Decision Trees with Short Explainable Rules

Ferdinando Cicalese
Department of Computer Science
University of Verona
Verona - Italy
ferdinando.cicalese@univr.it

Eduardo Sany Laber
Departamento de Informática
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
laber@inf.puc-rio.br

Marco Molinaro
Microsoft Research & Pontifical Catholic University of Rio de Janeiro
mmolinaro@microsoft.com

Victor F. Souza
Departamento de Informática
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
vfsouza@inf.puc-rio.br

Abstract

Decision trees are widely used in many settings where interpretable models are preferred or required. As confirmed by recent empirical studies, the interpretability/explainability of a decision tree critically depends on some of its structural parameters, like size and the average/maximum depth of its leaves. There is indeed a vast literature on the design and analysis of decision tree algorithms that aim at optimizing these parameters.

This paper contributes to this important line of research: we propose as a novel criterion of measuring the interpretability of a decision tree, the sparsity of the set of attributes that are (on average) required to explain the classification of the examples. We give a tight characterization of the best possible guarantees achievable by a decision tree built to optimize both our new measure (which we call the explanation size) and the more classical measures of worst-case and average depth. In particular, we give an algorithm that guarantees $O(\ln n)$-approximation (hence optimal if $P \neq NP$) for the minimization of both the average/worst-case explanation size and the average/worst-case depth. In addition to our theoretical contributions, experiments with 20 real datasets show that our algorithm has accuracy competitive with CART while producing trees that allow for much simpler explanations.

1 Introduction

Machine learning models and algorithms appear more and more frequently in systems that make decisions with an impact in our lives. Thus, it is highly desirable that the output of these methods

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

are interpretable so that we can use them more comfortably or, eventually, question its applicability [13].[1]

Decision trees are used in many settings as a tool to provide explainability. However, their explainability greatly depends on the depths of its leaves, as empirically demonstrated by [46]. In fact, based on empirical data from a survey with 98 questions answered by 69 respondents, the authors conclude that "question depth" (the depth of the deepest leaf that is required when answering questions about a tree) turns out to be the most important parameter. Essentially, users prefer trees where the information about the most common items are given at the top of the tree. Minimizing the average/worst-case depth indeed has been a classic goal for decision tree algorithms (see the Related Work section below).

However, another very important component for explainability is having decision rules that are sparse, namely, that use as few different attributes as possible to classify an object or make a prediction. For decision trees, this means that the path to any given leaf should test only a small number of different attributes. Figure 1 shows two trees for the `Sensorless` dataset [7] with similar accuracy. While the trees have the same size, the rightmost one gives much more concise classification rules. As a spoiler, the left tree was constructed using `CART` and the right one using the algorithm we propose in this paper. To exemplify their differences, let us consider the leaves that are marked in the figure with a thick rectangle. Both are assigned the same class and cover the maximum number of examples in the training set (approximately 11.500 example each). Despite this similar behaviour, the explanations for their classifications are quite different (the D_i's denote the attributes):

`CART:`	$D_{11} \in$ `[-0.11, 0.07] AND`
	$D_9 >$ `-0.01 AND`
	$D_{10} \leq$ `0.03`
`New algorithm:`	$D_{11} \in$ `[-0.01, 0.03]`

Define the explanation size of a leaf as the number of distinct attributes tested on the path from the root to this leaf. The above example shows that having trees whose leaves have small explanation size yields significantly simpler (hence easier to interpret/explain) rules. However, to the best of our knowledge there is no prior work considering decision trees optimized with respect to the explanation size.

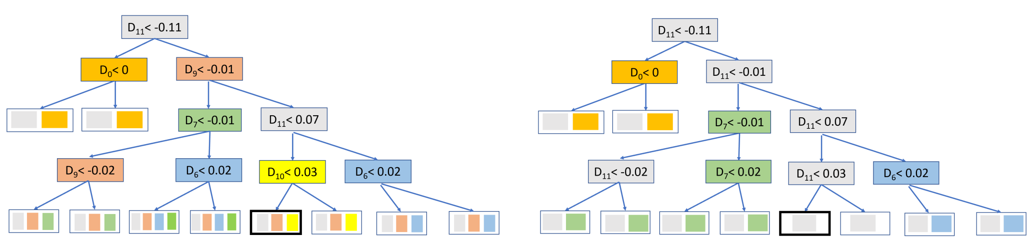

Figure 1: The left and right decision trees are built, respectively, by `CART` and the algorithm proposed in this paper for the `Sensorless` dataset. The maximum depth was set to 4. Internal nodes associated with the same attribute have the same color. The colors inside a leaf indicate the attributes that are used to obtain its classification.

Our contributions. In this work we propose the explanation size as a novel measure to capture the interpretability of a decision tree. In particular, we initiate a principled study of decision trees with small explanation size by focusing on: (i) the trade-off that optimizing this criterion imposes on other desirable metrics (e.g., average/worst-case depth); (ii) the design of efficient algorithms for building such trees that guarantee good performance in practice.

Our first result is that it is possible to essentially obtain a best-of-both worlds in terms of average/worst-case depth and explanation size: We show that there is always a binary decision tree that has the smallest average explanation size possible but also has average depth not much larger than that of the

[1] We use the words interpretability and explainability in a broad sense; for a detailed discussion of the different concepts related to the topic (see, for example, [41]).

optimal tree for the latter metric (Item 2 of Theorem 1). The same result also holds when considering worst-case explanation size and depth (Item 1 of Theorem 1). Remarkably, the latter (worst-case bound) turns out to be tight. Theorem 2 shows it matches a necessary trade-off between optimizing explanation size and depth, i.e., improving the bound on the depth in Theorem 1 can only be attained at the cost of a logarithmic factor loss in the explanation size.

Despite having strong theoretical guarantees, the construction to obtain these trees is too wasteful to be used in practice. Thus, our second contribution is an algorithm that still yields a tree that provably approximates both optimal average/worst-case explanation size and depth (Theorem 3) but has enough flexibility that it can be employed to obtain a good performance in practice. To demonstrate the applicability of our proposal, we compare it against `CART` [11], a quite popular method to build decision trees, on 20 real classification datasets. Our method leads to much more (resp. more) interpretable trees in terms of explanation size (resp. average depth), while having a performance similar to `CART` in terms of accuracy and speed.

2 Related work

Our work can be connected with an active line of research that proposes interpretable models for machine learning [13], in particular more interpretable rule-based models (e.g. decision lists, sets, and trees) [37, 30, 21, 9]. Our interpretability metric is closely related to rule sizes considered in the rule learning literature, e.g., [31, 47]. In addition, we can relate our work with those from the vast literature of methods with provable guarantees that are designed to build decision trees of "low complexity" (e.g. depth of leaves, number of nodes, etc.) [23, 22, 18, 8, 24].

More interpretable rule-based models. There is a body of work that aims to understand what makes a rule model more comprehensible via experiments with end users [4, 29, 46]. The paper [4] compares the comprehensibility of classifiers that are learned by decision trees and rule-learning algorithms, based on subjective comparisons of classifier pairs by 100 Computer Science students. They conclude that decision trees are more comprehensible. In [29], based on experiments with business students, it is concluded that decision tables are more interpretable than decision trees and propositional "if-then" rules. One potential limitation of this study is that these students had no experience with the representation formats, so it is not clear whether this conclusion can be extended for more experienced users. The work of [46] reports a survey with 69 respondents that was carried out to understand what makes a decision tree more interpretable. Among their findings is that the depth of the leaves required by users during the experiments had one of the biggest and most consistent impact on the usability of decision trees across multiple tasks (classify, explain, validate, and discover).

There are some recent works that try to optimize interpretabilty metrics when building rule-based models [37, 10, 56, 30, 40, 9]. We briefly discuss those that focus on decision trees. The paper [30] contends that splits that have at least one of its parts/child nodes being class-homogeneous (roughly this means that most examples have the same class) are important for interpretability, and propose a splitting criterion that tries to balance this homogeneity and the depth of the leaves. In contrast to our work, no provable guarantees are provided. The works [10, 56] employ Integer Linear Programming to build trees of a given maximum depth, while [40] employs Dynamic Programming techniques to develop an optimization framework that allows the construction of decision trees with few leaves that optimize a variety of objective functions such as F-score and `AUC`. These methods, while interesting, are much more complex to implement than ours and they do not run in polynomial time, which may compromise their application on large datasets.

Decision trees with provable guarantees. There is a vast literature dedicated to the problem of building decision trees with "low complexity", where the complexity of a tree can be measured in different ways, including worst-case/average leaf depth and number of nodes, among others [12]. It is known that building a decision tree that minimizes the worst-case or average leaf depth, among those that fit the training data, does not admit an $o(\log n)$ approximation unless P=NP [16, 36]. When the goal is minimizing the number of nodes, the problem is even harder to approximate [3, 26]. Regarding upper bounds, several algorithms with the optimal (within constants) $O(\log n)$-approximation are known for minimizing the worst-case and average depth [23, 22, 17, 18, 8, 24, 32, 39, 44]. What distinguishes each method is the generality of its guarantee; for example, some consider scenarios that include tests with noisy outcomes [32], while others consider items/examples with non-uniform weights and tests with non-uniform costs [18, 44].

We remark that the method we propose here also allows the use of non-uniform weights on the examples, which can be used, for instance, to prioritize models that yield simpler explanations for some classes of particular interest, by setting high weights for examples in these classes. The key difference between our method and the existing ones is that ours is the only one that provides theoretical guarantees on the explanation size.

3 Model

We describe the model used throughout for the theoretical analyses. For that, we adopt a terminology similar to that employed by some closely related works [1, 18]. On an instance I, there is a set of ordinal attributes A and a set of objects O, where each object $o \in O$ is described by the value it takes on each attribute $a \in A$; we denote this value by $a(o)$. Each object o also has a class $c(o)$ in some set C. In order to classify an object o, we are allowed to use threshold tests of the form "Is $a(o) < t$?" for some attribute a and threshold value t.

A (threshold) decision tree T is a rooted tree where each internal node ν is associated with a threshold test "Is $a(o) < t$?" and the edges from a node to its children are associated to the two possible outcomes "$a(o) < t$" and "$a(o) \geq t$". We also refer to this test by the attribute/threshold pair (a, t). Each leaf ℓ of T is associated with a class in C. Given a decision tree T, we say that an object o reaches a node ν of T if it agrees with all outcomes associated with the path from the root of T to ν. For each $o \in O$, we use $\ell(o)$ to denote the unique leaf reached by object o. Finally, we consider the exact classification model, namely a decision tree must correctly classify all objects of the instance, i.e., each object $o \in O$ reaches some leaf associated to its correct class $c(o)$.

We now formalize the measures of interpretability that were mentioned in the introduction.

Depth and explanation size of a tree. We start recalling the classical notions of worst-case and average tree depth. Given a decision tree T, the depth of a leaf ℓ is the number of tests/internal-nodes on the path from the root of T to ℓ, and is denoted by depth(ℓ). The worst-case depth of the tree T is obtained by considering the maximum depth over its leaves, namely

$$\mathrm{depth}_{wc}(T) := \max_{\ell \in \text{leaf}(T)} \text{depth}(\ell).$$

To measure the average depth of the tree, in addition to the datum above, as part of the input each object o has a non-negative weight $w(o) \in \mathbb{R}_+$ indicating its likelihood/importance. Letting $w(\ell)$ be the sum of the weights of all objects that reach leaf ℓ, the average depth of the tree T is then the weighted sum of the depth of its leaves, namely

$$\mathrm{depth}_{avg}(T) := \sum_{\ell \in \text{leaf}(T)} w(\ell) \cdot \text{depth}(\ell).$$

We now consider the novel measures of quality/interpretability of a tree using the notion of explanation size. The explanation size of a leaf ℓ, denoted by expl(ℓ), is the number of different attributes on the tests on the path from the root of T to ℓ. As an example, the explanation size of the leaf marked with a thick rectangle on the left tree of Figure 1 is 3. The worst-case and average explanation size of a tree T are then given as before by the largest and the weighted sum of the explanation sizes of its leaves, respectively:

$$\begin{aligned}\mathrm{expl}_{wc}(T) &= \max_{\ell \in \text{leaf}(T)} \text{expl}(\ell),\\ \mathrm{expl}_{avg}(T) &= \sum_{\ell \in \text{leaf}(T)} w(\ell) \cdot \text{expl}(\ell).\end{aligned}$$

Our goal is to obtain trees with as small worst-case/average depth and explanation size as possible. We use $\mathrm{depth}^*_{wc} = \mathrm{depth}^*_{wc}(I)$ to denote the smallest possible worst-case depth of a decision tree that solves instance I, and define the optimal values expl^*_{wc}, depth^*_{avg}, and expl^*_{avg} analogously.

4 Tradeoff between depth and explanation size

Our goal is to obtain trees that simultaneously have both small worst-case/average depth and explanation size. However, is this even possible? It is conceivable that in order to obtain trees with small depth, one may be required to use several different attributes along the paths to effectively classify the objects; but this would make the tree have a large explanation size. Conversely, in a tree with small explanation size, the few attributes along a path may need to be used many times in order to correctly classify the objects, leading to a large tree depth.

Perhaps surprisingly, we show that there are trees that simultaneously have optimal explanation size and almost optimal depth.

Theorem 1. *Given an instance of the classification problem, the following holds:*

1. *(Worst-case metrics) There exists a binary tree T for which simultaneously*
   - $\mathrm{expl}_{wc}(T) = \mathrm{expl}^*_{wc}$
   - $\mathrm{depth}_{wc}(T) \leq 2\,\mathrm{depth}^*_{wc} + \log n$.
2. *(Average metrics) There exists a binary tree T for which simultaneously*
   - $\mathrm{expl}_{avg}(T) = \mathrm{expl}^*_{avg}$
   - $\mathrm{depth}_{avg}(T) \leq 2\,\mathrm{depth}^*_{avg} + W \log n.$,

where W is the sum of the weights of the object in the instance.

We remark that these additive bounds on the worst-case and average depth imply $O(\log n)$ multiplicative approximations as well.

Observation 1. *The bounds from Theorem 1 imply*

$$\mathrm{depth}_{wc}(T) \leq \frac{3\log n}{\log c} \cdot \mathrm{depth}^*_{wc}$$
$$\mathrm{depth}_{avg}(T) \leq 3\log n \cdot \mathrm{depth}^*_{avg},$$

where c is the number of classes.

While the proof of Theorem 1 is deferred to Appendix B, we give here the main ideas behind it. We will consider here only the worst-case metric (Item 1), since the proof is simpler and more transparent.

To show the existence of our desired tree we make use of multiway trees, i.e., a decision tree where multiway tests are used rather than threshold tests. A multiway test associated with attribute a splits the objects based on all possible values of this attribute. As an example, if an attribute a takes 5 distinct values for the objects in the instance and we use a at the root of a multiway tree, then the root will have 5 children.

The starting point for the construction of the tree in Theorem 1 is the equivalence between optimal binary trees and optimal multiway trees in terms of worst-case explanation size. While this is formally proved in Lemmas 2 and 3, for an intuitive view of this equivalence first notice that there is a multiway tree M^* that is simultaneously optimal in terms of worst-case depth and worst-case explanation size, since each attribute only needs to be used once in a path. Also, this optimal multiway tree M^* has worst-case explanation size (equivalently worst-case depth) at most that of the best binary tree, namely $\mathrm{depth}_{wc}(M^*) = \mathrm{expl}_{wc}(M^*) \leq \mathrm{expl}^*_{wc}$, since intuitively multiway tests are more informative than binary tests. Conversely, we can transform an optimal multiway tree M^* into a binary decision tree T by simulating each multiway test on an attribute a by using multiple threshold tests "Is $a(o) < t$" with varying t (but same attribute a). Since explanation size only counts the number of distinct attributes used along a path, the tree T so created has exactly the same explanation sizes as M^*, and hence $\mathrm{expl}_{wc}(M^*) = \mathrm{expl}_{wc}(T) \geq \mathrm{expl}^*_{wc}$. Thus, we have the equivalence $\mathrm{expl}_{wc}(M^*) = \mathrm{expl}^*_{wc}$.

To prove Theorem 1, we start with the optimal multiway tree M^* and convert it into a binary tree, as above. However, the conversion in the previous paragraph is not enough: while it preserves the explanation size, it may greatly increase the depth of the leaves when using multiple threshold tests to simulate a multiway test (possibly yielding depth $\gg \mathrm{depth}^*_{wc}$). The key idea is to use a much more efficient simulation that is based on alphabetic codes, a classic notion from coding theory [28].

The following result from [2, Chp. 2, p. 341], rephrased in the terminology of decision trees, gives a sufficient condition for the existence of such codes with prescribed code-lengths d_i's.

Lemma 1. *Consider an instance of the classification problem with n objects but only 1 attribute. Then for any positive integers $d_1, \ldots, d_n$ such that $\sum_{j=1}^{n} 2^{-d_i} \leq \frac{1}{2}$, there exists a binary threshold decision tree with n leaves at depths $d_1, \ldots d_n$ such that the i-th object reaches the leaf at depth d_i.*

Then for each node ν of M^* (corresponding to an attribute a, and with children $ch_1, ch_2, \ldots$), we consider all objects that reach a child ch_i as a "single object" (with the corresponding value in attribute a) and applying the previous lemma we replace ν by a binary tree A where the leaf corresponding to the objects in ch_i end up in a leaf at depth (in A) $\ell_i = \lceil \log \frac{n(\nu)}{n(ch_i)} \rceil + 1$, where $n(node)$ is the number of objects that reach a given node in M^*. The final tree obtained, call it T, still has the same explanation sizes as M^*, so $\mathrm{expl}_{wc}(T) = \mathrm{expl}^*_{wc}$. Moreover, in terms of worst-case depth, any root-to-leaf path P in T has a corresponding path $P_{M^*} = \nu_0, \nu_1, \nu_2, \ldots$ in M^*, and the length of P is at most

$$\sum_{i=0}^{|P_{M^*}|-1} \left(\left\lceil \log \frac{n(\nu_i)}{n(\nu_{i+1})} \right\rceil + 1 \right) \;\leq\; \log n + 2 \cdot [\text{length of } P_{M^*}].$$

By looking at the longest such path we get

$$\mathrm{depth}_{wc}(T) \;\leq\; \log n + 2\, \mathrm{depth}_{wc}(M^*) \;\leq\; \log n + 2\, \mathrm{depth}^*_{wc},$$

where the last inequality holds because of the optimality of M^* with respect to worst-case depth. This gives Item 1 of Theorem 1.

The average-case part of the theorem (Item 2) uses similar ideas, but in addition relies on entropy-based calculations to argue about the average depth of the constructed tree.

Observation 2. *Theorem 1 is an existential result and the construction outlined above cannot be done in polytime, since it relies on the availability of an optimal multiway tree. However, one can obtain in polytime a tree that is simultaneously an $O(\log n)$-approximation for both worst-case (respectively average) explanation size and depth by replacing the optimal multiway tree by one that approximates within a factor of $O(\log n)$ the worst-case (resp. average) depth, which can be found in polytime (see, e.g., [18] and references therein quoted).*

Although guaranteeing asymptotically the desired optimal approximation, the construction leading to such trees might be wasteful in practice as it involves the use of distinct alphabetic codes to turn each multiway tests into a short sequences of threshold tests. Therefore, we present an alternative approach in Section 5 that achieves the same approximation guarantee and has also very good performance in practice.

4.1 Lower bound

Given these positive results, a natural question is whether it is possible to obtain a tree that is optimal for both depth and explanation size. The next result answers this in the negative, and shows that in a way the worst-case bound in Theorem 1 cannot be improved.

Theorem 2. *Fix c and $n \geq cst \cdot c$ for a sufficiently large constant cst. Then for every $\alpha \in [\frac{1}{2\log c}, \frac{1}{2}(\frac{\log(n/2)}{\log c} - 1)]$, there is a classification instance with n examples and c classes such that every binary decision tree T for this instance has either*

$$\mathrm{depth}_{wc}(T) > \frac{\alpha}{2} \cdot \mathrm{depth}^*_{wc}$$

or

$$\mathrm{expl}_{wc}(T) > \frac{1}{4\alpha} \cdot \log\left(\frac{n}{c}\right) \cdot \mathrm{expl}^*_{wc}.$$

Remark 1. *To get a more concrete idea for this lower bound, consider setting α at its upper limit, namely $\alpha \approx \frac{1}{2}\frac{\log n}{\log c}$. In this case we obtain that every tree with $\mathrm{depth}_{wc}(T) \lesssim \frac{1}{4}\frac{\log n}{\log c}\,\mathrm{depth}^*_{wc}$ must have $\mathrm{expl}_{wc}(T) \geq \Omega((1 - \frac{\log c}{\log n})\log c)\,\mathrm{expl}^*_{wc}$, which is $\Omega(\log n)\,\mathrm{expl}^*_{wc}$ if we set $c = \frac{\log n}{2}$. Comparing this against Theorem 1 and Observation 1 we see an interesting and subtle phenomenon: while you can have a tree with $\mathrm{depth}_{wc}(T) \leq \frac{3\log n}{\log c}\,\mathrm{depth}^*_{wc}$ and optimal $\mathrm{expl}_{wc}(T)$, if you require the approximation in the depth to be a constant factor smaller then you must lose a logarithmic factor in the approximation in the explanation size.*

5 An efficient and practical algorithm

In this section we design an algorithm, which we name Short Explanaible Rules (SER-DT), that always yields trees of approximately optimal average/worst-case explanation size and depth. Importantly, our method has enough flexibility that it can be tuned to trade-off accuracy and interpretability, as shown in our computational experiments (Section 6).

Similar to other algorithms in the area, ours chooses in each step a split that creates subtrees with "small impurity". However, unlike most such algorithms, it is not completely greedy and allows for the desired extra flexibility in the choices.

To describe the algorithm, consider a set of objects $S \subseteq O$ and let $S(a, i)$ (respectively $S(a, \leq i)$ and $S(a, > i)$) be the set of objects in S with value equal (respectively at most and larger than) i on attribute a. Moreover, let $w(S) := \sum_{o \in S} w(o)$ denote its total weight. Let o, o' be a pair of objects in S such that $c(o) \neq c(o')$. We will refer to such a pair as a misclassified pair because if both o and o' reach the same leaf in the decision tree then one of them will be surely misclassified.

We use $P(S)$ to denote the number of misclassified pairs in S. This quantity can be thought of as a measure of the amount of work that is needed to reach a correct classification of all objects in S and it has been previously used in [19, 22, 18]. To take into account also the importance/weight of set S we define $\texttt{wpm}(S) := P(S) \cdot w(S)$ as the weighted pair-wise misclassification of S.

As a pre-processing step, before executing SER-DT, each weight $w(o)$ smaller than $w(O)/n^3)$ is replaced with $w(O)/(w_{min} n^3)$, where w_{min} is the smallest positive weight among the objects in O. This idea (from [35]) is important to guarantee a logarithmic dependence on n instead of $w(O)$. After this preprocessing, SER-DT is called for the set of objects O.

The pseudo-code description of SER-DT is presented in Algorithm 1, First SER-DT tries to use **any** balanced test that reduces the weighted pair-wise misclassification of the current set of objects (in the worst case) by at least a $\frac{1}{2}$ factor. In any path of the tree built by SER-DT the amount of these balanced tests is at most logarithmic, so they can be easily handled in our analysis. If no balanced test exists, then the algorithm finds an attribute a^* and value t^* such that in the ternary split $S(a^*, < t)$, $S(a^*, t^*)$, $S(a^*, > t^*)$, only the middle set $S(a^*, t^*)$ has weighted pair-wise misclassification larger than the desired $\frac{1}{2}\texttt{wpm}(S)$. This 3-way partition is obtained by using two binary splits. Then, the algorithm recurses on each set. A critical issue is to show that in this case some progress is also achieved with the problematic subproblem on $S(a^*, t^*)$ where $\texttt{wpm}(S(a^*, t^*)) > \frac{1}{2}\texttt{wpm}(S)$. In fact, the choice of the attribute a^* is such that, the instance $S(a^*, t^*)$ has the minimum weighted pair-wise misclassification among all the attributes and other possible tripartitions. As a result, we can employ a lower bound on the optimum (Lemma 8 in appendix) that allows us to absorb the cost of the subtree for the subproblem $S(a^*, t)$ in the logarithmic guarantee.

Algorithm 1 SER-DT (S : set of objects)

if all objects in S are assigned to the same class, create a leaf assigned to such class and **return**
if there is a test τ that splits S into S^L and S^R such that $\max\{\texttt{wpm}(S^L), \texttt{wpm}(S^R)\} \leq \frac{1}{2}\texttt{wpm}(S)$ **then**
  Use any such test, say $\tau = (a, t)$, as the root of the decision tree
  Recurse on the children $S(a, \leq t)$ and $S(a, > t)$
else
  Let a^* be an attribute in $\underset{a}{\operatorname{argmin}}\{\max_i \texttt{wpm}(S(a, i))\}$
  Let t^* be the smallest value of the attribute a^* such that the "left child" $S(a^*, \leq t^*)$ satisfies

$$\texttt{wpm}(S(a^*, \leq t^*)) \geq \frac{1}{2} \cdot \texttt{wpm}(S).$$

  Use two binary tests to simulate the 3-way split $S(a^*, < t^*)$, $S(a^*, t^*)$, $S(a^*, > t^*)$. More precisely, at the root use a test on attribute a^* that splits S into the sets $S(a^*, < t^*)$ and $S(a^*, \geq t^*)$. Then, apply a test on the right child of the root, currently associated with $S(a^*, \geq t^*)$, creating two new children with objects $S(a^*, t^*)$ and $S(a^*, > t^*)$
  Recurse on each of the three leaf nodes in the current tree
end if

The following is the promised guarantee for the average/worst-case depth and explanation size of the trees produced by the algorithm.

Theorem 3. *Given an instance I, the algorithm SER-DT produces a tree T that satisfies*

1. *(Worst-case metrics)*
   - $\text{depth}_{avg}(T) \leq O(\log n)\, \text{depth}^*_{avg}$,
   - $\text{expl}_{avg}(T) \leq O(\log n)\, \text{expl}^*_{avg}$.
2. *(Average metrics)*
   - $\text{depth}_{wc}(T) \leq O(\log n)\, \text{depth}^*_{wc}$,
   - $\text{expl}_{wc}(T) \leq O(\log n)\, \text{expl}^*_{wc}$.

We remark that, in the light of the inapproximability results from [36, 16][2] this theorem says that SER-DT (with the preprocessing step) guarantees the best possible approximation obtainable by a polynomial algorithm with respect to both the measures under consideration: worst/average depth and worst/average explanation size.

6 Experiments

In this section, we report the experiments that were carried out to evaluate how our proposed algorithm SER-DT performs in practice.

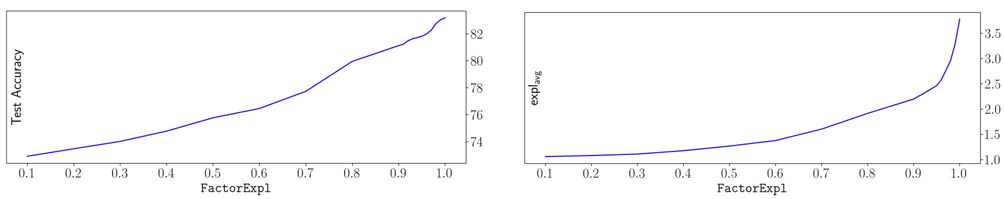

Figure 2: The left (resp. right) image shows the average accuracy (resp. expl_{avg}) over the 20 datasets as a function of `FactorExpl`.

We considered the 20 datasets that appear on Column 1 of Table 1 (see Appendix E for their main characteristics). For all of them, 70% of the examples were used for training and the remaining 30% for testing. Moreover, all the examples (objects) were considered equally important (weight=1/size of training set). During a preprocessing step we converted all categorical attributes into binary attributes via one-hot-encoding. See also Appendix E for more details on the experimental setup.

Recall that in (Line 2) of algorithm SER-DT any test that splits the current set of objects into subsets of small enough `wpm` could be used. We use this flexibility to select a test among these that should further help in obtaining small explanation sizes and high accuracy in practice. To explain our selection, recall the `Gini` impurity measure employed by `CART`. For a set of examples (objects) S, each of them labeled with a class in $\{1, \ldots, c\}$, the `Gini` impurity is given by

$$\texttt{Gini}(S) = 1 - \sum_{i=1}^{c} \left(\frac{|S_i|}{|S|} \right)^2,$$

where S_i is the set of examples of class i. Moreover, the weighted `Gini` impurity $\texttt{Gini}(\tau, \nu)$ induced by a test τ that divides the set of examples S that reach a node ν into S^L and S^R is given by

$$\texttt{Gini}(\tau, \nu) = \frac{|S^L|}{|S|}\texttt{Gini}(S^L) + \frac{|S^R|}{|S|}\texttt{Gini}(S^R).$$

Let `FactorExpl` be a hyper-parameter in the range $[0, 1]$. Because we are interested in trees that induce accurate classifiers with short explanation rules, to expand a given leaf ν, in Line 2 of the algorithm we select, among the permissible tests [3] the test τ that minimizes

$$\texttt{AdjustedGiniExpl}(\tau, \nu) := \texttt{I}(\tau, \nu) \times \texttt{Gini}(\tau, \nu),$$

[2] The hardness of approximation holds also for instances with only binary attributes. In this case explanation size coincides with depth and every test can be considered a threshold test.

[3] A permissible τ must satisfy $\texttt{Gini}(\tau, \nu) < \texttt{Gini}$ (Examples that reach ν), in addition to respect the condition of Line 2

where $\texttt{I}(\tau, \nu) = \texttt{FactorExpl}$ if the attribute associated with test τ has already appeared in the path from the root to ν, and $\texttt{I}(\tau, \nu) = 1$ otherwise. Since FactorExpl is used to favour attributes that have already appeared in the path, when FactorExpl is set to a low value we expect to obtain trees with short explanations but also with lower accuracy, as the effect of the Gini impurity is reduced.

In terms of stopping rules, we do not expand a leaf ν if it is either located at depth 6 or if there is no test τ for which $\texttt{Gini}(\tau, \nu)$ is smaller than $\texttt{Gini}(\text{Examples thaat reach } \nu)$. As a post-processing step, whenever two sibling leaves are assigned the same class, we delete them both and leave their parent as a leaf.

In our first experiment we study how the accuracy and the interpretability measures of the trees produced by our algorithm behave when FactorExpl is varied. Figure 2 shows the average accuracy (left image) and the average explanation size expl_{avg} (right image) as a function of FactorExpl. More precisely, for the left image, the y-value associated with a point x corresponds to the average accuracy on the testing set, calculated over the 20 datasets, when our algorithm is executed with $\texttt{FactorExpl} = x$. For the right image the same logic holds. As expected, the larger the FactorExpl the larger the accuracy and the expl_{avg}. The **interesting finding**, however, is that the accuracy increases relatively slow, for FactorExpl is close to 1, compared with the growth in expl_{avg} (see the Table 5 in the appendix for experiments with FactorExpl in the range [0.95,0.99]). This suggests that it is possible to obtain trees that are significantly more interpretable without sacrificing the accuracy.

Table 1: Test Accuracy, expl_{avg} and expl_{wc} for $\texttt{FactorExpl} = 0.97$. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set. Boldface values indicate a difference of more than 1% (columns 2,3) or a gain of at least 25% in favour of SER-DT (columns 4,5,6 and 7).

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	SER-DT	CART	SER-DT	CART	SER-DT	CART
anuran	94,8%	94,7%	4,78	5,24	6,0	6,0
audit risk	99,9%	99,9%	1,00	1,00	1,0	1,0
avila	61,5%	**63,2%**	**3,06**	4,22	4,9	5,4
banknote	97,6%	98,1%	2,44	2,55	3,8	3,4
bankruptcy polish	96,6%	96,9%	**2,56**	4,63	5,6	5,9
cardiotocography	89,5%	89,8%	4,30	5,30	5,9	6,0
collins	13,2%	**15,6%**	**2,13**	4,76	**4,4**	5,9
default credit card	82,0%	81,9%	**1,45**	4,29	**4,5**	6,0
dry bean	90,1%	89,8%	**3,32**	4,45	5,1	6,0
eeg eye state	74,1%	73,6%	3,69	4,29	5,9	6,0
htru2	97,7%	97,7%	**1,20**	2,03	4,3	4,9
iris	94,2%	93,6%	1,75	1,76	3,1	3,4
letter recognition	44,9%	**47,9%**	**3,34**	5,50	5,5	6,0
mice	99,9%	99,9%	3,05	3,05	3,6	3,6
obs network	**91,7%**	89,5%	3,48	4,26	5,3	5,9
occupancy room	99,4%	99,3%	4,18	4,54	5,3	5,7
online shoppers intention	89,3%	89,8%	3,30	4,00	5,1	6,0
pen digits	**88,6%**	86,9%	4,76	5,31	5,8	6,0
poker hand	52,9%	**55,0%**	**1,80**	4,30	**3,8**	5,1
sensorless	**87,4%**	80,1%	**2,94**	4,03	4,9	5,5
Average	82,3%	82,2%	2,93	3,97	4,69	5,19

In our second experiment we compare the results of our method, using $\texttt{FactorExpl} = 0.97$ with CART [11]. The value 0.97 is motivated by the above observation. To expand a leaf ν, recall that CART selects the test τ for which $\texttt{Gini}(\tau, \nu)$ is minimum and it only expands ν if there is a test τ for which $\texttt{Gini}(\tau, \nu) < \texttt{Gini}(\text{Examples that reach } \nu)$. To provide a fair comparison with our algorithm we set the maximum depth to 6 and applied the same aforementioned post-processing. Table 1 shows the accuracy on the testing set as well as the average explanation size expl_{avg} and the worst-case explanation size expl_{wc} for all datasets. Each entry in this table is the average of ten runs, where in each of them a different seed is used to split a dataset into training and testing set.

We notice that the accuracy of our method is very close to that obtained by CART, while the gain in terms of the interpretability metrics is significant. On 7 datasets (bold-faced on columns 2 and 3) we observe a difference larger than 1% on the accuracies; on 3 of them our algorithm outperforms CART while on the remaining 4, CART is better. In terms of the average explanation size, our algorithm is at least as good as CART for all datasets, and for 9 of them (bold-faced on columns 4 and 5) it improves the expl_{avg} by at least 25%. For the expl_{wc} the gain is also clear. For all datasets, but Banknote, our algorithm is at least as good as CART. Moreover, for 3 of them (bold-faced on columns 6 and 7), it provides a gain of at least 25%. Boxplots for the experiments in Table 1 are provided in Section E.5 of the appendix.

In Appendix E we compare CART and SER-DT in terms of depth_{avg} and depth_{wc}. For depth_{avg}, SER-DT performs better than CART while for depth_{wc} the results are similar. The result for depth_{wc} is not surprising since we set 6 as the maximum depth in our experiments. Regarding running time, the algorithms present similar behaviour, as it can be verified in Appendix E. This is somehow expected since both consist of mostly a greedy split selection at each node.

In the appendix we also present additional experiments and analyses. In particular, in Section E.8, we evaluate the impact of applying post-pruning to both CART and SER-DT. Moreover, in Section E.7, we show comparisons of SER-DT with one of the state of the art decision tree methods that optimize the average depth, namely the EC2 algorithm from [22]: the experiments suggest that SER-DT performs significantly better than EC2 on all metrics.

7 Conclusion

In this work, we proposed the explanation size as a new metric to capture intrepretability of decision trees and initiated a principled study of it. We presented upper and lower bound on the trade-off of simultaneously optimizing this new metric and metrics related to depths of the leaves.

We also proposed a practical algorithm that provably approximates the average explanation size and the average depth and showed, via experiments over 20 datasets, that it is competitive with the widely used CART algorithms in terms of accuracy while being much better in terms of producing trees with short explanation size.

On the basis of both the theoretical analysis (approximation guarantee) and the performance demonstrated in the empirical studies, we believe that our algorithm (or some variation based on its ideas) can be used to generate accurate and highly interpretable trees for practical applications.

Acknowledgments and Disclosure of Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The work of the second author is partially supported by CNPq (grant 307572/2017-0), by FAPERJ, grant E- 26/202.823/2018 and by the Air Force Office of Scientific Research under award number FA9550-22-1-0475.

The third author is supported by Bolsa de Produtividade em Pesquisa #312751/2021-4 from CNPq, and FAPERJ grant Jovem Cientista do Nosso Estado.

References

[1] Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. Algorithmica, 62(3-4):1112–1121, 2012.

[2] Rudolf Ahlswede and Ingo Wegener. Search problems. John Wiley & Sons, Inc., 1987.

[3] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. Learnability and automatizability. In 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings, pages 621–630. IEEE Computer Society, 2004.

[4] Hiva Allahyari and Niklas Lavesson. User-oriented assessment of classification model understandability. In Anders Kofod-Petersen, Fredrik Heintz, and Helge Langseth, editors, Eleventh Scandinavian Conference on Artificial Intelligence, (SCAI) 2011, Trondheim, Norway, May 24th - 26th, 2011, volume 227 of Frontiers in Artificial Intelligence and Applications, pages 11–19. IOS Press, 2011.

[5] E. Alpaydin and Fevzi Alimoglu. Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998.

[6] Banknote Authentication. UCI Machine Learning Repository, 2013.

[7] Martyna Bator. Dataset for Sensorless Drive Diagnosis. UCI Machine Learning Repository, 2015.

[8] Gowtham Bellala, Suresh K. Bhavnani, and Clayton Scott. Group-based active query selection for rapid diagnosis in time-critical situations. IEEE Transactions on Information Theory, 58(1):459–478, 2012.

[9] Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. Interpretable random forests via rule extraction. In Arindam Banerjee and Kenji Fukumizu, editors, The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event, volume 130 of Proceedings of Machine Learning Research, pages 937–945. PMLR, 2021.

[10] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. Mach. Learn., 106(7):1039–1082, 2017.

[11] Leo Breiman, Jerome Friedman, Charles J Stone, and R A Olshen. Classification and Regression Trees. Chapman & Hall/CRC, Philadelphia, PA, jan 1984.

[12] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. Theor. Comput. Sci., 288(1):21–43, 2002.

[13] Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. J. Artif. Intell. Res., 70:245–317, 2021.

[14] D. Campos and J. Bernardes. Cardiotocography. UCI Machine Learning Repository, 2010.

[15] Robert Cattral and Franz Oppacher. Poker Hand. UCI Machine Learning Repository, 2006.

[16] Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh K. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. ACM Trans. Algorithms, 7(2):15:1–15:22, 2011.

[17] Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Marco Molinaro. On greedy algorithms for decision trees. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, Algorithms and Computation - 21st International Symposium, ISAAC, 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part II, volume 6507 of Lecture Notes in Computer Science, pages 206–217. Springer, 2010.

[18] Ferdinando Cicalese, Eduardo Sany Laber, and Aline Medeiros Saettler. Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, volume 32 of JMLR Workshop and Conference Proceedings, pages 414–422. JMLR.org, 2014.

[19] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS'05, page 235?242, Cambridge, MA, USA, 2005. MIT Press.

[20] R.A. Fisher. Iris. UCI Machine Learning Repository, 1988.

[21] Rafael García Leiva, Antonio Fernández Anta, Vincenzo Mancuso, and Paolo Casari. A novel hyperparameter-free approach to decision tree construction that avoids overfitting by design. IEEE Access, 7:99978–99987, 2019.

[22] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada, pages 766–774. Curran Associates, Inc., 2010.

[23] Andrew Guillory and Jeff A. Bilmes. Average-case active learning with costs. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, Algorithmic Learning Theory, 20th International Conference, (ALT) 2009, Porto, Portugal, October 3-5, 2009. Proceedings, volume 5809 of Lecture Notes in Computer Science, pages 141–155. Springer, 2009.

[24] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive (tsp) problems. CoRR, abs/1003.0722, 2010.

[25] Te Sun Han and Kingo Kobayashi. Mathematics of information and coding, volume 203. American Mathematical Soc., 2002.

[26] Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. Inf. Comput., 126(2):114–122, 1996.

[27] Clara Higuera, Katheleen Gardiner, and Krzysztof Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. PloS one, 10:e0129126, 06 2015.

[28] T. C. Hu and A. C. Tucker. Optimal computer search trees and variable-length alphabetical codes. SIAM Journal on Applied Mathematics, 21(4):514–532, December 1971.

[29] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decis. Support Syst., 51(1):141–154, 2011.

[30] Sangheum Hwang, Hyeon Gyu Yeo, and Jung-Sik Hong. A new splitting criterion for better interpretable trees. IEEE Access, 8:62762–62774, 2020.

[31] Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and João Marques-Silva. A sat-based approach to learn explainable decision sets. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, Automated Reasoning - 9th International Joint Conference, (IJCAR) 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, volume 10900 of Lecture Notes in Computer Science, pages 627–645. Springer, 2018.

[32] Su Jia, Fatemeh Navidi, R Ravi, et al. Optimal decision tree with noisy outcomes. Advances in neural information processing systems, 32, 2019.

[33] Colonna. Juan, Eduardo Nakamura, Marco Cristo, and Marcelo Gordo. Anuran Calls (MFCCs). UCI Machine Learning Repository, 2017.

[34] Murat Koklu and Ilker Ali Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174:105507, 2020.

[35] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an optimal split tree problem. In Proceedings of the 6th International Workshop on Algorithms and Data Structures, WADS '99, pages 157–168, London, UK, 1999. Springer-Verlag.

[36] Eduardo Sany Laber and Loana Tito Nogueira. On the hardness of the minimum height decision tree problem. Discret. Appl. Math., 144(1-2):209–212, 2004.

[37] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 1675–1684. ACM, 2016.

[38] Letter Recognition. UCI Machine Learning Repository, 1990.

[39] Ray Li, Percy Liang, and Stephen Mussmann. A tight analysis of greedy yields subexponential time approximation for uniform decision tree. In Shuchi Chawla, editor, Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, pages 102–121. SIAM, 2020.

[40] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo I. Seltzer. Generalized and scalable optimal sparse decision trees. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 6150–6160. PMLR, 2020.

[41] Zachary C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue, 16(3):31–57, jun 2018.

[42] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. Monthly Notices of the Royal Astronomical Society, 459(1):1104–1123, 04 2016.

[43] Michael Mitzenmacher and Eli Upfal. Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press, 2017.

[44] Fatemeh Navidi, Prabhanjan Kambadur, and Viswanath Nagarajan. Adaptive submodular ranking and routing. Operations Research, 68(3):856–877, 2020.

[45] Hooda. Nishtha. Audit Data. UCI Machine Learning Repository, 2018.

[46] Rok Piltaver, Mitja Lustrek, Matjaz Gams, and Sanda Martincic-Ipsic. What makes classification trees comprehensible? Expert Syst. Appl., 62:333–346, 2016.

[47] Hugo Manuel Proença and Matthijs van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. Information Scieces, 512:1372–1393, 2020.

[48] J. Ross Quinlan. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[49] Ananth R. Room occupancy estimation data set. Kaggle.

[50] Adel Rajab. Burst Header Packet (BHP) flooding attack on Optical Burst Switching (OBS) Network. UCI Machine Learning Repository, 2017.

[51] Oliver Roesler. EEG Eye State. UCI Machine Learning Repository, 2013.

[52] C. Okan Sakar, S. Polat, Mete Katircioglu, and Yomi Kastro. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. Neural Computing and Applications, 31, 10 2019.

[53] C. De Stefano, M. Maniaci, F. Fontanella, and A. Scotto di Freca. Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case. Engineering Applications of Artificial Intelligence, 72:99–110, 2018.

[54] Sebastian Tomczak. Polish companies bankruptcy data. UCI Machine Learning Repository, 2016.

[55] Joaguin Vanschoren. collins dataset. OpenML.

[56] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In The Thirty-Third AAAI Conference on Artificial Intelligence, (AAAI), 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 1625–1632. AAAI Press, 2019.

[57] I-Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2, Part 1):2473–2480, 2009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] We have not identified a clear limitation of our work. That said, we will be happy to discuss in a revised version some potential limitation that the referees point to us.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] All the proofs can be found in the supplementary material
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplementary material we include the url for our anonymous repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] It is explained in the experimental section.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In the supplementary material we present boxplots for our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We give the specification of the PC employed in our experiments in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use public datasets, we are citing all of them
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Multiway trees and optimal explanation sizes

In this section we prove a characterization of optimal explanation sizes via multiway trees. First recall that the only difference between these and threshold decision trees is that in multiway trees each test corresponds to an attribute a, and the answer to the test is the value $a(o)$ that the desired object o has on attribute a; each internal node ν is associated with such a test τ_ν, and the edges from a node to its children are associated to the possible outcomes of τ_ν. The worst-case/average depth and explanation size of a multiway tree are defined exactly as for threshold decision trees, see Section 3.

We start by proving this equivalence for the worst-case metrics.

Lemma 2 (Multiway trees and expl. size, worst-case)**.** *Consider an instance of the classification problem, and let M^* be a multiway tree of minimal worst-case depth for this instance. Then*

$$\mathrm{expl}^*_{wc} = \mathrm{expl}_{wc}(M^*) = \mathrm{depth}_{wc}(M^*).$$

Proof. Some quick notation: Given a multiway tree M and one of its nodes ν, let a_ν be the attribute that is tested on this node. Also let $M_{\nu,i}$ be the subtree of M rooted at the child of ν reached by objects whose value for attribute a_ν is i. We use $\mathrm{int}(M)$ to denote the set of internal nodes of a tree M.

We first show that $\mathrm{expl}^*_{wc} \leq \mathrm{expl}_{wc}(M^*) \leq \mathrm{depth}_{wc}(M^*)$. For that, we convert M^* into a threshold decision tree that has explanation size at most $\mathrm{depth}_{wc}(M^*)$ by just simulating the multiway tests using multiple threshold tests in the natural way (see Fig.3 for a pictorial example of the following construction). More formally, let T_{a_ν} be binary search tree over the values of attribute a_ν, that is, it is a threshold decision tree where there is a one-to-one correspondence between its leaves and the values of the attribute a_ν (i.e. every object o reaches the unique leaf of T_{a_ν} corresponding to the value $a_\nu(o)$). Then let T be the threshold decision tree obtained from M^* by (starting from the root and proceeding downwards) replacing each internal node ν of M^* with T_{a_ν} and identifying the leaf of T_{a_ν} corresponding to value $a_\nu(\cdot) = i$ with the root of $M^*_{\nu,i}$.

Consider a leaf ℓ_{M^*} in M^*. Notice that the path P_{M^*} from the root of M^* to ℓ_{M^*} induces a path P_T in the tree T that goes from its root to one of its leaves ℓ_T (i.e., whenever the path P_{M^*} goes from a node ν to a subtree $M^*_{\nu,i}$, the path P_T goes from the root of the binary search tree T_{a_ν} to its leaf corresponding to value $a_\nu(\cdot) = i$). We can see that the set of attributes tested in the paths P_{M^*} and P_T is exactly the same, and thus the leaves ℓ_{M^*} and ℓ_T have the same explanation size. Looking at the largest explanation size of the leaves in these trees, we get that $\mathrm{expl}_{wc}(T) = \mathrm{expl}_{wc}(M^*)$. This gives

$$\mathrm{expl}^*_{wc} \;\leq\; \mathrm{expl}_{wc}(T) \;=\; \mathrm{expl}_{wc}(M^*) \;\leq\; \mathrm{depth}_{wc}(M^*) \tag{1}$$

as desired.

Now we show that $\mathrm{depth}_{wc}(M^*) \leq \mathrm{expl}^*_{wc}$, which together with (1) above gives the desired equalities in the lemma. Let T be a threshold decision tree achieving the optimal explanation size expl^*_{wc}. We will show how to build a multiway decision tree M such that $\mathrm{depth}_{wc}(M) = \mathrm{expl}_{wc}(T) = \mathrm{expl}^*_{wc}$ (refer to Fig. 4 for an example of the construction).

We first recursively turn each test of T into a multiway test on the same attribute in the natural way. More precisely, this can be done in a recursive way: If T is a leaf, then define the multiway tree $\tilde{M} = T$. Otherwise, let (a, t) be the test at root of T, and let $T_<$ and $T_\geq$ be the subtrees of T rooted at the children of the root and associated with the two results of the root-test, i.e., $T_<$ (resp. $T_\geq$) is the subtree reached by the objects o such that $a(o) < t$ (resp. $a(o) \geq t$). Then $\tilde{M}$ is obtained by putting the multiway test on attribute a on its root r, and for each value i of the attribute a, if $i < t$ (resp. $i \geq t$) the i-th child of the root r is the subtree obtained by recursively applying this construction to $T_<$ (resp. $T_\geq$).

Since this construction simply converts each threshold test (a, t) in T into a multiway test on a, it is easy to see that

$$\mathrm{expl}_{wc}(\tilde{M}) = \mathrm{expl}_{wc}(T) = \mathrm{expl}^*_{wc}. \tag{2}$$

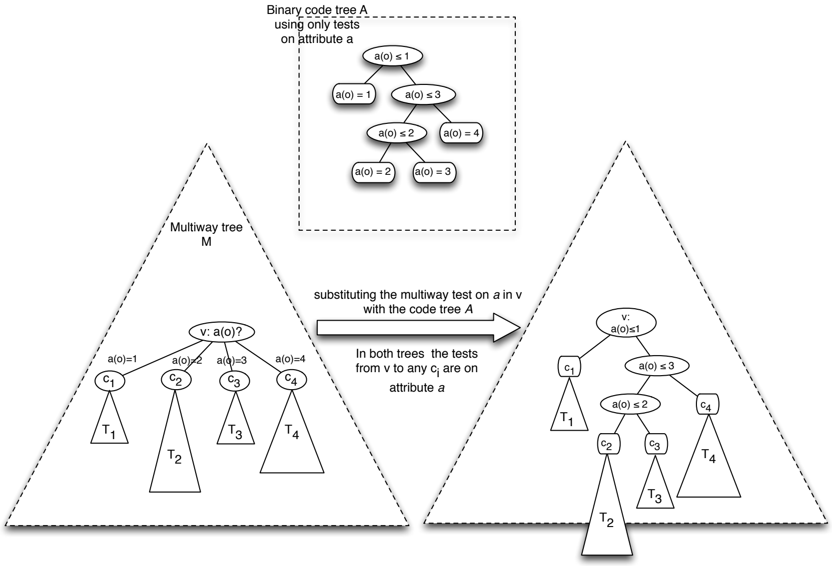

Figure 3: From multiway tree to threshold tree

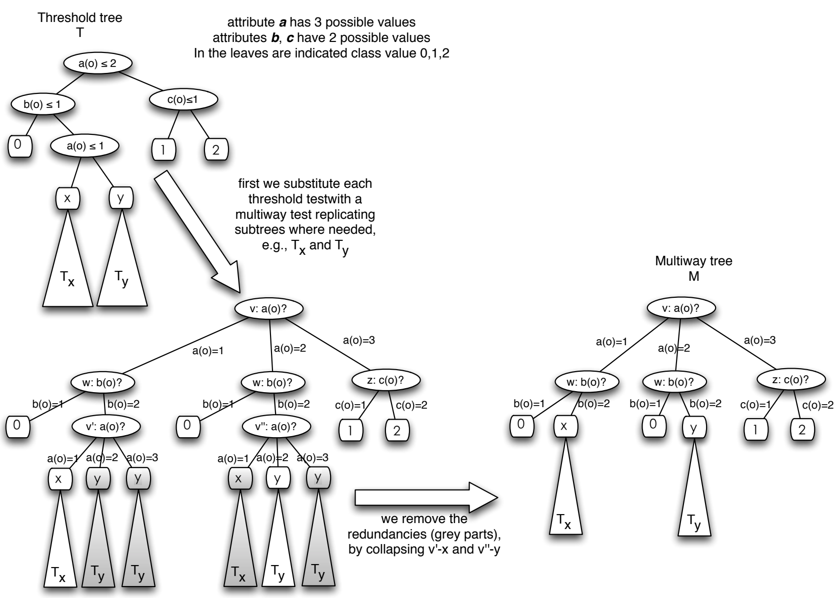

Figure 4: From threshold tree to multiway tree

However, the multiway tree $\tilde{M}$ may test the same attribute multiple times on a root-to-leaf path, because T may do so, and so we can have $\mathrm{depth}_{wc}(\tilde{M}) > \mathrm{expl}_{wc}(\tilde{M})$. But in $\tilde{M}$ these multiple tests are redundant, because a single multiway test reveals complete information about the attribute.

So we now remove these redundancies to obtain a multiway tree M with $\mathrm{depth}_{wc}(M) = \mathrm{expl}_{wc}(\tilde{M})$, as follows: If there is a node $\nu \in \mathrm{int}(\tilde{M})$ and a descendant ν' in one of the child-subtree $T_{\nu,i}$ that tests the same attribute as ν (i.e., $a_{\nu'} = a_{\nu}$), we replace the subtree $T_{\nu'}$ with the subtree $T_{\nu',i}$. Notice that since $\nu' \in T_{\nu,i}$, any object reaching node ν' has $a(\cdot) = i$, and thus would move next to the subtree $T_{\nu',i}$; our replacement operation just bypassed the node ν', and so we can see that the multiway tree obtained remains equivalent to $\tilde{M}$ for the classification problem. Repeat this replacement operation until no root-to-leaf path contains two nodes that test on the same attribute. Let M be the multiway tree obtained at the end.

Since no attribute is repeatedly tested in such paths, for every leaf $\ell \in \text{leaf}(M)$ we have depth$(\ell) =$ expl(ℓ), and hence

$$\text{depth}_{wc}(M) = \text{expl}_{wc}(M) \leq \text{expl}_{wc}(\tilde{M}),$$

the inequality following because M was obtained by deleting nodes of $\tilde{M}$. Together with inequality (2) and the fact that M^* is depth-optimal, this gives

$$\text{depth}_{wc}(M^*) \leq \text{depth}_{wc}(M) \leq \text{expl}^*_{wc},$$

as desired. This concludes the proof of the lemma. □

The same argument also shows the following analogous result about the average depth and explanation size.

Lemma 3 (Multiway trees and expl. size, average)**.** *Consider an instance of the classification problem, and let M^* be a multiway tree of minimal average depth for this instance. Then*

$$\text{expl}^*_{avg} = \text{expl}_{avg}(M^*) = \text{depth}_{avg}(M^*).$$

B Proof of Theorem 1

We prove the guarantees for the worst-case and average metrics (Items 1 and 2 of the theorem) separately. As before, given a tree T and one of its nodes ν, we use T_ν to denote the subtree rooted at ν. We also use $n(\nu)$ to denote the number of objects of the instance that reach node ν (and hence some leaf in the subtree rooted at ν); sometimes we also use a subscript $n_T(\nu)$ to make it clearer which tree we are talking about in order to avoid any confusion.

B.1 Proof of Item 1: Worst-case metrics

Recall that in this item we want a threshold tree T satisfying:

$$\text{expl}_{wc}(T) \;=\; \text{expl}^*_{wc} \tag{3}$$

$$\text{depth}_{wc}(T) \;\leq\; 2\,\text{depth}^*_{wc} + \log n. \tag{4}$$

Using Lemma 2, let M^* be an optimal multiway tree with respect to both worst-case explanation size and depth, and hence satisfying $\text{depth}_{wc}(M^*) = \text{expl}_{wc}(M^*) = \text{expl}^*_{wc}$. We show how to transform M^* into a binary tree T with $\text{expl}_{wc}(T) = \text{expl}_{wc}(M^*)$ and $\text{depth}_{wc}(T) \leq 2\,\text{depth}_{wc}(M^*) + \log n$ (which then will give (3)-(4)). Consider an internal node ν of M^*, which tests an attribute a and has as children $ch_1, ch_2, \ldots$. Let $d_i := \lceil \log \frac{n(\nu)}{n(ch_i)} \rceil + 1$ and notice that

$$\sum_i 2^{-(\lceil \log \frac{n(\nu)}{n(ch_i)} \rceil + 1)} \leq \frac{1}{2} \sum_i 2^{-\log \frac{n(\nu)}{n(ch_i)}} = \frac{1}{2}, \tag{5}$$

the last equation following because $\sum_i n(ch_i) = n(\nu)$. Then applying the result about the existence of alphabetic codes (Lemma 1) with the attribute a and considering all objects that reach the subtree $M^*_{ch_i}$ as a single object, we can get a threshold decision tree A with one leaf for each ch_i such that: 1) The leaf corresponding to ch_i is at height (in A) d_i, and; 2) Every object that in M^* reaches node ch_i, in A reaches the leaf corresponding to ch_i. It will be convenient to call A a <u>code tree</u>. Then, we proceed like in the proof of the first part of Lemma 2 (see also Fig. 3): replace the node ν of M^* by the tree A (with the root of the subtree $M^*_{ch_i}$ identified with the leaf of A corresponding to ch_i). Perform this replacement for every node of M^*, and let T be the resulting threshold tree.

We refer to the nodes of T that are the roots of the code trees A used in the process as the <u>original nodes</u>. The motivation for this terminology is that these nodes correspond to the nodes in the original tree M^*. Observe that if v is an <u>original node</u> in T and u its corresponding node in M^*, then $n_T(v) = n_{M^*}(u)$.

We first claim that $\text{expl}_{wc}(T) = \text{expl}_{wc}(M^*)$ (and hence, by definition of M^*, also equals depth^*_{wc}). Let ℓ_T be a leaf of T and ℓ_{M^*} be the corresponding leaf of M^*. By construction, the set of attributes tested on the path reaching ℓ_T is the same as the set of attributes tested on the path reaching ℓ_{M^*}, since the tests associated to the nodes ν on the latter path have been expanded in T to a path in the code tree used to replace ν and such a code tree uses only tests on the same attribute tested at ν. Therefore, we have that the explanation size of ℓ_T is the same as that of ℓ_{M^*}. Considering the leaves with largest explanation sizes in these trees, we get $\mathrm{expl}_{wc}(T) = \mathrm{expl}_{wc}(M^*) = \mathrm{expl}^*_{wc}$ as we wanted. This gives the desired equality (3).

Now we claim that $\mathrm{depth}_{wc}(T) \le 2\,\mathrm{depth}_{wc}(M^*) + \log n$. For that, consider any root-to-leaf path P in T, and let $\nu_1, \ldots, \nu_p$ be the original nodes in this path. Since both the root and leaves of T are original nodes, we see that ν_1 and ν_p are the first and last nodes of the path P, respectively. Letting $d(u, v)$ denote the distance (in number of edges) between two nodes u, v in T, we see that the length (number of internal nodes, or equivalently, edges) of P can be written as

$$|P| = \sum_{i=1}^{p-1} d(\nu_i, \nu_{i+1}). \tag{6}$$

Moreover, letting A be the alphabetic code tree used in the construction of T that is rooted at ν_i, we have that the subpath between the original nodes ν_i and ν_{i+1} coincides with the path of A between its root and its leaf corresponding to ν_{i+1}. Then, by the construction of A we have

$$d(\nu_i, \nu_{i+1}) \le \left\lceil \log \frac{n(\nu_i)}{n(\nu_{i+1})} \right\rceil + 1.$$

Replacing this bound in (6) gives

$$\begin{aligned} |P| \;&\le\; \sum_{i=1}^{p-1} \left(\left\lceil \log \frac{n(\nu_i)}{n(\nu_{i+1})} \right\rceil + 1 \right) && (7) \\ &\le\; 2(p-1) + \sum_{i=1}^{p-1} \log \frac{n(\nu_i)}{n(\nu_{i+1})} \;\le\; 2(p-1) + \log n \\ &\le\; 2\,\mathrm{depth}_{wc}(M^*) + \log n, && (8) \end{aligned}$$

where the last inequality follows because all of the original nodes $\nu_1, \ldots, \nu_{p-1}$ are (or correspond to) internal nodes in a root-to-leaf path in M^*, and so $\mathrm{depth}_{wc}(M^*) \ge p - 1$. Since the bound (8) holds for every root-to-leaf path P of T, we obtain that $\mathrm{depth}_{wc}(T) \le 2\,\mathrm{depth}_{wc}(M^*) + \log n$, proving the claim. Since by definition of M^* we have $\mathrm{depth}_{wc}(M^*) = \mathrm{expl}^*_{wc}$, which is at most depth^*_{wc}, this proves the desired inequality (4).

This concludes the proof of Item 1 of Theorem 1.

B.2 Proof of Item 2: Average metrics

We now prove Item 2 of Theorem 1, regarding the average explanation size and depth, instead of the worst-case ones, namely we want a threshold tree T satisfying:

$$\mathrm{expl}_{avg}(T) \;=\; \mathrm{expl}^*_{avg} \tag{9}$$

$$\mathrm{depth}_{avg}(T) \;\le\; 2\,\mathrm{depth}^*_{avg} + W \log n, \tag{10}$$

where $W = \sum_o w(o)$ denote the total weight of all objects. The proof is almost identical as that of the previous section, the only difference is that the weights of the items will now be taken into account when defining the d_i's in the code trees, and we use entropy-based calculations to argue about the average depth of the constructed tree.

Given a tree T and one of its nodes ν, let $w(\nu)$ denotes the total weights of the objects reaching node ν (we add the subscript $w_T(\nu)$ when we want to emphasize which tree we are referring to).

Let M^* be the multiway tree given by Lemma 3, which satisfies the guarantees $\mathrm{depth}_{avg}(M^*) = \mathrm{expl}_{avg}(M^*) = \mathrm{expl}^*_{avg}$ for the average metrics. We first note that one can compute the average cost of a tree by summing up the weights of its internal nodes, for example

$$\mathrm{depth}_{avg}(M^*) = \sum_{\nu \in \mathrm{int}(M^*)} w(\nu), \tag{11}$$

where again we use int$(\cdot)$ to denote the set of internal nodes of a tree.

We now show how to transform M^* into a threshold decision tree T of small average explanation size and depth. For an internal node ν of M^*, which tests an attribute a_ν and has as children the nodes $\nu_1, \nu_2, \ldots$, let p^ν be the probability distribution $\left(\frac{w(\nu_1)}{w(\nu)}, \frac{w(\nu_2)}{w(\nu)}, \ldots,\right)$ induced by the partition at ν. Then consider the code tree T_ν given by Lemma 1 with the attribute a and considering all objects that reach the subtree $M^*_{\nu_i}$ as a single object, but such that the leaf in T_ν corresponding to ν_i is now at depth (in T_ν) $d_i := \lceil \log \frac{w(\nu)}{w(\nu_i)} \rceil + 1$ (as in (5), we can see that such d_i's satisfy the requirement of the lemma). Then replace the node ν of M^* by the tree T_ν (with the root of the subtree $M^*_{\nu_i}$ identified with the leaf of T_ν corresponding to ν_i). Perform this replacement for every node of M^*, and let T be the resulting threshold tree.

Again we refer to the nodes of T that are the roots of the code trees used in the process as the original nodes, and observe that if v is an original node in T and u its corresponding node in M^*, then $w_T(v) = w_{M^*}(u)$.

Just as before, the set of attributes tested on the path in T from its root to one of its leaves ℓ_T is the same as the set of attributes tested on the path in M^* from its root to the leaf ℓ_{M^*} that corresponds to ℓ_T. This implies that $\text{expl}_{avg}(T) = \text{expl}_{avg}(M^*)$, which by definition of M^* also equals expl^*_{avg}. This proves inequality (9).

To prove inequality (10), let leaf$(M^*) = \{\ell_1, \ldots, \ell_m\}$ be the set of leaves of the multiway tree M^*. Let $p^L := (\frac{w(\ell_1)}{W}, \ldots, \frac{w(\ell_m)}{W})$ be the probability distribution induced on these leaves. Also, let $p(\nu) := \frac{w(\nu)}{W}$ be the sum of the probabilities of the leaves in the subtree rooted at ν. Given a distribution p, we use $\text{H}(p) := \sum_i p_i \log \frac{1}{p_i}$ to denote its Shannon entropy. The following fact records a well-known property that allows to compute the Shannon entropy of the leaf distribution as the weighted average of the entropy of the node distributions (see, e.g., [25, Chapter 3]).

Fact 1. *It holds that* $\text{H}(p^L) = \displaystyle\sum_{\nu \in int(M^*)} p(\nu) \cdot \text{H}(p^\nu)$.

Now, we can compute the average depth of the tree T as follows: Since in T, each node ν of M^* is replaced by the code tree T_ν, the corresponding contribution to the average depth of T is given by the average depth of T_ν, namely

$$\begin{aligned} \text{depth}_{avg}(T_\nu) &= \sum_i w(\nu_i) \cdot d_i \\ &= \sum_i w(\nu_i) \left(\left\lceil \log \frac{w(\nu)}{w(\nu_i)} \right\rceil + 1 \right) \le \sum_i w(\nu_i) \log \frac{w(\nu)}{w(\nu_i)} + 2w(\nu). \quad (12) \end{aligned}$$

We can use this in the computation of the average depth of the tree T, decomposing it into the contributions of the code trees that have been used to replace the nodes of M^*:

$$\begin{aligned} \text{depth}_{avg}(T) &= \sum_{\nu \in \text{int}(T)} w(\nu) = \sum_{\nu \in \text{int}(M^*)} \sum_{u \in \text{int}(T_\nu)} w(u) & (13) \\ &= \sum_{\nu \in \text{int}(M^*)} \text{depth}_{avg}(T_\nu) \\ &\le \sum_{\nu \in \text{int}(M^*)} w(\nu) \sum_i \frac{w(\nu_i)}{w(\nu)} \log \frac{w(\nu)}{w(\nu_i)} \\ &\quad + 2 \sum_{\nu \in \text{int}(M^*)} w(\nu) & (14) \\ &= W \sum_{\nu \in \text{int}(M^*)} p(\nu) \cdot \text{H}(p^\nu) + 2\, \text{depth}_{avg}(M^*) & (15) \\ &= W \cdot \text{H}(p^L) + 2\, \text{depth}_{avg}(M^*) & (16) \\ &\le W \log |\text{leaf}(M^*)| + 2\, \text{depth}_{avg}(M^*) & (17) \\ &\le W \log n + 2\, \text{depth}^*_{avg}, \end{aligned}$$

where the first line follows from (11); inequality (14) follows from using (12) to upper bound $\mathrm{depth}_{avg}(T_\nu)$ and then multiplying and dividing the first summation by $w(\nu)$; equation (15) follows again from (11); (16) follows from Fact 1; finally, inequality (17) follows from the standard entropy rank bound (i.e., the entropy of a distribution over m objects is at most $\log m$). This proves inequality (10), and concludes the proof of Item 2 of Theorem 1.

C Proof of Theorem 2

Construction of the hard instance. We will construct a deterministic instance for the lower bound as follows. There are n objects, and their classes are from the set $C = \{0, 1, \ldots, c-1\}$ and "alternate": object 1 has class 0, object 2 class 1, object 3 class 2, ..., object c class $c-1$; then this repeats, object $c+1$ has class 0, etc. We consider throughout that n is at least $cst \cdot c$ for a sufficiently large constant cst.

The first attribute in the instance, named A, has n values and gives the identity of the objects (i.e., object i has A-value equal to i).

The definition of the other attributes is more intricate and we first give their intuition and motivation. First, there is a parameter $t \in (\frac{2c}{n}, \frac{1}{2})$ (these bounds are important so that we can ignore the floor in $\lfloor t\frac{n}{c} \rfloor$). There are several binary attributes each of which discriminates a specific $(1-t)$-fraction of the objects of one of the classes $\neq 0$ from all the other objects (a more detailed description is given below). The idea is that the top levels of a decision tree of optimal height (which will only use tests on binary attributes) will have the following effect:

1. All of objects of class 0 will traverse the leftmost path
2. Only $t\frac{n}{c}$ of the objects of each class $\neq 0$ will follow this left path
3. All the other $(1-t)\frac{n}{c}$ objects of each class $\neq 0$ are classified correctly by the top levels of the tree.

Thus, the top of the tree "peels off" (namely, it completes the classification of) a $(1-t)$ fraction of the objects of each class $\neq 0$, and what is left to be "solved" is on the leftmost path. Then, starting from the last node on the leftmost path of these top levels the tree again will use a similar subtree to peel off another $(1-t)$ fraction of the objects of each class $\neq 0$, etc. The key to the lower bound is that for any decision tree not employing all these tests on the distinct binary attributes (in the attempt of reducing the number of the tests on binary attributes, and equivalently, the explanation size), the objects (of class $\neq 0$) that such not-performed tests would have correctly classified, will end up on the leftmost path together with the object of class 0. But then tests on the A attribute will need to be used to separate these objects and classify them correctly. Crucially, the objects separated by one of these binary attribute tests will be "uniformly spread out" over $[n]$, so they will "alternate" with the objects of class 0; this means that you need a lot of threshold tests on the attribute A to separate these remaining 0-class and non-0-class objects, paying a lot in terms height.

To make this more precise, let us define a round of a decision tree as the sequence of levels that are meant to achieve the result described in the three-item list above. The sets of objects $S_i \subseteq [n]$ that will be peeled off at round i need to have the following "spread out" property guaranteed by the next lemma, whose proof is deferred to Appendix C.1).

Lemma 4. *There are sets of objects $S_1, \ldots, S_w$ (with $w = \frac{\log(n/c)}{\log(1/t)}$) such that:*

- *These sets partition the set of elements of class $\neq 0$*
- *(Spread out) For each i, class $\chi \neq 0$, and interval $U \in [n]$, the set $S_i \cap U$ has at least*

$$\frac{1}{2}\frac{|U|}{n}t^{i-1}(1-t)\frac{n}{c} - 13\log(n/c)$$

objects of class χ.

Given these "peeling-off" sets, we can finally conclude the definition of the remaining attributes in the instance: in addition to attribute A, there is a binary attribute (i, j) for each $i \in [\log(n/c)/\log(1/t)]$ and $j \in [\log c]$ such that the test (i, j)

- sends to the right all the objects in S_i whose class have j-th bit equal to 1
- sends to the left all other objects (in particular all those outside of S_i).

This concludes the description of the instance.

Analysis. Given an integer x, we use $bits(x)$ to denote the binary expansion of x (as a vector). Given a (partial) decision tree, when all objects that reach a given leaf have the same class we say that the tree classifies them correctly (since this can be naturally obtained by assigning their class to the leaf).

We start by showing that this instance has a tree of "short" height.

Lemma 5. $\mathrm{depth}^*_{wc} \le \frac{\log(n/c)\log c}{\log(1/t)}$.

Proof. We construct such a short tree that satisfies the bound, hence a fortiori it must also hold for the optimum. Again, we use the term round to refer to a group of consecutive levels, where for each level there is a specific test which is performed in all nodes of that level. (Round 1) Build a complete binary tree of depth $\log c$, where at each node on level j the test $(1, j)$ is performed. We claim that these tests peel off the set S_1, and classify all of these objects correctly. To see this, observe that these tests give a tree with c leaves, each associated with a vector $y \in \{0,1\}^{\log c}$ corresponding to the outcome of these tests (where 0 means "go left" and 1 means "go right"). The main observation is that every object of S_1 of class $\chi \in \{1, \ldots, (\log c) - 1\}$ will traverse to the leaf corresponding to vector $y = bits(\chi)$. In addition, all objects outside of S_1 will end up in the left-most leaf (i.e. the one with vector $y = (0, \ldots, 0)$. With this we can conclude that:

1. All leaves except the left-most one have only objects with the same class (so we do not need to continue splitting these leaves). These leaves contain all elements in S_1 (recall that S_1 does not have any element of class 0).
2. The leftmost leaf has exactly all elements outside of S_1.

(Round 2) We need to further split this leftmost leaf. For that, build a new complete binary tree of depth $\log c$, where at each node on level j test $(2, j)$ is performed. Again by the same argument we get another tree with c leaves, where all objects in S_2 will be peeled out and only the elements in $[n] - (S_1 \cup S_2)$ are remaining on the leftmost leaf.

Keep repeating these rounds—each round $i = 1, \ldots, w$, corresponding to executing tests $\{(i,j)\}_{j=1,\ldots,\log c}$—until you finish peeling off all sets S_i. At this point the leftmost path has only objects $[n] - (S_1 \cup S_2 \cup \ldots \cup S_w)$, which are all objects of class 0, so they are correctly classified by this leaf. Also, by construction all the remaining objects $S_1 \cup S_2 \cup \ldots \cup S_w$ are also classified correctly in the rest of the tree. So we have obtained a valid tree of total height $w \cdot \log c = \frac{\log(n/c)\log c}{\log(1/t)}$, as desired. □

Now we give a lower bound on the depth of any tree that solves the problem.

Lemma 6. *Consider a tree that solves the instance described above. Then there is a root-to-leaf path P such that*

$$K_A + K_B \frac{\log(1/t)}{\log c} \ge \frac{1}{2} \log \frac{n}{c}. \tag{18}$$

where K_A and K_B are the number of A tests and binary tests (respectively) done on this path.

Proof. We construct the path P as follows: Start at the root of the tree; if the current node ν corresponds to a test on a binary attribute, go to the left; if it corresponds to a threshold test on the attribute A, go to the side where most examples would go if we ignore the tests on the **binary attributes** performed before node ν (but we do take into account the previous tests on attribute A).

We claim that unless (18) holds, at least one object of class 0 and one object of another class $\neq 0$ would reach the end of the path P, contradicting that the tree correctly classifies each object.

To see this claim, first consider executing all the binary tests of P and let us see which objects survive (i.e., the result of each test on P make them continue to the next node on P). First, all objects of class 0 survive these tests (since P only turns left on binary tests). Moreover, there are $\frac{\log(n/c)\log c}{\log 1/t}$ binary tests (i,j) and P is only doing K_B of them. Thus,

$$\frac{\log(n/c)\log c}{\log(1/t)} - K_B$$

tests (i,j) are not done. Since there are $\log c$ of these j's, an averaging argument shows that there is one j where at least

$$\frac{1}{\log c}\left[\frac{\log(n/c)\log c}{\log(1/t)} - K_B\right] = \frac{\log(n/c)}{\log(1/t)} - \frac{K_B}{\log c}$$

tests $\{(i,j)\}_i$ are not done; let us denote it with j^*. Since the i's range from 1 to $\frac{\log(n/c)}{\log(1/t)}$, this implies that there is one test (i,j^*) that is not performed for some $i \le \frac{K_B}{\log c}$. Denote this test by (i^*,j^*).

Look at the objects in the set S_{i^*} with class χ where $bits(\chi) = (0,\dots,0,1,0,\dots,0)$ (with 1 in the j^*-th position). Call this set $S^{\chi}_{i^*}$. Since all binary tests other than (i^*,j^*) send the objects $S^{\chi}_{i^*}$ to the left, again we see that all of these objects survive the binary tests of P. In summary: if we just perform the binary tests of P, all $\frac{n}{c}$ objects of class 0 survive (call them S^0) as well as all objects of the set $S^{\chi}_{i^*}$.

Now we execute all the threshold tests on the attribute A that are performed along the path P. The result of these tests is to select/isolate the objects in some interval U of $[n]$. That is, the objects $(S^0 \cup S^{\chi}_{i^*}) \cap U$ survive **all** the tests in the path P.

By construction of P, each one of such tests on the attribute A at most halves the size of the interval of objects selected by the previous tests on A, hence U has size at least $\frac{n}{2^{K_A}}$. Thus, since one every k object has class 0, it is immediate to see that

$$|S^0 \cap U| \;\ge\; \frac{|U|}{k} - 2 \;\ge\; \frac{1}{2^{K_A}}\frac{n}{c} - 2, \tag{19}$$

that is, at least these many objects of class 0 survive. Moreover, by Lemma 4 the number of surviving objects of class χ is at least:

$$\begin{aligned} |S^{\chi}_{i^*} \cap U| \;&\ge\; \frac{1}{2}\frac{|U|}{n} t^{i^*-1}(1-t)\frac{n}{c} - 13\log(n/c) \\ &\ge\; \frac{1}{2}\frac{|U|}{n} \cdot t^{i^*} \cdot \frac{n}{c} - 13\log(n/c) \\ &\ge\; \frac{1}{2^{K_A+1}} \cdot t^{\frac{K_B}{\log c}} \cdot \frac{n}{c} - 13\log(n/c). \end{aligned} \tag{20}$$

By assumption that the decision tree correctly classifies each object, we cannot have both classes surviving the path P. Hence, at least one of the RHSs in (19) or (20) has to be less than 1. By observing that the RHS of (20) is not larger than the RHS of (19), it must hold that

$$\frac{1}{2^{K_A+1}} \cdot t^{\frac{K_B}{\log c}} \cdot \frac{n}{c} - 13\log(n/c) \;\le\; 1$$

so in particular (since $n \ge cst \cdot c$ with cst a large constant)

$$\begin{aligned} &\frac{1}{2^{K_A+1}} \cdot t^{\frac{K_B}{\log c}} \cdot \frac{n}{c} \\ &\qquad\le\; 14\log(n/c) \;\overset{\log}{\equiv}\; (K_A+1) + K_B \cdot \frac{\log(1/t)}{\log c} \\ &\qquad\qquad\qquad \ge \log(n/c) - \log(14\log(n/c)). \end{aligned}$$

Again using the fact that $n \ge cst \cdot c$ for a sufficiently large constant, of the this implies the cleaner bound

$$K_A + K_B \cdot \frac{\log(1/t)}{\log c} \ge \frac{1}{2}\log(n/c),$$

which concludes the proof. □

Now we are ready to prove the main theorem.

Proof of Theorem 2. Set t such that $\alpha = \frac{1}{2}\frac{\log 1/t}{\log c}$. i.e., $t = \frac{1}{c^{2\alpha}}$. Since we assumed $\alpha \in [\frac{1}{2\log c}, \frac{1}{2}(\frac{\log(n/2)}{\log c} - 1)]$ we have that $t \in [\frac{2c}{n}, \frac{1}{2}]$, as required by the above argument. Then consider the instance $\mathcal{I}$ defined above for this value of t.

Consider any binary tree T solving the instance $\mathcal{I}$ that is an $\frac{\alpha}{2}$-approx for the depth. It suffices to show that this tree has $\mathrm{expl}_{wc}(T) \ge \frac{1}{4\alpha}\log(n/c)\mathrm{expl}^*_{wc}$, which means $\mathrm{expl}_{wc}(T) \ge \frac{1}{4\alpha}\log(n/c)$ (since $\mathrm{expl}^*_{wc} = 1$ for this instance, by using only tests on attribute A).

For that, look at the path of T given by Lemma 6. Since T is an $\frac{\alpha}{2}$-approx for the height, we have in particular (using Lemma 5)

$$K_A \le \frac{\alpha}{2}\,\mathrm{depth}^*_{wc} \le \frac{1}{4}\frac{\log 1/t}{\log c}\cdot\frac{\log(n/c)\log c}{\log 1/t} = \frac{1}{4}\log(n/c).$$

But then from Lemma 6 we have

$$K_B\frac{\log 1/t}{\log c} \ge \frac{1}{2}\log\frac{n}{c} - K_A, \text{ i.e., } K_B \ge \frac{1}{4\alpha}\log\frac{n}{c}.$$

Since each binary test in K_B contributes one unit to the explanation size, this proves that the explanation size of T is at least $\frac{1}{4\alpha}\log(n/c)$ as desired. □

C.1 Proof of Lemma 4

The lemma will be a direct consequence of the following result.

Lemma 7. *Consider $[m]$ and positive scalars m_i such that $m_1 + \ldots + m_w \le m$, with $w \le m$. Then there are sets $V_1, \ldots, V_w \subseteq [m]$ such that:*

1. *The V_i's partition $[m]$*
2. *For every interval $J \subseteq [m]$ and i, we have*

$$|V_i \cap J| \ge \frac{1}{2}\frac{|J|}{m}\cdot m_i - 12\log m.$$

Proof. Without loss of generality assume that $\sum_i m_i = m$, by increasing one of the m_i's if necessary. Construct the sets V_i's randomly as follows: independently for each $j \in [m]$, randomly put j in one of the sets V_i so that $\Pr(j \in V_i) = \frac{m_i}{m}$. Then by definition the V_i's so created partition $[m]$, giving Item 1 of the lemma.

For Item 2, take any interval J. First note that

$$\mathbb{E}|V_i \cap J| = \sum_{j\in J}\Pr(j \in V_i) = \frac{|J|}{m}\cdot m_i.$$

Moreover, using the multiplicative Chernoff bound (see, e.g., [43, Chapter 4]) we have that for a fixed i,

$$\Pr\left(|V_i \cap J| \le \frac{1}{2}\mathbb{E}|V_i \cap J|\right) \le e^{-\frac{\mathbb{E}|V_i\cap J|}{8}} = \exp\left(-\frac{1}{8}\frac{|J|}{m}\cdot m_i\right).$$

Moreover, for any $\delta \in (0,1)$, by considering the cases $\frac{1}{8}\frac{|J|}{m}\cdot m_i \ge \log(1/\delta)$ and $\frac{1}{8}\frac{|J|}{m}\cdot m_i \le \log(1/\delta)$ we see that this implies

$$\Pr\left(|V_i \cap J| \le \frac{1}{2}\Big[\mathbb{E}|V_i \cap J| - 8\log(1/\delta)\Big]\right) \le \delta.$$

Taking a union bound over all $i \in [w]$ (recall $w \le m$) and over all intervals J (there are less than m^2 of them) we get that with probability strictly more than $1 - m^3\delta$ we have

$$|V_i \cap J| > \frac{1}{2}\Big[\mathbb{E}|V_i \cap J| - 8\log(1/\delta)\Big] \qquad \forall i, \forall \text{ intervals } J \subseteq [m].$$

Setting $\delta = \frac{1}{m^3}$ shows that there is a scenario where

$$|V_i \cap J| > \frac{1}{2}\frac{|J|}{m} \cdot m_i - 12\log m \qquad \forall i, \forall \text{ intervals } J \subseteq [m].$$

This proves the lemma. □

Proof of Lemma 4. Bucket the objects $[n]$ in buckets $B_1, \ldots, B_{n/c}$, each of size c (with consecutive objects), that is, we have $B_1 = \{1, 2, \ldots, c\}$, $B_2 = \{c+1, \ldots, 2c\}$, ... until bucket $B_{n/c}$ (for the sake of simplifying the notation, we assume that n/c is an integer). Set $w = \frac{\log(n/c)}{\log(1/t)}$ and apply the previous lemma with parameter $m_i = t^{i-1}(1-t)\frac{n}{c}$ (notice $\sum_i m_i \leq \frac{n}{c}$) to get a partition $V_1, V_2, \ldots, V_w$ of the set of indices of buckets $[n/c]$. Then set

$$S_i := \bigcup_{\ell \in V_i} B_\ell^{\neq 0},$$

where $B_\ell^{\neq 0}$ denotes the set of objects in bucket B_ℓ of class different from 0.

By construction we satisfy Item 1 of the lemma, namely that the S_i's partition all the objects of class $\neq 0$. For the second item of the lemma, notice that for any interval $U \subseteq [n]$ (recall that S_i^χ is the set of objects in S_i of class χ)

$$|S_i^\chi \cap U| \geq \#\text{ buckets that compose } S_i \text{ and that are fully contained in } U,$$

since each bucket has 1 objects of class χ. Let $J \subseteq [n/c]$ be the indices of the buckets fully contained in U; so the previous bound is

$$|S_i^\chi \cap U| \geq |V_i \cap J| \geq \frac{1}{2}\frac{|J|}{n/c}t^{i-1}(1-t)\frac{n}{c} - 12\log(n/c),$$

where the last inequality follows from Lemma 7. Moreover, we can see that $|J| \geq \frac{|U|}{c} - 2$. Replacing this bound in the previous displayed inequality and again using the fact $n \geq cst \cdot c$ for a sufficiently large cst gives the desired result. □

D Proof of Theorem 3

D.1 The Bounds on the average case

In this section we show the bounds

$$\text{depth}_{avg}(T) \leq O(\log n)\,\text{depth}^*_{avg}, \tag{21}$$
$$\text{expl}_{avg}(T) \leq O(\log n)\,\text{expl}^*_{avg}. \tag{22}$$

for a tree T produced by the algorithm SER-DT.

In Line 2 of SER-DT, the threshold factor $\frac{1}{2}$ is used to select an attribute that reduces the weighted pairwise misclassification. The argument we present here consider a more general case where $\frac{1}{2}$ is substituted by any $\gamma \in [\frac{1}{2}, 1)$. To simplify the argument, assume w.l.o.g. that (after the preprocessing) the weights are integers and $w_{\min} = 1$. To prove both parts of 21, it suffices to show that for the constant $\alpha := \max\{4, 2(\ln\frac{1}{\gamma})^{-1}\}$. we have

$$\text{depth}_{avg}(T) \leq \alpha \cdot \ln\big(P(O)w(O)\big) \cdot \text{depth}_{avg}(M^*), \tag{23}$$

where M^* is a multiway decision tree[4] achieving the optimal average depth, depth^*_{avg}, for the instance I. Since $\ln(P(O)w(O)) \leq \ln(n^2 W) = O(\ln(nW))$, and using the equivalence in Lemma 3, this implies

$$\begin{aligned}\text{expl}_{avg}(T) \leq \text{depth}_{avg}(T) &\leq O\big(\ln\big(nW\big)\big) \cdot \text{depth}^*_{avg} \\ &= O\big(\ln\big(nW\big)\big) \cdot \text{expl}^*_{avg}.\end{aligned}$$

[4]Recall that this is a decision tree where each test corresponds to an attribute a, and making this test splits the objects based on all possible values of the this attribute, instead of just splitting them as $a(o) < t$ and $a(o) \geq t$ for some threshold t.

To prove (23), we are going to use the following lower bound on the optimum average depth of a multiway tree.

Lemma 8. *Let* (a^*, i^*) *be the pair attribute/value maximizing* $\max_{a,i} P(O(a,i))w(O(a,i))$. *Let* M^* *be an optimal multiway tree, i.e.,* $\text{depth}_{avg}(M^*) = \text{depth}^*_{avg}$. *Then*

$$\text{depth}_{avg}(M^*) \;\geq\; \frac{1}{2}\frac{P(O)w(O)^2}{P(O)w(O) - P(O(a^*,i^*))w(O(a^*,i^*))}.$$

Proof. For a node ν of M^*, let O_ν be the set of objects associated with the leaves of the subtree rooted at ν, and let a_ν be the attribute tested at node ν. Like in (11) we compute the average cost of the optimal tree M^* by summing up the weights of its internal nodes, $\text{depth}_{avg}(M^*) = \sum_{\text{int}(M^*)} w(O_\nu)$, where again we use $\text{int}(\cdot)$ to denote the set of internal nodes of a tree.

For a set of objects $S \subseteq O$ and attribute x, define the quantity

$$\begin{aligned}\Delta_x(S) \;:=\; \min_i \Big(P(S)w(S) - P(S(x,i))w(S(x,i))\Big) \\ \;=\; P(S)w(S) - \max_i P(S(x,i))w(S(x,i)).\end{aligned}$$

Notice that by definition, the attribute $x = a^*$ is the one that has the largest value $\Delta_x(O)$, that is, $\Delta_a(O) \leq \Delta_{a^*}(O)$ for every attribute a. Hence, we have

$$\text{depth}_{avg}(M^*) \;=\; \sum_{\text{int}(T^*)} w(O_\nu) \;\geq\; \frac{\sum_{\text{int}(T^*)} w(O_\nu)\cdot\Delta_{a_\nu}(O)}{\Delta_{a^*}(O)}. \tag{24}$$

To make the numerator more "local", we would like to replace $\Delta_{a_\nu}(O)$ by $\Delta_{a_\nu}(O_\nu)$. The next lemma shows we can do this and still have a valid lower bound for $\text{depth}_{avg}(M^*)$.

Claim 1. *For every attribute* x*, the function* $\Delta_x(\cdot)$ *is monotone, namely for any sets of objects* $V \subseteq U \subseteq O$ *we have* $\Delta_x(V) \leq \Delta_x(U)$.

Proof of the claim. Let $\bar{\imath} = \text{argmin}_i(P(U)w(U) - P(U(x,i))w(U(x,i)))$ be the attribute value that yields the definition of $\Delta_x(U)$. Since $\bar{\imath}$ is a feasible value in the definition of $\Delta_x(V)$, we have $\Delta_x(V) \leq P(V)w(V) - P(V(x,\bar{\imath}))w(V(x,\bar{\imath}))$; so to show $\Delta_x(V) \leq \Delta_x(U)$ it suffices to show

$$\begin{aligned}P(V)w(V) - P(V(x,\bar{\imath}))w(V(x,\bar{\imath})) \\ \leq\; P(U)w(U) - P(U(x,\bar{\imath}))w(U(x,\bar{\imath})).\end{aligned}$$

Let $S(x,\neq i) := S \setminus S(x,i)$ be the objects in S with value different from i in the attribute x, and notice $w(S) = w(S(x,i)) + w(S(x,\neq i))$. Applying this in the last displayed inequality, we see that it is equivalent to

$$P(V(x,\bar{\imath}))w(V(x,\neq\bar{\imath})) + \underbrace{\Big[P(V) - P(V(x,\bar{\imath}))\Big]}_{\underline{\text{top-term}}}\cdot w(V) \tag{25}$$

$$\leq\; P(U(x,\bar{\imath}))w(U(x,\neq\bar{\imath})) + \underbrace{\Big[P(U) - P(U(x,\bar{\imath}))\Big]}_{\underline{\text{bottom-term}}}\cdot w(U). \tag{26}$$

Since the number of pairs $P(S)$ and the weight $w(S)$ are monotone in S and since $V \subseteq U$, it is clear that each term in (25) (except top-term) is at most the corresponding term in (26) (except bottom-term). Thus, to prove this inequality, it suffices to argue that top-term $\leq$ bottom-term. But notice that top-term is the number of pairs of objects $o \neq o' \in V$ that have different classes and where at least one of objects has value $\neq \bar{\imath}$ in attribute x; bottom-term is the same thing, but over pairs of objects $o \neq o'$ in U. Since every pair in the former also belongs to the latter, we see that top-term $\leq$ bottom-term. This then proves inequality (25)-(26) and concludes the proof of the claim. □

Applying this on (24) we obtain the lower bound

$$\mathrm{depth}_{avg}(M^*) \;\geq\; \frac{\sum_{\mathrm{int}(T^*)} w(O_\nu) \cdot \Delta_{a_\nu}(O_\nu)}{\Delta_{a^*}(O)}. \tag{27}$$

We can think of the numerator above as the cost of a decision tree for an instance $\tilde{I}$ with the same tests and object set as I but with the additional property that applying the test on an attribute a at a point where there are S objects incurs a cost equal to $\Delta_a(S)$ In general, given a decision tree $\tilde{M}$ for the instance $\tilde{I}$, its cost is defined as

$$\sum_{\mathrm{int}(\tilde{M})} w(\tilde{O}_\nu) \cdot \Delta_{\tilde{a}_\nu}(\tilde{O}_\nu),$$

where $\tilde{O}_\nu$ is the set of objects associated with the leaves of the subtree of $\tilde{M}$ rooted at ν, and $\tilde{a}_\nu$ is the attribute tested at the node ν. Letting $\mathrm{depth}^*_{avg}(\tilde{I})$ denote the minimum cost of a multiway decision tree for $\tilde{I}$ with costs given by Δ, from (27) we have

$$\mathrm{depth}_{avg}(M^*) \;\geq\; \frac{\mathrm{depth}^*_{avg}(\tilde{I})}{\Delta_{a^*}(O)}\,. \tag{28}$$

The proof of Lemma 8 is then completed by proving the following claim.

Claim 2. *We have* $\mathrm{depth}^*_{avg}(\tilde{I}) \geq \frac{P(O)w(O)^2}{2}$.

Proof of the claim. We argue by induction on the number of pairs of distinct classes $P(O)$. For the base case $P(O) = 1$, we have that in an optimal tree the attribute a tested at the root node ρ must separate the two object belonging to the only pair — note that such an attribute must exists since we assume separability of objects belonging to different classes. Hence, it must hold that $P(O(a, i^a)) = 0$, and $\Delta_a(O) = P(O)w(O)$. Therefore we have

$$\mathrm{depth}^*_{avg}(\tilde{I}) \;\geq\; w(O_\rho)\Delta_a(O) \;=\; P(O)w(O)^2,$$

as desired.

Now assume $P(O) \geq 2$ and that the claim holds for all instances with the number of pairs of distinct classes smaller than $P(O)$. Let a be the attribute tested at the root of a tree attaining $\mathrm{depth}^*_{avg}(\tilde{I})$. Also let i^a be the value at attribute a that yields $\Delta_a(O)$, namely $i^a = \mathrm{argmin}_i(P(O)w(O) - P(O(a,i))w(O(a,i)))$. Because of the optimality of the tree we can assume that $P(O(a, i^a)) < P(O)$, i.e., the test a separates at least one of the pairs.

<u>Case 1.</u> $P(O(a, i^a)) = 0$. Then as in the base case $\Delta_a(O) = P(O)w(O)$, and again we have $\mathrm{depth}^*_{avg}(\tilde{I}) = w(O) \cdot \Delta_a(O) = P(O)w(O)^2$.

<u>Case 2.</u> $1 \leq P(S(a, i^a)) < P(S)$. Then we consider the contribution of the root node and the subtree rooted at the child corresponding to the value i^a. Let $\tilde{I}_{i^a}$ be the instance corresponding to the objects in the set $O(a, i^a)$, with number of pairs $P(O(a, i^a))$. We have

$$\begin{aligned}
\mathrm{depth}^*_{avg}(\tilde{I}) &\geq w(O) \cdot \Delta_a(O) + \mathrm{depth}^*_{avg}(\tilde{I}_{i^a}) && (29)\\
&\geq w(O) \cdot \Big(P(O)w(O) - P(O(a,i^a))w(O(a,i^a))\Big) \\
&\quad + \frac{P(O(a,i^a))w(O(a,i^a))^2}{2} && (30)\\
&\geq P(O)w(O)^2 - \frac{P(O(a,i^a))w(O)^2}{2} && (31)\\
&\geq \frac{P(S)w(S)^2}{2}, && (32)
\end{aligned}$$

where (30)) follows from using the inductive hypothesis on $\mathrm{depth}^*_{avg}(\tilde{I}_{i^a})$, (31) follows from the monotonicity of the weights $w(O(a, i^a)) \leq w(O)$, and (32) follows from monotonicity of the pairs $P(O(a, i^a)) \leq P(O)$. This concludes the proof of the inductive step, and hence of the claim. □

□

Given this lower bound on the average depth of an optimal multiway tree depth^*_{avg} we now show the desired bound (23). Skipping trivialities, we can assume $n \geq 2$. We argue by induction on the number of pairs of distinct classes $P(O)$ in the instance . The base case $|P(O)| = 1$ is easily settled by the assumption that for every pair of objects from distinct classes there exists a test splitting them.

So assume by induction that (23) holds for the tree build by the algorithm on every instance I' with objects O' such that $P(O') < P(O)$. To prove that it still holds for the instance I, we consider whether the algorithm entered the If in Line 2 or not.

Case 1: The algorithm enters the If at Line 2. Let (a, t) be the split used by the algorithm at the root. For $i = 1, 2$, let T_i be the subtree of T rooted at the left and right child of the root of T. Recall that T_1 is the tree built on the instance I_1 with object set $O_1 := O(a, \leq t)$ and T_2 is the the tree recursively built on the instance I_2 with object set $O_2 := O(a, > t)$.[5] By construction we have $\texttt{wpm}(O(a, t)) \leq \gamma \cdot P(O)w(O)$, which means that

$$P(O_i)w(O_i) \;\leq\; \gamma P(O)w(O), \quad i = 1, 2. \tag{33}$$

The average depth of the tree T build by the algorithm can then be decomposed as $\text{depth}_{avg}(T) = w(O) + \text{depth}_{avg}(T_1) + \text{depth}_{avg}(T_2)$. Moreover, we also have the following subadditivity property: $\text{depth}^*_{avg}(I) \geq \text{depth}^*_{avg}(I_1) + \text{depth}^*_{avg}(I_2)$. In fact, letting M^* (resp. M^*_1, M^*_2) being a multiway tree achieving $\text{depth}^*_{avg}(I)$ (resp. $\text{depth}^*_{avg}(I_1)$, $\text{depth}^*_{avg}(I_2)$) we have

$$\begin{aligned}
\text{depth}^*_{avg}(I) = \text{depth}_{avg}(M^*) &= \sum_{o \in O} w(o) \cdot \text{depth}^{T^*}_{avg}(\ell(o)) \\
&\geq \sum_{o \in O_1} w(o) \cdot \text{depth}^{M^*}_{avg}(\ell(o)) + \sum_{o \in O_2} w(o) \cdot \text{depth}^{M^*}_{avg}(\ell(o)) \\
&\geq \sum_{o \in O_1} w(o) \cdot \text{depth}^{M^*_1}_{avg}(\ell(o)) + \sum_{o \in O_2} w(o) \cdot \text{depth}^{M^*_2}_{avg}(\ell(o)) \\
&= \text{depth}^*_{avg}(I_1) + \text{depth}^*_{avg}(I_2),
\end{aligned}$$

where we used the notation $\text{depth}^M_{avg}(\ell(o))$ to indicate the explanation size of leaf reached by object o in a tree M. Then we have

$$\begin{aligned}
\frac{\text{depth}_{avg}(T)}{\text{depth}^*_{avg}(I)} &= \frac{w(O) + \text{depth}_{avg}(T_1) + \text{depth}_{avg}(T_2)}{\text{depth}^*_{avg}(I)} \\
&\leq \frac{w(O)}{\text{depth}^*_{avg}(I)} + \frac{\text{depth}_{avg}(T_1) + \text{depth}_{avg}(T_2)}{\text{depth}^*_{avg}(I_1) + \text{depth}^*_{avg}(I_2)} \\
&\leq \frac{w(O)}{\text{depth}^*_{avg}(I)} + \max_{i=1,2} \frac{\text{depth}_{avg}(T_i)}{OPT_M(I_i)}.
\end{aligned} \tag{34}$$

But $\text{depth}^*_{avg}(I) \geq w(O)$, and by the inductive hypothesis $\frac{\text{depth}_{avg}(T_i)}{\text{depth}^*_{avg}(I_i)} \leq \alpha \ln(P(O_i)w(O_i))$, which is at most $\alpha \ln(\gamma P(O)w(O))$ by (33). Plugging these bounds in the RHS of (34) gives

$$\begin{aligned}
\frac{\text{depth}_{avg}(T)}{\text{depth}^*_{avg}(I)} &\leq 1 + \alpha \ln\big(\gamma P(O)w(O)\big) \\
&= \alpha \ln\big(P(O)w(O)\big) + \Big(1 - \alpha \ln \frac{1}{\gamma}\Big) \leq \alpha \ln(P(O)w(O)),
\end{aligned}$$

where the last inequality follows from the fact $\alpha \geq (\ln \frac{1}{\gamma})^{-1}$. This proves the inductive hypothesis in this case.

[5] *N.B.: We have noticed that sometimes we consider threshold tests of the form "$a(o) < t$?" and sometimes of the form "$a(o) \leq t$?". These tests are equivalent, by simply shifting the threshold t. Nonetheless, we will update the next version of the paper to make the notation homogeneous.*

Case 2: The algorithm does not enter the If at Line 2. In this case, the top of the tree T produced by the algorithm uses the splits $(a^*, t^* - 1)$ and (a^*, t^*), and has (up to) three subtrees T_1, T_2, T_3 as children of these nodes, where:

- T_1 is constructed recursively for the instance I_1 with object set $O_1 := O(a^*, \leq t^* - 1)$
- T_2 is constructed recursively for the instance I_2 with object set $O_2 := O(a^*, t^*)$
- T_3 is constructed recursively for the instance I_3 with object set $O_3 := O(a^*, > t^*)$.

(Note that one of the instances I_1, I_3 may be empty.) The average depth of T can be decomposed based on the contribution of the two splits $(a^*, t^* - 1)$ and (a^*, t^*) (which is at most $2w(O)$) and of the subtrees T_i's, namely

$$\mathrm{depth}_{avg}(T) \leq 2w(O) + \mathrm{depth}_{avg}(T_1) + \mathrm{depth}_{avg}(T_2) + \mathrm{depth}_{avg}(T_3).$$

Thus, as in inequality (34) we have

$$\frac{\mathrm{depth}_{avg}(T)}{\mathrm{depth}^*_{avg}(I)} \leq \frac{2w(O)}{\mathrm{depth}^*_{avg}(I)} + \max_{i=1,2,3} \frac{\mathrm{depth}_{avg}(T_i)}{\mathrm{depth}^*_{avg}(I_i)},$$

where if an instance I_i is empty (so the tree T_i does not exist) we ignore the corresponding term in the max.

By definition of t^*, for $i = 1, 3$ the instance I_i satisfies $P(O_i)w(O_i) \leq \gamma P(O)w(O)$, and so applying the induction hypothesis on the instances I_i we get

$$\frac{\mathrm{depth}_{avg}(T)}{\mathrm{depth}^*_{avg}(I)} \leq \frac{2w(O)}{\mathrm{depth}^*_{avg}(I)} + \max\left\{\alpha \ln\Big(\gamma P(O)w(O)\Big)\,,\ \alpha \ln\Big(P(O_2)w(O_2)\Big)\right\}.$$

If $P(O_2)w(O_2) \leq \gamma P(O)w(O)$, the max becomes simply $\alpha \ln(\gamma P(O)w(O))$ and again using $\mathrm{depth}^*_{avg}(I) \geq w(O)$ we get

$$\frac{\mathrm{depth}_{avg}(T)}{\mathrm{depth}^*_{avg}(I)} \leq 2 + \alpha \ln\big(\gamma P(O)w(O)\big) \leq \alpha \ln(P(O)w(O)),$$

where the last inequality uses the fact $\alpha \geq 2(\ln \frac{1}{\gamma})^{-1}$; so the inductive hypothesis holds in this case.

Now consider the case where $P(O_2)w(O_2) > \gamma P(O)w(O)$, and hence the max becomes $\alpha \ln(P(O_2)w(O_2))$. In this case, we use the lower bound on $\mathrm{depth}^*_{avg}(I)$ from Lemma 8 to obtain (let a^* and i^* be defined as in the lemma)

$$\begin{aligned}\frac{\mathrm{depth}_{avg}(T)}{\mathrm{depth}^*_{avg}(I)} &\leq \frac{4\big(P(O)w(O) - P(O(a^*, i^*))w(O(a^*, i^*))\big)}{P(O)w(O)} \\ &\quad + \alpha \ln P(O_2)w(O_2) \\ &\leq 4\left(1 - \frac{P(O(a^*, i^*))w(O(a^*, i^*))}{P(O)w(O)}\right) \\ &\quad + \alpha \ln(P(O(a^*, i^*))w(O(a^*, i^*))) \\ &\leq 4 \ln \frac{P(O)w(O)}{P(O(a^*, i^*))w(O(a^*, i^*))} \\ &\quad + \alpha \ln(P(O(a^*, i^*))w(O(a^*, i^*))) \\ &\leq \alpha \ln\left(P(S)w(S)\right),\end{aligned}$$

where the second inequality follows the fact $P(O(a^*, i^*))w(O(a^*, i^*)) \geq P(O(a^*, t^*)w(a^*, t^*)) = P(O_2)w(O_2)$ (by optimality of i^*), the third inequality follows from the fact $1 - x \leq -\ln x$ valid for all $x > 0$ (since by convexity $\ln x \geq \ln(1) + \ln'(1)(x - 1)$), and the last inequality follows from the fact $\alpha \geq 4$.

This concludes the proof of the inductive step, and hence that inequality (23) holds and as a consequence, we have

$$\text{depth}_{avg}(T) \leq O\left(\ln\left(n\frac{W}{w_{\min}}\right)\right)\text{depth}^*_{avg},$$
$$\text{expl}_{avg}(T) \leq O\left(\ln\left(n\frac{W}{w_{\min}}\right)\right)\text{expl}^*_{avg},$$

where $W = w(O)$ is the total weight of the objects, $w_{\min} = \min_{o\in S} w(o)$ is the smallest weight of an object. Finally, by the preprocessing step (following the approach from [35]) we have that the approximation term $O\left(\ln\left(n\frac{W}{w_{\min}}\right)\right)$ becomes $O(\log n)$, i.e., we can remove the dependency on the weights and achieve the desired $O(\log n)$ approximation bound as given in the statement of the theorem.

D.2 Proof of Worst Case

We now focus on the proof of the worst case bounds

$$\text{depth}_{wc}(T) \leq O(\log n)\,\text{depth}^*_{wc}, \tag{35}$$
$$\text{expl}_{wc}(T) \leq O(\log n)\,\text{expl}^*_{wc}. \tag{36}$$

for the decision tree T built by SER-DT. Again to simplify the argument, assume w.l.o.g. that (after the preprocessing) the weights are integers and $w_{\min} = 1$.

Let $\mathcal{P}$ be longest path from the root to a leaf in T. For a node g in $\mathcal{P}$, its weighted pair-wise misclassification (`wpm`) is the `wpm` of the set of objects that reach g. The nodes of $\mathcal{P}$ can be split based on whether they were added in Line 3 or Line 8 of one of the recursive calls of SER-DT. More specifically, let G be the set of nodes in $\mathcal{P}$ that are associated with the set of objects $S(a^*, t^*)$ at Line 8 of SER-DT. We note that the total number of nodes in $\mathcal{P}$ is at most

$$\text{depth}_{wc}(T) = |\mathcal{P}| \leq 2|G| + \log(P(O)w(O)), \tag{37}$$

since in every recursive call where a node in G is added to $\mathcal{P}$, its parent is also added, hence the factor of 2 in the formula; on the other hand, if the recursive call adds a node to $\mathcal{P}$ outside of G (Line 3) then it means the condition in Line 2 was satisfied, and so the `wpm` of the node that is added to $\mathcal{P}$ is at most half of the `wpm` of its parent and, thus, only $O(\log(P(O)w(O))$ additions of this type are possible (recall that we are assuming (scaled) integer weights ≥ 1).

Let M^*_{wc} be a multiway decision tree achieving the minimum worst-case depth for the instance. We want to relate $|G|$ to $\text{depth}_{wc}(M^*_{wc})$. The following claim lower bounds the latter quantity and is the worst-case version of Lemma 8.

Lemma 9. *Consider a set $S \subseteq O$ of objects of the instance, and let (a^*, i^*) be the pair attribute/value solving the optimization $\min_a \max_i P(S(a,i))w(S(a,i))$. Then*

$$\text{depth}_{wc}(M^*_{wc}) \geq \frac{1}{2}\frac{P(S)w(S)}{P(S)w(S) - P(S(a^*,i^*))w(S(a^*,i^*))}.$$

Proof. Define the normalized weights $w'(o) := \frac{w(o)}{w(S)}$, which then form a probability distribution over the objects in S. Let M^*_{avg} be a multiway tree for S with minimum average depth with respect to the weights w', i.e. that—over all multiway decision tree M for the instance—minimizes

$$\text{depth}_{avg}(M, w') = \sum_{\ell \in \text{leaf}(M)} w'(\ell)\cdot \text{depth}(\ell),$$

where we included w' in $\text{depth}_{avg}(M, w')$ to avoid confusion.

Using the fact that the maximum is always at least the average (w.r.t. a probability distribution) and the optimality of M^*_{avg} we get

$$\text{depth}_{wc}(M^*_{wc}) \geq \text{depth}_{avg}(M^*_{wc}, w') \geq \text{depth}_{avg}(M^*_{avg}, w'). \tag{38}$$

Moreover, applying Lemma 8 to the instance with objects S and weights w' we obtain

$$\begin{aligned}\mathrm{depth}_{avg}(M^*_{avg}, w') &\geq \frac{1}{2}\frac{P(S)w'(S)^2}{P(S)w'(S) - P(S(a^*,i^*))w'(S(a^*,i^*))}\\ &= \frac{1}{2}\frac{P(S)w(S)}{P(S)w(S) - P(S(a^*,i^*))w(S(a^*,i^*))},\end{aligned} \tag{39}$$

where the equality follows from using the fact $w'(S) = 1$ to remove one of these terms in the numerator and then multiplying both numerator and denominator by $w(S)$. (Observe that (a^*, t^*) is exactly the one required in this application of Lemma 8 because the solutions to the problems $\min_a \max_i P(S(a,i))w(S(a,i))$ and $\min_a \max_i P(S(a,i))w'(S(a,i))$ are the same, as w' is just a scaling of w.) Combining (38) and (39) gives the desired result. $\square$

We are now ready to upper bound $|G|$ with respect to the quantity in (39). For that, let $G = (g_1, g_2, \ldots, g_{|G|})$ be the nodes in G in order of traversal of $\mathcal{P}$ (from root to leaf). For a node g_j, consider the call of SER-DT that created this node (in Line 8). Let S_j be the set of objects that was received as input by this call, and let (a^*_j, t^*_j) be the attribute and threshold value used in Line 8 to create node g_j. Thus, the objects of the whole instance that reach node g_j (in the tree T) are precisely $S_j(a^*_j, t^*_j)$.

Applying Lemma 9 to the set of objects S_j and reorganizing the terms gives

$$\begin{aligned}1 \;\leq\; & 2\cdot \mathrm{depth}_{wc}(M^*_{wc})\\ &\cdot \frac{P(S_j)w(S_j) - P(S_j(a^*_j,i^*_j))w(S_j(a^*_j,i^*_j))}{P(S_j)w(S_j)}.\end{aligned}$$

Thus:

$$\begin{aligned}|G| &= \sum_{j\leq|G|} 1\\ &\leq\; 2\cdot\mathrm{depth}_{wc}(M^*_{wc})\\ &\quad\cdot \sum_{j\leq|G|}\frac{P(S_j)w(S_j) - P(S_j(a^*,i^*))w(S_j(a^*,i^*))}{P(S_j)w(S_j)}\\ &\leq 2\cdot\mathrm{depth}_{wc}(M^*_{wc})\\ &\quad\cdot\sum_{j\leq|G|}\ln\left(\frac{P(S_j)w(S_j)}{P(S_j(a^*_j,i^*_j))w(S_j(a^*_j,i^*_j))}\right),\end{aligned}$$

where the last step uses the inequality $1 - \frac{1}{x} \leq \ln x$ which is valid for all $x \in (0,1]$. But by definition we have $S_{j+1} \subseteq S_j(a_j, t_j)$, and hence the denominator in the log can be lower bounded as $P(S_j(a^*_j,i^*_j))w(S_j(a^*_j,i^*_j)) \geq P(S_j)w(S_j)$, and hence we obtain the desired upper bound on G:

$$\begin{aligned}|G| &\leq\; 2\cdot\mathrm{depth}_{wc}(M^*_{wc})\cdot\sum_{j\leq|G|}\ln\left(\frac{P(S_j)w(S_j)}{P(S_{j+1})w(S_{j+1})}\right)\\ &\leq\; 2\cdot\mathrm{depth}_{wc}(M^*_{wc})\cdot\ln(P(S_1)w(S_1))\\ &\leq 2\cdot\mathrm{depth}_{wc}(M^*_{wc})\cdot\ln(P(O)w(O)).\end{aligned}$$

Plugging this bound in (37) and using the equivalence between the optimal worst-case explanation size and the optimal worst-case depth for multiway trees from Lemma 2, i.e., $\mathrm{expl}^*_{wc} = \mathrm{depth}^*_{wc} = \mathrm{depth}_{wc}(M^*_{wc})$, we obtain

$$\mathrm{depth}_{wc}(T) \;\leq\; 4\,\mathrm{expl}^*_{wc}\cdot\ln(P(O)w(O)) \;\leq\; O(\log n)\cdot\,\mathrm{expl}^*_{wc},$$

the last inequality holding since $P(O) \leq n^2$ and, because of the pre-processing step before the algorithm we have $w(O) \leq n^4$. This implies both the bounds in (35)-(36), $\mathrm{expl}_{wc}(T) \leq O(\log n)\cdot \mathrm{expl}^*_{wc}$ and $\mathrm{depth}_{wc}(T) \leq O(\log n)\cdot\,\mathrm{depth}^*_{wc}$, thus completing the proof of the lemma.

E Experiments – Additional Details and Statistics

Our experiments were executed with the following settings: Notebook Inspiron 14-7460; Processor: 7th generation Intel Core i5; 16GB RAM (DDR4 at 2400 MHz); Storage type: SSD; GPU NVIDIA GeForce 940MX.

For our implementation we employed Python 3.8.10; pandas 1.4.2; numpy 1.22.3; sklearn 1.0.2. Our code is available on

`https://github.com/user-anonymous-researcher/interpretable-dts`

Our datasets as well as their main characteristics are described in Table 2.

Table 2: Datasets. n is the number of examples; `numeric` the number of numerical features; `categ.` the number of categorical features; d the total of features (after doing the one-hot encoding) and `classes` the number of output classes.

Dataset	n	numeric	categ.	d	classes	source
Anuran	7195	22	0	22	4	[33]
Audit Risk	773	26	0	26	2	[45]
Avila	20867	10	0	10	12	[53]
Banknote	1372	4	0	4	2	[6]
Bankruptcy Polish	4885	64	0	64	2	[54]
Cardiotocography	2126	21	0	21	10	[14]
Collins	1000	19	0	19	30	[55]
Defaults Credit Card	30000	20	3	33	2	[57]
Dry Bean	13611	16	0	16	7	[34]
EEG Eye State	14980	14	0	14	2	[51]
Htru2	17898	8	0	8	2	[42]
Iris	150	4	0	4	3	[20]
Letter Recognition	20000	16	0	16	26	[38]
Mice	552	77	3	83	8	[27]
OBS Network	1060	19	2	24	4	[50]
Occupancy Room	10129	16	0	16	4	[49]
Online Shoppers Intention	12330	12	5	54	2	[52]
Pen Digits	10992	16	0	16	10	[5]
Poker Hand	1025010	10	0	10	10	[15]
Sensorless	58509	48	0	48	11	[7]

To generate the plots from Figure 2, we executed SER-DT with `FactorExpl` varying in the set

$$\{0.1 \times i | i = 1, \ldots, 9\} \cup \{0.9 + 0.01 \times i | i = 1, \ldots, 10\}.$$

In the following sections, we present some additional metrics and visualizations related to the experiments described in Section 6. Moreover, we also present two new experiments: in one of them, we remove the constraint on the depth of the trees while in the other we replace the `Gini` with entropy as a splitting criterion. We note that each entry presented in the following tables is given by the the average of 10 runs using different seeds to select the examples in the training and testing set.

E.1 Metrics related to depth and running time

Table 3 shows the metrics depth_{avg} and depth_{wc} for the decision trees produced by both `CART` and SER-DT (experiment from Section 6). The metric depth_{avg} for our algorithm is better than that of `CART` for 17 datasets and it is worse for only 3. For depth_{wc} the results are similar, which is somehow expected since we set the maximum allowed depth to 6.

Table 4 shows the average running times in seconds of `CART` and our algorithm. The ratio between the fastest and slowest is at most 2 for all datasets.

Table 3: depth_{avg} and depth_{wc} for the experiment described in Section 6. We bold-faced the cases where the difference in accuracy is larger than 1% and also the cases where the gain in terms of depth_{avg} or depth_{wc} is larger than 25%.

Dataset	Test Accuracy		depth_{avg}		depth_{wc}	
	SER-DT	CART	SER-DT	CART	SER-DT	CART
anuran	94,8%	94,7%	5,38	5,57	6,00	6,00
audit risk	99,9%	99,9%	1,00	1,00	1,00	1,00
avila	61,5%	**63,2%**	5,14	5,29	6,00	6,00
banknote	97,6%	98,1%	4,39	4,43	6,00	6,00
bankruptcy polish	96,6%	96,9%	4,45	4,94	6,00	6,00
cardiotocography	89,5%	89,8%	4,98	5,51	6,00	6,00
collins	13,2%	**15,6%**	5,89	5,91	6,00	6,00
default credit card	82,0%	81,9%	**2,15**	4,33	6,00	6,00
dry bean	90,1%	89,8%	4,76	5,41	6,00	6,00
eeg eye state	74,1%	73,6%	5,15	5,47	6,00	6,00
htru2	97,7%	97,7%	**2,80**	5,30	6,00	6,00
iris	94,2%	93,6%	2,50	2,52	4,90	4,80
letter recognition	44,9%	**47,9%**	5,96	5,94	6,00	6,00
mice	99,9%	99,9%	3,05	3,05	3,60	3,60
obs network	**91,7%**	89,5%	4,47	4,39	6,00	6,00
occupancy room	99,4%	99,3%	4,72	4,83	6,00	6,00
online shoppers intention	89,3%	89,8%	3,89	4,77	6,00	6,00
pen digits	**88,6%**	86,9%	5,73	5,80	6,00	6,00
poker hand	52,9%	**55,0%**	4,61	4,30	6,00	5,30
sensorless	**87,4%**	80,1%	5,26	5,33	6,00	6,00
Average	82,3%	82,2%	4,31	4,70	5,58	5,54

Table 4: Running times for CART and our algorithm for the experiment described in Section 6

Dataset	SER-DT (sec)	CART (sec)
anuran	7.0	3.7
audit risk	0.1	0.1
avila	9.0	9.2
banknote	0.2	0.1
bankruptcy polish	9.8	5.2
cardiotocography	1.4	1.7
collins	2.2	1.7
default credit card	18.1	19.3
dry bean	12.1	7.1
eeg eye state	3.5	4.0
htru2	5.0	3.0
iris	0.0	0.0
letter recognition	17.1	26.1
mice	1.7	1.0
obs network	0.5	0.5
occupancy room	2.7	3.7
online shoppers intention	9.1	11.7
pen digits	5.4	7.1
poker hand	330.8	443.6
sensorless	178.5	120.5

E.2 Sensitivity to `FactorExpl`

Figure 2 suggests that `FactorExpl` can be used to provide a trade-of between accuracy and explainability (measured according to expl_{avg}). To provide additional evidence, in Table 5 we show the test accuracy and expl_{avg} for `FactorExpl` $\in \{0.9, 0.95, 0.99\}$.

Table 5: Sensitivity with respect to `FactorExpl`(`FE` below)

Dataset	Test Accuracy			expl_{avg}		
	`FE`= 0,99	`FE`=0,95	`FE`=0,90	`FE`=0,99	`FE`=0,95	`FE`=0,90
anuran	94,8%	94,8%	94,6%	5,11	4,50	3,85
audit risk	99,9%	99,9%	99,9%	1,00	1,00	1,00
avila	66,0%	57,8%	55,9%	4,22	1,99	1,64
banknote	97,8%	97,6%	97,5%	2,51	2,38	2,36
bankruptcy polish	96,7%	96,6%	96,7%	3,73	1,92	1,50
cardiotocography	89,7%	89,3%	89,5%	4,51	4,10	3,98
collins	15,9%	13,1%	12,8%	3,74	1,61	1,35
default credit card	81,9%	82,0%	82,0%	1,94	1,41	1,40
dry bean	89,7%	90,0%	89,8%	3,60	3,25	3,05
eeg eye state	74,6%	72,9%	67,1%	4,32	3,15	1,82
htru2	97,7%	97,7%	97,6%	1,27	1,19	1,15
iris	94,2%	94,2%	94,2%	1,75	1,75	1,75
letter recognition	51,9%	42,4%	40,1%	4,20	2,90	2,59
mice	99,9%	99,9%	99,9%	3,05	3,05	3,05
obs network	91,7%	91,3%	90,3%	3,75	3,15	2,67
occupancy room	99,3%	99,4%	99,5%	4,25	4,15	4,07
online shoppers intention	89,9%	89,1%	88,7%	3,82	3,12	2,94
pen digits	88,5%	88,6%	87,7%	4,90	4,75	4,42
poker hand	53,0%	52,8%	52,7%	1,89	1,79	1,76
sensorless	87,5%	87,0%	85,9%	3,01	2,85	2,67
Average	83,0%	81,8%	81,1%	3,33	2,70	2,45

We observe that expl_{avg} has a very predictable behavior: the smaller the `FactorExpl` the smaller the expl_{avg} (last 3 columns of this table). In terms of accuracy, for 13 datasets the difference between `FactorExpl` $= 0.99$ and `FactorExpl` $= 0.9$ is smaller than 1%. This number increases to 16 when we consider `FactorExpl` $= 0.99$ and `FactorExpl` $= 0.95$. We do not recommend using small values for `FactorExpl`(< 0.9) in practice because the loss in terms of accuracy may be severe. In fact, we recommend using SER-DT with `FactorExpl` $\in [0.95, 0.99]$.

E.3 Removing the maximum depth constraint

In our experiments, in order to prevent (very) large trees, we set the maximum allowed depth to 6. Table 6 and 7 show the results when this constraint is removed. Some observations are in order:

- SER-DT builds trees that are much more shallower than those built by `CART` (Columns 6 and 7 of Table 7). This was not possible to observe when the maximum depth was set to 6;
- The gains of SER-DT over `CART` for all explainability metrics become larger when the depth constraint is removed.

E.4 Entropy as a split criterion

In our experiments on Section 6 we compared SER-DT against `CART`. We also implemented a variation of our method that employs the Shannon Entropy, rather than `Gini`, to evaluate the goodness of a split. We note that (a normalized version of) the entropy is employed by the widely used `C4.5` algorithm [48]. We tested our variation against a variation of `CART` that uses the entropy rather than `Gini` to evaluate the quality of a split and also to determine whether a leaf shall be expanded or not (stopping rule).

Table 8 presents the results for the test accuracy and the metrics related to the explanation size. The results are inline with those from Section 6.

Table 6: Test accuracy, expl_{avg} and expl_{wc} when no limit on the maximum depth is set. We bold-faced the cases where the difference in accuracy is larger than 1% and also the cases where the gain in terms of expl_{avg} or expl_{wc} is larger than 30%.

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	SER-DT	CART	SER-DT	CART	SER-DT	CART
anuran	95,7%	95,6%	**6,00**	8,74	9,4	11,1
audit risk	99,9%	99,9%	1,00	1,00	1	1
avila	92,8%	**98,4%**	5,47	5,81	9	9,6
banknote	97,8%	98,0%	2,46	2,56	3,8	3,8
bankruptcy polish	95,9%	95,8%	**3,14**	7,25	**8**	13
cardiotocography	87,6%	88,0%	**5,73**	8,54	**8,7**	14,6
collins	12,0%	**14,5%**	**4,30**	7,81	**7,3**	12,7
default credit card	72,7%	72,5%	**5,74**	13,08	**11,1**	20,2
dry bean	89,5%	89,3%	**4,60**	7,41	8,6	11,4
eeg eye state	82,2%	82,9%	6,48	8,31	11,1	13
htru2	96,7%	96,7%	**1,74**	4,24	6,2	7,3
iris	94,2%	93,6%	1,75	1,76	3,1	3,4
letter recognition	84,3%	**86,2%**	6,84	8,83	11,5	14,8
mice	99,9%	99,9%	3,05	3,05	3,6	3,6
obs network	100,0%	100,0%	3,94	4,83	6,5	8,2
occupancy room	99,6%	99,5%	4,26	4,98	6,9	7,9
online shoppers intention	86,2%	86,3%	**5,72**	9,15	**10,1**	16,4
pen digits	95,7%	96,0%	6,64	7,87	10,4	12,7
poker hand	**81,0%**	62,5%	**6,05**	8,74	10	10
sensorless	98,4%	98,1%	**4,93**	9,60	**10,3**	20,3
Average	88,1%	87,7%	4,49	6,68	7,83	10,75

Table 7: Test accuracy, depth_{avg} and depth_{wc} when no limit on the maximum depth is set. We bold-faced the cases where the difference in accuracy is larger than 1% and also the cases where the gain in terms of depth_{avg} or depth_{wc} is larger than 20%.

Dataset	Test Accuracy		depth_{avg}		depth_{wc}	
	SER-DT	CART	SER-DT	CART	SER-DT	CART
anuran	95,7%	95,6%	**7,69**	10,79	**11,2**	15,1
audit risk	99,9%	99,9%	1,00	1,00	1,0	1,0
avila	92,8%	**98,4%**	11,25	11,49	18,9	21,1
banknote	97,8%	98,0%	4,51	4,60	6,9	7,3
bankruptcy polish	95,9%	95,8%	**5,37**	8,22	**9,8**	16,6
cardiotocography	87,6%	88,0%	**7,56**	10,20	**11,5**	17,5
collins	12,0%	**14,5%**	9,44	10,55	**16,2**	21,1
default credit card	72,7%	72,5%	**12,46**	19,89	**21,0**	42,3
dry bean	89,5%	89,3%	**9,20**	12,16	**15,8**	25,0
eeg eye state	82,2%	82,9%	10,81	13,38	**18,1**	25,2
htru2	96,7%	96,7%	**8,26**	10,86	**14,4**	22,2
iris	94,2%	93,6%	2,50	2,52	4,9	4,8
letter recognition	84,3%	**86,2%**	11,39	12,92	22,4	27,8
mice	99,9%	99,9%	3,05	3,05	3,6	3,6
obs network	100,0%	100,0%	5,36	5,73	8,9	9,8
occupancy room	99,6%	99,5%	5,09	6,08	9,6	10,5
online shoppers intention	86,2%	86,3%	**9,01**	11,56	**15,8**	24,2
pen digits	95,7%	96,0%	8,91	9,73	14,1	17,6
poker hand	**81,0%**	62,5%	17,92	20,75	**30,5**	40,7
sensorless	98,4%	98,1%	**9,64**	13,66	**17,6**	31,6
Average	88,1%	87,7%	8,02	9,96	13,6	19,3

Table 8: expl_{avg} and expl_{wc} when Entropy (Entr.) is used instead of `GINI`. We bold-faced the cases where the difference in accuracy is larger than 1% and also the cases where the gain in terms of expl_{avg} or expl_{wc} is larger than 20%.

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	SER-DT $_{Ent}$	`Entr.`	SER-DT $_{Ent}$	`Entr.`	SER-DT $_{Ent}$	`Entr.`
anuran	94,5%	94,5%	4,92	5,06	6,0	6,0
audit risk	99,9%	99,9%	1,00	1,00	1,0	1,0
avila	**66,5%**	65,3%	3,54	4,18	5,0	5,1
banknote	97,9%	97,9%	2,41	2,51	3,8	3,8
bankruptcy polish	97,1%	97,3%	2,95	3,17	5,6	6,0
cardiotocography	89,1%	89,6%	4,36	4,69	6,0	6,0
collins	14,1%	14,2%	**3,51**	4,81	5,0	5,7
default credit card	82,0%	81,9%	**1,41**	3,82	**4,2**	6,0
dry bean	90,1%	90,0%	3,66	4,34	5,8	6,0
eeg eye state	**73,2%**	72,1%	3,42	4,26	5,6	6,0
htru2	97,8%	97,8%	**1,47**	1,97	4,6	5,0
iris	94,5%	94,2%	1,70	1,73	3,2	3,3
letter recognition	58,5%	**59,6%**	4,13	5,01	6,0	6,0
mice	99,9%	99,9%	3,00	3,00	3,0	3,0
obs network	**89,4%**	87,5%	3,54	4,17	5,2	6,0
occupancy room	99,5%	99,5%	3,27	3,37	4,6	5,0
online shoppers intention	89,1%	89,9%	**2,63**	3,47	**4,2**	5,9
pen digits	89,6%	89,7%	5,12	5,44	6,0	6,0
poker hand	53,0%	**55,7%**	**1,83**	4,53	**4,0**	5,0
sensorless	90,0%	90,0%	**3,08**	4,53	4,9	5,8
Average	83,3%	83,3%	3,05	3,71	4,68	5,13

E.5 Boxplots for `CART` and SER-DT

Figures 5-8 present boxplots for the test accuracy of `CART` and SER-DT for the experiments described on Section 6. Figures 9-12 present boxplots for the expl_{avg}.

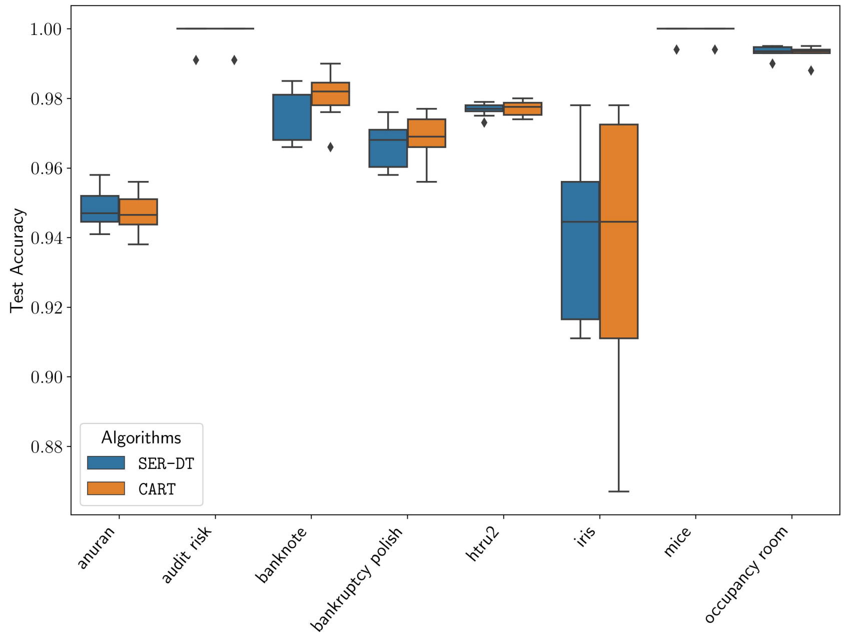

Figure 5: Test Accuracy for `CART` and SER-DT for some datasets

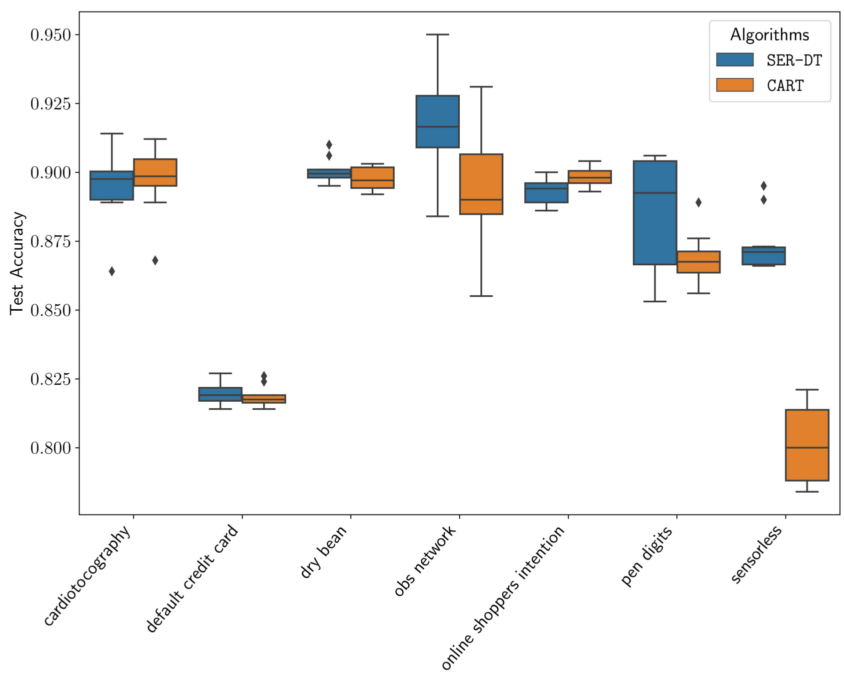

Figure 6: Test Accuracy for `CART` and SER-DT for some datasets

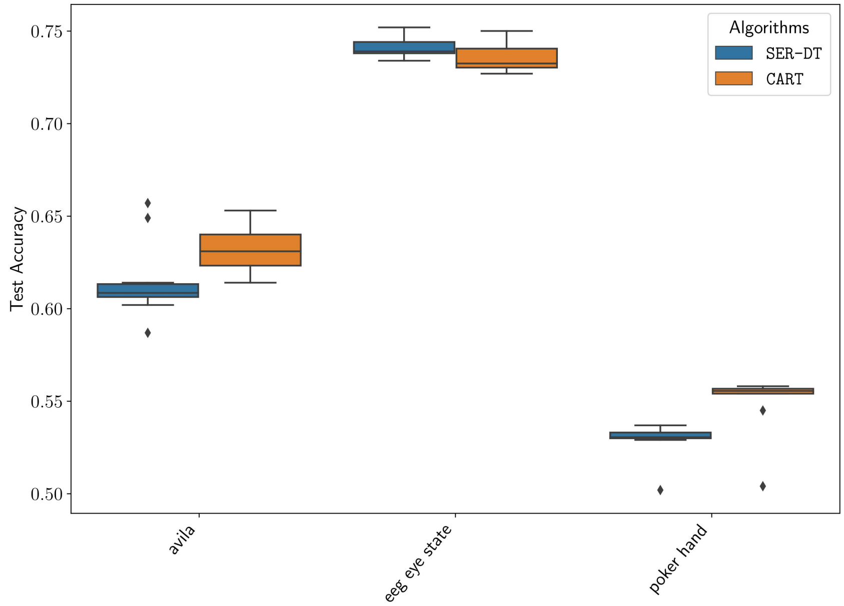

Figure 7: Test Accuracy for `CART` and SER-DT for some datasets

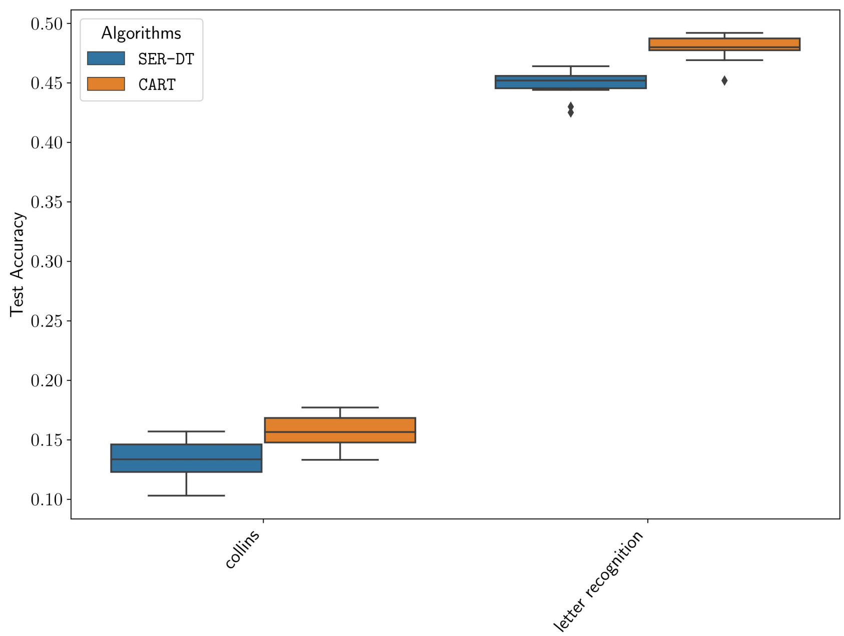

Figure 8: Test Accuracy for `CART` and SER-DT for some datasets

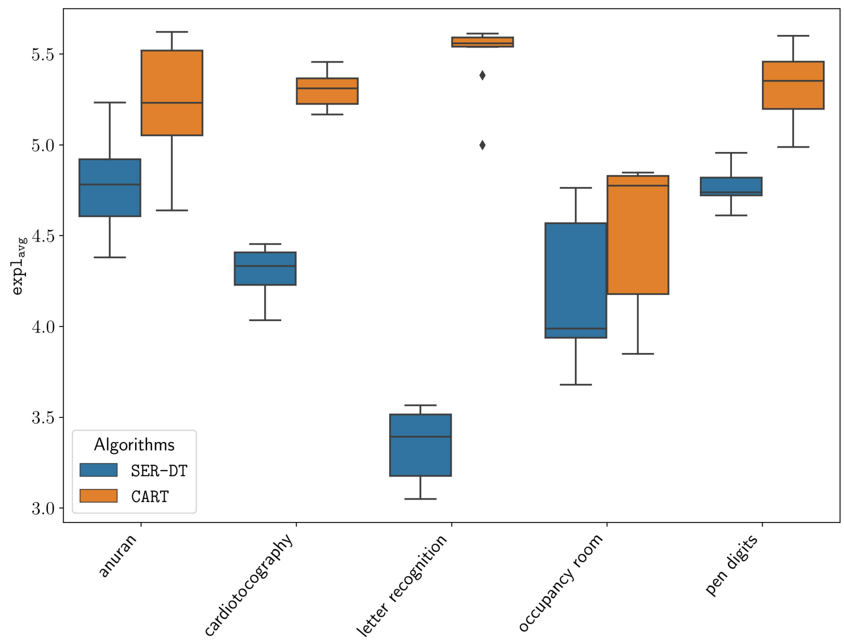

Figure 9: expl_{avg} for `CART` and SER-DT for some datasets

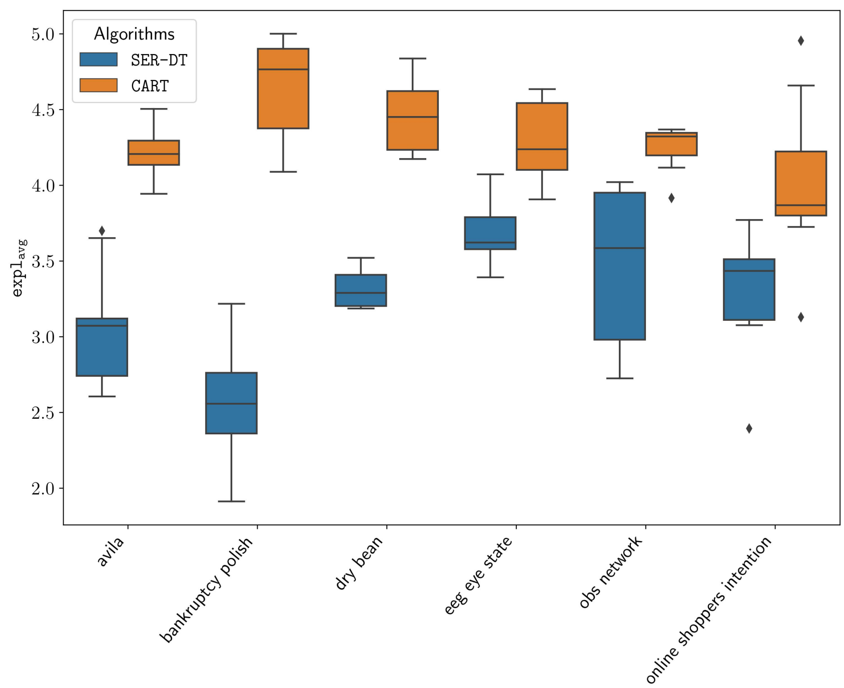

Figure 10: expl_{avg} for `CART` and SER-DT for some datasets

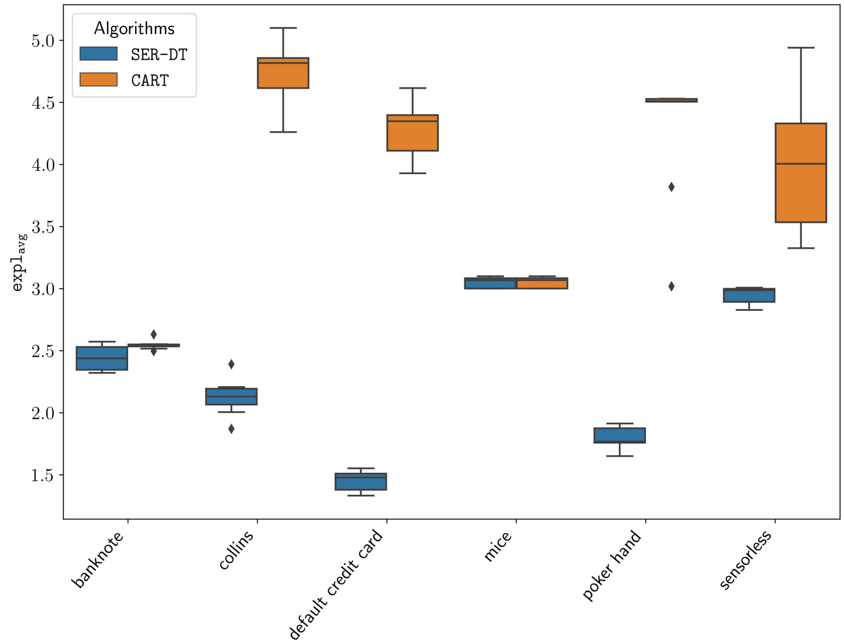

Figure 11: expl_{avg} for CART and SER-DT for some datasets

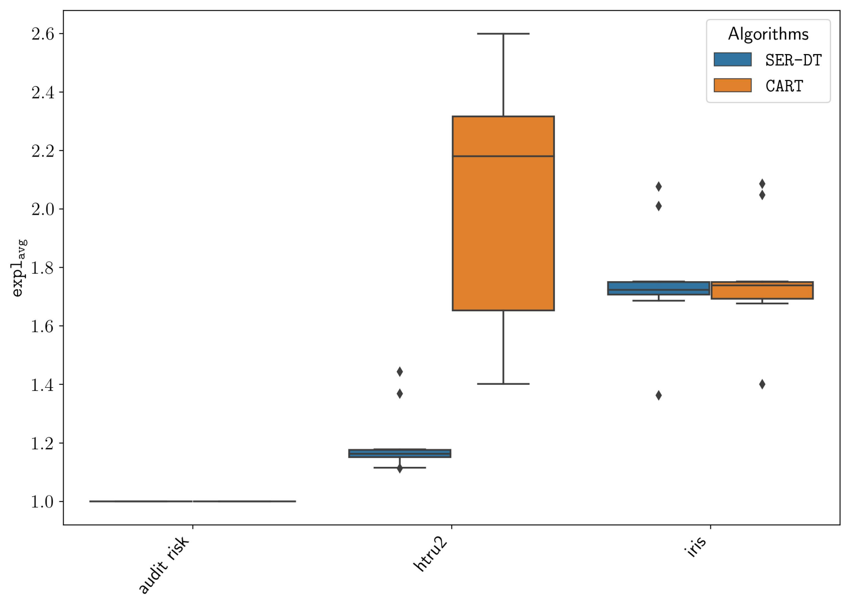

Figure 12: expl_{avg} for CART and SER-DT for some datasets

E.6 Other examples of trees

We present some other examples of trees produced by SER-DT and CART. These examples provide additional evidence that our algorithm generates trees with better explainability but similar accuracy compared to CART. All trees were constructed by setting a maximum allowed depth to 4. For SER-DT we set FactorExpl $= 0.97$, such as reported on Section 6.

Our first example employs dataset default credit card. Figures 13 and 14 show the trees produced by CART and SER-DT, respectively. CART has a Test Accuracy of 81.9%, while SER-DT obtains 81.8%. In contrast, visually the SER-DT tree is much simpler than that of CART, and SER-DT achieves $\text{expl}_{avg} = 1.19$ against $\text{expl}_{avg} = 2.27$ from CART.

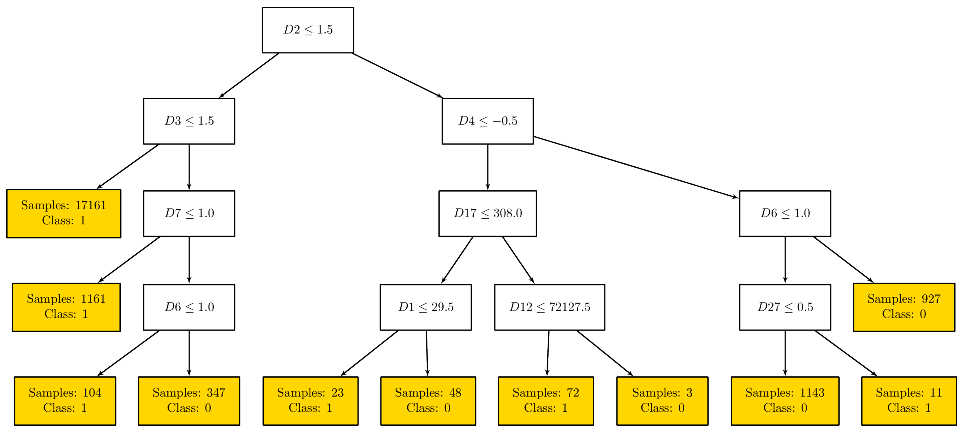

Figure 13: Tree produced with CART for dataset default credit cart

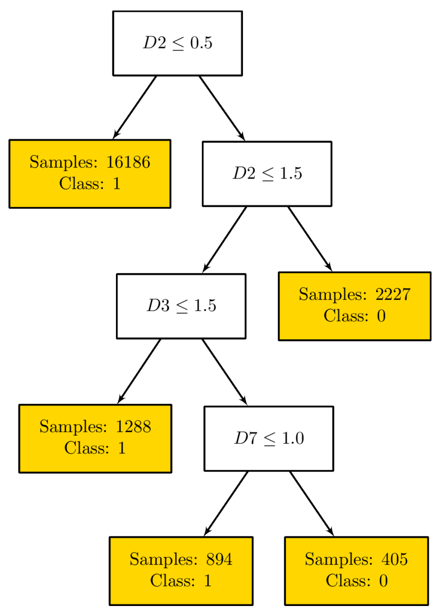

Figure 14: Tree produced with SER-DT for dataset default credit cart

.

Our second example uses dataset `online shoppers intention`, whose corresponding decision trees are represented in Figures 15 and 16. `CART` generates a tree with a Test Accuracy of 90.0%, while SER-DT achieves 89.1% for this metric. In this case, SER-DT also clearly yields a simpler tree, achieving $\text{depth}_{avg} = 1.54$ against $\text{depth}_{avg} = 2.38$ from `CART`.

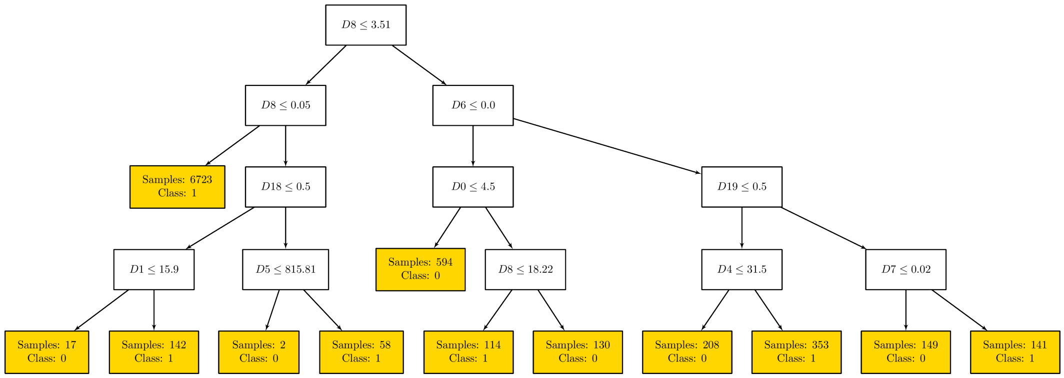

Figure 15: Tree produced with `CART` for dataset `online shoppers intention`

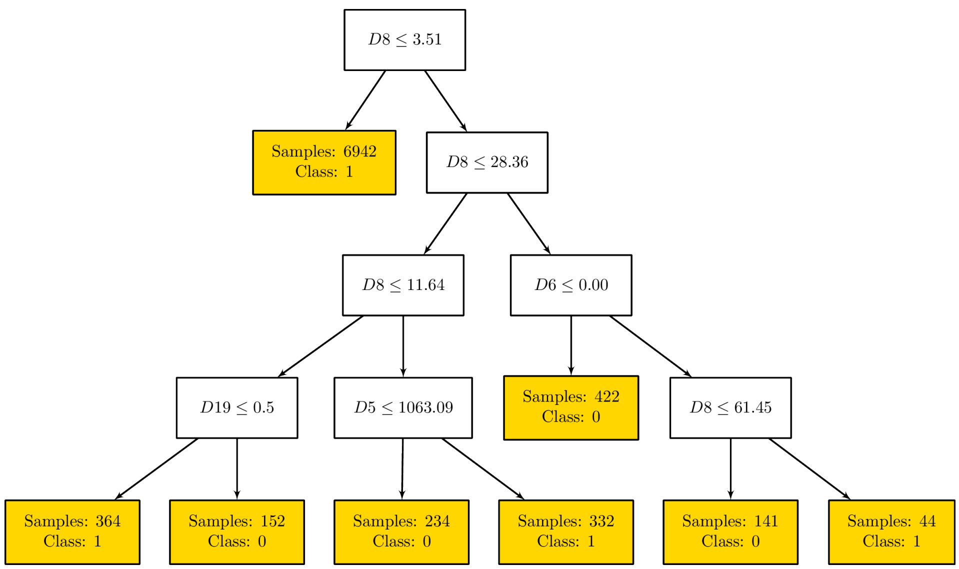

Figure 16: Tree produced with SER-DT for dataset `online shoppers intention`

The final example shows trees for dataset `dry bean`, in Figures 17 and 18. Although at first glance there doesn't seem to be much improvement, SER-DT yields a tree with expl_{avg} of 2.60, in contrast to 3.49 for `CART`, which is a reduction of 25.6%. In terms of prediction, SER-DT gives 82.8% of Test Accuracy, while `CART` gives 82.4%.

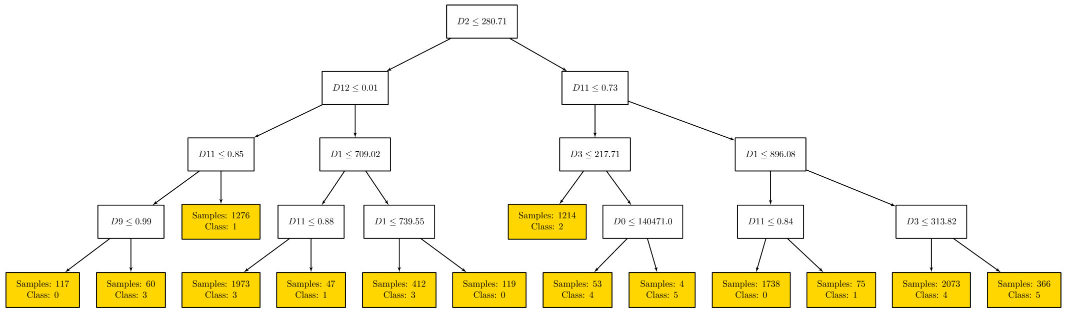

Figure 17: Tree produced with `CART` for dataset `dry bean`

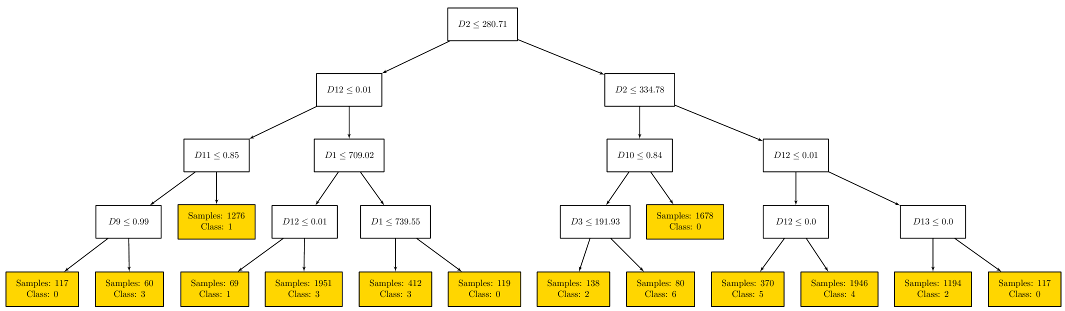

Figure 18: Tree produced with SER-DT for dataset `dry bean`

Table 9: Test Accuracy, expl_{avg} and expl_{wc} for `FactorExpl` $= 0.97$. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set.

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	SER-DT	EC^2	SER-DT	EC^2	SER-DT	EC^2
anuran	**94.8 $\pm$ 0.5%**	89.4 $\pm$ 1.7%	4.78	4.36	6.00	5.90
audit risk	99.9 $\pm$ 0.3%	99.3 $\pm$ 0.8%	**1.00**	3.07	**1.00**	3.90
avila	**61.5 $\pm$ 2.1%**	57.7 $\pm$ 1.2%	3.06	4.03	4.90	6.00
banknote	**97.6 $\pm$ 0.8%**	92.3 $\pm$ 1.7%	2.44	1.88	3.80	3.00
bankruptcy polish	96.6 $\pm$ 0.7%	97.4 $\pm$ 0.4%	2.56	2.19	5.60	5.50
cardiotocography	**89.5 $\pm$ 1.3%**	80.1 $\pm$ 3.3%	4.30	3.81	5.90	5.70
collins	13.2 $\pm$ 1.8%	**16.0 $\pm$ 1.2%**	**2.13**	4.95	**4.40**	6.00
default credit card	**82.0 $\pm$ 0.4%**	80.5 $\pm$ 0.3%	**1.45**	3.65	**4.50**	6.00
dry bean	**90.1 $\pm$ 0.4%**	83.9 $\pm$ 0.5%	3.32	4.40	5.10	6.00
eeg eye state	**74.1 $\pm$ 0.5%**	71.1 $\pm$ 0.5%	3.69	3.78	5.90	5.60
htru2	97.7 $\pm$ 0.2%	97.7 $\pm$ 0.1%	1.20	1.52	4.30	4.60
iris	**94.2 $\pm$ 2.6%**	86.7 $\pm$ 7.0%	**1.75**	2.57	3.10	3.60
letter recognition	**44.9 $\pm$ 1.2%**	37.8 $\pm$ 0.5%	**3.34**	5.32	5.50	6.00
mice	**99.9 $\pm$ 0.2%**	71.0 $\pm$ 4.1%	**3.05**	4.79	**3.60**	6.00
obs network	**91.7 $\pm$ 2.1%**	81.2 $\pm$ 1.5%	3.48	3.95	5.30	5.30
occupancy room	99.4 $\pm$ 0.2%	98.6 $\pm$ 0.4%	4.18	4.77	5.30	5.60
online shoppers intention	89.3 $\pm$ 0.5%	89.5 $\pm$ 0.6%	3.30	1.84	5.10	5.60
pen digits	**88.6 $\pm$ 2.0%**	75.3 $\pm$ 1.8%	4.76	4.74	5.80	6.00
poker hand	52.9 $\pm$ 1.0%	52.0 $\pm$ 0.7%	**1.80**	3.72	**3.80**	5.70
sensorless	**87.4 $\pm$ 1.0%**	74.4 $\pm$ 0.4%	**2.94**	4.19	4.90	6.00
Average	82.3%	76.6%	2.93	3.68	4.69	5.40

E.7 Comparision between SER-DT and $\texttt{EC}^2$

As noted in Section 5 and Section 6, SER-DT combines the theoretically best possible guarantee with very competitive performance in practice. The results shown in this section give evidence that in general optimal approximation guarantee on the (average) depth does not imply good results in terms of explanation size and accuracy.

Table 9 presents the comparison between the $\texttt{EC}^2$ algorithm and our SER-DT, with respect to accuracy and explanation size. $\texttt{EC}^2$ is a greedy algorithm proposed by [22] which is proved to achieve $O(\log n)$-approximation guarantee for the minimization of the average depth. Note that this means that it attains the best approximation guarantee that it is possible in polynomial time, under the hypothesis $P \neq NP$.

In terms of accuracy SER-DT performs significantly better than $\texttt{EC}^2$ on almost all datasets, and, similarly, in terms of the interpretability metrics the results also show a significant difference in favour of our algorithm. More precisely, on 14 datasets (bold-faced on columns 2 and 3) we observe a difference larger than 1% in terms of accuracies; on 13 of them, our algorithm outperforms $\texttt{EC}^2$ while only on 1, $\texttt{EC}^2$ is better. When considering the average explanation size, our algorithm is better than $\texttt{EC}^2$ on 14 datasets, and for 8 of them (bold-faced on column 4), it improves on the expl_{avg} of $\texttt{EC}^2$ by at least 25%. Finally, in terms of expl_{wc}, on 15 datasets our algorithm is at least as good as $\texttt{EC}^2$, and on 5 of them the improvement is at least 25% (bold-faced in column 6).

Table 10: Test Accuracy, depth_{avg} and depth_{wc} for $\texttt{FactorExpl} = 0.97$. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set.

Dataset	Test Accuracy		depth_{avg}		depth_{wc}	
	SER-DT	$\texttt{EC}^2$	SER-DT	$\texttt{EC}^2$	SER-DT	$\texttt{EC}^2$
anuran	**94.8 ±0.5%**	89.4 ± 1.7%	5.38	4.83	6.00	6.00
audit risk	99.9 ±0.3%	99.3 ± 0.8%	**1.00**	4.34	**1.00**	5.90
avila	**61.5 ±2.1%**	57.7 ± 1.2%	5.14	5.04	6.00	6.00
banknote	**97.6 ±0.8%**	92.3 ± 1.7%	4.39	3.85	6.00	6.00
bankruptcy polish	96.6 ±0.7%	97.4 ± 0.4%	4.45	2.40	6.00	6.00
cardiotocography	**89.5 ±1.3%**	80.1 ± 3.3%	4.98	5.15	6.00	6.00
collins	13.2 ±1.8%	**16.0 ± 1.2%**	5.89	5.89	6.00	6.00
default credit card	**82.0 ±0.4%**	80.5 ± 0.3%	**2.15**	3.79	6.00	6.00
dry bean	**90.1 ±0.4%**	83.9 ± 0.5%	4.76	5.03	6.00	6.00
eeg eye state	**74.1 ±0.5%**	71.1 ± 0.5%	5.15	5.19	6.00	6.00
htru2	97.7 ±0.2%	97.7 ± 0.1%	2.80	3.26	6.00	6.00
iris	**94.2 ±2.6%**	86.7 ± 7.0%	**2.50**	3.34	4.90	5.70
letter recognition	**44.9 ±1.2%**	37.8 ± 0.5%	5.96	5.87	6.00	6.00
mice	**99.9 ±0.2%**	71.0 ± 4.1%	**3.05**	4.93	**3.60**	6.00
obs network	**91.7 ±2.1%**	81.2 ± 1.5%	4.47	4.62	6.00	6.00
occupancy room	99.4 ±0.2%	98.6 ± 0.4%	4.72	5.64	6.00	6.00
online shoppers intention	89.3 ±0.5%	89.5 ± 0.6%	3.89	3.49	6.00	6.00
pen digits	**88.6 ±2.0%**	75.3 ± 1.8%	5.73	5.65	6.00	6.00
poker hand	52.9 ±1.0%	52.0 ± 0.7%	4.61	4.71	6.00	6.00
sensorless	**87.4 ±1.0%**	74.4 ± 0.4%	5.26	5.47	6.00	6.00
Average	82.3%	76.6%	4.31	4.62	5.58	5.98

Table 10 shows the analogous results for the metrics depth_{avg} and depth_{wc}. We bold-faced values in column 4 and 6 that show an improvements by at least 25% over $\texttt{EC}^2$. We notice that for 12 datasets our algorithm has better depth_{avg} than $\texttt{EC}^2$. Moreover, considering those where SER-DT's performance is worse, on only one the difference is larger than 25% (bankruptcy polish). The values for the metric depth_{wc} are similar for both algorithms, which is likely due to having set an upper bound of 6 on the maximum depth. Nonetheless, we observe that for 2 datasets the performance of SER-DT is significantly better (audit risk and mice).

E.8 Post-pruning experiments

In this section, we present the results obtained through the application of a post-pruning strategy for trees. The algorithm consisted of checking if merging two leaves would not decrease the accuracy in a validation set formed by half of the original test set. If so, these leaves are joined and the algorithm proceeds recursively to the root.

Table 11 presents the results obtained before and after post-pruning for SER-DT, limiting the maximum depth to 6 and `FactorExpl` $= 0.97$. We observed that post-pruning did not have a significant effect in terms of accuracy, leading to differences of not more than 0.5% for all datasets. In terms of explanation size, post-pruning produces smaller expl_{avg} for all datasets (as expected, since we are joining nodes) and for 4 the gain was more than 20% (bold-faced in table). Similarly, the pruning algorithm led to no worse values of expl_{wc} for all datasets, and for 3 of these, the difference was greater than 20%.

Table 11: Test Accuracy, expl_{avg} and expl_{wc} for SER-DT with `FactorExpl` $= 0.97$, with and without post-pruning. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set. We bold-faced the values (columns 4,5,6 and 7) that represent an improvement of more than 20% in terms of explainability.

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	No pruning	Pruning	No pruning	Pruning	No pruning	Pruning
anuran	94.7%	94.5%	4.78	4.07	6.00	6.00
audit risk	99.9%	99.9%	1.00	1.00	1.00	1.00
avila	61.2%	60.9%	3.06	2.98	4.90	4.90
banknote	97.8%	97.8%	2.44	2.37	3.80	3.40
bankruptcy polish	96.5%	97.3%	2.56	**1.45**	5.60	4.50
cardiotocography	89.1%	89.4%	4.30	3.76	5.90	5.60
collins	13.1%	13.1%	2.13	**1.40**	4.40	**3.30**
default credit card	82.0%	82.0%	1.45	1.29	4.50	3.90
dry bean	90.1%	89.9%	3.32	3.17	5.10	4.70
eeg eye state	73.9%	73.6%	3.69	3.56	5.90	5.80
htru2	97.7%	97.8%	1.20	1.09	4.30	3.70
iris	93.7%	93.2%	1.75	1.44	3.10	**1.70**
letter recognition	44.9%	44.4%	3.34	3.21	5.50	5.30
mice	100.0%	100.0%	3.05	3.05	3.60	3.60
obs network	92.0%	91.6%	3.48	3.35	5.30	5.30
occupancy room	99.3%	99.3%	4.18	**3.22**	5.30	4.70
online shoppers intention	89.3%	89.8%	3.30	**1.81**	5.10	**4.00**
pen digits	88.7%	88.3%	4.76	4.56	5.80	5.40
poker hand	52.9%	52.9%	1.80	1.78	3.80	3.80
sensorless	87.4%	87.4%	2.94	2.81	4.90	4.80
Average	82.2%	82.1%	2.93	2.57	4.69	4.27

Table 12 is similar to the previous table but shows metrics depth_{avg} and depth_{wc}. For all datasets, there is no worsening in these metrics and for 6 of these there is an improvement of more than 20% in depth_{avg}, while for 1 dataset there is an improvement of more than 20% in depth_{wc}.

Tables 13 and 14 show the results for `CART` and SER-DT after applying post-pruning. We observe similar accuracy values for both algorithms. On 7 of the datasets (bold-faced in columns 2 and 3) we observe a difference of more than 1% in accuracies; on 3 of them, SER-DT outperforms `CART` while on the remaining 4, `CART` is better. In terms of expl_{avg} and expl_{wc}, our algorithm is clearly better. Only for 2 datasets `CART` had better expl_{avg} than SER-DT and for 7 datasets our algorithm outperforms `CART` by more than 20%. Regarding expl_{wc} our algorithm was worst for only 3 datasets, and for 4 the improvement was more than 20% compared to `CART`.

For depth-related metrics, we noted more balanced values between the two algorithms when comparing to Table 3 (without post-pruning). As shown in Table 14, considering depth_{avg} our algorithm is better than `CART` for 12, worst for 6 and equal for 2 datasets. For 3 datasets the difference is more than 20%; for 2 of these SER-DT is better while for the remaining one, `CART` is the winner. In terms of depth_{wc} we observe close values between the algorithms: only for the dataset bankruptcy the difference is more than 20%. In this sense, we observe that dataset bankruptcy was an outlier considering explainability metrics.

Table 12: depth_{avg} and depth_{wc} for SER-DT with $\texttt{FactorExpl} = 0.97$, with and without post-pruning. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set. We bold-faced the values (columns 2, 3, 4 and 5) that represent an improvement of more than 20% in terms of explainability.

Dataset	depth_{avg}		depth_{wc}	
	No pruning	Pruning	No pruning	Pruning
anuran	5.38	4.44	6.00	6.00
audit risk	1.00	1.00	1.00	1.00
avila	5.14	4.99	6.00	6.00
banknote	4.39	3.69	6.00	5.70
bankruptcy polish	4.45	**2.01**	6.00	5.30
cardiotocography	4.98	4.12	6.00	6.00
collins	5.89	**4.38**	6.00	5.90
default credit card	2.15	1.76	6.00	5.90
dry bean	4.76	4.40	6.00	6.00
eeg eye state	5.15	4.85	6.00	6.00
htru2	2.80	**2.12**	6.00	5.80
iris	2.50	**1.74**	4.90	**2.30**
letter recognition	5.96	5.77	6.00	6.00
mice	3.05	3.05	3.60	3.60
obs network	4.47	4.31	6.00	6.00
occupancy room	4.72	**3.52**	6.00	6.00
online shoppers intention	3.89	**2.19**	6.00	5.90
pen digits	5.73	5.46	6.00	6.00
poker hand	4.61	4.56	6.00	6.00
sensorless	5.26	5.08	6.00	6.00
Average	4.31	3.67	5.58	5.37

Table 13: Test Accuracy, expl_{avg} and expl_{wc} for SER-DT (`FactorExpl` $= 0.97$) and `CART`, both with post-pruning. Each entry is the average of 10 runs using different seeds to select the examples in the training and testing set. We bold-faced the values that represent improvements of at least 1% in accuracies (columns 2 and 3) and increase of more than 20% in terms of explainability (columns 4, 5, 6 and 7).

Dataset	Test Accuracy		expl_{avg}		expl_{wc}	
	`CART`	SER-DT	`CART`	SER-DT	`CART`	SER-DT
anuran	94.3%	94.5%	4.58	4.07	6.00	6.00
audit risk	99.9%	99.9%	1.00	1.00	1.00	1.00
avila	**62.7%**	60.9%	4.17	**2.98**	5.40	4.90
banknote	98.2%	97.8%	2.51	2.37	3.20	3.40
bankruptcy polish	97.5%	97.3%	**1.15**	1.45	**2.50**	4.50
cardiotocography	89.4%	89.4%	4.02	3.76	5.40	5.60
collins	**14.5%**	13.1%	3.94	**1.40**	5.40	**3.30**
default credit card	82.1%	82.0%	2.38	**1.29**	5.10	**3.90**
dry bean	89.7%	89.9%	3.89	3.17	5.80	4.70
eeg eye state	72.9%	73.6%	4.18	3.56	5.80	5.80
htru2	97.8%	97.8%	1.15	1.09	3.70	3.70
iris	93.7%	93.2%	1.54	1.44	1.80	1.70
letter recognition	**47.9%**	44.4%	5.37	**3.21**	6.00	5.30
mice	100.0%	100.0%	3.05	3.05	3.60	3.60
obs network	88.4%	**91.6%**	4.09	3.35	5.80	5.30
occupancy room	99.3%	99.3%	3.17	3.22	4.90	4.70
online shoppers intention	89.8%	89.8%	2.49	**1.81**	5.50	**4.00**
pen digits	86.1%	**88.3%**	4.98	4.56	6.00	5.40
poker hand	**55.0%**	52.9%	4.28	**1.78**	5.10	**3.80**
sensorless	80.1%	**87.4%**	3.85	**2.81**	5.40	4.80
Average	82.0%	82.1%	3.29	2.57	4.67	4.27

Table 14: depth_{avg} and depth_{wc} for SER-DT (`FactorExpl` $= 0.97$) and `CART`, both with post-pruning. We bold-faced the values that represent an improvement of more than 20% in terms of explainability.

Dataset	depth_{avg}		depth_{wc}	
	`CART`	SER-DT	`CART`	SER-DT
anuran	4.72	4.44	6.00	6.00
audit risk	1.00	1.00	1.00	1.00
avila	5.19	4.99	6.00	6.00
banknote	3.74	3.69	6.00	5.70
bankruptcy polish	**1.17**	2.01	**2.60**	5.30
cardiotocography	4.19	4.12	5.70	6.00
collins	4.95	4.38	6.00	5.90
default credit card	2.39	**1.76**	5.10	5.90
dry bean	4.55	4.40	6.00	6.00
eeg eye state	5.22	4.85	6.00	6.00
htru2	2.52	2.12	5.80	5.80
iris	1.71	1.74	2.10	2.30
letter recognition	5.80	5.77	6.00	6.00
mice	3.05	3.05	3.60	3.60
obs network	4.19	4.31	6.00	6.00
occupancy room	3.29	3.52	5.70	6.00
online shoppers intention	2.75	**2.19**	6.00	5.90
pen digits	5.41	5.46	6.00	6.00
poker hand	4.28	4.56	5.20	6.00
sensorless	5.12	5.08	6.00	6.00
Average	3.76	3.67	5.14	5.37