Matheus de Sousa Suknaic

Approximate Born Again tree ensembles

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor : Prof. Marco Serpa Molinaro
Co-advisor: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
August 2021

Matheus de Sousa Suknaic

Approximate Born Again tree ensembles

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Rio de Janeiro, August 13th, 2021

Matheus de Sousa Suknaic

Obtained a bachelor in Computer Engineering (2019) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio).

Bibliographic data

Suknaic, Matheus de Sousa

Approximate Born Again tree ensembles / Matheus de Sousa Suknaic; advisor: Marco Serpa Molinaro; co-advisor: Thibaut Victor Gaston Vidal. – 2021.

63 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Conjuntos de árvores. 2. Árvores de decisão. 3. Machine Learning interpretável. 4. Modelos de compressão de dados. I. Molinaro, Marco. II. Vidal, Thibaut. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Acknowledgments

I would like to first thank my advisor, Marco Molinaro, my co-advisor, Thibaut Vidal, and Maximiliam Schiffer for all the insightful discussions and time/work dedicated.

I also would like to thank my parents for always encouraging me and being in every step with me.

Lastly, I would like to thank my boyfriend and my friends, you always have raised my spirits, even when I thought I could not do it.

This study was financed in part by the Coordenação de Aperfeiçoamento Pessoal de Nível Superior (CAPES) – Finance Code 001, and I am very grateful for this scholarship.

Abstract

Suknaic, Matheus de Sousa; Molinaro, Marco (Advisor); Vidal, Thibaut (Co-Advisor). **Approximate Born Again tree ensembles**. Rio de Janeiro, 2021. 63p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Ensemble methods in machine learning such as random forest, boosting, and bagging have been thoroughly studied and proven to have better accuracy than using a single predictor. However, their drawback is that they give models that can be much harder to interpret than those given by, for example, decision trees. In this work, we approach in a principled way the problem of constructing a decision tree that *approximately* reproduces a tree ensemble, exploring the tradeoff between accuracy and interpretability that can be obtained once exact reproduction is relaxed.

First, we formally define the problem of obtaining the decision tree of a given depth that is most *adherent* to a tree ensemble and give a Dynamic Programming algorithm for solving this problem. We also prove that the decision trees obtained by this procedure satisfy generalization guarantees related to the generalization of the original tree ensembles, a crucial element for their effectiveness in practice. Since the computational complexity of the Dynamic Programming algorithm is exponential in the number of features, we also design heuristics to compute trees of a given depth with good adherence to a tree ensemble.

Finally, we conduct a comprehensive computational evaluation of the algorithms proposed. The results indicate that in many situations, there is little or no loss in accuracy in working more interpretable classifiers: even restricting to only depth-6 decision trees, our algorithms produce trees with average accuracies that are within 1% (for the Dynamic Programming algorithm) or 2% (heuristics) of the original random forest.

Keywords

Tree ensembles; Decision Tree; Interpretable machine learning; Model compression.

Resumo

Suknaic, Matheus de Sousa; Molinaro, Marco; Vidal, Thibaut. **Árvores BA aproximadas**. Rio de Janeiro, 2021. 63p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Métodos ensemble como random forest, boosting e bagging foram extensivamente estudados e provaram ter uma acurácia melhor do que usar apenas um preditor. Entretanto, a desvantagem é que os modelos obtidos utilizando esses métodos podem ser muito mais difíceis de serem interpretados do que por exemplo, uma árvore de decisão. Neste trabalho, nós abordamos o problema de construir uma árvore de decisão que *aproximadamente* reproduza um conjunto de árvores, explorando o tradeoff entre acurácia e interpretabilidade, que pode ser alcançado quando a reprodução exata do conjunto de árvores é relaxada.

Primeiramente, nós formalizamos o problem de obter uma árvore de decisão de uma determinada profundidade que seja a mais *aderente* ao conjunto de árvores e propomos um algoritmo de programação dinâmica para resolver esse problema. Nós também provamos que a árvore de decisão obtida por esse procedimento satisfaz garantias de generalização relacionadas a generalização do modelo original de conjuntos de árvores, um elemento crucial para a efetividade dessa árvore de decisão em prática. Visto que a complexidade computacional do algoritmo de programação dinâmica é exponencial no número de features, nós propomos duas heurísticas para gerar árvores de uma determinada profundidade com boa aderência em relação ao conjunto de árvores.

Por fim, nós conduzimos experimentos computacionais para avaliar os algoritmos propostos. Quando utilizados classificadores mais interpretáveis, os resultados indicam que em diversas situações a perda em acurácia é pequena ou inexistente: restrigindo a árvores de decisão de profundidade 6, nossos algoritmos produzem árvores que em média possuem acurácias que estão a 1% (considerando o algoritmo de programção dinâmica) ou 2% (considerando os algoritmos heurísticos) do conjunto original de árvores.

Palavras-chave

Conjuntos de árvores; Árvores de decisão; Machine Learning interpretável; Modelos de compressão de dados.

Table of contents

List of figures

List of tables

List of Abreviations

DP	Dynamic Programming
BA	Born Again
DT	Decision Tree

1
Introduction

Ensemble methods such as random forest [Breiman, 2001], boosting [Freund and Schapire, 1997, Schapire and Freund, 2012, Breiman, 1997], and bagging [Breiman, 1996] have been thoroughly studied and proven to have better accuracy than using a single predictor. For this reason, they are well-known and widespread, being present in many different machine learning applications. However, the main drawback of ensemble methods is that as the size of the ensemble grows, its complexity makes it harder to understand how and why the model made a specific decision. Indeed, many applications must fulfill safety, privacy, and ethical guidelines. For these cases, interpretability and explainability are as crucial as having a model with strong predictive performance. The area of interpretable and explainable machine learning has attracted much attention recently [Guidotti et al., 2018, Rudin et al., 2021, Carvalho et al., 2019].

A specific case that illustrates the tradeoff between accuracy and interpretability is that of decision trees (ensembles). 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.

Motivated by the idea to understand the behavior of a complex ensemble classifier through a simpler model, [Breiman and Shang, 1996] proposed the construction of a single decision tree to reproduce the behavior of a tree ensemble. Recently, [Vidal and Schiffer, 2020] revisited this idea and formally defined the problem of finding the shortest decision tree that reproduces the decisions of a tree ensemble *exactly over the whole space* $\mathbb{R}^d$. However, requiring exact reproduction is a strong requirement that forces the use of large decision trees, undermining its interpretability.

1.1
Our contributions

In this work, using [Vidal and Schiffer, 2020] as the starting point, we approach in a principled way the problem of constructing a decision tree that only approximately reproduces a tree ensemble, exploring the tradeoff between accuracy and interpretability that can be obtained once exact reproduction is relaxed. We call these *approximate born-again trees*. More precisely, our contributions are the following:

1. First, we formally define the problem of obtaining the decision tree of a given depth that is most adherent to a tree ensemble. The depth is used to enforce interpretability of the resulting BA tree. In order to preserve the generalization properties of tree ensembles, adherence is not defined based on samples but actually over the continuous space $\mathbb{R}^d$, based on the cells induced by the ensemble (see Section 3.1). The general formulation allows the use of any measure of cell-importance. Moreover, we give a Dynamic Programming algorithm for solving this general problem.

2. We prove that for several measures of cell-importance, the approximate BA trees obtained by this procedure satisfy generalization guarantees related to the generalization of the original tree ensembles. The interpretable trees constructed (approximately) inherit the strong generalization properties of tree ensembles, a crucial element for their effectiveness in practice.

3. Since the computational complexity of the Dynamic Programming algorithm is exponential in the number of features and hence not applicable to larger datasets, we design two heuristics to compute approximate BA trees of a given depth with good adherence to a tree ensemble.

4. We conduct a comprehensive computational evaluation of the model and algorithms proposed to better understand the accuracy/interpretability tradeoffs obtained, using random forests as the tree ensemble. Considering the concept of transparency proposed in [Lipton, 2018], we only consider trees ranging from depth one to six since we assume deeper trees to no longer be easily interpretable. The results indicate that in many situations, there is little or no loss in accuracy in working with more interpretable classifiers: even restricting to only depth 6 decision trees, our algorithms produce BA trees with average accuracies that are within 1% (for the Dynamic Programming algorithm) or 2% (heuristics) of the original random forest.

.

1.2 Organization

This work is organized as follows: Chapter 2 focuses on the related work, divided into tree ensemble pruning methods, rule extraction methods, and ensemble representation using a decision tree. Chapter 3 formally defines the problem of obtaining the decision tree of a given depth that is most adherent to a tree ensemble and the cell-importance functions. Chapter 4 describes the dynamic programming and heuristic methods. Chapter 5 proves the statistical guarantees for BA trees created using different cell-importance functions. Chapter 6 discusses the conducted experiments, their results, and our observations. Finally, Chapter 7 expresses our concluding remarks about our findings and discusses possible trajectories for future work.

2
Literature Review

In this section, we review studies related to our work, conducting a systematic literature review. In total, we have retrieved 37 papers related to our study. They can be divided into three main categories: pruning methods for tree ensembles, rule extraction methods, and ensemble representation using a decision tree.

2.1
Tree ensemble pruning methods

Pruning methods were one of the first approaches studied to create more interpretable classifiers (in addition to helping their generalization properties). One of the first works in pruning of tree ensembles was [Margineantu and Dietterich, 1997], which proposed different pruning algorithms for an ensemble produced using AdaBoost.

Following this work, many studies focused on creating a selection strategy to extract a set of the best decision trees to represent the tree ensemble. [Bernard et al., 2009] selected subsets of trees using SFS (Sequential Forward Selection) and SBS (Sequential Backward Selection) that outperform the original tree ensemble. [Latinne et al., 2001] created a procedure based on the McNemar non-parametric test of significance that chooses a combination of a minimal number of classifiers, obtaining a prediction accuracy level similar to the one obtained with the combination of larger ensembles. [Hernández-Lobato et al., 2009] have developed a statistical method on how many classifiers need to be queried in an ensemble to determine the prediction of the complete ensemble with a specified confidence level and, [Meinshausen, 2009] proposed a pruning method for tree ensembles, where nodes are gathered in groups and afterwards is decided which groups should remain and which should be discarded.

Succeeding these works, [Joly et al., 2012] reformulated the tree ensemble-based model as a linear model in terms of node indicator functions and used a L1-norm regularization approach to select a minimal subset of these indicator functions while maintaining predictive accuracy. [Nan et al., 2016] have formulated the pruning problem as a 0-1 integer linear program that incorporates feature-reuse constraints, whereas [Ren et al., 2015] have proposed a global pruning method that merges insignificant leaves through the creation of a mathematical programming model. [Painsky and Rosset, 2018] have created a method to compress a random forest based on probabilistic modeling of the ensemble's trees, followed by model clustering via Bregman divergence. The outcome is a simpler model that is able to retain the completeness and accuracy of the forest.

There have also been tree sub-selection procedures taking diversity into account. For example, [Adnan and Islam, 2016] simultaneously selected a set of accurate and diverse trees through a genetic algorithm, which produces an effective sub-forest. [Zhou and Tang, 2003] also developed a genetic algorithm to decide which of the trees from the original ensemble should remain and which should be discarded, based on their validation error, and [Zouggar and Adla, 2019] created a diversity-based heuristic measure that is used to simplify a random forest ensemble.

Lastly, several other works defined new tree ensemble metrics to be used for pruning. [Yang et al., 2012] proposed four margin metrics and a pruning method for random forests that is formulated as a margin optimization problem, and [Jiang et al., 2017] selected which branches to prune based on a new metric called *branch importance*, which indicates the importance of a branch (or a node) concerning the whole ensemble.

2.2 Rule Extraction methods

Even though rule-based methods are not the goal of our study, they have a similar approach to decision trees: their decisions are based on a set of rules. Therefore, we decide to mention some of the works in this domain, where the main goal is also to provide interpretability to an ensemble classifier.

One of the earliest works was proposed by [Friedman and Popescu, 2008] and focused on extracting rules from an ensemble while retaining an approximate accuracy through reformulating the original model using a regularized linear model. Following this work, [Sirikulviriya and Sinthupinyo, 2011] concentrated on integrating a pair of rules from a random forest incrementally and then have these newly integrated rules replace the original rules. [Deng, 2014] created a framework to extract, measure, prune and select rules from a tree ensemble to compose into a simple learner.

In recent years, [Hara and Hayashi, 2018] have extracted a simple model from a tree ensemble by adopting a probabilistic model representation of the tree ensemble, then reducing it to a bayesian model selection problem, whereas

[Mollas et al., 2020] developed a framework to make a local explanation to a random forest, through the extraction of its rules. This framework reduces the number of features and paths, having as output a set of rules interpreted in natural language.

However, note that unlike our main goal, these previous works do not provide a way of represent tree ensembles using a *single* decision tree.

2.3 Ensemble representation using a single decision tree

The existing works in this direction can be divided into two categories: methods tailored explicitly for tree ensembles and methods used on any ensemble learner.

2.3.1 General methods

Initial studies were proposed by [Craven and Shavlik, 1995] and [Krishnan et al., 1999]. [Craven and Shavlik, 1995] have created a method for extracting comprehensible, symbolic representations from trained neural networks into a decision tree using an oracle. During the same period, [Krishnan et al., 1999] extracted decision trees from input data generated from trained neural networks instead of doing it directly from the data, using a genetic algorithm and prototype selection procedure.

Following these studies, [Prodromidis and Stolfo, 2001] focused on creating a method based on decision tree pruning to prune an ensemble of meta-classifiers and map the ensemble to a decision tree classifier. In contrast, [Boz, 2002] extracted decision trees from trained neural networks by defining a new evaluation for splitting and pruning of the resulting tree in a way to maximize the fidelity between this simpler classifier and the Neural Network.

In recent years, [Frosst and Hinton, 2017] used a neural network to train a decision tree with lower accuracy, however higher interpretability, and [Bai et al., 2020] have proposed a knowledge distillation method to create rectified decision trees that use hard and soft labels from a more complex model. [Yang et al., 2018] developed a method that recursively partitions the input variable space by maximizing the difference in the contribution of input variables averaged from local explanations between these spaces, generating a binary decision tree, whereas [Bastani et al., 2019] have built a tree from a black-box model fitting a mixture of Gaussians for the underlying distribution, actively sampling new samples from it and then proposing a greedy algorithm to construct the classifier. Lastly, [Zhou and Hooker, 2016] proposed a procedure to build a decision tree that approximates the performance of complex machine learning models. They have introduced an improved splitting method designed to stabilize tree structure.

2.3.2 Methods specific to tree ensembles

The three initial studies were conducted by [Breiman and Shang, 1996], [Shannon and Banks, 1999], and [Quinlan, 1999]. [Breiman and Shang, 1996] created more training samples using a tree ensemble, and this new data was used to create a decision tree to represent the ensemble, while [Quinlan, 1999] was interested in first applying boosting on small tree ensembles and afterwards combining all decision trees into one. [Shannon and Banks, 1999] have proposed a distribution on tree structures and a distance metric between two trees of the ensemble. A numerical search is performed using this distance metric to find the maximum likelihood estimate of the central tree parameter. This maximum likelihood estimate of the central tree is proposed as the representation of the tree structure.

Succeeding these initial studies, [Van Assche and Blockeel, 2007] have created a procedure to learn a first-order decision tree that approximates the decisions made by an ensemble of first-order decision trees through candidate tests and heuristics. [Schetinin et al., 2007] used a probabilistic interpretation of Bayesian decision tree ensembles based on the quantitative evaluation of uncertainty of each decision tree, allowing experts to find the tree that provides high predictive accuracy and confident outcomes.

In the last decade, there were many different research directions proposed. [Tan et al., 2020] have used prototypes, representative points of the dataset, that are selected for each class and used to provide a better interpretation of the tree ensemble classifier. [Johansson et al., 2011] have proposed a method to approximate a random forest to a single decision tree, using an oracle coach (a strong classifier), whereas [Vandewiele et al., 2017] constructed a decision tree from a tree ensemble through a genetic algorithm.

From all these previous studies, the only two that have considered the ensemble behavior over the entire feature space were [Quinlan, 1999] and [Vandewiele et al., 2017]. The main drawback in [Quinlan, 1999] is in the fact that only small ensembles of trees (three decision trees) were considered, and the resulting decision tree was already large (a considerable number of leaves), not being interpretable. As for [Vandewiele et al., 2017], the authors do not mention the fidelity nor the depth of the resulting decision trees. Thus, it is not easy to assess how well the resulting model reproduces the ensemble and if it is interpretable.

The most important paper for our work, indeed its starting point, is that of [Vidal and Schiffer, 2020]. There, the authors have considered the task of reproducing the behavior of a tree ensemble *exactly over the whole feature space* using a decision tree, i.e., a decision tree that for all possible data points makes the same classification as the original tree ensemble. They have called these *Born-Again (BA) trees*. They have proposed a dynamic programming algorithm to build the smallest BA tree (in terms of depth or number of leaves). In addition, they have proposed a heuristic that still produces Born-Again trees (i.e., completely faithful to the tree ensemble) that aim to minimize the tree's size. The main disadvantage of these methods is that since they require exact reproduction of the tree ensemble, even the smallest BA tree can be quite large and undermine the desired interpretability.

After considering all proposed works, we have decided to leverage the notion of exact BA trees used in [Vidal and Schiffer, 2020] to obtain in a principled way a more practical decision tree representation of tree ensembles.

3
Problem definition

Broadly speaking, we are interested in supervised classification: There is a p-dimensional feature space $\mathbb{R}^p$ and a set of possible classes $\mathcal{Y}$. Given labeled examples $(x_1, y_1), \dots (x_n, y_n) \in \mathbb{R}^p \times \mathcal{Y}$ sampled from an unknown distribution μ, the overall goal is to learn a (interpretable) classifier $F : \mathbb{R}^p \to \mathcal{Y}$ with small classification error over the whole distribution, i.e. $\Pr_{(X,Y)\sim\mu}(F(X) \neq Y)$.

A *tree ensemble* τ is a collection of decision trees T with weights w_T. The ensemble τ classifies a point $x \in \mathbb{R}^p$ by taking weighted majority vote over the classifications $T(x)$ given by its trees.[1] We use $\tau(x)$ to denote the classification it assigns to point $x \in \mathbb{R}^p$.

3.1 Problem MaxAdherence

Our main goal in this work is to obtain a decision tree T with at most a given depth[2] d that has the closest behavior possible to a given tree ensemble τ. This similarity of behavior should hold over the whole space $\mathbb{R}^p$, and not only labeled examples available in order for T to obtain performance comparable to that of τ on unseen data.

To formally define the notion "closest behavior over the whole $\mathbb{R}^p$", we need the concept of a *cell* of τ. Recall that each node of a decision tree is associated with a hyperplane of the form $\{x \in \mathbb{R}^p : x_j = c\}$, that is, it defines the split of the feature j at value c. Given a tree ensemble τ, let H_j be the set of all hyperplanes for feature j over all nodes of all trees in τ. The union of all these hyperplanes $\bigcup_{j=1}^{p} H_j$ partitions $\mathbb{R}^p$ into *cells*, that is, a set of disjoint (up to measure 0) hypercubes in $\mathbb{R}^p$ (see Figure 3.1). We use $\mathcal{C}(\tau)$ to denote the set of all cells of τ.

The problem MaxAdherence is then that of finding a decision tree of bounded depth that exactly reproduces "most" of the cells of τ. More precisely: *Given a tree ensemble τ, a cell-importance function $v : \mathcal{C}(\tau) \to \mathbb{R}$, and a depth*

[1]In case of tie, when $\mathcal{Y}$ is a subset of $\mathbb{R}$ we assume the ensemble returns the class that is the smallest number.

[2]By depth of a tree, we mean the number of edges on the longest root-to-leaf path. In particular, a tree consisting of a single leaf is said to have depth 0.

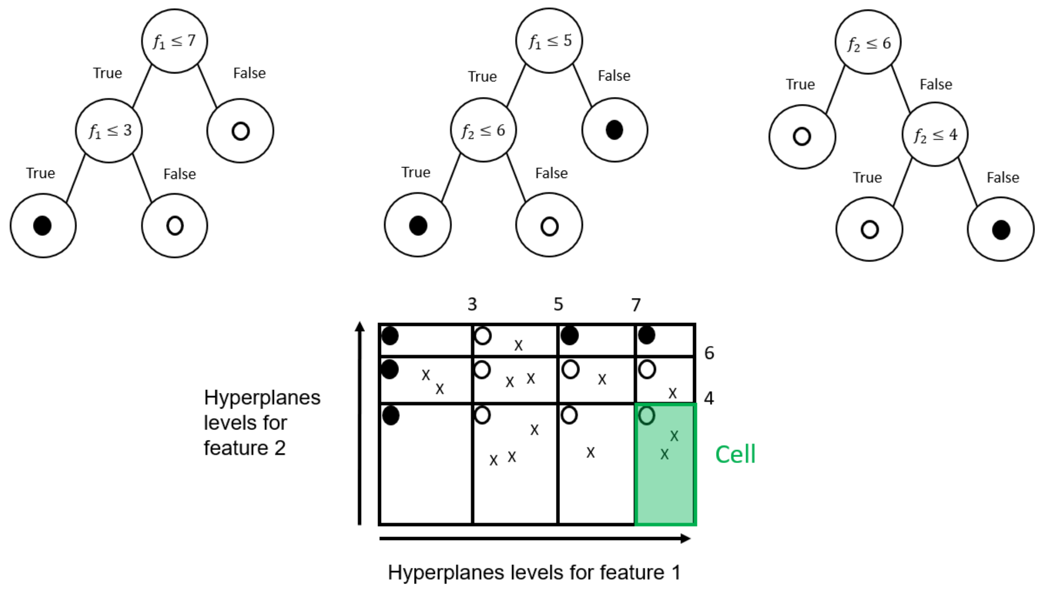

Figure 3.1: Representation of the random forest, the resulting feature space with two features (f_1 and f_2), two classes (represented by the black and white circles), the labeled examples (represented as x's) and a cell. Actually the cells on the first and last rows and columns extend to infinity to cover the remainder of the space.

upper bound $d \in \mathbb{N}$, find the decision tree T that solves

$$\max_T \sum_{C \in \mathcal{C}(\tau)} v(C) \cdot \mathbb{1}(T(C) = \tau(C)) \qquad \text{(MaxAdherence)}$$
$$\text{depth}(T) \le d,$$

where $\mathbb{1}(T(C) = \tau(C))$ is the indicator that T's and τ's classifications agree on **all** points in the cell C. The objective value $\sum_{C \in \mathcal{C}(\tau)} v(C) \cdot \mathbb{1}(T(C) = \tau(C))$ is called the *adherence* of T (w.r.t. τ).

Since the cells of τ may have different importance for the classification task at hand (e.g. there may be a large/small probability $\mu(C)$ of encountering a data point in cell C), MaxAdherence allows this to be taken into account via the function $v(\cdot)$. Even though the true distribution μ of the data is not known, there are several natural ways of assigning importance to these cells.

Example 1 (Volume) *If no sample data is available, one can also set $v(C)$ to be the Lebesgue volume of the cell C. Since cells are hypercubes $\prod_j [a_j, b_j]$, this volume can be computed very efficiently.*

Example 2 (Empirical measure) *If sample data $(x_1, y_1), \dots, (x_n, y_n)$ is available, one can set $v(C)$ to be the empirical measure of the cell C, namely the fraction of point x_i's that lie inside C. This provides an unbiased estimation of the probability $\Pr_{(X,Y)\sim\mu}(X \in C)$ that a (new) example falls in this cell. (In* *fact, one can replace the empirical measure with more robust estimates, such as kernel estimates.)*

Example 3 (Parametric distributions) *If one further assumes that the marginal $\mu|_{\mathbb{R}^p}$ of the distribution μ over the features belongs (or is close to) to a specific family $\mathcal{F}$ of distributions, one can use the available data to obtain an estimate $\hat{\mu}|_{\mathbb{R}^p} \in \mathcal{F}$ of $\mu|_{\mathbb{R}^p}$ and use $v(C) = \hat{\mu}|_{\mathbb{R}^p}(C)$. The advantage is that one can typically obtain stronger statistical guarantees about these estimates $\hat{\mu}|_{\mathbb{R}^p}(C) \approx \mu|_{\mathbb{R}^p}(C)$ over all cells C simultaneously. A canonical choice of family $\mathcal{F}$ is that of a mixture of Gaussians.*

4
Exact and heuristic methods

In this chapter, we first introduce a dynamic programming algorithm used to solve to optimality the problem proposed in Chapter 3. Afterwards, we describe heuristic methods that approximately solve the proposed problem and mitigate the DP algorithm's exponential nature.

4.1
Dynamic Programming Algorithm

Throughout this and the following Section 4.2, fix a tree ensemble τ input for the MaxAdherence Problem. The main observation is that the optimal tree for MaxAdherence only uses the hyperplanes in the tree ensemble τ since one error in a cell makes it lose all its value. Recall that we use $H_j = \{h_j^1, \dots, h_j^{|H_j|}\}$ to denote the set of all hyperplanes for feature j over all nodes of all trees in τ, and that $\mathcal{C}(\tau)$ is the set of cells of τ. For each element h_j in a set H_j, we consider that $h_j^i < h_j^{i+1}$, where $i \in \{1, \dots, |H_j| - 1\}$. To simplify the notation, let $S_{elem} := \prod_j \{1, 2, \dots, |H_j|\}$ denote the set of all vectors that index the hyperplanes of τ.

Consider two vectors $z^L \leq z^R$ in S_{elem}, where the first can be thought as the "lower-left corner" and the second as the "top-right corner". These vectors define a hypercube in $\mathbb{R}^p$, a *region* of cells denoted by (z^L, z^R), see Figure 4.1. More formally, if a cell C is the region enclosed within the parallel hyperplanes $(h_1^{i_1}, h_1^{i_1+1}), \dots, (h_p^{i_p}, h_p^{i_p+1})$, then this cell belongs to the region (z^L, z^R) if and only if the indices satisfy $z_j^L \leq i_j \leq z_j^R$ for all coordinates j.

Given a region (z^L, z^R) and depth limit d, we define the following as the sub-problem of the dynamic programming procedure:

$$\begin{aligned}\mathrm{OPT}(z^L, z^R, d) &= \text{maximum adherence to } \tau \text{ obtainable on region } (z^L, z^R) \\ &\quad\ \text{with a decision tree of depth at most } d \\ &= \max_T \left\{ \sum_{C \in (z^L, z^R)} v(C) \cdot \mathbb{1}(T(C) = \tau(C)) \ :\ \mathrm{depth}(T) \leq d \right\}\end{aligned}$$

When the depth limit is $d = 0$, that is, only decision trees consisting of a single leaf are allowed, $\mathrm{OPT}(z^L, z^R, 0)$ is obtained by classifying all points in the region (z^L, z^R) as with the class that has highest total importance in the region, namely

$$\mathrm{OPT}(z^L, z^R, 0) = \max_{y \in \mathcal{Y}} \sum_{C \in (z^L, z^R): \tau(C) = y} v(C).$$

When $d > 0$, i.e., can consider decision trees that branch at the root node, the optimal value adherence is obtained by testing all features j and cell-breakpoints ℓ that yield a non-trivial branching and computing the value on the left and right branches recursively (see Figure 4.1):

$$\mathrm{OPT}(z^L, z^R, d) = \begin{cases} v(z^L, z^R) & \text{, if } z^L = z^R \\ \max\limits_j \max\limits_{z_j^L \le \ell < z_j^R} \Big(\mathrm{OPT}\big(z^L, (z^R)^{j \to \ell}, d-1\big) + \mathrm{OPT}\big((z^L)^{j \to \ell}, z^R, d-1\big)\Big) & \text{, if } z^L \neq z^R \end{cases}$$

where for a vector $u \in \mathbb{R}^p$ we use $u^{j \to \ell}$ to denote the vector obtained by changing the j-th coordinate of u to value ℓ. Then using $z_{\text{first}} = (1, \ldots, 1)$ and $z_{\text{last}} = (|H_1|, \ldots, |H_p|)$ to denote the "first" and "last" cells of the ensemble τ, we see that $\mathrm{OPT}(z_{\text{first}}, z_{\text{last}}, d)$ is the solution to the MaxAdherence problem, namely the decision tree of depth at most d with maximum adherence to τ.

Given the above recurrence relations between the subproblems, the optimal adherence $\mathrm{OPT}(z_{\text{first}}, z_{\text{last}}, d)$ can then be computed recursively using standard Dynamic Programming techniques.

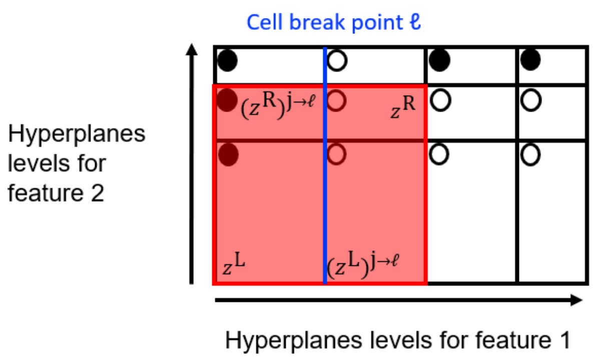

Figure 4.1: Representation of a feature space with two features and the division of the region (z^L, z^R) into two subregions with the selected feature $j = 1$ and cell-breakpoint ℓ.

Notice that the number of regions (z^L, z^R) is $\prod_{j \in [p]} \binom{|H_j|}{2} = \Omega(\prod_j |H_j|^2)$, and so we have at least these many potential subproblems $\mathrm{OPT}(z^L, z^R, d)$. Hence the worst-case complexity of the algorithm is exponential in the number of features p and deteriorates as the number of hyperplanes present in the ensemble τ increases. We remark, however, that by using a recursive implementation of Dynamic Programming, not necessarily all subproblems will be evaluated, which leads to less extreme running times in practice. Nonetheless, this exact algorithm is prohibitively slow on larger datasets. For this reason, in the next section we propose heuristics for approximating the optimal value $\mathrm{OPT}(z^L, z^R, d)$.

4.2 Heuristic methods

We propose two heuristic methods in this section: beam search and greedy. These two heuristics significantly reduce the number of evaluated recursions by approximately evaluating each region, selecting the best cell break-points, and only branching on them, thus mitigating the DP exponential nature.

4.2.1 Beam search and greedy heuristics

Similarly as we have done in the previous section 4.1, given a region (z^L, z^R) and a depth limit d, we define the following as the sub-problem of the beam search heuristic procedure:

$$\begin{aligned}\mathrm{Proxy}(z^L, z^R, d) = {} & \text{maximum approximate adherence to } \tau \text{ obtainable on region } (z^L, z^R) \\ & \text{with a decision tree of depth at most } d\end{aligned}$$

However, instead of branching in all possible cell break-points as described in 4.1, the beam search heuristic only does for a set of K cell break-points. Therefore, this heuristic calculates an approximate adherence instead of the optimal adherence for the region (z^L, z^R) and depth d. The greedy heuristic is a particular case of the beam search, where we set $K = 1$.

For $d = 0$, we have that $\mathrm{Proxy}(z^L, z^R, 0)$ is obtained by classifying all points in the region (z^L, z^R) as with the class that has highest total importance in the region, thus

$$\mathrm{Proxy}(z^L, z^R, 0) = \max_{y \in \mathcal{Y}} \sum_{C \in (z^L, z^R) : \tau(C) = y} v(C),$$

which is the same definition proposed for $\mathrm{OPT}(z^L, z^R, 0)$.

When $d > 0$, for each feature j, it is selected the cell-break point ℓ that maximizes the sum of the values of the two sub-regions: $(z^L, (z^R)^{j \to \ell})$ and

$((z^L)^{j\to\ell}, z^R)$; assuming that all points in these sub-regions are classified with the class that has highest total importance in the region, namely:

$$\max_{z_j^L \le \ell < z_j^R} \Big(\mathrm{Proxy}\big(z^L, (z^R)^{j\to\ell}, 0\big) + \mathrm{Proxy}\big((z^L)^{j\to\ell}, z^R, 0\big)\Big), \text{for } j \in [p]$$

From these best j cell-break points, the K which yield the highest values are chosen and the optimal value is computed on the left and right branches recursively:

$$\mathrm{Proxy}(z^L, z^R, d) = \begin{cases} v(z^L, z^R) & \text{, if } z^L = z^R \\ \max_{\ell \in B} \big(\mathrm{Proxy}(z^L, (z^R)^{j\to\ell}, d-1) + \mathrm{Proxy}((z^L)^{j\to\ell}, z^R, d-1)\big) & \text{, if } z^L \neq z^R \end{cases}$$

where B represents the set with the best K cell-break points. Formally, the algorithm that implements this procedure is the following:

Algorithm 1 Beam Search

```
 1: Input: Region $(z^L, z^R)$ and depth $d$.
 2: Output: Maximum approximate adherence to $\tau$ obtainable on region $(z^L, z^R)$ with a decision
    tree of depth at most $d$.
 3: if $z^L = z^R$ then
 4:     return $z^L$ value
 5: end if
 6: if $(z^L, z^R, d)$ exists in memory then
 7:     return Proxy$(z^L, z^R, d)$
 8: end if
 9: if $d = 0$ then
10:     Proxy$(z^L, z^R, 0) \leftarrow$ Value of the class that has highest total importance in $(z^L, z^R)$
11:     Store $(z^L, z^R, 0)$ and Proxy$(z^L, z^R, 0)$ in memory
12: end if
13: for $j \in \{1, ..., p\}$ do
14:     for $\ell \in \{z_j^L \le \ell < z_j^R\}$ do
15:         $values_{j\ell} \leftarrow$ Proxy$(z^L, (z^R)^{j\to\ell}, 0) +$ Proxy$((z^L)^{j\to\ell}, z^R, 0)$
16:     end for
17: end for
18: $B \leftarrow$ The $K$ hyperplanes that have the highest values in $values$
19: for $\ell \in B$ do
20:     $best_\ell \leftarrow$ Proxy$\big(z^L, (z^R)^{j\to\ell}, d-1\big) +$ Proxy$\big((z^L)^{j\to\ell}, z^R, d-1\big)$
21: end for
22: Proxy$(z^L, z^R, d) \leftarrow$ Highest value from $best$
23: Store $(z^L, z^R, d)$ and Proxy$(z^L, z^R, d)$ in memory
24: return Proxy$(z^L, z^R, d)$
```

5
Statistical guarantees

In this chapter, we prove the statistical guarantees for BA trees created using different cell-importance functions.

For that, we use the standard statistical learning setup: There is an unknown distribution μ over $\mathbb{R}^d \times \mathcal{Y}$ (labeled examples), and there are independent samples $(X_1, Y_1), \ldots, (X_m, Y_m) \sim \mu$ available for building a classification model $M : \mathbb{R}^d \to \mathcal{Y}$. We are interested in models that have small population error, namely

$$\mathrm{err}(M) := \Pr_{(X,Y)\sim\mu}(M(X) \neq Y).$$

To make precise the idea of "tree T constructed based on a tree ensemble τ, we say that T is *subordinated to* τ if for every cell $C \in \mathcal{C}(\tau)$ the $T(\cdot)$ is the constant function, i.e., it gives the same classification to all points in C. Notice that all trees produced by the DP procedure from Chapter 4 satisfy this property. Given such trees, define its adherence error with respect to the cell-importance function v as:

$$\mathrm{diff}_v(T, \tau) := \sum_{C \in \mathcal{C}(\tau) : T(C) \neq \tau(C)} v(C).$$

Notice that our problem MaxAdherence is precisely that of finding a tree of depth at most d that minimizes $\mathrm{diff}_v(T, \tau)$.

5.1
Guarantee for empirical measure

Given samples $(X_1, Y_1), \ldots, (X_m, Y_m)$ from the true distribution μ, let $\hat{\mu}(A) := \frac{1}{m} \sum_i \mathbb{1}(X_i \in A)$ for all $A \subseteq \mathbb{R}^d$ denote its empirical measure. As indicated in Example 2, we can use the empirical measure to compute a "good" decision tree T from a tree ensemble τ as follows:

1. Obtain samples $(X_1, Y_1), \ldots, (X_m, Y_m)$ from μ.

2. Use the empirical measure $\hat{\mu}$ as the cell-importance function, namely set $v(C) = \hat{\mu}(C)$ for every cell $C \in \mathcal{C}(\tau)$, and solve the MaxAdherence to

obtain the tree T with largest adherence to τ w.r.t. $\hat{\mu}$ (or equivalently, with minimum $\mathrm{diff}_{\hat{\mu}}(T,\tau)$).

The next result shows that the tree T constructed in this way has generalization error that can be compared to that of τ, the bound depending on how "far" (in terms of $\mathrm{diff}_{\hat{\mu}}$) τ is from trees of depth d.

Theorem 5.1 *Consider a tree ensemble τ, m samples $(X_1,Y_1),\dots,(X_m,Y_m) \sim \mu$, and the empirical importance function $\hat{\mu} : \mathcal{C}(\tau) \to \mathbb{R}$ defined above. Let T be a (random) decision tree subordinated to τ constructed using τ and the samples (X_i,Y_i). Then with probability at least $1-\delta$*[1]

$$err(T) \;\le\; err(\tau) + diff_{\hat{\mu}}(T,\tau) + \frac{\sum_C \sqrt{\mu(C)(1-\mu(C))}}{\sqrt{m}} + \sqrt{\frac{2\log 1/\delta}{m}},$$

where the expectations is with respect to the samples $Y_1,\dots,Y_m$ defining $\hat{\mu}$.

As we show in our computational experiments, in practice the adherence error $\mathrm{diff}_{\hat{\mu}}(T,\tau)$ is not large for tree T of reasonable depth, see Section 6.3.1.

To prove Theorem 5.1 we need McDiarmid's inequality.

Lemma 1 (Theorem 6.2 of [Boucheron et al., 2013]) *Let $f : \mathcal{Z} \to \mathbb{R}$ be a function such that*

$$f(z_1,\dots,z_m) - f(z_1,\dots,z_{i-1},z_i',z_{i+1},\dots,z_m) \le 1$$

for all i and all $z_i, z_i' \in \mathcal{Z}$. Then, if $Z_1,\dots,Z_m$ are independent random variables taking values in $\mathcal{Z}$, we have

$$\Pr\Big(f(Z_1,\dots,Z_m) > \mathbb{E}f(Z_1,\dots,Z_m) + \lambda\Big) \;\le\; e^{\frac{\lambda^2}{2m}} \qquad \forall \lambda \ge 0.$$

Proof. Let $\Delta(T,\tau) := \Pr_{(X,Y)\sim\mu}(T(X) \neq \tau(X))$ be the probability that T and τ have different classifications. The population error of T can be upper bounded using the error of τ plus this probability: for every scenario of the samples $X_1,\dots,X_m$,

$$\mathrm{err}(T) = \Pr_{(X,Y)\sim\mu}(T(X)\neq Y) \le \Pr_{(X,Y)\sim\mu}(\tau(X)\neq Y) + \Pr_{(X,Y)\sim\mu}(T(X)\neq\tau(X)) = \mathrm{err}(\tau) + \Delta(T,\tau). \tag{5-1}$$

To simplify the notation let $\mathcal{C} = \mathcal{C}(\tau)$. It will also be convenient to see the discrete distributions $(\mu(C))_C$ and $(\hat{\mu}(C))_C$ as vectors in $\mathbb{R}^{\mathcal{C}}$, namely the

[1]The quantity $\sum_C \sqrt{\mu(C)(1-\mu(C))}$ is known as the *Bhattacharyya coefficient* between the distributions $(\mu(C))_C$ and $(1-\mu(C))_C$, which (by Cauchy-Schwarz) is at most $\sqrt{|\mathcal{C}(\tau)|-1}$.

vectors $p := (\mu(C))_{C\in\mathcal{C}}$ and $\hat{p} := \frac{1}{m}\sum_i u(X_i)$ where $u(x) := (\mathbb{1}(x \in C))_{C\in\mathcal{C}}$. Then $\Delta(T,\tau)$ can be upper bounded

$$\Delta(T,\tau) = \sum_{C\in\mathcal{C}:T(C)\neq\tau(C)} \mu(C) \;\leq\; \underbrace{\sum_{C\in\mathcal{C}:T(C)\neq\tau(C)} \hat{\mu}(C)}_{\mathrm{diff}_{\hat{\mu}}(T,\tau)} \;+\; \|p-\hat{p}\|_1. \tag{5-2}$$

Finally, we upper bound the last term with high probability. Its expectation is

$$\mathbb{E}\,\|p-\hat{p}\|_1 = \sum_{C\in\mathcal{C}} \mathbb{E}|p_C - \hat{p}_C| \leq \sum_{C\in\mathcal{C}} \sqrt{\mathbb{E}(p_C-\hat{p}_C)^2} = \sum_{C\in\mathcal{C}} \sqrt{\tfrac{p_C(1-p_C)}{m}}, \tag{5-3}$$

where the inequality follows from concavity of the function $\sqrt{x}$ and Jensen's inequality, and the last equation is because $m\cdot\hat{p}_C$ follows a binomial distribution with m trials and success probability p_C and so its variance $m^2\cdot\mathbb{E}(p_C-\hat{p}_C)^2$ equals $m\cdot p_C\cdot(1-p_C)$.

Moreover, define the function f given by

$$f(x_1,\ldots,x_m) := \|p - \tfrac{1}{m}\textstyle\sum_i u(x_i)\|_1,$$

so that $f(X_1,\ldots,X_m) = \|p-\hat{p}\|_1$. The function f satisfies the bounded-difference condition: by triangle inequality

$$\begin{aligned} f(x_1,\ldots,x_m) &\leq \|p - \tfrac{1}{m}\textstyle\sum_{j\neq i} u(x_j)\|_1 + \|\tfrac{1}{m}u(x_i)\|_1 \quad \text{and} \\ f(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_m) &\geq \|p - \tfrac{1}{m}\textstyle\sum_{j\neq i} u(x_j)\|_1 - \|\tfrac{1}{m}u(x_i')\|_1 \end{aligned}$$

and hence

$$f(x_1,\ldots,x_m) - f(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_m) \leq \frac{2}{m} \qquad \forall i, x_i, x_i'.$$

Then applying McDiarmid's Inequality (Lemma 1) to $f(X_1,\ldots,X_m)$ gives

$$\Pr\left(\|p-\hat{p}\|_1 \;\geq\; \mathbb{E}\,\|p-\hat{p}\|_1 + \sqrt{\tfrac{2\log 1/\delta}{m}} \right) \;\leq\; \delta.$$

Combining this inequality with inequalities (5-1), (5-2), and (5-3) concludes the proof of the Theorem 5.1 ■

5.2 Guarantee for learnable distributions

Given two continuous distributions ν, ν' recall that their $\mathcal{L}_1$-distance is

$$\|\nu - \nu'\|_{\mathcal{L}_1} := \int |pdf_\nu(x) - pdf_{\nu'}(x)| \, \mathrm{d}x.$$

Definition 5.2 (**$(\alpha, \varepsilon, \delta)$-learnability**) *A family $\mathcal{D}$ of continuous distributions is* $(\alpha, \varepsilon, \delta)$-learnable with m samples *if there is a procedure that does the following: for any continuous distribution ν (possibly outside of $\mathcal{D}$), given m samples $X_1, X_2, ..., X_m$ from ν it outputs a distribution $\hat{\nu} \in \mathcal{D}$ such that with probability at least $1 - \delta$*

$$\|\hat{\nu} - \nu\|_{\mathcal{L}_1} \;\leq\; \alpha \cdot \min_{\nu' \in \mathcal{D}} \|\nu' - \nu\|_{\mathcal{L}_1} + \varepsilon.$$

We call such a procedure an $(\alpha, \varepsilon, \delta)$-learner for $\mathcal{D}$.

As an example of learnable distributions we have the family of mixtures of k Gaussians in $\mathbb{R}^d$, which has been recently proved to be $(12, \varepsilon, \delta)$-learned with $O(\frac{kd^2}{\epsilon^2} \cdot \text{polylog}(\frac{kd}{\varepsilon\delta}))$ samples [Ashtiani et al., 2020].

This notion suggests the following way of computing a "good" decision tree T from a tree ensemble τ:

1. Obtain samples $(X_1, Y_1), \ldots, (X_m, Y_m)$ from the true distribution μ
2. Choose a learnable distribution $\mathcal{D}$ and use the samples to obtain a distribution $\hat{\mu} \in \mathcal{D}$ that is approximately closest μ
3. Use $\hat{\mu}$ as the cell-importance function, namely set $v(C) = \hat{\mu}(C)$ for every cell $C \in \mathcal{C}(\tau)$, solve the MaxAdherence to obtain the tree T with largest adherence to τ w.r.t. $\hat{\mu}$ (or equivalently, with minimum $\text{diff}_{\hat{\mu}}(T, \tau)$).

The next bound shows that the generalization power of the tree T computed in this way can be compared to that τ, depending on how far the true distribution μ is to the set $\mathcal{D}$.

Theorem 5.3 *Consider m samples $(X_1, Y_1), \ldots, (X_m, Y_m) \sim \mu$. Consider a (random) tree ensemble τ that may depend on the samples (X_i, Y_i)'s.*

Let $\mathcal{D}$ be a family of distributions $(\alpha, \varepsilon, \delta)$-learnable with m samples, and let $\hat{\mu}$ be the distribution returned by an $(\alpha, \varepsilon, \delta)$-learner. Let T be any (random) decision tree subordinated to τ constructed using τ, the samples (X_i, Y_i), and $\hat{\mu}$. Then with probability at least $1 - \delta$

$$err(T) \;\leq\; err(\tau) + \alpha \cdot \|\mu - \mu^*\|_{\mathcal{L}_1} + \varepsilon,$$

where $\mu^* = \mathrm{argmin}_{\mu' \in \mathcal{D}} \|\mu' - \mu\|_{\mathcal{L}_1}$ *is the closest distribution to* μ *in* $\mathcal{D}$.

Proof. Again, from (5-1) we have for every scenario of the samples

$$\mathrm{err}(T) \leq \mathrm{err}(\tau) + \Pr_{(X,Y)\sim\mu}(T(X) \neq \tau(X)).$$

Moreover,

$$\begin{aligned}
\Pr_{(X,Y)\sim\mu}(T(X) \neq \tau(X)) - \mathrm{diff}_{\hat{\mu}}(T, \tau) &= \sum_{C \in \mathcal{C}(\tau): T(C) \neq \tau(C)} (\mu(C) - \hat{\mu}(C)) \\
&\leq \sum_{C \in \mathcal{C}(\tau)} \int_C |pdf_{\mu}(x) - pdf_{\hat{\mu}}(x)| \, \mathrm{d}x \\
&= \|\mu - \hat{\mu}\|_{\mathcal{L}_1}
\end{aligned}$$

Finally, since $\hat{\mu}$ is constructed by an $(\alpha, \varepsilon, \delta)$-learner we have that with probability at least $1 - \delta$

$$\|\mu - \hat{\mu}\|_{\mathcal{L}_1} \leq \alpha \cdot \|\mu - \mu^*\|_{\mathcal{L}_1} + \varepsilon$$

The theorem then follows by chaining the previous three inequalities. ■

6
Experimental Results

In this chapter we conduct computational experiments to evaluate the quality of the BA tree obtained with the different methods proposed. The experiments use random forests as the tree ensembles considered.

6.1
Datasets and experimental setup

For the datasets considered, we use all datasets from the latest work on BA trees [Vidal and Schiffer, 2020], with additional datasets from the work of [Fernández-Delgado et al., 2014] where a range of classifiers are compared over 121 datasets. Given the computational complexity of the Dynamic Programming algorithm, the only datasets considered from [Fernández-Delgado et al., 2014] are those with at most 20 features. In total, we consider 62 datasets.

For each data set, we used one-hot encoding on categorical data, binned continuous features into ten ordinal scales to obtain discrete numerical features, and performed a ten-fold cross-validation to generate the training $(train_1, ..., train_{10})$ and test $(test_1, ..., test_{10})$ sets. Each of the training instances was used to compute a random forest to be reproduced by the BA-tree algorithms. We selected the parameters (number of trees and maximum tree depth) of the random forest for each instance to maximize accuracy via five-fold cross-validation, where the number of trees ranged from 10 to 50, and the depth from 3 to 10.

We then constructed the random forests for each instance using these optimal parameters. We ran the BA-tree algorithms with a 30-minute time limit to reproduce these random forests. If any of the algorithms reached the time limit, the entire data set (i.e., its 10 instances) was discarded. After applying this procedure, 21 datasets remained: Acute Inflammations (Acute Inflammation and Acute Nephritis), Balance Scale, Car, Vertebral Column[1], COMPAS-ProPublica, Fertility, FICO, Haberman, Hayes Roth, Iris, Led Display, Lenses, Monks-1, Monks-2, Monks-3, Nursery, Post Operative,

[1]There are two classification tasks in this dataset. We consider the classification task with 3 classes.

Tic-Tac-Toe, Titanic and Zoo. All these datasets can be found in the UCI machine learning repository, aside from COMPAS-ProPublica and FICO, found in [Hu et al., 2020]. Further details for each data set, such as the number of samples, features, and number of classes, are displayed in Table 6.1.

We have also computed decision trees directly from each of the training instances. The decision trees were used as the baseline to compare against our approximate born again trees. Similarly as it was done for the random forests, we selected the pruning parameter of the decision tree for each instance to maximize accuracy via five-fold cross-validation, where the considered pruning values were $\{0, 0.01, 0.02, 0.05, 0.1, 0.2\}$.

Datasets	Characteristics			Random forest params.	
	Samples	Features	Classes	Number of trees	Max depth
Acute Inf.	120	6	2	10	3
Acute Neph.	120	6	2	10	3
Balance scale	625	4	3	50	5
Car	1728	6	4	34	10
Column	310	6	3	21	5
COMPAS	6907	12	2	10	6
Fertility	100	9	2	28	7
FICO	10459	17	2	12	6
Haberman	306	3	2	17	4
Hayes Roth	160	3	3	24	6
Iris	150	4	3	10	4
Led display	1000	7	10	12	5
Lenses	24	4	3	11	3
Monks-1	556	6	2	31	9
Monks-2	601	6	2	50	10
Monks-3	554	6	2	11	6
Nursery	12960	8	3	49	10
Post operative	87	8	3	41	3
Tic-Tac-Toe	958	9	2	33	9
Titanic	2201	3	2	10	4
Zoo	101	16	7	11	6

Table 6.1: Dataset characteristics, and number of trees and maximum tree depth for the computed random forests for each dataset.

The proposed algorithms were implemented in C++ and compiled with GCC 9.2 using flag -O regarding our computational setup. The original random forests and decision trees (baseline) were obtained in python using the library Scikit-learn v0.24. All our experiments were run on a single thread on an Intel(R)Core(TM) i7-10750H CPU @2.60GHz, with 64GB of RAM available and running Ubuntu 20.04.

6.2 Algorithms tested

We compared the Dynamic Programming, the greedy and beam search heuristics described in Chapter 4. The beam search heuristic was computed using $K = 3$. For each algorithm we considered the cell-importance functions:

- **Volume.** This is the function used in Example 1, where the importance $v(C)$ of a cell $C \in \mathcal{C}(\tau)$ is just its (Lebesgue) volume.
- **Empirical measure.** This is the function used in Example 2, where the importance $v(C)$ of a cell $C \in \mathcal{C}(\tau)$ is the fraction of points in the training set of the instance that lie on C.

We have decided not to evaluate the **Parametric distributions** cell-importance function as in Example 3, since the results for many datasets led to infeasible computational time.

6.3 Dynamic Programming algorithm

6.3.1 Adherence and computational time

In order to evaluate how well the random forests can be reproduced using decision trees with limited depth, we computed the adherence of the BA trees returned by our Dynamic Programming algorithm. Recall that adherence is precisely what is maximized by the Dynamic Program: the adherence of a decision tree T to a random forest τ with respect to the cell-importance function v is the total importance of the cells where the classifiers agree: $\sum_{C \in \mathcal{C}(\tau)} v(C) \cdot \mathbb{1}(T(C) = \tau(C))$. For each instance (and its associated random forest), cell-importance function {`Volume`, `Emp.Measure`}, and depth limit $d \in \{1, 2, \dots, 6\}$, we have computed the max adherence decision tree using the Dynamic Programming algorithm, and observed the adherence value obtained. Figure 6.1 reports the average adherence for each configuration.

We notice that shallow decision trees are able to reproduce random forests with a high adherence: with both cell-importance functions, even depth 4 BA trees achieve on average at least approximately 90% of perfect reproduction of the random forests over the whole space. Moreover, the averages steadily increase and become closer to the 95% mark at depth 6. Appendix A.2 presents detailed adherence information for depth limits 4, 5, 6. In particular, these show that even with only depth 6, on 5 out of the 21 datasets the BA constructed obtained *perfect* adherence with both cell-importance functions.

Regarding the computational complexity of the procedure, Figure 6.2 reports average running time. The algorithm has shown to be quite practical for the datasets considered. Moreover, in the additional information presented in Appendix A.2, even at depth 6 the algorithm runs in under 2 seconds for all but 7 of the datasets, and the maximum running time is 440.70 seconds. However,

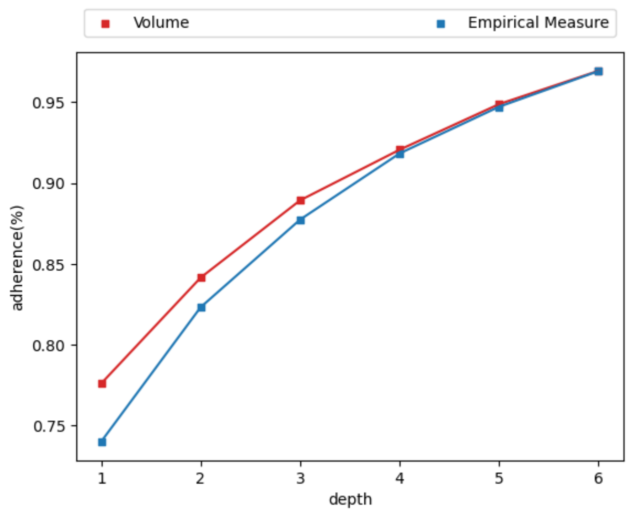

Figure 6.1: Average adherence obtained by the Dynamic Programming algorithm.

when compared against the average runtime for the decision tree constructed directly from data (< 0.01s for all datasets), it is considered significantly slower.

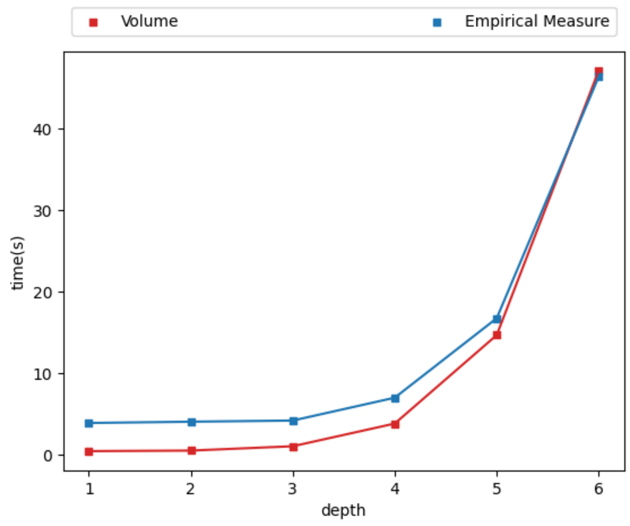

Figure 6.2: Average runtime of the Dynamic Programming algorithm.

We clearly see that the computational complexity quickly increases as a function of the depth limit. Also, note that running times do not seem directly correlated to the number of features in the instance of the parameters of the random forest to be reproduced (see Table 6.1). The explanation for both of these phenomena is that since the DP algorithm was implemented recursively, it may not visit all sub-problems, and the number of sub-problems visited depends on the number of distinct values (after binning) on the features.

6.3.2 Classification accuracy

Next, we analyze the accuracy of the BA trees computed by the Dynamic Programming algorithm. We compare this accuracy against that of the original random forests to be reproduced and also against the accuracy of decision trees of given depth build directly on the instances. Figure 6.3 plots the average accuracies. Tables 6.2-6.4 present detailed results for each dataset for depth limits 4, 5, 6.

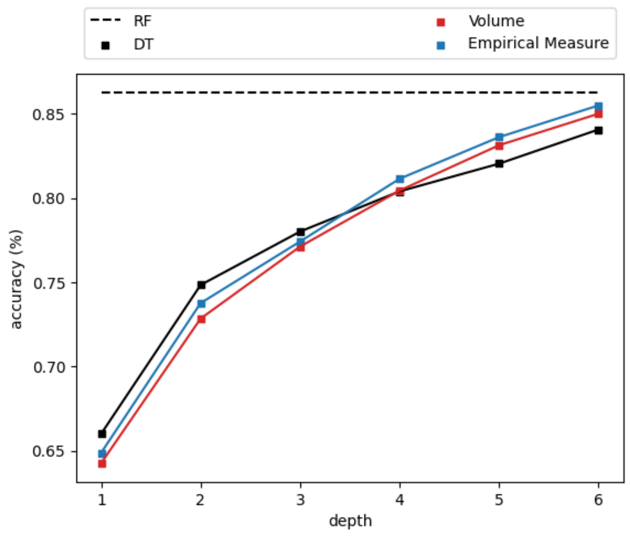

Figure 6.3: Dynamic Programming average accuracy over all instances. RF denotes the average accuracy of the random forests reproduced, and DT the average accuracy of the decision trees computed directly from the data.

Once more, we observe how well shallow BA trees (especially with depth 5, 6) can obtain accuracy comparable to that of the random forest. In particular, with depth 6 the BA trees stayed on average within 1% of the accuracy of random forests. Looking at Table 6.4 we see that in 11 out of 21 datasets, the BA trees with depth 6 had accuracy at least as good as that of the random forest and even surpassed it in some datasets. Thus, these results indicate that there is little or no loss in accuracy in working with more interpretable classifiers in many situations.

It is also interesting to see how these BA trees compare against decision trees of the same depth computed directly over the data. At depth limit set to 5 and 6, BA trees have an advantage over the decision trees. In particular, with depth 6 BA trees present a huge gain of 16,6% in accuracy on the dataset Monks-1, as well as very significant gains of 8.8%, 4.9%, and 4.2% on the datasets Balance-scale, Column and Tic-Tac-Toe, respectively. Finally, we note

(Acc, depth = 4)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	0.774	0.768	0.785	0.851
Car	0.835	**0.868**	0.845	0.975
Column	0.709	0.716	0.767	0.787
COMPAS	0.654	**0.664**	0.659	0.667
Fertility	0.86	0.86	0.88	0.850
FICO	0.688	**0.699**	0.695	0.702
Haberman	**0.732**	0.71	0.71	0.726
Hayes Roth	**0.806**	**0.787**	0.656	0.850
Iris	0.96	0.953	0.966	0.946
Led display	0.584	0.67	0.699	0.688
Lenses	0.783	0.783	0.833	0.783
Monks-1	**0.969**	**0.974**	0.836	0.992
Monks-2	0.632	**0.665**	0.657	0.948
Monks-3	0.983	0.981	0.989	0.976
Nursery	**0.889**	**0.89**	0.858	0.979
Post operative	0.713	0.713	0.713	0.713
Tic-Tac-Toe	0.745	0.732	0.767	0.936
Titanic	0.786	0.786	0.787	0.786
Zoo	**0.79**	**0.82**	0.78	0.960
Average	**0.804**	**0.811**	0.803	0.862

Table 6.2: Accuracy for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 4. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

that both cell-importance functions considered had very similar behavior, with the Empirical Measure performing slightly better.

(Acc, depth = 5)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	**0.798**	**0.811**	0.763	0.851
Car	**0.903**	**0.913**	0.872	0.975
Column	0.722	0.732	0.764	0.787
COMPAS	0.66	0.665	0.667	0.667
Fertility	0.85	0.869	0.88	0.85
FICO	0.696	0.696	0.699	0.702
Haberman	**0.729**	0.71	0.71	0.726
Hayes Roth	**0.843**	**0.825**	0.693	0.85
Iris	0.953	0.953	0.966	0.946
Led display	0.675	0.689	0.699	0.688
Lenses	0.783	0.783	0.833	0.783
Monks-1	**0.996**	**1.0**	0.836	0.992
Monks-2	0.747	**0.805**	0.745	0.948
Monks-3	0.978	0.98	0.989	0.976
Nursery	**0.909**	**0.915**	0.885	0.979
Post operative	0.713	0.713	0.713	0.713
Tic-Tac-Toe	**0.826**	**0.823**	0.816	0.936
Titanic	0.786	0.786	0.787	0.786
Zoo	0.89	0.89	0.91	0.960
Average	**0.831**	**0.836**	0.82	0.862

Table 6.3: Accuracy for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 5. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

(Acc, depth = 6)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	**0.833**	**0.836**	0.748	0.851
Car	0.929	**0.942**	0.931	0.975
Column	0.735	**0.787**	0.738	0.787
COMPAS	0.662	0.667	0.669	0.667
Fertility	0.86	0.86	0.88	0.850
FICO	0.698	0.7	0.7	0.702
Haberman	**0.723**	**0.719**	0.71	0.726
Hayes Roth	**0.85**	**0.85**	0.843	0.850
Iris	0.946	0.946	0.966	0.946
Led display	0.675	0.69	0.7	0.688
Lenses	0.783	0.783	0.833	0.783
Monks-1	**0.996**	**1.0**	0.834	0.992
Monks-2	**0.981**	**0.981**	0.953	0.948
Monks-3	0.976	0.976	0.989	0.976
Nursery	**0.933**	**0.937**	0.909	0.979
Post operative	0.713	0.713	0.713	0.713
Tic-Tac-Toe	**0.862**	**0.86**	0.82	0.936
Titanic	0.786	0.786	0.787	0.786
Zoo	0.909	0.92	0.93	0.960
Average	**0.85**	**0.854**	0.841	0.862

Table 6.4: Accuracy for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 6. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

6.4 Heuristic methods

We now focus our attention to the more computational efficient heuristic constructions of BA trees given by the greedy and the beam search algorithms.

6.4.1 Adherence and computational time

The average adherence of the BA trees computed by the heuristic methods is presented in Figure 6.4, with additional information presented in Appendix A.3. Again, we can see that they do a fairly good job in obtaining shallow trees that approximate the behavior of the input random forests over the whole space. As expected, the beam search heuristic obtains better adherence, especially for deeper trees. In addition, we do not observe a significant loss of adherence between the beam search and the (optimal) Dynamic Programming algorithm (for example, for depth 6, the beam search with Volume cell-importance function obtained average adherence of 0.952, versus an optimal adherence of 0.969).

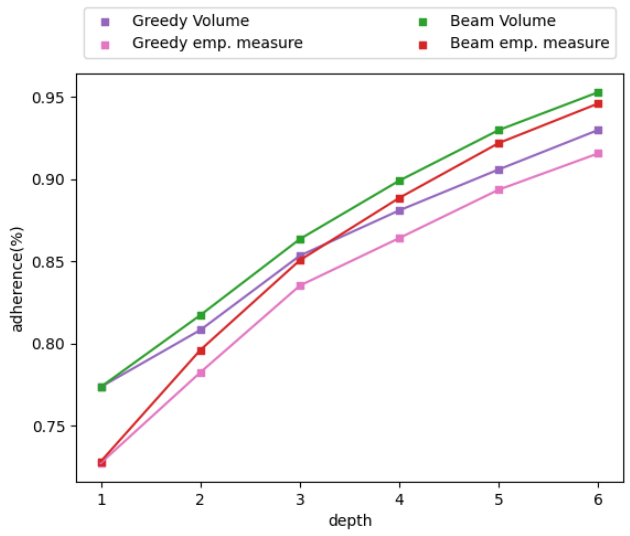

Figure 6.4: Average adherence obtained by the heuristic algorithms.

However, in terms of running time, the heuristics are at least an order of magnitude faster than the Dynamic Programming algorithms, as shown in Figure 6.5 (with details in Appendix A.3). Interestingly, the choice of cell-importance function had the most significant impact on the running time of the different heuristics, using Empirical Measure adding significant overhead. We note that the average running time was below 3 seconds on all datasets for the Volume function. On the other hand, there was little difference in running time between greedy and beam search, although Figure 6.5 indicates that the cost of beam search becomes increasingly steeper as the depth limit increases. Nonetheless, up to the depth limit 6 considered, both heuristics exhibited behavior almost linear in this parameter.

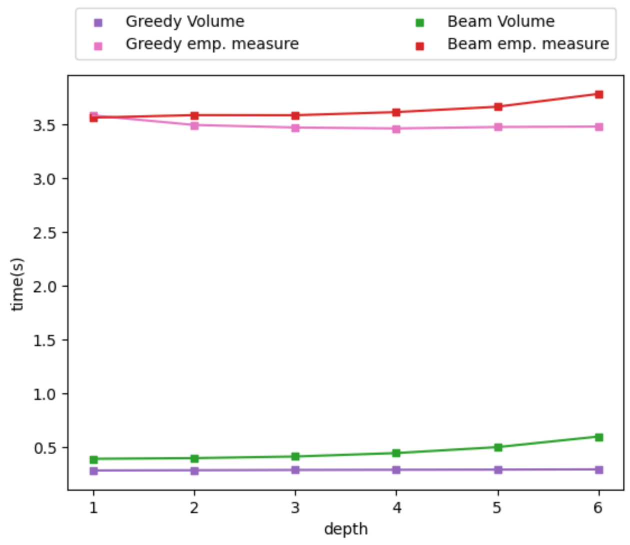

Figure 6.5: Average runtime obtained by the heuristic algorithms.

6.4.2
Classification accuracy

Finally, we evaluate the classification accuracy of these efficient heuristics. Average accuracies are presented in Figure 6.6 and detailed information for depths 4, 5, 6 are presented in Tables 6.5-6.7. Once more, we can see that these heuristics are able to compute shallow BA trees (especially with depth 5, 6) with accuracy comparable to that of the random forest. The beam search with the Empirical Measure function obtained the best result on average across all depth limits, and for depth 6 obtained accuracy within 2% of the original random forests. This represents a loss of only about 1% compared to the accuracy of the Dynamic Programming algorithm, however, with a much larger computational advantage. In more than half of the datasets (11 out of 21), this heuristic with depth 6 had accuracy at least as good as that of the random forest, even surpassing it in some datasets.

As expected, the beam search performs consistently better than the greedy heuristic. Moreover, while computationally more expensive, using the Empirical Measure as cell-importance function also presented better results. It is intuitively clear that the empirical measure is more relevant for the problem, given that it approximates the true distribution of the data and also the statistical guarantee from Theorem 5.1.

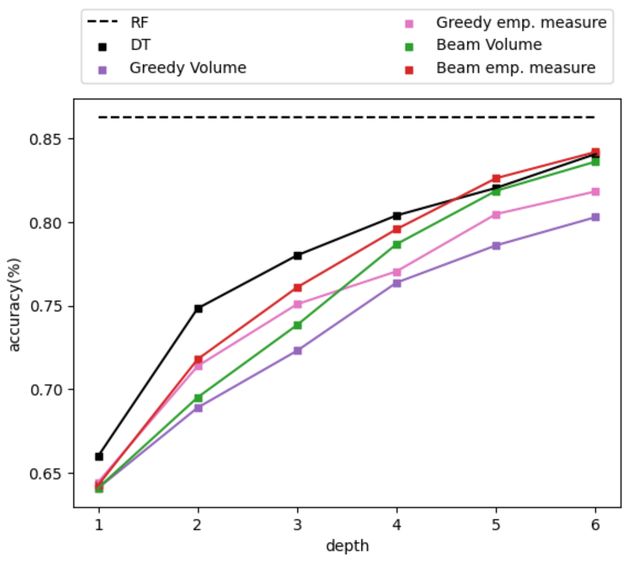

Figure 6.6: Average accuracy obtained by the heuristic algorithms. RF denotes the average accuracy of the random forests reproduced, and DT the average accuracy of the decision trees computed directly from the data.

Comparing against the accuracy of decision trees of the same depth computed directly over the data, we see that the BA trees computed by the beam search procedure are better on average only for the depth set as 5, although this advantage is less marked than of the Dynamic Programming algorithm. On average the heuristic methods produce BA trees that have similar accuracies to the decision trees. Nonetheless, for some of the datasets, these BA trees still provide significant gains: for depth 6 and the Empirical Measure function, there is still a gain of 16.6% in accuracy on the dataset Monks-1, and a gain of 5.1% on the dataset Balance scale.

(Acc, depth = 4)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf	0.916	1.0	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	0.71	0.72	0.736	0.766	0.785	0.851
Car	0.779	0.733	0.803	0.765	0.845	0.975
Column	0.561	0.696	0.638	0.693	0.767	0.787
COMPAS	0.654	0.657	0.653	**0.663**	0.659	0.667
Fertility	0.88	0.88	0.88	0.88	0.88	0.850
FICO	0.671	0.69	0.652	0.668	0.695	0.702
Haberman	0.707	**0.726**	**0.723**	**0.729**	0.71	0.726
Hayes Roth	**0.743**	**0.681**	**0.737**	**0.706**	0.656	0.850
Iris	0.893	0.893	0.893	0.913	0.966	0.946
Led display	0.576	0.605	0.587	0.639	0.699	0.688
Lenses	0.783	0.783	0.783	0.783	0.833	0.783
Monks-1	**0.926**	**0.841**	**0.926**	**0.956**	0.836	0.992
Monks-2	**0.658**	0.614	0.648	**0.663**	0.657	0.948
Monks-3	0.799	0.86	0.983	0.989	0.989	0.976
Nursery	0.806	0.836	0.824	0.855	0.858	0.979
Post operative	0.713	0.713	0.713	0.713	0.713	0.713
Tic-Tac-Toe	0.708	0.716	0.767	0.753	0.767	0.936
Titanic	0.786	0.786	0.786	0.786	0.787	0.786
Zoo	0.77	0.75	**0.79**	**0.79**	0.78	0.960
Average	0.763	0.77	0.786	0.795	0.803	0.862

Table 6.5: Accuracy for the greedy and the beam search heuristics with volume and empirical measure cell-importance functions and depth set as 4. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

(Acc, depth = 5)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf.	0.966	1.0	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	0.737	**0.768**	**0.772**	0.76	0.763	0.851
Car	0.793	0.789	0.844	0.858	0.872	0.975
Column	0.590	0.712	0.674	0.735	0.764	0.787
COMPAS	0.655	0.66	0.658	0.665	0.667	0.667
Fertility	0.88	0.88	0.869	0.879	0.88	0.85
FICO	**0.711**	**0.712**	0.694	0.703	0.699	0.702
Haberman	**0.713**	**0.726**	**0.729**	**0.72**	0.71	0.726
Hayes Roth	**0.793**	**0.806**	**0.825**	**0.793**	0.693	0.85
Iris	0.86	0.9	0.92	0.913	0.966	0.946
Led display	0.611	0.676	0.675	0.681	0.699	0.688
Lenses	0.783	0.783	0.783	0.783	0.833	0.783
Monks-1	**1.0**	**0.982**	**1.0**	**1.0**	0.836	0.992
Monks-2	0.657	0.623	**0.747**	**0.805**	0.745	0.948
Monks-3	0.799	0.862	0.978	0.985	0.989	0.976
Nursery	0.849	0.865	0.858	0.881	0.885	0.979
Post operative	0.713	0.713	0.713	0.702	0.713	0.713
Tic-Tac-Toe	0.730	0.746	0.796	0.8	0.816	0.936
Titanic	0.786	0.786	0.786	0.786	0.787	0.786
Zoo	0.88	0.91	0.87	0.9	0.91	0.960
Average	0.786	0.804	0.818	**0.826**	0.82	0.862

Table 6.6: Accuracy for the greedy and beam heuristics with volume and empirical measure cell-importance functions and depth set as 5. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

(Acc, depth = 6)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf.	0.991	1.0	0.991	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	**0.758**	**0.779**	**0.784**	**0.795**	0.748	0.851
Car	0.805	0.834	0.859	0.894	0.931	0.975
Column	0.590	0.719	0.69	0.716	0.738	0.787
COMPAS	0.658	0.666	0.659	0.667	0.669	0.667
Fertility	0.88	0.88	0.87	0.87	0.88	0.850
FICO	**0.71**	**0.71**	0.687	0.695	0.7	0.702
Haberman	0.703	**0.72**	**0.732**	**0.729**	0.71	0.726
Hayes Roth	0.818	0.818	0.831	0.8	0.843	0.850
Iris	0.873	0.9	0.94	0.953	0.966	0.946
Led display	0.651	0.688	0.677	0.687	0.7	0.688
Lenses	0.783	0.783	0.783	0.783	0.816	0.783
Monks-1	**1.0**	**1.0**	**1.0**	**1.0**	0.834	0.992
Monks-2	0.715	0.707	**0.993**	**1.0**	0.953	0.948
Monks-3	0.906	0.926	0.976	0.976	0.989	0.976
Nursery	0.863	0.886	0.869	0.899	0.909	0.979
Post operative	0.713	0.713	0.702	0.702	0.713	0.713
Tic-Tac-Toe	0.748	0.759	0.818	0.817	0.82	0.936
Titanic	0.786	0.786	0.786	0.786	0.787	0.786
Zoo	0.909	0.909	0.91	0.909	0.93	0.960
Average	0.802	0.818	0.836	0.841	0.841	0.862

Table 6.7: Accuracy for the greedy and beam search heuristics with volume and empirical measure cell-importance functions and depth set as 6. The results in **bold** represent the instances where the BA tree has a higher accuracy than the decision tree.

7
Conclusions

In this work, we have proposed in a principled way the problem of constructing a decision tree that approximately reproduces a tree ensemble, exploring the tradeoff between accuracy and interpretability. First, we have defined the problem of obtaining the most adherent tree to the ensemble and proposed distinctive cell-importance functions. After that, we have proposed a dynamic programming algorithm to solve the problem and have proved that the decision trees obtained by this procedure satisfy generalization guarantees related to the generalization of the original tree ensembles. Afterwards, to mitigate the dynamic programming algorithm's exponential nature in the number of features, we have presented heuristic algorithms (greedy and beam search) that significantly reduce the number of recursions needed to compute the approximate BA tree.

The computational results on 21 datasets have shown that for approximate BA trees constructed using the DP, even depth-4 BA trees achieve on average at least approximately 90% of perfect reproduction of the random forests over the whole space. Moreover, the averages steadily increase and become closer to the 95% mark at depth 6. When considering accuracy, BA trees stayed on average within 1% of the accuracy of random forests. Compared against decision trees of the same depth computed directly from data, at depth limit set to 5, BA trees have an advantage over the decision trees, which is seen also seen at trees of depth 6.

Considering the heuristic methods, we have observed that these heuristics are able to compute shallow BA trees (especially with depth 5, 6) with accuracy comparable to that of the random forest. The beam search with the Empirical Measure function obtained the best result on average across all depth limits, and for depth 6 obtained accuracy within 2% of the original random forests. This represents a loss of only 1% compared to the accuracy of the Dynamic Programming algorithm, however, with a much more significant computational advantage. Comparing against the accuracy of decision trees of same depth computed directly over the data, we see that the BA trees computed by the beam search procedure are better on average for depth set as 5, although this advantage is less marked than of the Dynamic Programming algorithm.

However, on average the heuristic results produce BA trees that have similar accuracies to the decision trees constructed directly from data.

Overall, the results indicate that there is little or no loss in accuracy in the computed approximate BA trees in many situations, meaning that it is possible to work with more interpretable classifiers. For future works, we suggest focusing on approaches that lead to scalability, such as: finding tighter bounds to the exact and heuristic methods and maybe even using another parameter aside from cells to evaluate the feature space. Another direction might be to create a decision tree with at least chosen adherence instead of at most a depth limit. This seems to be an even more complicated problem to solve. Nevertheless, it remains an interesting research question.

Bibliography

[Adnan and Islam, 2016] Adnan, M. N. and Islam, M. (2016). Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm. *Knowledge-Based Systems*, 110:86–97.

[Ashtiani et al., 2020] Ashtiani, H., Ben-David, S., Harvey, N., Liaw, C., Mehrabian, A., and Plan, Y. (2020). Near-optimal sample complexity bounds for robust learning of gaussian mixtures via compression schemes. *Journal of the ACM*, 67:1–42.

[Bai et al., 2020] Bai, J., Li, Y., Li, J., Jiang, Y., and Xia, S. (2020). Rectified decision trees: Towards interpretability, compression and empirical soundness.

[Bastani et al., 2019] Bastani, O., Kim, C., and Bastani, H. (2019). Interpreting blackbox models via model extraction.

[Bernard et al., 2009] Bernard, S., Heutte, L., and Adam, S. (2009). On the selection of decision trees in random forests.

[Boucheron et al., 2013] Boucheron, S., Lugosi, G., and Massart, P. (2013). *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press.

[Boz, 2002] Boz, O. (2002). Extracting decision trees from trained neural networks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 456–461, New York, NY, USA. Association for Computing Machinery.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2):123–140.

[Breiman, 1997] Breiman, L. (1997). Arcing the edge.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

[Breiman and Shang, 1996] Breiman, L. and Shang, N. (1996). Born again trees.

[Carvalho et al., 2019] Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8).

[Craven and Shavlik, 1995] Craven, M. W. and Shavlik, J. (1995). Extracting thee-structured representations of thained networks.

[Deng, 2014] Deng, H. (2014). Interpreting tree ensembles with intrees.

[Fernández-Delgado et al., 2014] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(90):3133–3181.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

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

[Frosst and Hinton, 2017] Frosst, N. and Hinton, G. (2017). Distilling a neural network into a soft decision tree.

[Guidotti et al., 2018] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

[Hara and Hayashi, 2018] Hara, S. and Hayashi, K. (2018). Making tree ensembles interpretable: A bayesian model selection approach. In Storkey, A. and Perez-Cruz, F., editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 77–85. PMLR.

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

[Hu et al., 2020] Hu, X., Rudin, C., and Seltzer, M. (2020). Optimal sparse decision trees.

[Jiang et al., 2017] Jiang, X., Wu, C.-a., and Guo, H. (2017). Forest pruning based on branch importance. *Computational Intelligence and Neuroscience*, 2017:1–11.

[Johansson et al., 2011] Johansson, U., Sönströd, C., and Löfström, T. (2011). One tree to explain them all. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1444–1451.

[Joly et al., 2012] Joly, A., Schnitzler, F., Geurts, P., and Wehenkel, L. (2012). L1-based compression of random forest models. *20th European Symposium on Artificial Neural Networks*.

[Krishnan et al., 1999] Krishnan, R., Sivakumar, G., and Bhattacharya, P. (1999). Extracting decision trees from trained neural networks. *Pattern Recognition*, 32(12):1999–2009.

[Latinne et al., 2001] Latinne, P., Debeir, O., and Decaestecker, C. (2001). Limiting the number of trees in random forests. volume 2096, pages 178–187.

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

[Margineantu and Dietterich, 1997] Margineantu, D. D. and Dietterich, T. G. (1997). Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 211–218, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Meinshausen, 2009] Meinshausen, N. (2009). Forest garrote.

[Mollas et al., 2020] Mollas, I., Bassiliades, N., Vlahavas, I., and Tsoumakas, G. (2020). Lionforests: Local interpretation of random forests.

[Nan et al., 2016] Nan, F., Wang, J., and Saligrama, V. (2016). Pruning random forests for prediction on a budget.

[Painsky and Rosset, 2018] Painsky, A. and Rosset, S. (2018). Lossless (and lossy) compression of random forests.

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

[Quinlan, 1999] Quinlan, J. R. (1999). Miniboosting decision trees.

[Ren et al., 2015] Ren, S., Cao, X., Wei, Y., and Sun, J. (2015). Global refinement of random forest. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 723–730.

[Rudin et al., 2021] Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges.

[Schapire and Freund, 2012] Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and Algorithms*. The MIT Press.

[Schetinin et al., 2007] Schetinin, V., Fieldsend, J., Partridge, D., Coats, T., Krzanowski, W., Everson, R., Bailey, T., and Hernandez, A. (2007). Confident interpretation of bayesian decision tree ensembles for clinical applications. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 11:312–9.

[Shannon and Banks, 1999] Shannon, W. and Banks, D. (1999). Combining classification trees using mle. *Statistics in medicine*, 18(6):727—740.

[Sirikulviriya and Sinthupinyo, 2011] Sirikulviriya, N. and Sinthupinyo, S. (2011). Integration of rules from a random forest.

[Tan et al., 2020] Tan, S., Soloviev, M., Hooker, G., and Wells, M. T. (2020). Tree space prototypes: Another look at making tree ensembles interpretable.

[Van Assche and Blockeel, 2007] Van Assche, A. and Blockeel, H. (2007). Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In Kok, J. N., Koronacki, J., Mantaras, R. L. d., Matwin, S., Mladenič, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, pages 418–429, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Vandewiele et al., 2017] Vandewiele, G., Lannoye, K., Janssens, O., Ongenae, F., De Turck, F., and Hoecke, S. (2017). A genetic algorithm for interpretable model extraction from decision tree ensembles. pages 104–115.

[Vidal and Schiffer, 2020] Vidal, T. and Schiffer, M. (2020). Born-again tree ensembles. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9743–9753. PMLR.

[Yang et al., 2018] Yang, C., Rangarajan, A., and Ranka, S. (2018). Global model interpretation via recursive partitioning.

[Yang et al., 2012] Yang, F., Lu, W.-h., Luo, L.-k., and Li, T. (2012). Margin optimization based pruning for random forest. *Neurocomputing*, 94:54–63.

[Zhou and Hooker, 2016] Zhou, Y. and Hooker, G. (2016). Interpreting models via single tree approximation.

[Zhou and Tang, 2003] Zhou, Z. and Tang, W. (2003). Selective ensemble of decision trees. In *RSFDGrC*.

[Zouggar and Adla, 2019] Zouggar, S. T. and Adla, A. (2019). A Pruning of Random Forests: a diversity-based heuristic measure to simplify a random forest ensemble. *INFOCOMP Journal of Computer Science*, 18(1):01–08.

A
Appendix

A.1
Filtering cell-breakpoints

In order to reduce the number of recursive calls, before applying the Dynamic Programming algorithm, we apply a simple process of filtering out cell-breakpoints ℓ described in [Vidal and Schiffer, 2020]. The idea is that we can merge two adjacent "rows" of cells of the ensemble τ if both cells on each "column" have the same class (see Figure A.1). More precisely, we can filter out for each feature j any cell-breakpoint $\ell \in \{1, \dots, |H_j|\}$ where $\tau(z) = \tau(z + e_j)$ for all cells z with $z_j = \ell$ (e_j is the j-th canonical basis vector).

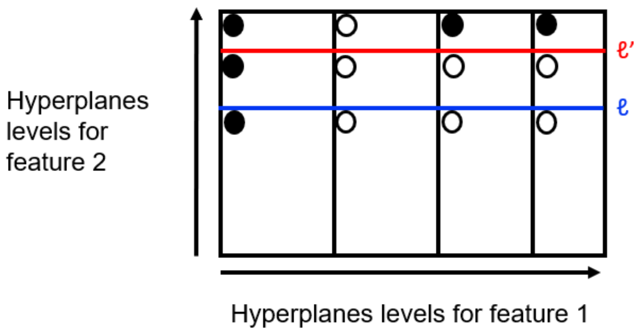
Hyperplanes levels for feature 2
ℓ'
ℓ
Hyperplanes levels for feature 1

Figure A.1: Representation of two cell break-points ℓ and ℓ', ℓ can be discarded, while ℓ' cannot.

A.2
Adherence and computational time for the Dynamic Programming algorithm

(Depth = 4)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.974	0.01>	0.986	0.01>
Acute Neph.	0.933	0.01>	0.966	0.01>
Balance scale	0.911	0.04	0.907	0.04
Car	0.913	0.129	0.870	0.228
Column	0.988	15.31	0.901	15.96
COMPAS	0.834	0.05	0.911	0.616
Fertility	0.961	32.69	0.96	33.19
FICO	0.831	2.55	0.835	63.92
Haberman	0.983	1.93	0.91	1.4
Hayes Roth	0.889	0.01>	0.934	0.01>
Iris	0.974	0.03	0.957	0.02
Led display	0.724	0.01>	0.894	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.977	0.01>	0.973	0.01>
Monks-2	0.851	0.01	0.715	0.01
Monks-3	0.996	0.01>	0.994	0.01>
Nursery	0.897	1.47	0.9	6.02
Post operative	0.988	0.08	0.971	0.06
Tic-Tac-Toe	0.885	18.35	0.874	18.95
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.823	8.42	0.826	7.22
Average	0.920	3.86	0.918	7.03

Table A.1: Adherence and runtime for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 4.

(Depth = 5)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.998	0.01>	0.997	0.01>
Acute Neph.	0.960	0.01>	0.976	0.01>
Balance scale	0.95	0.12	0.949	0.14
Car	0.937	0.39	0.914	0.47
Column	0.992	67.29	0.934	69.19
COMPAS	0.870	0.12	0.937	0.66
Fertility	0.965	135.37	0.964	128.28
FICO	0.854	6.23	0.859	62.83
Haberman	0.991	8.57	0.94	6.58
Hayes Roth	0.98	0.01>	0.989	0.01>
Iris	0.992	0.12	0.981	0.09
Led display	0.856	0.01>	0.96	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01>	0.998	0.01>
Monks-2	0.915	0.03	0.844	0.03
Monks-3	0.998	0.01>	0.996	0.01>
Nursery	0.919	3.47	0.924	7.48
Post operative	0.991	0.22	0.978	0.171
Tic-Tac-Toe	0.906	63.1	0.896	56.42
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.852	23.75	0.854	19.78
Average	0.948	14.7	0.947	16.77

Table A.2: Adherence and runtime for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 5.

(Depth = 6)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	1.0	0.01>	1.0	0.01>
Acute Neph.	1.0	0.01>	1.0	0.01>
Balance scale	0.983	0.29	0.979	0.3
Car	0.961	1.06	0.947	1.09
Column	0.994	237.14	0.955	230.49
COMPAS	0.898	0.3	0.953	0.81
Fertility	0.97	440.70	0.969	420.28
FICO	0.873	15.09	0.878	70.36
Haberman	0.996	25.45	0.963	20.03
Hayes Roth	1.0	0.01>	1.0	0.01>
Iris	0.998	0.31	0.995	0.22
Led display	0.942	0.01>	0.99	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01	0.998	0.01
Monks-2	0.998	0.07	0.994	0.05
Monks-3	0.999	0.01>	0.999	0.01>
Nursery	0.95	8.77	0.952	11.58
Post operative	0.994	0.57	0.985	0.45
Tic-Tac-Toe	0.925	197.97	0.916	167.28
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.878	61.36	0.88	50.64
Average	0.969	47.1	0.969	46.36

Table A.3: Adherence and runtime for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 6.

A.3
Adherence and computational time for the heuristics

(Depth = 4)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.926	0.01>	0.976	0.01>
Acute Neph.	0.927	0.01>	0.962	0.01>
Balance scale	0.824	0.01>	0.803	0.01
Car	0.867	0.01>	0.732	0.1
Column	0.976	0.43	0.818	0.76
COMPAS	0.817	0.01	0.886	0.5
Fertility	0.959	1.64	0.959	2.07
FICO	0.819	0.66	0.823	56.18
Haberman	0.948	0.01	0.827	0.02
Hayes Roth	0.78	0.01>	0.843	0.01>
Iris	0.878	0.01>	0.859	0.01>
Led display	0.645	0.01>	0.807	0.01>
Lenses	0.999	0.01>	0.996	0.01>
Monks-1	0.964	0.01>	0.854	0.01>
Monks-2	0.829	0.01>	0.665	0.01
Monks-3	0.881	0.01>	0.887	0.01>
Nursery	0.806	0.58	0.84	5.31
Post operative	0.966	0.01	0.932	0.02
Tic-Tac-Toe	0.867	1.57	0.856	6.03
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.82	1.02	0.82	1.62
Average	0.88	0.288	0.864	3.46

Table A.4: Adherence and runtime for the greedy heuristic with volume and empirical measure cell-importance functions and depth set as 4.

(Depth = 5)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.975	0.01>	0.984	0.01>
Acute Neph.	0.947	0.01>	0.974	0.01>
Balance scale	0.868	0.01>	0.836	0.01
Car	0.882	0.01>	0.793	0.1
Column	0.978	0.44	0.836	0.76
COMPAS	0.842	0.01	0.906	0.49
Fertility	0.959	1.6	0.959	2.07
FICO	0.84	0.66	0.846	56.4
Haberman	0.954	0.01	0.838	0.02
Hayes Roth	0.858	0.01>	0.908	0.01>
Iris	0.886	0.01>	0.869	0.01>
Led display	0.765	0.01>	0.922	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01>	0.977	0.01>
Monks-2	0.84	0.01>	0.692	0.01
Monks-3	0.881	0.01>	0.908	0.01>
Nursery	0.863	0.58	0.873	5.33
Post operative	0.966	0.01	0.932	0.02
Tic-Tac-Toe	0.878	1.58	0.869	6.03
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.841	1.03	0.841	1.63
Average	0.905	0.287	0.893	3.47

Table A.5: Adherence and runtime for the greedy heuristic with volume and empirical measure cell-importance functions and depth set as 5.

(Depth = 6)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.994	0.01>	0.995	0.01>
Acute Neph.	0.994	0.01>	0.997	0.01>
Balance scale	0.896	0.01>	0.866	0.01
Car	0.887	0.01>	0.840	0.1
Column	0.979	0.44	0.848	0.77
COMPAS	0.867	0.01	0.924	0.498
Fertility	0.959	1.67	0.959	2.08
FICO	0.852	0.67	0.857	56.48
Haberman	0.961	0.01	0.848	0.03
Hayes Roth	0.937	0.01>	0.959	0.01>
Iris	0.898	0.01>	0.883	0.01>
Led display	0.882	0.01>	0.976	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01>	0.998	0.01>
Monks-2	0.876	0.01>	0.756	0.01
Monks-3	0.963	0.01>	0.962	0.01>
Nursery	0.873	0.58	0.892	5.33
Post operative	0.966	0.01	0.932	0.02
Tic-Tac-Toe	0.885	1.58	0.876	6.04
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.858	1.04	0.858	1.64
Average	0.929	0.291	0.915	3.48

Table A.6: Adherence and runtime for the greedy heuristic with volume and empirical measure cell-importance functions and depth set as 6.

(Depth = 4)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.965	0.01>	0.978	0.01>
Acute Neph.	0.933	0.01>	0.965	0.01>
Balance scale	0.851	0.01>	0.844	0.01
Car	0.887	0.03	0.763	0.11
Column	0.978	0.69	0.83	1.02
COMPAS	0.828	0.02	0.906	0.51
Fertility	0.959	2.75	0.959	3.13
FICO	0.814	1.13	0.815	56.61
Haberman	0.959	0.02	0.845	0.03
Hayes Roth	0.8	0.01>	0.877	0.01>
Iris	0.9	0.01>	0.883	0.01>
Led display	0.71	0.01>	0.867	0.01>
Lenses	0.999	0.01>	0.996	0.01>
Monks-1	0.964	0.01>	0.949	0.01>
Monks-2	0.851	0.01>	0.711	0.01
Monks-3	0.994	0.01>	0.992	0.01>
Nursery	0.823	0.626	0.857	5.37
Post operative	0.966	0.01	0.932	0.02
Tic-Tac-Toe	0.874	2.42	0.863	6.89
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.823	1.52	0.826	2.11
Average	0.898	0.44	0.888	3.61

Table A.7: Adherence and runtime for the beam search heuristic with volume and empirical measure cell-importance functions and depth set as 4.

(Depth = 5)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.995	0.01>	0.994	0.01>
Acute Neph.	0.96	0.01>	0.976	0.01>
Balance scale	0.898	0.01>	0.882	0.01
Car	0.905	0.04	0.857	0.11
Column	0.981	0.733	0.857	1.09
COMPAS	0.862	0.03	0.931	0.52
Fertility	0.96	2.85	0.959	3.24
FICO	0.837	1.71	0.844	57.05
Haberman	0.969	0.02	0.864	0.03
Hayes Roth	0.904	0.01>	0.941	0.01>
Iris	0.927	0.01>	0.911	0.01>
Led display	0.845	0.01>	0.955	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01>	0.998	0.01>
Monks-2	0.914	0.01>	0.842	0.01
Monks-3	0.996	0.01>	0.994	0.01>
Nursery	0.867	0.654	0.89	5.4
Post operative	0.966	0.02	0.934	0.02
Tic-Tac-Toe	0.891	2.58	0.88	7.03
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.849	1.74	0.849	2.34
Average	0.929	0.49	0.921	3.66

Table A.8: Adherence and runtime for the beam search heuristic with volume and empirical measure cell-importance functions and depth set as 5.

(Depth = 6)	Volume		Empirical Measure	
	adh(%)	time(s)	adh(%)	time(s)
Acute Inf.	0.997	0.01>	0.996	0.01>
Acute Neph.	0.999	0.01>	0.999	0.01>
Balance scale	0.933	0.01	0.921	0.01
Car	0.919	0.05	0.894	0.12
Column	0.983	0.8	0.88	1.21
COMPAS	0.889	0.06	0.949	0.56
Fertility	0.96	3.02	0.96	3.42
FICO	0.859	2.81	0.863	58.53
Haberman	0.976	0.03	0.878	0.04
Hayes Roth	0.971	0.01>	0.982	0.01>
Iris	0.952	0.01>	0.939	0.01>
Led display	0.939	0.01>	0.989	0.01>
Lenses	1.0	0.01>	1.0	0.01>
Monks-1	0.998	0.01>	0.998	0.01>
Monks-2	0.997	0.01	0.993	0.01
Monks-3	0.998	0.01>	0.997	0.01>
Nursery	0.887	0.71	0.907	5.46
Post operative	0.973	0.02	0.95	0.03
Tic-Tac-Toe	0.905	2.83	0.895	7.27
Titanic	1.0	0.01>	1.0	0.01>
Zoo	0.872	2.1	0.873	2.7
Average	0.952	0.59	0.945	3.78

Table A.9: Adherence and runtime for the beam search heuristic with volume and empirical measure cell-importance functions and depth set as 6.

A.4
F1-score for the DP and heuristics

(F1, depth = 4)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	0.745	0.738	0.757	0.819
Car	0.825	0.858	0.841	0.974
Column	0.680	0.704	0.766	0.789
COMPAS	0.649	0.661	0.658	0.665
Fertility	0.817	0.817	0.79	0.836
FICO	0.675	0.69	0.695	0.702
Haberman	0.698	0.666	0.676	0.678
Hayes Roth	0.805	0.787	0.652	0.849
Iris	0.961	0.954	0.966	0.948
Led display	0.551	0.672	0.692	0.687
Lenses	0.841	0.841	0.83	0.841
Monks-1	0.969	0.974	0.810	0.992
Monks-2	0.633	0.669	0.616	0.947
Monks-3	0.983	0.981	0.989	0.976
Nursery	0.878	0.879	0.848	0.978
Post operative	0.605	0.605	0.648	0.605
Tic-Tac-Toe	0.749	0.736	0.744	0.934
Titanic	0.758	0.758	0.757	0.758
Zoo	0.771	0.806	0.764	0.963
Average	0.790	0.779	0.785	0.854

Table A.10: F1-score for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 4.

(F1, depth = 5)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	0.769	0.781	0.743	0.819
Car	0.899	0.909	0.859	0.974
Column	0.708	0.727	0.765	0.789
COMPAS	0.657	0.662	0.665	0.665
Fertility	0.812	0.823	0.794	0.836
FICO	0.694	0.695	0.699	0.702
Haberman	0.682	0.657	0.694	0.678
Hayes Roth	0.842	0.823	0.68	0.849
Iris	0.954	0.954	0.959	0.948
Led display	0.671	0.688	0.682	0.687
Lenses	0.841	0.841	0.83	0.841
Monks-1	0.996	1.0	0.783	0.992
Monks-2	0.753	0.808	0.747	0.947
Monks-3	0.978	0.98	0.989	0.976
Nursery	0.907	0.913	0.874	0.978
Post operative	0.605	0.605	0.605	0.605
Tic-Tac-Toe	0.825	0.821	0.784	0.934
Titanic	0.758	0.758	0.757	0.758
Zoo	0.879	0.886	0.91	0.963
Average	0.82	0.825	0.8	0.854

Table A.11: F1-score for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 5.

(F1, depth = 6)	Cell-importance functions		Decision Tree	Random Forest
	Volume	Emp Meas		
Acute Inf.	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0
Balance scale	0.802	0.805	0.755	0.819
Car	0.929	0.942	0.934	0.974
Column	0.723	0.787	0.787	0.789
COMPAS	0.66	0.665	0.669	0.665
Fertility	0.817	0.817	0.794	0.836
FICO	0.697	0.699	0.7	0.702
Haberman	0.671	0.673	0.669	0.678
Hayes Roth	0.849	0.849	0.838	0.849
Iris	0.948	0.948	0.959	0.948
Led display	0.672	0.688	0.702	0.687
Lenses	0.841	0.841	0.83	0.841
Monks-1	0.996	1.0	0.767	0.992
Monks-2	0.981	0.981	0.953	0.947
Monks-3	0.976	0.976	0.989	0.976
Nursery	0.93	0.934	0.898	0.978
Post operative	0.605	0.605	0.605	0.605
Tic-Tac-Toe	0.86	0.858	0.793	0.934
Titanic	0.758	0.758	0.757	0.758
Zoo	0.909	0.915	0.936	0.963
Average	0.839	0.844	0.825	0.854

Table A.12: F1-score for the dynamic programming method with volume and empirical measure cell-importance functions and depth set as 6.

(F1, depth = 4)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf.	0.914	1.0	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	0.682	0.692	0.706	0.736	0.757	0.819
Car	0.75	0.697	0.777	0.742	0.841	0.974
Column	0.464	0.663	0.569	0.657	0.766	0.789
COMPAS	0.647	0.650	0.646	0.660	0.658	0.665
Fertility	0.828	0.828	0.828	0.828	0.79	0.836
FICO	0.646	0.679	0.625	0.650	0.695	0.702
Haberman	0.666	0.674	0.688	0.683	0.676	0.678
Hayes Roth	0.734	0.684	0.728	0.710	0.652	0.849
Iris	0.887	0.886	0.887	0.908	0.966	0.948
Led display	0.526	0.586	0.557	0.63	0.692	0.687
Lenses	0.841	0.841	0.841	0.841	0.83	0.841
Monks-1	0.925	0.837	0.925	0.956	0.810	0.992
Monks-2	0.53	0.525	0.649	0.667	0.616	0.947
Monks-3	0.794	0.859	0.983	0.989	0.989	0.976
Nursery	0.789	0.825	0.813	0.845	0.848	0.978
Post operative	0.605	0.605	0.605	0.605	0.648	0.605
Tic-Tac-Toe	0.713	0.72	0.767	0.751	0.744	0.934
Titanic	0.758	0.758	0.758	0.758	0.757	0.758
Zoo	0.753	0.733	0.771	0.771	0.764	0.963
Average	0.735	0.749	0.767	0.768	0.785	0.854

Table A.13: F1-score for the greedy and the beam search heuristics with volume and empirical measure cell-importance functions and depth set as 4.

(F1, depth = 5)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf.	0.965	1.0	1.0	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	0.709	0.739	0.742	0.731	0.743	0.819
Car	0.758	0.765	0.818	0.838	0.859	0.974
Column	0.499	0.683	0.619	0.711	0.765	0.789
COMPAS	0.651	0.656	0.655	0.663	0.665	0.665
Fertility	0.828	0.828	0.823	0.828	0.794	0.836
FICO	0.705	0.708	0.686	0.699	0.699	0.702
Haberman	0.659	0.676	0.688	0.674	0.694	0.678
Hayes Roth	0.792	0.804	0.826	0.793	0.68	0.849
Iris	0.853	0.893	0.922	0.908	0.959	0.948
Led display	0.594	0.671	0.674	0.68	0.682	0.687
Lenses	0.841	0.841	0.841	0.841	0.83	0.841
Monks-1	1.0	0.982	1.0	1.0	0.783	0.992
Monks-2	0.614	0.592	0.753	0.808	0.747	0.947
Monks-3	0.794	0.861	0.978	0.985	0.989	0.976
Nursery	0.838	0.855	0.846	0.872	0.874	0.978
Post operative	0.605	0.605	0.605	0.599	0.605	0.605
Tic-Tac-Toe	0.736	0.75	0.798	0.801	0.784	0.934
Titanic	0.758	0.758	0.758	0.758	0.757	0.758
Zoo	0.873	0.91	0.859	0.889	0.91	0.963
Average	0.765	0.789	0.777	0.813	0.8	0.854

Table A.14: F1-score for the greedy and the beam search heuristics with volume and empirical measure cell-importance functions and depth set as 5.

(F1, depth = 6)	Greedy		Beam heuristic		Decision Tree	Random Forest
	Volume	Emp Meas	Volume	Emp Meas		
Acute Inf.	0.991	1.0	0.991	1.0	1.0	1.0
Acute Neph.	1.0	1.0	1.0	1.0	1.0	1.0
Balance scale	0.730	0.750	0.754	0.766	0.755	0.819
Car	0.781	0.819	0.833	0.879	0.934	0.974
Column	0.506	0.690	0.648	0.704	0.787	0.789
COMPAS	0.654	0.662	0.654	0.665	0.669	0.665
Fertility	0.828	0.828	0.823	0.823	0.794	0.836
FICO	0.705	0.706	0.684	0.693	0.7	0.702
Haberman	0.655	0.672	0.683	0.684	0.669	0.678
Hayes Roth	0.816	0.817	0.83	0.798	0.838	0.849
Iris	0.871	0.893	0.941	0.954	0.959	0.948
Led display	0.641	0.686	0.675	0.685	0.702	0.687
Lenses	0.841	0.841	0.841	0.841	0.83	0.841
Monks-1	1.0	1.0	1.0	1.0	0.767	0.992
Monks-2	0.711	0.697	0.993	1.0	0.953	0.947
Monks-3	0.905	0.926	0.976	0.976	0.989	0.976
Nursery	0.852	0.877	0.861	0.894	0.898	0.978
Post operative	0.605	0.605	0.599	0.599	0.605	0.605
Tic-Tac-Toe	0.752	0.762	0.818	0.817	0.793	0.934
Titanic	0.758	0.758	0.758	0.758	0.757	0.758
Zoo	0.904	0.904	0.904	0.904	0.936	0.963
Average	0.786	0.804	0.793	0.83	0.825	0.854

Table A.15: F1-score for the greedy and the beam search heuristics with volume and empirical measure cell-importance functions and depth set as 6.