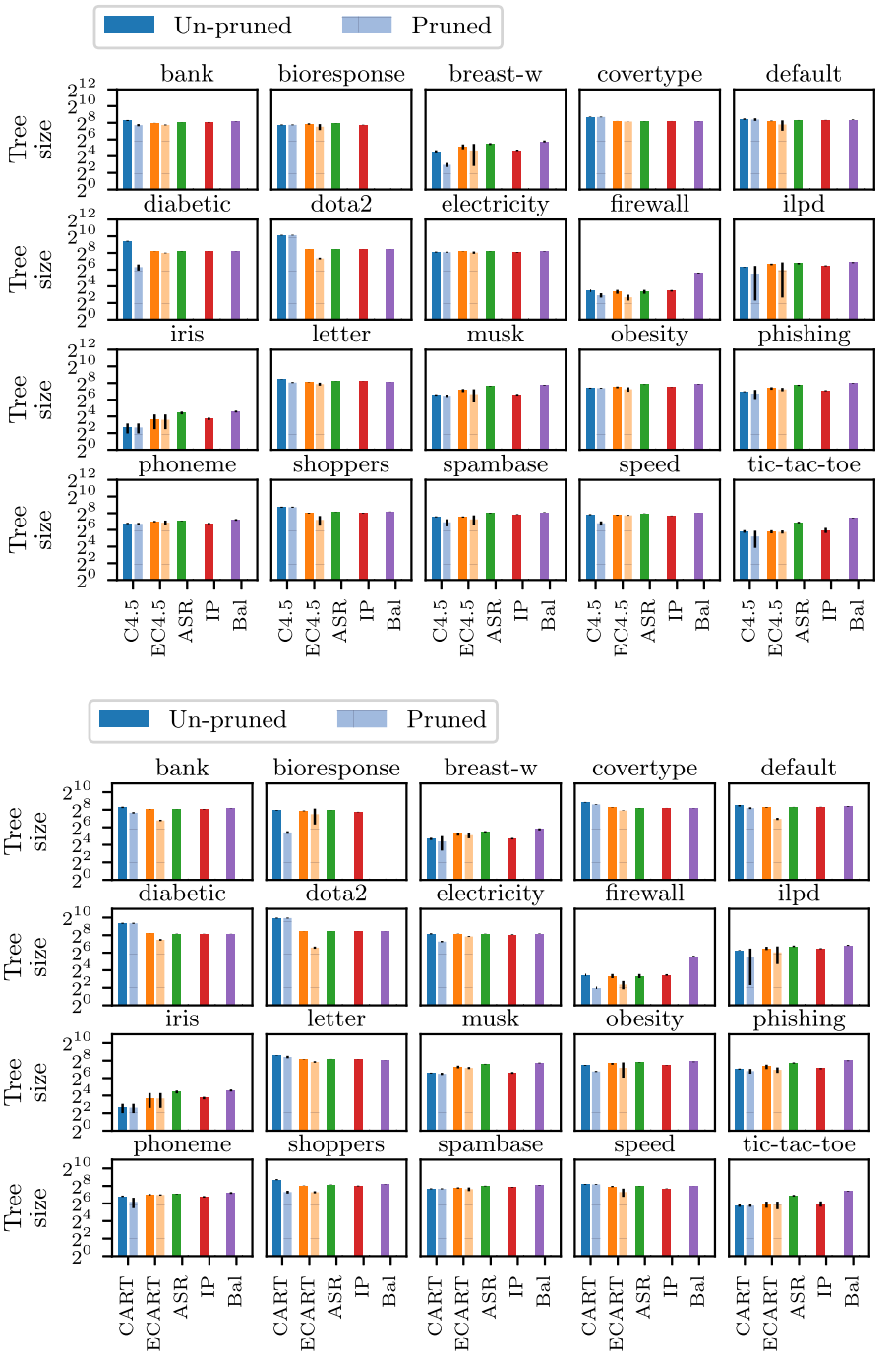


Fig. 10 Performance results with unit test costs. All plots in the same row share the same x- and y-axes. Error bars are also shown

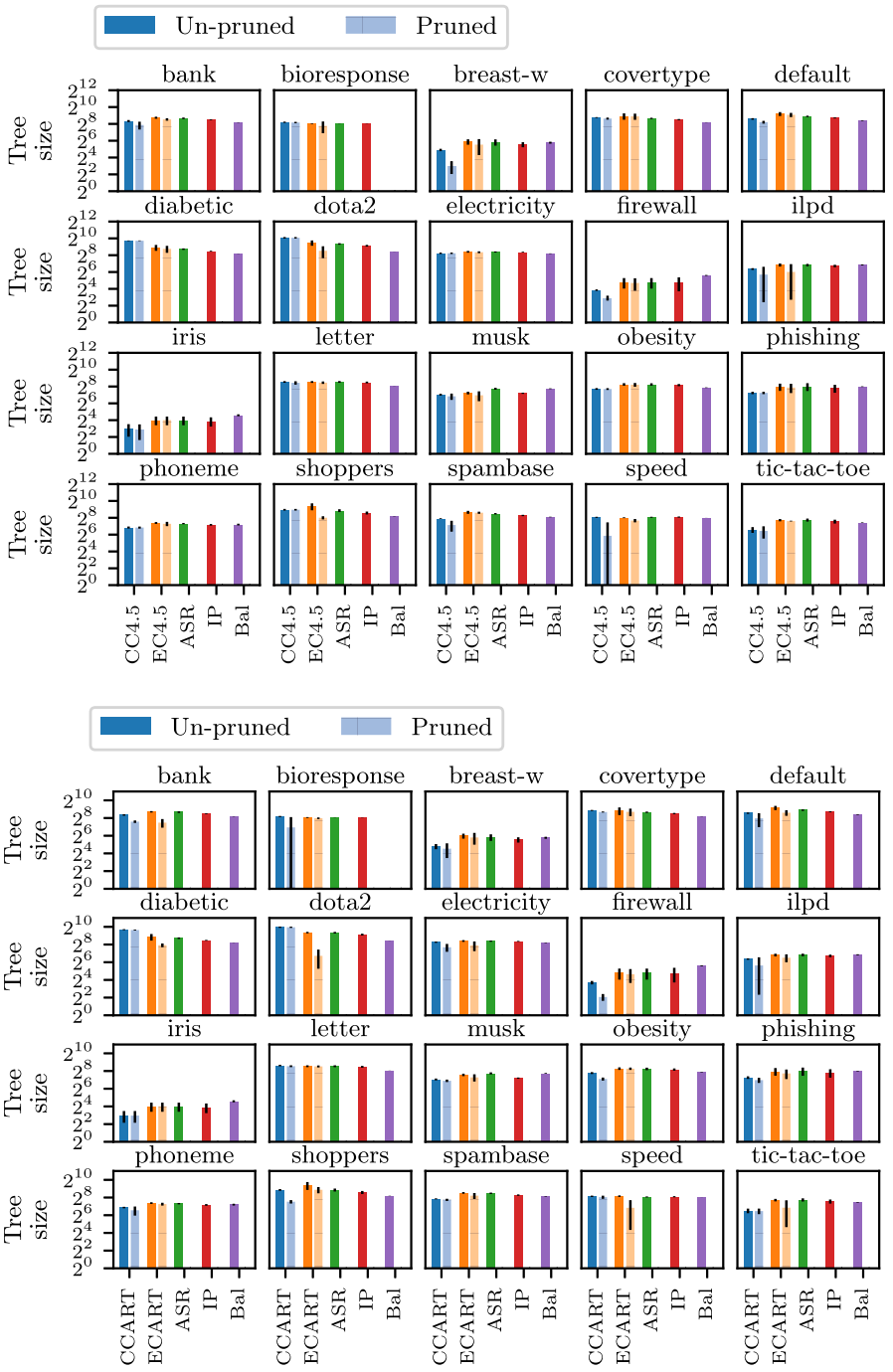


Fig. 11 Performance results with non-uniform test costs

Appendix E Further statistical tests: pairwise Nemenyi

See Figs. 12 and 13.

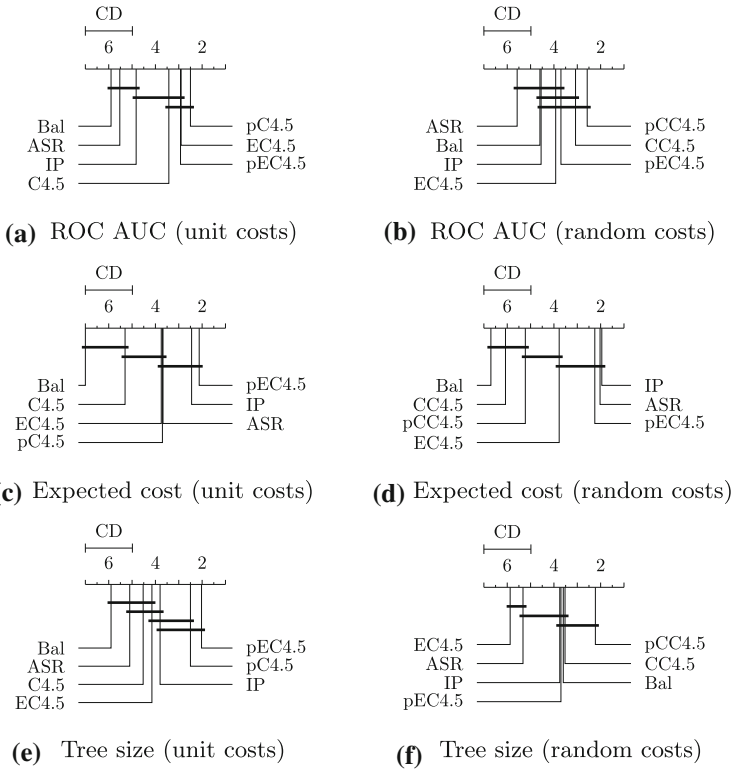


Fig. 12 Critical difference for the Nemenyi test on significance level $\alpha = 0.05$ for average ranks of algorithms among 20 tested datasets. Methods closer to the right end have a better rank. Any pair of methods which are not connected with an horizontal line have an average rank that is different with statistical significance

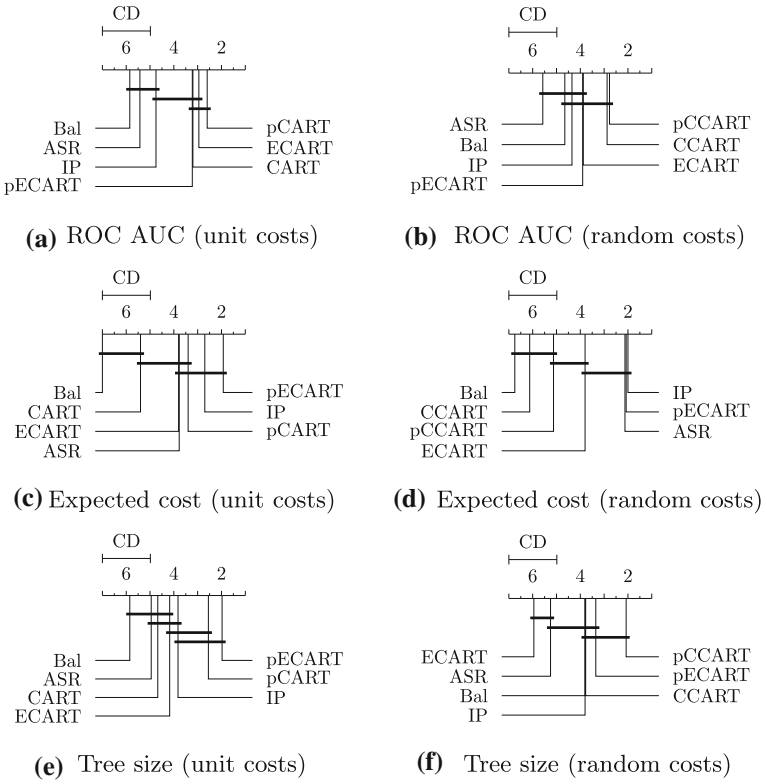


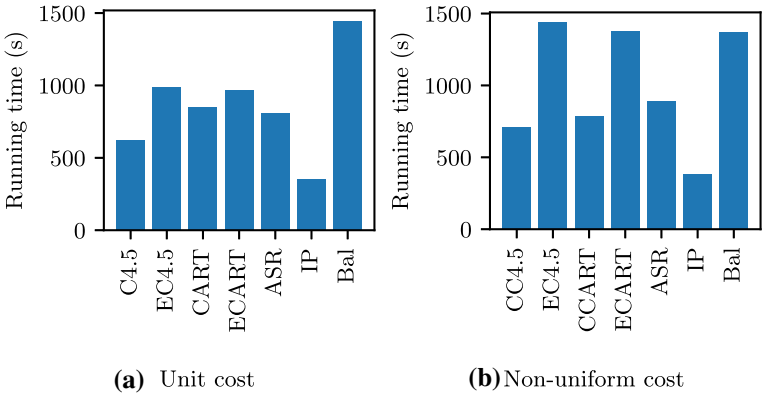
Fig. 13 Critical difference for the Nemenyi test on significance level $\alpha = 0.05$ for average ranks of algorithms among 20 tested datasets. Methods closer to the right end have a better rank. Any pair of methods which are not connected with an horizontal line have an average rank that is different with statistical significance

Appendix F Running time

Note that algorithm BAL misses results over some datasets because its running time is too long Fig. 14.

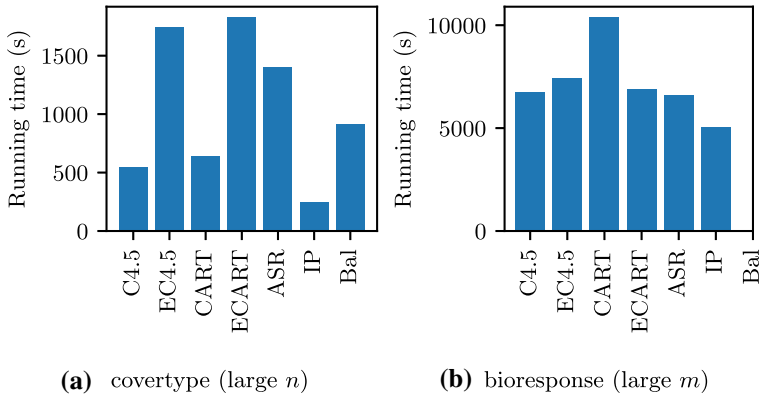
All experiments were carried out on a server equipped with 24 processors of AMD Opteron(tm) Processor 6172 (2.1 GHz), 62GB RAM, running Linux 2.6.32−754.35.1.el6.x86_64. We use Python 3.8.5.

We also demonstrate the running time for two selected datasets with unit costs, with a large number of data objects and features, to explore the impact of number of objects and features to the running time. In general, as reflected in the worst-case time complexity $\mathcal{O}(Hmn)$, the algorithms complete their computation quickly in the case of a large number of data objects (n), *or* large number of features (m), but not both. Furthermore, the dependency on n is slightly worse than on m, as the tree height H typically has a logarithmic dependence on n (Fig. 15).



(a) Unit cost **(b)** Non-uniform cost

Fig. 14 Running time, average over all datasets



(a) covertype (large n) **(b)** bioresponse (large m)

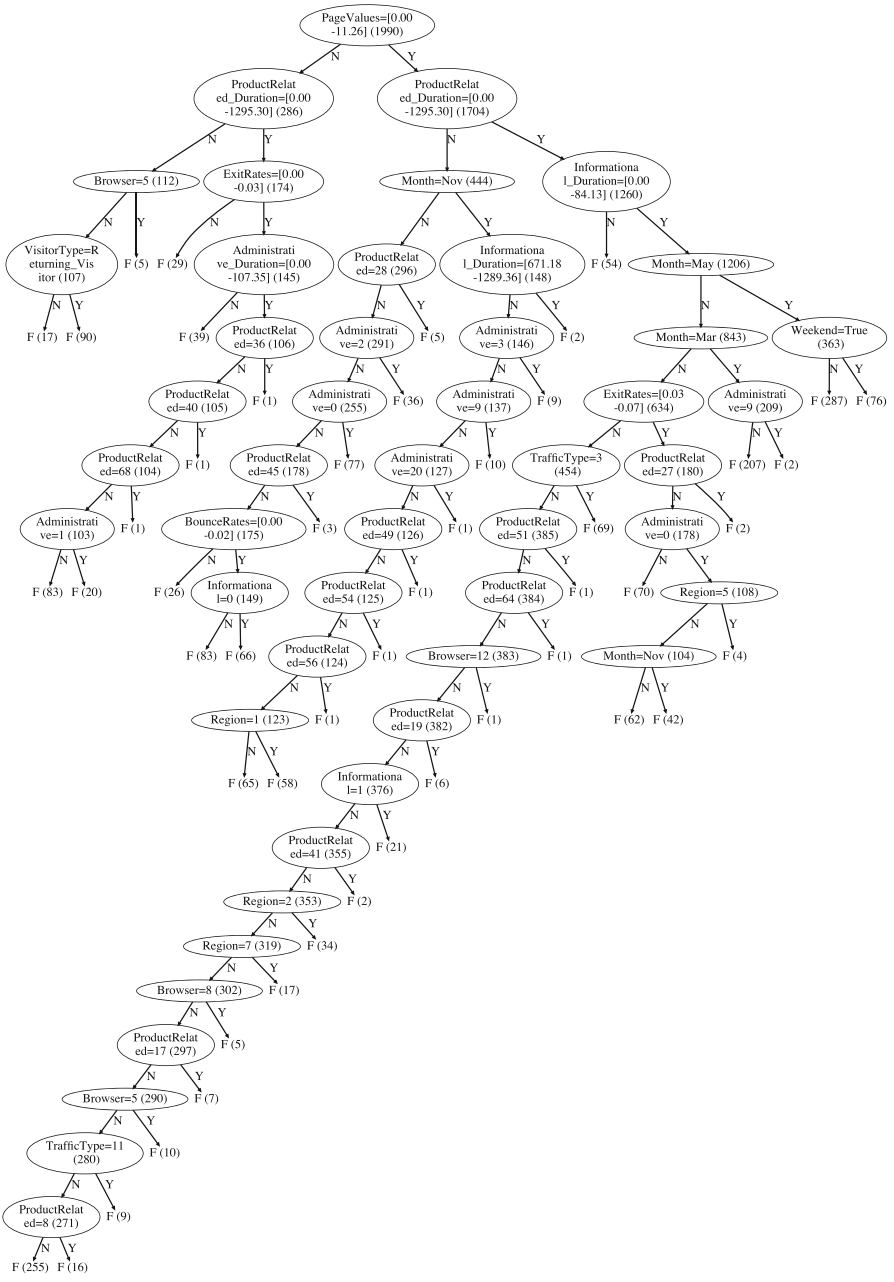
Fig. 15 Running time on selected datasets

Appendix G Visual examples of real-life datasets

We visualize datasets that have meaningful features and whose trees are small enough to be contained in the paper. We also adjust the minimum leaf size (1% of the data size) to produce smaller trees.

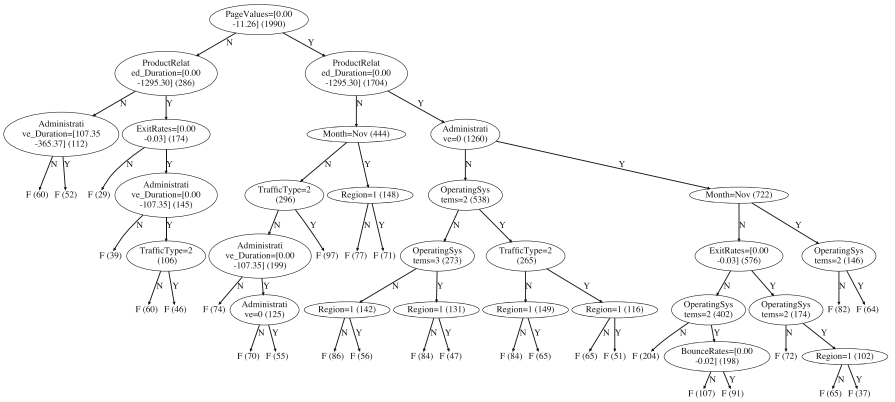
G.1 Visualization of decision trees for shoppers dataset

See Figs. 16, 17 and 18.



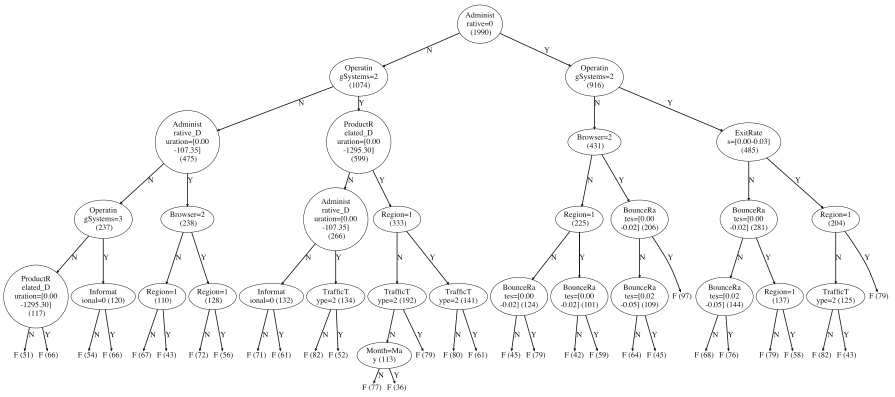
C4.5 (AUC ROC 0.841 and expected height 8.9)

Fig. 16 Visualization of EC4.5 decision tree for shoppers dataset



EC4.5 (AUC ROC 0.862 and expected height 5.49)

Fig. 17 Visualization of EC4.5 decision tree for shoppers dataset

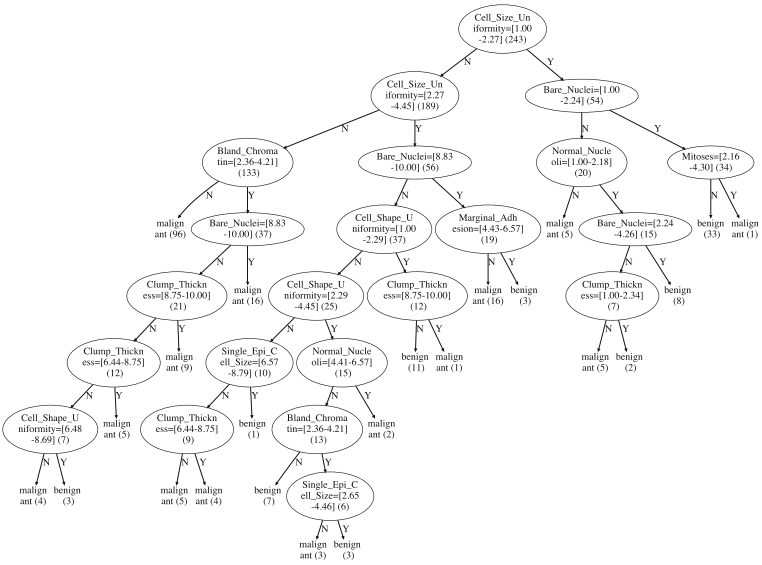


ASR (AUC ROC 0.666 and expected height 5)

Fig. 18 Visualization of ASR decision tree for shoppers dataset

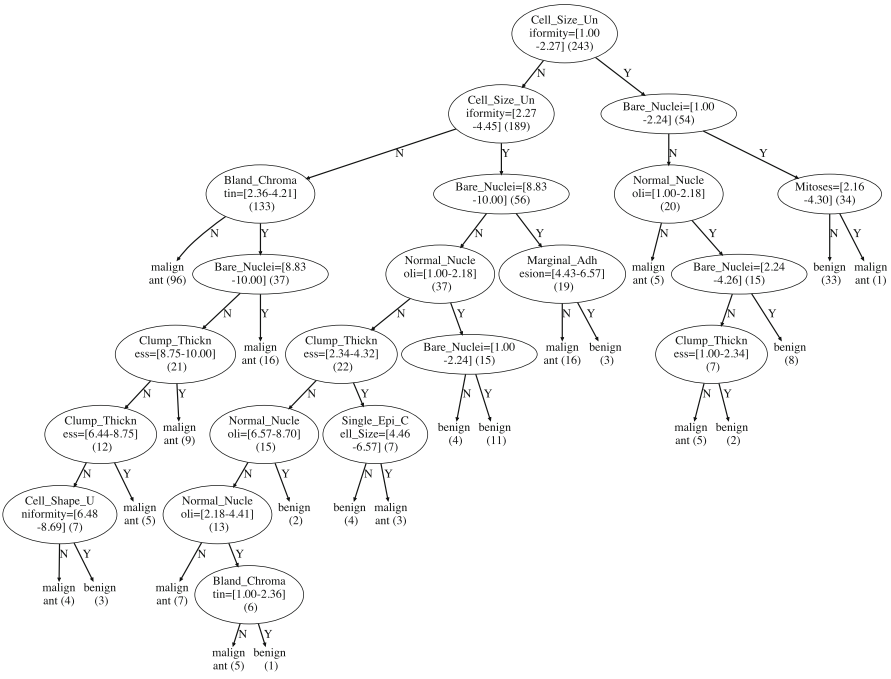
G.2 Visualization of decision trees for breast-w dataset

See Figs. 19, 20 and 21.



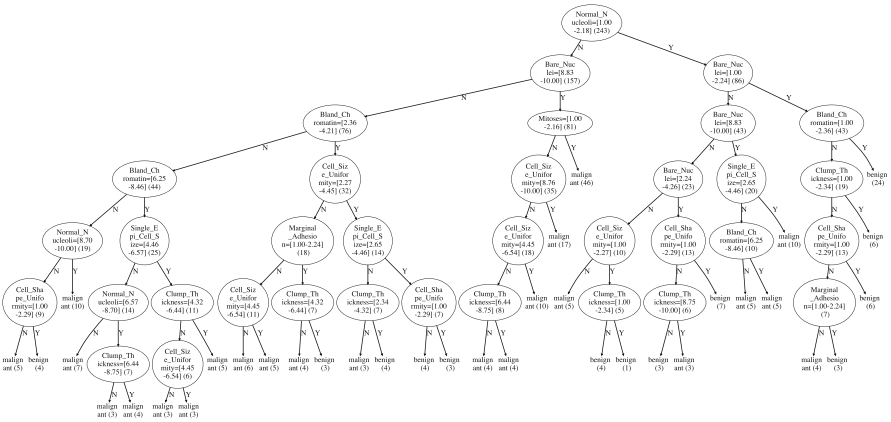
C4.5 (AUC ROC 0.968 and expected height 3.5)

Fig. 19 Visualization of C4.5 decision tree for breast-w dataset



EC4.5 (AUC ROC 0.982 and expected height 3.48)

Fig. 20 Visualization of EC4.5 decision tree for breast-w dataset

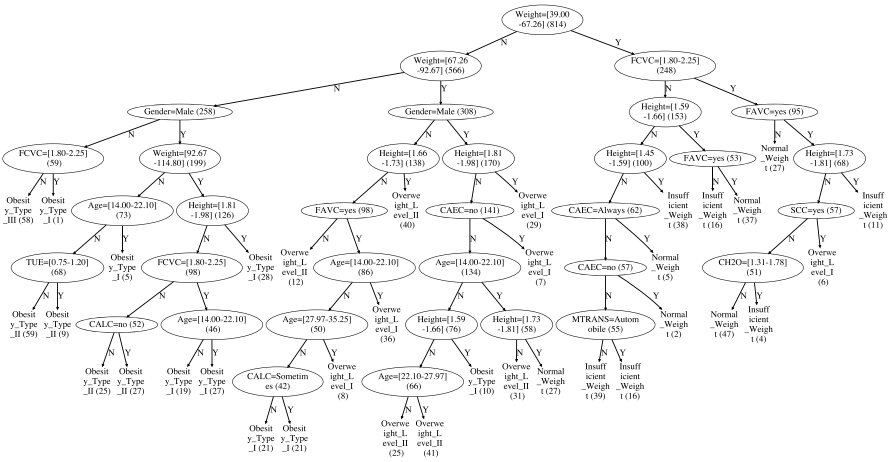


ASR (AUC ROC 0.967 and expected height 4.08)

Fig. 21 Visualization of ASR decision tree for breast-w dataset

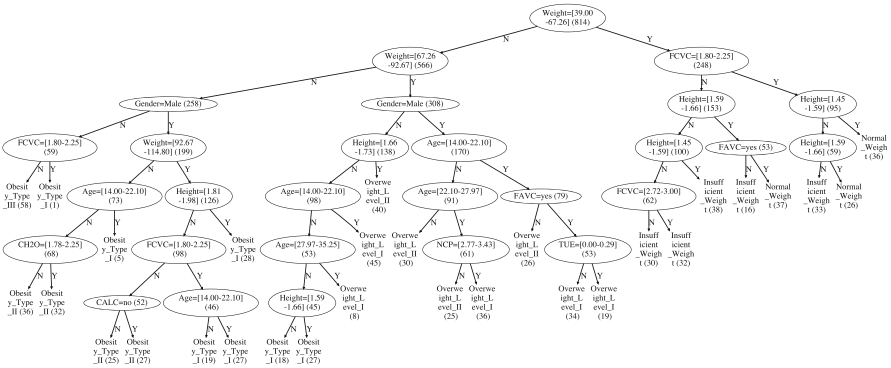
G.3 Visualization of decision trees for obesity dataset

See Fig. 22, 23 and 24.



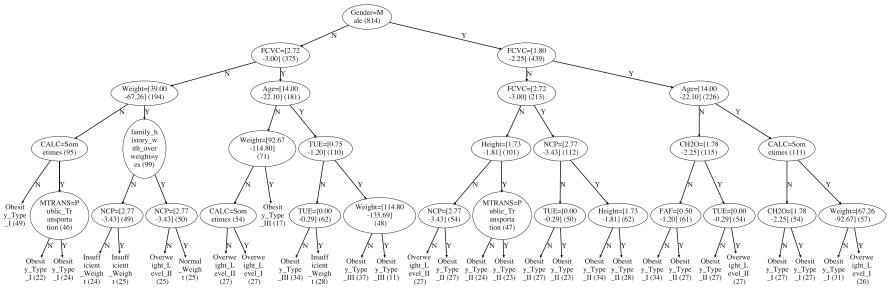
C4.5 (AUC ROC 0.932 and expected height 5.7)

Fig. 22 Visualization of C4.5 decision tree for obesity dataset



EC4.5 (AUC ROC 0.930 and expected height 5.1)

Fig. 23 Visualization of EC4.5 decision tree for obesity dataset

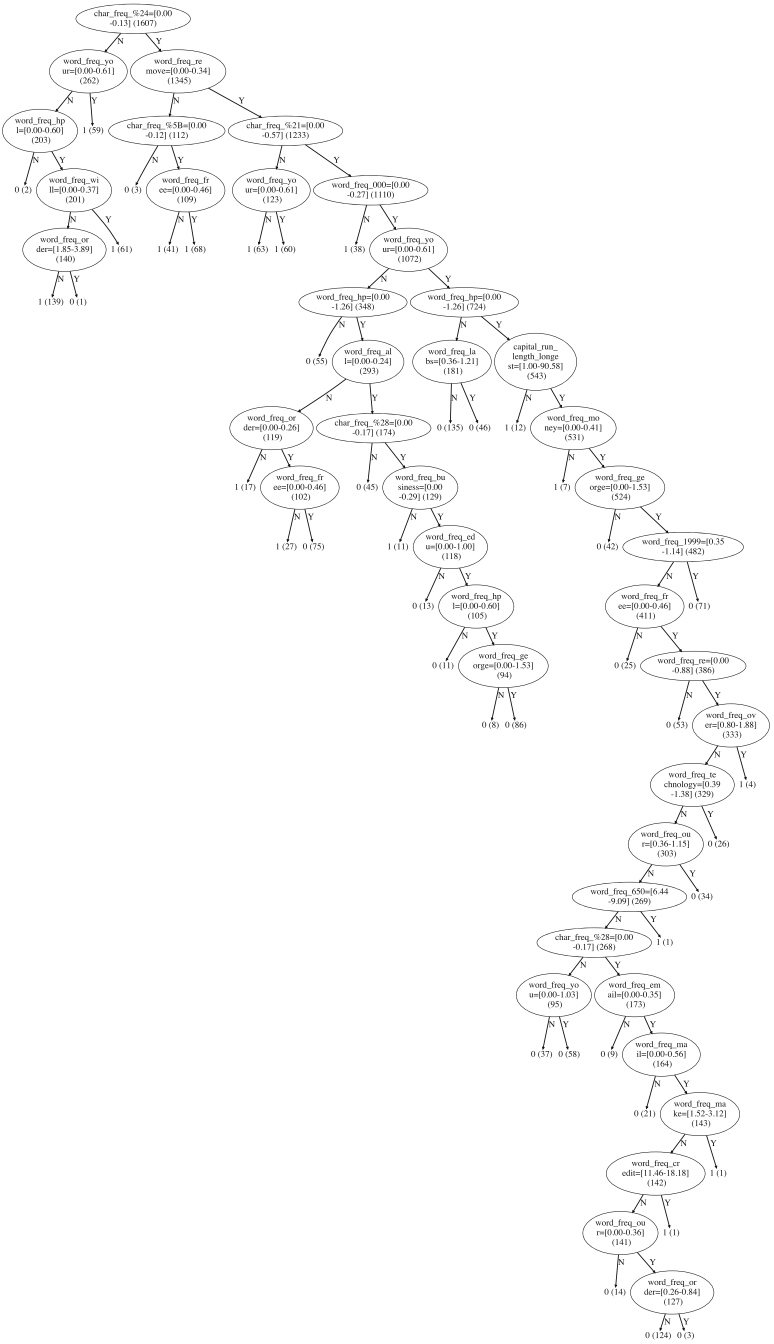


ASR (AUC ROC 0.822 and expected height 4.9)

Fig. 24 Visualization of ASR decision tree for obesity dataset

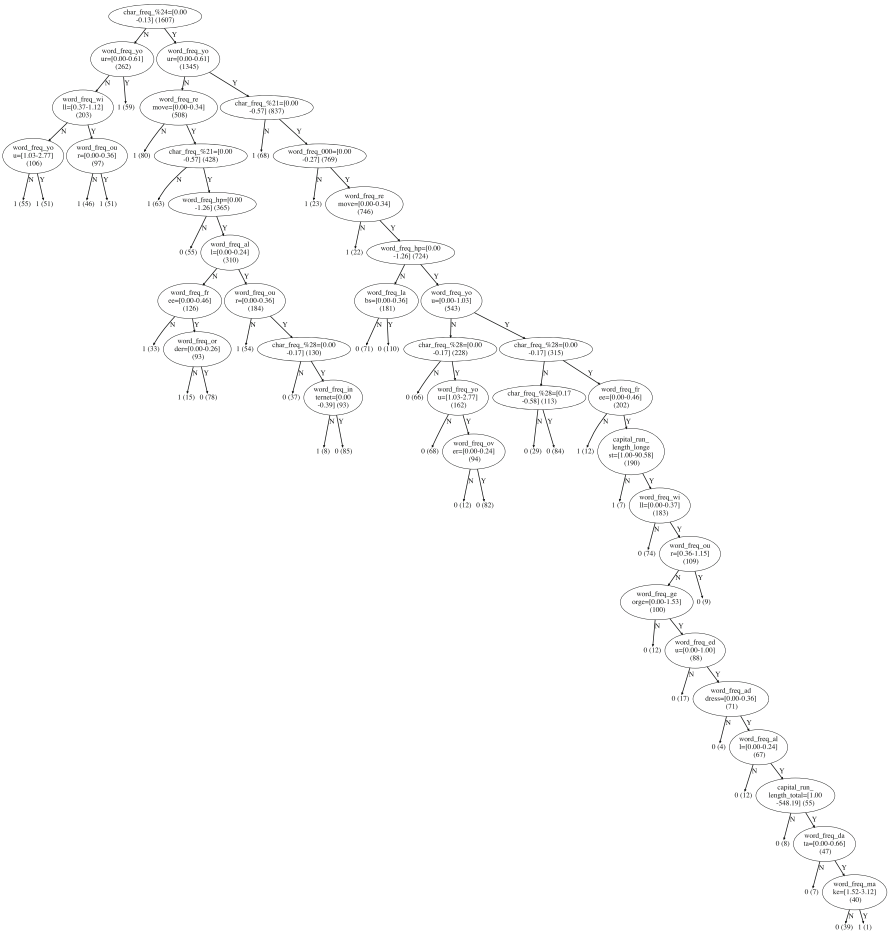
G.4 Visualization of decision trees for spambase dataset

See Figs. 25, 26 and 27.



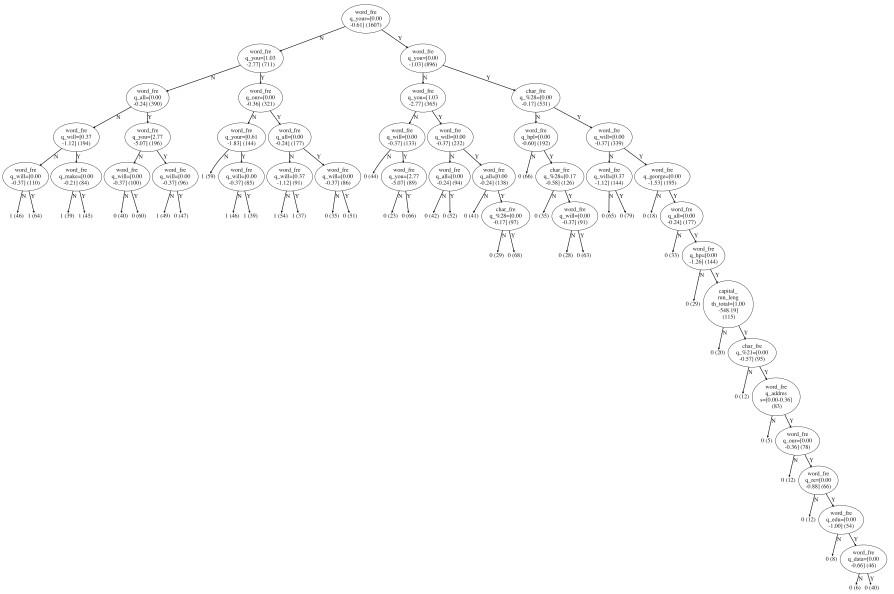
C4.5 (AUC ROC 0.932 and expected height 9.9)

Fig. 25 Visualization of C4.5 decision tree for spambase dataset



EC4.5 (AUC ROC 0.928 and expected height 7.76)

Fig. 26 Visualization of EC4.5 decision tree spambase dataset

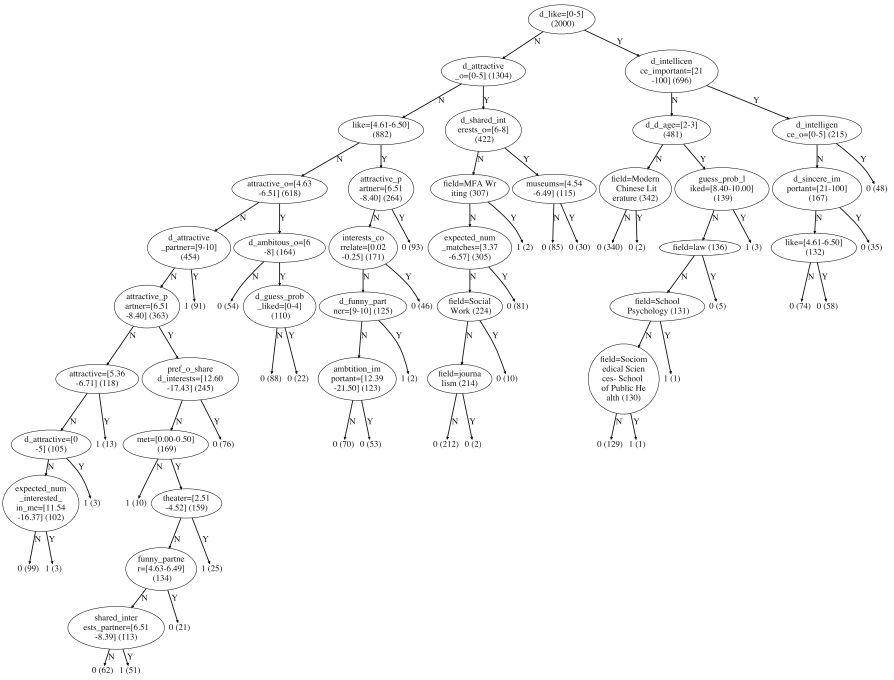


ASR (AUC ROC 0.839 and expected height 5.8)

Fig. 27 Visualization of ASR decision tree for spambase dataset

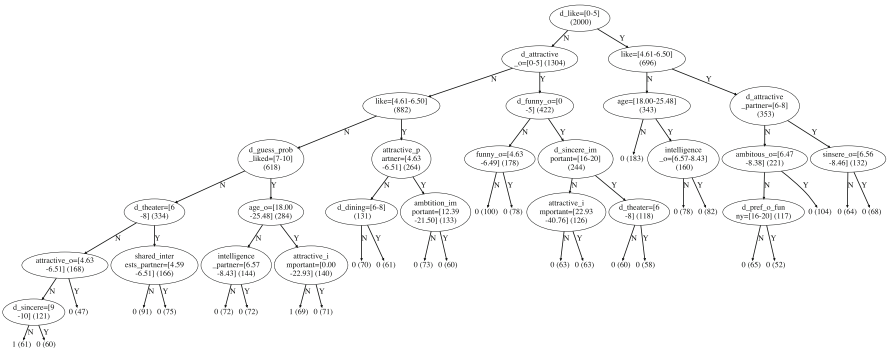
G.5 Visualization of decision trees for speed-dating dataset

See Figs. 28, 29 and 30.



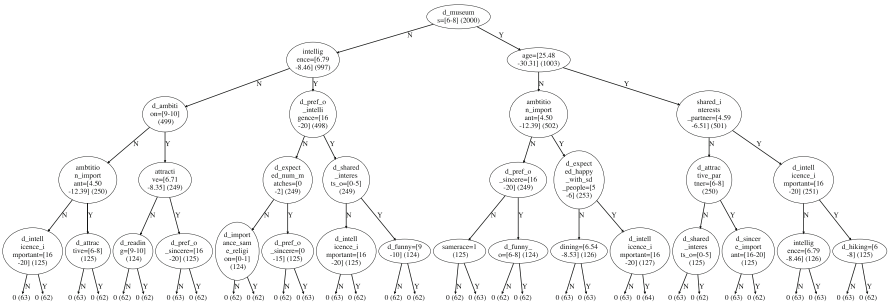
C4.5 (AUC ROC 0.779 and expected height 5.9)

Fig. 28 Visualization of C4.5 decision tree for speed-dating dataset



EC4.5 (AUC ROC 0.790 and expected height 4.9)

Fig. 29 Visualization of EC4.5 decision tree for speed-dating dataset



ASR (AUC ROC 0.569 and expected height 5)

Fig. 30 Visualization of ASR decision tree for speed-dating dataset

References

Adler M, Heeringa B (2008) Approximating optimal binary decision trees. In: Goel A, Jansen k, Rolim JDP, Rubinfeld R (eds) Approximation, randomization and combinatorial optimization. Algorithms and techniques. Springer, Berlin, pp 1–9

Bellala G, Bhavnani SK, Scott C (2012) Group-based active query selection for rapid diagnosis in time-critical situations. IEEE Trans Inf Theory 58(1):459–478

Blanc G, Lange J, Tan LY (2020) Top-down induction of decision trees: rigorous guarantees and inherent limitations. In: 11th innovations in theoretical computer science conference (ITCS 2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik

Breiman L, Friedman J, Stone CJ et al (1984) Classification and regression trees. CRC Press, Boca Raton

Brutzkus A, Daniely A, Malach E (2019) On the optimality of trees generated by ID3. arXiv:1907.05444

Chakaravarthy VT, Pandit V, Roy S et al (2007) Decision trees for entity identification: approximation algorithms and hardness results. In: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, pp 53–62

Cicalese F, Laber E, Saettler AM (2014) Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In: International conference on machine learning, pp 414–422

Dasgupta S (2005) Analysis of a greedy active learning strategy. In: Advances in neural information processing systems, pp 337–344. https://proceedings.neurips.cc/paper/2004/hash/c61fbef63df5ff317aecdc3670094472-Abstract.html

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Deshpande A, Hellerstein L, Kletenik D (2016) Approximation algorithms for stochastic Boolean function evaluation and stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. ACM Trans Algorithms 12(42):1–42

Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning. arXiv:1702.08608

Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml

Feige U (1998) A threshold of ln n for approximating set cover. J ACM: JACM 45(4):634–652

Feige U, Lovász L, Tetali P (2004) Approximating min sum set cover. Algorithmica 40(4):219–234

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

Garey MR (1972) Optimal binary identification procedures. SIAM J Appl Math 23(2):173–186

Golovin D, Krause A (2011) Adaptive submodularity: theory and applications in active learning and stochastic optimization. J Artif Intell Res 42:427–486

Golovin D, Krause A, Ray D (2010) Near-optimal Bayesian active learning with noisy observations. In: Lafferty JD, Williams CKI and Taylor JS and Richard S. Zemel and Aron Culotta Advances in neural information processing systems, pp 766–774

Grammel N, Hellerstein L, Kletenik D et al (2016) Scenario submodular cover. In: International workshop on approximation and online algorithms. Springer, pp 116–128

Guillory A, Bilmes J (2009) Average-case active learning with costs. In: International conference on algorithmic learning theory. Springer, pp 141–155
Guillory A, Bilmes JA (2011) Simultaneous learning and covering with adversarial noise. In: International conference on machine learning
Gupta A, Nagarajan V, Ravi R (2017) Approximation algorithms for optimal decision trees and adaptive TSP problems. Math Oper Res 42(3):876–896
Hancock T, Jiang T, Li M et al (1996) Lower bounds on learning decision lists and trees. Inf Comput 126(2):114–122
Im S, Nagarajan V, van der Zwaan R (2012) Minimum latency submodular cover. In: Czumaj A, Mehlhorn K, Pitts AM, Wattenhofer R (eds) International colloquium on automata, languages, and programming, Vol 7391. Springer, pp 485–497. https://doi.org/10.1007/978-3-642-31594-7_41/
Kearns M, Mansour Y (1999) On the boosting ability of top-down decision tree learning algorithms. J Comput Syst Sci 58(1):109–128
Kosaraju SR, Przytycka TM, Borgstrom R (1999) On an optimal split tree problem. In: Workshop on algorithms and data structures. Springer, pp 157–168
Laurent H, Rivest RL (1976) Constructing optimal binary decision trees is np-complete. Inf Process Lett 5(1):15–17
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
Murthy SK (1998) Automatic construction of decision trees from data: a multi-disciplinary survey. Data Min Knowl Disc 2(4):345–389
Navidi F, Kambadur P, Nagarajan V (2020) Adaptive submodular ranking and routing. Oper Res 68(3):856–877
Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, Burlington
Strobl C, Boulesteix AL, Zeileis A et al (2007) Bias in random forest variable importance measures: illustrations, sources and a solution. BMC Bioinform 8(1):1–21
Vanschoren J, van Rijn JN, Bischl B et al (2013) OpenML: networked science in machine learning. SIGKDD Explor 15(2):49–60. https://doi.org/10.1145/2641190.2641198