Exploring the Whole Rashomon Set of Sparse Decision Trees

Anonymous Author(s)
Affiliation
Address
email

Abstract

In any given machine learning problem, there may be many models that could 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 within a loss function. The *Rashomon set* is the set of these all almost-optimal models. Rashomon sets can be extremely complicated, 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: 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.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

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 enumerations 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 computed 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 the 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 and $y_i \in \{0,1\}$ are labels. 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 denote $\theta_\epsilon := (1+\epsilon) \times Obj(t_{\text{ref}}, \mathbf{x}, \mathbf{y})$ as the threshold of 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 will have 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 depth of 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 $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, H_t)$, 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.*

We can tighten the basic Rashomon lower bound by using knowledge of *equivalent points*. 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 this 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 a "lookahead" bound, used in GOSDT [31].

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. 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 if $\mathbf{x}_i$ satisfies the Boolean assertion a. 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. **Lines 2-4**: Using the best objective from Line 1, compute θ_ϵ, as defined in Definition 1. Configure and execute (the modified) GOSDT to produce a dependency graph containing all subproblems in the Rashomon set. **Lines 5-8**: 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 7-8**: 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 [39] 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.
1: $opt \leftarrow$ *gosdt*$(\mathbf{x}, \mathbf{y}, \lambda)$ *// Use GOSDT to find opt, the objective of the optimal tree.*
2: $\theta_\epsilon \leftarrow opt * (1 + \epsilon)$ *// Compute θ_ϵ, which is the threshold of the Rashomon Set.*
3: *gosdt.configure_bounds*(θ_ϵ) *// Set GOSDT's lookahead bound to θ_ϵ, and disable the leaf support and incremental accuracy bounds that would remove trees from the Rashomon set.*
4: $G \leftarrow$ *gosdt.get_graph*$(\mathbf{x}, \mathbf{y}, \lambda, \theta_\epsilon)$ *// Return dependency graph using the Rashomon bound.*
5: $MS \leftarrow []$ *// Initialize Rashomon Model Set MS*
6: $P \leftarrow ones(|\mathbf{y}|)$ *// Entire dataset*
7: extract(G, P, θ_ϵ) *// Fill in MS with trees in the Rashomon set*
8: **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 MS with the Rashomon set for sub within scope.
// Check if we have already solved the subproblem sub. SOLVED is presented in Alg. 3
1: **if** SOLVED$(MS, sub, scope)$ **then return**
2: **if** $G[sub] \leq scope$ **then** *// Check if we should create a leaf. (Theorem 3.2 and 3.1)*
3: $MS \leftarrow MS \cup$ *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
4: **for** each feature $j \in [1, M]$ **do**
5: $sub_l, sub_r \leftarrow split(sub, j)$
6: **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
7: *extract*$(G, sub_l, scope - G[sub_r])$
8: *extract*$(G, sub_r, scope - G[sub_l])$
9: *left*, *right* $\leftarrow MS_{sub_l}, MS_{sub_r}$
10: **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)
11: **if** $obj(m_l) + obj(m_r) > scope$ **then continue**
12: $MS \leftarrow MS \cup add(sub, obj(m_l) + obj(m_r), m_l, m_r)$ *// Add pair to Model Set*
// MS now contains all MSI for sub with objective less than or equal to scope.

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 only managed to calculate MCR_- for convex loss in linear models [2]. For decision trees, the problem is nonconvex and intractable using previous methods. However, since our method 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. $q^+ + q^- = 1$. 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. The next two theorems guide us to find the Balanced Accuracy or F1-Score Rashomon set from the Accuracy Rashomon set. Proofs to them are presented in Appendix E. 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*

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

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. 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 Theorem 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* [40].) 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 $\{\mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}\}$ 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 $\epsilon \geq \frac{2|K|}{n \times Obj(t^*, \mathbf{x}, \mathbf{y})}$, 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 $\forall t \in R_{set}(\epsilon', \tilde{t}^*, \mathcal{T}, \mathbf{x}_{[\setminus K,\cdot]}, \mathbf{y}_{[\setminus K]}), t \in R_{set}(\epsilon, t^*, \mathcal{T}, \mathbf{x}, \mathbf{y})$.*

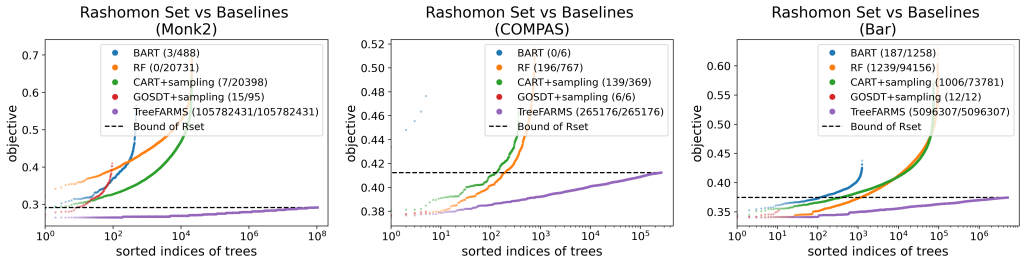


Figure 1: Comparison of trees in the Rashomon set ($\lambda = 0.01, \epsilon = 0.1$) and trees generated by baselines. Trees in the Rashomon set have objective below the dashed line. (A/B) in legend represents that A trees among B trees trained by the baseline are in the Rashomon set. For example, RF (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.

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. How does the size of the Rashomon set change with respect to λ, ϵ, and the dataset? (§G.3), 4. What does the Rashomon set look like? What can we learn about its structure? (§G.4), 5. What does MCR look like for real datasets? (§6.2), 6. How do balanced accuracy and F1-score Rashomon sets compare to the accuracy Rashomon set? (§6.3), and 7. 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 41], a penguin dataset [42], a criminal recidivism dataset [COMPAS, shared by 39], the Fair Isaac (FICO) credit risk dataset [43] used for the Explainable ML Challenge, and four coupon datasets (Bar, Coffee House, Cheap Restaurant, and Expensive Restaurant) [44] that come from surveys. More details are in Appendix F.

6.1 Performance and Timing Experiments

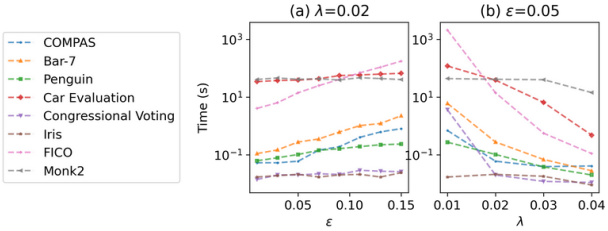
Our method directly constructs the Rashomon set of decision trees of a dataset. While, to the best of our knowledge, there is no previous directly comparable work, there are several methods one might naturally consider to find this set. One might use methods that sample from the high-posterior region 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 [45, 46]. 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.

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 of 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 Rset, because we do not need to store the samples. Since MCR requires computing extreme value statistics (max and min over the Rset), 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 is usually sufficient. More results are shown in Appendix G.5.

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 within 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

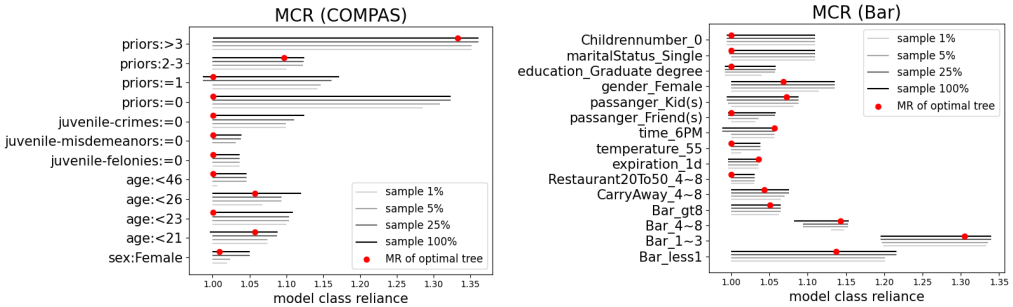


Figure 3: Variable Importance: Model class reliance on the COMPAS and Bar ($\lambda = 0.01, \epsilon = 0.05$). Red dots indicate the model reliance (variable importance) calculated by the optimal tree. Each line connects MCR_- and MCR_+ showing the range of variable importance among all good models.

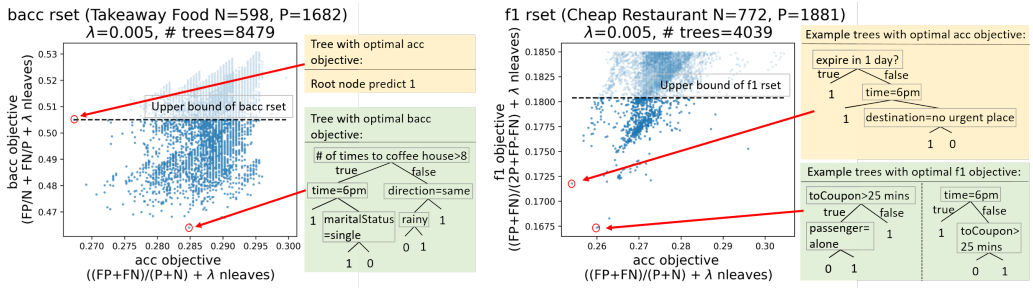


Figure 4: Example of Balanced Accuracy Rashomon set (left) and F1-Score Rashomon set (right). # trees indicates the number of trees within the Balanced Accuracy or F1-score Rashomon set. Trees in the yellow region have optimal accuracy objective and trees in the green region have optimal balanced accuracy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

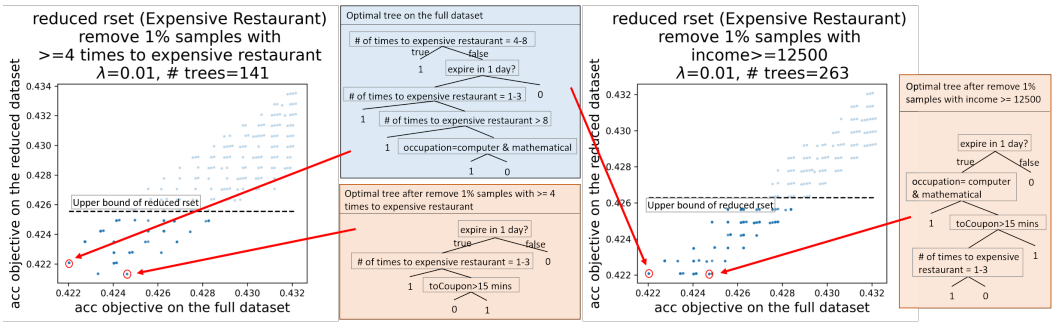


Figure 5: Example Rashomon sets and optimal trees after we remove the 1% of samples with "number of times to expensive restaurant $\geq$ 4" (left) and "income $\geq$ \$12,500" (right) on the Expensive Restaurant dataset. The optimal tree on the full dataset is shown in the gray region and optimal trees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

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.6.

6.4 Rashomon set after removing a group of samples

Figure 5 shows accuracy objective on the full dataset versus objective on the reduced dataset after 1% of samples with "number of times to expensive restaurant $\geq$ 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 within 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.7.

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.

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

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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]
   (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 license

   (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]