Born-again Tree Ensembles

Thibaut Vidal [1] **Toni Pacheco** [1] **Maximilian Schiffer** [2]

Abstract

The use of machine learning algorithms in finance, medicine, and criminal justice can deeply impact human lives. As a consequence, research into interpretable machine learning has rapidly grown in an attempt to better control and fix possible sources of mistakes and biases. Tree ensembles offer a good prediction quality in various domains, but the concurrent use of multiple trees reduces the interpretability of the ensemble. Against this background, we study born-again tree ensembles, i.e., the process of constructing a single decision tree of minimum size that reproduces the exact same behavior as a given tree ensemble. To find such a tree, we develop a dynamic-programming based algorithm that exploits sophisticated pruning and bounding rules to reduce the number of recursive calls. This algorithm generates optimal born-again trees for many datasets of practical interest, leading to classifiers which are typically simpler and more interpretable without any other form of compromise.

1. Introduction

Tree ensembles constitute a core technique for prediction and classification tasks. Random forests (Breiman, 2001) and boosted trees (Friedman, 2001) are used in various application fields, e.g., in medicine for recurrence risk prediction and image classification, in criminal justice for custody decisions, or in finance for credit risk evaluation. Although tree ensembles offer a high prediction quality, distorted predictions in high-stakes decisions can be exceedingly harmful. Here, interpretable machine learning models are essential to understand potential distortions and biases. Research in this domain has significantly increased (Murdoch et al., 2019) with numerous works focusing on the construction of optimal sparse trees (Hu et al., 2019) or on the interpretability of neural networks (Zhang et al., 2018; Melis & Jaakkola, 2018).

Currently, there exists a trade-off between the interpretability and the performance of tree (ensemble) classifiers. Single decision trees (e.g., those produced by CART) are well-known for their interpretability, whereas tree ensembles and gradient boosting approaches allow for high prediction quality but are generally more opaque and redundant. Against this background, we study born-again tree ensembles in a similar notion as born-again trees (see, Breiman & Shang, 1996), and search for a simpler classifier that faithfully reproduces the behavior of a tree ensemble.

Formally, let $(\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i, y_i\}_{i=1}^n$ be a training set in which each $\mathbf{x}_i \in \mathbb{R}^p$ is a p-dimensional numerical feature vector, and each $y_i \in \mathbb{N}$ is its associated class. Each sample of this training set has been independently drawn from an unknown distribution $(\mathcal{X}, \mathcal{Y})$. Based on this training set, a tree ensemble $\mathcal{T}$ learns a function $F_{\mathcal{T}} : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts y_i for each $\mathbf{x}_i$ drawn from $\mathcal{X}$. With these notations, we state Problem 1, which is the core of our studies.

Problem 1 (Born-again tree ensemble) *Given a tree ensemble $\mathcal{T}$, we search for a decision tree T of minimal* size *such that $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^p$.*

We note that the condition $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ applies to the entire feature space. Indeed, our goal is to faithfully reproduce the behavior of the tree ensemble for all possible inputs in $\mathcal{X}$. In other words, we are looking for a *new representation of the same classifier*. Problem 1 depends on the definition of a *size* metric. In this study, we refer to the size of a tree either as its *depth* (D) or *number of leaves* (L). Additionally, we study a hierarchical objective (DL) which optimizes depth in priority and then the number of leaves. For the sake of brevity, we detail the methodology for the *depth* objective (D) in the main paper. The supplementary material contains the algorithmic adaptations needed to cover the other objectives, rigorous proofs for all theorems as well as additional illustrations and experimental results.

Theorem 1 states the computational complexity of Problem 1.

Theorem 1 *Problem 1 is NP-hard when optimizing depth,*

[1]Department of Computer Science, PUC Rio, Rio de Janeiro, Brazil [2]TUM School of Management, Technical University of Munich, 80333 Munich, Germany. Correspondence to: Thibaut Vidal <vidalt@inf.puc-rio.br>.

number of leaves, or any hierarchy of these two objectives.

This result uses a direct reduction from 3-SAT. In this work, we show that despite its NP-hardness, Problem 1 can be solved *to proven optimality* for various datasets of practical interest, and that the solution of this problem permits significant advances regarding tree ensembles simplification, interpretation, and analysis.

1.1. State of the art

Our work relates to the field of interpretable machine learning, especially thinning tree ensembles and optimal decision tree construction. We review these fields concisely and refer to Guidotti et al. (2018), Murdoch et al. (2019) and Rudin (2019) for surveys and discussions on interpretable machine learning, as well as to Rokach (2016) for an overview on general work on decision forests.

Thinning tree ensembles has been studied from different perspectives and divides in two different streams, *i*) classical thinning of a tree ensemble by removing some weak learners from the original ensemble and *ii*) replacing a tree ensemble by a simpler classifier, e.g., a single decision tree.

Early works on thinning focused on finding reduced ensembles which yield a prediction quality comparable to the full ensemble (Margineantu & Dietterich, 1997). Finding such reduced ensembles has been proven to be NP-hard (Tamon & Xiang, 2000) and in some cases reduced ensembles may even outperform the full ensemble (Zhou et al., 2002). While early works proposed a static thinning, dynamic thinning algorithms that store the full ensemble but dynamically query only a subset of the trees have been investigated by Hernández-Lobato et al. (2009), Park & Furnkranz (2012), and Martínez-Muñoz et al. (2008). For a detailed discussion on this stream of research we refer to Rokach (2016), who discusses the development of ranking-based methods (see, e.g., Prodromidis et al., 1999; Caruana et al., 2004; Banfield et al., 2005; Hu et al., 2007; Partalas et al., 2010; Rokach, 2009; Zhang & Wang, 2009) and search-based methods (see, e.g., Prodromidis & Stolfo, 2001; Windeatt & Ardeshir, 2001; Zhou et al., 2002; Zhou & Tang, 2003; Rokach et al., 2006; Zhang et al., 2006).

In their seminal work about born-again trees, Breiman & Shang (1996) were the first to introduce a thinning problem that aimed at replacing a tree ensemble by a newly constructed simpler classifier. Here, they used a tree ensemble to create a data set which is then used to build a born-again tree with a prediction accuracy close to the accuracy of the tree ensemble. Ensuing work followed three different concepts. Meinshausen (2010) introduced the concept of *node harvesting*, i.e., reducing the number of decision nodes to generate an interpretable tree. Recent works along this line used tree space prototypes to sparsen a tree (Tan et al., 2016) or rectified decision trees that use hard and soft labels (Bai et al., 2019). Friedman & Popescu (2008) followed a different concept and proposed a linear model to extract rules from a tree ensemble, which can then be used to rebuilt a single tree. Similarly, Sirikulviriya & Sinthupinyo (2011) focused on deducing rules from a random forest, while Hara & Hayashi (2016) focused on rule extraction from tree ensembles via bayesian model selection, and Mollas et al. (2019) used a local-based, path-oriented similarity metric to select rules from a tree ensemble. Recently, some works focused on directly extracting a single tree from a tree ensemble based on stabilized but yet heuristic splitting criteria (Zhou & Hooker, 2016), genetic algorithms (Vandewiele et al., 2017), or by actively sampling training points (Bastani et al., 2017a;b). All of these works focus on the creation of sparse decision trees that remain interpretable but can be used to replace a tree ensemble while securing a similar prediction performance. However, all these approaches offer only an approximative solution, such that the replacement of the tree ensemble by a new classifier can change its prediction performance.

In the field of neural networks, related studies were done on *model compression* (Buciluǎ et al., 2006). The proposed approaches often use knowledge distillation, i.e., using a high-capacity teacher to train a compact student with similar knowledge (see, e.g., Hinton et al., 2015). Recent works focused on creating soft decision trees from a neural network (Frosst & Hinton, 2017), decomposing the gradient in knowledge distillation (Furlanello et al., 2018), deriving a class of models for self-explanatory neural networks (Melis & Jaakkola, 2018), or specified knowledge representations in high conv-layers for interpretable convolutional neural networks (Zhang et al., 2018). Focusing on feed-forward neural networks, Frankle & Carbin (2018) proposed pruning techniques that identify subnetworks which perform close to the original network. Clark et al. (2019) studied born-again multi task networks for natural language processing, while Kisamori & Yamazaki (2019) focused on synthesizing an interpretable simulation model from a neural network. As neural networks are highly non-linear and even less transparent than tree ensembles, all of these approaches remain predominantly heuristic.

Constructing optimal trees. Since the 1990's, some works focused on constructing decision trees based on mathematical programming techniques. Bennett (1992) used linear programming to construct trees with linear combination splits and showed that this technique performs better than conventional univariate split algorithms. Bennett & Blue (1996) focused on building global optimal decision trees to avoid overfitting, while Nijssen & Fromont (2007) presented an exact algorithm to build a decision tree for specific depth, accuracy, and leaf requirements. Recently,

Bertsimas & Dunn (2017) presented a mixed integer programming formulation to construct optimal classification trees. On a similar note, Günlük et al. (2018) presented an integer programming approach for optimal decision trees with categorical data, and Verwer & Zhang (2019) presented a binary linear program for optimal decision trees. Hu et al. (2019) presented a scalable algorithm for optimal sparse binary decision trees. While all these works show that decision trees are in general amenable to be built with optimization techniques, none of these works focused on constructing born-again trees that match the accuracy of a given tree ensemble.

Summary. Thinning problems have been studied for both tree ensembles and neural networks in order to derive interpretable classifiers that show a similar performance than the aforementioned algorithms. However, all of these works embed heuristic construction techniques or an approximative objective, such that the resulting classifiers do not guarantee a behavior and prediction performance equal to the original tree ensemble or neural network. These approaches appear to be plausible for born-again neural networks, as neural networks have highly non-linear structures that cannot be easily captured in an optimization approach. In contrast, work in the field of building optimal decision trees showed that the construction of decision trees is generally amenable for optimization based approaches. Nevertheless, these works focused so far on constructing sparse or optimal trees that outperform heuristically created trees, such that the question whether one could construct an optimal decision tree that serves as a born-again tree ensemble remains open. Answering this question and discussing some of its implications is the focus of our study.

1.2. Contributions

With this work, we revive the concept of born-again tree ensembles and aim to construct a single tree that exactly represents the original tree ensemble. Specifically, our contribution is fourfold. First, we formally define the problem of constructing optimal born-again tree ensembles and prove that this problem is NP-hard. Second, we highlight several properties of this problem and of the resulting born-again tree. These findings allow us to develop a dynamic-programing based algorithm that solves this NP-hard problem efficiently and constructs an optimal born-again tree out of a tree ensemble. Third, we discuss specific pruning strategies for the born-again tree that allow to reduce redundancies that cannot be identified in the original tree ensemble. Fourth, besides providing theoretical guarantees, we present numerical studies which allow to analyze the characteristics of the born-again trees in terms of interpretability and accuracy. Further, these studies show that our algorithm is amenable to a wide range of real-world data sets.

We believe that our results and the developed algorithms open a new perspective in the field of interpretable machine learning. With this approach, one can construct simple classifiers that bear all characteristics of a tree ensemble. Besides interpretability gains, this approach casts a new light on tree ensembles and highlights new structural properties.

2. Fundamentals

In this section, we introduce some fundamental definitions. Afterwards, we discuss a worst-case bound on the depth of an optimal born-again tree.

Tree ensemble. We define a tree ensemble $\mathcal{T}$ as a set of trees $t \in \mathcal{T}$ with weights w_t. For any sample $\mathbf{x}$, the tree ensemble returns the majority vote of its trees: $F_{\mathcal{T}}(\mathbf{x}) = \text{WEIGHTED-MAJORITY}\{(F_t(\mathbf{x}), w_t)\}_{t\in\mathcal{T}}$.[1]

Cells. Let H_j be the set of all split levels (i.e., hyperplanes) extracted from the trees for each feature j. We can partition the feature space $\mathbb{R}^p$ into cells $\mathcal{S}_{\text{ELEM}} = \{1, \dots, |H_1|+1\} \times \cdots \times \{1, \dots, |H_p|+1\}$ such that each cell $\mathbf{z} = (z_1, \dots, z_p) \in \mathcal{S}_{\text{ELEM}}$ represents the box contained between the $(z_j - 1)^{\text{th}}$ and z_j^{th} hyperplanes for each feature $j \in \{1, \dots, p\}$. Cells such that $z_j = 1$ (or $z_j = |H_j| + 1$) extend from $-\infty$ (or to ∞, respectively) along dimension j. We note that the tree-ensemble prediction $F_{\mathcal{T}}(\mathbf{z})$ is constant in the interior of each cell $\mathbf{z}$, allowing us to exclusively use the hyperplanes of $\{H_j\}_{j=1}^d$ to construct an optimal born-again tree.

Regions. We define a *region* of the feature space as a pair $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) \in \mathcal{S}^2_{\text{ELEM}}$ such that $\mathbf{z}^{\text{L}} \leq \mathbf{z}^{\text{R}}$. Region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ encloses all cells $\mathbf{z}$ such that $\mathbf{z}^{\text{L}} \leq \mathbf{z} \leq \mathbf{z}^{\text{R}}$. Let $\mathcal{S}_{\text{REGIONS}}$ be the set of all regions. An optimal born-again tree T for a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is a tree of minimal size such that $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ within this region.

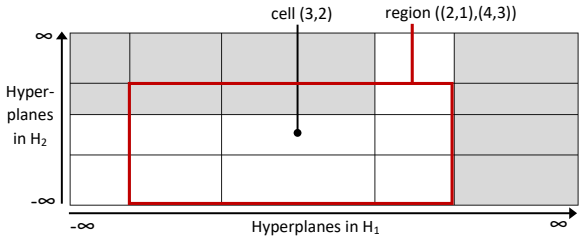

Figure 1. Simple example of a feature space, a cell and a region.

Figure 1 depicts a cell and a region on a two-dimensional feature space. A more extensive example can be found in the supplementary material. The number of cells and regions increases rapidly with the number of hyperplanes

[1] In case of a tie, the smallest index is returned

and features, formally:

$$|\mathcal{S}_{\text{ELEM}}| = \prod_{j=1}^{p}(|H_j|+1) \tag{1}$$

$$|\mathcal{S}_{\text{REGION}}| = \prod_{j=1}^{p}\frac{|H_j|(|H_j|+1)}{2}. \tag{2}$$

Moreover, Theorem 2 gives initial bounds on the size of the born-again decision tree.

Theorem 2 *The depth of an optimal born-again tree T satisfies $\Phi(T) \leq \sum_{t\in\mathcal{T}}\Phi(t)$, where $\Phi(t)$ represents the depth of a tree t. This bound is tight.*

This bound corresponds to a worst case behavior which is usually attained only on purposely-designed pathological cases. As highlighted in our computational experiments, the average tree depths remains generally lower than this analytical worst case. Beyond interpretability benefits, the tree depth represents the number of sequential operations (hyperplane comparisons) needed to determine the class of a given sample during the test stage. Therefore, an optimized born-again tree is not only more interpretable, but it also requires less test effort, with useful applications for classification in embarked systems, typically occurring within limited time and processing budgets.

3. Methodology

In this section, we introduce a dynamic programming (DP) algorithm which optimally solves Problem 1 for many data sets of practical interest. Let $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ be the depth of an optimal born-again decision tree for a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) \in \mathcal{S}_{\text{REGION}}$. We can limit the search to optimal born-again trees whose left and right sub-trees represent optimal born-again trees for the respective sub-regions. Hence, we can recursively decompose a larger problem into subproblems using

$$\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \tag{3}$$
$$\begin{cases} 0 \text{ if } \textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) \\ \min\limits_{1\leq j\leq p}\left\{\min\limits_{z_j^{\text{L}}\leq l<z_j^{\text{R}}}\left\{1+\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\}\right\}\right\}, \end{cases}$$

in which $\textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ takes value TRUE if and only if all cells $\mathbf{z}$ such that $\mathbf{z}^{\text{L}} \leq \mathbf{z} \leq \mathbf{z}^{\text{R}}$ admit the same weighted majority class. In this equation, $\mathbf{z}^{\text{R}}_{jl} = \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}})$ represents the "top right" corner of the left region obtained in the subdivision, and $\mathbf{z}^{\text{L}}_{jl} = \mathbf{z}^{\text{L}} + \mathbf{e}_j(l+1-z_j^{\text{L}})$ is the "bottom left" corner of the right region obtained in the subdivision.

While Equation (3) bears the main rationale of our algorithm, it suffers in its basic state from two main weaknesses that prevent its translation into an efficient algorithm: firstly, each verification of the first condition (i.e., the base case) requires evaluating whether $\textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is true and possibly requires the evaluation of the majority class on an exponential number of cells *if done brute force*. Secondly, the recursive call considers all possible hyperplanes within the region to find the minimum over $j \in \{1, \ldots, p\}$ and $z_j^{\text{L}} \leq l < z_j^{\text{R}}$. In the following, we propose strategies to mitigate both drawbacks.

To avoid the evaluation of $\textsc{id}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ by inspection, we integrate this evaluation within the recursion to profit from the memory structures of the DP algorithm. With these changes, the recursion becomes:

$$\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \tag{4}$$
$$\min_j\left\{\min_{z_j^{\text{L}}\leq l<z_j^{\text{R}}}\{\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\}\}\right\}$$
$$\text{where } \mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \begin{cases} 0 & \text{if } \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) = 0 \\ & \text{and } F_{\mathcal{T}}(\mathbf{z}^{\text{L}}) = F_{\mathcal{T}}(\mathbf{z}^{\text{R}}); \\ 1 & \text{otherwise.} \end{cases}$$

To limit the number of recursive calls, we can filter out for each dimension j any hyperplane $l \in \{1, \ldots, |H_j|\}$ such that $F_{\mathcal{T}}(\mathbf{z}) = F_{\mathcal{T}}(\mathbf{z} + \mathbf{e}_j)$ for all $\mathbf{z}$ such that $\mathbf{z}_j = l$, and exploit two additional properties of the problem.

Theorem 3 *Let $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ and $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$ be two regions such that $\mathbf{z}^{\text{L}} \leq \bar{\mathbf{z}}^{\text{L}} \leq \bar{\mathbf{z}}^{\text{R}} \leq \mathbf{z}^{\text{R}}$, then $\Phi(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$.*

Theorem 3 follows from the fact that any feasible tree satisfying $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ on a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ also satisfies this condition for any subregion $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$. Therefore, $\Phi(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$ constitutes a lower bound of $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$. Combining this bound with Equation (4), we get

$$\begin{aligned} &\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \\ &\quad\leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) \\ &\quad\leq \mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \end{aligned}$$

for each $j \in \{1, \ldots, p\}$ and $z_j^{\text{L}} \leq l < z_j^{\text{R}}$.

This result will be fundamental to use bounding techniques and therefore save numerous recursions during the DP algorithm. With Theorem 4, we can further reduce the number of candidates in each recursion.

Theorem 4 *Let $j \in \{1, \ldots, p\}$ and $l \in \{z_j^{\text{L}}, \ldots, z_j^{\text{R}} - 1\}$.*

- *If $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$ then $\forall l' > l$*

$$\begin{aligned} &\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \\ &\leq \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\} \end{aligned}$$

- *If* $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$ *then* $\forall l' < l$

$$\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\}$$
$$\leq \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Based on Theorem 4, we can discard all hyperplane levels $l' > l$ in Equation (4) if $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$. The same argument holds when $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$ with $l' < l$. We note that the two cases of Theorem 4 are not mutually exclusive. No other recursive call is needed for the considered feature when an equality occurs. Otherwise, at least one case holds, allowing us to search the range $l \in \{z^{\text{L}}_j, \ldots, z^{\text{R}}_j - 1\}$ in Equation (4) by binary search with only $O(\log(z^{\text{R}}_j - z^{\text{L}}_j))$ subproblem calls.

General algorithm structure. The DP algorithm presented in Algorithm 1 capitalizes upon all the aforementioned properties. It is initially launched on the region representing the complete feature space, by calling BORN-AGAIN($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$) with $\mathbf{z}^{\text{L}} = (1, \ldots, 1)^\intercal$ and $\mathbf{z}^{\text{R}} = (|H_1| + 1, \ldots, |H_p| + 1)^\intercal$.

Firstly, the algorithm checks whether it attained a base case in which the region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is restricted to a single cell (Line 1). If this is the case, it returns an optimal depth of zero corresponding to a single leaf, otherwise it tests whether the result of the current subproblem defined by region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is not yet in the DP memory (Line 2). If this is the case, it directly returns the known result.

Past these conditions, the algorithm starts enumerating possible splits and opening recursions to find the minimum of Equation (4). By Theorem 4 and the related discussions, it can use a binary search for each feature to save many possible evaluations (Lines 9–10). By Theorem 3, the exploitation of lower and upper bounds on the optimal solution value (Lines 7,9,19, and 20) allows to stop the iterative search whenever no improving solution can exist. Finally, the special case of Lines 13 and 14 covers the case in which $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) = 0$ and $F_{\mathcal{T}}(\mathbf{z}^{\text{L}}) = F_{\mathcal{T}}(\mathbf{z}^{\text{R}})$, corresponding to a homogeneous region, in which all cells have the same majority class. As usual in DP approaches, our algorithm memorizes the solutions of sub-problems and reuses them in future calls (Lines 15,16, and 25).

We observe that this algorithm maintains the optimal solution of each subproblem in memory, but not the solution itself in order to reduce memory consumption. Retrieving the solution after completing the DP can be done with a simple inspection of the final states and solutions, as detailed in the supplementary material.

The maximum number of possible regions is $|\mathcal{S}_{\text{REGION}}| = \prod_j |H_j| \frac{|H_j|+1}{2}$ (Equation 2) and each call to BORN-AGAIN takes up to $O(\sum_j \log |H_j|)$ elementary operations due to Theorem 4, leading to a worst-case complexity of $O(|\mathcal{S}_{\text{REGION}}| \sum_j \log |H_j|)$ time for the overall recursive algorithm. Such an exponential complexity is expectable for an NP-hard problem. Still, as observed in our experiments, the number of regions explored with the bounding strategies is much smaller in practice than the theoretical worst case.

Algorithm 1 BORN-AGAIN($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$)

```
 1: if (z^L = z^R) return 0
 2: if (z^L, z^R) exists in memory then
 3:    return MEMORY(z^L, z^R)
 4: end if
 5: UB ← ∞
 6: LB ← 0
 7: for j = 1 to p and LB < UB do
 8:    (LOW, UP) ← (z^L_j, z^R_j)
 9:    while LOW < UP and LB < UB do
10:       l ← ⌊(LOW + UP)/2⌋
11:       Φ_1 ← BORN-AGAIN(z^L, z^R + e_j(l − z^R_j))
12:       Φ_2 ← BORN-AGAIN(z^L + e_j(l + 1 − z^L_j), z^R)
13:       if (Φ_1 = 0) and (Φ_2 = 0) then
14:          if f(z^L, T) = f(z^R, T) then
15:             MEMORIZE((z^L, z^R), 0) and return 0
16:             MEMORIZE((z^L, z^R), 1) and return 1
17:          end if
18:       end if
19:       UB ← min{UB, 1 + max{Φ_1, Φ_2}}
20:       LB ← max{LB, max{Φ_1, Φ_2}}
21:       if (Φ_1 ≥ Φ_2) then UP ← l
22:       if (Φ_1 ≤ Φ_2) then LOW ← l + 1
23:    end while
24: end for
25: MEMORIZE((z^L, z^R), UB) and return UB
```

4. Computational Experiments

The goal of our computational experiments is threefold:

1. Evaluating the computational performance of the proposed DP algorithm as a function of the data set characteristics, e.g., the size metric in use, the number of trees in the original ensemble, and the number of samples and features in the datasets.
2. Studying the structure and complexity of the born-again trees for different size metrics.
3. Measuring the impact of a simple pruning strategy applied on the resulting born-again trees.

The DP algorithm was implemented in C++ and compiled with GCC 9.2.0 using flag -O3, whereas the original random forests were generated in Python (using scikit-learn v0.22.1). All our experiments were run on a single thread of an Intel(R) Xeon(R) CPU E5-2620v4 2.10GHz, with 128GB of available RAM, running CentOS v7.7. In the remainder of this section, we discuss the preparation of the data and then

describe each experiment. Detailed computational results, data, and source codes are available in the supplementary material.

4.1. Data preparation

We focus on a set of six datasets from the UCI machine learning repository [UCI] and from previous work by Smith et al. (1988) [SmithEtAl] and Hu et al. (2019) [HuEtAl] for which using random forests (with ten trees) showed a significant improvement upon stand-alone CART. The characteristics of these datasets are summarized in Table 1: number of samples n, number of features p, number of classes K and class distribution CD. To obtain discrete numerical features, we used one-hot encoding on categorical data and binned continuous features into ten ordinal scales. Then, we generated training and test samples for all data sets using a ten-fold cross validation. Finally, for each fold and each dataset, we generated a random forest composed of ten trees with a maximum depth of three (i.e., eight leaves at most), considering $p/2$ random candidate features at each split. This random forest constitutes the input to our DP algorithm.

Table 1. Characteristics of the data sets

Data set	n	p	K	CD	Src.
BC: Breast-Cancer	683	9	2	65-35	UCI
CP: COMPAS	6907	12	2	54-46	HuEtAl
FI: FICO	10459	17	2	52-48	HuEtAl
HT: HTRU2	17898	8	2	91-9	UCI
PD: Pima-Diabetes	768	8	2	65-35	SmithEtAl
SE: Seeds	210	7	3	33-33-33	UCI

4.2. Computational effort

In a first analysis, we evaluate the computational time of Algorithm 1 for different data sets and size metrics. Figure 2 reports the results of this experiment as a box-whisker plot, in which each box corresponds to ten runs (one for each fold) and the whiskers extend to 1.5 times the interquartile range. Any sample beyond this limit is reported as outlier and noted with a "$\circ$". D denotes a depth-minimization objective, whereas L refers to the minimization of the number of leaves, and DL refers to the hierarchical objective which prioritizes the smallest depth, and then the smallest number of leaves. As can be seen, constructing a born-again tree with objective D yields significantly lower computational times compared to using objectives L and DL. Indeed, the binary search technique resulting from Theorem 4 only applies to objective D, leading to a reduced number of recursive calls in this case compared to the other algorithms.

In our second analysis, we focus on the FICO case and randomly extract subsets of samples and features to produce smaller data sets. We then measure the computational effort of Algorithm 1 for metric D (depth optimization) as a function of the number of features ($p \in \{2, 3, 5, 7, 10, 12, 15, 17\}$), the number of samples ($n \in \{250, 500, 750, 1000, 2500, 5000, 7500, 10459\}$), and the number of trees in the original random forest (T $\in \{3, 5, 7, 10, 12, 15, 17, 20\}$). Figure 3 reports the results of this experiment. Each boxplot corresponds to ten runs, one for each fold.

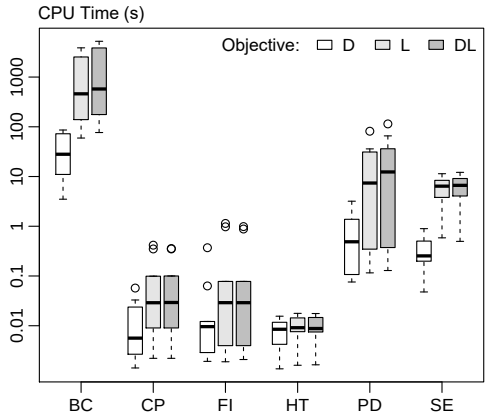

Figure 2. Computational times to construct a born-again tree from a random forest with 10 trees and depth 3, for each objective (D/L/DL) and data set

We observe that the computational time of the DP algorithm is strongly driven by the number of features, with an exponential growth relative to this parameter. This result is in line with the complexity analysis of Section 3. The number of trees influences the computational time significantly less. Surprisingly, the computational effort of the algorithm actually decreases with the number of samples. This is due to the fact that with more sample information, the decisions of the individual trees of the random forest are less varied, leading to fewer distinct hyperplanes and therefore to fewer possible states in the DP.

4.3. Complexity of the born-again trees

We now analyze the depth and number of leaves of the born-again trees for different objective functions and datasets in Table 2.

Table 2. Depth and number of leaves of the born-again trees

	D		L		DL	
Data set	Depth	# Leaves	Depth	# Leaves	Depth	# Leaves
BC	12.5	2279.4	18.0	890.1	12.5	1042.3
CP	8.9	119.9	8.9	37.1	8.9	37.1
FI	8.6	71.3	8.6	39.2	8.6	39.2
HT	6.0	20.2	6.3	11.9	6.0	12.0
PD	9.6	460.1	15.0	169.7	9.6	206.7
SE	10.2	450.9	13.8	214.6	10.2	261.0
Avg.	9.3	567.0	11.8	227.1	9.3	266.4

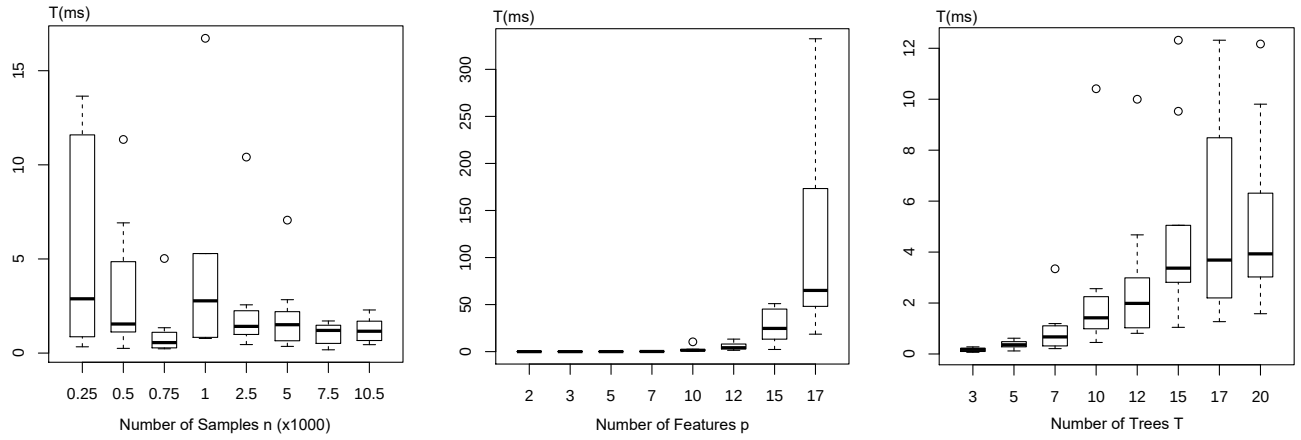

Figure 3. Growth of the computational time (in milliseconds) of Algorithm 1 as a function of the number of samples, features and trees. In each experiment, the parameters which are not under scrutiny are fixed to their baseline values of $n = 2.5 \times 10^3$, $p = 10$ and $T = 10$.

As can be seen, the different objectives can significantly influence the outcome of the algorithm. For several data sets, the optimal depth of the born-again tree is reached with any objective, as an indirect consequence of the minimization of the number of leaves. In other cases, however, prioritizing the minimization of the number of leaves may generate 50% deeper trees for some data sets (e.g., PD). The hierarchical objective DL succeeds in combining the benefits of both objectives. It generates a tree with minimum depth and with a number of leaves which is usually close to the optimal one from objective L.

4.4. Post-pruned born-again trees

Per definition, the born-again tree reproduces the same exact behavior as the majority class of the original ensemble classifier *on all regions* of the feature space $\mathcal{X}$. Yet, some regions of $\mathcal{X}$ may not contain any training sample, either due to data scarcity or simply due to impossible feature combinations (e.g., "gender = MALE" and "pregnant = TRUE"). These regions may also have non-homogeneous majority classes from the tree ensemble viewpoint due to the *combinations* of decisions from multiple trees. The born-again tree, however, is agnostic to this situation and imitates the original classification within all the regions, leading to some splits which are mere artifacts of the ensemble's behavior but never useful for classification.

To circumvent this issue, we suggest to apply a simple post-pruning step to eliminate inexpressive tree sub-regions. We therefore verify, from bottom to top, whether both sides of each split contain at least one training sample. Any split which does not fulfill this condition is pruned and replaced by the child node of the branch that contains samples. The complete generation process, from the original random forest to the *pruned* born-again tree is illustrated in Figure 4. In this simple example, it is noteworthy that the born-again tree uses an optimal split at the root node which is different from all root splits in the ensemble. We also clearly observe the role of the post-pruning step, which contributes to eliminate a significant part of the tree.

To observe the impact of the post-pruning on a larger range of datasets, Table 3 reports the total number of leaves of the random forests, as well as the average depth and number of leaves of the born-again trees before and after post-pruning. As previously, the results are averaged over the ten folds. As can be seen, post-pruning significantly reduces the size of the born-again trees, leading to a final number of leaves which is, on average, smaller than the total number of leaves in the original tree ensemble. This indicates a significant gain of simplicity and interpretability.

Table 3. Comparison of depth and number of leaves

	Random Forest	Born-Again		BA + Pruning	
Data set	# Leaves	Depth	# Leaves	Depth	# Leaves
BC	61.1	12.5	2279.4	9.1	35.9
CP	46.7	8.9	119.9	7.0	31.2
FI	47.3	8.6	71.3	6.5	15.8
HT	42.6	6.0	20.2	5.1	13.2
PD	53.7	9.6	460.1	9.4	79.0
SE	55.7	10.2	450.9	7.5	21.5
Avg.	51.2	9.3	567.0	7.4	32.8

However, post-pruning could cause a difference of behavior between the original tree ensemble classifier and the final pruned born-again tree. To evaluate whether this filtering had any significant impact on the classification performance of the born-again tree, we finally compare the out-of-sample accuracy (Acc.) and F1 score of the three classifiers in Table 4.

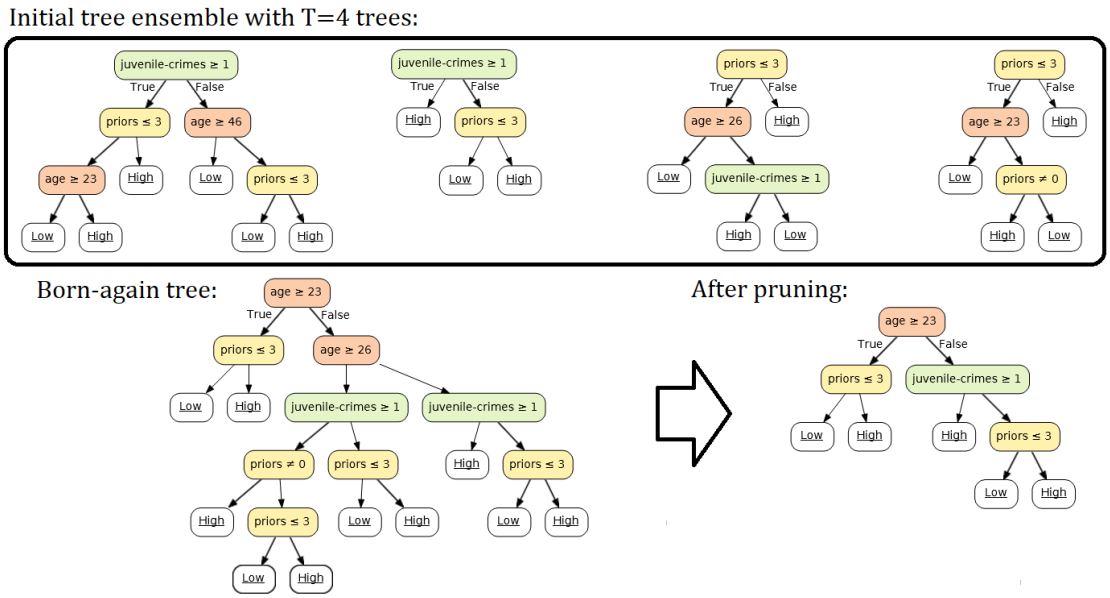

Figure 4. Example of a post-pruned born-again tree on a simple data set

Table 4. Accuracy and F1 score comparison

	Random Forest		Born-Again		BA + Pruning	
Data set	Acc.	F1	Acc.	F1	Acc.	F1
BC	0.953	0.949	0.953	0.949	0.946	0.941
CP	0.660	0.650	0.660	0.650	0.660	0.650
FI	0.697	0.690	0.697	0.690	0.697	0.690
HT	0.977	0.909	0.977	0.909	0.977	0.909
PD	0.746	0.692	0.746	0.692	0.750	0.700
SE	0.790	0.479	0.790	0.479	0.790	0.481
Avg.	0.804	0.728	0.804	0.728	0.803	0.729

First of all, the results of Table 4 confirm the faithfulness of our algorithm, as they verify that the prediction quality of the random forests and the born-again tree ensembles are identical. This was expected *per definition* of Problem 1. Furthermore, only marginal differences were observed between the out-of-sample performance of the born-again tree with pruning and the other classifiers. For the considered datasets, pruning contributed to eliminate inexpressive regions of the tree without much impact on classification performance.

5. Conclusions

In this paper, we introduced an efficient algorithm to summarize a random forest into a single smallest-possible decision tree. Our algorithm is optimal, and provably returns a single tree with the same behavior as the original tree ensemble in the entire sample space. In brief, we obtain a different representation of the same classifier, which helps us to analyze random forests from a different angle. Interestingly, when investigating the structure of the results, we observed that born-again decision trees contain many inexpressive regions designed to faithfully reproduce the behavior of the original ensemble, but which do not contribute to effectively classify samples. It remains an interesting research question to properly understand the purpose of these regions and their contribution to the generalization capabilities of random forests. In a first simple experiment, we attempted to apply post-pruning on the resulting tree. Based on our experiments on six structurally different datasets, we observed that the resulting pruning does not diminish the quality of the predictions but significantly simplifies the born-again trees. Overall, the final pruned trees represent simple, interpretable, and high-performance classifiers, which we believe can be useful for a variety of application areas.

As a perspective for future work, we recommend to progress on the solution of the born-again tree ensembles problem, proposing new algorithms to effectively handle datasets with a larger number of features. Heuristics and mathematical programming techniques can certainly be exploited to quickly find upper bounds during the search and efficiently discard candidate hyperplanes. Another interesting research line concerns the combination of the dynamic programming algorithm for the construction of the born-again tree with active pruning during construction. This may lead to a different definition of the recursion and to different base-case evaluations. Finally, we recommend to pursue the investigation of the structural properties of tree ensembles in light of this new representation.

References

Bai, J., Li, Y., Li, J., Jiang, Y., and Xia, S. Rectified decision trees: Towards interpretability, compression and empirical soundness. *arXiv preprint arXiv:1903.05965*, 2019.

Banfield, R. E., Hall, L. O., Bowyer, K. W., and Kegelmeyer, W. P. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6(1):49–62, 2005.

Bastani, O., Kim, C., and Bastani, H. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*, 2017a.

Bastani, O., Kim, C., and Bastani, H. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017b.

Bennett, K. Decision tree construction via linear programming. In *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, Illinois*, 1992.

Bennett, K. and Blue, J. Optimal decision trees. Technical report, Rensselaer Polytechnique Institute, 1996.

Bertsimas, D. and Dunn, J. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

Breiman, L. and Shang, N. Born again trees. Technical report, University of California Berkeley, 1996.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In *Proceedings of the twenty-first International Conference on Machine Learning*, pp. 18, 2004.

Clark, K., Luong, M.-T., Khandelwal, U., Manning, C. D., and Le, Q. V. Bam! born-again multi-task networks for natural language understanding. *arXiv preprint arXiv:1907.04829*, 2019.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Friedman, J. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

Friedman, J. H. and Popescu, B. E. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3): 916–954, 2008.

Frosst, N. and Hinton, G. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):1–42, 2018.

Günlük, O., Kalagnanam, J., Menickelly, M., and Scheinberg, K. Optimal decision trees for categorical data via integer programming. *arXiv preprint arXiv:1612.03225*, 2018.

Hara, S. and Hayashi, K. Making tree ensembles interpretable: A bayesian model selection approach. *arXiv preprint arXiv:1606.09066*, 2016.

Hernández-Lobato, D., Martinez-Muoz, G., and Suárez, A. Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):364–369, 2009.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hu, Q., Yu, D., Xie, Z., and Li, X. Eros: Ensemble rough subspaces. *Pattern recognition*, 40(12):3728–3739, 2007.

Hu, X., Rudin, C., and Seltzer, M. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, 2019.

Kisamori, K. and Yamazaki, K. Model bridging: To interpretable simulation model from neural network. *arXiv preprint arXiv:1906.09391*, 2019.

Margineantu, D. and Dietterich, T. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference Machine Learning*, 1997.

Martínez-Muñoz, G., Hernández-Lobato, D., and Suárez, A. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):245–259, 2008.

Meinshausen, N. Node harvest. *The Annals of Applied Statistics*, pp. 2049–2072, 2010.

Melis, D. A. and Jaakkola, T. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Mollas, I., Tsoumakas, G., and Bassiliades, N. Lionforests: Local interpretation of random forests through path selection. *arXiv preprint arXiv:1911.08780*, 2019.

Murdoch, W., Singh, C., Kumbier, K., Abassi-Asl, R., and Yu, B. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592v1*, 2019.

Nijssen, S. and Fromont, E. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

Park, S. and Furnkranz, J. Efficient prediction algorithms for binary decomposition techniques. *Data Mining and Knowledge Discovery*, 24(1):40–77, 2012.

Partalas, I., Tsoumakas, G., and Vlahavas, I. An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, 81(3):257–282, 2010.

Prodromidis, A. L. and Stolfo, S. J. Cost complexity-based pruning of ensemble classifiers. *Knowledge and Information Systems*, 3(4):449–469, 2001.

Prodromidis, A. L., Stolfo, S. J., and Chan, P. K. Effective and efficient pruning of metaclassifiers in a distributed data mining system. *Knowledge Discovery and Data Mining Journal*, 32, 1999.

Rokach, L. Collective-agreement-based pruning of ensembles. *Computational Statistics & Data Analysis*, 53(4): 1015–1026, 2009.

Rokach, L. Decision forest: Twenty years of research. *Information Fusion*, 27:111–125, 2016.

Rokach, L., Maimon, O., and Arbel, R. Selective votinggetting more for less in sensor fusion. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(03): 329–350, 2006.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Sirikulviriya, N. and Sinthupinyo, S. Integration of rules from a random forest. In *International Conference on Information and Electronics Engineering*, volume 6, 2011.

Smith, J., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Applications in Medical Care*, pp. 261–265. IEEE Computer Society Press, 1988.

Tamon, C. and Xiang, J. On the boosting pruning problem. In *Proceedings of the 11th European Conference on Machine Learning*, 2000.

Tan, H. F., Hooker, G., and Wells, M. T. Tree space prototypes: Another look at making tree ensembles interpretable. *arXiv preprint arXiv:1611.07115*, 2016.

Vandewiele, G., Lannoye, K., Janssens, O., Ongenae, F., De Turck, F., and Van Hoecke, S. A genetic algorithm for interpretable model extraction from decision tree ensembles. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017.

Verwer, S. and Zhang, Y. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Windeatt, T. and Ardeshir, G. An empirical comparison of pruning methods for ensemble classifiers. In *International Symposium on Intelligent Data Analysis*, 2001.

Zhang, H. and Wang, M. Search for the smallest random forest. *Statistics and its Interface*, 2(3):381, 2009.

Zhang, Q., Nian Wu, Y., and Zhu, S.-C. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Zhang, Y., Burer, S., and Street, W. N. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7(Jul):1315–1338, 2006.

Zhou, Y. and Hooker, G. Interpreting models via single tree approximation. *arXiv preprint arXiv:1610.09036*, 2016.

Zhou, Z., Wu, J., and Tang, W. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137: 239–263, 2002.

Zhou, Z.-H. and Tang, W. Selective ensemble of decision trees. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, 2003.

Supplementary Material – Proofs

Proof of Theorem 1. We show the NP-hardness of the born-again tree ensemble problem by reduction from 3-SAT. Let P be a propositional logic formula presented in conjunctive normal form with three literals per clause. For example, consider $\mathrm{P} = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$. 3-SAT aims to determine whether there exist literal values $x_i \in \{\textsc{True}, \textsc{False}\}$ in such a way that P is true. From a 3-SAT instance with k clauses and l literals, we construct an instance of the born-again tree ensemble problem with $2k-1$ trees of equal weight as follows:

- As illustrated in Figure 5, the first k trees $(t_1, \ldots, t_k)$ represent the clauses. Each of these trees is complete and has a depth of three, with eight leaves representing the possible combinations of values of the three literals. As a consequence of this construction, seven of the leaves predict class TRUE, and the last leaf predicts class FALSE.
- The last $k-1$ trees contain only a single leaf as root node predicting FALSE.

Finding the optimal born-again decision tree for this input leads to one of the two following outcomes:

- If the born-again decision tree contains only one leaf predicting class FALSE, then 3-SAT for P is FALSE.
- Otherwise 3-SAT for P is TRUE.

Indeed, in the first case, if the born-again tree only contains a single FALSE region (and since it is faithful to the behavior of the original tree ensemble) there exists no input sample for which TRUE represents the majority class for the $2k-1$ trees. As such, the first k trees cannot jointly predict TRUE for any input and 3-SAT is FALSE. In the second case, either the optimal born-again decision tree contains a single leaf (root node) of class TRUE, or it contains multiple leaves among which at least one leaf predicts TRUE (otherwise the born-again tree would not be optimal). In both situations, there exists a sample for which the majority class of the tree ensemble is TRUE and therefore for which all of the first k trees necessarily return TRUE, such that 3-SAT is TRUE. This argument holds for any objective involving the minimization of a monotonic size metric, i.e., a metric for which the size of a tree does not decrease upon addition of a node. This includes in particular, the depth, the number of leaves, and the hierarchical objectives involving these two metrics.

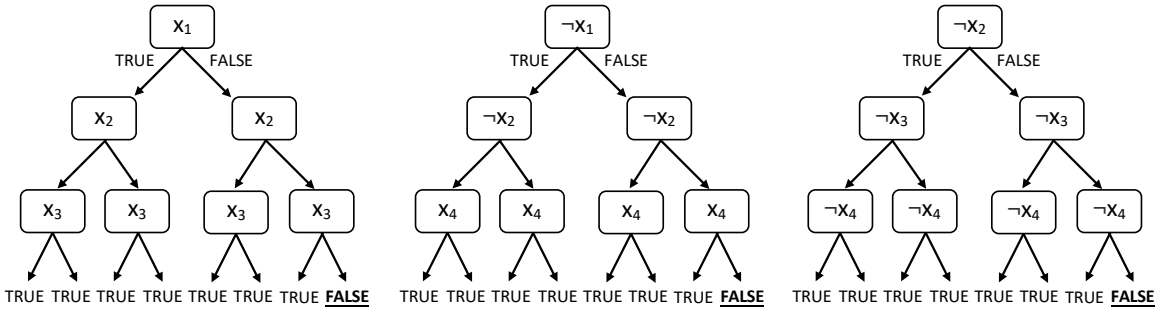

Figure 5. Trees representing the 3-SAT clauses for $\mathrm{P} = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$

Proof of Theorem 2. Consider a tree ensemble $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$. We construct a sequence of decision trees starting with $T_1 = t_1$ by iteratively appending a copy of tree t_k for $k \in \{2, \ldots, |\mathcal{T}|\}$ in place of each leaf of the tree T_{k-1} to form T_k. This leads to a born-again tree $T_{|\mathcal{T}|}$ of depth $\sum_i \Phi(t_i)$. Each leaf of this tree represents a region of the feature space over which the predicted class of the trees $t \in \mathcal{T}$ is constant, such that the ensemble behavior on this region is faithfully represented by a single class. With this construction, tree $T_{|\mathcal{T}|}$ faithfully reproduces the behavior of the original tree ensemble. Since the optimal born-again tree T has a depth no greater than that of $T_{|\mathcal{T}|}$, we conclude that $\Phi(T) \leq \sum_i \Phi(t_i)$.

Moreover, we prove that this bound is tight, i.e., it is attained for a family of tree ensembles with an arbitrary number of trees. To this end, we consider the feature space $\mathcal{X} = \mathbb{R}^d$ and the following $2d-1$ trees with equal weight:

- For $i \in \{1, \ldots, d\}$, tree t_i contains a single internal node representing the split $x_i \leq 0$, leading to a leaf node predicting class 0 when the splitting condition is satisfied, and to a leaf node predicting class 1 otherwise.
- The remaining $d-1$ trees contain a single leaf at the root node predicting class 1.

In the resulting tree ensemble, class 0 represents the majority if and only if $x_i \leq 0$ for all $i \in \{1, \ldots, d\}$. To be faithful to the original tree ensemble, the optimal born-again decision tree must verify that $x_i \leq 0$ for all $i \in \{1, \ldots, d\}$ to declare a sample as part of class 0. This requires at least d comparisons. The depth of the born-again decision tree needed to make these tests is $\Phi(T) = d = \sum_i \Phi(t_i)$.

Proof of Theorem 3. Any tree T satisfying $F_T(\mathbf{x}) = F_{\mathcal{T}}(\mathbf{x})$ on a region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ also satisfies this condition for any subregion $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$. Therefore, every feasible solution (tree) of the born-again tree ensemble problem for region $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ is feasible for the subproblem restricted to $(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$. As a consequence, the optimal solution value $\Phi(\bar{\mathbf{z}}^{\text{L}}, \bar{\mathbf{z}}^{\text{R}})$ for the subproblem is smaller or equal than the optimal solution value $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ of the original problem.

Proof of Theorem 4. We will use the extended notation $\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ to denote $\mathbb{1}_{jl}$. Firstly, we observe that $\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ for all l and l'. Indeed, regardless of l and j,

$$(\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) = 0 \text{ and } F_{\mathcal{T}}(\mathbf{z}^{\text{L}}) = F_{\mathcal{T}}(\mathbf{z}^{\text{R}})) \Leftrightarrow \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) = 0.$$

Next, we observe that $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'})$ and $\Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$ for all $l' > l$ follows from Theorem 3. If $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$, then $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) \geq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})$ follows from the two previous inequalities and:

$$\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} = \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) = \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Analogously, we observe that based on Theorem 3 $\Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})$ and $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}) \leq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl})$ for all $l' < l$ holds. If $\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})$, then $\Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) \geq \Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'})$ follows from the two previous inequalities and:

$$\max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} = \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}}) \leq \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}}) = \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Combining these results with the first observation, we obtain in both cases that:

$$\mathbb{1}_{jl}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl}), \Phi(\mathbf{z}^{\text{L}}_{jl}, \mathbf{z}^{\text{R}})\} \leq \mathbb{1}_{jl'}(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}) + \max\{\Phi(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}_{jl'}), \Phi(\mathbf{z}^{\text{L}}_{jl'}, \mathbf{z}^{\text{R}})\}.$$

Supplementary Material – Pseudo-Codes for Objectives D and DL

Our solution approach is applicable to different tree size metrics, though the binary search argument resulting from Theorem 4 is only applicable for depth minimization. We considered three possible objectives.

- (D) Depth minimization;
- (L) Minimization of the number of leaves;
- (DL) Depth minimization as primary objective, and then number of leaves as a secondary objective.

The dynamic programming algorithm for depth minimization (D) is described in the main body of the paper. Algorithms 1 and 2 detail the implementation of the dynamic programming algorithm for objectives L and DL, respectively. To maintain a similar structure and use the same solution extraction procedure in Section 5, these two algorithms focus on minimizing the number of splits rather than the number of leaves, given that these quantities are proportional and only differ by one unit in a proper binary tree. Moreover, the hierarchical objective DL is transformed into a weighted sum by associating a large cost of M for each depth increment, and 1 for each split. This allows to store each dynamic programming result as a single value and reduces memory usage.

The main differences with the algorithm for objective D occur in the loop of Line 8, which consists for L and DL in an enumeration instead of a binary search. The objective calculations are also naturally different. As seen in Line 19, the new number of splits is calculated as $1 + \Phi_1 + \Phi_2$ for objective L (i.e., the sum of the splits from the subtrees plus one). When using the hierarchical objective DL, we obtain the depth and number of splits from the subproblems as $\lfloor \Phi_i / \text{M} \rfloor$ and $\Phi_i \% \text{M}$, respectively, and use these values to obtain the new objective.

Supplementary Material – Solution Extraction

To reduce computational time and memory consumption, our dynamic programming algorithms only store the optimal objective value of the subproblems. To extract the complete solution, we exploit the following conditions to recursively retrieve the optimal splits from the DP states. For a split to belong to the optimal solution:

1. Both subproblems should exist in the dynamic programming memory.
2. The objective value calculated from the subproblems should match the known optimal value for the considered region.

These conditions lead to Algorithm 3, which reports the optimal tree in DFS order.

Algorithm 2 BORN-AGAIN-L($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$)

if $(\mathbf{z}^{\text{L}} = \mathbf{z}^{\text{R}})$ **return** 0
if $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ exists in memory **then**
  return MEMORY$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$
end if
UB $\leftarrow \infty$
LB $\leftarrow 0$
for $j = 1$ **to** p and LB $<$ UB **do**
  for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ and LB $<$ UB **do**
    $\Phi_1 \leftarrow$ BORN-AGAIN-L$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}))$
    $\Phi_2 \leftarrow$ BORN-AGAIN-L$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}})$
    if $(\Phi_1 = 0)$ and $(\Phi_2 = 0)$ **then**
      if $f(\mathbf{z}^{\text{L}}, \mathcal{T}) = f(\mathbf{z}^{\text{R}}, \mathcal{T})$ **then**
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 0)$ **and return** 0
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 1)$ **and return** 1
      end if
    end if

    UB $\leftarrow \min\{\text{UB}, 1 + \Phi_1 + \Phi_2\}$
    LB $\leftarrow \max\{\text{LB}, \max\{\Phi_1, \Phi_2\}\}$
  end for
end for
MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), \text{UB})$ **and return** UB

Algorithm 3 BORN-AGAIN-DL($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}$)

if $(\mathbf{z}^{\text{L}} = \mathbf{z}^{\text{R}})$ **return** 0
if $(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$ exists in memory **then**
  return MEMORY$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$
end if
UB $\leftarrow \infty$
LB $\leftarrow 0$
for $j = 1$ **to** p and LB $<$ UB **do**
  for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ and LB $<$ UB **do**
    $\Phi_1 \leftarrow$ BORN-AGAIN-DL$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}))$
    $\Phi_2 \leftarrow$ BORN-AGAIN-DL$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}})$

    if $(\Phi_1 = 0)$ and $(\Phi_2 = 0)$ **then**
      if $f(\mathbf{z}^{\text{L}}, \mathcal{T}) = f(\mathbf{z}^{\text{R}}, \mathcal{T})$ **then**
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), 0)$ **and return** 0
        MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}})$,M+1) **and return** M+1

      end if
    end if
    DEPTH $\leftarrow 1 + \max\{\lfloor \Phi_1/\text{M} \rfloor, \lfloor \Phi_2/\text{M} \rfloor\}$
    SPLITS $\leftarrow 1 + \Phi_1 \% \text{M} + \Phi_2 \% \text{M}$
    UB $\leftarrow \min\{\text{UB}, \text{M} \times \text{DEPTH} + \text{SPLITS}\}$
    LB $\leftarrow \max\{\text{LB}, \max\{\Phi_1, \Phi_2\}\}$
  end for
end for
MEMORIZE$((\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}), \text{UB})$ **and return** UB

Algorithm 4 EXTRACT-OPTIMAL-SOLUTION($\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}}, \Phi_{\text{OPT}}$)

if $\Phi_{\text{OPT}} = 0$ **then**
  EXPORT a leaf with class MAJORITY-CLASS$(\mathbf{z}^{\text{L}})$
  return
else
  for $j = 1$ **to** p **do**
    for $l = z_j^{\text{L}}$ **to** $z_j^{\text{R}} - 1$ **do**
      $\Phi_1 \leftarrow$ MEMORY$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}))$
      $\Phi_2 \leftarrow$ MEMORY$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}})$
      if $\Phi_{\text{OPT}} =$ CALCULATE-OBJECTIVE(Φ_1, Φ_2) **then**
        EXTRACT-OPTIMAL-SOLUTION$(\mathbf{z}^{\text{L}}, \mathbf{z}^{\text{R}} + \mathbf{e}_j(l - z_j^{\text{R}}), \Phi_1)$
        EXTRACT-OPTIMAL-SOLUTION$(\mathbf{z}^{\text{L}} + \mathbf{e}_j(l + 1 - z_j^{\text{L}}), \mathbf{z}^{\text{R}}, \Phi_2)$
        EXPORT a split on feature j with level z_j^{L}
        return
      end if
    end for
  end for
end if

Supplementary Material – Detailed Results

In this section, we report additional computational results which did not fit in the main paper due to space limitations. Tables 5 to 7 extend the results of Tables 2 and 3 in the main paper. They report for each objective the depth and number of leaves of the different classifiers, as well as their minimum and maximum values achieved over the ten runs (one for each training/test pair).

Table 5. Complexity of the different classifiers – Considering objective D

	Random Forest			Born Again Tree			Born Again Tree + Pruning								
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	12.5	11	13	2279.4	541	4091	9.1	8	11	35.9	26	44
CP	46.7	40	55	8.9	7	11	119.9	23	347	7.0	4	9	31.2	10	50
FI	47.3	40	52	8.6	3	13	71.3	5	269	6.5	3	9	15.8	4	27
HT	42.6	36	49	6.0	2	7	20.2	3	38	5.1	2	6	13.2	3	22
PD	53.7	45	63	9.6	7	12	460.1	101	1688	9.4	7	12	79.0	53	143
SE	55.7	51	60	10.2	9	11	450.9	159	793	7.5	6	8	21.5	16	31
Overall	51.2	36	68	9.3	2	13	567.0	3	4091	7.4	2	12	32.8	3	143

Table 6. Complexity of the different classifiers – Considering objective L

	Random Forest			Born Again Tree			Born Again Tree + Pruning								
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	18.0	17	20	890.1	321	1717	9.0	7	11	23.1	17	32
CP	46.7	40	55	8.9	7	11	37.1	10	105	6.5	3	8	11.4	4	21
FI	47.3	40	52	8.6	3	13	39.2	4	107	6.3	3	8	12.0	4	20
HT	42.6	36	49	6.3	2	8	11.9	3	19	4.3	2	6	6.4	3	9
PD	53.7	45	63	15.0	12	19	169.7	50	345	11.0	8	17	30.7	20	42
SE	55.7	51	60	13.8	12	16	214.6	60	361	7.7	6	9	14.2	9	19
Overall	51.2	36	68	11.8	2	20	227.1	3	1717	7.5	2	17	16.3	3	42

Table 7. Complexity of the different classifiers – Considering objective DL

	Random Forest			Born Again Tree			Born Again Tree + Pruning								
	#Leaves			Depth			#Leaves			Depth			#Leaves		
Data set	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
BC	61.1	57	68	12.5	11	13	1042.3	386	2067	8.9	8	10	27.7	18	39
CP	46.7	40	55	8.9	7	11	37.1	10	105	6.5	3	8	11.4	4	21
FI	47.3	40	52	8.6	3	13	39.2	4	107	6.3	3	8	12.0	4	20
HT	42.6	36	49	6.0	2	7	12.0	3	19	4.6	2	6	6.5	3	10
PD	53.7	45	63	9.6	7	12	206.7	70	387	8.9	7	11	42.1	28	62
SE	55.7	51	60	10.2	9	11	261.0	65	495	7.4	6	9	17.0	12	24
Overall	51.2	36	68	9.3	2	13	266.4	3	2067	7.1	2	11	19.5	3	62

Finally, Tables 8 to 10 extend the results of Table 4 in the main paper. They report for each objective the average accuracy and F1 scores of the different classifiers, as well as the associated standard deviations over the ten runs on different training/test pairs.

Table 8. Accuracy of the different classifiers – Considering objective D

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.946	0.047	0.941	0.046
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.750	0.067	0.700	0.069
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.790	0.196	0.481	0.208
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.803	0.065	0.729	0.073

Table 9. Accuracy of the different classifiers – Considering objective L

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.943	0.052	0.938	0.053
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.751	0.064	0.698	0.068
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.790	0.193	0.479	0.207
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.803	0.065	0.727	0.074

Table 10. Accuracy of the different classifiers – Considering objective DL

	Random Forest				Born Again Tree				Born Again Tree + Pruning			
	Acc.		F1		Acc.		F1		Acc.		F1	
Data set	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
BC	0.953	0.040	0.949	0.040	0.953	0.040	0.949	0.040	0.941	0.051	0.935	0.049
CP	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024	0.660	0.022	0.650	0.024
FI	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049	0.697	0.049	0.690	0.049
HT	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044	0.977	0.009	0.909	0.044
PD	0.746	0.062	0.692	0.065	0.746	0.062	0.692	0.065	0.747	0.069	0.693	0.076
SE	0.790	0.201	0.479	0.207	0.790	0.201	0.479	0.207	0.781	0.195	0.477	0.210
Avg.	0.804	0.064	0.728	0.072	0.804	0.064	0.728	0.072	0.801	0.066	0.726	0.075

Supplementary Material – Born-Again Tree Illustration

Finally, Figure 6 illustrates the born-again tree ensemble problem on a simple example with an original ensemble composed of three trees. All cells and the corresponding majority classes are represented. There are two classes, depicted by a ● and a ○ sign, respectively.

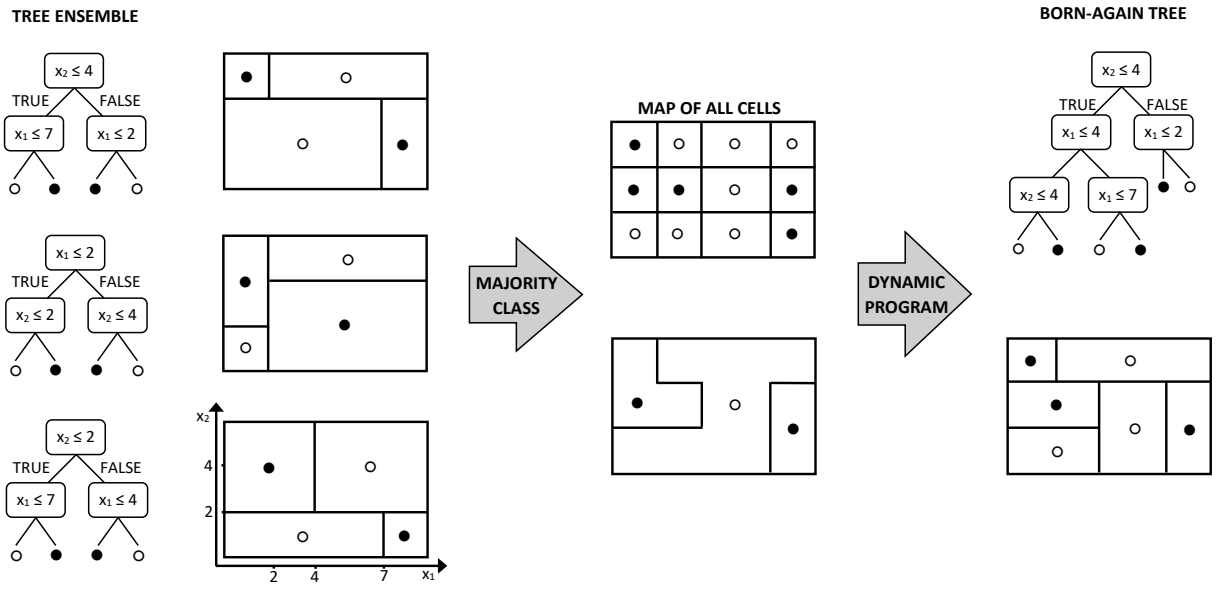

Figure 6. Cells and born-again tree on a simple example