Exploring the Whole Rashomon Set of Sparse Decision Trees

Rui Xin[1*] **Chudi Zhong**[1*] **Zhi Chen**[1*]

Takuya Takagi[2] **Margo Seltzer**[3] **Cynthia Rudin**[1]

[1] Duke University [2] Fujitsu Laboratories Ltd. [3] The University of British Columbia
{rui.xin926, chudi.zhong, zhi.chen1}@duke.edu
takagi.takuya@fujitsu.com, mseltzer@cs.ubc.ca, cynthia@cs.duke.edu

Abstract

In any given machine learning problem, there might be many models that explain the data almost equally well. However, most learning algorithms return only one of these models, leaving practitioners with no practical way to explore alternative models that might have desirable properties beyond what could be expressed by a loss function. The *Rashomon set* is the set of these all almost-optimal models. Rashomon sets can be large in size and complicated in structure, particularly for highly nonlinear function classes that allow complex interaction terms, such as decision trees. We provide the first technique for completely enumerating the Rashomon set for sparse decision trees; in fact, our work provides the first complete enumeration of any Rashomon set for a non-trivial problem with a highly nonlinear discrete function class. This allows the user an unprecedented level of control over model choice among all models that are approximately equally good. We represent the Rashomon set in a specialized data structure that supports efficient querying and sampling. We show three applications of the Rashomon set: 1) it can be used to study variable importance for the set of almost-optimal trees (as opposed to a single tree), 2) the Rashomon set for accuracy enables enumeration of the Rashomon sets for balanced accuracy and F1-score, and 3) the Rashomon set for a full dataset can be used to produce Rashomon sets constructed with only subsets of the data set. Thus, we are able to examine Rashomon sets across problems with a new lens, enabling users to choose models rather than be at the mercy of an algorithm that produces only a single model.

1 Introduction

The *Rashomon set* is the set of almost-equally-optimal models [1, 2]. Rashomon sets were named for the *Rashomon effect* of Leo Breiman, whereby many equally good models could explain the data well [3]. The Rashomon effect helps us understand that there is not just one "best" explanation for the data, but many diverse equally predictive models. The existence of Rashomon sets has important practical implications: practitioners may not want to use the single model from the Rashomon set that was output by a machine learning algorithm. Instead, they may want to explore the Rashomon set to find models with important properties, such as interpretability, fairness, or use of specific variables, or they may want to choose a model that agrees with possible causal hypotheses, monotonicity trends, ease of calculation, or simply domain intuition. In light of viewing the learning problem this way, the whole paradigm that machine learning should provide only one "optimal" model makes little sense:

[*] Equal Contribution

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

perhaps we should turn the *optimization* problem into a *feasibility* problem, and insist that we find all approximately-equally-good models and allow the user to choose among them. Perhaps the tiny sacrifice of a small amount of empirical risk can make the difference between a model that can be used and one that cannot.

While wishing for full Rashomon sets is easy, actually finding them can be extremely difficult for nonlinear function classes. Consider sparse decision trees, which is the focus of this work. Trees have wildly nonlinear relationships between features, where every leaf of depth d represents an interaction between features of order d. Even for trees of depth at most 4 with only 10 binary features, the number of possible trees (the size of the hypothesis space) is more than 9.338×10^{20} models [4]. This complexity explains why no previous work has been able to provide full Rashomon sets for optimal sparse trees, nor for any non-trivial function class with a large number of interactions.

While the hypothesis space of sparse trees is huge, the Rashomon set of good sparse trees is a small subset of it. In fact, the Rashomon set is often small enough to store, even if the function class itself is too large to be enumerated. Thus, rather than enumerate all sparse trees and store just those in the Rashomon set, we use *analytical bounds to prove that large portions of the search space do not contain any members of the Rashomon set* and can safely be excluded. This allows us to home in on just the portion of the space containing the Rashomon set. We *store the Rashomon set in a specialized data structure*, which permits memory-efficient storage and easy indexing of the Rashomon set's members. The combination of these strong bounds and efficient representation enable us to provide **the first complete enumeration of the Rashomon set for sparse decision trees on real-world datasets**. Our method is called TreeFARMS, for "Trees FAst RashoMon Sets."

The Rashomon sets we compute give us an unprecedented lens through which to examine learning problems. We can now directly answer questions such as: **Size:** How large is the Rashomon set? Does its size vary between datasets? **Variable importance:** How does variable importance change among trees within the Rashomon set? Perhaps this will give us a better sense of how important a variable is to a dataset, rather than just to one model. **Variability in predictions:** Do the models in the Rashomon set all predict similarly on the data? **Robustness:** How does the Rashomon set change if we remove a subset of data? **Rashomon sets for other losses:** How can we construct Rashomon sets for balanced accuracy and F1-score? In this work, we show how the answers to these questions can be computed directly after running our algorithm for finding the Rashomon set.

2 Related Work

Rashomon sets. Leo Breiman [3] proposed the *Rashomon effect* to describe the fact that many accurate-but-different models exist for the same data. Rashomon sets occur in healthcare, finance, criminal justice, etc. [5, 6]. They have been used for decision making [7] and robustness of estimation [8]. Semenova et al. [1] show that when the Rashomon set is large, models with various important properties such as interpretability and fairness can exist inside it. Other works use the Rashomon set to study the range of variable importance among all well-performing models [2, 9], which provides a better sense of how important a variable is in general, rather than how important it is to a specific model. Other works find diverse near-optimal solutions for integer linear programs [10, 11]. Kissel et al. [12] find collections of accurate models through model path selection. They collect models using forward selection and use these as a collection of accurate models. Our work differs from these works in that we find the *full* Rashomon set of a class of sparse decision trees on real problems.

Model enumeration. Ruggieri [13, 14] enumerates trees built from greedy top-down decision tree induction. Our method differs as our trees are not built using greedy induction but by searching the entire binary decision tree space, and our goal is to enumerate all well-performing trees, not simply the features in the greedy trees. Hara et al. [15, 16] enumerate linear models and rule lists or rule sets in descending order of their objective values. However, they find only one model for any given set of itemsets (association rules). Thus, they enumerate and store only a small subset of the Rashomon set.

Decision trees. Decision tree algorithms have a long history [17, 18, 19], but the vast majority of work on trees has used greedy induction [20, 21] to avoid solving the NP-complete problem of finding an optimal tree [22]. However, greedy tree induction provides suboptimal trees, which has propelled research since the 1990s on mathematical optimization for finding optimal decision trees [4, 23, 24, 25, 26, 27, 28, 29, 30], as well as dynamic programming with branch-and-bound [31, 32, 33, 34]. We refer readers to two recent reviews of this area [35, 36].

Bayesian trees. Bayesian analysis has long aimed to produce multiple almost-optimal models through sampling from the posterior distribution. While it might reasonably seem like a method such as BART [37] or random forest [38] might sample effectively from the Rashomon set, as our experiments show (Figure 1), they do not.

3 Bounds for Reducing the Search Space

We denote the training dataset as $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \{0,1\}^p$ are binary features. Our notation uses $y_i \in \{0,1\}$ as labels, while our method can be generalized to multiclass classification as well. Let $\ell(t, \mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}[\hat{y}_i \neq y_i]$ be the loss of tree t on the training set, where $\{\hat{y}_i\}_{i=1}^n$ are predicted labels given t. We define the objective function as the combination of the misclassification loss and a sparsity penalty on the number of leaves: $Obj(t, \mathbf{x}, \mathbf{y}) = \ell(t, \mathbf{x}, \mathbf{y}) + \lambda H_t$, where H_t is the number of leaves in tree t and λ is a regularization parameter.

Definition 1. *(ϵ-Rashomon set) Let t_{ref} be a benchmark model or reference model from $\mathcal{T}$, where $\mathcal{T}$ is a set of binary decision trees. The ϵ-Rashomon set is a set of all trees $t \in \mathcal{T}$ with $Obj(t, \mathbf{x}, \mathbf{y})$ at most $(1+\epsilon) \times Obj(t_{\text{ref}}, \mathbf{x}, \mathbf{y})$:*

$$R_{set}(\epsilon, t_{\text{ref}}, \mathcal{T}) := \{t \in \mathcal{T} : Obj(t, \mathbf{x}, \mathbf{y}) \leq (1+\epsilon) \times Obj(t_{\text{ref}}, \mathbf{x}, \mathbf{y})\}. \tag{1}$$

For example, if we permit models within 2% of the reference objective value, we would set ϵ to be 0.02. Note that we use $R_{\text{set}}(\epsilon)$ to represent $R_{\text{set}}(\epsilon, t_{\text{ref}}, \mathcal{T})$ when t_{ref} and $\mathcal{T}$ are clearly defined. We use $\theta_\epsilon := (1+\epsilon) \times Obj(t_{\text{ref}}, \mathbf{x}, \mathbf{y})$ to denote the threshold of the Rashomon set. Typically, the reference model is an empirical risk minimizer $t_{\text{ref}} \in \arg\min_{t \in \text{trees}} Obj(t, \mathbf{x}, \mathbf{y})$. Recent advances in decision tree optimization have allowed us to find this empirical risk minimizer, specifically using the GOSDT algorithm [31, 32]. Our goal is to store $R_{set}(\epsilon, t_{\text{ref}}, \mathcal{T})$, sample from it, and compute statistics from it.

Customized analytical bounds, leveraging tools from [31, 4], help reduce the search space for Rashomon set construction. As every possible tree is being grown in the process of dynamic programming, some of its leaves will have been determined ("fixed"), and others will not have yet been determined ("unfixed"). Bounds for these incomplete trees compare two quantities: the performance that might be achieved in the best possible case, when every unfixed part of the tree has perfect classification for all of its points, and θ_ϵ, the threshold of the Rashomon set. If the first is larger – i.e., worse – than θ_ϵ, we know that extensions of the tree will never be in the Rashomon set. Each partial tree t (that is, a tree that can be extended) is represented in terms of five variables: t_{fix} are the "fixed" leaves we do not extend during this part of the search (there will be a different copy of the tree elsewhere in the search where these leaves will potentially be split), δ_{fix} are the labels of the data points within the fixed leaves, t_{split} are the "unfixed" leaves we could potentially split during the exploration of this part of the search space (with labels δ_{split}) and H_t is the number of leaves in the current tree. Thus, our current position in the search space is a partial tree $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, H_t)$. The theorems below allow us to exclude large portions of the search space.

Theorem 3.1. *(Basic Rashomon Lower Bound) Let θ_ϵ be the threshold of the Rashomon set. Given a tree $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, H_t)$, let $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) := \ell(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \lambda H_t$ denote the lower bound of the objective for tree t. If $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, then the tree t and all of its children are not in the ϵ-Rashomon set.*

We can tighten the basic Rashomon lower bound by using knowledge of *equivalent points* [39]. Data points are equivalent if they have exactly the same feature values. Let Ω be a set of leaves. *Capture* is an indicator function that equals 1 if $\mathbf{x}_i$ falls into one of the leaves in Ω, and 0 otherwise, in which case we say that $\text{cap}(\mathbf{x}_i, \Omega) = 1$. Let e_u be a set of equivalent points and q_u be the minority class label among points in e_u. A dataset consists of multiple sets of equivalent points. Let $\{e_u\}_{u=1}^U$ enumerate these sets. The bound below incorporates equivalent points.

Theorem 3.2. *(Rashomon Equivalent Points Bound) Let θ_ϵ be the threshold of the Rashomon set. Let t be a tree with leaves $t_{\text{fix}}, t_{\text{split}}$ and lower bound $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y})$. Let $b_{equiv}(t_{\text{split}}, \mathbf{x}, \mathbf{y}) := \frac{1}{n}\sum_{i=1}^n \sum_{u=1}^U cap(\mathbf{x}_i, t_{\text{split}}) \wedge \mathbb{1}[\mathbf{x}_i \in e_u] \wedge \mathbb{1}[y_i = q_u]$ be the lower bound on the misclassification loss of the unfixed leaves. Let $B(t, \mathbf{x}, \mathbf{y}) := b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + b_{equiv}(t_{\text{split}}, \mathbf{x}, \mathbf{y})$ be the Rashomon lower bound of t. If $B(t, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, tree t and all its children are not in the ϵ-Rashomon set.*

We can use this bound recursively on all subtrees we discover during the process of dynamic programming. If, at any time, we find the sum of the lower bounds of subproblems created by a split

exceed the threshold of the Rashomon set, the split that led to these subproblems will never produce any member of the Rashomon set. This is formalized in the Rashomon Equivalent Points Bound for Subtrees, Theorem E.1 in Appendix E, which dramatically helps reduce the search space. We also use the "lookahead" bound, used in GOSDT [31], which looks one split forward and cuts all the children of tree t if $B(t, \mathbf{x}, \mathbf{y}) + \lambda > \theta_\epsilon$.

4 Storing, Extracting, and Sampling the Rashomon Set

The key to TreeFARMS's scalability is a novel *Model Set* representation. The **Model Set (MS)** is a set of hierarchical maps; each map is a **Model Set Instance (MSI)**. Conceptually, we identify a MSI by a <subproblem, objective> pair; in reality, we use pointers to improve execution time and reduce memory consumption. A MSI can represent a terminal (leaf) node, an internal node, or both (See Appendix A). A leaf MSI stores only the subproblem's prediction and the number of false positives and negatives (or false predictions of each class in the case of multiclass classification). An internal MSI, M, is a map whose keys are the features on which to split the subproblem and whose values are an array of pairs, each referring to left and right MSIs whose objectives sum to the objective of M. TreeFARMS's efficiency stems from the fact that the loss function for decision trees takes on a discrete number of values (approximately equal to the number of samples in the training data set), while the number of trees in the Rashomon set is frequently orders of magnitude larger. Therefore, many trees (and subtrees) have the same objective. By grouping together trees with the same objective, we avoid massive amounts of data duplication and computation. See Appendix A for an example. Equipped with these data structures, we now present our main algorithm.

4.1 TreeFARMS Implementation

We implement TreeFARMS in GOSDT [31], which uses a dynamic-programming-with-bounds formulation to find the optimal sparse decision trees. Each *subproblem* is defined by a support set $s_a \in \{0, 1\}^n$ such that the ith element is 1 iff $\mathbf{x}_i$ satisfies the Boolean assertion a that corresponds to a decision path in the tree. For each subproblem in the dynamic program, GOSDT keeps track of upper and lower bounds on its objective. It stores these subproblems and their bounds in a *dependency graph*, which expresses the relationships between subproblems. TreeFARMS transforms GOSDT in two key ways. First, while searching the space, TreeFARMS prunes the search space by removing only those subproblems whose objective lower bound is greater than the thresholds defined by the Rashomon set bound, θ_ϵ, rather than GOSDT's objective-based upper bound. Second, rather than finding the *single* best model expressed by the dependency graph, TreeFARMS returns *all models in the Rashomon set defined by* θ_ϵ.

4.2 Extraction Algorithm

TreeFARMS (Alg. 1) constructs the dependency graph using the bounds from Section 3, and Extract (Alg. 2) extracts the Rashomon set from the dependency graph.

TreeFARMS (Algorithm 1): **Line 1**: Call GOSDT to find the best objective. **Line 2**: Using the best objective from Line 1, compute θ_ϵ, as defined in Definition 1. **Lines 3-6**: Configure and execute (the modified) GOSDT to produce a dependency graph containing all subproblems in the Rashomon set. **Lines 7-10**: Initialize the parameters needed by `extract` and then call it to extract the Rashomon set from the dependency graph.

Extract (Algorithm 2): We present an abbreviated version of the algorithm here with the full details in Appendix B. **Line 1**: Check to see if we already have the Rashomon set for the given problem and scope; if so, return immediately. **Lines 2-3**: If we can make a prediction for the given subproblem that produces loss less than or equal to scope (using Theorems 3.1 and 3.2), then the leaf for this subproblem should be part of trees in the Rashomon set, so we add it to our Model Set. **Lines 4-12**: Loop over each feature and consider splitting the current problem on that feature. **Lines 5-6**: Skip over any splits that either do not appear in the dependency graph or whose objectives produce a value greater than scope (using Theorem E.1). **Lines 7-9**: Find all subtrees for left and right that should appear in trees in the Rashomon set, and construct the set of MSI identifiers for each. **Lines 10-12**: Now, take the cross product of the sets of MSI identifiers. For each pair, determine if the sum of the objectives for those MSI are within scope (using Theorem E.1). If so, we add the left/right pair to

the appropriate MSI, creating a new MSI if necessary. When this loop terminates, all trees in the Rashomon set are represented in MS.

4.3 Sampling from the Rashomon Set

If we can store the entire Rashomon set in memory, then sampling is unnecessary. However, sometimes the set is too large to fit in memory (e.g., the COMPAS data set [40] with a regularization of 0.005 and a Rashomon threshold that is within 15% of optimal produces 10^{12} trees). Our Model Set representation permits easy uniform sampling of the Rashomon set that can be used to explore the set with a much lower computational burden. Appendix C presents a sampling algorithm.

Algorithm 1 TreeFARMS$(\mathbf{x}, \mathbf{y}, \lambda, \epsilon) \rightarrow R_{\text{set}}$

// Given a dataset $(\mathbf{x}, \mathbf{y})$, λ, and ϵ, return the set, R_{set}, of all trees whose objective is in θ_ϵ.
opt $\leftarrow$ *gosdt.optimal_obj*$(\mathbf{x}, \mathbf{y}, \lambda)$ *// Use GOSDT to find opt, the objective of the optimal tree.*
$\theta_\epsilon \leftarrow opt * (1+\epsilon)$ *// Compute θ_ϵ, which is the threshold of the Rashomon Set.*
gosdt.best_current_obj $\leftarrow \theta_\epsilon$ *// Set and fix GOSDT's best current objective R^c to θ_ϵ*
// Disable leaf accuracy bound and incremental accuracy bound in GOSDT (see Algorithm 7 in [31]), which are used to find optimal trees but could remove near-optimal trees in the Rashomon set
gosdt.leaf_accuracy $\leftarrow False$ *// Disable leaf accuracy bound.*
gosdt.incremental_accuracy $\leftarrow False$ *// Disable incremental accuracy bound.*
gosdt.fit$(\mathbf{x}, \mathbf{y}, \lambda)$ *// Run the branch-and-bound algorithm using new settings of bounds.*
$G \leftarrow$ *gosdt.get_graph*$()$ *// Return dependency graph obtained from branch-and-bound.*
global $MS \leftarrow \{\}$ *// Initialize Rashomon Model Set MS, which is a global data structure.*
$P \leftarrow ones(|\mathbf{y}|)$ *// Create the subproblem representation for the entire dataset.*
extract(G, P, θ_ϵ) *// Fill in MS with trees in the Rashomon set.*
return MS_P

Algorithm 2 extract$(G, sub, scope)$ (Detailed algorithm in Appendix B)

// Given a dependency graph, G; a subproblem, sub; and a maximum allowed objective value, $scope$, populate the global variable MS with the Rashomon set for sub within scope.
// Check if we have already solved the subproblem sub. SOLVED is presented in Alg. 3.
if SOLVED$(MS, sub, scope)$ **then return**
if $G[sub] \leq scope$ **then** *// Check if we should create a leaf. (Theorems 3.1 and 3.2).*
  $MS \leftarrow MS \cup$ *newLeaf*(sub) *// newLeaf is presented in Alg. 3.*
// Consider splits on each feasible split feature skipping those not in G or with bounds too large.
for each feature $j \in [1, M]$ **do**
  $sub_l, sub_r \leftarrow split(sub, j)$
  if either sub_l, sub_r **not in** G **or** $G[sub_l] + G[sub_r] > scope$ **then continue**
  // Find Model Sets Instances for left and right.
  extract$(G, sub_l, scope - G[sub_r])$
  extract$(G, sub_r, scope - G[sub_l])$
  left, right $\leftarrow MS_{sub_l}, MS_{sub_r}$
  for each $(m_l, m_r) \in$ (*left* $\times$ *right*) **do** *// Consider cross product of left/right MSI.*
    // Skip trees with objective outside of scope. (Theorem E.1).
    if $obj(m_l) + obj(m_r) > scope$ **then continue**
    $MS \leftarrow MS \cup add(sub, obj(m_l) + obj(m_r), m_l, m_r)$ *// Add pair to Model Set.*
// MS now contains all MSI for sub with objective less than or equal to scope.
return

5 Applications of the Rashomon Set

Besides allowing users an unprecedented level of control over model choice, having access to the Rashomon set unlocks powerful new capabilities. We present three example applications here.

5.1 Variable Importance for Models in the Rashomon Set via Model Class Reliance

The problem with classical variable importance techniques is that they generally provide the importance of one variable to one model. However, just because a variable is important to one model does not mean that it is important in general. To answer this more general question, we consider model class reliance (MCR) [2]. MCR provides the range of variable importance values across the set of all well-performing models. MCR_- and MCR_+ denote the lower and upper bounds of this range, respectively. A feature with a large MCR_- is important in all well-performing models; a feature with a small MCR_+ is unimportant to every well-performing model. Past work managed to calculate MCR_- only for convex loss in linear models [2]. For decision trees, the problem is nonconvex and intractable using previous methods. A recent work [41] estimates MCR for random forest, leveraging the fact that the forest's trees are grown greedily from the top down; the optimized sparse trees we consider are entirely different, and as we will show, many good sparse trees are difficult to obtain through sampling. Since TreeFARMS can enumerate the whole Rashomon set of decision trees, we can *directly* calculate variable importance for every tree in the Rashomon set and then find its minimum and maximum to compute the MCR, as described in Appendix D. See Section 6.2 for the results. If the Rashomon set is too large to enumerate, sampling (Section 4.3) can be used to obtain sample estimates for the MCR (shown in Section 6.2).

5.2 Rashomon Sets Beyond Accuracy: Constructing the Rashomon Set for Other Metrics

For imbalanced datasets, high accuracy is not always meaningful. Metrics such as balanced accuracy and F1-score are better suited for these datasets. We show that, given the Rashomon set constructed using accuracy, we can directly find the Rashomon sets for balanced accuracy and F1-score.

Let q^+ be the proportion of positive samples and q^- be the proportion of negative samples, i.e., $q^+ + q^- = 1$. Let $q_{\min} := \min(q^+, q^-)$ and $q_{\max} := \max(q^+, q^-)$. Let FPR and FNR be the false positive and false negative rates. We define the Accuracy Rashomon set as $A_\theta := \{t \in \mathcal{T} : q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \theta\}$, where θ is the objective threshold of the Accuracy Rashomon set, similar to θ_ϵ in Section 3. The next two theorems guide us to find the Balanced Accuracy or F1-Score Rashomon set from the Accuracy Rashomon set. Their proofs are presented in Appendix E. We use δ in these theorems to denote the objective threshold of Balanced Accuracy or F1-Score Rashomon sets.

Theorem 5.1. *(Accuracy Rashomon set covers Balanced Accuracy Rashomon set) Let $B_\delta := \{t \in \mathcal{T} : \frac{FPR_t + FNR_t}{2} + \lambda H_t \leq \delta\}$ be the Balanced Accuracy Rashomon set. If*

$$\theta \geq \min\Big(2q_{\max}\delta, q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda 2^d\Big),$$

where d is the depth limit, then $\forall t \in B_\delta, t \in A_\theta$.

Theorem 5.2. *(Accuracy Rashomon set covers F1-Score Rashomon set) Let*

$$F_\delta := \left\{t \in \mathcal{T} : \frac{q^- FPR_t + q^+ FNR_t}{2q^+ + q^- FPR_t - q^+ FNR_t} + \lambda H_t \leq \delta\right\}$$

be the F1-score Rashomon set. Suppose $q^+ \in (0,1)$, $q^- \in (0,1)$, and $\delta - \lambda H_t \in (0,1)$. If $\theta \geq \min\Big(\max\big(\frac{2q^+\delta}{1-\delta}, \frac{2q^+(\delta - \lambda 2^d)}{1-(\delta-\lambda 2^d)} + \lambda 2^d\big), \mathbb{1}[\delta < \sqrt{2} - 1]\frac{2\delta}{1+\delta} + \mathbb{1}[\delta \geq \sqrt{2} - 1](\delta + 3 - 2\sqrt{2})\Big)$, then $\forall t \in F_\delta$, $t \in A_\theta$.

We can use Theorems 5.1 and 5.2 above to find all trees in the Balanced Accuracy or F1-score Rashomon set directly by searching through the Accuracy Rashomon set with objective threshold θ that satisfies the inequality constraint. In our implementation, we first use GOSDT to find the optimal tree w.r.t. misclassification loss and then calculate its objective w.r.t. balanced accuracy and F1-score. Then we set δ and the corresponding θ. This guarantees that the Rashomon sets for balanced accuracy and F1-score objective are not empty. Experiments in Section 6.3 illustrate this calculation.

5.3 Sensitivity to Missing Groups of Samples

Though sparse decision trees are usually robust (since predictions are made separately in each leaf), we are also interested in how a sample or a group of samples influences all well-performing models

(i.e., whether this subset of points is *influential* [42].) Influence functions cannot be calculated for decision trees since they require differentiability. The following two theorems help us find optimal or near-optimal trees for a dataset in which a group of instances has been removed by searching through the Rashomon set obtained from the full dataset. Here, we consider the misclassification loss.

Let $\tilde{t}^*$ be the optimal tree trained on the reduced dataset $\{\mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}\}$ where K is a set of indices of instances that we remove from the original dataset. Let $|K|$ denote the cardinality of the set K. Overloading notation to include the dataset, let $R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y}) = R_{set}(\epsilon, t^*, \mathcal{T})$ (see Eq (1)) be the Rashomon set of the original dataset, where t^* is the optimal tree trained on the original dataset. We define the ϵ'-Rashomon set on the reduced dataset as

$$R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) := \left\{ t \in \mathcal{T} : Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) \leq (1+\epsilon') \times Obj(\tilde{t}^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) \right\}.$$

Theorem 5.3. *(Optimal tree after removing a group of instances is still in full-dataset Rashomon set) If* $\epsilon \geq \frac{2|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$*, then* $\tilde{t}^* \in R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.

Now we consider not only the optimal tree on the reduced dataset but also the near-optimal trees.

Theorem 5.4. *(Rashomon set after removing a group of instances is within full-dataset Rashomon set) If* $\epsilon \geq \epsilon' + \frac{(2+\epsilon')|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$*, then* $\forall t \in R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}), t \in R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.

6 Experiments

Our evaluation answers the following questions: 1. How does TreeFARMS compare to baseline methods for searching the hypothesis space? (§6.1), 2. How quickly can we find the entire Rashomon set? (§6.1), 3. What does the Rashomon set look like? What can we learn about its structure? (§G.2), 4. What does MCR look like for real datasets? (§6.2), 5. How do balanced accuracy and F1-score Rashomon sets compare to the accuracy Rashomon set? (§6.3), and 6. How does removing samples affect the Rashomon set? (§6.4).

Finding the Rashomon set is computationally difficult due to searching an exponentially growing search space. We use datasets from the UCI Machine Learning Repository [Car Evaluation, Congressional Voting Records, Monk2, and Iris, see 43], a penguin dataset [44], a criminal recidivism dataset [COMPAS, shared by 40], the Fair Isaac (FICO) credit risk dataset [45] used for the Explainable ML Challenge, and four coupon datasets (Bar, Coffee House, Cheap Restaurant, and Expensive Restaurant) [46] that come from surveys. More details are in Appendix F.

6.1 Performance and Timing Experiments

Our method directly constructs the Rashomon set of decision trees of a dataset. While, to the best of our knowledge, there is no previous directly comparable work, there are several methods one might naturally consider to find this set. One might use methods that sample from the high-posterior region of tree models, though we would not know how many samples we need to extract the full Rashomon set. Thus, the first baseline method we consider is sampling trees from the posterior distribution of Bayesian Additive trees [47, 48]. We used the R package BART [37], setting the number of trees in each iteration to 1. Many ensemble methods combine a diverse set of trees. The diversity in this set comes from fitting on different subsets of data. Trees produced by these methods would be natural approaches for finding the Rashomon set. We thus generated trees from three different methods (Random Forest [38], CART [20], and GOSDT [31]), on many subsets of our original data.

Figure 1 compares the Rashomon set with the four baselines on the Monk2, COMPAS, and Bar datasets. We show the number of distinct trees versus the objective value. We sort the trees with respect to their objective values, so all methods show an increasing trend. TreeFARMS (in purple) found ***orders of magnitude more distinct trees in the Rashomon set*** than any of the four baselines on all of the datasets. The baseline methods tend to find many duplicated trees. For example, in 46 seconds, BART finds only 488 distinct trees on Monk2, whereas TreeFARMS found 10^8. Other methods find ∼20,000 distinct trees. Most trees found by the baselines are not even in the Rashomon set, i.e., most of their trees have objective values higher than the threshold of the Rashomon set.

Figure 2 shows run times, specifically, Fig. 2(c) shows that TreeFARMS ***finds trees in the Rashomon set at a dramatically faster rate*** than the baselines. Appendix G.1 has more results.

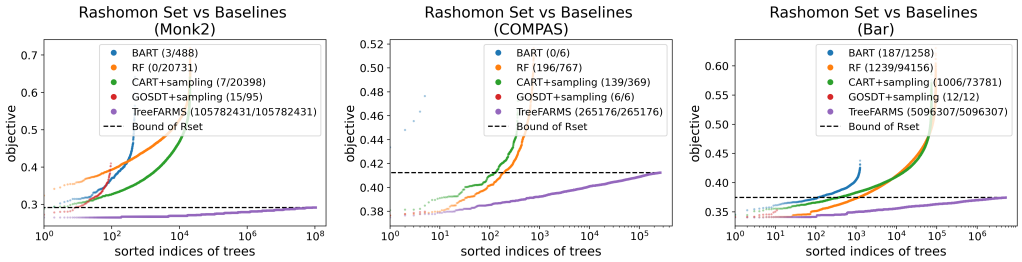
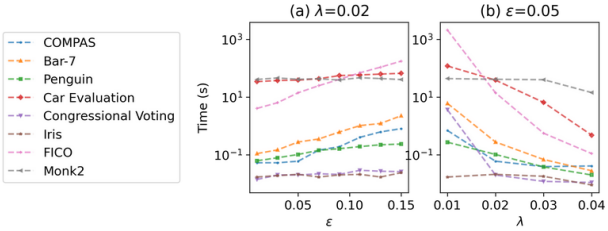


Figure 1: Comparison of trees in the Rashomon set ($\lambda = 0.01, \epsilon = 0.1$) and trees generated by baselines. Trees in the Rashomon set have objective below the dashed line. (A/B) in legend represents that A trees among B trees trained by the baseline are in the Rashomon set. For example, RF (196/767) means 196 trees among 767 distinct trees trained by Random Forest are in the Rashomon set. Indices are in log scale to accommodate differences in orders of magnitude of tree counts among methods.

The takeaway from this experiment is that ***natural baselines find at best a tiny sliver of the Rashomon set***. Further, ***the way we discovered this was to develop a method that actually enumerates the Rashomon set***. We would not have known, using any other way we could think of, that sampling-based approaches barely scratch the surface of the Rashomon set.



(c) # trees found in the Rashomon set per second

Algorithm \ Dataset	Monk2	COMPAS	Bar
TreeFARMS	**2.28×10^6**	**87459.10**	**19102.16**
BART	0.03 ± 0.02	0.14 ± 0.17	0.15 ± 0.29
RF	0.00 ± 0.00	68.67 ± 2.34	4.78 ± 0.14
CART + sampling	0.20 ± 0.11	44.83 ± 2.17	3.88 ± 0.11
GOSDT + sampling	0.31 ± 0.04	2.20 ± 0.34	0.06 ± 0.01

Figure 2: (a), (b) Run times for computing Rashomon sets as a function of ϵ and λ respectively. (c) Number of trees in the Rashomon set found by each method per second. The total time is approximately 46, 3, and 270 seconds for Monk2, COMPAS, and Bar respectively. TreeFARMS is the only algorithm *guaranteed* to find all trees in the Rashomon set. (Appendix G.1 has results for all the datasets.)

6.2 Variable Importance: Model Class Reliance

Without TreeFARMS, it has not been possible to compute overall variable importance calculations such as MCR for complex function classes with interaction terms such as decision trees. Here, we exactly compute MCR on the COMPAS and Bar datasets (see Figure 3). For the COMPAS dataset (left subfigure)features related to prior counts generally have high MCR_+, which means these features are very important for some trees in the Rashomon set. For the Bar dataset (right subfigure), features "Bar_1-3" and "Bar_4-8" have dominant MCR_+ and MCR_- compared with other features, indicating that for all well-performing trees, these features are the most important. This makes sense, since people who go to bars regularly would be likely to accept a coupon for a bar.

Sampling for MCR: Sampling has a massive memory benefit over storing the whole Rashomon set, because we do not need to store the samples. Since MCR requires computing extreme value statistics (max and min over the Rashomon set), it poses a test for the sampling technique posed in Section 4.3. Figure 3 shows sampled MCR and its convergence to true MCR. 25% of the samples are usually sufficient. More results are shown in Appendix G.3.

6.3 Balanced Accuracy and F1-score Rashomon set from Accuracy Rashomon set

As discussed in Section 5.2, Rashomon sets of balanced accuracy and F1-Score are contained in the Rashomon set for accuracy. Figure 4 shows trees in the Accuracy Rashomon set, which covers the Balanced Accuracy Rashomon set (left) and F1-score Rashomon set (right). The black dashed line indicates the corresponding objective thresholds and blue dots below the dashed line are trees

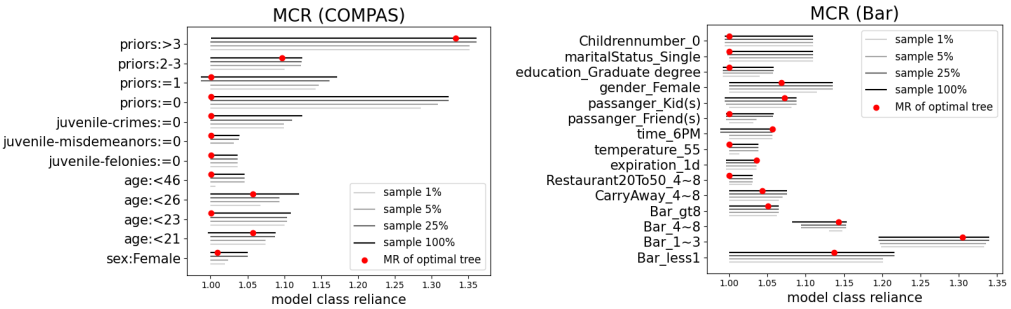


Figure 3: Variable Importance: Model class reliance on the COMPAS and Bar ($\lambda = 0.01, \epsilon = 0.05$). Red dots indicate the model reliance (variable importance) calculated from the optimal tree. Each line connects MCR_- and MCR_+ showing the range of variable importance among all good models.

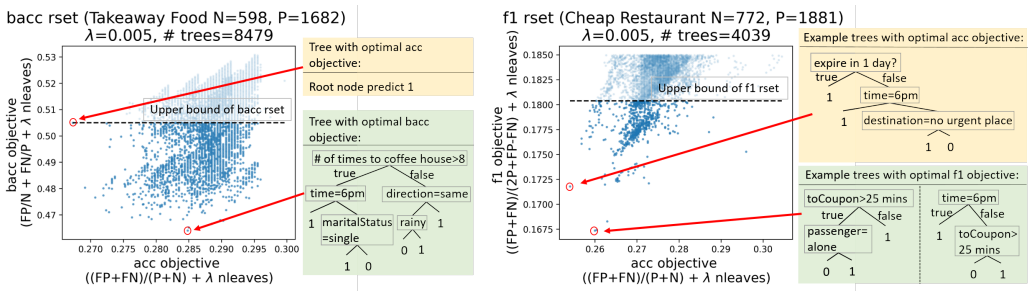


Figure 4: Example of Balanced Accuracy Rashomon set (left) and F1-Score Rashomon set (right). # trees indicates the number of trees in the Balanced Accuracy or F1-score Rashomon set. Trees in the yellow region have optimal accuracy objective and trees in the green region have optimal balanced accuracy or F1-score objective.

in these Rashomon sets. Note that the tree with the minimum misclassification objective is not the tree that optimizes other evaluation metrics. For example, in the left subfigure, a single root node that predicts all samples as 1 has the optimal accuracy objective (in yellow), while another 6-leaf tree minimizes the balanced accuracy objective (in green). Actually, many trees have better balanced accuracy objective than the tree that minimizes the accuracy objective. A similar pattern holds for the F1-score Rashomon set (see right subfigure). Some trees that have worse accuracy objectives are better in terms of the F1-score objective. More figures are shown in Appendix G.4.

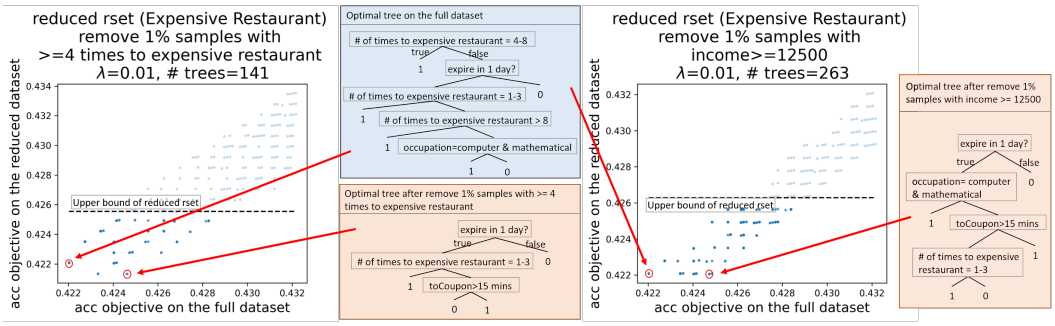


Figure 5: Example Rashomon sets and optimal trees after we remove the 1% of samples with "number of times to expensive restaurant ≥ 4" (left) and "income $\geq$ \$12,500" (right) on the Expensive Restaurant dataset. The optimal tree on the full dataset is shown in the gray region and optimal trees on the corresponding reduced datasets are in the orange region.

6.4 Rashomon set after removing a group of samples

Figure 5 shows accuracy objective on the full dataset versus the objective on the reduced dataset after 1% of samples with "number of times to expensive restaurant ≥ 4" (left) and "income $\geq$ \$12,500" (right) are removed. The black dashed line indicates the objective threshold of the reduced Rashomon set and blue dots below the dashed line are trees in the reduced Rashomon set. As we can see, both scatter plots show a high correlation between the accuracy objective on the full dataset and the reduced dataset, indicating sparse near-optimal trees are robust to the shift in sample distribution. In other words, well-performing trees trained on the full dataset are usually still well-performing if some samples are removed. Optimal trees on the reduced dataset might be different, as we see by comparing the trees in the orange region and blue region. More results are shown in Appendix G.5.

7 Conclusion

This work opens the door to interesting discussions on variable importance, distributional shift, and user options. By efficiently representing *all* optimal and slightly suboptimal models for complex nonlinear function classes with interactions between variables, we provide a range of new user-centered capabilities for machine learning systems, and a new understanding of the importance of variables. Importantly, TreeFARMS allows users a *choice* rather than handing them a single model.

Acknowledgements

We thank the anonymous reviewers for their suggestions and insightful questions. The authors acknowledge funding from the National Science Foundation under grants IIS-2147061 and IIS-2130250, National Institute on Drug Abuse under grant R01 DA054994, Department of Energy under grants DE-SC0021358 and DE-SC0023194, and National Research Traineeship Program under NSF grants DGE-2022040 and CCF-1934964. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). Nous remercions le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) de son soutien.

Code Availability

Implementations of TreeFARMS is available at https://github.com/ubc-systopia/treeFarms.

References

[1] Lesia Semenova, Cynthia Rudin, and Ronald Parr. On the existence of simpler machine learning models. In *ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*, 2022.

[2] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[3] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001.

[4] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

[6] Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, pages 6765–6774. PMLR, 2020.

[7] Theja Tulabandhula and Cynthia Rudin. Robust optimization using machine learning for uncertainty sets. *arXiv preprint arXiv:1407.1097*, 2014.

[8] Beau Coker, Cynthia Rudin, and Gary King. A theory of statistical inference for ensuring the robustness of scientific results. *Management Science*, 2021.

[9] Jiayun Dong and Cynthia Rudin. Exploring the cloud of variable importance for the set of all good models. *Nature Machine Intelligence*, 2(12):810–824, 2020.

[10] Emilie Danna, Mary Fenelon, Zonghao Gu, and Roland Wunderling. Generating multiple solutions for mixed integer programming problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 280–294. Springer, 2007.

[11] Izuwa Ahanor, Hugh Medal, and Andrew C Trapp. Diversitree: Computing diverse sets of near-optimal solutions to mixed-integer optimization problems. *arXiv preprint arXiv:2204.03822*, 2022.

[12] Nicholas Kissel and Lucas Mentch. Forward stability and model path selection. *arXiv preprint arXiv:2103.03462*, 2021.

[13] Salvatore Ruggieri. Enumerating distinct decision trees. In *International Conference on Machine Learning*, pages 2960–2968. PMLR, 2017.

[14] Salvatore Ruggieri. Complete search for feature selection in decision trees. *Journal of Machine Learning Research*, 20:104–1, 2019.

[15] Satoshi Hara and Takanori Maehara. Enumerate lasso solutions for feature selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[16] Satoshi Hara and Masakazu Ishihata. Approximate and exact enumeration of rule models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[17] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963.

[18] Harold J. Payne and William S. Meisel. An algorithm for constructing optimal binary decision trees. *IEEE Transactions on Computers*, 26(09):905–916, 1977.

[19] Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348, 2014.

[20] L Breiman, JH Friedman, R Olshen, and CJ Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[21] J Ross Quinlan. C 4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning*, 1993.

[22] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[23] Kristin P Bennett and Jennifer A Blue. Optimal decision trees. *Rensselaer Polytechnic Institute Math Report*, 214:24, 1996.

[24] Alireza Farhangfar, Russell Greiner, and Martin Zinkevich. A fast way to produce near-optimal fixed-depth decision trees. In *Proceedings of the 10th international symposium on artificial intelligence and mathematics (ISAIM-2008)*, 2008.

[25] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[26] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1625–1632, 2019.

[27] Matheus Guedes Vilas Boas, Haroldo Gambini Santos, Luiz Henrique de Campos Merschmann, and Greet Vanden Berghe. Optimal decision trees for the algorithm selection problem: integer programming based approaches. *International Transactions in Operational Research*, 28(5):2759–2781, 2021.

[28] Oktay Günlük, Jayant Kalagnanam, Minhan Li, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, pages 1–28, 2021.

[29] Sina Aghaei, Andrés Gómez, and Phebe Vayanos. Strong optimal classification trees. *arXiv preprint arXiv:2103.15965*, 2021.

[30] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, Joao Marques-Silva, and IS RAS. Learning optimal decision trees with SAT. In *IJCAI*, pages 1362–1368, 2018.

[31] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[32] Hayden McTavish, Chudi Zhong, Reto Achermann, Ilias Karimalis, Jacques Chen, Cynthia Rudin, and Margo Seltzer. Fast sparse decision tree optimization via reference ensembles. In *AAAI Conference on Artificial Intelligence*, 2022.

[33] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3146–3153, 2020.

[34] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: optimal classification trees via dynamic programming and search. *Journal of Machine Learning Research*, 23(26):1–47, 2022.

[35] Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Mathematical optimization in classification and regression trees. *Top*, 29(1):5–33, 2021.

[36] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[37] Rodney Sparapani, Charles Spanbauer, and Robert McCulloch. Nonparametric machine learning and efficient computation with Bayesian additive regression trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66, 2021.

[38] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[39] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18:1–78, 2018.

[40] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. How we analyzed the COMPAS recidivism algorithm. *ProPublica*, 2016.

[41] Gavin Smith, Roberto Mansilla, and James Goulding. Model class reliance for random forests. *Advances in Neural Information Processing Systems*, 33:22305–22315, 2020.

[42] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[43] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[44] Kristen B Gorman, Tony D Williams, and William R Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PloS one*, 9(3):e90081, 2014.

[45] FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. Explainable Machine Learning Challenge. `https://community.fico.com/s/explainable-machine-learning-challenge`, 2018.

[46] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[47] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[48] Hugh A Chipman, Edward I George, and Robert E McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization. *J Mach. Learn. Res*, 22:1–73, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] TreeFARMS's models should not be interpreted as causal. TreeFARMS creates sparse binary decision trees and thus has limited capacity. TreeFARMS's models inherit flaws from data it was trained on. TreeFARMS is not yet customized to a given application, which can be done in future work.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We cannot think of any.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] see Appendix E
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see Appendix F and G
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Appendix F and G
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Appendix G
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Appendix F and G
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] The code and data we used all have MIT licenses.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our code in the supplementary material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Table of Contents

A The Model Set Representation **16**

B The TreeFARMS Algorithm **17**

C Sampling from the Rashomon Set **19**

D Model Class Reliance of Decision Trees **19**

E Theorems and Proofs **21**

F Experimental Datasets **29**
F.1 Preprocessing 30

G More Experimental Results **31**
G.1 Scalability and Efficiency of Calculating Rashomon set with TreeFARMS versus Baselines 31
G.2 Landscape of the Rashomon set 32
G.3 Variable Importance: Model Class Reliance 33
G.4 Balanced Accuracy and F1-score Rashomon set from Accuracy Rashomon set . . 34
G.5 Rashomon set after removing a group of samples 36

A The Model Set Representation

In Section 4, we described the Model Set representation for the Rashomon Set. Figures 6 and 7 illustrate these data structures in more detail. Figure 6 introduces major components of our Model Set Instance (MSI) representation. Figure 7 provides an example of the representation on a toy dataset.

The Model Set representation's memory efficiency stems from two key properties: grouping models for the same subproblem together and then referencing these subproblems rather than duplicating them. Figure 7 illustrates this for a tiny, 10-sample dataset. Let's say that our Rashomon threshold is 0.40 and that we can split the entire problem on some feature to produce left and right children, each containing exactly half the data set – the first five samples for the left subproblem and the last five samples for the right subproblem.

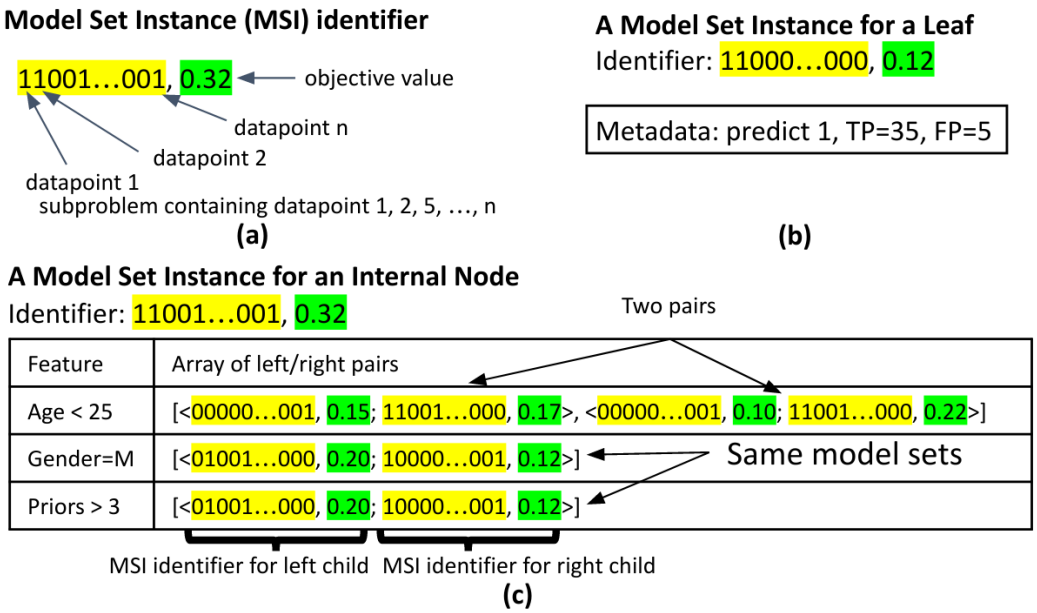


Figure 6: Model Set Instance (MSI) Representation. Each MSI represents a subproblem (yellow)/objective (green) pair (a). The subproblem is described by a bitvector identifying the samples in the subproblem. Figure (b) represents a leaf subproblem with objective 0.12. Figure (c) represents a subproblem that can be split to produce an objective of 0.32. There are three features (age, gender, and priors) for which a split produces this objective. If we split on age<25, then there are two different pairs of identifiers that both produce objectives of 0.32 (represented by the two pairs of MSI identifiers). Each pair of identifiers would further split on different features and have different tree structures. The splits on gender and priors happen to produce the same pair of model set instances, as shown by the matching MSI identifiers in the bottom two lines of the table. Referencing these sets of trees by MSIs avoids massive data duplication.

extract (see Algorithm 2, 3) looks up the left subproblem and finds that there are three MSIs for it, L1 represents a leaf and L2 and L3 each represent internal nodes with different objectives, each of which produce that objective in two ways. Thus, on the left, there are five possible trees. On the right, we have only two MSIs, one leaf (R1) and one internal node (R2). These represent three trees. Although the cross-product of the left and right children produce fifteen trees, we need only consider the 6 possible objectives values produced by combining each pair of model sets (i.e., L1+R1, L1+R2, L2+R1, etc.). All but one of these satisfy the Rashomon threshold of 0.40 (L1+R1 produces an objective of 0.50, which is greater than the threshold), and they produce only three new model sets: one with objective 0.40 containing the two trees produced by combining L1 and R2 and the two trees produced by combining L2 and R1; one with objective 0.30 containing the four trees produced by combining L2 and R2 and the two trees produced by combining L3 and R1; and one with the objective 0.20 containing the four trees represented by combining L3 and R2.

All Model set instances for left child: subproblem 1111100000

L1. Identifier: 1111100000, 0.30
Leaf: Predicts T; 1 FP, 2 FN

L2. Identifier: 1111100000, 0.20

Feature	Array of left/right pairs
F1	[<1111000000, 0.15; 0000100000, 0.05>]
F2	[<0011000000, 0.10; 1100100000, 0.10>]

L3. Identifier: 1111100000, 0.10

Feature	Array of left/right pairs
F3	[<1110000000, 0.08; 0001100000, 0.02>]
F4	[<0010100000, 0.04; 1101000000, 0.06>]

All Model set instances for right child: subproblem 0000011111

R1. Identifier: 0000011111, 0.20
Leaf: Predicts F; 1 FP, 1 FN

R2. Identifier: 0000011111, 0.10

Feature	Array of left/right pairs
F5	[<0000010101, 0.03; 0000001010, 0.07>]
F6	[<0000011100, 0.09; 0000000011, 0.01>]

Figure 7: An example model set representation of the left and right children of a root split on a 10-sample dataset with Rashomon threshold of 0.40. In the left child, L1 represents a leaf, and L2 and L3 represent internal nodes with different objectives. In the right child, R1 represents a leaf, and R2 represents an internal node. Each of the internal nodes (L2, L3, R2) produce their objective in two ways. All children Model Set Instances of internal nodes (L2, L3, R2) contain only one tree. Thus, on the left, there are five possible trees, and the right represents three trees.

Our implementation augments the Model Set with two indexes: the **Subproblem Index (SI)** and the **Subproblem-Objective Index (SOI)**. The **Subproblem Index (SI)** is an index from a subproblem to the set of all MSI identifiers for that subproblem, with objectives under a certain value named *scope*. This value helps ensure that we do not extract MSI that are provably not in the Rashomon set using Theorem E.1. The **Subproblem-Objective Index (SOI)** maps from a subproblem/objective pair to an MSI. For example, in Figure 7, SI maps subproblem 1111100000 to identifiers of L1, L2, L3, and SOI maps the identifier of L1 to its metadata.

B The TreeFARMS Algorithm

In Section 4.2, we presented a condensed version of the extraction algorithm (Algorithm 2); Algorithm 3 presents the full detail.

We maintain global data structures, MS, the collection of all Model Set Instances in the hierarchical Model Set, SI the Subproblem Index, and SOI, the Subproblem-Objective Index as described in Section 4 and Appendix A. They are all initialized to be empty before calling the recursive procedure *extract*, shown below, on the full data set.

Let $MS_{s,o}$ be the MSI with objective o for subproblem s. We implement $MS_{s,o}$ using the Subproblem-Objective Index, SOI.

Let MS_s return a pair consisting of a scope and set of MSI identifiers, (*scope, msiSet*). The scope returned is the objective such that all trees with objective under that value are represented by *msiSet* for subproblem s. The set of MSI identifiers contains all MSI for subproblem s with objective less than or equal to *scope*. We implement MS_s using the Subproblem Index, SI.

More formally:

if $scope, \{msiSet\} \in MS_s$ **then**

$scope, \{msiSet\} \in SI[s]$ **and** $\forall\, m$ **in** $msiSet,\ obj(m) \leq scope$

When $extract$ returns, the Rashomon Set is represented by MS_p where p is the complete data set.

Algorithm 3 *extract*$(G, sub, scope)$

```
// Given a dependency graph, G, a subproblem, sub, and a scope, scope, populate MS with the
// Rashomon set for sub.
scope', msi_set' ← SI[sub]
// When we have solved a problem for a given scope value, we guarantee to have found all possible
// objectives less than or equal to that scope. This line implements SOLVED in Algorithm 2.
if scope' is not None and scope ≤ scope' then
    // This subproblem is already solved.
    return
msi_set ← {}
p ← G[sub] // Find problem sub in the dependency graph.
base_bound ← p.ub // Objective if this node is a leaf.
if base_bound ≤ scope then // It is possible for this subproblem to be a leaf
    // Select prediction that minimizes loss for this node. This line and the following implement
    // newLeaf in Algorithm 2.
    prediction ← CALCULATE_PREDICTION(p)
    // Create single leaf tree represented in an MSI. newMSI constructs an MSI using data
    // provided in its arguments.
    msi ← newMSI(leaf, sub, base_bound, prediction)
    msi_set ← msi_set ∪ {msi.identifier}
    SOI[sub, base_bound] ← msi
// Check all possible features on which we might split.
for each feature j ∈ [1, M] do
    // Create subproblems by splitting problem by feature j.
    sub_l, sub_r ← split(sub, j)
    // If either subproblem is not in dependency graph, we need not consider this split.
    if sub_l not in G or sub_r not in G then
        continue
    p_l ← G[sub_l] // p_l is a node in the dependency graph.
    p_r ← G[sub_r] // p_r is a node in the dependency graph.
    if p_l.lb + p_r.lb > scope then
        continue
    // Populate Model Sets for the left and right subproblems. Recursively call extract and
    // decrement scope using the lower bound of the other side.
    extract(G, sub_l, scope − p_r.lb)
    extract(G, sub_r, scope − p_l.lb)
    // MS now contains all Model Sets for the children; find all left and right model set
    // instances.
    left_scope, left_msi_set ← SI[sub_l]
    // Remove all MSI whose objectives are too big.
    left_msi_set ← {msi | msi ∈ left_msi_set and obj(msi) ≤ scope}
    right_scope, right_msi_set ← SI[sub_r]
    right_msi_set ← {msi | msi ∈ right_msi_set and obj(msi) ≤ scope}
    // For each pair of model set instances in the cross product of left and right, if the sum of
    // their objectives is less than or equal to scope, either create a new Model Set for this
    // problem/objective, or add this pair to an existing Model Set for this problem/objective pair.
    for each pair of model set instances (m_l, m_r) ∈ (left_msi_set × right_msi_set) do
        // Skip leaf trivial extensions from the output.
        if is_leaf(m_l) and is_leaf(m_r) and m_l.predicts == m_r.predicts then
            continue
        new_obj ← m_l.obj + m_r.obj
        if new_obj > scope then // Skip combinations that exceed current scope.
            continue
        msi ← SOI[sub, new_obj]
        if not exists msi then
            msi ← newMSI(internal, sub, new_obj)
        // Now, add this left/right pair to the model set for feature j.
        msi[j].append(< m_l, m_r >)
        msi_set ← msi_set ∪ {msi.identifier}
        SOI[sub, new_obj] ← msi
// We store solved instance should we need it later.
SI[sub] ← scope, msi_set
return
```

Algorithm 4 CALCULATE_PREDICTION(p) $\rightarrow$ *prediction*
// Compute the prediction of the given node.

// Prediction if we end this node as a leaf. Changes to 1 if there are more positives than negatives.
prediction $\leftarrow 0$
// Calculate the total positive and negative weights of each equivalence class.
// $s \in \{0,1\}^n$ is a bitvector indicating datapoints we are considering for this subproblem.
$s \leftarrow p.keys$
// Compute negatives using samples from only those points captured in this subproblem.
negatives $\leftarrow \sum_{i=1}^{n} \mathbb{1}[y_i = 0 \wedge s_i = 1]$
positives $\leftarrow \sum_{i=1}^{n} \mathbb{1}[y_i = 1 \wedge s_i = 1]$
// Set the leaf prediction based on class with the higher selected total weight.
if *negatives* $<$ *positives* **then**
    // Leaf predicts the majority class as 1 since positive weights are higher.
    // Ties are predicted negative w.l.o.g. since the error rate is the same either way.
    prediction $\leftarrow 1$
return *prediction*

C Sampling from the Rashomon Set

To facilitate sampling from Rashomon sets too large to materialize in memory, we add a small amount of metadata to our Model Set Representation. Each MSI for a splittable problem retains a count of the total number of trees (unique trees, not MSIs) as well as counts for each entry in the map for the MSI (that is, a count of the number of unique trees attributable to each possible split). Assume that $count(msi)$ returns the total number of trees represented by an entire Model Set Instance and $count(msi[j])$ returns the number of trees represented by a particular entry in the map of msi. Algorithms 5 and 6 present a brute-force algorithm for translating an index value between $[0, |R_{set}|)$ into a unique tree. To sample a tree, we randomly draw an index from the count of all trees (or the sum of count of all MSIs at the problem set level), and then call Algorithm 5.

Algorithm 5 *ndx_to_tree*(MS, ndx) $\rightarrow$ t

// Find the ndx^{th} tree in the Rashomon set represented by MS

// Get all MSI representing the full data set
msi_set, scope $\leftarrow$ MS_p
for msi **in** *msi_set* **do**
    if $count(msi) < ndx$ **then**
        break
    // Align indices to the start of the next MSI
    $ndx \leftarrow ndx - count(msi)$
// Either ndx is larger than the size of the Rashomon set or msi is the Model Set Instance
// in which we will find the tree corresponding to ndx.
if $ndx \geq count(msi)$ **then**
    // No tree with this ndx exists.
    return NULL
return *ndx_to_tree_in_msi*(msi, ndx)

D Model Class Reliance of Decision Trees

In this appendix, we present the model class reliance calculation in detail. Given a dataset $\{\mathbf{x}, \mathbf{y}\}$, the error of a decision tree t is defined by a nonnegative loss function ℓ:

$$e_{\text{orig}}(t) := \frac{1}{n} \sum_{i=1}^{n} \ell(t, \mathbf{y}_{[i]} \mathbf{x}_{[i,k]}, \mathbf{x}_{[i,\setminus k]}), \tag{2}$$

Algorithm 6 *ndx_to_tree_in_msi*$(msi, ndx) \rightarrow t$

// Recursive procedure to construct the ndx^{th} *tree in Model Set Instance* msi

if $has_terminal(msi)$ **then**
  // this is a leaf; see if we want to return it, or a more complicated tree (with the same
  // objective).
  if $ndx = 0$ **then**
    return *make_leaf_node*$(msi($*prediction*$))$
  // Continue on to a more complicated model, but account for the leaf model.
  $ndx \leftarrow ndx - 1$

// Iterate over features to determine in which feature this index appears.
for f **in** $msi.keys()$ **do**
  if $count(msi[f]) < ndx$ **then**
    break
  $ndx \leftarrow ndx - count(msi[f])$

// At this point, f is the feature upon which we will split; next we need to find the pair in which ndx
// appears.
for each pair (*left_msi*, *right_msi*) **in** $msi[f]$ **do**
  left_count $\leftarrow count($*left_msi*$)$
  right_count $\leftarrow count($*right_msi*$)$
  if $ndx <$ *left_count* $\times$ *right_count* **then**
    break
  $ndx \leftarrow ndx - ($*left_count* $\times$ *right_count*$)$

// We now have the precise pair in which we'll find ndx.
$node \leftarrow$ *make_internal_node*(f)
$node['true'] \leftarrow$ *ndx_to_tree_in_msi*$($*left_msi*$, ndx$ // *left_count*$)$
$node['false'] \leftarrow$ *ndx_to_tree_in_msi*$($*left_msi*$, ndx$ % *left_count*$)$
return $node$

where $\mathbf{x}_{[i,k]}$ is the kth feature of sample i and $\mathbf{x}_{[i,\setminus k]}$ is the set of remaining features. To show how much the accuracy of a fixed tree t relies on variable k, we define the permutation loss:

$$e_{\text{divide}}(t) := \frac{1}{2\lfloor n/2 \rfloor} \sum_{i=1}^{\lfloor n/2 \rfloor} [\ell\{t, (\mathbf{y}_{[i]}, \mathbf{x}_{[i,\setminus k]}, \mathbf{x}_{[i+\lfloor n/2 \rfloor,k]})\} + \ell\{t, (\mathbf{y}_{[i+\lfloor n/2 \rfloor]}, \mathbf{x}_{[i+\lfloor n/2 \rfloor,\setminus k]}, \mathbf{x}_{[i,k]})\}]. \tag{3}$$

We define the model reliance (MR) by division, i.e., $\text{MR}(t) := (e_{\text{divide}}(t) + \lambda H_t)/(e_{\text{orig}}(t) + \lambda H_t)$. This definition is slightly different from [51] as we include the leaf penalty term in both numerator and denominator to take tree complexity into consideration. If $\text{MR}(t)$ is large, it means the error went up substantially when feature k was altered, thus it is important.

MR evaluates how important a feature is for a specific tree t. However, such an estimation may overestimate or underestimate the feature's general importance. For example, a feature with high model reliance with respect to a tree t might have low model reliance for another tree t'. Therefore, we are more interested in how much *any* well-performing model from a decision tree class $\mathcal{T}$ relies on a feature. Given an ϵ-Rashomon set (see Eq (1)), the model class reliance is defined by:

$$[\text{MCR}_-, \text{MCR}_+] := [\min_{t \in R_{\text{set}}(\epsilon, t_{\text{ref}}, \mathcal{T})} \text{MR}(t), \max_{t \in R_{\text{set}}(\epsilon, t_{\text{ref}}, \mathcal{T})} \text{MR}(t)]. \tag{4}$$

According to [51], MCR_- is usually easy to calculate as long as the loss function ℓ is convex and there is no leaf penalty term, but MCR_+ is hard to calculate since even if ℓ is convex, the maximization problem is usually non-convex. In the case of trees (or of any discrete functional class with interaction terms), this problem is discrete and certainly nonconvex, and it is hard to compute either MCR_- or MCR_+ without access to the type of algorithm presented in this work for enumerating the Rashomon set.

E Theorems and Proofs

We first recall some notation. A leaf set $t = (l_1, l_2, ..., l_{H_t})$ contains H_t distinct leaves, where l_i is the classification rule of leaf i. If a leaf is labeled, then $\hat{y}_i$ is the label prediction for all data in leaf i. A partially grown tree t with the leaf set $t = (l_1, l_2, ..., l_{H_t})$ could be rewritten as $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, H_t)$, where t_{fix} is a set of fixed leaves that are not permitted to be further split (those leaves will be split in another instance of this tree separately) and δ_{fix} is a set of predicted labels of leaves in t_{fix}. Similarly, t_{split} is the set of leaves that can be further split and δ_{split} are the predicted labels of leaves t_{split}. We denote $\sigma(t)$ as the set of all t's child trees whose fixed leaves contain t_{fix}.

Theorem 3.1 *(Basic Rashomon Lower Bound) Let θ_ϵ be the threshold of the Rashomon set. Given a tree $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, H_t)$, we denote $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) := \ell(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \lambda H_t$ as the lower bound of the objective for tree t. If $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, then the tree t and all of its children are not in the ϵ-Rashomon set.*

Proof. According to the definition of the lower bound for tree t, we know

$$Obj(t, \mathbf{x}, \mathbf{y}) = \ell(t, \mathbf{x}, \mathbf{y}) + \lambda H_t \geq \ell(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \lambda H_t = b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}).$$

By the same logic, for any child tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, H_{t'}) \in \sigma(t)$, we will also have $Obj(t', \mathbf{x}, \mathbf{y}) \geq b(t', \mathbf{x}, \mathbf{y})$. Since the child trees all have the fixed leaves of the parent, $t_{\text{fix}} \subseteq t'_{\text{fix}}$, and they have more leaves, $H_{t'} \geq H_t$, we have:

$$b(t'_{\text{fix}}, \mathbf{x}, \mathbf{y}) = \ell(t'_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \lambda H_{t'} \geq \ell(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \lambda H_t = b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}).$$

Thus, if $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, then $Obj(t', \mathbf{x}, \mathbf{y}) \geq b(t'_{\text{fix}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, i.e., t' is not in the ϵ-Rashomon set.

Therefore, if $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, we can eliminate the tree t and its all of children from the search space. □

We use notation defined in the main text for equivalent points. As a reminder, a set of points is equivalent if they have the same feature values; thus, those points will always receive identical predictions, and hence, some will be misclassified if they have opposite labels.

Theorem 3.2 *(Rashomon Equivalent Points Bound) Let θ_ϵ be the threshold of the Rashomon set. Let t be a tree with leaves $t_{\text{fix}}, t_{\text{split}}$ and lower bound $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y})$. Let $b_{equiv}(t_{\text{split}}, \mathbf{x}, \mathbf{y}) := \frac{1}{n}\sum_{i=1}^{n}\sum_{u=1}^{U} cap(\mathbf{x}_i, t_{\text{split}}) \wedge \mathbb{1}[\mathbf{x}_i \in e_u] \wedge \mathbb{1}[y_i = q_u]$ be the lower bound on the misclassification loss of leaves that can be further split. Let $B(t, \mathbf{x}, \mathbf{y}) := b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + b_{equiv}(t_{\text{split}}, \mathbf{x}, \mathbf{y})$ be the Rashomon lower bound of t. If $B(t, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, tree t and all its children are not in the ϵ-Rashomon set.*

Proof. According to the definition of $b(t_{\text{fix}}, \mathbf{x}, \mathbf{y})$ and $b_{\text{equiv}}(t_{\text{split}}, \mathbf{x}, \mathbf{y})$, we know

$$Obj(t, \mathbf{x}, \mathbf{y}) = \ell(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + \ell(t_{\text{split}}, \mathbf{x}, \mathbf{y}) + \lambda H_t \geq b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + b_{\text{equiv}}(t_{\text{split}}, \mathbf{x}, \mathbf{y}) = B(t, \mathbf{x}, \mathbf{y}).$$

For $t' \in \sigma(t)$, $Obj(t', \mathbf{x}, \mathbf{y}) \geq b(t_{\text{fix}}, \mathbf{x}, \mathbf{y}) + b_{\text{equiv}}(t_{\text{split}}, \mathbf{x}, \mathbf{y}) = B(t, \mathbf{x}, \mathbf{y})$. When $B(t, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, $Obj(t, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$ and thus $Obj(t', \mathbf{x}, \mathbf{y}) > \theta_\epsilon$. □

Theorem E.1. *(Rashomon Equivalent Points Bound for Subtrees) Let t be a tree such that the root node is split by a feature, where two subtrees t_{left} and t_{right} are generated with $H_{t_{\text{left}}}$ and $H_{t_{\text{right}}}$ leaves for t_{left} and t_{right} respectively. Let $B(t_{\text{left}}, \mathbf{x}, \mathbf{y})$ and $B(t_{\text{right}}, \mathbf{x}, \mathbf{y})$ be the Rashomon equivalent points bound for the left and right subtrees respectively. (Note that $B(t_{\text{left}}, \mathbf{x}, \mathbf{y}) \leq \ell(t_{\text{left}}, \mathbf{x}, \mathbf{y}) + \lambda H_{t_{\text{left}}}$ and $B(t_{\text{right}}, \mathbf{x}, \mathbf{y}) \leq \ell(t_{\text{right}}, \mathbf{x}, \mathbf{y}) + \lambda H_{t_{\text{right}}}$). If $B(t_{\text{left}}, \mathbf{x}, \mathbf{y}) + B(t_{\text{right}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, then the tree t is not a member of the ϵ-Rashomon set.*

Proof.

$$Obj(t, \mathbf{x}, \mathbf{y}) = \ell(t_{\text{left}}, \mathbf{x}, \mathbf{y}) + \ell(t_{\text{right}}, \mathbf{x}, \mathbf{y}) + \lambda(H_{t_{\text{left}}} + H_{t_{\text{right}}}) \geq B(t_{\text{left}}, \mathbf{x}, \mathbf{y}) + B(t_{\text{right}}, \mathbf{x}, \mathbf{y}).$$

If $B(t_{\text{left}}, \mathbf{x}, \mathbf{y}) + B(t_{\text{right}}, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$, then $Obj(t, \mathbf{x}, \mathbf{y}) > \theta_\epsilon$. Therefore, tree t is not in the ϵ-Rashomon set. □

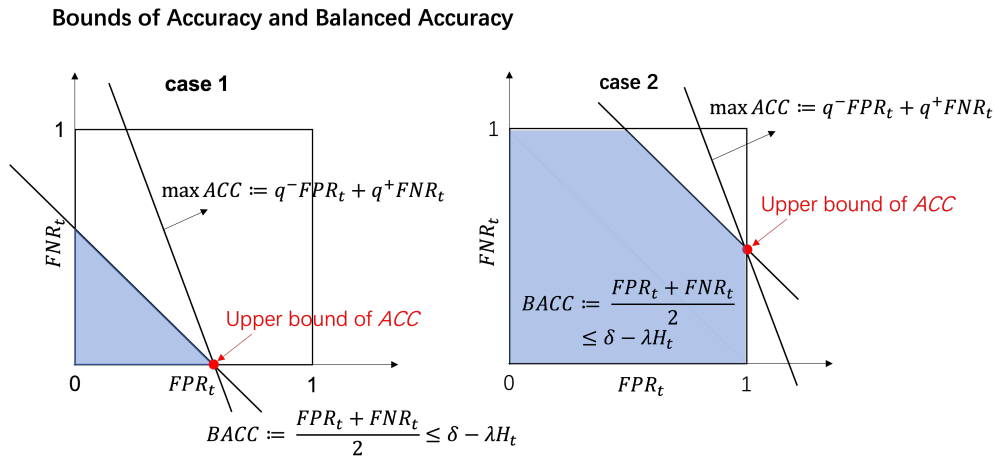


Figure 8: Two cases that can bound the accuracy of tree t. Note that for a tree t, its FNR_t and FPR_t are in range $[0, 1]$. Left: when $2(\delta - \lambda H_t) < 1$, the blue area identifies the feasible region. The point that maximize the accuracy is either $(0, 2(\delta - \lambda H_t))$ or $(2(\delta - \lambda H_t), 0)$. Right: when $2(\delta - \lambda H_t) \geq 1$, the blue area identifies the feasible region. The point that maximize the accuracy is either $(2(\delta - \lambda H_t) - 1, 1)$ or $(1, 2(\delta - \lambda H_t) - 1)$.

We recall some notations used for Theorem 5.1 and 5.2. Let q^+ be the proportion of positive samples and q^- be the proportion of negative samples, i.e., $q^+ + q^- = 1$. We denote $q_{\min} := \min(q^+, q^-)$ and $q_{\max} := \max(q^+, q^-)$. Let FPR and FNR be the false positive and false negative rates. We notate the Accuracy Rashomon set as $A_\theta := \{t \in \mathcal{T} : q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \theta\}$, where θ is the objective threshold of the Accuracy Rashomon set, similar to θ_ϵ in Section 3. We denote δ in these theorems as the objective threshold of Balanced Accuracy or F1-Score Rashomon sets.

Theorem 5.1 *(Accuracy Rashomon set covers Balanced Accuracy Rashomon set) Let* $B_\delta := \{t \in \mathcal{T} : \frac{FPR_t + FNR_t}{2} + \lambda H_t \leq \delta\}$ *be the Balanced Accuracy Rashomon set. If*

$$\theta \geq \min\left(2q_{\max}\delta, q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda 2^d\right),$$

where d *is the depth limit, then* $\forall t \in B_\delta, t \in A_\theta$.

Proof. To prove this theorem, we first get the bound values through geometric intuition, and then prove the inequalities formally. Consider the plane with FPR and FNR being two axes, shown in Figure 8, $\forall t \in B_\delta$, inequalities $\frac{FPR_t + FNR_t}{2} + \lambda H_t \leq \delta$ and $0 \leq FPR_t, FNR_t \leq 1$ bound the feasible region. Our goal is to find the upper bound of the accuracy objective in this feasible region, so that when θ is greater than this upper bound, we have $\forall t \in B_\delta, t \in A_\theta$. As shown in Figure 8 there are two different cases that can bound the accuracy of tree t.

Case 1:

When $2(\delta - \lambda H_t) < 1$ (see Figure 8 left), the accuracy loss is a line in the plane and thus maximized at $(0, 2(\delta - \lambda H_t))$ or $(2(\delta - \lambda H_t), 0)$. The corresponding maximum value is $\max(q^+ \times 2(\delta - \lambda H_t), q^- \times 2(\delta - \lambda H_t)) = q_{\max} 2(\delta - \lambda H_t)$. Inspired by this geometric intuition, formally we want to show,

$$q^- FPR_t + q^+ FNR_t \leq q_{\max} 2(\delta - \lambda H_t). \tag{5}$$

Eq (5) can be shown as follows,

$$\begin{aligned} q^- FPR_t + q^+ FNR_t &\leq q_{\max}(FPR_t + FNR_t) \quad (q_{\max} \geq q^+, q_{\max} \geq q^-) \\ &\leq q_{\max} 2(\delta - \lambda H_t) \quad \text{(see definition of } B_\delta) \end{aligned} \tag{6}$$

Therefore,

$$\begin{aligned} q^- FPR_t + q^+ FNR_t + \lambda H_t &\leq q_{\max} 2(\delta - \lambda H_t) + \lambda H_t \\ &= 2q_{\max}\delta - 2q_{\max}\lambda H_t + \lambda H_t \\ &= 2q_{\max}\delta - (2q_{\max} - 1)\lambda H_t \\ &\leq 2q_{\max}\delta \quad (\text{since } q_{\max} \geq 0.5). \end{aligned} \tag{7}$$

Case 2: Inspired by Figure 8 right. When $2(\delta - \lambda H_t) \geq 1$, the the accuracy loss is maximized at either $(1, 2(\delta - \lambda H_t) - 1)$ or $(2(\delta - \lambda H_t) - 1, 1)$ on the FPR-FNR plane, and the corresponding maximum value is $\max(q^+ \times (2(\delta - \lambda H_t) - 1) + q^-, q^- \times (2(\delta - \lambda H_t) - 1) + q^+) = q_{\max} + (2(\delta - \lambda H_t) - 1)q_{\min}$. First, we want to show,

$$q^- FPR_t + q^+ FNR_t \leq q_{\max} + (FPR_t + FNR_t - 1)q_{\min}. \tag{8}$$

$$\begin{aligned} \text{Eq (8) right – Eq (8) left} &= q_{\max} + (FPR_t + FNR_t - 1)q_{\min} - q^- FPR_t - q^+ FNR_t \\ &= q_{\max} - q_{\min} + q_{\min}(FPR_t + FNR_t) - q^- FPR_t - q^+ FNR_t \\ &= q_{\max} - q_{\min} + (q_{\min} - q^-)FPR_t + (q_{\min} - q^+)FNR_t. \end{aligned} \tag{9}$$

If $q^- \geq q^+$, then Eq (9) = $q^- - q^+ + (q^+ - q^-)FPR_t = (q^- - q^+) - (q^- - q^+)FPR_t \geq 0$ since $q^- - q^+ \geq 0$ and $1 - FPR_t \geq 0$. Similarly, if $q^+ \geq q^-$, then Eq (9) $= (q^+ - q^-) - (q^+ - q^-)FNR_t \geq 0$. Therefore, Eq (9) is always greater than or equal to 0. Hence, Eq (8) holds. Therefore,

$$\begin{aligned} q^- FPR_t + q^+ FNR_t &\leq q_{\max} + (FPR_t + FNR_t - 1)q_{\min} \\ &\leq q_{\max} + (2(\delta - \lambda H_t) - 1)q_{\min} \quad (\text{see definition of } B_\delta), \end{aligned} \tag{10}$$

which is the same as the maximum value we got intuitively from Figure 8 right. Adding λH_t to both sides, we have

$$\begin{aligned} q^- FPR_t + q^+ FNR_t + \lambda H_t &\leq q_{\max} + (2(\delta - \lambda H_t) - 1)q_{\min} + \lambda H_t \\ &= q_{\max} + (2\delta - 1)q_{\min} - 2q_{\min}\lambda H_t + \lambda H_t \\ &= q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda H_t \\ &\leq q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda 2^d \quad (\text{if depth is at most } d). \end{aligned} \tag{11}$$

Combining these two cases,

$$q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \min\left(2q_{\max}\delta, q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda 2^d\right). \tag{12}$$

Therefore, if $\theta \geq \min\left(2q_{\max}\delta, q_{\max} + (2\delta - 1)q_{\min} + (1 - 2q_{\min})\lambda 2^d\right)$, then $\forall t \in B_\delta$, $q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \theta$. In other words, $\forall t \in B_\delta$, $t \in A_\theta$. □

Theorem 5.2 *(Accuracy Rashomon set covers F1-score Rashomon set) Let*

$$F_\delta := \left\{ t \in \mathcal{T} : \frac{q^- FPR_t + q^+ FNR_t}{2q^+ + q^- FPR_t - q^+ FNR_t} + \lambda H_t \leq \delta \right\}$$

be the F1-score Rashomon set. Suppose $q^+ \in (0, 1)$*,* $q^- \in (0, 1)$*, and* $\delta - \lambda H_t \in (0, 1)$*. If* $\theta \geq \min\left(\max\left(\frac{2q^+\delta}{1-\delta}, \frac{2q^+(\delta - \lambda 2^d)}{1-(\delta - \lambda 2^d)} + \lambda 2^d\right), \mathbb{1}[\delta < \sqrt{2} - 1]\frac{2\delta}{1+\delta} + \mathbb{1}[\delta \geq \sqrt{2} - 1](\delta + 3 - 2\sqrt{2})\right)$*, then* $\forall t \in F_\delta$*,* $t \in A_\theta$*.*

Proof. Similar to Theorem 5.1, we first get the bound values through geometric intuitions, and then prove the inequalities formally. First, we want to represent the F1 loss with FPR and FNR so that we can put it in the FPR-FNR plane. The F1-score is harmonic mean of precision and recall and is

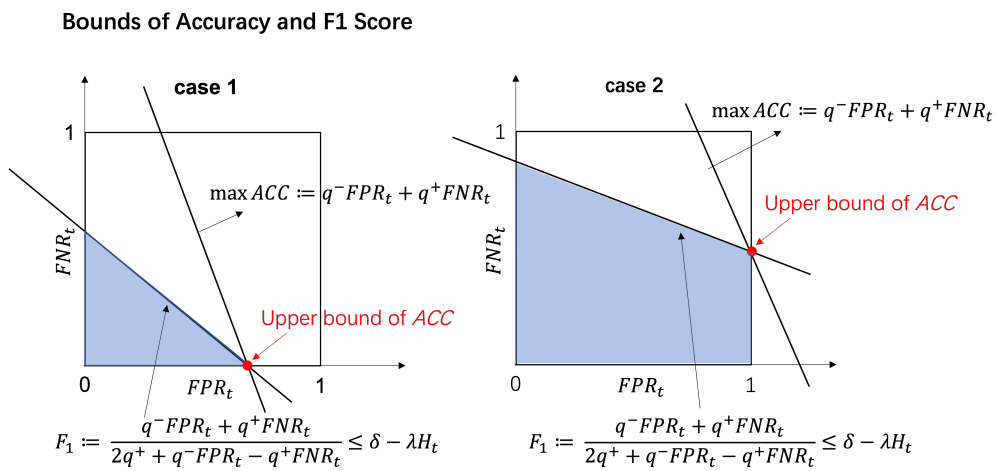


Figure 9: Two cases that can bound the accuracy of tree t. Note that for a tree t, its FNR_t and FPR_t are in range $[0, 1]$. Left: when $FPR_t < 1$, the blue area identifies the feasible region. The point that maximize the accuracy is when $FNR_t = 0$. Right: when $FPR_t = 1$, the blue area identifies the feasible region. The point that maximize the accuracy is when the line intersects the right boundary.

usually written in terms of FP and FN as $1 - \frac{FP+FN}{2N^+ + FP - FN}$, where N^+ is the number of positive samples. With some operations we can rewrite this formula in terms of FPR and FPN,

$$\begin{aligned}
\frac{FP + FN}{2N^+ + FP - FN} &= \frac{(FP + FN)/n}{(2q^+ \times n + FP - FN)/n} \\
&= \frac{\frac{FP}{n} \times \frac{q^-}{q^-} + \frac{FN}{n} \times \frac{q^+}{q^+}}{2q^+ + \frac{FP}{n} \times \frac{q^-}{q^-} - \frac{FN}{n} \times \frac{q^+}{q^+}} \\
&= \frac{q^-FPR + q^+FNR}{2q^+ + q^-FPR - q^+FNR}
\end{aligned}$$

Here, for simplicity, we use $f1_t := \frac{q^-FPR_t + q^+FNR_t}{2q^+ + q^-FPR_t - q^+FNR_t}$ to denote the F1 loss, which is 1 minus the F1-score of tree t. Based on the definition of F_δ, $\forall t \in F_\delta$, we have

$$\begin{aligned}
& \frac{q^-FPR_t + q^+FNR_t}{2q^+ + q^-FPR_t - q^+FNR_t} + \lambda H_t \leq \delta \\
\Rightarrow \quad & q^-FPR_t + q^+FNR_t \leq (\delta - \lambda H_t) \times (2q^+ + q^-FPR_t - q^+FNR_t) \\
\Rightarrow \quad & q^+FNR_t(1 + \delta - \lambda H_t) \leq 2q^+(\delta - \lambda H_t) + q^-(\delta - \lambda H_t - 1)FPR_t \\
\Rightarrow \quad & FNR_t \leq \frac{2(\delta - \lambda H_t)}{1 + \delta - \lambda H_t} + \frac{q^-(\delta - \lambda H_t - 1)}{q^+(1 + \delta - \lambda H_t)} FPR_t \quad (1 + \delta - \lambda H_t > 1)
\end{aligned} \tag{13}$$

This is a line. The slope of the F1-score boundary line is $\frac{q^-(\delta - \lambda H_t - 1)}{q^+(1 + \delta - \lambda H_t)}$ and the slope of the accuracy boundary is $\frac{-q^-}{q^+}$. Since $\delta - \lambda H_t \in (0, 1)$, $\frac{\delta - \lambda H_t - 1}{1 + \delta - \lambda H_t} < 0$ and $|\frac{\delta - \lambda H_t - 1}{1 + \delta - \lambda H_t}| < 1$. Therefore, both slopes are negative and the slope of F1-score boundary is always larger than the slope of the accuracy boundary (see Figure 9). In addition, we can also show the intercept, which is also the upper bound of FNR in Eq (13), $\frac{2(\delta - \lambda H_t)}{1 + \delta + \lambda H_t} < 1$, since $\delta - \lambda H_t \in (0, 1)$ and $\frac{2(\delta - \lambda H_t)}{1 + \delta + \lambda H_t}$ is monotonically increasing with $\delta - \lambda H_t$. Therefore,

$$FNR < 1. \tag{14}$$

Since $0 \leq FPR, FNR \leq 1$, $\forall t \in F_\delta$, there are also two different cases that can bound the accuracy of tree t (see Figure 9).

Case 1: Based on Figure 9 left, when $FPR_t < 1$, the point that maximizes the accuracy is when $FNR_t = 0$. Using Eq (13), we know the point is $(\frac{-2q^+(\delta-\lambda H_t)}{q^-(\delta-\lambda H_t-1)}, 0)$, and the corresponding loss is $\frac{2q^+(\delta-\lambda H_t)}{1+\lambda H_t-\delta}$. Formally, we want to show

$$q^- FPR_t + q^+ FNR_t \leq \frac{2q^+(\delta - \lambda H_t)}{1 - (\delta - \lambda H_t)} \tag{15}$$

Eq (15) is shown as follows,

$$\begin{aligned}
q^- FPR_t + q^+ FNR_t &\leq \frac{q^- FPR_t + q^+ FNR_t}{1 - FNR_t} \quad \text{(Eq (14), and denominator} \leq 1) && (16)\\
&= \frac{2q^+(q^- FPR_t + q^+ FNR_t)}{2q^+(1 - FNR_t)} && (17)\\
&= \frac{2q^+(q^- FPR_t + q^+ FNR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)}{(2q^+ - 2q^+ FNR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)} && (18)\\
&= \frac{2q^+(q^- FPR_t + q^+ FNR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)}{\frac{2q^+ + q^- FPR_t - q^+ FNR_t - q^- FPR_t - q^+ FNR_t}{2q^+ + q^- FPR_t - q^+ FNR_t}} && (19)\\
&= \frac{2q^+(q^- FPR_t + q^+ FNR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)}{1 - f1_t} && (20)\\
&= \frac{2q^+ f1_t}{1 - f1_t}. && (21)
\end{aligned}$$

Note that the denominator cannot be 0, because $FNR < 1$ as shown in Eq (14). Eq (21) is monotonically increasing in $f1_t$, since its first derivative is $\frac{2q^+(1-f1_t)-2q^+ f1_t(-1)}{(1-f1_t)^2} = \frac{2q^+}{(1-f1_t)^2} > 0$. Since $t \in F_\delta$, the maximum F1-score is $\delta - \lambda H_t$. Therefore,

$$q^- FPR_t + q^+ FNR_t \leq \frac{2q^+ f1_t}{1 - f1_t} \leq \frac{2q^+(\delta - \lambda H_t)}{1 - (\delta - \lambda H_t)}. \tag{22}$$

Adding the leaf penalty on both sides, we get

$$q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \frac{2q^+(\delta - \lambda H_t)}{1 - (\delta - \lambda H_t)} + \lambda H_t. \tag{23}$$

Since H_t is a variable, for simplicity, we let $a := \lambda H_t$. Then we can define $g(a) := \frac{2q^+(\delta-a)}{1-(\delta-a)} + a$.

Solving $g'(a) = \frac{-2q^+}{(1-\delta+a)^2} + 1 = 0$, we get $a = \delta - 1 \pm \sqrt{2q^+}$. Since a is the leaf penalty, $a \geq 0$, we have only one solution for a. That is, $a = \delta - 1 + \sqrt{2q^+}$. Now we consider $g''(a) = \frac{4q^+(1-\delta+a)}{(1-\delta+a)^4}$, and, plugging in our solution for a, we see $g''(a) \geq 0$. Therefore, $g(a)$ achieves the minimum value when $a = \delta - 1 + \sqrt{2q^+}$.

If $0 \leq \delta - 1 + \sqrt{2q^+} \leq \lambda 2^d$, then for $a \in [0, \delta - 1 + \sqrt{2q^+}]$, $g'(a) \leq 0$, indicating $g(a)$ monotonically decreases in this range, while for $a \in [\delta - 1 + \sqrt{2q^+}, \lambda 2^d]$, $g'(a) \geq 0$, indicating $g(a)$ monotonically increases. Therefore, for $a \in [0, \lambda 2^d]$, $g(a)$ first increases and then decreases, and thus maximized at the two ends, i.e.,

$$\max g(a) = \max(g(a = 0), g(a = \lambda 2^d)) = \max\left(\frac{2q^+\delta}{1 - \delta}, \frac{2q^+(\delta - \lambda 2^d)}{1 - (\delta - \lambda 2^d)} + \lambda 2^d\right).$$

In summary,

$$q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \max\left(\frac{2q^+\delta}{1 - \delta}, \frac{2q^+(\delta - \lambda 2^d)}{1 - (\delta - \lambda 2^d)} + \lambda 2^d\right). \tag{24}$$

Case 2: Inspired by Figure 9 right, we can find that the maximizer when $FPR_t = 1$ at $(1, \frac{(q^+ + 1)(\delta - \lambda H_t) - q^-}{q^+(1 + \delta - \lambda H_t)})$ using Eq (13). And the corresponding loss is $\frac{2(\delta - \lambda H_t)}{1 + \delta - \lambda H_t}$. Formally, we want to show

$$q^- FPR_t + q^+ FNR_t \leq \frac{2(\delta - \lambda H_t)}{1 + \delta - \lambda H_t}. \tag{25}$$

To show Eq (25),

$$\begin{aligned} q^- FPR_t + q^+ FNR_t &\leq \frac{q^- FPR_t + q^+ FNR_t}{q^+ + q^- FPR_t} \; (FPR_t \leq 1, \text{denominator} \in (0,1]) && (26) \\ &= \frac{(q^- FPR_t + q^+ FNR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)}{(q^+ + q^- FPR_t)/(2q^+ + q^- FPR_t - q^+ FNR_t)} && (27) \\ &= \frac{f1_t}{(1 + f1_t)/2} && (28) \\ &= \frac{2f1_t}{1 + f1_t}. && (29) \end{aligned}$$

Eq (29) is monotonically increasing with $f1$, since its derivative $= \frac{2(1+f1_t)-2f1_t}{(1+f1_t)^2} = \frac{2}{(1+f1_t)^2} \geq 0$. Moreover, since $t \in F_\delta$, the maximum f1-score loss is $\delta - \lambda H_t$. Therefore,

$$q^- FPR_t + q^+ FNR_t \leq \frac{2f1_t}{1 + f1_t} \leq \frac{2(\delta - \lambda H_t)}{1 + \delta - \lambda H_t}. \tag{30}$$

Adding leaf penelty to two sides, we can get

$$q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \frac{2(\delta - \lambda H_t)}{1 + \delta - \lambda H_t} + \lambda H_t. \tag{31}$$

Let's change the variable. Let $a = \lambda H_t$. Then the right-hand side of Eq (31) is a function of a. Let $g(a) := \frac{2(\delta - a)}{1+\delta-a} + a$.

$$\frac{dg}{da} = \frac{-2(1 + \delta - a) - 2(\delta - a)(-1)}{(1 + \delta - a)^2} + 1 = \frac{-2}{(1 + \delta - a)^2} + 1 \tag{32}$$

Solving $\frac{dg}{da} = 0$, we get

$$a = 1 + \delta \pm \sqrt{2}. \tag{33}$$

Since a is the leaf penalty term, $\delta - a > 0$ according to our assumption. If $a = 1 + \delta + \sqrt{2}$, $\delta - a = -2.414 < 0$ which contradicts the assumption. Therefore, only $a = 1 + \delta - \sqrt{2}$ could be the valid stationary point. We then calculate $g''(a) = \frac{-4(1+\delta-a)}{(1+\delta-a)^4} \leq 0$. Therefore, $a = 1 + \delta - \sqrt{2}$ is the maximizer of $g(a)$.

(i) when $\delta + 1 - \sqrt{2} < 0$, we still have that a cannot be smaller than 0. Therefore, the maximum value of $g(a)$ happens when $a = 0$, which is equal to $\frac{2\delta}{1+\delta}$.

(ii) when $\delta + 1 - \sqrt{2} \geq 0$, the maximum value of $g(a)$ is when $a = \delta + 1 - \sqrt{2}$, which is equal to $\delta + 3 - 2\sqrt{2}$.

In summary,

$$q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \mathbb{1}[\delta < \sqrt{2} - 1] \frac{2\delta}{1 + \delta} + \mathbb{1}[\delta \geq \sqrt{2} - 1](\delta + 3 - 2\sqrt{2}). \tag{34}$$

Combining the two cases together,

$$\begin{aligned} q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \min \Bigg(& \max \left(\frac{2q^+ \delta}{1 - \delta}, \frac{2q^+ (\delta - \lambda 2^d)}{1 - (\delta - \lambda 2^d)} + \lambda 2^d \right), \\ & \mathbb{1}[\delta < \sqrt{2} - 1] \frac{2\delta}{1 + \delta} + \mathbb{1}[\delta \geq \sqrt{2} - 1](\delta + 3 - 2\sqrt{2}) \Bigg). \end{aligned} \tag{35}$$

Therefore, If $\theta \geq \min\Big(\max\big(\frac{2q^+\delta}{1-\delta}, \frac{2q^+(\delta-\lambda 2^d)}{1-(\delta-\lambda 2^d)} + \lambda 2^d\big), \mathbb{1}[\delta < \sqrt{2}-1]\frac{2\delta}{1+\delta} + \mathbb{1}[\delta \geq \sqrt{2}-1](\delta + 3-2\sqrt{2})\Big)$, then $\forall t \in F_\delta$, $q^- FPR_t + q^+ FNR_t + \lambda H_t \leq \theta$. In other words, $\forall t \in F_\delta$, $t \in A_\theta$. $\square$

Recall notation for Theorem 5.3 and 5.4. Let $\tilde{t}^*$ be the optimal tree trained on $\{\mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}\}$ where K is a set of indices of instances that we wish to analyze. We denote $|K|$ as the cardinality of the set K. Overloading notation to include the dataset, let $R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y}) = R_{set}(\epsilon, t^*, \mathcal{T})$ (see Eq (1)) be the Rashomon set of the original dataset, where $t_{\text{ref}} = t^*$ is the optimal tree trained on the original dataset, and we define the ϵ'-Rashomon set on the reduced dataset as

$$R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) := \left\{t \in \mathcal{T} : Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) \leq (1+\epsilon') \times Obj(\tilde{t}^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]})\right\}.$$

Theorem 5.3 *(Optimal tree after removing a group of instances is still in full-dataset Rashomon set) If* $\epsilon \geq \frac{2|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$, $\tilde{t}^* \in R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.

Proof. The objective of t^* on the original dataset and the reduced dataset are

$$Obj(t^*, \mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] + \lambda H_{t^*} \text{ and } Obj(t^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) = \frac{1}{n-|K|}\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] + \lambda H_{t^*}.$$

Similarly, the objective of $\tilde{t}^*$ on the original dataset and the reduced dataset are

$$Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] + \lambda H_{\tilde{t}^*} \text{ and } Obj(\tilde{t}^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) = \frac{1}{n-|K|}\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] + \lambda H_{\tilde{t}^*}.$$

Since $\tilde{t}^*$ is the optimal tree trained on $\{\mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}\}$,

$$Obj(\tilde{t}^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) \leq Obj(t^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}). \tag{36}$$

Step 1: bound the difference between $Obj(t^*, \mathbf{x}, \mathbf{y})$ and $Obj(t^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]})$

Since $\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] = \sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] + \sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}]$, we can get

$$\begin{aligned}
Obj(t^*, \mathbf{x}, \mathbf{y}) - Obj(t^*, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] - \frac{1}{n-|K|}\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \\
&= \frac{1}{n}\Big(\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] + \sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}]\Big) - \frac{1}{n-|K|}\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \\
&= \frac{(n-|K|)\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] + (n-|K|)\sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}]}{n(n-|K|)} \\
&\quad - \frac{n\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}]}{n(n-|K|)} \\
&= \frac{(n-|K|)\sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] - |K| \times \sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}]}{n(n-|K|)}.
\end{aligned} \tag{37}$$

Since $\sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \in \{0, 1, \cdots, |K|\}$,

$$(n-|K|)\sum_{i \in K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \in \{0, n-|K|, \cdots, (n-|K|)|K|\}.$$

Similarly, $\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \in \{0, 1, \cdots, n-|K|\}$, therefore,

$$|K|\sum_{i \notin K} \mathbb{1}[y_i \neq \hat{y}_i^{t^*}] \in \{0, |K|, \cdots, |K|(n-|K|)\}.$$

Combining the extreme values of these two terms together, we know Eq (37) is

$$Obj(t^*, \mathbf{x}, \mathbf{y}) - Obj(t^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) \geq \frac{-|K|}{n}. \tag{38}$$

Step 2: bound the difference between $Obj(\tilde{t}^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]})$ and $Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y})$.

$$\begin{aligned}
Obj(\tilde{t}^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) - Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y}) &= \frac{1}{n-|K|}\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] \\
&= \frac{1}{n-|K|}\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] - \frac{1}{n}\left[\sum_{i\in K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] + \sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]\right] \\
&= \frac{n\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]}{(n-|K|)n} \\
&\quad - \frac{(n-|K|)(\sum_{i\in K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] + \sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}])}{n(n-|K|)} \\
&= \frac{|K|\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}] - (n-|K|)\sum_{i\in K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]}{n(n-|K|)}.
\end{aligned} \tag{39}$$

Since $\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]) \in \{0, 1, \cdots, n-|K|\}$,

$$|K|\sum_{i\notin K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]) \in \{0, |K|, \cdots, |K|(n-|K|)\}.$$

Similarly, $\sum_{i\in K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]) \in \{0, 1, \cdots, |K|\}$,

$$(n-|K|)\sum_{i\in K} \mathbb{1}[y_i \neq \hat{y}_i^{\tilde{t}^*}]) \in \{0, n-|K|, \cdots, |K|(n-|K|)\}.$$

Combining the extreme values of these two terms, we know Eq (39) obeys

$$Obj(\tilde{t}^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) - Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y}) \geq \frac{-|K|}{n}. \tag{40}$$

Given Eq (36), (38), (40), we can get

$$\begin{aligned}
Obj(t^*, \mathbf{x}, \mathbf{y}) &\geq Obj(t^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) + \frac{-|K|}{n} \quad \text{(see Eq (38))} \\
&\geq Obj(\tilde{t}^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) + \frac{-|K|}{n} \quad \text{(see Eq (36))} \\
&\geq Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y}) + \frac{-2|K|}{n} \quad \text{(see Eq (40))}.
\end{aligned} \tag{41}$$

In other words, $Obj(\tilde{t}^*, \mathbf{x}, \mathbf{y}) \leq Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{2|K|}{n}$. To guarantee that $Rset(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$ covers $\tilde{t}^*$, i.e., $Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{2|K|}{n} \leq (1+\epsilon) \times Obj(t^*, \mathbf{x}, \mathbf{y})$, we impose that $\epsilon \geq \frac{2|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$. □

Theorem 5.4 *(Rashomon set after removing a group of instances is within full-dataset Rashomon set) If $\epsilon \geq \epsilon' + \frac{(2+\epsilon')|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$, then $\forall t \in R_{\text{set}}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]})$, we have $t \in R_{\text{set}}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.*

Proof. Given a decision tree $t \in R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]})$, we know

$$\begin{aligned}
Obj(t, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) &\leq (1+\epsilon') \times Obj(\tilde{t}^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) \\
&\leq (1+\epsilon') \times Obj(t^*, \mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}) \quad (\tilde{t}^* \text{is the optimal tree on } \{\mathbf{x}_{[\backslash K,\cdot]}, \mathbf{y}_{[\backslash K]}\}) \\
&\leq (1+\epsilon') \times \left[Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{|K|}{n}\right] \quad \text{(see Eq (38))}.
\end{aligned} \tag{42}$$

Then we bound the difference between $Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]})$ and $Obj(t, \mathbf{x}, \mathbf{y})$.

$$\begin{aligned}
Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) - Obj(t, \mathbf{x}, \mathbf{y}) &= \frac{1}{n-|K|}\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] - \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}[y_i \neq \hat{y}_i^t] \\
&= \frac{1}{n-|K|}\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] - \frac{1}{n}\left[\sum_{i\in K}\mathbb{1}[y_i \neq \hat{y}_i^t] + \sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t]\right] \\
&= \frac{n\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] - (n-|K|)\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t]}{(n-|K|)n} \\
&\quad - \frac{(n-|K|)(\sum_{i\in K}\mathbb{1}[y_i \neq \hat{y}_i^t])}{n(n-|K|)} \\
&= \frac{|K|\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] - (n-|K|)\sum_{i\in K}\mathbb{1}[y_i \neq \hat{y}_i^t]}{n(n-|K|)}.
\end{aligned} \tag{43}$$

Since $\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] \in \{0, 1, \cdots, n-|K|\}$,

$$|K|\sum_{i\notin K}\mathbb{1}[y_i \neq \hat{y}_i^t] \in \{0, |K|, \cdots, |K|(n-|K|)\}.$$

Similarly, $\sum_{i\in K}\mathbb{1}[y_i \neq \hat{y}_i^t]) \in \{0, 1, \cdots, |K|\}$,

$$(n-|K|)\sum_{i\in K}\mathbb{1}[y_i \neq \hat{y}_i^t]) \in \{0, n-|K|, \cdots, |K|(n-|K|)\}.$$

Combining the extreme values of these two terms we know Eq (43) obeys

$$Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) - Obj(t, \mathbf{x}, \mathbf{y}) \geq \frac{-|K|}{n}. \tag{44}$$

Combining Eq (42) and (44), we get

$$\begin{aligned}
Obj(t, \mathbf{x}, \mathbf{y}) &\leq Obj(t, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}) + \frac{|K|}{n} \quad \text{(see Eq 44)} \\
&\leq (1+\epsilon') \times \left[Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{|K|}{n}\right] + \frac{|K|}{n} \quad \text{(see Eq (42))} \\
&= (1+\epsilon') \times Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{(2+\epsilon')|K|}{n}.
\end{aligned} \tag{45}$$

Thus, when $\epsilon \geq \epsilon' + \frac{(2+\epsilon')|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$, Eq (45) extends to

$$\begin{aligned}
Obj(t, \mathbf{x}, \mathbf{y}) &\leq (1+\epsilon') \times Obj(t^*, \mathbf{x}, \mathbf{y}) + \frac{(2+\epsilon')|K|}{n} \\
&\leq (1+\epsilon') \times Obj(t^*, \mathbf{x}, \mathbf{y}) + (\epsilon - \epsilon') \times Obj(t^*, \mathbf{x}, \mathbf{y}) \\
&= (1+\epsilon) \times Obj(t^*, \mathbf{x}, \mathbf{y}).
\end{aligned} \tag{46}$$

In other words, when $\epsilon \geq \epsilon' + \frac{(2+\epsilon')|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$, $\forall t \in R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}), t \in R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.

□

F Experimental Datasets

We present results for 14 datasets: four from the UCI Machine Learning Repository [52] (Car Evaluation, Congressional Voting Records, Monk2, Iris, and Breast Cancer), a penguin dataset [53], a recidivism dataset (COMPAS) [54], the Fair Isaac (FICO) credit risk dataset [55] used for the Explainable ML Challenge, and four coupon datasets (Bar, Coffee House, Takeaway Food, Cheap

Restaurant, and Expensive Restaurant), which were collected on Amazon Mechanical Turk via a survey [56]. We predict which individuals are arrested within two years of release on the COMPAS dataset, whether an individual will default on a loan for the FICO dataset, and whether a customer will accept a coupon for takeaway food or a cheap restaurant depending on their coupon usage history, current conditions while driving, and coupon expiration time on two coupon datasets. All the datasets are publicly available.

F.1 Preprocessing

Table 1 summarizes all the datasets after preprocessing.

Dataset	Samples	Binary Features
Car Evaluation	1728	15
Congressional Voting Records	435	32
Monk2	169	11
Penguin	333	13
Iris	151	15
Breast Cancer	699	10
COMPAS	6907	12
FICO	10459	17
Bar-7	1913	14
Bar	1913	15
Coffee House	3816	15
Takeaway Food	2280	15
Cheap Restaurant	2653	15
Expensive Restaurant	1417	15

Table 1: Preprocessed datasets

Car Evaluation, Congressional Voting Records, Monk2, Penguin: We preprocess these datasets, which contain only categorical features, using one-hot encoding.

Iris: We select thresholds produced by splitting each numerical feature into three equal parts. We used an implementation of qcut in Pandas [57] to split features. This yields 15 features.

Breast Cancer: We select features and thresholds that are used by a gradient boosted tree with 40 decision stumps, which are "Clump_Thickness = 10", "Uniformity_Cell_Size=1", "Uniformity_Cell_Size=10", "Uniformity_Cell_Shape=1", "Marginal_Adhesion=1", "Single_Epithelial_Cell_Size=2", "Bare_Nuclei=1", "Bare_Nuclei=10", "Normal_Nucleoli=1", "Normal_Nucleoli=10".

COMPAS: We use the same discretized binary features of COMPAS produced in [58], which are the following: "sex = Female", "age $<$ 21", "age $<$ 23", age $<$ 26, "age $<$ 46", "juvenile felonies = 0", "juvenile misdemeanors = 0", "juvenile crimes = 0", "priors = 0", "priors = 1", "priors = 2 to 3", "priors $>$ 3".

FICO: We use the same discretized binary features of FICO produced in [58], which are the following: "External Risk Estimate $<$ 0.49" , "External Risk Estimate $<$ 0.65", "External Risk Estimate $<$ 0.80", "Number of Satisfactory Trades $<$ 0.5", "Trade Open Time $<$ 0.6", "Trade Open Time $<$ 0.85", "Trade Frequency $<$ 0.45", "Trade Frequency $<$ 0.6", "Delinquency $<$ 0.55", "Delinquency $<$ 0.75", "Installment $<$ 0.5", "Installment $<$ 0.7", "Inquiry $<$ 0.75", "Revolving Balance $<$ 0.4", "Revolving Balance $<$ 0.6", "Utilization $<$ 0.6", "Trade W. Balance $<$ 0.33".

Bar, Coffee House, Takeaway Food, Cheap Restaurant, Expensive Restaurant: We selected features "destination", "passanger", "weather", "temperature", "time", "expiration", "gender", "age", "maritalStatus", "childrenNumber", "education", "occupation", "income", "Bar", "Coffee-House", "CarryAway", "RestaurantLessThan20", "Restaurant20To50", "toCouponGEQ15min", "toCouponGEQ25min", "directionSame" and the label Y, and removed observations with missing values. We used one-hot encoding to transform these categorical features into binary features. We then selected 15 binary features with the highest variable importance value trained using gradient boosted trees with 100 max-depth 3 weak classifiers.

- **Bar**: The selected binary features are "Bar = 1 to 3", "Bar = 4 to 8", "Bar = less1", "maritalStatus = Single", "childrenNumber = 0", "Bar = gt8", "passanger = Friend(s)", "time = 6PM", "passanger = Kid(s)", "CarryAway = 4 to 8", "gender = Female", "education = Graduate degree (Masters Doctorate etc.)", "Restaurant20To50 = 4 to 8", "expiration = 1d", "temperature = 55".
- **Coffee House**: The selected binary features are "CoffeeHouse = 1 to 3", "CoffeeHouse = 4 to 8", "CoffeeHouse = gt8", "CoffeeHouse = less1", "expiration = 1d", "destination = No Urgent Place", "time = 10AM", "direction = same", "destination = Home", "toCoupon = GEQ15min", "Restaurant20To50 = gt8", "education = Bachelors degree", "time = 10PM", "income = $75000 - $87499", "passanger = Friend(s)".
- **Takeaway Food**: The selected binary features are "expiration = 1d", "time = 6PM", "CoffeeHouse = gt8", "education= Graduate degree", "weather = Rainy", "maritalStatus = Single", "time = 2PM", "occupation = Student", "income =$62500-$74999", "occupation = Legal", "occupation = Installation Maintenance & Repair", "direction = same", "destination = No Urgent Place", "income = $100000 or more", "Bar = less1".
- **Cheap Restaurant**: The selected binary features are "toCoupon = GEQ25min", "expiration = 1d", "time = 6PM", "destination = No Urgent Place", "time = 10PM", "CarryAway = less1", "passanger = Kid(s)", "weather = Snowy", "passanger = Alone", "income = $87500 - $99999", "occupation = Retired", "CoffeeHouse = gt8", "age = 36", "weather = Rainy", "direction = same".
- **Expensive Restaurant**: The selected binary features are "expiration = 1d", "CoffeeHouse = 1 to 3", "Restaurant20To50 = 4 to 8", "Restaurant20To50 = 1 to 3", "occupation = Office & Administrative Support", "age = 31", "Restaurant20To50 = gt8", "income =$12500 - $24999", "toCoupon = GEQ15min", "occupation = Computer & Mathematical", "time = 10PM", "CoffeeHouse = 4 to 8", "income=$50000 - $62499", "passanger = Alone", "destination = No Urgent Place".

Bar-7: We use the same discretized binary features of Bar-7 produced in [59], which are the following: "passanger = Kids", "age = 21", "age = 26","age = 31","age = 36", "age = 41","age = 46", "age = 26", "age = 50plus", "Bar = 1 to 3", "Bar = 4 to 8", "Bar = gt8", "Bar = less1", "Restaurant20to50 $\geq$4, "direction = same".

G More Experimental Results

G.1 Scalability and Efficiency of Calculating Rashomon set with TreeFARMS versus Baselines

Collection and Setup: We ran this experiment on 10 datasets: **Monk2, COMPAS, Car Evaluation, FICO, Congressional Voting Records, Bar, Bar7, Coffee House, Cheap Restaurant, Expensive Restaurant**. To run TreeFARMS, we set λ to 0.01 for Monk2, COMPAS, FICO, Congressional Voting Records, Bar, Bar7, Coffee House and Expensive Restaurant and to 0.005 for Car Evaluation and Cheap Restaurant. These choices yielded sufficiently larger Rashomon sets. We set ϵ to 0.15 for Congressional Voting Records and 0.10 for all other datasets. This means we store all models within 10% of the optimal solution.

We used the R package BART [60] and set the number of trees in each iteration to 1. We sampled models from the posterior with 10-iteration intervals between draws. To get sparser models, we used a sparse Dirichlet prior. To get a set of trees from Random Forest and CART, we used RandomForestClassifier from scikit-learn [61] and set min_samples_split to $\max(\lceil 2n\lambda \rceil, 2)$, min_samples_leaf to $\lceil n\lambda \rceil$, and max_leaf_nodes=$\lfloor 1/(2\lambda) \rfloor$. We set max_features to "auto" for Random Forest and "None" for CART. For GOSDT, we sampled 75% of the dataset size, with replacement to improve diversity, and fit an optimal tree on the sampled data. To improve performance for BART, Random Forest, and CART with sampling, we collapsed trivial splits (nodes that have two leaves with the same predicted label) into a single node. We record the time used to construct the Rashomon set and ran the baselines for a similar time to sample trees. Thus, all methods were given the same amount of time to produce good trees. For each dataset and baseline method, we used 5 different random seeds to compute the average and standard deviation of number of trees found in Rashomon set and run time. We did not

report errors on TreeFARMS as it is deterministic. We used results obtained from the first random seed to generate Figure 10.

We ran this experiment on a 2.7Ghz (768GB RAM 48 cores) Intel Xeon Gold 6226 processor. We set a 200-GB memory limit.

Results: Figure 10 compares the Rashomon set with the four baselines on the Car Evaluation, Coffee House, Cheap and Expensive Restaurant datasets. Similar to the patterns shown in Figure 1, the four subfigures on the top show that TreeFARMS (in purple) found ***many more distinct trees in the Rashomon set*** than any of the four baselines on all of the datasets. The baseline methods tend to find many duplicated trees, and most trees found by the baselines have objective values higher than the threshold of the Rashomon set, and thus, are not in the Rashomon set. The four subfigures on the bottom show the distribution of objective values of trees from each method. We note that in the method GOSDT with sampling the optimal tree for a given subsample of the dataset might not be identical to the optimal tree to the original dataset. Thus, GOSDT, which finds provably optimal trees for a given dataset, does not obtain many distinct trees that are in the Rashomon set: instead, many subsamples produce either the same optimal solutions or trees that are outside of the full dataset's Rashomon set, which is why its curve is not visible. TreeFARMS is the only method *guaranteed* to find all trees in the Rashomon set.

Table 2 shows that TreeFARMS finds ***more trees in the Rashomon set per second*** than all other baselines on all the datasets.

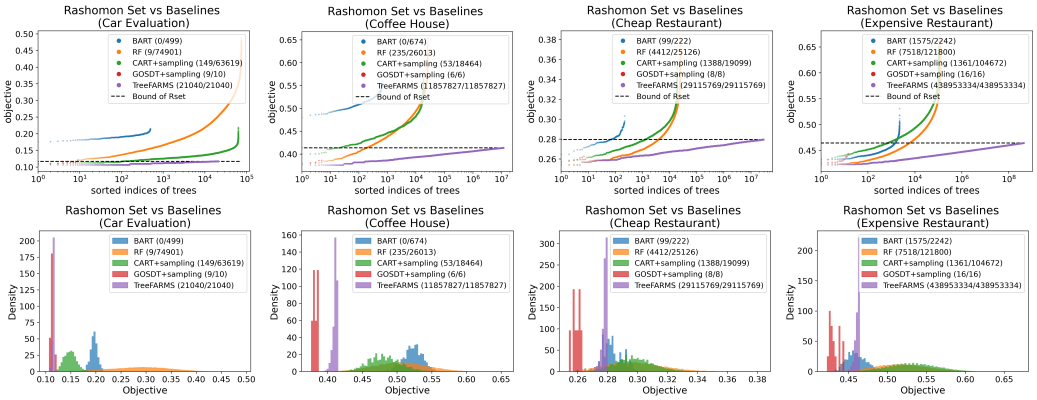


Figure 10: Comparison of trees in the Rashomon set and trees generated by baselines. (A/B) in the legend indicates that A trees of the B trees produced by the baseline method are in the Rashomon set. For example, GOSDT+sampling (9/10) means that 9 trees of 10 distinct trees found by GOSDT with sampling are in the Rashomon set.

G.2 Landscape of the Rashomon set

Collection and Setup: We ran this experiment on **Monk2**. We used dimension reduction techniques to visualize the Rashomon set.

We first generated the Rashomon set with $\lambda = 0.025$ and $\epsilon = 0.2$. We then selected the best tree for each unique set of features in the Rashmon set; that is, if two trees used the same features, we kept the tree with the better objective. Next, we reduced the Rashomon set to $\mathbb{R}^2$ using the PaCMAP dimension reduction technique [which handles global structure better than techniques such as t-SNE and UMAP, see 62].

Results: Figure 11 shows the embedding using three different distance metrics: tree edit distance (a), prediction set distance (b), and feature set distance (c). The *tree edit distance* is the edit distance between two trees using the operations add-node, delete-node, or swap-nodes, all with equal weight. The *prediction set distance* is the Hamming distance between the set of predictions each tree creates for the dataset (up to n). The *feature set distance* between two trees is the Hamming distance between the feature sets they used (up to p).

	TreeFARMS		BART		RF		CART + sampling		GOSDT + sampling	
	time (s)	# trees in Rset/s	norm. time	# trees in Rset/s	norm. time	# trees in Rset/s	norm. time	# trees in Rset/s	norm. time	# trees in Rset/s
Monk2	46.46	2.28×10^6	0.99 ± 0.02	0.03 ± 0.02	1.01 ± 0.00	0.00 ± 0.00	0.99 ± 0.02	0.20 ± 0.11	0.95 ± 0.00	0.31 ± 0.04
COMPAS	3.03	8.75×10^4	0.98 ± 0.03	0.14 ± 0.17	0.99 ± 0.01	68.67 ± 2.34	1.00 ± 0.00	44.83 ± 2.17	0.96 ± 0.01	2.20 ± 0.34
Car Evaluation	207.76	101.27	0.95 ± 0.00	0.00 ± 0.00	1.01 ± 0.01	0.03 ± 0.02	1.00 ± 0.01	0.71 ± 0.04	0.97 ± 0.02	0.03 ± 0.01
Bar	266.79	1.91×10^4	0.96 ± 0.00	0.15 ± 0.29	1.00 ± 0.02	4.78 ± 0.14	1.00 ± 0.01	3.88 ± 0.11	0.96 ± 0.01	0.06 ± 0.01
Coffee House	81.77	1.45×10^5	0.99 ± 0.01	0.00 ± 0.00	1.00 ± 0.00	2.60 ± 0.17	1.00 ± 0.00	0.67 ± 0.03	0.99 ± 0.03	0.06 ± 0.01
Expensive Restaurant	316.40	1.39×10^6	0.96 ± 0.01	3.72 ± 1.31	1.01 ± 0.00	24.44 ± 0.76	1.01 ± 0.01	4.51 ± 0.15	0.96 ± 0.01	0.06 ± 0.01
Cheap Restaurant	74.91	3.89×10^5	0.99 ± 0.00	0.98 ± 0.39	1.00 ± 0.00	59.63 ± 0.43	1.01 ± 0.01	17.75 ± 0.55	0.96 ± 0.01	0.15 ± 0.03
Bar-7	13.11	1.13×10^4	1.01 ± 0.01	0.67 ± 0.83	1.00 ± 0.00	22.78 ± 1.35	1.00 ± 0.00	8.06 ± 0.35	1.03 ± 0.08	0.33 ± 0.07
FICO	3841.94	21.08	0.98 ± 0.03	0.00 ± 0.00	1.02 ± 0.01	3.17 ± 0.03	1.00 ± 0.02	0.05 ± 0.00	0.94 ± 0.00	0.00 ± 0.00
Congressional Voting Records	16.67	0.06	1.00 ± 0.01	0.00 ± 0.00	1.00 ± 0.00	0.06 ± 0.00	1.00 ± 0.00	0.06 ± 0.00	0.96 ± 0.02	0.06 ± 0.00

Table 2: Runtime and number of trees found for TreeFARMS and the four baselines. We show the runtime (in seconds) for TreeFARMS to find the Rashomon set. For the baselines, we show their average runtime and standard deviation normalized to that of TreeFARMS. For instance, a normalized time of 1.1 means more time than TreeFARMS by 10%. We also show the average number of trees in the Rashomon set produced per second and its standard deviation. The Congressional Voting Record dataset has only one tree in the Rashomon set. *The only method guaranteed to find the full Rashomon set is TreeFARMS.*

The images for the Monk2 dataset, shown in Figure 11, yield interesting results. When designing decision trees, we usually expect that the root split contains most of the information, and thus we might expect that much of the Rashomon set shares the same root split. All three embeddings in Figure 11, in which color identifies the feature that splits the root node, show that this is not the case. Interestingly, Figure 11a exhibits clustering, where each cluster includes models with different root splits, which is not what one might expect. Figure 11c suggests the importance of two prominent features that often appear at the root split, *holding_sword* and *is_smiling*, while still highlighting the diversity in the root node.

Querying the Rashomon set is now extremely easy. Here are some examples.

– *Find the 10 best trees*. The circles in Figure 11a identify the ten models with the best objective values. Interestingly, these trees appear in different parts of the space and have diverse structure.

– *Find the five sparsest trees with accuracy above 0.7.* Figure 11b shows the result. These trees are (again) diverse.

– *Find trees with feature head_shape=square but neither head_shape=round nor jacket_color=red.*

The red circles in Figure 11c identify these trees. Constraining models in this way produces trees that use features quite different from all the trees on the left, even when they use the same root.

G.3 Variable Importance: Model Class Reliance

Collection and Set: We ran this experiments on **COMPAS, Bar, Coffee House, Expensive Restaurant**. We find the whole Rashomon set of each dataset given $\lambda = 0.01$ and $\epsilon = 0.05$, and calculate

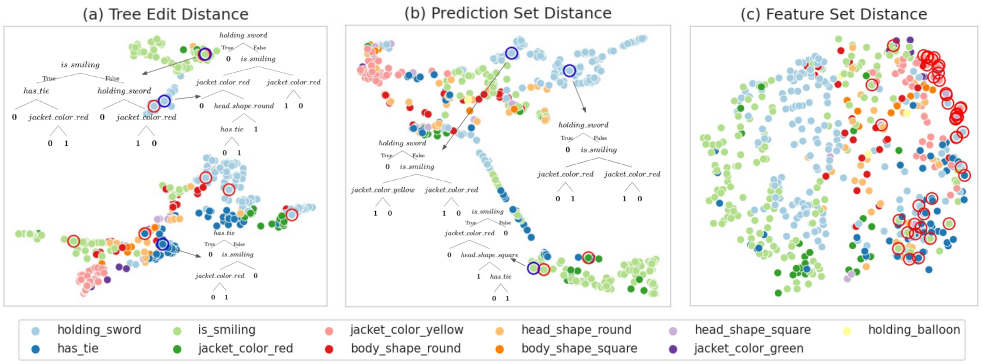


Figure 11: Dimension-reduced images of the Rashomon set for Monk2. Colors represent the feature on which we split at the root. Subfigures (a), (b), and (c) show the embedding using tree edit distance, prediction distance, and feature set distance, respectively. In (a), red and blue circles identify the 10 trees with the lowest objective; the blue circles correspond to the trees on the side. The circles in (b) identify the 5 sparsest trees with accuracy above 0.7. The circles in (c) correspond to trees that use feature *head_shape=square* but not *head_shape=round* and not *jacket_color=red*.

MCR_- and MCR_+ of each feature as described in Appendix D. We then study the how sampling can help us estimate the true model class reliance by sampling 1%, 5%, and 25% trees from the Rashomon set and calculating the MCR based on these subsets of the Rashomon set. Figure 12 shows sampled MCR converges to true MCR. For each sampling proportion, we use 5 different random seeds to compute the average and standard deviation of MCR_- and MCR_+. We ran this particular experiment on a 2.7Ghz (768GB RAM 48 cores) Intel Xeon Gold 6226 processor. We set a 200-GB memory limit.

Results: Figure 12 shows the model class reliance on four different datasets. For the COMPAS dataset (top-left subfigure), features related to prior counts generally have high MCR_+, which means these features are very important for some of trees in the Rashomon set. For the Bar dataset (top-right subfigure), features "Bar_1-3" and "Bar_4-8" have dominant MCR_+ and MCR_- compared with other features, indicating that for all well-performing trees, these features are the most important. Similarly, features "CoffeeHouse_1 3" and "CoffeeHouse_4 8" have dominant MCR_+ and MCR_-. This makes sense, since people who go to bar or coffee house regularly would be likely to accept a coupon for a bar or a coffee house.

Sampling can help us calculate MCR more efficiently by considering only part of the whole Rashomon set. Figure 12 shows that sampled MCR, in general, converges to true MCR even if only 1% or 5% of trees are sampled.

G.4 Balanced Accuracy and F1-score Rashomon set from Accuracy Rashomon set

Collection and Set: We ran this experiments on three imbalanced datasets **Breast Cancer, Cheap Restaurant, Takeaway Food**. We ran this particular experiment on a 2.7Ghz (768GB RAM 48 cores) Intel Xeon Gold 6226 processor. We set a 200-GB memory limit.

Results: Figure 13, 14 show trees in the Accuracy Rashomon set which covers the Balanced Accuracy Rashomon set and F1-score Rashomon set respectively. The black dashed line indicates the corresponding objective thresholds and blue dots below the dashed line are trees within these Rashomon sets. In some cases, the tree with the minimum misclassification objective is also the tree with the minimum other metric objective (see Figure 13 left and Figure 14 right). But this is not always guaranteed. For example, in the right subfigure of Figure 13, a single split node has the optimal accuracy objective (in yellow), while another three-leaf tree minimizes the balanced accuracy objective (in green). Actually, many trees have better balanced accuracy objective than the tree that minimizes the accuracy objective. A similar pattern holds for the F1-score Rashomon set (see left subfigure Figure 14).

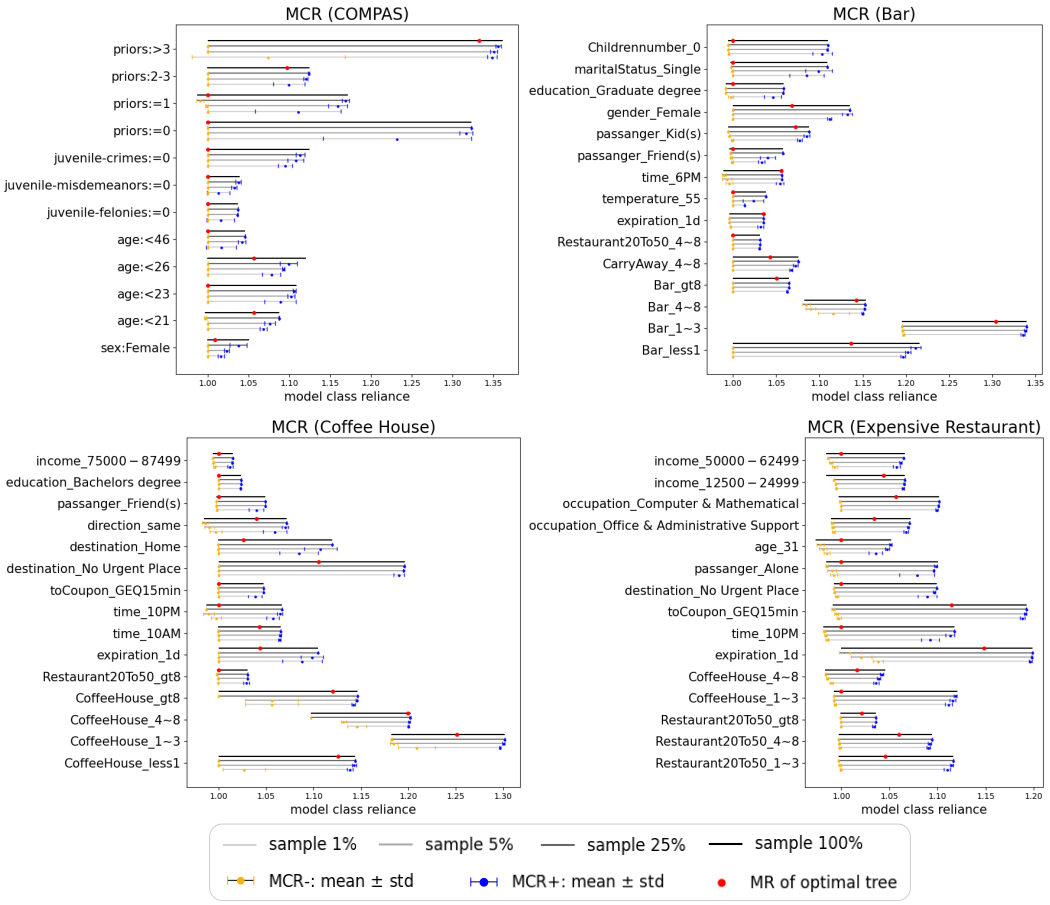


Figure 12: Variable Importance: Model class reliance on the COMPAS, Bar, Coffee House, and Expensive Restaurant ($\lambda = 0.01$, $\epsilon = 0.05$). Four different line segments colored in different gray levels are model class reliance calculated by sampling different proportions from the whole Rashomon set. Yellow/blue dots with bars indicate $\text{MCR}_-/\text{MCR}_+$ within one standard deviation of the mean. "Sample 100%" means using the whole Rashomon set and there is no variation. Red dots indicate the model reliance (variable importance) calculated by the optimal tree.

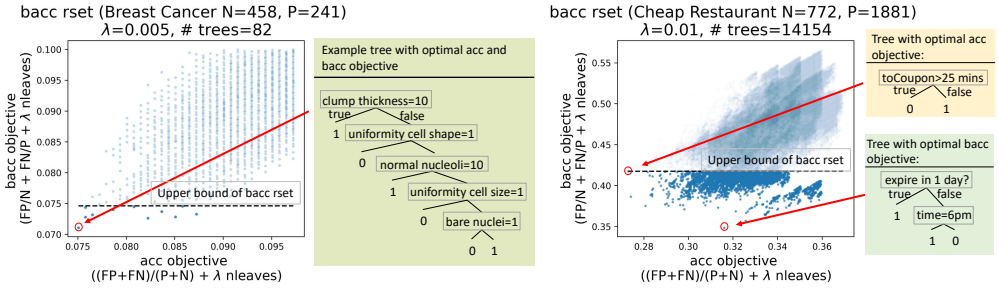


Figure 13: Example of Balanced Accuracy Rashomon sets. # trees indicates the number of trees within the Balanced Accuracy Rashomon set. Trees in the yellow region have optimal accuracy objective and trees in the green region have optimal balanced accuracy. (Breast Cancer: $\lambda = 0.005$, Cheap Restaurant: $\lambda = 0.01$)

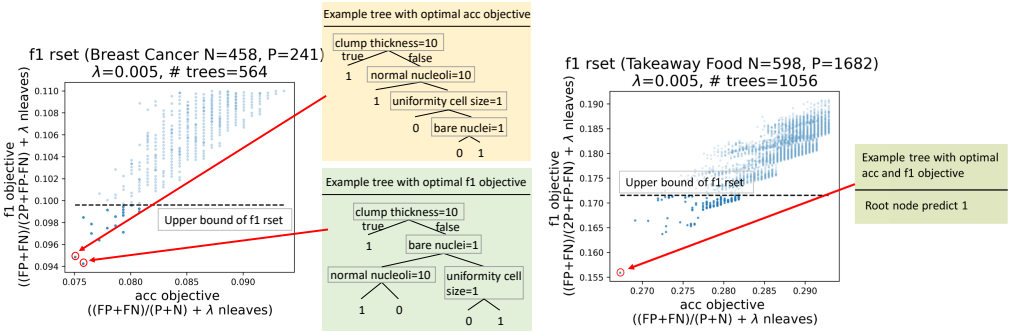
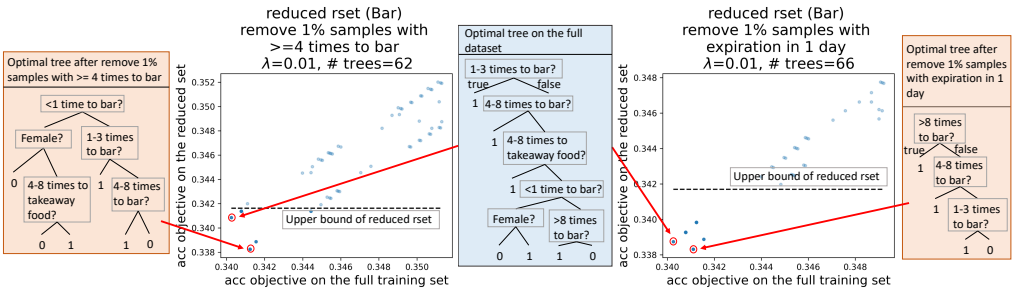


Figure 14: Example of F1-Score Rashomon set. # trees indicates the number of trees within the F1-score Rashomon set. Trees in the yellow region have optimal accuracy objective and trees in the green region have optimal F1-score objective.(Breast Cancer: $\lambda = 0.005$, Takeaway Food: $\lambda = 0.005$)

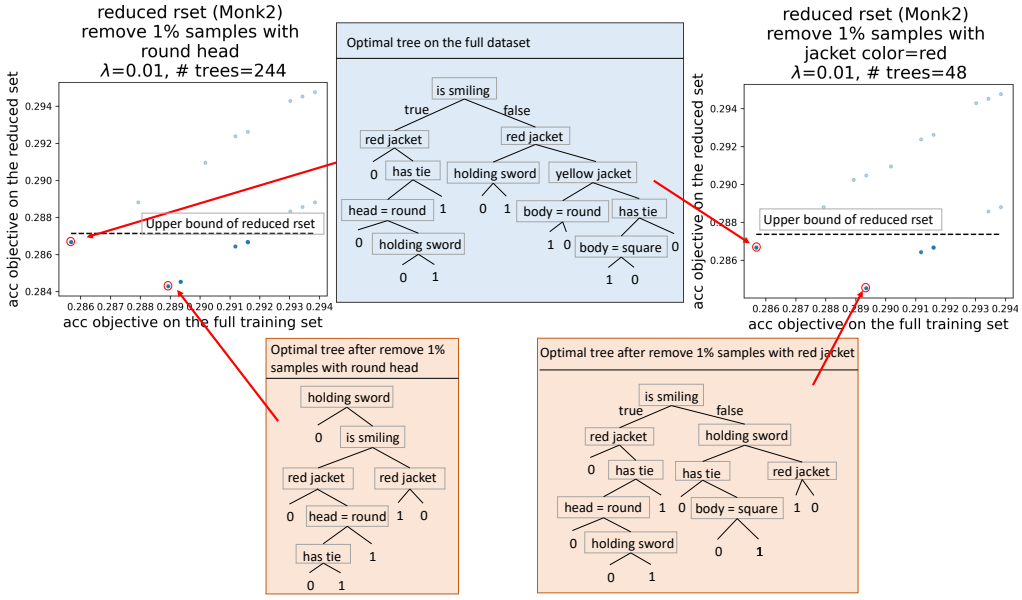
G.5 Rashomon set after removing a group of samples

Collection and Set: We ran this experiments on **Monk2, COMPAS, Bar, Expensive Restaurant**. We ran this particular experiment on a 2.7Ghz (768GB RAM 48 cores) Intel Xeon Gold 6226 processor. We set a 200-GB memory limit.

Results: Figure 15 show accuracy objective on the full dataset versus objective on the reduced dataset after 1% of different samples are removed on the Bar and Monk2 datasets. The black dashed line indicates the objective threshold of the reduced Rashomon set and blue dots below the dashed line are trees within the reduced Rashomon set. Similar to Figure 5 all scatter plots show a high correlation between the accuracy objective on the full dataset and the reduced dataset, indicating sparse near-optimal trees are robust to the shift in sample distribution. Optimal trees on the reduced dataset might be different, as we see by comparing the trees in the orange region and blue region.



(a) Example Rashomon sets and optimal trees after we remove the 1% of samples with "number of times to bar $\geq$ 4" (left) and "expiration of coupon in 1 day" (right) on the Bar dataset.



(b) Example Rashomon sets and optimal trees after we remove the 1% of samples with "round head" (left) and "red jacket" (right) on the Monk2 dataset.

Figure 15: Example Rashomon sets and optimal trees after 1% of samples are removed. The optimal tree on the full dataset is shown in the blue region and optimal trees on the corresponding reduced datasets are in the orange region.

Appendix References

[51] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[52] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[53] Kristen B Gorman, Tony D Williams, and William R Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PloS one*, 9(3):e90081, 2014.

[54] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. How we analyzed the COMPAS recidivism algorithm. *ProPublica*, 2016.

[55] FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. Explainable Machine Learning Challenge. `https://community.fico.com/s/explainable-machine-learning-challenge`, 2018.

[56] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[57] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[58] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[59] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[60] Rodney Sparapani, Charles Spanbauer, and Robert McCulloch. Nonparametric machine learning and efficient computation with Bayesian additive regression trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66, 2021.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization. *J Mach. Learn. Res*, 22:1–73, 2021.